Stellingen
behorende bij het proefschrift
Non-Cellular Wireless Communication Systems
door
Jack P.F. Glas

1. Een vaak vergeten voordeel van multi-disciplinair werken is dat interessante doorbraken mogelijk zijn omdat er vanuit verschillende vakgebieden naar een probleem gekeken wordt.

2. Naarmate inspraak een voornamere rol speelt, wordt motiverend leiderschap belangrijker.

3. Om tot een efficiënt ontwerp te komen, moet hardware/software partitioning tijdens het ontwerptraject gelijktijdig worden uitgevoerd met het configureren van de processor-core (hoofdstuk 2 van dit proefschrift).

4. Niet de aanval, maar zelfkritiek is vaak een goede manier om een doel te bereiken.

5. De uitdaging van het komen tot een systeem-specificatie ligt in het vinden van een goed compromis tussen alle technische oplossingen op basis van een goede inschatting van de wensen van de beoogde gebruiker (hoofdstuk 3 van dit proefschrift).

6. De afwezigheid van een rigoureuze analyse van een algoritme is geen reden om terug te vallen op een minder efficiënt maar goed te analyseren oplossing (hoofdstuk 5 van dit proefschrift).

7. Juist niet-cellulaire communicatiesystemen profiteren van de "interference-limited" eigenschap van CDMA technieken als "Direct-Sequence" of "Frequency-Hopping" (hoofdstuk 1 van dit proefschrift).

8. Tijdsdruk dwingt mensen zich te beperken tot belangrijke en relevante zaken maar kan ook leiden tot een onzorgvuldige afhandeling. Net zoals bij andere trade-offs dient daarom ook hier een goed compromis gevonden te worden.

9. Om tot een efficiënt ontwerp van mobiele apparatuur te komen moet niet alleen de barrière tussen hardware en software worden geslecht (hoofdstuk 2 van dit proefschrift); Het afbreken van de traditionele grens tussen het analoge front-end en het digitale back-end is minstens even belangrijk.

10. Mensen zijn geneigd zaken waarvan men weinig kennis heeft als eenvoudig af te doen.

11. Het terugvallen op formele regels getuigt van zwakte.

12. Het World Wide Web biedt studenten en promovendi een uitstekende mogelijkheid goedkoop en buitengewoon effectief bekendheid te verwerven.

13. Problemen in een ontwerptraject blijken pas bij het doorlopen van dit traject voor een bepaalde toepassing. Het is daarom van belang een dergelijk traject uit te leggen aan de hand van een voorbeeld; dit vergroot tevens het inzicht.

14. Alle automatisering ten spijt ligt de uitdaging van het implementeren nog steeds in het vinden van doorbraken die eenvoudige oplossingen combineren met goede prestaties (hoofdstuk 5 van dit proefschrift).

Non-Cellular Wireless Communication Systems

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College van Dekanen aangewezen,
op donderdag 5 december 1996 te 10.30 uur

door

Jacobus Petrus Franciscus GLAS

elektrotechnisch ingenieur
geboren te Haarlemmermeer

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. R.H.J.M. Otten

Samenstelling promotiecommissie:

Rector Magnificus, voorzitter	
Prof. dr. ir. R.H.J.M. Otten, promotor	Technische Universiteit Delft
Prof. dr. ir. W.M.G. van Bokhoven	Technische Universiteit Eindhoven
Prof. dr. ir. G. Brussaard	Technische Universiteit Eindhoven
Dr. H. Corporaal	Technische Universiteit Delft
Prof. dr. ir. P.M. Dewilde	Technische Universiteit Delft
Dr. ir. J.P.M.G. Linnartz	Philips Research, Eindhoven
Prof. ir. K.H.J. Robers	Technische Universiteit Delft
Prof. dr. ir. A.H.M. van Roermund	Technische Universiteit Delft

Ir. L.K. Regenbogen heeft als begeleider in belangrijke mate aan het totstandkomen van het proefschrift bijgedragen.

An electronic version of this book is available on the Internet. From within your WWW-browser open the location `http://cas.et.tudelft.nl`, then go to "people" and follow the link labeled with the author's name.

Correspondence about this thesis and related subjects can be directed via email to: `glas@cas.et.tudelft.nl`.

ISBN: 90-5326-024-2

Contents

Summary VII

Acknowledgements IX

1 Introduction 1
- 1.1 Multi-Access . . . 2
- 1.2 CDMA techniques . . . 4
- 1.3 Claims of the thesis . . . 8
- 1.4 Overview . . . 10

2 An Embedded Spread Spectrum Processor 13
- 2.1 Introduction . . . 13
- 2.2 Why an Embedded Realization? . . . 14
- 2.3 The design process . . . 16
- 2.4 Selecting a Processor-Framework . . . 18
- 2.5 Requirements for the Hardware/Software partitioning stage . . . 20
- 2.6 Available resources . . . 21
- 2.7 Conclusions . . . 22

3 Hybrid DS/FH Spread Spectrum Communication System 23
- 3.1 Introduction . . . 23
- 3.2 System Specification . . . 24
- 3.3 Clock control . . . 35
- 3.4 Fixing the system parameters for WISSCE . . . 39
- 3.5 Conclusion . . . 41

4 Performance analysis 43
- 4.1 Introduction . . . 43
- 4.2 Degradation of the data detection SNR . . . 44
- 4.3 Relation between SNR-in and the BER-performance . . . 50
- 4.4 Code selection . . . 57
- 4.5 Conclusion . . . 66

5 Implementation alternatives **67**
5.1 Introduction 67
5.2 Data detection 69
5.3 Synchronization 84
5.4 Front-end considerations 103
5.5 Conclusions 105

6 Hardware/Software partitioning **107**
6.1 Introduction 107
6.2 System description 109
6.3 Partitioning process 116
6.4 Conclusions 119

7 WISSCE on the MOVE **121**
7.1 Introduction 121
7.2 Hardware design 122
7.3 Firmware design 128
7.4 Co-Simulation 131
7.5 Conclusions 139

8 Conclusions **141**

A Pseudo-Random Noise Sequences **145**
A.1 Selected Code-set 145
A.2 Implementing the relative delays 147

Bibliography **151**

Glossary **159**

Index **165**

Samenvatting **169**

About the author **171**

List of Figures

1.1 Wireless, point-to-point communication concept 1
1.2 Direct-Sequence spreading 6
1.3 DS-concept, before and after despreading 6
1.4 Near-Far effect illustrated 7
1.5 Illustration of the frequency hopping concept 7

2.1 Embedded system design process 17
2.2 Structure of a transport triggered architecture 19
2.3 3-D view on a Sea-of-Gates circuit 21

3.1 Receive and transmit frequency bands 28
3.2 Trade-off between processing gain and symbol-rate 29
3.3 Trade-off between data speed and number of modulation levels . 29
3.4 A possible user-address 30
3.5 Trade-off between $N_{\mathrm{FH}}$and N_{DS} 31
3.6 Low-frequency part of the front-end architecture 33
3.7 Schematic view of possible transceiver architecture 35
3.8 Principle of a direct digital synthesizer (DDS) 37
3.9 Proposed clocking scheme 41

4.1 Effects of input-filtering illustrated 45
4.2 Partial interference in DS/FH spreading scheme 47
4.3 MA-interference illustrated 48
4.4 BER verses input-SNR 53
4.5 BER in a fading-environment 56
4.6 Frequency-domain comparison of a Walsh and an M-sequence . . 59
4.7 Kasami-code generator scheme 63
4.8 $E\{p_{\mathrm{hit}}(k,K)\}$ as a function of K active users 65

5.1 Implementation trade-offs 68
5.2 Evaluation of $tw_{\cos}[1,k]$ 74
5.3 Evaluation of $tw_{sin}[1,k]^2 + tw_{cos}[1,k]^2$. 76
5.4 Phase dependence for MFSK-symbols "1" and "8" 77

5.5 Signal suppression-factors: calculated and simulated 78
5.6 Limiter SNR-gain as a function of the input-SNR 78
5.7 Sensitivity as a function of the input SNR 79
5.8 Phase dependence as function of the input SNR 80
5.9 Data-detection scheme 81
5.10 BER as a function of the input-SNR 82
5.11 Output power of data-detector as a function of the frequency-error 85
5.12 Code-acquisition algorithm 89
5.13 LTR-control circuit 92
5.14 Acquisition trajectories for different noise situations 93
5.15 Acquisition trajectory, 2 strong interferers present 94
5.16 Acquisition trajectory with only one strong interferer 94
5.17 Typical code-tracking scheme 95
5.18 Typical code-tracking curve 96
5.19 MCTL-architecture 96
5.20 Adjusted MCTL-scheme 97
5.21 Tracking curve of the WISSCE-tracking algorithm 98
5.22 Code-tracking filter 99
5.23 Controlling the local time reference (LTR) 99
5.24 Tracking-loop 101
5.25 Calculated and simulated tracking curves 101
5.26 Simulation of tracking-process 102
5.27 Schematic view of possible transceiver architecture 105

6.1 SIR-representation of WISSCE's receiving process 110
6.2 Parallelism in the WISSCE-receiver 111
6.3 HW/SW-partitioning result 118

7.1 Structure of a transport triggered architecture with four busses . . 122
7.2 Interfacing with the DFT-CE-FU 125
7.3 Interfacing with the PN-CODE GENERATOR-FU 126
7.4 Interfacing with the DDS-FU 127
7.5 Interfacing with the Squaring-FU 127
7.6 Interfacing with the data-in FU 128
7.7 WISSCE-MOVE processor configuration 129
7.8 Firmware operation during normal operation 130
7.9 PTOLEMYrepresentation of the integer-unit as a *galaxy* 133
7.10 WISSCE-model in PTOLEMY 135
7.11 WISSCE-MOVE processor in PTOLEMY 137
7.12 Co-simulation in process 138

SUMMARY

People always wanted to communicate and their demands grew with the possibilities offered by technology. Nowadays, modern technology enables mobile communications in many situations. An important component in mobile communications are non-cellular wireless communication systems for short distances.

This thesis will show how modern resources and design methods can be applied in the design of such systems. To show full advantage of the embedded design methodology used to narrow down the traditional gap between hardware and software, the emphasis was on the digital baseband processing. Important consequences of the system specification on the front-end were however mentioned as well.

Realizing a communication system involves a large number of decisions. Problematic, at least for presenting the material, is that all these decisions are in some way related and it was therefore, at least for presenting the material, not possible to first enlist all design trade-offs and then at the end fix all parameters to obtain an optimal realization. Parameters have to be fixed at the appropriate design stage to keep the design space manageable.

The selection of a multiple-access scheme influences all further design stages and has to be specified early at an early stage. On one hand it was concluded that Code Division Multiple Access (CDMA) techniques nicely fit the ad-hoc nature of communication links in non-cellular systems. On the other hand all CDMA techniques have there own disadvantages. We show that by combining two of the more common CDMA techniques, the high processing gain of direct-sequence is retained while near-far effects are effectively reduced due to frequency-hopping.

Not only the multiple-access scheme is important. This technique should be properly embedded in a complete system specification. During the system definition numerous trade-offs situations became visible. Market potential formulated as user demands and their relative importance were used as a guidance to arrive at sensible compromises. We illustrate this process by using a real world example. In this way it becomes clear where in the design problems appear. This example is named WISSCE: Wireless Indoor Spread Spectrum Communication Equipment.

Another important consideration early in the system specification are the available resources to implement the system. In many time-critical systems like communication systems, software implementations have advantages because they are inexpensive, flexible and fit the nature of the algorithms that have to be implemented. However, the existing timing constraints often preclude such an implementation. A practical solution is then to move critical functionality from software to hardware yielding an embedded system. To obtain the highest efficiency, the hardware and software parts have to cooperate seamlessly. This implies a processor framework configurable in both its general purpose capability and dedicated functionality. An excellent option in this sense is the transport triggered architecture.

Once specified both the (communication) system and its implementation concept, other implementation issues come into play. Standard solutions do not always match with the available resources. "Intelligent algorithms" have to be found that combine simple implementations with acceptable losses. For WISSCE's baseband processing this process was illustrated by proposing algorithms for data-detection and synchronization. Simulation runs justified the application of these algorithms.

Simulations are still mandatory to check the operation of any system but they can also serve another goal. The "program" used for simulation, for instance in the programming language "C", can be used as a specification of the system. This specification together with timing and area-constraints and cost-data on different implementation alternatives can be used as an input to a hardware/software partitioning stage. To obtain many profiles of possible hardware/software partitionings, an automatic tool is required. We show how the partitioning tool HSPART can be applied to partition WISSCE's baseband processing into hardware and software components.

The final part of the design is to map the hardware and software parts on the transport triggered architecture. Five application specific functional units were required to implement the hardware functionality of WISSCE. After specifying both the general purpose part and the application specific part of the embedded system, another simulation run has to be performed. This simulation was performed at a lower level than the system simulation performed earlier as the cooperation of hardware and software had to be verified. The simulation framework PTOLEMY, has been extended to allow for co-simulation. After satisfactory cosimulation results, the mapping to real hardware can be performed.

We conclude that an embedded system design methodology is suitable for implementing systems such as the baseband processing of a transceiver in a non-cellular communication system for short distances. The available resources are adequate while a transport triggered architecture provides enough flexibility.

keywords: CDMA, spread spectrum, direct sequence, frequency hopping, communication system design, code-synchronization, code-acquisition, code-tracking, PN-CODES, hardware/software partitioning, embedded system design, co-design, sea-of-gates, OCEAN, HSPART, transport triggered architectures, move-processor.

ACKNOWLEDGEMENTS

The work presented in this thesis was carried out in the Circuits and Systems group at Delft University of Technology. The main focus of the work was to realize a mobile communication system able to provide ad-hoc communication links.

That system was given the name WISSCE, an acronym standing for Wireless Indoor Spread Spectrum Communication Equipment. The large number of interested students as well as the many visits of the WISSCE home page [WIS] and our spread spectrum pages [Gla] on the internet were indications of a great interest in this project from both within and outside the university.

After a system-definition stage, the project focus was directed towards implementation aspects. It appeared that the WISSCE baseband processing could be efficiently implemented as an embedded system. During the hardware/software partitioning stage WISSCE functioned in this sense as a "real-world" example for the partitioning tool HSPART. For prototyping the in-house developed OCEAN tool-set was used.

The many people involved in the WISSCE-project as well as other members of the Circuits and Systems group, especially Marion de Vlieger and Viorica Simion, were very motivating factors in the continuation of this work.

At this place I like to thank a number of people that were of great help in the WISSCE-project. First of all, Ralph Otten for coaching the project, furthermore he proposed an embedded system design approach and critically reviewed this thesis. Loek Regenbogen contributed many useful ideas during the design process. Richard den Dulk initiated my interest in spread spectrum communications. Irek Karkowski wrote the partitioning tool HSPART. Martin Stehouwer, Wim Zwart and Cenk Tekin were involved in designing the first prototype-circuits. Jan Nieuwstad worked on the co-simulation tool and was of great help in designing the cover.

Finally, I would like to thank my family and friends for supporting me and showing me the importance of other activities beside work.

Jack Glas
Delft, November 1996

Chapter 1

INTRODUCTION

People always wanted to communicate and their demands grew with the possibilities offered by technology. Nowadays, modern technology enables mobile communications in many situations. An important component in mobile communications are non-cellular wireless communication systems for short distances.

Wireless communication systems can be roughly divided into two categories: cellular and non-cellular systems. In cellular systems the area to be covered is divided in a number of cells. All communication in a cell goes via a single base-station located in that cell. Hand-over protocols and connections between base-stations enable roaming over cell borders. The consequence of this concept is that both an infrastructure and a complex protocol are required. Non-cellular systems form another category of wireless communication systems for which no infrastructure is required. In this sense the complete system itself is mobile.

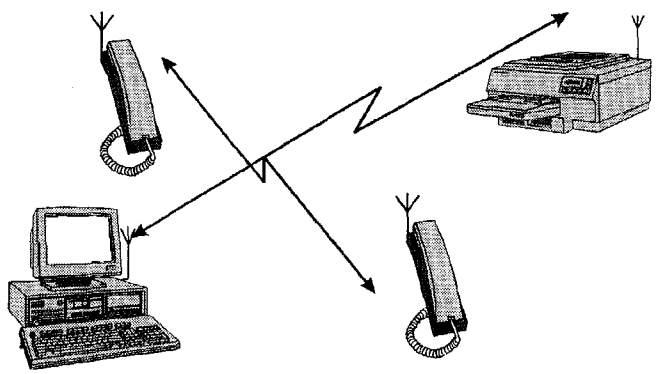


Figure 1.1. Wireless, point-to-point communication concept

This thesis deals with systems providing short distance, multi-access, ad-hoc based, point to point communication links. Shortly, we focus on "non-cellular wireless communication systems". Applications can primarily be found as indoor data communication systems as illustrated in figure 1.1. As this figure only shows a

number of properties of the target communication system, the main properties are listed below:

- **Non-cellular** systems do not require an infrastructure. Our target system is a non-cellular communication system for short distances.
- **Multi-access** enables several simultaneous communication links. In the figure two such links are shown.
- **Random-access**, this term is in this context defined as the ability of users to initiate a communication link at any arbitrary moment.
- **Digital data-communication links** Although not completely clear from the picture, the target system is a digital communication system. Applications can be both data-links and (via a speech-coder) speech-links.

This thesis addresses the different aspects of the realization trajectory of such communication systems. Explaining a design methodology by using a concrete design example has the advantage of exactly showing where "difficult" points can be found. For this reason, the design trajectory of a communication system referred to as WISSCE is used to illustrate the proposed design methodology.

Concerning such a communication system, a number of sales-points exist. For instance the net data-speed should be high enough to enable reasonable data-transmission speeds. A user would like to have this speed as high as possible, there are however technical limitations. In our system we will therefore use as a constraint a minimum speed of 64 kbit/s which is equal to a single ISDN channel. Another issue is reliability: In most situations a user should be able to initiate a communication link. During communication, the bit error rate (BER) has to be acceptable. Other points would be that a user can initiate a transmission link at any arbitrary moment. Also a user does not like long synchronization times or the requirement to install an expensive infrastructure. The possibility to take the system itself and use it at another place would be beneficial. For transmission speed reasons it is is also important to provide duplex connections. In this way transmission and reception at the same time becomes possible. The user's wishes are addressed in more detail in chapter 3 where they will be used as a basis to find a system specification.

1.1 Multi-Access

An important issue in wireless communication systems is multiple random access: communication links can be activated at any moment while several links can be active simultaneously. As multi-access and random-access are properties mainly

determined by the chosen data-communication technique it is important to keep these requirements in mind from the very beginning. Three possible concepts to realize a multi-access communication system are in use:

1. *FDMA*
   Frequency Division Multiple Access, commonly used in conventional telephone systems: every user gets a certain frequency band assigned and can use this part of the spectrum to perform its communication. If only a small number of users is active, not the whole resource (frequency-spectrum) is used. Assignment of the channels can be done centrally or by carrier-sensing in a mobile. The latter possibility enables random-access.

2. *TDMA*
   Time Division Multiple Access, applied nowadays in mobile phone systems: every user is assigned a (set of) time-slots. Transmission of data is only possible during this time-slot, after that the transmitter has to wait until it gets another time-slot. Synchronization of all users is an important issue in this concept. Consequently, there must be a central unit (base-station) that controls the synchronization and the assignment of time-slots. This means that this technique is difficult to apply in random-access systems.

3. *CDMA*
   Code Division Multiple Access (Spread Spectrum). A unique code is assigned to each user. This code is used to "code" the data message. As codes are selected for low cross-correlation properties, all users can transmit simultaneously in the same frequency channel while a receiver is still capable of recovering the desired signal. Synchronization between links is not strictly required and so random-access is possible. A practical application at the moment is the cellular-CDMA phone system IS-95 [Qua92].

Combinations are also possible, the popular European cellular phone systems DECT and GSM for instance use a combination of TDMA and FDMA. There a single transmission-cell is defined by a combination of a frequency channel and a time-slot.

From the above list it is clear that both FDMA and CDMA are candidate transmission techniques to enable multiple random access. There are however a number of reasons for choosing CDMA over FDMA. The first alternative provides [Sch94, SOSL85a, Dix84]:

- **Interference limited operation.** In all situations the whole frequency-spectrum is used. As a result the more active users are present, the higher the interference level will be.

- **Privacy due to unknown codes.** The applied codes are - in principle - unknown to a hostile user. This means that it is hardly possible to detect the message of another user.
- **Applying spread spectrum implies the reduction of multi-path effects.** By using a wide frequency-band, the influence of narrow-band fades is reduced.
- **Random access possibilities.** Users can start their transmission at any arbitrary time (no infrastructure required).
- **Good anti-jamming performance.** Small-band interference is reduced as explained in the next section.

These were the reasons for selecting CDMA as multi-access technique in the non-cellular target communication system. As this choice has a large impact on further design stages, the next section provides an introduction to CDMA-techniques.

1.2 CDMA techniques

Code Division Multiple Access (CDMA) is used in spread spectrum systems to enable multiple-access. It is a transmission technique in which the frequency spectrum of a data-signal is spread using a code uncorrelated with that signal and unique to every addressee. As the applied codes are selected for their low cross-correlation values, it is possible to make a distinction between the different signals. An initiator knows the code of the intended addressee and is so capable of activating the desired communication link.

The first applications were in the military field because of the difficulty to jam or detect spread spectrum signals. Nowadays however spread spectrum systems are gaining popularity also in commercial applications (for instance: IS-95 [Qua92]).

If a signal is combined with a code the bandwidth of the original signal increases. The spectrum is "spread" which justifies the name "spread spectrum". At the same time the spectral power density decreases as the total transmitted power stays equal. The ratio of transmission and information bandwidth is therefore an important parameter in spread spectrum systems. This ratio is referred to as "processing gain":

$$G_p = \frac{BW_t}{BW_i} \tag{1.1}$$

which is the "spreading factor". The processing gain also determines the number of users that can be allowed in a system, the amount of multi-path effect reduction,

the difficulty to jam or detect a signal etc. For spread spectrum systems it is advantageous to have a processing gain as high as possible.

Different spread spectrum techniques exist: Direct-Sequence (DS), Frequency-Hopping (FH), Time-Hopping (TH) and Multi-Carrier CDMA (MC-CDMA). It is also possible to make use of combinations. Overviews of the various spread spectrum techniques can be found in [SOSL85a, Hol82, Dix84, PSM82, SH85, YLF94].

We will now concentrate on the two more popular techniques: direct-sequence and frequency-hopping.

1.2.1 Direct Sequence

Direct Sequence is the most popular Spread Spectrum Technique. The data signal is multiplied with a pseudo random bit sequence, often referred to as pseudo random noise code (PN-CODE).

A PN-CODE is a sequence existing of chips (see figure 1.2) valued -1 and 1 (polar) or 0 and 1 (non-polar). Such bit-sequences have noise-like properties like spectral flatness and low cross and auto correlation values, and thus complicate jamming or detection by non-target receivers [Gol67, HdV71, Roe77, Gla92].

Several families of binary PN-CODES exist: M-sequences, Gold-codes and Kasami-codes where the latter two can be created by combining a number of selected M-sequences. An usual way to create a PN-CODE is by means of shift-registers with feed-back taps. By putting the feed-back taps at specific positions, the output sequence of a shift register is of "maximum length". The above mentioned code-families [Gol67] have this property. When the length of a shift-register is n, the length of the resulting sequences is [Gol67]:

$$N_{\mathrm{DS}} = 2^n - 1. \tag{1.2}$$

In direct-sequence systems the length of the code is equal to the spreading-factor, so:

$$G_p(\mathrm{DS}) = N_{\mathrm{DS}}. \tag{1.3}$$

This can also be seen from figure 1.2, where the spreading processes is illustrated, in this example $N_{\mathrm{DS}} = 7$. The bandwidth of the data signal is now multiplied with a factor N_{DS}. The power contents however stays the same, with the result that the spectral power density is lowered.

The generation of PN-CODEs is relatively easy. A number of shift-registers with feed-back taps is all that is required. For this reason it is easy to obtain a large processing-gain in Direct-Sequence systems.

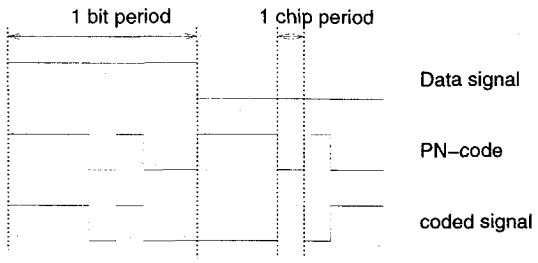


Figure 1.2. Direct-Sequence spreading

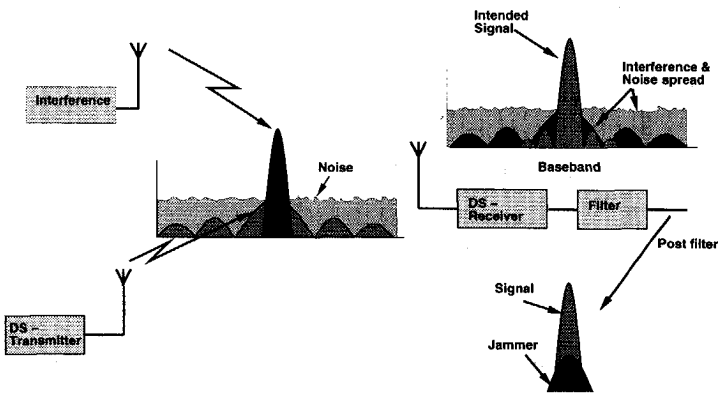


Figure 1.3. DS-concept, before and after despreading

In the receiver, the received signal is multiplied again with the same (synchronized) PN-CODE. Since a code exists of +1s and -1s, this operation completely removes the code from the signal and the original data-signal is left. Another observation is that the despread operation is the same as the spread operation. The consequence is that a possible jamming or interference signal in the radio channel will be spread before data-detection is performed. In this way jamming effects are reduced (see figure 1.3 [Hen84]).

A large problem with multi-access direct sequence spreading is the so-called near-far effect which is illustrated in figure 1.4. This effect is present when a CDMA interfering transmitter is much closer to the receiver than the intended transmitter. Although the cross-correlation of "code A" and "code B" is low, the correlation of the received signal from the interfering transmitter with "code A" in the receiver can exceed the correlation of the received signal from the intended transmitter and

the correct code. As a result proper data detection is hardly possible.

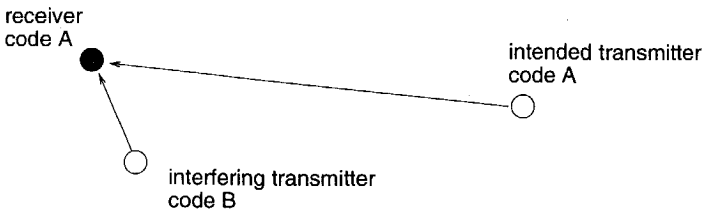


Figure 1.4. Near-Far effect illustrated

1.2.2 Frequency Hopping

When applying frequency hopping, the carrier frequency is "hopping" according to a unique sequence (an FH-SEQUENCE of length N_{FH}). In this way the bandwidth is increased by a factor N_{FH} (if the channels are non-overlapping):

$$G_p(\mathrm{FH}) = N_{\mathrm{FH}}. \tag{1.4}$$

The process of frequency hopping is illustrated in figure 1.5. A disadvantage of frequency-hopping compared to direct-sequence is that it is hard to obtain a high processing gain. A frequency synthesizer is required that is capable of rapidly hopping over a set of carrier (FH) frequencies. The more FH-frequencies, the higher the processing gain and the more demanding the frequency synthesizer becomes.

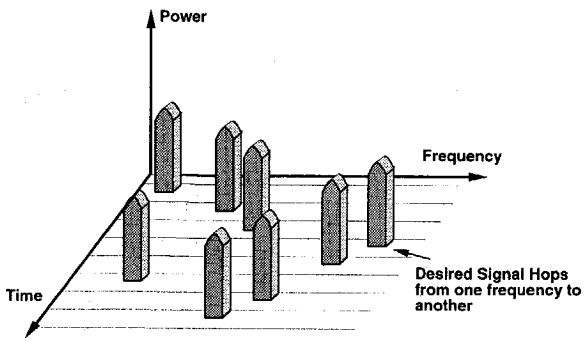


Figure 1.5. Illustration of the frequency hopping concept

On the contrary, frequency-hopping is less vulnerable to the near-far effect than direct-sequence. Frequency-hopping sequences have only a limited number of "hits" with each other. This means that if a near-interferer is present, not the whole signal is blocked but only a limited number of "frequency-hops". From the "hops" that are not blocked it should be possible to recover the original data-message, for instance by applying error correcting techniques.

Two types of frequency-hopping techniques can be distinguished. In "fast frequency hopping" the period of a "frequency-hop" is smaller than a data symbol-period while in "slow frequency hopping" the period of a "frequency-hop" is larger than a data symbol-period. Choosing one of those techniques has consequences on the error correcting coding to be applied.

1.3 Claims of the thesis

This thesis is about the design of a complete communication system. from front-end related issues to digital baseband processing aspects as well as the complete design trajectory: from idea down to a ready for layout description of the target system.

The goal of this section is to mention the highlights that surfaced while working towards this thesis. These points can be grouped into three categories:

Concept development of communication systems concerns the step from idea (application) to system definition. Aside from the combination of all system-concept issues into a system definition, the following points may be of special interest to the reader:

1. *Selecting an adequate multi-access technique*
   As CDMA-techniques combine random access with an interference limited system-utilization, they are very suitable to provide multi-access abilities in ad-hoc communication systems. However, all CDMA-techniques have their specific disadvantages. The choice of an adequate multi-access technique is therefore an interesting one that also has a large impact on further design stages.

2. *An acquisition search algorithm independent of a threshold*
   Usual code-acquisition search algorithms make use of a threshold. This threshold however, is difficult to obtain as a number of parameters are unknown to the system. We will therefore propose an acquisition search algorithm that does not primarily depend on a threshold value.

3. *Suitable MA-interference analysis*
   The modelling of the interference from all other users as Gaussian noise

is not realistic in non-cellular systems. A more suitable MA-interference analysis has been worked out.

4. *Effectively reducing the effects of near-user interference*
   Combining salient features of code-acquisition and frequency-hopping leads to an acquisition algorithm independent of near-user interference.

Implementation issues in a CDMA-receiver During system implementation, we often find that the requirements of the systems engineer cannot be mapped on the available hardware and software resources. To tackle this problem one can always resort to more advanced (and more costly) resources or degrade the desired properties. In other words: the trade-off between performance-loss and cost becomes visible.

1. *Complexity reduction of the data-detection algorithm*
   The complexity of the MFSK data-detector (for MFSK data modulation see chapter 3) can be reduced by applying a discrete fourier transform while reducing the number of bits used to represent internal numbers.

2. *Applying 2-level input signals*
   A considerable reduction of hardware cost can be obtained by limiting the input-signal. This turns out to have only a small effect on the BER-performance.

3. *Novel code-tracking loop*
   Through modification of an existing code tracking loop the complexity of the tracking-algorithm is reduced while keeping a satisfactory code-tracking performance.

Building a transceiver as an embedded system includes a novel design-flow to implement an actual design of this kind. A description in the programming language C of the transceiver's algorithm, and trade-off tables for different implementation alternatives function as inputs to the hardware/software partitioning stage. Special attention is asked for:

1. *Applying automatic tools*
   When applying an automatic tool, a designer can explore the design space in an efficient way. Profiling results guide the designer to a "close to optimal" hardware/software partitioning.

2. *The designer stays in control*
   In spite of the fully automated partitioner, the designer stays in control of the design process.

3. *Graphical representation of the algorithm*
   To exercise this control, a graphical representation of the operation of a system can be of great help in increasing efficiency.

4. *Handling real-time constraints*
   Real-time system design requires a quite different approach from the design of systems without hard timing constraints.

The claims above are verified against the experience with the design of the WISSCE CDMA-transceiver.

1.4 Overview

This thesis discusses a number of important issues existing in the process of realizing non-cellular wireless communication systems. Though we tried to touch all aspects that exist in the design of a communication system, emphasis was on the aspects mentioned in the previous section.

A non-cellular communication system has the advantage to be "mobile" itself. Users can communicate with each other anywhere as long as their relative distance is within specification. The systems that will be addressed here are also meant for short-distances, for instance to provide ad-hoc indoor communication links. We saw that code division multiple access (CDMA) closely fits the ad-hoc nature of non-cellular communication systems for indoor use.

Before making a system design it is important to have a notion of the resources that are available to build the system. Chapter 2 therefore deals with the available resources to implement the baseband processing (synchronization and recovering of the transmitted data-message) of the transceiver. In this chapter we will conclude that an embedded realization almost completely matches our desires towards an implementation. We will also motivate why a transport triggered architecture provides a good processor framework.

The starting point of the system specification will be an analysis of the customer's wishes. Although this aspect was already partly covered in the first sections of this chapter, chapter 3 clarifies the user demands. That chapter also deals with the next step in system design: "How to translate the user-demands into a system-specification?".

Chapter 4 addresses the relation between the bit error rate and the signal to noise ratio before analog to digital conversion. Also specific CDMA related issues as code-selection and code-synchronization are also addressed.

Before we can actually map the system-specification on the available resources, a major problem has to be solved. The system engineer's ideas usually turn out to be

too demanding to fit on the available hardware and software. To this end chapter 5 introduces a number of simplifications to the receiver algorithm, to validate these steps an evaluation of the introduced performance loss will be made.

Now that there is a "realizable" system design in the form of an algorithm, a start can be made mapping this algorithm on the available resources. A hardware/software partitioning stage is required as we selected an embedded system concept to realize the baseband processing. The HW/SW-partitioning process using the partitioning tool HSPART is addressed in chapter 6. The chapter concludes with an evaluation of the partitioning results of a practical application.

How the actual co-design is done will be discussed in chapter 7. Here the application specific hardware modules as well as the transceiver's firmware are described. This chapter also deals with the usage of a co-simulation tool. Co-simulation is important in the evaluation of the cooperation of both hardware and software.

After discussion the important aspects existing in the realization of a communication system, an overall conclusion towards the followed trajectory will be given in chapter 8.

For the reader's convenience, a glossary is included at the end that summarizes all symbols and abbreviations used throughout this thesis. An index provides the ease to look up certain terms.

Chapter 2

AN EMBEDDED SPREAD SPECTRUM PROCESSOR

Contents

2.1 **Introduction** 13
2.2 **Why an Embedded Realization?** 14
2.3 **The design process** 16
2.4 **Selecting a Processor-Framework** 18
2.5 **Requirements for the Hardware/Software partitioning stage** 20
2.6 **Available resources** 21
2.7 **Conclusions** 22

2.1 Introduction

Apart from system constraints, implementation issues are to be considered as well. In this chapter we will discuss the the implementation concept, the CAD-tools to apply and the available resources. It is important to face these aspects before doing the actual system design. In this way impractical designs can be discarded at an early stage.

Next section addresses implementation problems specific to real-time applications like wireless communication systems. It will be shown that there exist good reasons for applying an embedded system design methodology. The embedded design trajectory will be discussed as well.

In designing an embedded system the choice of processor is of great importance. The performance and structure of such a processor influences the choice of what functionality to implement in software under the existing constraints. The choice of a processor architecture is discussed in section 2.4.

Once a target processor architecture is chosen and the system is formally described, the hardware/software (HW/SW) partitioning stage can start with the automatic generation of profiles. Section 2.5 outlines the requirements we have for the results of this stage.

2.2 Why an Embedded Realization?

Most transceiver architectures have a natural partitioning into two parts: an analog front-end and a digital back-end. The implementation of the latter, and the baseband processing (synchronization and recovering of the transmitted data-message) in particular will be the focus of the following discussion. To enable this discussion, an unambiguous algorithmic description of the digital back-end is required. An additional requirement is that verification of the system should be possible. For this reason it is important that the algorithmic description can be simulated easily. A programming language like C can for instance be used for this purpose.

Baseband operations exist to a large extent of controlling functions and mathematical processing. For this reason it would be appropriate to implement the baseband processing completely in software. Software implementations have the advantages of being cheap (no application specific hardware) and flexible. However, software is in general slow compared to hardware.

Whether the complete operation can be implemented in software depends on the timing constraints that exist for a certain application. For the target system it is already questionable whether today's computers are sufficient. Consequently, it is infeasible to implement the complete baseband processing on the resources we have available.

Real-time systems with hard timing constraints like TV-sets, automotion systems, radio-receivers etc. do not function properly if timing constraints are not met. This in contrast to systems which do not show hard timing constraints like laser-printers, washing-machines etc. In such applications timing affects the speed, not the functionality.

Obviously, if a TV-set is on average not able to process one frame before the next frame starts, functionality is lost. Most communication systems are examples of real-time systems with hard timing constraints. Like in the TV-set example, processing has to be completed within a fixed frame, otherwise the system does not work correctly.

As a consequence, there is not a simple trade-off between cost and speed. To enable proper operation, a minimal processing speed has to be satisfied if the performance is fixed. Due to the hard timing constraints it is unlikely that implementing the transceiver exclusively in software will still be possible. A logical solution now is to move certain functionality from software into hardware to preserve software domination.

This observation leads to an embedded system: an embedded system implements certain real-time functionality by using an optimal combination of dedicated hardware and software working together and concurrently. Another property of an embedded system is that it runs the same software over and over again.

Once a communication procedure is completely specified, it has to be decided what functionality to put in hardware. Traditionally an embedded system was just

a piece of hardware and one or more processing elements which cooperated to perform a certain functionality. After the manual partitioning of what functionality to implement in hardware and what to do in software the two parts were implemented independently.

Nowadays requirements concerning efficiency are getting more stringent. For this reason the interaction between hardware and software has to be taken into account in the design process. By designing the hardware and software parts of a system "together", the barrier between the two parts lowers. It now becomes possible to move functionality from hardware to software and the other way around.

Usually an embedded design trajectory starts by writing an algorithmic description of the target system in a high-level language. Then the designer is in control of choosing a processor, doing the HW/SW-partitioning and designing the different parts [Str94].

Different opinions exist on the process of hardware/software partitioning. In many situations, people define an embedded system as consisting of a combination of a general purpose processor and a co-processor [EHB93], or as a standard programmable element together with an ASIC [GCD94]. In these situations, a gap between hardware and software still remains. To reduce the distance between hardware and software further, the software and hardware parts of a system can be implemented in the same processor framework. Such an framework can for instance consist of a single chip.

As the hardware and software parts are designed concurrently, the hardware and software parts are now optimally "tuned" to each other. In that case the processor-architecture can be configured in such a way that user-defined functionality can be included as well as general purpose functionality. The question of HW/SW-partitioning now becomes the question of what application specific functionality to include in the processor framework.

Still the question remains of what functionality to implement in hardware and what functionality to put in software. In our point of view the designer should be presented a lot of profiling results which can be used by the designer to efficiently evaluate the large design space.

A number of issues exists in the partitioning process:

1. As a designer is usually a hardware designer *or* a software designer, he/she will have an eye for either an hardware or an software optimization, not for the complete system. A tool that gives "objective" profiles has advantages from this point of view.

2. As the size of the systems grows, it is getting more and more difficult for a designer to keep track of all relevant details.

3. An automatic system is capable of reviewing a large number of sample-points from the large design space of the HW/SW-partitioning process. From there the designer is able to get a "feeling" for the possibilities.

4. An automatic tool enables the development of tools with which non-experts can arrive at acceptable designs.

5. Although we state that it would be advantageous that non-experienced designers will be able to solve the HW/SW-partitioning problem, there is no tool available to us capable of providing this service. It is therefore still required that a designer stays in control of the partitioning process.

To summarize: Implementing the baseband processing of a communication transceiver completely in software would nicely fit its nature and provide a flexible and cheap solution. In many situations however, hard timing constraints disqualify this approach. As a result, part of the functionality has to be implemented in hardware. Therefore an embedded implementation is proposed to obtain an efficient design. In such a system the hardware and software parts form a single framework. An HW/SW-partitioning tool is required to help the designer obtaining a HW/SW-partitioning that can be verified using co-simulation in a later stage.

2.3 The design process

There are three inputs to the design process:

- *Specification*
  An algorithmic description that specifies the behavior of the system.

- *Constraints*
  There are constraints to be satisfied during the implementation stage. These can be for instance timing, area or power constraints. We will concentrate on both timing and area constraints.

- *Processor architecture*
  The choice of a processor architecture family is essential as it has a large influence on the costs of software implementations. It is however time costly to evaluate different processors and the choice should off-line: it affects the partitioning process from the very beginning.

After supplying these inputs a translation stage starts. Here the system specification is analyzed to select those functions for which the HW/SW-partitioning question plays. This translation results in two graphs: firstly a control-flow graph that shows the sequence and parallelism of operations to be performed. And secondly a data-flow graph that represents the data transport between operations.

For all functions and variables appearing in these graphs cost-data has to be supplied. This data appears in the form of tables: for all possible implementation choices and interfaces, cost-numbers on area and timing are provided. As a fast alternative usually takes more area than a slow implementation and vice versa, these tables will typically show a trade-off between area and speed.

The graphs, constraints and delay/area trade-offs are the input to the HW/SW-partitioning stage. During this step the HW/SW-partitioning design space is evaluated to arrive at an optimal partitioning for the given inputs. This is an interactive and iterative process: before a partitioning run, the designer is able to lock functions into certain alternatives and to let the tool optimize for sets of remaining functions. If the result is not satisfactory another partitioning run can be started.

Once a partitioning result is accepted, the implementation alternatives should be implemented. To verify the cooperation between hardware and software however, first a co-simulation of the selected configuration is performed. On basis of the co-simulation results we may want to start the HW/SW-partitioning process again or change the input specification. After obtaining satisfactory simulation results, the actual hardware can be realized.

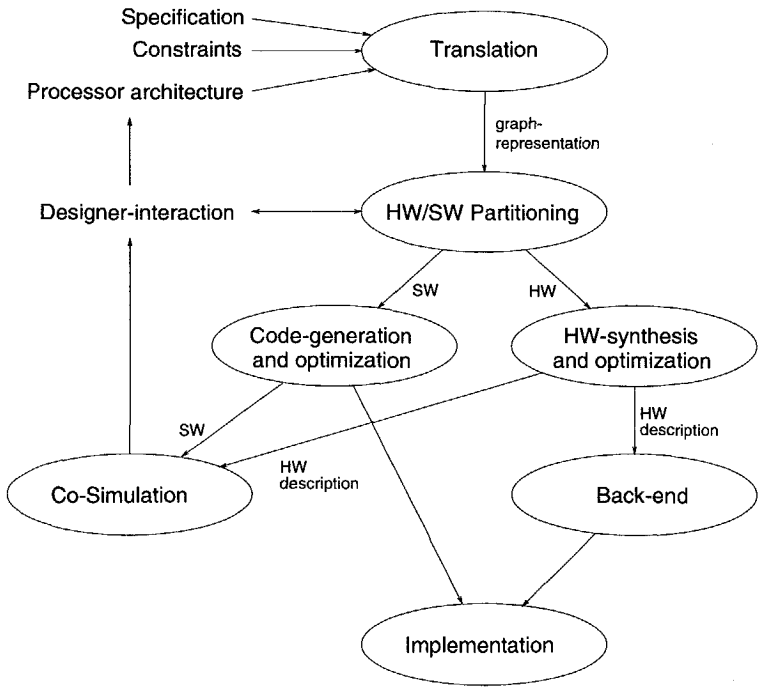


Figure 2.1. Embedded system design process

2.4 Selecting a Processor-Framework

It is evident that lowering the barrier between software and hardware as much as possible leads to an efficient implementation in the sense that hardware and software almost seamlessly work together. However, before performing the hardware/software partitioning stage, a processor framework has to be selected. The choice of a processor framework is important: it determines to a large extent the costs of the software implementations. The following requirements are formulated for a processor framework:

1. *Generality*
   It should be possible to fully configure the processor core with desired "general purpose" functionality.

2. *Easy incorporation of dedicated functionality*
   Dedicated functionality also has to be implemented within the processor framework.

3. *Uniform interfacing*
   The interface between the processor-core and general purpose functionality or application specific functionality should be uniform. If however, application specific functionality has external I/O, a processor-hardware synchronization protocol is likely to be required.

4. *Various execution times*
   Different functionality will show different execution times, the processor has to adapt to this property.

5. *Instruction parallelism*
   An important advantage of an hardware realization is that it enables parallel processing. If the processor architecture is sequentially going through a set of operations, or stays idle while application specific functionality is active, this advantage is lost.

The so-called transport triggered architecture (TTA) [Cor95] closely fits these requirements. This clocked architecture is illustrated in figure 2.2. Central to a TTA-architecture is a set of busses. Different kinds of functional units (FUs) can be connected to these busses via so-called SOCKETs.

An FU typically has three registers: an operand register, a trigger register and a output-register. The operand and trigger registers are inputs. As soon as a trigger-register detects new data, the FU starts its operation. After a certain "latency" (execution time in clock cycles), the result appears at the output-register. For example, to add two numbers a and b, a is moved to the operand register of the adder-FU and b is moved to the trigger-register. After that, the addition starts and a number of cycles (latency) later the result can be read from the output-register.

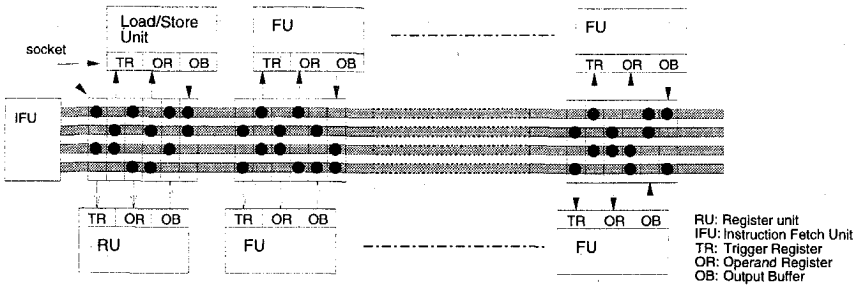


Figure 2.2. Structure of a transport triggered architecture

The advantages of this architecture are its simplicity and flexibility while preserving its completeness: it is still possible to use the processor in general purpose situations. The flexibility however is also creating a new problem: how to handle the large design-space? We already mentioned that the number and kind of FUs can be chosen freely, but other parameters have to be specified as well.

bus width The bus width determines the size of the words to be processed by the FUs. Typically multiples of 8 are chosen, today's computers often show a bus width of 32 [Cor95]. This number however, is likely to be an overkill for embedded systems as our target system. As there are a number of busses, reduction of the bus width leads to a smaller area-occupation. As a result a trade-off between the possibility to handle large numbers and area-occupancy becomes visible.

number of busses As the FUs can operate concurrently, the degree of potential parallelism is determined by both, the number of FUs and the number of busses. As a bus consists of a number of data lines (bus width), reducing the number of busses again leads to substantial area saving (potential parallelism against area).

width of address-bus Beside a data-memory, also an instruction-memory exists. This instruction-memory contains the moves that have to be performed. The wider the address-bus, the more FUs can be addressed in a processor and the more flexible the processor becomes (flexibility against area).

number of register-units As a register-file enables the processor to have fast access to temporarily data, the size of the register file determines to a certain extent the speed of a process (speed against area). A constraint towards the minimum size of the register-file however exists (to enable scheduling [Hoo96]).

Concluding: we saw that a transport triggered architecture is quite suitable as an processor framework in an embedded system such as our target system. This

architecture will be the basis for implementing the baseband processing. In chapter 7 a detailed description of the mapping the receiver algorithm onto this processor is given.

2.5 Requirements for the Hardware/Software partitioning stage

Evaluating the large design space of hardware/software partitioning is initially impossible without the usage of automatic tools. To this end, tools are being developed. Examples are COSYMA [EHB93], VULCAN [GCD94], PAES-I [BISH96] and HSPART [KO95, Kar95].

These tools require cost-data on the possible implementation alternatives to find profiles. As the cycle-time is bounded below by the chosen processor architecture, we will express timing-costs in terms of latencies: the number of cycles that it takes to complete an operation. Area cost is expressed in terms of gate usage.

As cost-data is not available at this stage, the designer will usually have a hard time collecting it. A safe way to do this, is by designing the actual software and hardware implementations and then use profiles and simulations. This however introduces the need for extra manpower. In general even the data resulting from such exercises will contain uncertainties which in their turn introduce the risk of obtaining inefficient or even invalid partitioning results. So investing much effort in the extraction of cost-data is not a guarantee for obtaining an optimal result.

This problem can be coped with by applying HSPART [1], a tool built on top of the CASTLE-environment [TSV94]. The application of this tool avoids the requirement to supply exact cost-data. The algorithm implemented in this partitioner uses imprecise (possibilistic) input data: supplying a "most-possible" value, a "minimum" value and a "maximum" value is sufficient. Usually it is not difficult to find these numbers. For instance area-costs: a minimum value can correspond to the number of gates you need without wiring while the maximum value can correspond to the result a fast layout-run. The influence of the most-possible value can be controlled. The salient feature of HSPART is the possibility to control the risk of getting out of the specification due to wrong-guessed cost-values.

To summarize, we expect the HW/SW-partitioning-stage to be guided by an automatic tool but controlled by the designer. It should provide the following:

1. Profiling results of possible hardware/software partitionings.
2. Suggestions concerning the amount of "standard functionality" to include in the processor framework.
3. Suggestions how to handle clustering: what functionality can be combined to arrive at a reasonable number of application specific functional units?

[1] The usage of HSPART is described in chapter 6 where the tool is applied in a practical design

2.6 Available resources

The choice of what resources to use is essentially based on the the available resources at the circuits and systems group in Delft. The OCEAN [GS93, Str94] tool-set provides placement and routing for semi-custom Sea-of-Gates IC-design. At the same moment such ICs can be further processed at DIMES.

The digital part of the transceiver should therefore, if possible, be implemented on a single Sea-of-Gates (SOG) chip. Such a chip is based on a semi-custom ic fabrication process available in Delft and uses the *fishbone* image: a gate-isolation image in a 1.6μ CMOS process with 2-level metallization. A single chip has has about 100.000 N/P-MOS transistor pairs. On this chip the hardware-functionality as well as the software functionality should fit. The sea-of-gates design system OCEAN is being used for prototyping. As an illustration figure 2.3 gives a 3-D view of part of a Sea-of-Gates circuit. Achievable clock speeds for a processor on Sea-of-Gates are up to 50 MHz.

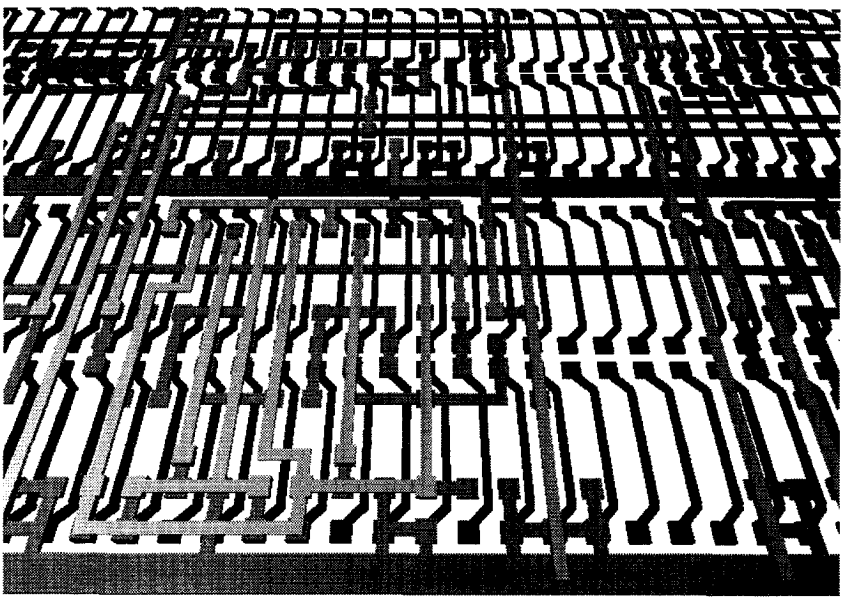


Figure 2.3. 3-D view on a Sea-of-Gates circuit

2.7 Conclusions

The goal of this chapter was to find an implementation frame to be used to realize the digital baseband processing of the target communication system. We concluded that a complete software realization fits the nature of the operations to be implemented, however, such an approach is out of the question because of the existing hard timing constraints. On the other hand a complete hardware realization is not flexible and expensive. An approach in which hardware and software cooperate seamlessly provides a solution. This is what we call an embedded system.

The design-flow existing in embedded system design differs from usual design flows in the sense that the operation of the target system is partitioned into hardware and software functionality. We saw that an automatic tool to guide the designer through the large design space is highly desirable to efficiently perform the partitioning.

An important choice is to select an appropriate processor framework. For efficiency reasons we make this choice before the partitioning process starts (to limit the enormous design space). After evaluating our requirements towards such a processor framework, we selected a transport triggered processor architecture (TTA) to be the core of our embedded system.

Another way to limit the design space to a reasonable size was by selecting the available resources. At the end of this chapter an indication of these resources was given.

Chapter 3

HYBRID DS/FH SPREAD SPECTRUM COMMUNICATION SYSTEM

Contents

3.1 Introduction **23**
3.2 System Specification **24**
3.2.1 Frequency bands 27
3.2.2 Transmission capacity 28
3.2.3 CDMA technique 30
3.2.4 Modulation format 32
3.2.5 Front-end related issues 32
3.2.6 Bit Error Probability 33
3.2.7 Analog to digital interfacing 33
3.3 Clock control **35**
3.3.1 Front-end clocks 36
3.3.2 FH-synthesizer clock 37
3.3.3 Baseband processing clocks 38
3.4 Fixing the system parameters for WISSCE **39**
3.4.1 System related issues 40
3.4.2 Clock related issues 40
3.5 Conclusion **41**

3.1 Introduction

The system specification stage can be characterized as a translation of user demands into a technical specification. In this chapter we will concentrate on the trade-offs that exist during the system design stage. As an illustration, the design of a non cellular communication system referred to as WISSCE will be explained.

This acronym "WISSCE" stands for Wireless Indoor Spread Spectrum Communication Equipment. This system will function as example of the design process throughout the remainder of this thesis.

This chapter is primarily focussed on system-level aspects. We however will keep in mind the implementation issues addressed in the previous chapter. In the next section the trade-offs that exist between the system parameters will be discussed. Section 3.3 addresses the way the various clocks in WISSCE can be derived from a single time-base. The actual choice of parameters is made in section 3.4. At the end a conclusion shortly summarizes the results of the system specification stage and provides an summary of WISSCE's system specification.

3.2 System Specification

In chapter 1 the application domain of the target communication system was already briefly described. The goal of this section is to clarify the communication concept and evaluate the user demands. In this way a basis for the system specification can be found.

The concept of non-cellular communication systems requires no installation effort. A complete system only consists of a set of transceivers and is therefore mobile itself. A group of people might for instance go somewhere and still be able to communicate with each other as long as their relative distance is within specification.

So a non-cellular solution provides an enormous flexibility while it is also keeps prices acceptable in the sense that no additional investment in infrastructure is required, on the condition that the price of a single transceiver is comparable to the price of a mobile in a similar cellular system.

To enable communication, connections have to be made which requires a protocol. Such a protocol should fit the flexible nature of the proposed application domain and therefore has to be simple. A simple and effective protocol which will be used in the following, works as follows: an initiator starts transmission by making a call to an arbitrary other device (addressee). To this end the initiator transmits using the address (CDMA-code) of the intended addressee. Once the addressee acquired synchronization, it gives an acknowledgement back to the initiator. If the initiator can synchronize to this message within a certain amount of time, the transmission of the intended data-message can start. The consequence of this approach is that during the synchronization stage both transceivers are transmitting simultaneously. During "normal" data communication this duplex connection is also required to remain synchronized. As a result the transmission system must provide full-duplex communication links.

Now that the non-cellular concept is explained, the user demands can be evaluated to fix technical parameters.

1. *Universal usage*
   Paramount to the user is undisturbed, low risk usage. For a large number of countries this means that a communication system has to obey legal requirements. The dutch situation requires spread spectrum communication systems to operate in the 2.4 GHz ISM-band. In most other western countries this frequency-band is available as well. The usage of this frequency band imposes a number of restrictions. The most stringent restriction is the bandwidth limitation. In the following we will refer to this user-demand as "meeting legal requirements".
2. *Reliable operation*
   Users wish error-free transmission. As complete error free transmission is not possible, the error probability should be minimized. In data communication it would be possible to "retransmit" erroneous messages on the cost of a lower transmission capacity. For voice applications however, this is not possible.
3. *In-door usage*
   The system is targeted for short distances so an important application will be as indoor communication equipment. Consequently operation in the (hostile) in-door environment should be possible.
4. *High transmission capacity*
   Pure data communication applications nowadays demand high transmission speeds. The transmission speed is however limited by the available bandwidth and the number of users that have to be allowed in the system.
5. *High active user capacity*
   A difference can be made between "active user capacity" and "total user capacity". The first capacity is determined by the amount of interference that can be allowed in the system. The total user capacity however, is determined by the number of available addresses (CDMA-codes). At this point the focus is on the "active user capacity".
6. *Large area coverage*
   An important consideration is the maximum allowed distance between users. The system is meant for short-distances. For flexible usage however, this "short-distance" should be not too short. On the other hand the larger the distance, the higher the energy usage will be and the more interference will be generated for the other (near) users. The latter results in a worse mean receiving quality.

   The further users are apart, the worse the receiving quality will be. So the radio link will fail "softly".
7. *Low purchasing costs*
   The complete communication systems consists of only transceivers and is

meant for the mass-market. In comparison with cellular systems, a non-cellular approach provides flexibility. It is however important that the price of a transceiver is comparable to that of mobiles in a similar cellular system. The purchasing costs therefore should be low.

8. *Low operation costs*
   Power consumption should be low to decrease operation costs. A service provider is not involved as the system does not depend on an infrastructure.
9. *Flexibility*
   The system should be flexible in the sense that many different applications can be allowed.
10. *Mobility*
    A complete communication system consists of only transceivers and is therefore mobile itself. For the situation in which a transceiver is used in a mobile situation (e.g. as hand-held), power consumption should be low as well.
11. *Short access time*
    The time to build a transmission link should be short.

In table 3.1 the relation between important user-demands and technical specifications is given. At the left side of the table the user-demands are listed that were described above. At the top technical properties are given. In the first column the relative importance of a demand for the user is shown. A single "+" means "rather important" while "+ + +" expresses a meaning of "mandatory". For instance: "meeting legal constraints" is mandatory while "large area coverage" is only desirable. The next 7 columns to the right show the relation between user demands and technical properties like bandwidth, processing gain etc. A "+" on one hand expresses a positive relationship: the larger value of the property (for instance "larger bandwidth"), the higher the customer satisfaction. On the other hand a "–" shows a negative relationship, to satisfy the user the property should have a low value. The right 2 columns show properties for which grading does not make sense. A "+" expresses in these columns that a user-demand is in favor of the property while a "-" shows the opposite.

From the table we observe that different user demands lead to contradictory technical solutions. For instance a "high user capacity" is in favor of a low symbol rate while "high transmission capacity" wants the opposite. This is the reason for taking the relative importance of all user demands into account.

The available *bandwidth* is an important system parameter and therefore the first property to be described in more detail (section 3.2.1). After fixing the available frequency-bands, it is logical to treat the *symbol-rate* and the *number of bits per symbol* to find a relation between CDMA spreading-factor, bit-rate and available bandwidth (section 3.2.2). The CDMA-spreading technique and *processing gain* is addressed in section 3.2.3 while the modulation-format itself is coped with in section 3.2.4.

	Importance	bandwidth	symbol rate	number of bits per symbol	processing gain	frontend complexity	output power	sampling speed	coherent data detection	software implementation
meeting legal requirements	+++	-					-			
in-door usage	++				+	+	+		-	
high transmission capacity	++	+	+	+		+		+		-
high user capacity	+	+	-		+	+		+		
reliable operation	++	+	-			+	+	+	+	
large area coverage	+						+			
low purchasing costs	+	-	-	-	-	-		-	-	+
low operation costs	+						-	-	+	+
flexibility	++	+	+							+
mobility	+						-			
short access time	+		+	-	-		+		-	

Table 3.1. relation between user-demands and technical properties

The term *front-end complexity* can be interpreted as the number and quality of the components used in the front-end. As this property and the *output power* are strongly related, they are both dealt with in section 3.2.5. The required reliability ("reliable operation") is addressed in section 3.2.6, then finally section 3.2.7 addresses the interface between the analog and time-discrete domain.

Obtaining the system parameters is a highly iterative stage, all parameters influence each other. As a result it is difficult to discuss the system design process step by step. The following sections shows an attempt. However, cross-references throughout the sections can not be avoided.

3.2.1 Frequency bands

In the Netherlands the available frequency band for spread spectrum communications is the so-called "2.4 GHz ISM-band" with a total of 30 MHz (2445-2475 MHz).

We saw that full-duplex communication links are required (page 24). If using a single frequency band for both receiving and transmitting, a transmitter will function as a near-interferer to its own receiver and correct data transmission is hardly possible. For this reason the available 30 MHz frequency band has to be divided into a transmission and a reception band. An additional problem is that both bands cannot be located directly next to each other: a guard-band is required to allow for filtering in the front-end.

At this stage a trade-off between occupied bandwidth for a single channel (BW_{total}) and requirements towards the front-end can be observed. The higher the occupied bandwidth, the higher the possible data speed and processing gain but the steeper (and more expensive) filters are required in the front-end. As a compromise we propose a guard-band of 10 MHz. This fixes BW_{total} to maximally 10 MHz as well.

An important issue is that as a transceiver can be both an initiator and an addressee, it should be possible to swap transmission and reception band. This imposes extra requirements on the front-end.

The frequency-plan as introduced in this section is illustrated in figure 3.1.

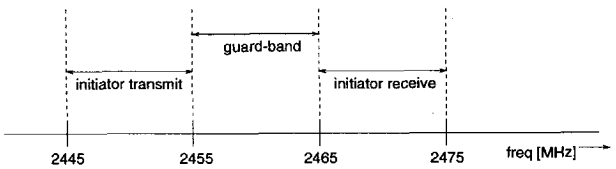


Figure 3.1. Receive and transmit frequency bands

3.2.2 Transmission capacity

From table 3.1 it is clear that the user desires a high transmission capacity (overall data speed). However, it can also be seen that this user demand leads shows conflicts with the demands "low purchasing costs", "short access time", "meeting legal requirements" and others. As a result a compromise should be found.

The relation between occupied bandwidth, processing gain and data speed is:

$$BW_{\text{total}} = G_p \cdot r_{\text{symb}} + \delta_{\text{mod}} \tag{3.1}$$

where G_p is the processing gain (see chapter 1, page 4). δ_{mod} expresses the extra bandwidth usage due to the applied modulation type which is usually small in comparison with the other bandwidth term. A plot illustrating this trade-off is given in figure 3.2.

The symbol-rate (r_{symb}) can be expressed in terms of the bit-rate (r_d):

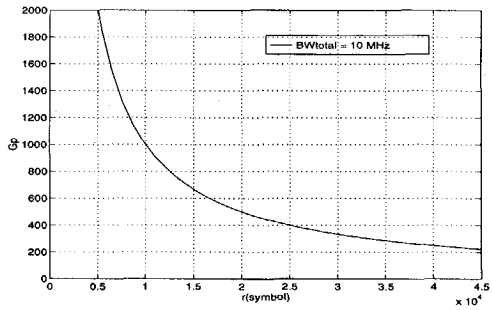


Figure 3.2. Trade-off between processing gain and symbol-rate

$$r_d \geq^2 \log(\#\text{ modulation levels}) \cdot r_{\text{symb}} \,. \tag{3.2}$$

From these formulas follows that the data speed can be increased independently of the occupied bandwidth by increasing the number of modulation levels (= number of bits per symbol). However, increasing this number of bits per symbol increases both the implementation cost and the bit error probability.

Summarizing: increasing the data speed leads to either a larger bandwidth occupancy or higher hardware costs and a worse bit error rate.

Figure 3.3 shows the trade-off between data speed and the number of modulation levels. This plot makes clear that it is advantageous to have a rather small number of modulation levels: the data speed increases with the "^{2}log" of the number of levels while the data detection complexity increases linearly. Appropriate choices are 8, 16 or 32. The actual choice of parameters will be made after considering all trade-offs (section 3.4).

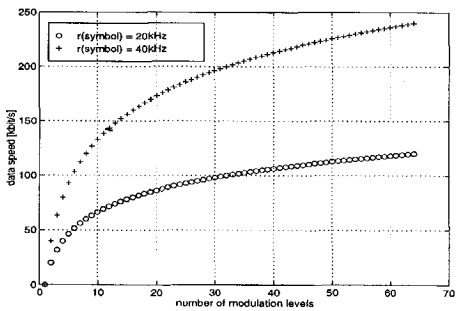


Figure 3.3. Trade-off between data speed and number of modulation levels

3.2.3 CDMA technique

It was already shown that CDMA (Spread Spectrum) techniques have very promising properties for using them in a communication system that has to provide ad-hoc communication links. However, each technique has its own disadvantages: direct-sequence suffers from the near-far effect, while it is hard to obtain a high processing gain with frequency-hopping.

In cellular systems the near-far effect can be reduced by applying power-control. Power-control is an algorithm in which the transmitted power of the mobiles is controlled in such a way that the received power of all mobiles at the base-station is equal. If such a power control algorithm functions as desired, near-far effects are not an issue in such systems. In non-cellular systems however, there is no base-station, consequently, power-control is hardly possible. To retain the advantages of both DS and FH while cancelling their shortcomings, a technique is selected that combines the large processing gain of DS with a reduction of the near-far effect due to FH [1]. This combined technique will be referred to as the hybrid DS-FH spread spectrum communication technique.

In this technique, a user-address consists of a frequency-hopping pattern of length N_{FH} and N_{FH} (possibly different) PN-CODEs of length N_{DS}. The start of a new set of PN-CODEs and an FH-sequence are linked. This is important when regarding the code-synchronization problem (chapter 5, page 85). An example user-address existing of an FH-sequence of length 7 and 7 PN-CODEs is shown in figure 3.4. Every data-symbol is combined with a PN-CODE causing the direct-sequence spreading. Subsequent data-symbols are transmitted in different FH-channels according to a certain sequence to perform FH-spreading. As a result we use neither fast nor slow frequency-hopping. A frequency-synthesizer only has to "hop" at a speed equal to the symbol-rate.

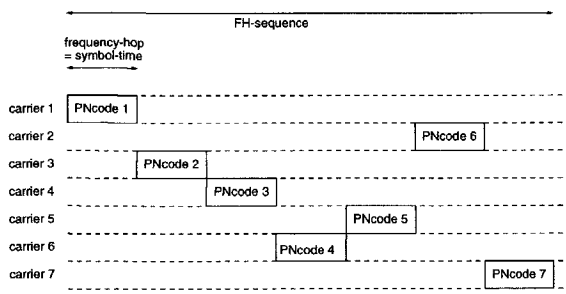


Figure 3.4. A possible user-address

[1] To enable a reduction of the near-far effect, the FH frequency bands should be non-overlapping

For the relation between N_{DS}, N_{FH}, the symbol-rate and the occupied bandwidth holds:

$$BW_{total} = (N_{FH} + 1) \cdot N_{DS} \cdot r_{symb} + \delta_{mod} \tag{3.3}$$

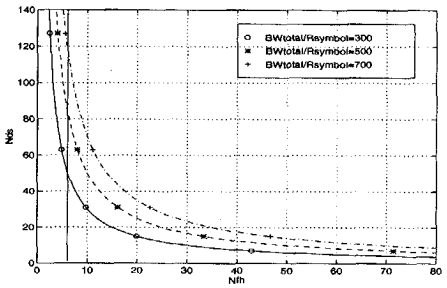


Figure 3.5. Trade-off between N_{FH} and N_{DS}

For constant BW_{total}/r_{symb} and a small δ_{mod}, the trade-off curves shown in figure 3.5 can be obtained for three values of BW_{total}/r_{symb}. Concerning this trade-off the following remarks can be made:

- It is likely that the DS-code will be made using shift-registers (so-called "shift register sequences" [Gol67]). The code-length is then $2^n - 1$ with n being an integer. For this reason, only the marked points in figure 3.5 are possible trade-off points.
- Increasing N_{DS} is advantageous (page 4) and can be realized at low implementation costs.
- Increasing N_{FH} results in the requirement for a more expensive FH-frequency-synthesizer and is for this reason undesired. A minimum number of FH-channels is however required to limit the influence of the near-far effect (near-interference).

To find a minimum value for N_{FH}, the user demands "reliable operation" and "large user capacity" are translated into a technical compromise that maximally 2 near users may be present. As the frequency-hopping sequences are chosen in such a way that they have at most 2 partial hits with another sequence (see section 4.4.2), this compromise converts into the worst case possibility of having at most 4 partial hits per FH-sequence. Furthermore we need at least two frequency-hopping channels without near-interference to enable proper synchronization (see also section 5.3.3) . As a result the minimum value of N_{FH} is 6, a vertical line representing this value is also shown in figure 3.5.

After evaluating other trade-offs as well, section 3.4 will address the actual choice of the values of N_{DS} and N_{FH}.

3.2.4 Modulation format

An issue not yet addressed is the choice of modulation format. Requirements towards this digital modulation format are low implementation costs, the possibility of applying multi-level modulation (having multiple bits per symbol) and a low susceptibility to effects caused by the indoor radio-path.

Three likely candidates for the modulation format exist: amplitude modulation, phase modulation and frequency modulation. Detection of amplitude modulated symbols is problematic in combination with multi-level modulation while frequency-hopping in combination with phase shift keying is susceptible to multi-path effects and requires phase-continuous frequency hopping or a return-to-zero code. This leaves frequency shift keying (FSK) as the better alternative.

A well known disadvantage of FSK is that it requires a larger bandwidth (higher value of δ_{mod}) however this value will still be small in comparison with the already occupied spread spectrum bandwidth. Increasing the number of modulation levels is possible by using more than two frequencies. The resulting modulation technique is referred to as multiple frequency shift keying (MFSK). An additional property of frequency modulation is the possibility of non-coherent data detection. This saves hardware costs as a carrier tracking loop is not required if frequency errors are small.

3.2.5 Front-end related issues

The front-end of a transceiver is responsible for conversion and amplification of the transmitted signal from baseband to an RF-frequency and transmitting it. In addition the front-end should amplify the desired received signal and convert it to a baseband signal.

The maximum distance between transmitter and receiver depends on the transmitted power of a transmitter, the sensitivity of the receiver, the amount of interference that is added to the signal before data-detection takes place and the required bit error probability (BER).

The system is targeted for in-door communication so the distance will probably be much less than 1 km. Multi-path effects however, can dramatically reduce the received signal-to-noise ratio (see chapter 4, page 53). We will therefore require a free-space coverage of 1 km which corresponds to a loss of 100 dB.

How this translates to a required output-power, depends on the "quality" of the front-end (amount of interference added), the number of interfering users, the required BER and the relation between input signal to noise ratio (SNR) to the baseband processing and BER (see chapter 4). Preliminary investigations showed that the required maximum output power will be below 1 mW.

3.2.6 Bit Error Probability

Reliable operation implies a low bit error probability (BER). This performance measure is dependent both on the received SNR-level and the sensitivity of the receiver.

However, using the BER as a performance measure immediately introduces another problem. In an indoor environment fading heavily influences the BER-performance, while the multi-path situation (which introduces fading) can hardly be caught in a "typical" description. The multi-path properties are very irregular.

Yet a BER-performance measure is important as it gives the best indication of the "reliability" of a system. A BER-requirement of 10^{-6} in an unfaded channel (only Gaussian noise present) without error correction in combination with the existing multi-path effects and the application of error correcting coding is expected to result in a satisfactory performance.

In section 4.2 the multi-path effects on the relation between the SNR and the BER will be analyzed.

3.2.7 Analog to digital interfacing

Analog to digital conversion consists of two parts: sampling and quantizing. In this section which deals with the system definition we will only consider the conversion from the analog to the time-discrete domain (sampling) as it has a large influence on the system architecture. Quantizing is considered an implementation issue which will be coped with in section 5.2.

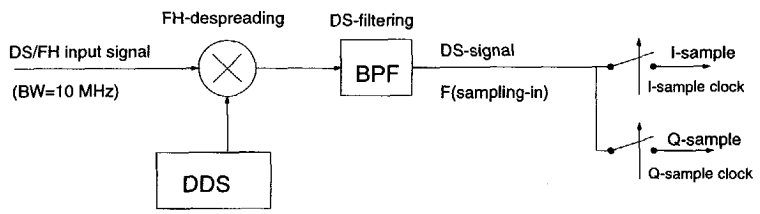


Figure 3.6. Low-frequency part of the front-end architecture

The more important issue in sampling is where in the architecture it takes place. Sampling directly at the antenna leads to a so called "software radio" [Mit95, Bai95, KS95]. This implies large demands on the sensitivity of the analog to digital conversion and the speed of the digital processing to follow. By moving the sampling operation from the antenna in the direction of the baseband processing, the analog part of radio gets more complex (and expensive) while the digital processing becomes simpler. As a compromise, sampling will be performed after FH-despreading and filtering (figure 3.6).

A critical choice is the input center frequency:

1. zero-frequency is attractive as it enables a low sampling frequency. However due to non-linear effects, conversion to baseband also introduces a DC-component. This component is in general hard to deal with as this component is located in the middle of the desired signal and can saturate the sampler. Another drawback is that an extra conversion stage is required.

2. An intermediate frequency, in this situation the DC-component can be filtered out. A disadvantage however is the fact that the sampling frequency should be at last twice the highest signal frequency. So this alternative increases the minimal sample frequency and consequently the speed of the digital processing.

3. Use an intermediate frequency but apply sampling and a frequency translation at the same time. In this way it is possible to convert the signal to DC and enabling a low sampling-frequency while the usual disadvantages of signals at DC are cancelled.

In the latter approach a low sampling frequency and the absence of any DC-component are combined and is therefore the more attractive one. A number of issues concerning the sampling frequency have to be dealt with:

- To minimize the required sampling-frequency, the input center-frequency should be "folded" back to DC. For this reason the sampling frequency has to be an integer times this center-frequency.

- If the desired frequency-band is converted to around DC, quadrature sampling is required to distinguish positive and negative frequency components. Consequently, the digital baseband signal (signal after analog to digital conversion) contains an in-phase and quadrature component.

- The sampling frequency should exceed twice the bandwidth of the baseband signal (Nyquist requirement).

- The sampling frequency should be asynchronous to the chip-rate (r_c) to enable proper code-tracking [FdD86].

- Bit error rate simulations showed that a sample frequency of about four times the chip-rate enables proper operation (the intended quantizing method also plays a role, see section 5.2.1.2 for more information on this aspect).

As the center of the intermediate frequency-band is converted to DC, quadrature sampling is required to enable a distinction between positive and negative frequency components. The quadrature sampling clock will be $\pi/2$ rad. of the input center-frequency out of phase with the in-phase sampling clock. This introduces a quadrature error as a time corresponding to $\pi/2$ rad. at the input-center frequency does not correspond to the same phase difference over the whole frequency band. This is a reason for choosing the input-center frequency high compared to the frequency band used by modulation (δ).

3.3 Clock control

The term "clock-control" did not appear as a technical property in table 3.1. However, this does not mean that it is not a system issue, it just shows that a direct relation with the user demands is not obvious. The clock-control issue is related to the frequency plan in the transmitter front-end, the data-detection method and the requirements of the digital signal processing taking place in the transceiver back-end.

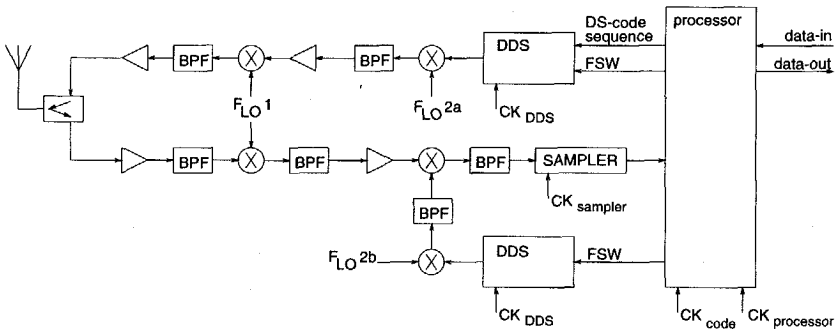


Figure 3.7. Schematic view of possible transceiver architecture

The usage of different clocks is illustrated in figure 3.7, this figure shows a schematic view of a possible transceiver architecture. Both the transmitter (upper) and the receiver (lower) chain) are shown.

The translation from a baseband signal to the RF-transmit frequency is done in a number of steps (super heterodyne). In the transmission chain a frequency-hopping synthesizer (direct digital synthesizer: DDS) takes care of data modulation (MFSK, via a frequency setting words: FSW), FH-spreading (also via the FSW) and DS-spreading (using a DS-code sequence). To generate the DS-code sequence, a code-clock (CK_{code}) is required. The output signal of the DDS is both DS-spread and FH-spread and will be converted in two stages ($f_{LO}1$ and $f_{LO}2a$) to the RF-transmit frequency.

In the receiver chain the RF-signal is converted to baseband in two steps. First using local oscillator signal $f_{LO}1$ to go to an intermediate frequency and secondly using a second local oscillator signal to convert the signal to baseband. This second local oscillator is not fixed, it "hops" according to a pattern directed by frequency setting words (FSW). After the second conversion stage the signal is sampled using a sampling clock ($CK_{sampler}$) to enable the remaining of the processing (including DS-despreading) in the digital domain.

From the above discussion it appears that spreading is performed in the "analog" domain while despreading is done after sampling. The reason for this is that

spreading is easily combined with the generation of the transmit signal (in the DDS) while in the receiving chain more despreading paths are required (see section 5.3.3) so a digital DS-despreading operation is profitable.

Section 5.4 will more specifically address the front-end. A conclusion at this stage is that several clock-signals are required:

1. Front-end clocks, to enable the translation of baseband signals to and from high frequency RF-signals ($f_{LO}1$, $f_{LO}2a$ and $f_{LO}2b$)
2. An input clock to the frequency-synthesizer that performs the frequency-hopping spreading (CK_{DDS})
3. Several clocks used by the digital baseband processing: the sample-clocks ($CK_{sampler}$) and a number of processor clocks ($CK_{processor}$).

3.3.1 Front-end clocks

Front-end clocks are required as local oscillator signals to translate baseband signals to high frequency RF-signals and vice versa. As there is a user demand to meet legal requirements, the RF-frequency is in the 2.4 GHz ISM-band. The exact local oscillator frequencies depend on the frequency plan of the front-end.

A related issue is the fact that when generating frequencies in the 2.4 GHz band, considerable frequency errors will appear due to the limited accuracy of a crystal oscillator. If applying a coherent data demodulation scheme, the frequency error must be controlled (by a carrier tracking loop). To avoid this effort, we chose a possibly non coherent data detection method (section 3.2.2) where the maximum acceptable frequency error follows from the modulation method (see section 5.3.1).

There are two ways to compensate for the inaccuracy of standard crystal oscillators at an RF-frequency of 2.4 GHz: either by using a carrier-tracking loop or by applying a more accurate crystal oscillator. The inaccuracies in the output-frequency are mainly caused by temperature dependencies. A "more accurate" crystal oscillator can therefore be based on the principle of an MCXO (microprocessor controlled crystal oscillator) [BMH89] where using the property of a crystal that it can resonate at different frequencies simultaneously. A dual-mode oscillator provides both the fundamental frequency ($f_{fundamental}$) and the third overtone ($f_{3\text{-overtone}}$). By frequency subtraction of the third overtone and three times the fundamental, a beat frequency is obtained:

$$f_{beat} = f_{3\text{-overtone}} - 3 \cdot f_{fundamental} \tag{3.4}$$

This beat frequency provides a measure for the temperature of the crystal. This measure can be used to control the output frequency. MCXO's are commercially available and reported accuracies are better than $3 \cdot 10^{-8}$ [Q-T94].

Our approach towards an inexpensive implementation of this principle is to extract the beat frequency in the usual way and then use this frequency to control a fractional-N frequency synthesizer to get an accurate reference frequency [Reg].

To summarize: The main problem towards the generation of the clock-signals in the front-end is the accuracy. To solve the problem we propose to apply a crystal oscillator based on the principle of an MCXO.

3.3.2 FH-synthesizer clock

The FH-synthesizer clock is required as a reference clock to the frequency synthesizer that provides the "hopping" carrier. To enable rapid hopping from the one frequency to an other, a direct frequency synthesis technique should be applied. A direct digital synthesizer (DDS) is an almost perfect compromise between all requirements and is therefore selected.

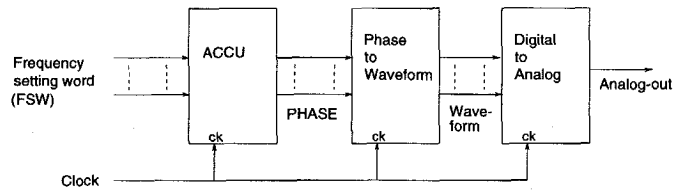


Figure 3.8. Principle of a direct digital synthesizer (DDS)

The DDS-principle is illustrated in figure 3.8. Every clock cycle a digital *frequency setting word* (FSW) is added to an accumulator. The contents of the accumulator represents the phase of the output signal. This phase representation is converted into a waveform and then into an analog output signal. By changing the FSW, it is possible to change the output frequency. In this way a "hopping" frequency synthesizer can be realized.

The "sampling"-frequency of the digital to analog converter should exceed twice the desired output frequency of the DDS. As this "sampling"-frequency is equal to the clock frequency, the latter is an important parameter. To fix a value for this clock, the desired output frequencies should be known. The exact frequencies however are not fixed yet (dependent on N_{FH}, N_{DS}, data speed, etc.), but the occupied bandwidth is about 10 MHz. This means:

1. As the DDS has to "hop" throughout the whole frequency band of 10 MHz, the frequency-band to be covered is 10 MHz as well.

2. The clock-frequency should be as low as possible to save power and allow for an inexpensive implementation.

3. The center-frequency should be as high as possible to reduce mirror-frequencies in the front-end.

4. The combination of center-frequency and clock-frequency should be chosen in such a way that harmonics do not fold back in the desired frequency band.

If a center frequency of about 25 MHz is used, the clock frequency should then be approximately 80 MHz. The exact values should be chosen in such a way that they "fit" other constraints.

3.3.3 Baseband processing clocks

There are mainly 3 categories of low-frequency clocks: sample clocks, processor clocks and the transmitter code-clock.

Sample-clock signals used in the analog to digital conversion of the input signals. Those clock signals were addressed in the section 3.2.7 on analog to digital interfacing.

Processor clock signals As motivated in the previous chapter, the digital back-end will be implemented as an embedded system. The processor architecture which is the core of the embedded system as well as some of the application specific functional units need various clock-signals:

1. All processor clocks should be synchronous to avoid timing problems.
2. To enable an implementation on Sea-of-Gates, the main processor clock should not exceed 50 MHz.
3. The processor-clock should be high enough to enable the completion of a whole "software"-loop within a single symbol-period.
4. To control interfering signals, the exact processor clock frequency should be chosen in such a way that a fixed relationship with other clocks exist.

It is likely that the main processor clock will be in the range 30-50 MHz.

Code-clocks Two code-clocks exist in a transceiver: the receiver code-clock and the transmitter code-clock. In the receiving chain despreading is performed digitally (see section 3.2.7), the receiver code-clock will therefore be derived from the processor clock.

In the transmitter chain however, DS-spreading is performed in the transmitter-DDS using a DS-code sequence that is generated within the processor framework. To enable the generation of the DS-code signal at the appropriate frequency a code-clock signal is required. The frequency of this clock is equal to the chip-rate:

$$r_c = N_{DS} \cdot r_{symb} \cdot \tag{3.5}$$

As can be seen from this equation, the exact frequency of this clock signal depends on system parameters addressed before.

To summarize: in the clock control section we concluded that several clock signals occur in a transceiver. To avoid synchronization problems all clocks should be derived from the same reference clock which is accurate which implies a fixed relationships between all clocks.

The exact frequencies of a number of clock-signals are also dependent on several design issues discussed before. Those frequencies as well as a possible clock scheme will be discussed below.

3.4 Fixing the system parameters for WISSCE

Although still various parameters of WISSCE have to be fixed, a summary of the already established relations is:

$$(N_{FH}+1) \cdot N_{DS} \cdot r_{symb} \approx 10\,\text{MHz} \tag{3.6a}$$

$$N_{FH} \geq 6 \tag{3.6b}$$

$$r_d =^2 \log(\#\text{ modulation levels}) \cdot r_{symb} \tag{3.6c}$$

$$\#\text{ modulation levels} = 8,\ 16 \text{ or } 32 \tag{3.6d}$$

$$30MHz \leq \text{CK}_{\text{processor}} = \frac{n_1}{k_1} \text{f}_{\text{ref}} \leq 50MHz \tag{3.6e}$$

$$\text{CK}_{\text{DDS}} = \frac{n_2}{k_2} \text{f}_{\text{ref}} \approx 80\,\text{MHz} \tag{3.6f}$$

$$r_c = \text{CK}_{\text{code}} = N_{DS} \cdot r_{symb} = \frac{n_3}{k_3} \text{f}_{\text{ref}} \tag{3.6g}$$

$$\text{CK}_{\text{sampling}} = \frac{n_4}{k_4} \text{f}_{\text{ref}} \approx 4\,\text{CK}_{\text{code}} \tag{3.6h}$$

$$\text{f}_{\text{sampler-in}} = n_5\,\text{CK}_{\text{sampling}} \tag{3.6i}$$

f_{ref} is a reference clock-frequency from which all clocks required in a transceiver are derived. n_i and k_i are integers. This set of relations can be divided into two parts: system-related issues and clock-control matters.

3.4.1 System related issues

Common data communication systems nowadays have data-speeds of at least 64 kbit/s. To leave some space for error-correction we therefore choose a target data-speed of 80 kbit/s, this fixes r_d.

To find the symbol-rate the number of bits per symbol has to be determined. A large number of bits per symbol leads to a more complex data detection circuit while it decreases the required bandwidth. By choosing the number of bits per symbol to be 4, the modulation format becomes 16-MFSK and the symbol-speed 20 ksymbol/s.

Evaluating equation (3.1) now results in a processing gain of about 500. This number has to be divided over N_{DS} (length of DS-code) and N_{FH} according to formula (3.3). Increasing N_{DS} is advantageous for a number of reasons (see page 4) while increasing N_{FH} leads to a higher reduction of the near-far effect. This trade-off was shown in figure 3.5.

Two constraints exist: the minimum value of N_{FH} is 6 and for N_{DS} only certain numbers are possible $(2^n - 1)$. From earlier discussions followed that N_{DS} should be as high as possible while increasing N_{FH} leads to higher implementation costs. From figure 3.5 we conclude that a proper value for N_{DS} is 63, for the resulting value of N_{FH} follows 7 and the chip-rate (r_c) is 1260 kHz (equation 3.5).

The power density spectrum of a DS-spread signal has a sync^2 shape where the zero-values are the chip-rate apart. For this reason the FH-spacing is fixed to be equal to the chip-rate ($\Delta_{FH} = 1260$kHz). To obtain orthogonal frequency shift keying, the frequency-distance between the MFSK-frequencies should be an integer times the symbol-rate. To minimize δ_{mod}, Δ_{FSK} is fixed to be 20 kHz.

3.4.2 Clock related issues

With the system related parameters fixed, it is possible to make the clock relations more specific. The first observation is that CK_{DDS} is the highest clock and is therefore the most obvious to use as a reference clock. The generation of both the sample-clocks, the processor-clock and the code-clock can then be done as follows:

$$r_c = CK_{code} = 1260\,\text{kHz} \tag{3.7a}$$

$$CK_{DDS} = 65\,CK_{code} = 81.9\,\text{MHz} \tag{3.7b}$$

$$CK_{processor} = \frac{32}{63} \cdot CK_{DDS} = 41.6\,\text{MHz} \tag{3.7c}$$

$$CK_{sampling} = \frac{4}{63} \cdot CK_{DDS} = 5.2\,\text{MHz} \tag{3.7d}$$

$$f_{sampler\text{-}in} = \frac{8}{63} \cdot CK_{DDS} = 10.4\,\text{MHz} \tag{3.7e}$$

A clock scheme that is capable of generating these clock signals is shown in figure 3.9. This clock scheme is built from two phase-lock-loop loops (see also [Zwa96]).

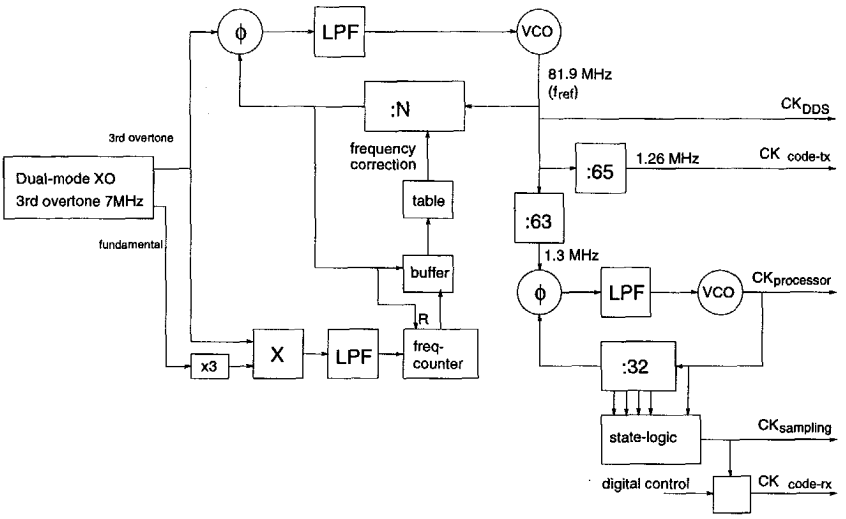


Figure 3.9. Proposed clocking scheme

The first loop is a fractional-N synthesizer, with an input frequency of 7 MHz and in the feedback loop a divider capable of dividing by 11 and 12. The mean division ratio is 11.7, which results in an output frequency 81.9 MHz. The exact division ratio can be controlled by the beat-frequency (MCXO-principle).

Dividing the 81.9 MHz clock by 65 produces the transmitter code-clock (r_{chip}) and a 63-divider produces the input clock to the second loop. This loop is a frequency multiplication loop. The input frequency is derived from the 81.9 MHz clock by dividing that frequency by 63 (1.3 MHz). In the feed-back loop the signal is divided by 32. The output-frequency is 41.6 MHz (processor-clock). From the outputs of the 32-divider, it is possible to generate the other clock-signals with their appropriate phases such as the receiver code-clock and the symbol clock.

3.5 Conclusion

In this chapter a system definition for a non-cellular wireless communication system is formulated. It turned out that different user demands lead to contradictory technical solutions. Evaluation of the existing trade-offs in those situations resulted in effective compromises. This process applied to the WISSCE-example resulted in a system-specification which is summarized in table 3.2.

	CDMA technique	DS/FH (1 symbol per hop)
	kind of data modulation	16-MFSK
N_{DS}	direct-sequence code length	63
N_{FH}	frequency-hopping code length	7
Δ_{FH}	frequency-hopping spacing	1260 kHz
r_{symb}	symbol-rate	20,000 symbols/s
Δ_{FSK}	MFSK-spacing	20 kHz
r_d	data-rate	80 kbit/s
r_c	code-rate	1260 kchip/s
r_{FH}	hopping-rate	20,000 hops/s
	initiator transmit band	2445-2455 MHz
	initiator receive band	2465-2475 MHz
$CK_{sampling}$	sampling technique/frequency	quadrature, 5.2 MHz
$f_{sampler\text{-}in}$	input center-frequency to sampler	10.4 MHz
CK_{DDS}	DDS input-clock	81.9 MHz
$CK_{processor}$	processor clock	41.6 MHz

Table 3.2. overview of system parameters

Concerning the multiple-access technique, a high processing gain of direct-sequence is combined with reduction of the near-far effect due to frequency hopping. The resulting concept so combines the best of two worlds while cancelling their shortcomings.

To generate the required clock-signals in the transceiver, a single time-reference is mandatory. We saw that a clocking-circuitry consisting of two phase lock loops and a number of dividers is capable of generating all necessary clocks.

Chapter 4

PERFORMANCE ANALYSIS

Contents

4.1 Introduction **43**

4.2 Degradation of the data detection SNR **44**

4.2.1 Influences of non-ideal despreading 44

4.2.2 Influences of Multi-Access Interference 45

4.3 Relation between SNR-in and the BER-performance **50**

4.3.1 Performance analysis in an additive Gaussian noise channel. 50

4.3.2 Performance analysis in a multi-path channel 53

4.3.3 Conclusions 57

4.4 Code selection **57**

4.4.1 Choosing a PN-CODE 58

4.4.2 Choosing an FH-SEQUENCE 63

4.5 Conclusion **66**

4.1 Introduction

The large number of factors that determine the sensitivity of a receiver can be split into two categories: front-end and digital baseband processing issues. This chapter concentrates on the latter: the relation between the bit error rate (BER) and the signal to noise ratio at the input of the sampler (input-SNR). Once this relation is known, requirements on the front-end architecture and the received signal strength can be formulated.

To incorporate the important issues of multi-access interference and non-ideal despreading, first the degradation of the input-SNR due to these factors will be addressed. Thereafter the relation between the BER and the input-SNR for a Gaussian Noise channel and a multi-path channel is analyzed.

For the BER-analysis, PN-CODES will be assumed to be random. This approach enables an analysis and is commonly used [Pur77, Wan91, Ger85, Tur84]. In practice however, the codes will *not* be completely random. For this reason, section 4.4 addresses the code selection process.

4.2 Degradation of the data detection SNR

A number of system properties cause a reduction of the SNR level at the input of the analog to digital conversion. Both non-ideal despreading and multi-access interference degrade the input-SNR.

4.2.1 Influences of non-ideal despreading

In reality the despreading operation will not be perfect for two reasons: a misalignment in time between the incoming signal and the local code-generator will exist and an input filter is present which affects the received signal.

- **Timing-misalignment** is due to the residual error after code-synchronization. In steady-state operation this error will be small. During code-acquisition however the chip-misalignment can be typically up to a quarter of a chip-period. The autocorrelation-value for small code-misalignments can be written as:

$$R_{PN}(\tau) \triangleq \overline{c(t)\,c(t+\tau)} = \begin{cases} 1-\frac{|\tau|}{T_c} & |\tau| \le T_c \\ 0 & |\tau| > T_c \end{cases} \tag{4.1}$$

  where τ is the misalignment (timing-error) and T_c is equal to a chip-period. A power correction factor to account for this timing-error is:

$$L_t(\tau) = (R_{PN}(\tau))^2 \ . \tag{4.2}$$

  During acquisition this loss may reach 3 dB.

- **Filtering effects:** the input-filter not only removes the signals in adjacent channel, but also suppresses the side-lobs of the intended signal. The power reduction factor can be calculated by the formula:

$$L_f = \int_{-\infty}^{\infty} S(f)\,|H(j2\pi f)|^2\,df \tag{4.3}$$

where $S(f)$ is the power spectral density of the input-signal and $H(j2\pi f)$ the transfer function of the input filter. A practical situation is illustrated in figure 4.1, where a second order butterworth filter is used with a bandwidth equal to the chip-rate. The power-loss in this example is 0.5 dB.

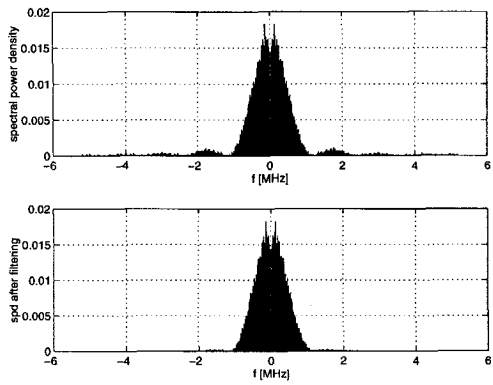


Figure 4.1. Effects of input-filtering illustrated

As filtering affects the shape of the autocorrelation function, the two effects mentioned in this section are not independent and cannot be added. Actually there is again a trade-off: a narrow filter smoothes the autocorrelation curve, but also reduces the output noise power.

4.2.2 Influences of Multi-Access Interference

Multi-access is both the essence and the limitation of WISSCE. The limit on the maximal number of users results from the amount of interference present in the channel and the desired BER.

Multi-access (MA) interference is present when more than one transmitter is active at the same time. A common way to incorporate multi-access interference in the BER analysis is by modelling this interference as a Gaussian noise component [Pur77, WM92, Ger85]. This approach however assumes that there are numerous interferers which are all received with equal power.

As WISSCE applies a non-cellular transmission concept, the received signal from all users cannot be kept constant by means of power control. To create a handle on this problem, two groups of interferers are distinguised:

1. A group of interferers relatively far from the receiver (far-interference). The interference caused by this group in total will be translated to a Gaussian noise component following the usual approach. Adding a Gaussian noise component results in a reduction of the input-SNR.

2. A small group of interferers close to the receiver responsible for the near-interference. This interference cannot be modelled as a Gaussian noise component. Too many of these interferers can block communication links.

There is no clear separation between the two categories. A rule of thumb can be that as long as the signal-power from a user after despreading exceeds the signal-power from the intended user, that user belongs to the category of near-users.

4.2.2.1 Far interference

This kind of interference is caused by a fairly large group of transmitters relatively far from the reference transmitter. The analysis of this interference is based on concepts from [Ger85, Ger86, Pur77, PS77] adapted for WISSCE. If the number of interferers is large, the pseudo-random noise sequences can be assumed to be random. This assumption allows us to derive the variance of the sum of all the far-interference [Roe77]. This variance is equal to the far-interference power and therefore gives the SNR degradation due to far MA-interference. This is illustrated by the following equation for the total detected in energy of a symbol at time n:

$$Z(n) = B_{input} T_s N_0/2 + E_s + S\, T_s \mathrm{Var}\left[\sum_{k \neq i} I_{k,i}(n)\right] . \tag{4.4}$$

S is the power of the received desired signal and T_s represents the time duration of a symbol period. The first term ($B_{input} N_0/2$) at the right hand side represents noise energy that passes the predetection filter during a symbol time. N_0 is the single sided noise spectral power density [Pro89, p.156]. The second term is the desired energy term (energy per symbol):

$$E_s = S\, T_s . \tag{4.5}$$

$I_{k,i}(n)$ denotes the amplitude ratio between multi-access interference (user k) and the user signal (user i) during the n^{th} symbol. The variance of this term is multiplied with the energy per symbol to obtain the interference energy per symbol.

$$I_{k,i}(n) = T_s^{-1}\left[\delta(f_i^n, f_k^n)\; R_{i,k}(\tau_k) + \delta\left(f_i^n, f_k^{n+1}\right)\; \widehat{R}_{i,k}(\tau_k)\right] \tag{4.6}$$

where the Kronecker function $\delta(\cdot,\cdot)$ is defined by $\delta(u,v) = 1$, if $u = v$ and $\delta(u,v) = 0$ otherwise. f_i^n is the frequency hopping code for user i during symbol period n. τ_k is the relative time-delay between the received signals from the intended user

and the interfering user k. $R_{k,i}(\tau)$ and $\widehat{R}_{k,i}(\tau)$ are partial cross-correlation terms [SP80].

As a result the first term corresponds to a hit of the users k and i during the earlier part of symbol period n, while the second term corresponds to a hit during the later part of this symbol period. It is also leads to the conclusion that the partial cross-correlation functions $R_{k,i}(\tau)$ and $\widehat{R}_{k,i}(\tau)$ have only influence if there occurs a hit.

It is clear that the MA-interference could be divided into two partial interference terms (as illustrated in figure 4.2). In this figure "code 1" of the reference user and "code k" of an interfering user are partially overlapping. One term is responsible for the interference in the first part of symbol period and the other term for the second part.

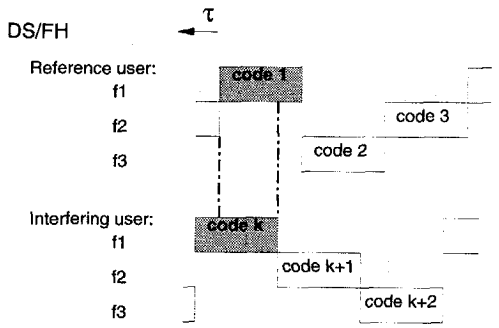


Figure 4.2. Partial interference in DS /FH spreading scheme

From the figure it is also clear that in a hybrid DS/FH system, there is at most 1 (partial) hit per symbol-period. This gives:

$$\delta(f_i^n, f_k^n) \cdot \delta\left(f_i^n, f_k^{n+1}\right) = 0 \quad \forall n \in \mathbb{N} \; . \tag{4.7}$$

Taking this into account (4.6) can be written as:

$$I_{k,i} = \left[\delta(f_i^n, f_k^n) + \delta\left(f_i^n, f_k^{n+1}\right) \right] \cdot T_s^{-1} R'_{k,i}(\tau_k) \tag{4.8}$$

in which $R'_{k,i}(\tau)$ is defined as:

$$R'_{k,i}(\tau_k) = \int_{\tau_{k1}}^{\tau_{k2}} c_k(t+\tau_k) c_i(t) dt \tag{4.9}$$

c_k expresses the k^{th} chip of a PN-CODE, while τ_{k1} and τ_{k2} are related to τ_k in the following way:

$$\tau_{k1} = \begin{cases} 0, & \tau_k \leq 0 \\ \tau_k, & \tau_k > 0 \end{cases} \qquad \tau_{k2} = \begin{cases} \tau_k + T_s, & \tau_k \leq 0 \\ T_s, & \tau_k > 0. \end{cases}$$

This integral (4.9) is illustrated in figure 4.3. In this figure the sequences i and k overlap for 5 1/3 chip period.

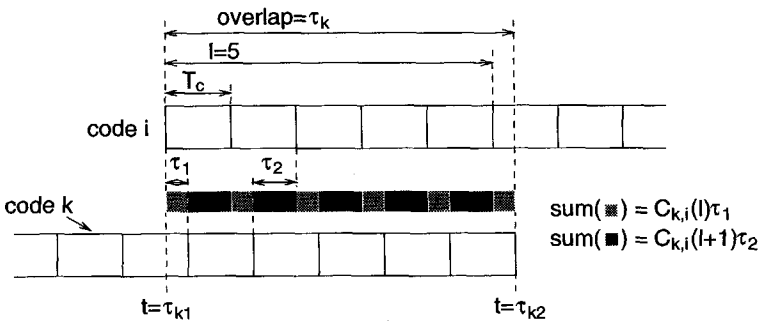


Figure 4.3. MA-interference illustrated

An integer l is now defined which is equal to the number of complete overlapping chips ($-T_s \leq lT_c \leq \tau_k \leq (l+1)T_c \leq T_s$, $l \in \mathbf{Z}$). This number is negative if the later part of code i overlaps with code k. In the example of figure 4.3 l is equal to 5. The integral in equation (4.9) can be written as:

$$R'_{k,i}(\tau_k) = C_{k,i}(l) \cdot [(l+1)T_c - \tau_k] + C_{k,i}(l+1) \cdot [\tau_k - lT_c] \tag{4.10}$$

where the first term corresponds to the summation of all "dark" overlaps in figure 4.3 and the second term corresponds to sum of all "light" overlaps. $C_{k,i}(l)$ is the discrete aperiodic cross correlation function [PS77]:

$$C_{k,i}(l) = \begin{cases} \sum_{j=0}^{N_{DS}-1-l} c_k[j]\, c_i[j+l], & 0 \leq l \leq N_{DS}-1 \\ \sum_{j=0}^{N_{DS}-1+l} c_k[j-l] c_i[j], & 1-N_{DS} \leq l < 0 \\ 0, & |l| \geq N_{DS} \end{cases} \tag{4.11}$$

where the usage of square-braces expresses the time-discreteness of the PN-CODEs. If a hit occurs between the intended and interfering user, the variance of the multi-access interference term from (4.8) is:

$$\mathrm{Var}\left[\, I_{k,i}\right] = \sigma_{k,i}^2 = \frac{1}{2\,T_s^3}\int_{-T_s}^{T_s} R'^2_{k,i}(\tau_k)\, d\tau_k \,. \tag{4.12}$$

The right hand side can be written as [Roe77, Ger86]:

$$\frac{1}{2\,T_s^3} E\left[\int_{-T_s}^{T_s} R'^2_{k,i}(\tau) d\tau\right] = \frac{1}{2\,T_s^3}\frac{T_c^3}{3}\, E\left[\rho_{k,i}\right] = \frac{1}{6N_{\mathrm{DS}}}\, E\left[\rho_{k,i}\right] \tag{4.13}$$

in which:

$$\rho_{k,i} = \sum_{l=1-N_{\mathrm{DS}}}^{N_{\mathrm{DS}}-1} \left\{ C_{k,i}^2(l) + C_{k,i}(l) C_{k,i}(l+1) + C_{k,i}^2(l+1) \right\} . \tag{4.14}$$

At this point we will make use of the common assumption that the applied sequences are random [Roe77, Ger85, Ger86, Pur77, PS77]. A discussion on the selection of PN-CODEs is given in section 4.4. After calculation follows for random sequences:

$$\mathrm{E}\left[\rho_{k,i}\right] = 2N_{\mathrm{DS}}^2 - 1 \approx 2N_{\mathrm{DS}}^2 \,. \tag{4.15}$$

This yields:

$$E\left[\sigma_{k,i}^2\right] = \frac{1}{3\,N_{\mathrm{DS}}} \,. \tag{4.16}$$

The power in the multi-access far-interference term:

$$I_{FAR} \;=\; E\left[\, p_{\mathrm{hit}}(k, K_{FAR})\right] \frac{S\,T_s}{3\,N_{\mathrm{DS}}} \;=\; E_s\, \frac{p_{\mathrm{hit}}(k, K_{\mathrm{FAR}})}{3\,N_{\mathrm{DS}}} \tag{4.17}$$

in which $E\left\{p_{\mathrm{hit}}(k, K_{\mathrm{FAR}})\right\}$ is the mean number of far-interferers that "hit" the reference user in the used frequency-hop channel. E_s denotes the energy per symbol: ST_s. The hit-probability will be addressed during the discussion of the FH-SEQUENCEs in section 4.4.2.

Concerning the MA-interference the conclusion can be drawn that every far-user that hits the reference user during a FH-period contributes maximally $1/3N_{\mathrm{DS}}$ times the desired signal power. To enable the analysis the power of all interfering users was assumed to be equal to the desired power so equation (4.17) provides a number on the interference for a worst case scenario.

4.2.2.2 Near interference

Near interference is more problematic in the sense that it can completely block frequency-hopping channels. When too large a number of such near-users are active, communication is not possible. For this reason the number of near-interferers allowed in the system is limited. The hit-probability (P_{hit}), see section 4.4.2) is determinative for the performance.

The number of near-users that can be allowed is also determined by the kind of error-correction that is applied. During system-specification it was required at least two frequency-hopping channels have no near-interference (see page 31). The system specification shows that N_{FH} is equal to seven which means that it is likely that there are three near-interference free FH-channels. The synchronizaiton algorithm also makes use of this property (see section 5.3.3 on page 86).

Concluding: it is difficult to give a quantitative analysis of the near-user interference. The amount of interference is to a large extent dependent on the momentary situation. The only thing we can say is that if specifications are met, there will be at least three FH-channels without near-interference.

4.2.2.3 Conclusion

To enable multi-access interference analysis, the interferers were split into two groups: far-interferers and near-interferers. The interference of the first category of users can be translated into a Gaussian Noise term that deteriorates the input-SNR. The average power that every far-interferer contributes is equal to $1/3N_{\text{DS}}$ times the energy per symbol. The second category of interferers is more problematic: a worst case scenario is that only three FH-channels without near-interference.

4.3 Relation between SNR-in and the BER-performance

The bit error rate performance is an important quality in communication systems. In this section we will analyze the WISSCE receiver to find a relation between the bit error rate (BER) and the input SNR to the analog-to-digital conversion stage. This relation needs to be known to formulate requirements for the front-end and the received signal strength.

The BER-analysis is divided into two steps: first the analysis is done for a situation in which no multi-path channel is present (additive Gaussian noise channel). Then the same is done for a multi-path environment.

4.3.1 Performance analysis in an additive Gaussian noise channel.

As a reference SNR-level the SNR value at the input of the data-detector is used (predetection or input-SNR) where the bandwidth is 1.26 MHz (equal to the chip-

rate). The relation between the single-sided noise power (σ_n^2), the two-sided noise spectral density ($N_0/2$) and the bandwidth is [Pro89, p.156]:

$$\sigma_n^2 = \frac{N_0 B_{\text{input}}}{2} .$$

The received signal in a single-user, multi-path free environment is:

$$r(t) = \sqrt{S}\, c(t) \exp[j\,(d(t)\Delta_{FSK} + \omega_c)\,t + \theta] + n(t) \tag{4.18}$$

where:

S = received signal power
$d(t)$ = data symbol $\in \{-8,-7,\ldots,7,8\}\backslash\{0\}$
Δ_{FSK} = MFSK-spacing in radial frequency
$c(t)$ = binary pseudo-random noise sequence
ω_c = carrier radial frequency
θ = arbitrary phase
$n(t)$ = noise signal with a two-sided noise spectral density of $N_0/2$.

After DS-despreading the PN-CODE c(t) is removed from the signal by the local code, for the despread signal $x(t)$ yields:

$$x(t) = \sqrt{S} \exp[j\,(d(t)\Delta_{FSK} + \omega_c)\,t + \theta] + n(t) . \tag{4.19}$$

The influence of the despreading operation on the Gaussian noise is small and will therefore be neglected. During MFSK-detection $x(t)$ is, in the M MFSK-channels, correlated with the expected signal for the appropriate MFSK-channel. These expected signals can be written as:

$$v_m(t) = \exp[j(m\Delta_{FSK} + \omega_c)t] . \tag{4.20}$$

The decision variable in the m^{th} channel during the n^{th} symbol-period is via square-law detection found in the following way:

$$Z_m(n) = \left| \int_{t=nT_s}^{(n+1)T_s} x(t) \cdot v_m(t) dt \right|^2 . \tag{4.21}$$

Following the analysis in [Pro89, p.295], the decision variable in the MFSK-CHANNEL that contains signal (channel i) will have a non-central chi-square distribution:

$$p_{s(i)}(u) = \begin{cases} \frac{1}{2\sigma_n^2} \exp\left[-\left(\frac{u}{2\sigma_n^2} + \gamma\right)\right] I_0\left(2\sqrt{\frac{\gamma u}{2\sigma_n^2}}\right) & u \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.22}$$

In which γ is the detection (post-despreading) SNR:

$$\gamma = \frac{E_s}{N_0} = \gamma_p \frac{r_{\text{symb}}}{B_{\text{input}}} .$$

γ_p is the predetection SNR. The factor between the detection and predetection SNR is equal to the DS-processing gain. This factor is equal to 18 dB for a DS-sequence length of 63 ($B_{\text{input}}/r_{\text{symb}}$).

The other channels will contain no signal, the decision-variable will therefore have a central chi-square distribution:

$$p_{\bar{s}(m\neq i)}(u) = \begin{cases} \frac{1}{2\sigma_n^2} \exp\left(-\frac{u}{2\sigma_n^2}\right) & u \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.23}$$

A symbol error will occur if the MFSK-CHANNEL containing the signal (i) contains less energy than at least one of the other channels. The probability on correct detection can be written as:

$$P_{c,symbol}(i) = \int_{u=0}^{\infty} \prod_{\substack{-8\leq m\leq 8 \\ m\neq\{0,i\}}} \left\{ \int_{x=0}^{u} p_{\bar{s}(i)}(x)dx \right\} p_{s(n)}(u)\,du . \tag{4.24}$$

Since $Z_m(n)$ $m \in (-8,-7,\ldots,8)$, $m \neq \{0,i\}$ are statistically independent and identically distributed (4.24) can be written as:

$$\begin{aligned} P_{\text{c, symbol}} &= \int_{u=0}^{\infty} \left[1 - e^{-\frac{u}{2\sigma_n^2}}\right]^{15} p_{s(n)}(u)\,du \\ &= \sum_{m=0}^{15} (-1)^m \binom{15}{m} \frac{e^{-\gamma m/(m+1)}}{m+1} . \end{aligned} \tag{4.25}$$

And the bit error rate can be found by (see also [Pro89, p.250]:

$$P_{\text{e, bit}} = \frac{8}{15} \sum_{m=1}^{15} (-1)^{m+1} \binom{15}{m} \frac{e^{-\gamma m/(m+1)}}{m+1} . \tag{4.26}$$

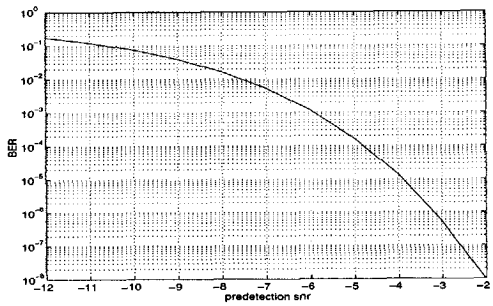


Figure 4.4. BER verses input-SNR

In conclusion we observe that the BER-curve of figure 4.4 is similar to the usual 16-MFSK BER-curve for these detectors [Pro89, p.297]. The difference lays in the x-axis. The figure here has the pre-detection SNR (SNR value at the input of the data-detector) on that axis. At first sight this might seem strange, however during the implementation stage (section 5.2.3) it becomes clear that the pre-detection SNR provides a better reference than the usual E_b/N_0 value.

4.3.2 Performance analysis in a multi-path channel

4.3.2.1 Fading Channels

Multi-path is the effect that the signal received at the receiver does not come as a single "beam" directly from the transmitter. Especially in an in-house environment, the transmitted signal will reflect at several places and will reach the receiver via various paths of different length and consequently with unequal time delays.

In a fading environment the receiver will get the desired signal from different paths, every path having its own propagation delay (τ_n) and attenuation factor (ε_n). If $s(t)$ is a single path signal, the received signal can be written as:

$$r(t) = \sum_n \varepsilon_n(t) s[t - \tau_n(t)] \tag{4.27}$$

where both ε_n and τ_n are functions of time, indicating that the channel is time varying. As both ε_n and the phase (because of τ_n) are randomly changing, $r(t)$ can

be modelled as a random process [Pro89, p.705]. When there are a large number of paths, the central limit theorem can be applied, i.e. $r(t)$ can then be modelled as a complex-valued Gaussian random process. A possible situation would be that signals from a few different paths add destructively. In this way the signal would "fade-out". These fading effects are due to the multi-path characteristics of the channel.

Two important numbers in characterizing the channel are the RMS-delay spread and the doppler-spread. The RMS delay-spread is the range of values of τ for which the channel transfer function is unequal to zero, and is referred to as T_{rms}. The reciprocal of T_{rms} is the coherence bandwidth of the channel [Pro89, p.707]:

$$(\Delta \text{f})_c \approx \frac{1}{T_{\text{rms}}} . \tag{4.28}$$

This bandwidth can be interpreted as the bandwidth over which the channel transfer function does not change: thus two sinusoids with a frequency separation larger than $(\Delta \text{f})_c$ are affected differently by the channel. Typical values for the RMS time delay-spread in an indoor environment at frequencies round 910 MHz are in the range of $50ns$ up to $250ns$ [BMS89]. For the 2.4 GHz band values between $10ns$ and $40ns$ are reported [JP92, HT94, Nik95]. In WISSCE a chip-period takes about $795ns$ which is much longer than T_{rms} as a consequence there is only a single resolvable path.

Where the coherence bandwidth specifies a bandwidth over which the channel does not change, the coherence time specifies a time-period over which the channel stays the same. The coherence time ($(\Delta \text{t})_c$) gives an indication of the speed of the variations of the channel. The reciprocal value is called the *Doppler spread.* Indoor measurements [HP90] show a maximum Doppler spread of 6.1 Hz. Therefore it is safe to assume that the indoor channel will not change within a symbol period.

4.3.2.2 Influences of the fading channel

If the channel does not change significantly during a symbol-period, the received signal is:

$$r(t) = \varepsilon e^{-j\phi} s(t) + n(t) \tag{4.29}$$

in which ε is a constant attenuation factor and ϕ a constant phase-offset. $s(t)$ is the desired signal and $n(t)$ represents an additive Gaussian noise term.

The data-detection algorithm in WISSCE is independent of phase-offsets. Consequently the fading influence lays in the attenuation factor. As a result the detection SNR becomes a random variable.

Several ways exist to describe the indoor communication channel [Has93]. The most straightforward technique is to model the received signal amplitude as being Rayleigh distributed. The physical justification for this is that the received signal only contains multi-path components. In the receiver the amplitude is equal to the square-root of the summation of the squares of the in-phase and quadrature components. As both those components have a Gaussian distribution, the amplitude will be Rayleigh distributed (Rayleigh fading channel). The SNR is proportional to the received power and consequently, the detection-SNR (γ) will have a central chi-square distribution with 2 degrees of freedom:

$$p(\gamma) = \frac{1}{\bar{\gamma}} e^{-\gamma/\bar{\gamma}} \tag{4.30}$$

where $\bar{\gamma}$ is the expectation of γ. This parameter is equal to the non-faded value of γ times the expectation of ε^2.

In an indoor environment however, often a line of sight (LOS) signal path is present [PMK90]. The resulting amplitude will then be Rician distributed (Rician fading channel) and γ will have a non-central chi-square distribution:

$$p(\gamma) = \frac{1}{2\sigma^2} e^{-\frac{(s^2+\gamma)}{2\sigma_\gamma^2}} I_0\left(\sqrt{\frac{\gamma s^2}{\sigma^4}}\right). \tag{4.31}$$

To relate the parameters σ and s to the non-faded SNR, the Rice-factor is used. This factor is the ratio of LOS-signal power to the random-path signal power. The relation can be expressed as:

$$s^2 = \left(\frac{R-1}{R}\bar{\gamma}\right) \tag{4.32a}$$

$$2\sigma^2 = \frac{\bar{\gamma}}{R} \tag{4.32b}$$

here R is the Rice-factor. For $R = 0dB$ (direct-path power is as strong as the multi-path power) the distribution function from (4.31) evaluates to a Rayleigh distribution, while for $R \to \infty$, the distribution becomes one-valued (no-fading situation).

The BER-formula (4.26) becomes:

$$P_{\text{e, bit}} = \frac{8}{15}\sum_{m=1}^{15}(-1)^{m+1}\binom{15}{m}\int_{\gamma=0}^{\infty}\frac{e^{-\gamma m/(m+1)}}{m+1}\,p(\gamma)\,d\gamma$$
$$= \frac{8}{15}\sum_{m=1}^{15}\frac{(-1)^{m+1}R}{\bar{\gamma}(m+1)}\binom{15}{m}\int_{\gamma=0}^{\infty}\exp\left[1-\left(R\left(1+\frac{\gamma}{\bar{\gamma}}\right)+\frac{\gamma m}{m+1}\right)\right] \quad (4.33)$$
$$\cdot I_0\left(2\sqrt{\frac{\gamma(R-1)R}{\bar{\gamma}}}\right)d\gamma.$$

Performing the integration yields:

$$P_{\text{e, bit}} = \frac{8}{15}\sum_{m=1}^{15}(-1)^{m+1}\binom{15}{m}\frac{R}{\bar{\gamma}m+R+mR}\exp\left[-\left(\frac{\bar{\gamma}m(R-1)}{\bar{\gamma}m+R+mR}\right)\right]. \quad (4.34)$$

From this formula it is clear that for $R = 0dB$ the performance over a Rayleigh fading channel is obtained (see also [Pro89, p.716]). For $R \to \infty$ the fading effects disappear (compare formula (4.26)). This is also shown in figure 4.5 where lines are plotted for $R = 0dB$ (Rayleigh), $R = 6.8dB$, $R = 11dB$ and $R = \infty$ (no fading). Typical values of this parameter for indoor channels at the appropriate frequency are: 6.8 dB and 11 dB [BMS89, Bul87].

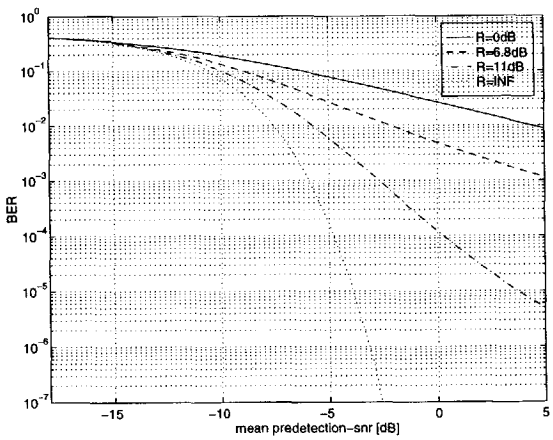


Figure 4.5. BER in a fading-environment

The conclusion of the fading analysis in this section is threefold:

- Indoor fading characteristics cannot be caught in a typical description; it is therefore impossible to give an exact analysis.
- The impact of a number of typical fading situations is calculated and is presented in figure 4.5.
- If a user manages to situate him/herself at a very nasty location (in a deep fade), it is possible that communication gets impossible. The only solution in such a situation would be changing the location of the antenna. A more elegant solution is adding a second antenna. This option however is discarded because of considerably higher costs.

4.3.3 Conclusions

In this section a relation between the data detection input SNR and the BER was made for both an additive Gaussian noise channel and a multi-path channel.

A difference with a usual BER-analysis is the usage of the data-detection input SNR as a reference value. In WISSCE the bandwidth at this stage in the receiver is about 1.3 MHz. As a consequence, proper SNR values are below 0 dB. We did this to enable easy incorporation of implementation properties in the next chapter.

It appeared that WISSCE has to operate in a slowly fading channel. It is however difficult to catch the channel properties in a typical description. This is illustrated in figure 4.5, the different curves show the relation between SNR and BER for various values of the strength of the line-of-sight path.

4.4 Code selection

The spreading of the data signal in a CDMA system is done by applying a code, independent of the data-signal. Code-selection has a large impact on the performance of the system. On the one hand the longer the code, the higher the processing gain which enables us to allow more users in the system. On the other hand a larger processing gain implies the usage of more bandwidth. Another important property of the codes is the cross-correlation. If the codes which are used are not completely orthogonal, the cross-correlation factor is unequal to zero. In this situation the different users are interferers to each other, hence the near-far problem appears.

During the BER-analysis in the previous section the usual approach to assume PN-CODES to be random [Roe77] was applied. In practical situations however, this assumption needs to be considered more carefully. This section proposes a strategy to select codes that are "'close to random".

WISSCE is a hybrid DS/FH-system where both a code for direct-sequence spreading (a PN-CODE) and a code for frequency-hopping spreading (an FH-SEQUENCE) has to be selected. First the selection of PN-CODEs will be discussed.

4.4.1 Choosing a PN-CODE

4.4.1.1 Choosing a code-family

A PN-CODE used for DS-spreading consists of N_{DS} units, called chips. These chips can have 2 values: -1/1 (polar) or 0/1. In the following polar bit-sequences are used unless stated otherwise. As every data symbol is combined with a single complete PN-CODE, the DS processing gain is equal to the code-length. To be usable for direct-sequence spreading, a PN-CODE must meet the following constraints:

- The PN-CODEs are 2-leveled (bit-sequences).
- The codes must have a sharp (1-chip wide) autocorrelation peak to enable code-synchronization and to achieve equal spreading over the whole frequency-band.
- The codes must have low cross-correlation values. The lower this cross-correlation, the more users can be allowed in the system. This holds for both full-code and partial-code overlap. The latter because in most situations there will not be a full-period correlation of two codes and it is more likely that codes will only correlate partially (due to random-access nature).
- To avoid a DC-component in the spread signal, the codes should be "balanced": the difference between ones and zeros in the code must be 1.

Codes that can be found in practical DS-systems are: Walsh-Hadamard codes, M-sequences, Gold-codes and Kasami-codes. Walsh sequences [Bea75] are orthogonal while the other sequences show cross-correlation values unequal to zero [Gol67, Roe77, SP80].

Walsh Hadamard codes Walsh-sequences have the advantage to be orthogonal, in this way we should get rid of any multi-access interference. There are however a number of drawbacks:

- The codes do not have a single, narrow autocorrelation peak. As a consequence code-synchronization becomes difficult.
- The spreading is not very efficient: the energy is only spread over a small number of discrete frequency-components (figure 4.6).
- Although the full-sequence cross-correlation is identically zero, this does not hold for partial-sequence cross-correlation function. The consequence is that the advantage of using orthogonal codes is lost when all users are not synchronized to a single time base.

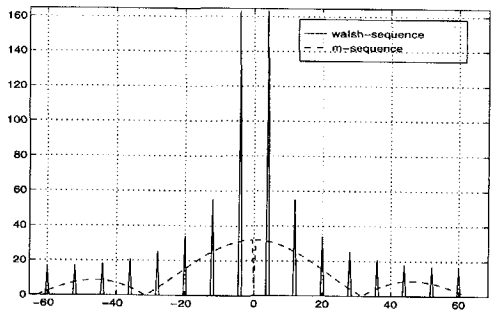


Figure 4.6. Frequency-domain comparison of a Walsh and an M-sequence

- Orthogonality is also affected by channel properties like multi-path. In practical systems equalization is applied to recover the original signal.

These drawbacks make Walsh-sequences not suitable for a system like WISSCE. Systems in which Walsh-sequences are applied are for instance multi-carrier CDMA [YLF94] and the cellular CDMA system IS-95 [Qua92]. Both systems are based on a cellular concept in which all users (and so all interferers) are synchronized with each other.

Shift-Register sequences are not orthogonal, but they do have a narrow auto-correlation peak. The name already implies that the codes can be created using a shift-register with feedback-taps. By using a single shift-register, maximum length sequences (M-sequences) can be obtained. Such sequences can be created by applying a single shift-register with a number of specially selected feedback-taps. If the shift-register size is n then the length of the code is equal to $2^n - 1$. The number of possible codes is dependent on the number of possible sets of feedback-taps that produce an M-sequence. These sequences have a number of special properties. Some of them that will be considered in the code selection process are mentioned here.

- M-sequences are balanced: the difference between number of ones and minus ones is only 1.
- The spectrum of an M-sequence has a sync^2-envelope. In figure 4.6 the spectra of a Walsh-sequence of length 64 and an M-sequence of length 63 are shown, both sequences contain (almost) the same power. The figure shows that applying an M-sequence better distributes the power over the whole available frequency range.

- The shift-and-add property can be formulated as follows:

$$T^k u = T^i u \cdot T^j u \tag{4.35}$$

  here T denotes a delay of one chip and u is an M-sequence. By combining two shifts of this sequence (relative shifts i and j) the same M-sequence is obtained, yet with another relative shift.

- The auto-correlation function is two-valued:

$$R_u(\tau) = \begin{cases} N & \tau = kN \\ -1 & \tau \neq kN \end{cases}$$

  where k is an integer value, and τ is the relative shift as a multiple of a chip-period.

- There is no general formula for the cross-correlation of two M-sequences, only some rules can be formulated [Roe77].

- A so called "preferred pair" is a combination of M-sequences for which the cross-correlation only shows 3 different values: -1, $-2^{\lfloor (n+2)/2 \rfloor} - 1$ and $2^{\lfloor (n+2)/2 \rfloor} + 1$. There do not exist preferred pairs for shift-registers with a length equal to $4k$ where k is an integer.

The product of two M-sequences which form a "preferred pair" leads to a so-called Gold-code. By giving one of the codes a delay with respect to the other code, we can get different sequences. The number of sequences that are available is $2^n + 1$ (the two M-sequences alone, and a combination with $2^n - 1$ different shift positions). The maximum full-code cross-correlation has a value of $2^{\lfloor (n+2)/2 \rfloor} + 1$.

In WISSCE the PN-CODE-length is chosen to be 63 (see section 3.2.3), consequently $n = 6$. For this situation there are 65 different Gold-codes available and the maximum full-code cross-correlation is 17.

If a Gold-code is combined with a decimated version of one of the 2 M-sequences that form this Gold-code a "Kasami-code from the large set" is obtained. Such a code can then be formulated as follows:

$$c = u \cdot T^k v \cdot T^m w$$

where u and v form a preferred pair of M-sequences of length $N_{DS} = 2^n - 1$ (n even). w is an M-sequence resulting after decimation the v-code with a value $2^{n/2} + 1$. k is the offset of the v-code with respect to the u-code and m is the offset of the w-code with respect to the u-code. Offsets are relative to the state in which all register elements contain a one or a minus one (all-ones state). In the large set of Kasami-codes a number of special subsets can be observed:

- The two M-sequences that are used to create the PN-CODE
- The Gold-codes that can be created using these M-sequences
- The small set of Kasami-codes can be obtained by combining one M-sequence with the decimated version of itself, so leaving out the other M-sequence.

Kasami-codes have the same correlation properties as Gold-codes. The difference is the number of codes that can be created. For the large set of Kasami-codes this number is equal to $2^{n/2}(2^n+1)$. Choosing n equal to 6 as in WISSCE gives 520 possible codes. As the number of codes determines the number of different code addresses that can be created, the large set of Kasami-codes is used as a code-set in WISSCE. A large code-set also enables us to select those codes which show good cross-correlation characteristics.

In WISSCE the DS code-length (N_{DS}) is chosen to be 63, this implies using shift-registers of length 6. Two codes that form a preferred pair are: $M(6,1)$ (feedbacks from the 6^{th} and 1^{st} feedback taps) and $M(6,5,2,1)$. In the following we will use the notation of (4.36). Here n is even and equal to the length of the shift-registers used to create u and v, u is the M-sequence with feedbacks (6,1) and v is the M-sequence with feedbacks (6,5,2,1). When decimating $M(6,1)$ sequence w is obtained, w is also an M-sequence: $M(3,2)$. The otherwise meaningless values for k and m are used to denote special subsets.

$$kas(n,k,m) = \begin{cases} u \cdot T^k v \cdot T^m w, & ;0 \le k \le 2^n-2, 0 \le m \le 2^{n/2}-2 \\ u \cdot T^m w, \text{ (small set)} & ;k = 2^n-1, 0 \le m \le 2^{n/2}-2 \\ v \cdot T^m w, & ;k = 2^n, 0 \le m \le 2^{n/2}-2 \\ u \cdot T^k v, \text{ (Gold codes)} & ;0 \le k \le 2^n-2, m = 2^{n/2}-1 \\ v, \text{ (M-sequence)} & ;k = 2^n-1, m = 2^{n/2}-1 \\ u, \text{ (M-sequence)} & ;k = 2^n, m = 2^{n/2}-1\,. \end{cases} \tag{4.36}$$

4.4.1.2 Choosing a code-set with good cross-correlation properties

Before starting the selection process, first the size of the code-set will be reduced by adding the constraint that the codes must be balanced to avoid a DC-offset. The initial code-set size now reduces from 520 to 241. The 241 balanced codes are used as a candidate set to the further code selection process. Having chosen a candidate code-set, we have to find a sub-set of this code-family which has good cross-correlation properties. As mentioned before partial sequence correlation is more important than full-sequence correlation. The reason for this is that all users in the system are asynchronous so the sequences will only overlap partially.

We saw that the multiple-access interference in a CDMA-system depends on the $\rho_{k,i}$-factor of two sequences k and i (equation 4.13 on page 49). So this parameter

can be used as a criterion to select a code-set. The required computational power however, to find a code-set with a bounded value of the $\rho_{k,i}$-factor among all codes in that set, goes exponentially with the size of the initial candidate code-set. For a code-set size of 241 the calculation of a code-set with bounded $\rho_{k,i}$-factor is therefore infeasible. Instead an algorithm as described in [EKB87] will be used. Following this concept the codes are ordered in increasing value of a cost-function. Assigning codes starts at the top of this list. In this way the code-set is used which is likely to have the best correlation properties. The cost-function is defined as:

$$R_i^{(K_{\text{set}})} = \sum_{\substack{k=1 \\ k \neq i}}^{K_{\text{set}}} \rho_{k,i} \tag{4.37}$$

where K_{set} is equal to the size of the code-set (in this case 241). The resulting sequence of codes can be found in appendix A.

4.4.1.3 Implementation issues

Kasami code sequences can be implemented by multiplication of the outputs of 3 shift-registers (u, v and w) with proper feedbacks. Two shift-registers have length n (in WISSCE equal to 6) and form a "preferred pair", while the third sequence (w) resulted from decimation of the first sequence (u). The last shift-register has a length of 3. All three shift-registers create M-sequences.

To obtain all possible Kasami-codes, the 3 M-sequences should have different relative shifts with respect to each other. Two "shift values" play a role: the relative shift of the v-sequence and of the w-sequence, see (4.36).

As a relative shift of for instance 35 is costly to implement (we would need $35-6=29$ extra delay-elements), the "shift-and-add" property can be applied (see (4.35)) to obtain the desired relative shift. It appears that all relative shifts can be created by using only 6 respectively 3 delayed versions of the v and w code. This is illustrated in figure 4.7, an additional requirement is that the sequences have their all-zero state at the same shift. In the figure the shift-parameters are referred to as m' and k' as there is a translation step between the relative shift values and the tap-numbers that are required to obtain those shifts. A description of the translation step can be also found in appendix A.

4.4.1.4 Conclusions

The code-selection process can be summarized in the following way:

- We found that non-orthogonal shift-register sequences are more suitable in a system like WISSCE than orthogonal sequences like for instance Walsh-Hadamard sequences.

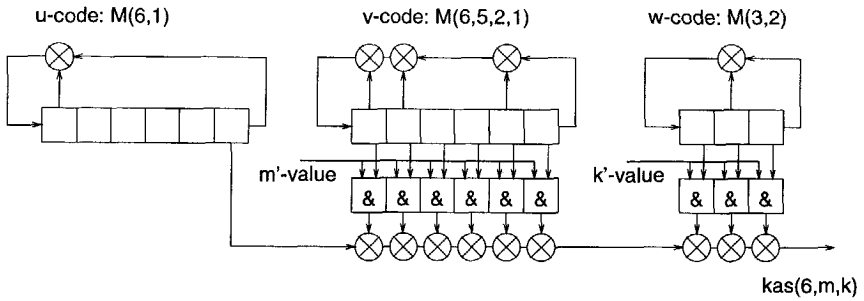


Figure 4.7. Kasami-code generator scheme

- Using the large-set of Kasami-codes with length 63 provides us with 520 different sequences with bounded cross correlation values.
- By adding a requirement that only using "balanced codes" we make sure that there does not appear a non-suppressed carrier in the resulting spread output spectrum.
- The 241 remaining "balanced codes" are put in order on basis of cross-correlation properties. In this way the "best" addresses can be assigned first.
- It is easily possible to generate all different Kasami-codes by applying the shift-and-add property.

4.4.2 Choosing an FH-SEQUENCE

In WISSCE FH-spreading is applied to reduce the problems due to the Near-Far effect. To keep the necessary bandwidth within limits and the frequency synthesizers realizable, the FH-SEQUENCE-length (N_{FH}) is chosen equal to 7. For this rather short code-length it is possible to perform a manual extensive search to find possible sequences. As a boundary condition to the sequences we specified that a near-interferer should maximally have 2 hits during a single FH-SEQUENCE. A possible code-set existing of 6 sequences is given in table 4.1.

A more general rule can be found in [MT92]. Hyperbolic codes which have the above mentioned property can be formed in the following way: if p is a prime, and there are p frequency-hop channels there exist $p-1$ codes of length $p-1$ which have in case of an asynchronous interferer at most 2 hits during a complete FH-SEQUENCE. We choose the first alternative because that provides longer sequences.

0	1	2	3	4	5	6
0	2	4	6	1	3	5
0	3	6	2	5	1	4
0	4	1	5	2	6	3
0	5	3	1	6	4	2
0	6	5	4	3	2	1

Table 4.1. FH-SEQUENCES

The hit-probability is the probability that a number of interfering users are transmitting in the same frequency-hop channel as the reference user. This probability will be referred to as $p_h(K)$, where K is the total number of active users. This probability is dependent on a number of system characteristics like:

- The number of frequency-hopping channels available
- The number of active users
- The kind of frequency-hopping sequences used and their length.

As there are N_{FH} FH-channels, the chance that two users "hit" is $1/N_{\text{FH}}$. The FH-sequences are selected on their property to have at most two partial hits which leads to the following hit-probability [Gla92]:

$$P_h = \frac{2}{N_{\text{FH}}} . \tag{4.38}$$

This formula is only valid if random hopping is applied which is true as aa user can start transmitting at any arbitrarily moment.

The chance that 2 users have the same FH-SEQUENCE is:

$$P_{\text{h; FH}} = \frac{1}{N_{\text{FH}} - 1} . \tag{4.39}$$

Two situations can be distinguished:

1. *two users having the same* FH-SEQUENCE
   In this situations the users will hit always or hit not at all. The first probability is equal to the hit-probability (4.38).

2. *two users having a different* FH-SEQUENCE
   If two users have a different sequence, they will either have 2 partial hits or a single full hit. So the probability that two users will hit during a "hop-period" is: P_h/N_{FH}.

We obtain for the complete hit probability:

$$P_{\mathrm{hit}} = \frac{2}{N_{\mathrm{FH}}(N_{\mathrm{FH}}-1)} + \frac{2\,(N_{\mathrm{FH}}-2)}{N_{\mathrm{FH}}^2(N_{\mathrm{FH}}-1)}\,. \tag{4.40}$$

However, the probability of interest is the chance that a certain number (k) of interfering users are present in the same frequency-hop channel as the intended user. This probability is given by:

$$p_{\mathrm{hit}}(k,K) = \binom{K-1}{k} P_{\mathrm{hit}}^{k}\,(1-P_{\mathrm{hit}})^{K-1-k} \tag{4.41}$$

where K is the total number of active users to be considered. In figure 4.8 the expectation of (4.41) is shown as a function of K. In this example N_{FH} is equal to 7. The values from this plot can be used in formula (4.17) on page 49.

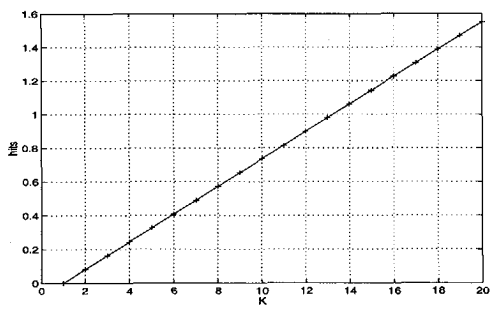


Figure 4.8. $E\left\{p_{\mathrm{hit}}(k,K)\right\}$ as a function of K active users

From the figure above it is clear that even when quite a number of near-users is present, the "mean" value of the number of hits is limited to reasonable values. The WISSCE receiver should be able to operate under situations where more than 2 FH-channels are blocked. From the figure it appears that even if there are 20 near users, the mean number of hits is about 1.5.

4.5 Conclusion

In this chapter an attempt was made to formulate a requirement concerning the SNR at the input of the pre-sampling filter. From the previous chapter we know that despreading and data detection take place in the digital domain. So to be more specific the SNR requirement holds for the signal before sampling, quantizing, despreading and detecting of the data symbols.

Multi-access abilities are essential to WISSCE. It was shown that a far-user contributes a Gaussian interference energy of about $1/3N_{DS}$ times the energy per symbol. The number of near users allowed in the system is fixed by the number of frequency-hopping channels available. In the multi-access interference analysis we made the common assumption that PN-CODES are random. This is however not completely true in practice. For this reason this chapter also addressed to question of how to select codes in such a way that they are close to random.

Finally from the relation between BER and data-detection input-SNR it appeared that in an unfaded environment an SNR-level of about -3 dB is enough to obtain a BER of 10^{-6} which relates to a requirement concerning the SNR level at the input of the pre-sampling filter of about -2 dB.

In an environment with fading it seems hard to catch the multi-path characteristics of that channel in a typical description. For a set of channels found in literature however, the relation between BER and mean SNR was found. For a situation in which the relation between line-of-sight power and specular power is 11 dB, an input-SNR level of 0 dB gives a BER of 10^{-4}. For this case the SNR level at the input of the pre-sampling filter should be about 1 dB.

Chapter 5

IMPLEMENTATION ALTERNATIVES

Contents

5.1 Introduction **67**
5.2 Data detection **69**
5.2.1 Complexity reduction of the data detection algorithm . 72
5.2.2 Noisy environments 76
5.2.3 Performance analysis 80
5.2.4 Conclusion 83
5.3 Synchronization **84**
5.3.1 Carrier-synchronization 84
5.3.2 Code-synchronization 85
5.3.3 Acquisition 86
5.3.4 Code-tracking 95
5.4 Front-end considerations **103**
5.4.1 Introduction 103
5.4.2 Considerations 104
5.4.3 Conclusions 104
5.5 Conclusions **105**

5.1 Introduction

During system implementation we often find that it is not possible to map the desires of the systems engineer on the available hardware and software resources. It seems that two options are left: either we buy more advanced resources or we suggest the systems engineer to accept a less favorable solution and try to compensate for this in other parts of the design.

Actually a trade-off exists between SNR-loss and complexity as illustrated in figure 5.1. Complexity can be either software or hardware complexity, but usually a higher complexity leads to higher costs. An increase of the SNR-loss can have different negative consequences. Concerning the data detection for example, a higher SNR-loss will lead to a worse BER-performance and consequently to a higher required input signal strength at the antenna or a more advanced front-end.

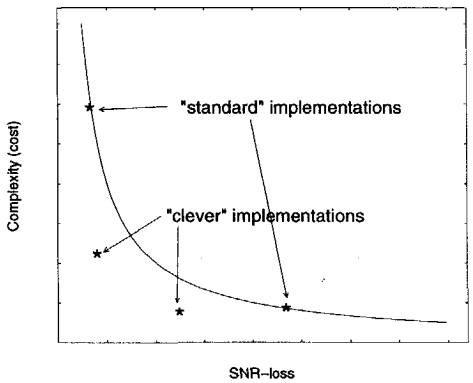


Figure 5.1. Implementation trade-offs

Figure 5.1 shows a trade-off curve of possible "standard" implementations for an arbitrary case. As stated before, it is often the case that existing "desires" are not met choosing one of these implementations. As a result a break-through is required. The "art" of doing a system implementation is now to find implementation alternatives that lay below the standard "trade-off curve".

This chapter investigates two situations in which the complexity can be greatly reduced at the cost of an acceptable SNR-loss. So both cases describe implementations that lay below the standard trade-off curve. The first one concerns the MFSK-data detection unit (section 5.2). After describing the implementation itself, a BER-performance analysis is done to find the relation between input-SNR and BER for this particular situation. The second case concerns the code-synchronization circuitry. In section 5.3 the aspects of the synchronization scheme are discussed and simulation results are presented.

To complete this chapter, section 5.4 deals with a number of front-end issues that exist for communication systems like WISSCE. This section is not meant to give a detailed description of the front-end implementation, instead only WISSCE-specific aspects are addressed.

5.2 Data detection

Data detection is the process of recovering the originally transmitted data-message. The way to perform this operation is for a large extent dependent on the applied data modulation scheme. In WISSCE this scheme is chosen to be 16-MFSK (see section 3.2). The goal of this section is to explain the operation of the data detection algorithm used in WISSCE.

In the receiver the function of data detection can be interpreted as measuring the energy present in all MFSK-CHANNELS, and selecting that MFSK-CHANNEL with the highest energy. The symbol corresponding with this MFSK-CHANNEL is most likely the transmitted symbol.

Evaluation of the energy in a number of frequency-bands (for instance MFSK-CHANNELS) can be done in a number of ways. Traditionally the received signal goes into a filter-bank, then per channel a square-operation is applied to obtain the power. After that, an I&D-filter is used to determine the energy over a symbol-time (square-law detection). This approach has the disadvantage that for every MFSK-CHANNEL a band-pass filter is required, which makes the detector expensive.

A different way of estimating the transmitted symbol is by applying a fourier transform on the input signal. The output of this operation gives the frequency-representation of the signal. A square-operation now produces the energy in a channel over a symbol-time. As we are dealing with digital inputs, a short-time DFT is appropriate to determine the energy per symbol. An advantage is that area intensive filters are replaced by a transformation which runs on a shared resource (embedded processor).

The number of samples in a symbol is equal to the ratio of sample-speed and symbol-speed:

$$N = \frac{r_s}{r_{\text{symbol}}} = \frac{5.2\,\text{MHz}}{20\,\text{kHz}} = 260, \tag{5.1}$$

as a result, there are 260 time-points available to recover the transmitted data symbol.

Implementing a DFT leaves several options: One possibility is to apply an FFT-structure, in this way the operation takes place with worst case time complexity $\mathcal{O}(N\log N)$ instead of $\mathcal{O}(N^2)$ for a normal DFT. However, an FFT computes a number of frequency-points equal to the number of input-samples, that is 260 numbers where only 16 are needed (and usually a power of 2). As a consequence the application of an FFT does not have advantages in this situation, a DFT will be faster.

The way the DFT-operation is implemented in WISSCE can be described in several ways. We derive this operation by starting from a usual fourier transform.

The Discrete Fourier Transform of a despread, quadrature sampled input signal ($x[k] = I[k] + jQ[k]$) for the n^{th} frequency channel, based on N time samples, can be written as:

$$F[n] = \sum_{k=0}^{N-1} (I[k] + j\,Q[k]) \exp\left[-j\frac{2\pi kn}{N}\right] \tag{5.2}$$

or as:

$$F[n] = \Re[n] + j\,\Im[n]$$

in which:

$$\Re[n] = \sum_{k=0}^{N-1} \left(I[k] \cos\left[\frac{2\pi kn}{N}\right] + Q[k] \sin\left[\frac{2\pi kn}{N}\right] \right) \tag{5.3a}$$

$$\Im[n] = \sum_{k=0}^{N-1} \left(Q[k] \cos\left[\frac{2\pi kn}{N}\right] - I[k] \sin\left[\frac{2\pi kn}{N}\right] \right) . \tag{5.3b}$$

The cosine and sine terms will be referred to as TWIDDLE-FACTORS:

$$tw_{\cos}[n,\ k] = cos\left[\frac{2\pi kn}{N}\right] \tag{5.4a}$$

$$tw_{\sin}[n,\ k] = sin\left[\frac{2\pi kn}{N}\right] . \tag{5.4b}$$

Now (5.3) can be written in terms of ACCUMULATION-FACTORS:

$$\Re[n] = \mathrm{Ics}[n] + \mathrm{Qsn}[n] \tag{5.5a}$$

$$\Im[n] = \mathrm{Qcs}[n] - \mathrm{Isn}[n] . \tag{5.5b}$$

These are defined as:

$$\mathrm{Ics}[n] \triangleq \sum_{k=0}^{N-1} I[k]tw_{cos}[n,\ k] \tag{5.6a}$$

$$\mathrm{Isn}[n] \triangleq \sum_{k=0}^{N-1} I[k]tw_{sin}[n,\ k] \tag{5.6b}$$

$$\mathrm{Qcs}[n] \triangleq \sum_{k=0}^{N-1} Q[k]tw_{cos}[n,\ k] \tag{5.6c}$$

$$\mathrm{Qsn}[n] \triangleq \sum_{k=0}^{N-1} Q[k]tw_{sin}[n,\ k] \tag{5.6d}$$

The equations (5.6) show that a correlation is applied of the input signal with sine and cosine waveforms representing all possible MFSK-frequencies. For this reason we will refer to this detector as an MFSK correlation engine (MFSK-CE).

For the data detection process the phase of the input signal does not contain relevant information. A decision is based on the power contents of the signal:

$$\begin{aligned} P[n] &= \Re[n]^2 + \Im[n]^2 \\ &= (\mathrm{Ics}[n] + \mathrm{Qsn}[n])^2 + (\mathrm{Qcs}[n] - \mathrm{Isn}[n])^2 \ . \end{aligned} \tag{5.7}$$

To reduce the number of required bits to represent the power level $P[n]$, a digital shift is applied (division by 512). For the decision variable follows (compare equation (4.21) on page 51):

$$\begin{aligned} P'[n] &= \lfloor \Re[n]^2/512 \rfloor + \lfloor \Im[n]^2/512 \rfloor \\ &= \lfloor (\mathrm{Ics}[n] + \mathrm{Qsn}[n])^2/512 \rfloor + \lfloor (\mathrm{Qcs}[n] - \mathrm{Isn}[n])^2/512 \rfloor \end{aligned} \tag{5.8a}$$

where $\lfloor \cdot \rfloor$ denotes the floor function, consequently results are rounded down to integer values.

In WISSCE there are 16 MFSK-CHANNELS, 8 on the positive frequency-axis and 8 mirrored on the negative axis. Because of this property it is possible to calculate the energy in a positive MFSK-CHANNEL and its mirrored negative version from the same ACCUMULATION-FACTORS (and consequently the same TWIDDLE-FACTORS):

$$P'[-n] = \lfloor (\mathrm{Ics}[n] - \mathrm{Qsn}[n])^2/512 \rfloor + \lfloor (\mathrm{Qcs}[n] + \mathrm{Isn}[n])^2/512 \rfloor \ . \tag{5.8b}$$

This already reduces the number of operations to perform data detection with a factor of almost 2 and halves the required number of TWIDDLE-FACTORS. There is however still a discrepancy between the available resources and the required computational power. This results in the requirement to further reduce the complexity of the data detection algorithm.

5.2.1 Complexity reduction of the data detection algorithm

To further reduce the required computational power we introduce 2 simplifications:

- To avoid the need for automatic gain control, a limiter will be applied during the analog to digital conversion stage. As a consequence the number of bits to represent the input signal is reduced to one.
- The number of levels to represent the TWIDDLE-FACTORS will be reduced to 3.

As a result the decision variables get much easier to calculate: full-size multiplications become 1 bit additions and subtractions. There is however also another side: a loss in performance (BER) will appear as well. In the following this trade-off will be analyzed. We will show how the reduction of complexity is achieved and what the consequences are on the performance.

5.2.1.1 3-leveled Twiddle-Factors

The quality of the data detection is dependent on to what extent the twiddle-factors meet the following properties:

1. Equal sensitivity for all frequencies.
2. Not containing other frequencies than its own. (So if for instance symbol "2" is transmitted, only $tw_{\sin}[2,\ k]$ and $tw_{\cos}[2,\ k]$ should show correlating with that signal, the others should be uncorrelated.)
3. $tw_{\sin}[n,\ k]$ and $tw_{\cos}[n,\ k]$ should be orthogonal. (Otherwise the power calculated according to (5.8) becomes phase dependent.)

By reducing the number of levels to represent the TWIDDLE-FACTORS, harmonics will appear. Especially the third harmonic is of major importance. It makes the TWIDDLE-FACTORS sensitive to other frequencies than its own. To reduce the power-levels in the third and fifth harmonic while retaining orthogonality between sine and cosine TWIDDLE-FACTORS, 3-leveled factors can be applied [Reg]. Those factors are derived on the basis of delta-sigma modulation:

$$tw_{cos}[n,\ k] = \frac{d}{dk}\left[\frac{N}{2\pi n}\sin\left(\frac{nk2\pi}{N}\right)\right]$$
$$tw_{sin}[n,\ k] = \frac{d}{dk}\left[-\frac{N}{2\pi n}\cos\left(\frac{nk2\pi}{N}\right)\right]$$

where d/dk symbolizes a differentiation to the time-step (k). In WISSCE N is chosen to have a value of 260. To obtain the 3-leveled factors, we perform the following calculation:

$$tw_{\cos}[n,\ k] = \lfloor sn[n,k+1]\rfloor - \lfloor sn[n,k]\rfloor \quad (5.9a)$$
$$tw_{\sin}[n,\ k] = \lfloor cs[n,k+1]\rfloor - \lfloor cs[n,k]\rfloor \quad (5.9b)$$

where $\lfloor\cdot\rfloor$ denotes the floor function and:

$$sn[n,\ k] = 0.5 + \frac{a}{n}\sin\left(\frac{nk\pi}{130}\right) \quad (5.10a)$$
$$cs[n,\ k] = 0.5 - \frac{a}{n}\cos\left(\frac{nk\pi}{130}\right). \quad (5.10b)$$

To minimize the third and fifth harmonic, the value of a is fixed on a value of 40. As an illustration of how such TWIDDLE-FACTORS can look like, $tw_{\cos}[1,k]$ is shown in figure 5.2. This TWIDDLE-FACTOR represents a cosine waveform of frequency "1".

In table 5.1, the sensitivities of the 3-leveled twiddle-factors for all 16 possible input symbol frequencies are shown. Horizontally the transmitted symbols are listed, while vertically the detected power-levels are given. It can be observed that applying 3-leveled twiddle-factors leads to unequal sensitivity for different MFSK-CHANNELS. They differ from 123 to 125, a difference that can be allowed.

Furthermore it looks like the TWIDDLE-FACTORS are only sensitive to their own MFSK-CHANNEL. This is however not completely true, the sensitivity for other frequencies than its own is just not visible due to the 512-division factor and the application of a floor function (see equation (5.8)).

5.2.1.2 Limiting of the input signal

A significant further reduction in complexity can be achieved by applying a limiter in the analog to digital conversion. On one hand automatic gain is not required and the complexity of the digital processing afterwards reduces. On the other hand, this simplification will lead to a performance loss. Two problems are introduced:

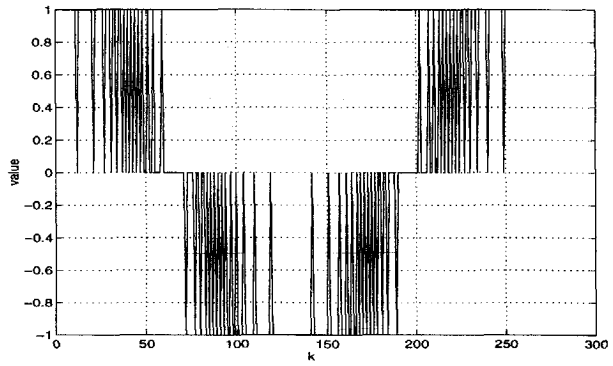


Figure 5.2. Evaluation of $tw_{\cos}[1,k]$

Table 5.1. sensitivity of TWIDDLE-FACTORS

	channel of transmitted symbol →															
	-8	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7	8
-8	124	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-7	0	125	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-6	0	0	124	0	0	0	0	0	0	0	0	0	0	0	0	0
-5	0	0	0	124	0	0	0	0	0	0	0	0	0	0	0	0
-4	0	0	0	0	123	0	0	0	0	0	0	0	0	0	0	0
-3	0	0	0	0	0	122	0	0	0	0	0	0	0	0	0	0
-2	0	0	0	0	0	0	123	0	0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0	123	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	123	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	123	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	122	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	123	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	124	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	124	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	125	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	124

1. A loss in SNR, if the desired signal is not the strongest component of the received signal. A usual way to express this loss is using the signal suppression factor α which represents the signal loss [Lin72].

2. The introduction of odd harmonics. If for example the transmitted symbol is "1", also a response is observed at symbol "3" and symbol "5". Thus the BER-performance will deteriorate.

These problems have a number of consequences: firstly the sensitivity matrix in table 5.1 changes. Secondly the detection-scheme becomes phase-dependent. Only applying the 3-leveled TWIDDLE-FACTORS already introduces phase-dependence that is because $tw_{sin}[n,k]^2 + tw_{cos}[n,k]^2$ is not constant for all k. If the TWIDDLE-FACTORS are combined with a limited input signal, the consequence becomes noticeable. And finally the limiting operation changes the signal to noise ratio as well. The latter consequence is a noise consideration which will be discussed in the following section. For this moment, we will concentrate on the first two issues.

Sensitivity Another sensitivity table, also incorporating the limiting of the input signals, is shown in table 5.2. From this table it is clear that limiting the input signal has an undesirable effect on the sensitivities of the twiddle-factors.

Table 5.2. sensitivity of TWIDDLE-FACTORS on limited input signals

	channel of transmitted symbol →															
	-8	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7	8
-8	201	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-7	0	221	0	0	0	0	0	0	4	0	0	0	0	0	0	0
-6	0	0	220	0	0	0	0	0	0	24	0	0	0	0	0	0
-5	0	0	0	202	0	0	0	8	0	0	0	0	0	0	0	0
-4	0	0	0	0	200	0	0	0	0	0	0	0	0	0	0	0
-3	0	0	0	0	0	190	0	0	21	0	0	0	0	0	0	0
-2	0	0	0	0	0	0	200	0	0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0	200	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	200	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	200	0	0	0	0	0	0
3	0	0	0	0	0	0	0	21	0	0	190	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	200	0	0	0	0
5	0	0	0	0	0	0	0	0	8	0	0	0	202	0	0	0
6	0	0	0	0	0	0	24	0	0	0	0	0	0	220	0	0
7	0	0	0	0	0	0	0	4	0	0	0	0	0	0	221	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	201

The detection in some MFSK-CHANNELS now shows correlation with other frequencies than its own (violation of point 2 on page 72). For instance: if MFSK-frequency "2" is transmitted, energy is found in the MFSK-CHANNEL corresponding to the 3^{rd} harmonic: "-6" ($sens[2,-6] = 24$). So due to limiting only a part of

the signal can be found in the original frequency band (zone) while the rest of the power can be found as higher harmonics.

Phase-dependence Introducing 3-leveled TWIDDLE-FACTORS in combination with a limited input signal introduces phase dependence. The output power of an ideal Fourier Transform is phase independent because of the property that $\sin(x)^2 + \cos(x)^2 = 1$. This property is met if either the TWIDDLE-FACTORS or the input signal is sine-shaped. Unfortunately this is not the case. By looking at the evaluation of $tw_{sin}[n,k]^2 + tw_{cos}[n,k]^2$ for frequency $n = 1$ in figure 5.3, it becomes clear that the sum is not constant as desired.

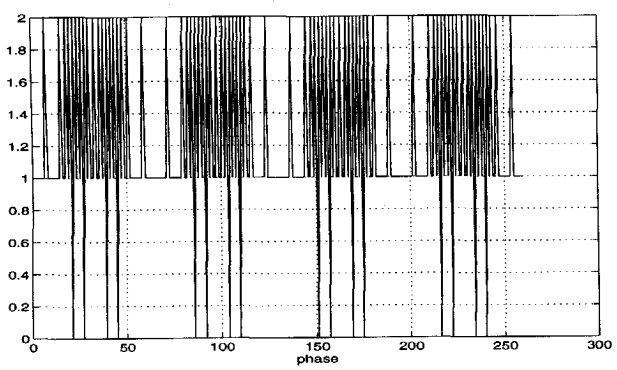


Figure 5.3. Evaluation of $tw_{sin}[1,k]^2 + tw_{cos}[1,k]^2$.

Figure 5.4 shows the sensitivity of the data-detection algorithm for two MFSK-CHANNELS as a function of the phase. It illustrates the existing phase-dependence of the data-detection.

At first sight, by looking at both the sensitivity matters and the phase-dependence problems, one might arrive at the conclusion that the proposed simplifications of the data detection algorithm are not an acceptable alternative. In the next section we will however see that the problems reduce considerably in real situations with appropriate SNR-values.

5.2.2 Noisy environments

Adding noise to a signal degrades the input SNR-level and usually leads to a worse data-detection quality. In systems where a limiter is applied, noise has other implications as well.

Limiting a signal introduces harmonics which are referred to as "zones". Also, due to the constant output-power of a limiter, the desired output signals decrease if the

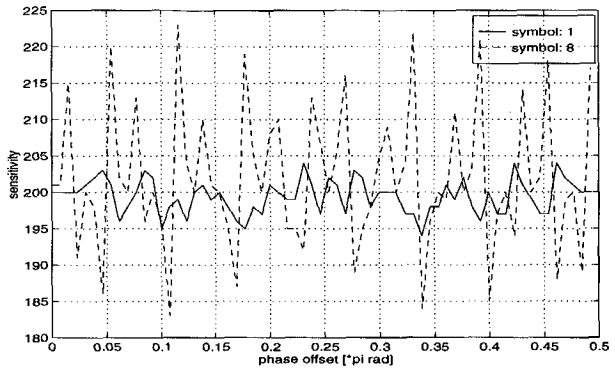


Figure 5.4. Phase dependence for MFSK-symbols "1" and "8"

input SNR-level decreases. The amount of reduction is usually expressed using the "signal suppression factor" α_k. As this factor is smaller for higher zones, the harmonics decrease faster than the desired signal itself. In this section we will show that the harmonics vanish almost completely for appropriate SNR-values. The BER performance now becomes better compared to a situation in which all harmonics are present.

If P_k is the limiter output-power of the k^{th} harmonic, Lindsey [Lin72] wrote the output signal-power of the k^{th} harmonic (P_{ks}) as:

$$P_{ks} = \alpha_k^2 P_k \tag{5.11}$$

$$\alpha_k(SNR_i) = \frac{1}{2}\sqrt{\pi \cdot SNR_i} e^{-SNR_i/2} \cdot \left[I_{\frac{k-1}{2}}(SNR_i/2) + I_{\frac{k+1}{2}}(SNR_i/2) \right] \tag{5.12}$$

where SNR_i is the input signal to noise ratio. Figure 5.5 gives the suppression factor for the first, third and fifth harmonic as a function of the input-SNR (in dB). Also in simulation results are presented in this figure to verify the suppression factors: the power in the different frequency bands is measured as a function of the input-SNR. For low values of the SNR there is a small deviation visible caused by the noise-power contents at the output of the limiter.

P_1 (first harmonic) can also be written in terms of the signal-amplitude (L):

$$P_k = \frac{8L}{\pi^2}. \tag{5.13}$$

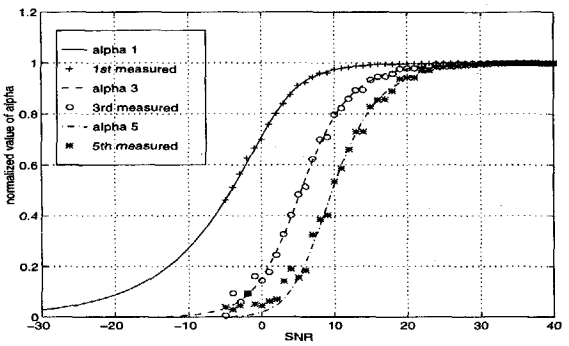


Figure 5.5. Signal suppression-factors: calculated and simulated

For simplicity, L is fixed to be 1. In case of small SNR, the output SNR as a function of the input SNR for the first harmonic is [Lin72]:

$$\mathrm{SNR}_{ok} \approx \frac{\pi}{4}\mathrm{SNR}_i \,. \tag{5.14}$$

So the maximum loss due to the limiter is $\pi/4$ (1 dB). This relation also assumes that the input bandwidth is equal to the output bandwidth. The limiter SNR-gain as a function of the input-SNR for the first harmonic is shown in figure 5.6.

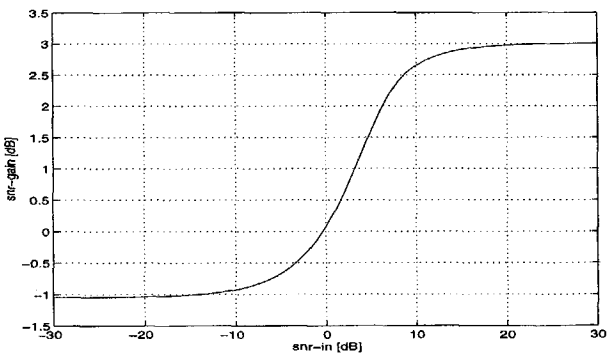


Figure 5.6. Limiter SNR-gain as a function of the input-SNR

To incorporate this effect in the performance analysis, the resulting sensitivity factors from table 5.2 should be multiplied with the suppression-factor for the appropriate harmonic and SNR. The sensitivity for the first, third and fifth harmonic

as a function of the input SNR is shown in figure 5.7. This figure shows the situation that symbol "1" was transmitted, the detection for the third harmonic was at MFSK-CHANNEL "-3" and for the fifth harmonic at MFSK-CHANNEL "5". The curves in the figure represent the elements $Sens[1,1]$, $Sens[-3,1]$ and $Sens[5,1]$ from table 5.2. From this figure can be concluded that for SNR values below 0 dB, no harmonics appear in the output signal.

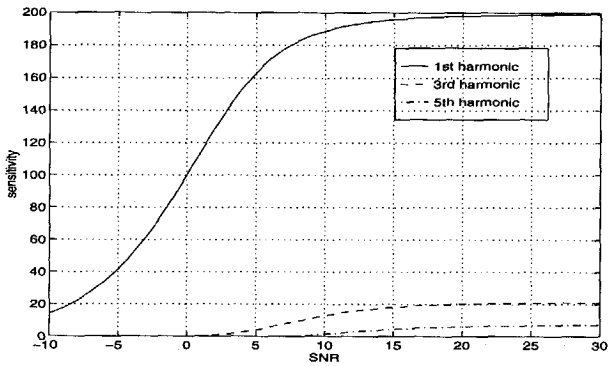


Figure 5.7. Sensitivity as a function of the input SNR

Concerning the SNR one remark should be made. When talking about an SNR-value one should specify a bandwidth. In this case the bandwidth is equal to the sampling bandwidth which in its turn is equal to the chip-rate: 1.26 MHz. The relation between the SNR at this point (referred to as the predetection-SNR: γ_p) and E_s/N_0 (equal to the detection-SNR: γ) is given in (5.15).

$$\gamma = \frac{r_s}{B_{\text{input}}} \gamma_p = \frac{1}{63} \frac{E_s}{N_0} . \tag{5.15}$$

So there is a difference of 18 dB between E_s/N_0 and γ. This means that for all appropriate values of E_s/N_0 the predetection-SNR is below 0 dB. As a consequence the harmonics in the limited signal do not play a significant role (see figure 5.5). Figure 5.7 shows the values from table 5.1 multiplied with the appropriate suppression-factor.

Together with the harmonics also the phase dependence disappears. Limiting in the presence of noise gives the input-signal sine-like properties, which again leads to a phase independent response. Figure 5.8 shows the phase dependence in the detection of different data-symbols ("1", "4" and "8") as a function of the input SNR. The figure shows the variance over phase-shifts in the interval $(0, \pi/2)$, normalized to the variance at an input SNR of 30 dB. The first up to the 19^{th}

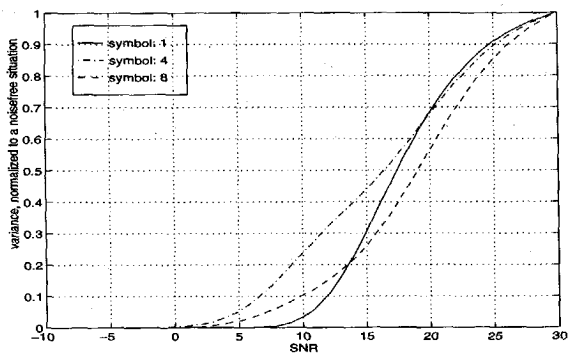


Figure 5.8. Phase dependence as function of the input SNR

harmonic are taken into account. The phase dependence vanishes for SNR-values below 0 dB.

If we take the low input SNR into account, we can conclude the following about the proposed data-detection algorithm:

- Using 3-leveled twiddle-factors has a marginal effect on the sensitivity of the decision variable.
- Limiting the input-signal leads to an increasing difference in detection-sensitivity for different MFSK-CHANNELS.
- Limiting the input-signal does not lead to any significant harmonics when the input-SNR is below 0 dB, as is the case in usual situations. Limiting the signal at lower SNR-values does influence the detection results in the sense that there only appears an extra input-SNR loss of 1 dB.
- The phase dependence of the twiddle-factors vanishes for input SNR-values below 0 dB

These observations justify the conclusion that the introduction of the 3-leveled twiddle-factors in combination with limiting of the input-signal only influences the detection performance by introducing a loss in SNR of maximally $\pi/4$ (1.06 dB).

5.2.3 Performance analysis

A schematic overview of the implemented data-detection algorithm is shown in figure 5.9. After limiting the I- and Q-input signals are multiplied with the PN-CODE to remove the DS-spreading. After that, the signals enter the DFT-CE (DFT

correlation engine) which calculates the real and imaginary parts of signals in the 16 MFSK-CHANNELS. A square operation provides the power-levels at the output. Finally, that MFSK-CHANNEL is selected that has the highest energy contents.

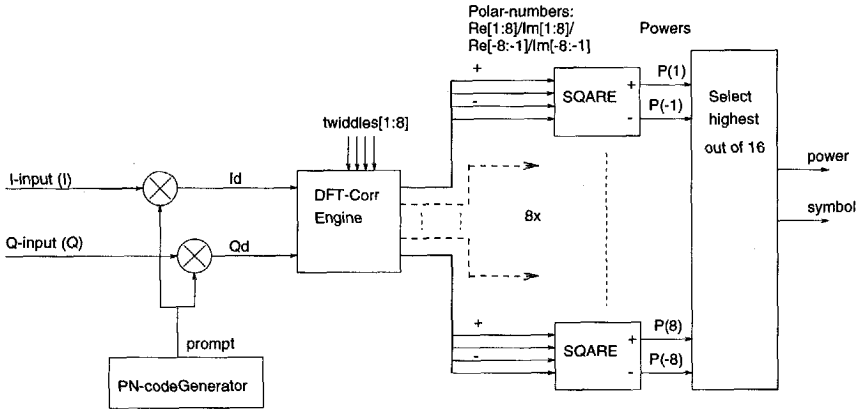


Figure 5.9. Data-detection scheme

The decision variable on which the data detection is based is the energy at the output of an MFSK-CHANNEL (see equation (5.8)). This energy is calculated as the sum of the squares of two ACCUMULATION-FACTORS. These factors result from N ($N = 260$) additions of independent samples, this validates the use of the Central Limit Theorem. Consequently the ACCUMULATION-FACTORS have a Gaussian distribution. As these factors are squared and accumulated, the result (5.8) has a non-central chi-square distribution with 2 degrees of freedom [Mil75, p.56].

The analysis for this BER figure follows the same line as the derivation of the BER performance in an additive Gaussian Noise channel (section 4.3.1 on page 50). Additional aspects due to implementation issues will be incorporated here. Formula (4.26) on page 53 is used as a starting point. For convenience reasons this formula is repeated here:

$$P_{\mathrm{e,\,bit}} = \frac{8}{15}\sum_{m=1}^{15}(-1)^{m+1}\binom{15}{m}\frac{e^{-\gamma m/(m+1)}}{m+1} \tag{5.16}$$

in which γ is the predetection SNR:

$$\gamma = \frac{S}{2\sigma^2} = \frac{E_s}{N_0}\frac{r_{\mathrm{symbol}}}{B_{\mathrm{input}}}.$$

The aspect we want to investigate here is the performance loss due the the input limiter. As the behavior of the limiter depends on the input SNR ($= \gamma_p$), this value will be used as a reference as in earlier analyzes. Applying the data from figure 5.7 yields:

$$\gamma' = \frac{B_{\text{input}}}{r_{\text{symbol}}} \alpha_1^2(\gamma_p)\, \gamma_p \,. \tag{5.17}$$

where γ' is the detection SNR corrected for the limiter behavior. Furthermore, the following relation between the "sensitivity-value" and the (post-limiting) noise variance exists:

$$\gamma' = \frac{sens}{2\sigma_n^2} \,. \tag{5.18}$$

The symbol error rate can be calculated using formula (5.16) by replacing γ for γ'. This leads to the BER-plot in figure 5.10. In this figure also a line representing the "ideal" performance is shown.

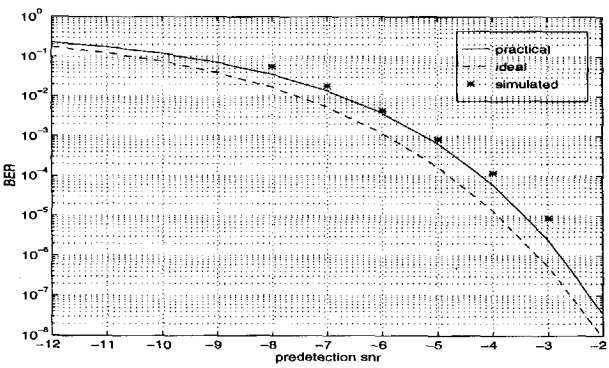


Figure 5.10. BER as a function of the input-SNR

To verify the analytical results we also performed BER-simulations. These simulations were done by modeling the receiver and the environment in C. As a channel model only the additive Gaussian noise channel was considered.

From the simulation results, also shown in figure 5.10, we see that the analytical results are valid for input-SNR values below -3 dB. Is the input-SNR higher, the harmonics of the twiddle-factors have to be taken into account (table 5.2). This however heavily complicates the calculation as the non-zero entries of table 5.2 have to be taken into account. Formula (5.16) was based on the fact that there

is only one frequency-band which contains power and as a result does not hold anymore. If we redefine the probability of the power in a frequency-channel (reconsider the central/non-central chi-square distributions (4.22) and (4.23) on page 52), $P[n,i]$ is the probability density function of the output power in frequency channel n if frequency i is transmitted.

$$p(P[n,i]) = \begin{cases} \frac{1}{2\sigma^2} \exp\left[-\frac{1}{2}\left(\frac{P}{\sigma^2} + \gamma[n,i]\right)\right] I_0\left(\sqrt{\frac{\gamma[n,i]P}{\sigma^2}}\right) & \text{Sens[n,i]} \neq 0 \\ \frac{1}{2\sigma^2} \exp\left(-\frac{P}{2\sigma^2}\right) & \text{Sens[n,i]} = 0 \end{cases} \tag{5.19}$$

where:

$$\gamma[n,i] = \frac{Sens[n,i]}{2\sigma^2} .$$

This leads to:

$$P_c[n] = \int_{P_s=0}^{\infty} \prod_{\substack{-8 \le i \le 8 \\ i \neq \{0,n\}}} \left\{ \int_{P[n,i]=0}^{P_s} p(P[n,i]) d(P[n,i]) \right\} p(P[n,n])\, d(P_s) . \tag{5.20}$$

This probability can only be numerically determined which is time consuming. Fortunately for input-SNR values below 0 dB, the BER-performance is rather good and a deep BER-analysis is not required.

5.2.4 Conclusion

In this section an implementation of an MFSK data-detector was described which combines a much lower complexity (in comparison to a standard DFT-solution) at the cost of a small SNR-loss. The introduced SNR-loss is about 1 dB for appropriate SNR-values.

The reduction in complexity was obtained in three steps:

1. Calculating the power contents of positive and negative MFSK-channels at the same time. This gives a reduction of complexity of about a factor 2.

2. Representing internal numbers of the MFSK-detector with 1.5 bits. This simplifies the required multiplication operations considerably.

3. The multiplication steps are simplified even further by introducing a limiter in the analog to digital conversion stage. The multiplication-operation has evolved from a full-size (8 bit) multiplication to a 1 bit times 1.5 bit operation. It also becomes possible to implement the multiply/accumulate stage as a simple ripple counter [Tek96, Reg].

In conclusion can be said that the required input-SNR is 1 dB higher, which can be compensated by increasing the output power-level, decreasing the maximal distance between users or by using more advanced resources in the front-end. The gain however is much larger: the MFSK data detection engine is now likely to fit as a functional unit on a sea-of-gates chip.

5.3 Synchronization

To recover the transmitted data-message, a receiver must be synchronized to the received signal. This section describes the synchronization algorithm to be implemented in WISSCE. WISSCE has a number of properties that make "standard" synchronization algorithms not suitable. There also exists a strong demand for a low implementation complexity.

The synchronization problem in spread spectrum systems can be split into two parts: carrier-synchronization and code-synchronization.

5.3.1 Carrier-synchronization

Carrier synchronization is not specific to spread spectrum systems, it takes care of frequency and phase differences between the received signal and the signal that is expected by the receiver. The maximal allowed phase or frequency error is determined by the data-detection scheme.

When applying an MFSK modulation scheme as in WISSCE, data-detection can be performed using a non-coherent detector. Following this strategy implies that phase lock of the local oscillator (or demodulation algorithm) to the received signal is not required. Not requiring phase-lock leaves the question of: "how large is the maximally allowed frequency error?"

Figure 5.11 shows the transfer function of a non-coherent, DFT-based MFSK-detector as a function of the input-frequency. In the left figure we see the range between 5 and 35 kHz. As a single channel has a width of 20 kHz, this represents 1.5 MFSK-CHANNEL. The channel center is located at 20 kHz.

The amount of power loss that can be tolerated depends on a number of factors, for instance the required BER and the loss introduced by other parts of the system. If a 1% loss is allowed at this point, the power loss due to the frequency-error small compared to other sources of loss.

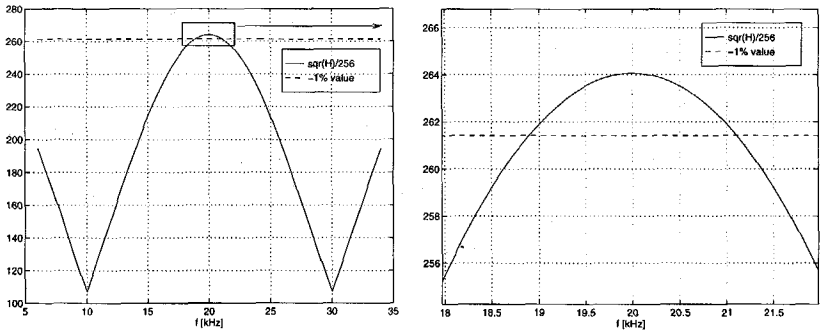


Figure 5.11. Output power of data-detector as a function of the frequency-error

In figure 5.11 there is also a "-1%"-line is shown. This line represents 99% value of the maximum. From the right figure (focused on the top of the transfer-function) it can be seen that the maximum frequency-error is 1kHz to both sides.

Our approach is to use a crystal oscillator (MCXO) with an accuracy high enough to get a maximum frequency-error of 1 kHz in the detector. The principle of the MCXO as well as the clock-control scheme is addressed in section 3.3.

Our target is mainly in-house communications, consequently serious doppler problems are not likely.

5.3.2 Code-synchronization

One of the basic operations in a CDMA receiver is the removal of the "code" which was used by the transmitter to spread the data-message. This operation is known as despreading. In WISSCE despreading includes both removal of the FH-spreading and the DS-spreading. Despreading is performed by combining the received signal with the same code which was used by the transmitter to code the data.

To enable a low BER, the local-code should be aligned with the received signal. An alignment error results is a loss of SNR (see section 4.2.1 for the impact of such an error). Obtaining this code-alignment is the code-synchronization process.

An important observation is that at the beginning of a new PN-CODE also a new symbol and a new frequency-hop starts. Obtaining PN-CODE synchronization therefore implies symbol and FH-SEQUENCE synchronization.

The clock that controls the PN-CODE generator and consequently also the starting of a new symbol-period will be referred to as Local Time Reference (LTR).

This synchronization process can be split into two parts:

- *Acquisition-stage*
  This is the coarse code synchronization process. The objective of this stage

is to resolve the code phase error to within certain bounds which can be further reduced by the tracking-stage.

- *Tracking-stage*
  The remaining error after the acquisition-stage is too large to guarantee proper operation. A fine-tuning process, "code-tracking" is needed. This process is a two-way search, meaning that the LTR can be shifted forward and backward. Tracking is performed continuously during data-detection and keeps the timing-error below an acceptable level.

5.3.3 Acquisition

5.3.3.1 Acquisition strategy

The search for acquisition is based on the auto-correlation properties of the applied PN-CODES [SOSL85b, PG94]: the auto-correlation is high if the receiver is synchronized and low in other situations. This translates in a high detected power-level in case of synchronization and a low power-level in other situations.

The acquisition search-space is set of all possible relative shifts of the local code with respect to the received signal. This search-space is divided in q_{acq} search-CELLs. The process of acquisition is identifying the so-called SYNC-CELL, that is the CELL that corresponds to a situation in which the receiver is synchronized. Searching a single CELL takes a so-called dwell-time (t_{dwell}), or integration-time. After this dwell-time the power at the output of the data-detector is calculated. This power-level is used as a decision variable to select the SYNC-CELL.

In WISSCE the size of a CELL corresponds to half a chip-period ($1/2T_c$) as in typical situations. When decreasing the size, the total number of CELLs will increase and consequently the acquisition-time will increase. Enlarging the shift makes the acquisition-detection process more difficult as the acquisition-decision variable will contain more noise. The total number of CELLs in the acquisition search-space is:

$$q_{\text{acq}} = 2 \cdot N_{\text{DS}} \cdot N_{\text{FH}} = 882. \tag{5.21}$$

The optimal value of the dwell-time is dependent on the SNR. In WISSCE some of the FH-channels might also be blocked by near-interference. It is therefore advantageous to examine at least all FH-channels to ensure that channels without near-interference are examined as well. In this situation the dwell-time will exist of N_{FH} symbol-periods, where all symbol-periods are examined separately. As a result the minimum dwell-time is as follows:

$$t_{\text{dwell}} = \frac{N_{\text{FH}}}{r_{\text{symbol}}} = 350\ \mu s. \tag{5.22}$$

There are two ways to search for acquisition: The first one is called *serial search*: use a single correlator and search the CELLs sequentially. A clear disadvantage is that it takes long since a large number of CELLs is analyzed sequentially to find the SYNC-CELL. Another way to find acquisition is by applying *parallel search*: examine more CELLs at the same time. A number of correlators operate in parallel which causes the acquisition time to decrease. It also increases complexity to analyze the power-contents of the parallel stages. The required amount of computational power easily grows then beyond the available resources.

To summarize: a serial-search strategy is slow, but cheap in terms of resource usage. A parallel-search strategy is fast, but expensive in chip-area and required computational power. For WISSCE the synchronization time should be reasonable short (see user demands), which means that the acquisition time is not critical. Searching all possible CELLs takes about 0.3 s. To achieve low complexity, the *serial search* scheme is selected as the acquisition strategy in WISSCE.

Two important measures determine the "performance" of an acquisition-scheme:

- The false-alarm probability is the chance that acquisition is declared at a wrong CELL.
- The detection probability is the chance that if there is acquisition, this is also detected.

The usual way to tackle the serial-search acquisition problem [PG94] is as follows: After examining a CELL the power-contents of that particular CELL is calculated. If this power exceeds a certain threshold, acquisition is declared for that CELL and normal operation starts directly. If the power does not exceed the threshold, the acquisition algorithm moves to the next CELL ("threshold-search" strategy).

There is however a problem associated with this strategy: the acquisition performance (false alarm probability) is very much determined by the threshold value. The optimal threshold-value is in its turn dependent on the power distribution of the detected signal in CELLs without acquisition and in the SYNC-CELL. For a proper setting of this threshold, the knowledge of the power-level in the presence and absence of synchronization is needed. This level depends on two properties:

1. The input SNR, which is unknown for two reasons: the distance to the transmitter is not known and the fading characteristics of the channel in that particular situation are undefined. Both aspects make the input SNR, and consequently, the detected power level before acquisition unknown.
2. By looking at a plot of detected output power-level against a relative shift value of the LTR (in a serial-search scheme the detected energy as a function of time), we not only observe a single "peak" at the point of synchronization, several other - smaller - auto-correlation peaks appear as well.

This property can considerably increase the detected power-level and thus causing false-locks if the threshold-value does not take this into account. The fact that the power-level is dependent on the partial auto-correlation function, makes an closed formula hard to find.

The conclusion is that the threshold value cannot be determined properly [Gla92, SOSL85b] which results in a non-optimal acquisition strategy. A solution could be to implement an automatic decision threshold control loop [Gli91]. Such a device however increases the required computational power considerably. In a hybrid DS/FH system it is even more problematic as every FH-channel would need its own threshold-loop.

We therefore propose a scheme not directly dependent on a threshold. This scheme searches all possible CELLs and keeps track of the CELL with the highest energy. At the end, that CELL is selected which had the highest energy ("select-highest" strategy). Concerning this strategy we observe the following:

- The mean acquisition-time of the usual "threshold-search" is slightly higher than the time required to search half the number of CELLs [SOSL85b, p.20-27]. Following the "select-highest" scheme results in a mean acquisition time corresponding to the time needed to search all CELLs. So the mean acquisition time doubles with respect to the usual approach.

- For the maximum acquisition-time another story holds. As the detection probability is not equal to 1, it is possible to miss the CELL containing acquisition in the "threshold-search" scheme. If this happens all CELLs have to be searched again and the acquisition time is extended by the time required to perform a "select-highest search".

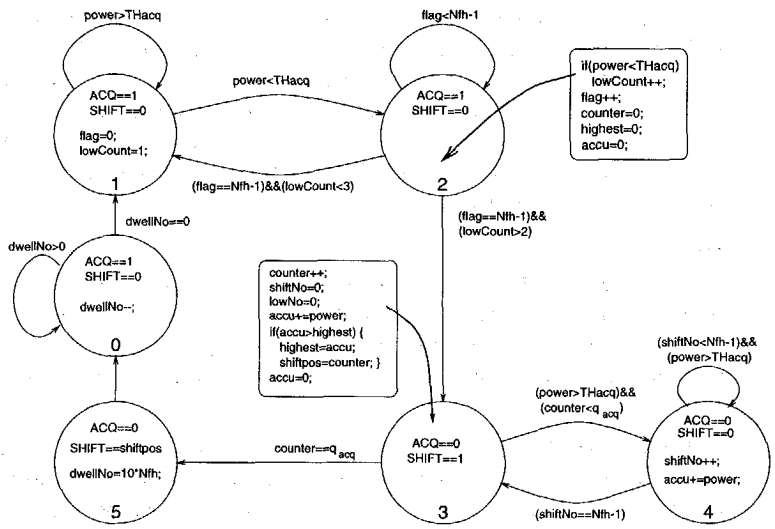
  The problem of the "threshold-search" scheme was that the threshold is difficult to determine. If the threshold is chosen such that autocorrelation properties are taken into account, the resulting detection probability will be low.

The "select-highest" scheme provides a safe way to obtain acquisition within a certain amount of time. When we recall that in WISSCE's protocol (see section 3.2.1), the initiator needs an acknowledgment within a certain amount of time, for this reason we conclude that the "select-highest" scheme is suitable to implement in the WISSCE receiver.

5.3.3.2 Acquisition algorithm

An algorithm implementing the acquisition search process should satisfy the following requirements:

1. The LTR should be shifted when changing CELLs
2. After a dwell-time the resulting power has to be analyzed
3. The algorithm should keep track of which CELL is being searched and which CELLs are already searched
4. After searching all CELLs the algorithm should put the LTR at the right shift-position
5. In normal operation, acquisition testing should give an answer to the question whether the system is still in synchronization (without interruption).
6. The influence of near-far interference should be ruled out as much as possible



ACQ: Is the system synchronized? (1:yes, 0:no)
SHIFT: shift LTR, 1 corresponds to a shift to the next Cell
power: detected power at the end of a symbol-period
THacq: threshold to determine whether the system is in sync
flag: variable used in acquisition detection process
lowCount: counts succeeding number of times that "power" is below "THacq"
counter: counts number of CELLs processed
highest: contains the highest power-level of the examined CELLs until now
shiftpos: CELL that contains the highest power-level
accu: summation of energy in a dwell-time
shiftNo: number of symbols processed in this dwell-time
dwellNo: counter that introduces delay after acquiring acquisition

Figure 5.12. Code-acquisition algorithm

The acquisition algorithm can be caught in the form of the state-diagram in figure 5.12. The operation is as follows:

- *State 1*
  This state corresponds to normal operation, meaning that acquisition is assumed (*ACQ==1*) and the LTR is not shifted with respect to the incoming signal (*SHIFT==0*). The algorithm stays in this state as long as the measured power exceeds a threshold TH_{acq}, this threshold is a lower threshold which is equal to the minimum signal level that guarantees a proper BER. This threshold only has a second order effect on the acquisition performance. Once the power-level drops under the threshold the algorithm goes to the next state.

- *State 2*
  This state is meant to reduce the probability of starting an acquisition search when the system still is in synchronization: it is a buffer-state between normal operation and the start of an acquisition search. After N_{FH} symbol-periods we examine the number of times that the power-level did not exceed the threshold. If this number is larger then two, an acquisition search starts, otherwise the algorithm goes back to normal operation. The value of two is chosen because of the property that there should be at least two FH-channels without near-interference.

- *State 3*
  In this state the LTR is shifted to the next CELL (*SHIFT==1*). This state is usually the state corresponding to the last symbol-period of a dwell-time after which also the data from this dwell-time is processed. The first time the algorithm reaches this state however, this state does nothing but moving to the next CELL.

  When the algorithm comes from state 4, the detected energy obtained in that state (*accu*) is added to the energy detected in this state. In this way a decision variable is obtained which will be used to select the SYNC-CELL. The variable *highest* keeps track of the highest value until now. The CELL number corresponding to this "highest" value is kept in *shiftpos*.

- *State 4*
  The algorithm stays in this state for the first part of the dwell-time. The last symbol-period of the dwell-time the algorithm will be in state 3, so the stay in this state will be for N_{FH}-1 symbol-periods. The algorithm accumulates the detected power from all symbol-periods.

- *State 5*
  This state puts the LTR at the CELL for which was decided that it contained acquisition (*SHIFT=shiftpos*).

- *State 0*
  After finding acquisition the system is not yet completely synchronized: the tracking algorithm takes care of that. State 0 introduces a delay to give the tracking algorithm time to tune the LTR exactly to the received signal.

The acquisition time is equal to q_{acq} times a dwell-time, this is in this particular situation 0.3 s (formulas (5.21) and (5.22)). A number of errors can occur during an acquisition search. It is however hardly possible to give a detailed analysis of these errors as they depend on the current SNR, auto and cross correlation properties and the number of near-interferers present. For this reason we will briefly mention possible errors and give a handle how they can be influenced. Simulation results are presented in the next section.

- *The system starts a new acquisition search while still synchronized.*
  If such a failure takes place, a new acquisition search will start. As a consequence many symbol-errors are likely to occur.

  This error probability can be reduced by either lowering the threshold used to determine whether the system is still synchronized (*THacq*) or by lowering the number of MFSK-channels in which this threshold should be exceeded. During the simulation runs presented in the next section, *THacq* is equal to 14 while two MFSK-channels must show an energy higher than this threshold. By choosing the threshold equal to 14, the false-alarm probability is equal to the complement of the detection probability for an input-SNR of -5dB, no near-interferers present and auto and cross-correlation peaks not taken into account.

  To avoid the start of an acquisition search shortly after finding the SYNC-CELL, the initial value of *dwellNo* should be high enough so that the tracking-process can stabilize.

- *The system does not start a new search while not synchronized.*
  This probability is not as critical as the one above. It just takes longer before an acquisition search starts.

  This probability can be reduced by either increasing *THacq* or increasing the number of MFSK-channels in which the energy should exceed this threshold.

- *The wrong* CELL *is considered to be the* SYNC-CELL
  This error occurs if the acquisition decision variable (accumulated energy during a dwell-time) is not maximal for the SYNC-CELL. This error results in a new acquisition search. The probability on this error can only be affected by combining the detected energy from the symbols in a dwell-time in an other way.

5.3.3.3 Implementation issues

Code-synchronization is the process of controlling the local time reference (LTR) in such a way that it gets locked to the received signal. Here the implementation of the LTR is discussed as well as the mechanism to the control the LTR.

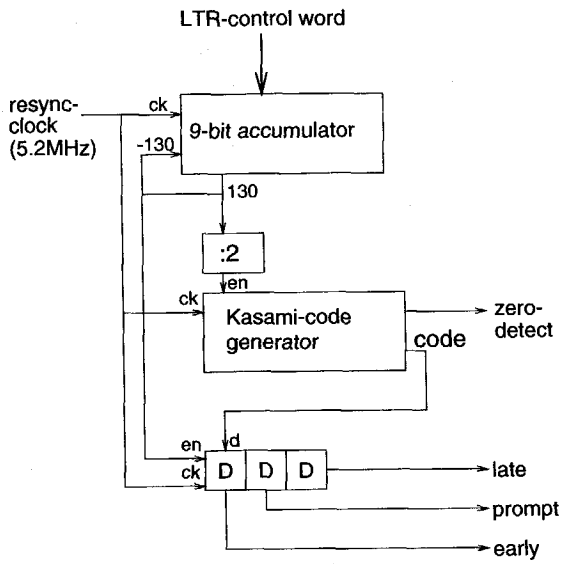


Figure 5.13. LTR-control circuit

Figure 5.13 shows a schematic view on the LTR. Every sample-period an LTR-control word is added to a 9-bit accumulator which adds modulo-130. If the accumulator exceeds 130, this number is subtracted, and an *130-output* pulse is generated.

After the generation of two *130-output* output pulses, the Kasami PN-CODE GENERATOR is shifted one position. The generator output is then fed into a shift-register clocked at double the frequency (using the *130-output* without divider). In this way the early/prompt/late signals can be generated. After a complete PN-CODE-period of 63 chips, the generator reaches its initial all-zero state. Once this state is detected, a pulse is generated to start the processing of a new symbol-period.

The ratio of the sample-rate to the code-rate is 5.2 MHz/1.26 MHz = 260/63. So, the LTR-control word during full-synchronization has a value of 63. In this way it takes 260 sample-periods to complete a symbol period.

Shifting the LTR with respect to the incoming signal can be done by changing the LTR-control word once at the start of a symbol. By doing this, the PN-CODE

GENERATOR is shifted with respect to the incoming code in units of 1/260 of a chip-period. In this way the the start of a symbol-period is also shifted with respect to the incoming signals because the two clocks are linked.

During an acquisition-search however, we do not want to shift the LTR in small steps. Instead the LTR shift-value should correspond to half a chip-period. This shift is performed by changing the LTR-control word to 193 at the start of a symbol-period. A complete chip-period corresponds to a LTR-control word of 260, so a control word of 193 corresponds to the "standard" shift-value of 63 increased by an extra LTR-shift of half a chip-period. As a result there are only 258 samples in a symbol-period during the shift to a next acquisition search CELL (instead of the usual 260).

Figures 5.14, 5.15 and 5.16 show simulated acquisition-trajectories using the proposed algorithm. All simulations use the same (randomly chosen) set of code-phase offset values for the active users. The consequence is that acquisition should be found at the same CELL every simulation-run. The applied PN-CODES (7 per user) are different for every user and are taken from table A.1 in appendix A. For frequency-hopping the strategy explained in section 4.4.2 is applied. It is assumed that a user only adds interference if that user transmits in the same FH-channel as the intended user.

To enable this kind of simulation a code-tracking algorithm had to be implemented as well. Next section discusses the followed code-tracking strategy.

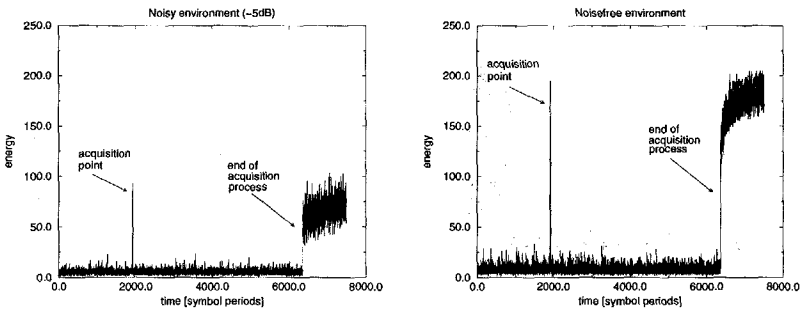


Figure 5.14. Acquisition trajectories for different noise situations

The plots in figure 5.14 show acquisition-trajectories for situations where no interferers are present. It can be seen that in a noisy environment the received signal-power is lower than in the noise-free case. This was already discussed in section 5.2.2. The noisy situation in this figure corresponds to an input-SNR of -5 dB (compare figure 5.10).

In the figures 5.15 two interferers are active. These interferers have a power-level 100 times that of the intended user. For the noisy environment we add noise

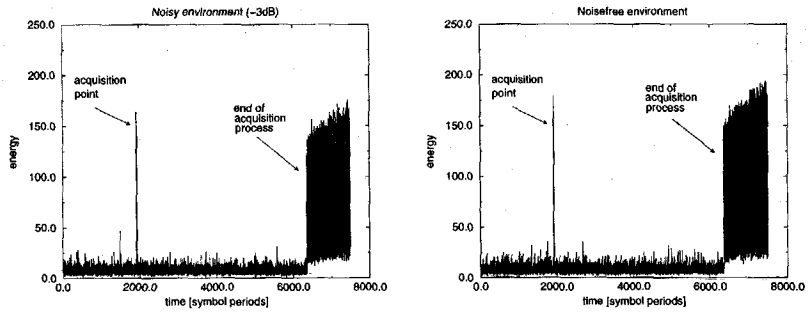


Figure 5.15. Acquisition trajectory, 2 strong interferers present

equivalent to -3 dB input SNR. From the right plot we can see that some FH-slots are jammed while others still contain full energy.

Figure 5.16 shows the acquisition trajectory in the case of a single interferer having 100 times more power than the reference user (near-interference). The right hand-side of this figure shows the system in lock in more detail. It is clear that two out of seven FH-channels are hit. The acquisition system will therefore operate in states 1 and 2.

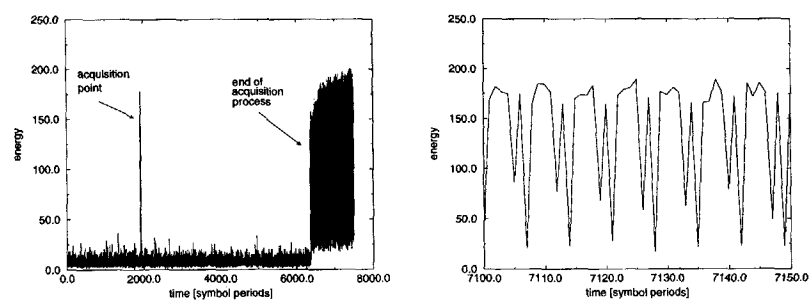


Figure 5.16. Acquisition trajectory with only one strong interferer

5.3.3.4 Conclusion

This section clarified why a "highest-search" acquisition scheme is suitable for application in a WISSCE transceiver. This acquisition scheme was explained and simulation results were presented. The simulations were done using a C-model of the WISSCE-receiver. The acquisition algorithm showed an expected behavior. The simulation results as well as the analyzed acquisition error probabilities justified the choice of this algorithm as an acquisition scheme in WISSCE.

5.3.4 Code-tracking

5.3.4.1 Code-tracking algorithm

Once acquisition is obtained the maximum timing misalignment between the received signal and the locally generated code is equal to the time corresponding to the size of an acquisition search-CELL ($1/2T_c$). Proper operation in such a situation is however not likely. The loss in SNR can be as high as 3 dB (see also section 4.2.1). This leads to a substantial degradation in the BER performance. A second synchronization stage therefore takes care about fine-tuning. This process is called code-tracking.

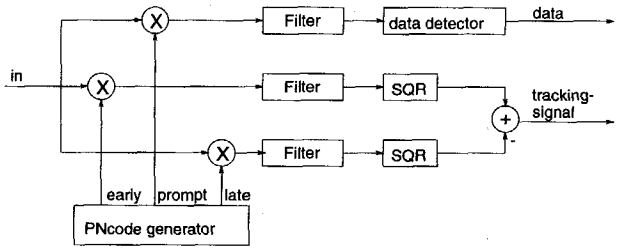


Figure 5.17. Typical code-tracking scheme

The operation of code-tracking schemes [SOSL85b] is usually based on correlating the received signal with an advanced and delayed version of the code-sequence. This process is illustrated in figure 5.17. There are three signal paths in which despreading takes place:

1. The prompt-path which is used for data-detection: here the received signal in correlated with a prompt (reference) code.
2. The early-path in which an advanced version of the code is used.
3. The late-path in which a delayed version of the code is used.

By subtracting the detected powers in the early and late-path from each other, a tracking-signal results. Figure 5.18 shows the tracking-signal as a function of the misalignment error for a typical situation. In this figure T is equal to the time interval between the early and the late-code while "delta" is the misalignment in units of T. This time interval is usually twice the size of an acquisition search-CELL (a chip-period). From the figure we see that, depending on the sign of the tracking signal, we have to shift the LTR in one direction or the other. Another observation is that the tracking-curve is linear in the region from -delta/T to delta/T.

A disadvantage of the early/late tracking scheme is the fact that three signal paths are necessary in which despreading takes place. Extra signal paths also introduce

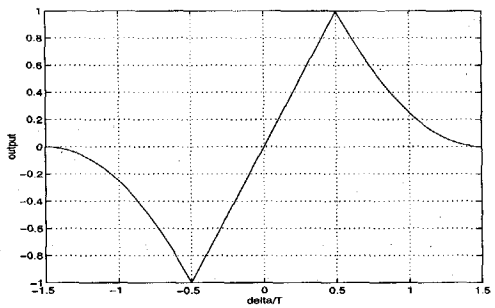


Figure 5.18. Typical code-tracking curve

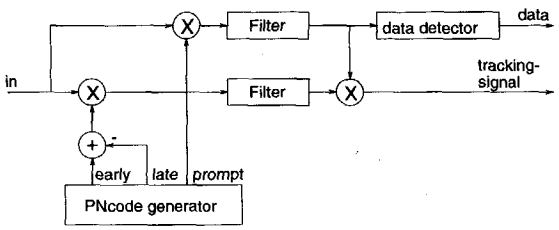


Figure 5.19. MCTL-architecture

the need for extra energy detectors. It would therefore be quite advantageous to reduce the number of paths.

The Modified Code Tracking Loop (MCTL) [YB82] combines the two tracking-paths (see figure 5.19) to reduce the number of signal paths. This scheme subtracts the early and late codes before despreading. A consequence is that square-law detectors cannot be used anymore. A square-law detector removes the "sign" of the signal while this "sign" is required in the MCTL code-tracking scheme.

The WISSCE data detection scheme however, is based on square-law detection. To apply the MCTL code-tracking scheme in this application, three problems have to be addressed:

1. The sign of the tracking-control signal is difficult to determine in a non-coherent receiver structure because the phases of the detected signals are unknown.

2. Because of the square-law detection applied in WISSCE, the output of the detector is a squared amplitude. This results in a flatness around zero in the tracking-curve .

3. During data-detection the energy in all 16 MFSK-channels is calculated. In the code-tracking scheme we do not want to repeat this calculation.

The first problem can be solved by realizing that knowledge of the absolute phase is not required. If the sign of the phase in the *prompt-path* is equal to the sign in the *track-path* the tracking-control signal should be positive. If they have opposite signs the control-signal must be negative. So only the detection of the sign of the signals in the two paths is required. In section 5.3.4.2 it will be shown that this is relatively easy, which leads to the conclusion that this first problem is solved.

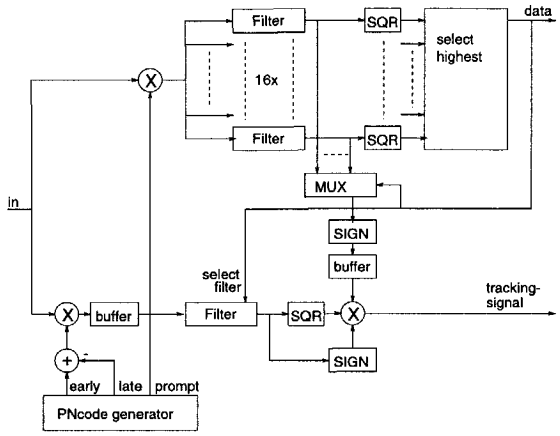


Figure 5.20. Adjusted MCTL-scheme

The second point could be solved by applying a square-root operation on the output-signal. However from the implementation point of view this is very undesirable (large area-costs). Besides, this would be the only place in the receiving algorithm where such an operation is required. Shared usage is therefore not possible.

Another solution is to multiply the power-signal from the *prompt-path* with the output of the *track-path*, but instead use only the output of the *track-path* extended with the right sign. In this way the required computational power is reduced. The price for this simplification is a non-linearity in the tracking-curve.

The third problem can be tackled by first performing all operations in the prompt-path. This results in an estimation on the received data-symbol. During the next symbol-period, the tracking-path processing can use this knowledge to only calculate the power in a single MFSK-channel.

The proposed code-tracking scheme is shown in figure 5.20, the main differences with the standard approach are the following:

- prompt-path detection and track-path detection is not performed at the same time. Only after determining the data-symbol, filtering for the appropriate channel is applied in the tracking-path. Now only a single filtering operation

needs to be performed instead of 16. The two buffers (FIFO) in the figure provide the necessary buffering of data between the pipeline stages.

- After data-detection is finished it is known what channel has to be used as an input for sign-comparison. The output of this operation (1 bit) is during tracking operation multiplied with the power in the tracking-path to obtain a tracking-signal.

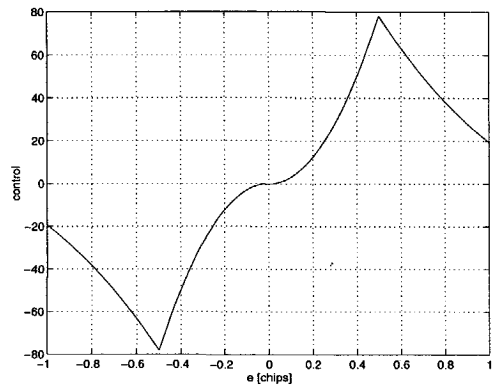


Figure 5.21. Tracking curve of the WISSCE-tracking algorithm

The resulting tracking signal can be expressed as:

$$R_N(\tau) = [R_{PN}(\tau + T_c/2) - R_{PN}(\tau - T_c/2)]^2 \tag{5.23}$$

and is shown in figure 5.21. $R_{PN}(\tau)$ was defined in section 4.2.1 to be:

$$\begin{aligned} R_{PN}(\tau) &\triangleq \overline{c(t)\, c(t+\tau)} \\ &= \begin{cases} 1 - \frac{|\tau|}{T_c} & |\tau| \le T_c \\ 0 & |\tau| > T_c \,. \end{cases} \end{aligned} \tag{5.24}$$

The scaling of the y-axis is according to WISSCE's receiving algorithm. An undesired property of this curve is that the shape of the curve has a flatness around the 0-value. Simulation results in the following chapter will however justify the use of this detector.

We will now focus on the other parts of the tracking algorithm. Aside from the "phase-detector" that provides the tracking-signal, there is also a filter present.

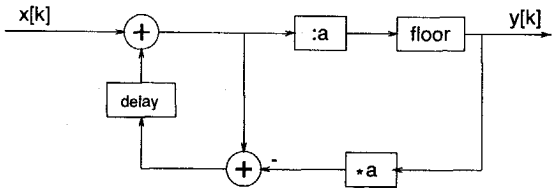


Figure 5.22. Code-tracking filter

The location of this filter is between the "detector" and the time-reference. The function of this filter is to reduce the loop bandwidth and thus reducing the noise level.

An intuitive approach was applied to arrive at the filter shown in figure 5.22. The "floor"-function representing a rounding-down operation. $x[k]$ is the tracking control signal from the detector and $y[k]$ is the output. The transfer function of this filter is:

$$y[k] = \left\lfloor \frac{x[k] + \sum_{n=0}^{k-1} (x[n] - a\, y[n])}{a} \right\rfloor . \tag{5.25}$$

To fix the value of a the operation of the local time reference (LTR) should be taken into account. A part of the tracking-scheme is shown in figure 5.23. The LTR is responsible for giving "shift"-commands to the PN-CODE-generator. Also at the start of a symbol the LTR offers the possibility to introduce a phase step. This phase-step should be specified in steps of 1/63 of a chip-time, due to the chosen implementation.

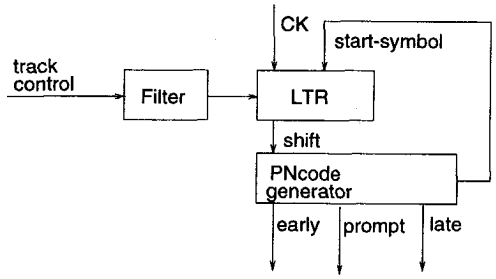


Figure 5.23. Controlling the local time reference (LTR)

Simulation results of the complete synchronization algorithm gave directions for choosing a value of a. Objectives were fast initial tracking behavior and a small residual tracking-error. The value of this a was determined to be 16. The simulation results itself are discussed in section 5.3.4.2.

The following statements can be made concerning the code-tracking process:

1. By following a code-tracking scheme based on the MCTL we combine the early and late signal into a single path. This reduces the required number of operations considerably.
2. The MCTL-scheme is adjusted in two ways: the architecture is changed in such a way that it can be combined with MFSK modulated signals and no square-root operation takes place.
3. Avoiding the square-root operation leads to a flatness in the tracking-curve around zero. This loss however will be tolerated to save the need for extra functionality (square-root operation) not required elsewhere in the transceiver.
4. Due to non-linear effects the tracking-loop is hard to analyze. This was the reason for choosing an intuitive approach to determine a loop filter.

5.3.4.2 Implementation issues

During code-tracking it is important to synchronize the local time reference to the received signal. In section 5.3 we already described the main principle. Here the focus will be on the implementation of the tracking algorithm.

The data-detection and tracking algorithm is shown in figure 5.24. Here we see the translation from the scheme introduced in the previous chapter (figure 5.20 on page 97). The upper side shows the data-detection algorithm as explained in the beginning of this chapter. After multiplying the incoming signal (I and Q) with the prompt-code, the DFT-CE calculates the real and imaginary parts of the signals in the 16 MFSK-CHANNELS. A square operation calculates the energy in the 16 MFSK-channels. The channel with the highest energy is determined and the symbol corresponding to this channel is most likely the transmitted symbol.

In addition this symbol is used to determine the sign of the corresponding real or imaginary part, depending on which is largest. This sign is required in the tracking-loop.

At the bottom of the figure, the incoming signal is multiplied with an early-late code like in the MCTL. Here a DFT-CE only calculates the signal from that frequency-bin that was detected to have the highest energy contents. Again the power and the sign is calculated. Multiplying these signals with the sign from the prompt-path yields a tracking-control signal. It is important that the phase-offset of the TWIDDLE-FACTORS used in the prompt-path is equal to the phase-offset of the TWIDDLE-FACTORS used in the tracking-path. If this is not the case, the detected signs in both paths cannot be compared.

Code-tracking simulation can be performed in several ways. Actually the simulated acquisition-trajectories presented in the previous section already showed a

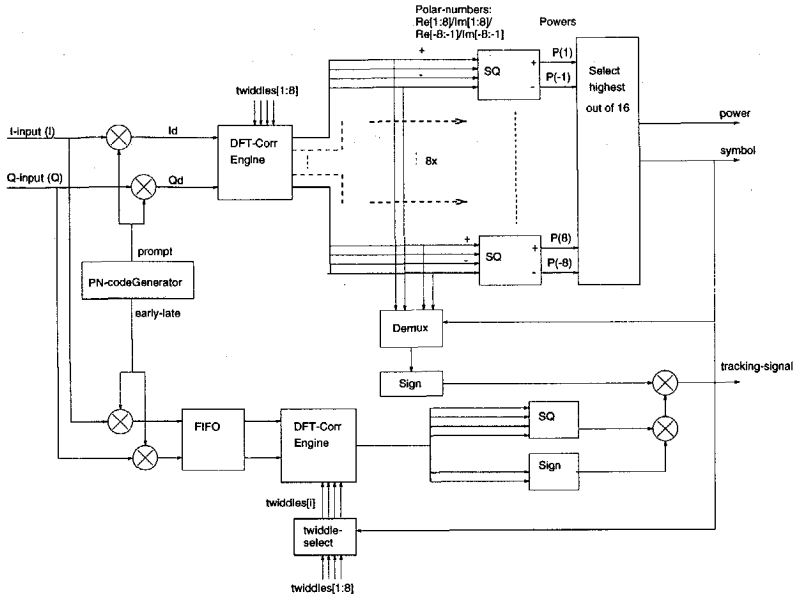


Figure 5.24. Tracking-loop

part of the code-tracking behavior. In this section first the simulated tracking-curve will be compared with the calculated curve. Then the code-tracking behavior will be shown as a function of time.

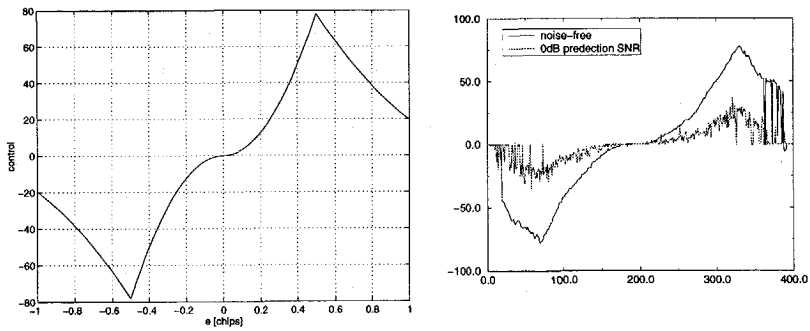


Figure 5.25. Calculated and simulated tracking curves

Figure 5.25 shows a comparison between the simulated and calculated tracking curve. We observe the following:

1. In the noise-free situation the calculated and simulated tracking-curves are

similar in the region for $-T_c/2$ to $T_c/2$. Outside this region the simulated curve shows a number of dips. These are positions where the receiver drops "out of lock".

2. The amplitude decreases if the noise-level increases. As a result the tracking-process becomes slower (due to the structure of the filter, the integration-time increases).

3. Variations in the output-amplitude can be high due to noise. This could have the undesired effect of going up and down, although the loop-filter will decrease these effects.

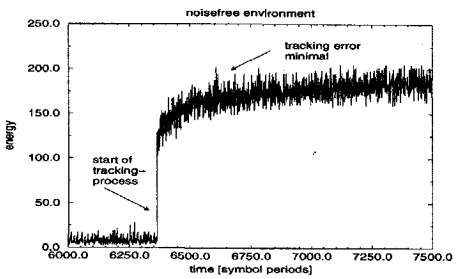


Figure 5.26. Simulation of tracking-process

The tracking-process itself is shown in the plots of figure 5.26. This figure covers the same simulation results as presented in figure 5.14, however we now concentrate on the tracking-process. At a certain moment the LTR is put at the SYNC-CELL. At this moment the output power is about 125 (units). After this moment we find that the received power level increases. This is due to the tracking-process being active. After about 30 symbol-periods the system arrives at a minimum code-tracking error. Another interesting aspect of this plot is the autocorrelation peak at time-stamp 2027. This shows that these autocorrelation peaks can be quite high.

In the left-hand side plot (note the different scale) we see the tracking-process in a noisy environment. The first observation is that the final power-level is lower. A second observation is that the tracking-process is slower. In this situation the system needs more than 60 symbol-periods to come to a stable state.

One conclusion concerning simulation of the tracking process can be that although the code-tracking loop is not analytically analyzed, simulation runs suggest that it behaves as desired. The misalignment disappears while the time spent in initial code-tracking is short to the acquisition time.

5.3.4.3 Conclusion

This section dealt with the code-tracking algorithm implemented in WISSCE. We found that a "standard" code-tracking scheme could not be implemented because of the non-coherent MFSK-modulation technique applied in WISSCE. Also there was an important demand towards a low complexity solution, which was a reason for further simplifying the algorithm.

Because of a number of non-linearities, an analytical evaluation of the code-tracking loop appeared to be very hard. For this reason we relied on simulation runs during both design and evaluation. The final simulation results presented in this section showed satisfactory code-tracking characteristics.

5.4 Front-end considerations

5.4.1 Introduction

The front-end is that part of the system that copes with the high-frequency antenna input signals. It mainly transforms that input signal to a "format" which can be converted into the digital domain. The "front-end" is defined as the part of a transceiver that is situated between antenna and digital baseband processing. In the transmission chain from DDS to antenna and in the receiving chain from antenna to sampler. The front-end has mainly four tasks:

Amplification takes care of amplifying the input-signal which is usually very weak (typically below -80dBm) to a power-level to be used as an input for a limiter converter.

Channel selection and frequency translation is the operation in which the transmitter signal is up-converted to the RF-frequency and the receive signal at the RF-frequency is converted to baseband. Aside from this frequency-translation operation also filtering takes place to select a desired frequency band in the incoming signal (channel selection).

In CDMA-systems channel selection is usually not applied: a single communication links uses the complete receive frequency band and no channel needs to be selected. In WISSCE however, the front-end takes also care about the FH-despreading. This means that there is an frequency-synthesizer present in the front-end which creates the frequency-hopping carrier signals. In this way some kind of selectivity appears in the front-end.

Filtering Beside the desired signals an antenna will also catch other, interfering, signals. As long as these signals are located in other frequency-bands, they can be filtered out.

An issue specific to WISSCE is the fact that the front-end must be able to swap receive and transmit band. This introduces the need for switching filter-bands in the RF-chain.

Removal of the spread spectrum coding In WISSCE despreading is split into two parts: removal of the FH-sequence (FH-spreading) using a direct digital frequency synthesizer (DDS) and removal of the PN-CODE. The latter operation takes place in the digital domain and is consequently of little concern for the front-end. FH-despreading however takes place in the analog domain in the front-end and should be taken into account.

5.4.2 Considerations

Some issues specific to a system like WISSCE are:

1. The frequency-band of interest is wide: the whole DS-band (1.3 MHz) is processed in the digital domain. This desired frequency-band is much wider than in usual systems. The consequence is that using standard channel selection filters is not possible.

2. Because of full-duplex operation, a transceiver must be able to swap receive and transmit bands. This introduces the need for switching filter-bands in the RF-chain.

3. An extra conversion stage is required to perform FH-despreading. The consequence of using a direct digital synthesizer [Puc94, Hol94] is that also spurious components appear in the output spectrum. Some of them are interfering signals and should be filtered out.

The transceiver configuration already shown earlier is repeated here in figure 5.27 for convenience reasons. From this figure it is clear that the frequency translation as well as the channel selection is divided over a number of stages. The frequency-hopping is done using a direct digital synthesizer (DDS, see section 3.3) .

A complete specification of WISSCE's front-end will be described in [Com96].

5.4.3 Conclusions

In this section we briefly addressed the front-end design of WISSCE. As the design of a front-end is not a central research-theme in this thesis, there was a preference to use standard component.

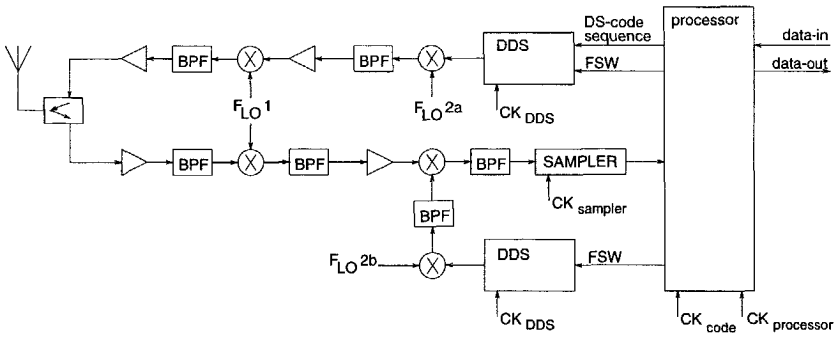


Figure 5.27. Schematic view of possible transceiver architecture

To follow this strategy it is however important to face issues specific to a full-duplex CDMA system like WISSCE. The purpose of this section was to indicate in what senses the WISSCE front-end differs from usual systems.

5.5 Conclusions

This chapter dealt with several implementation aspects. Two important parts of the system were considered to find "efficient" implementations: the data detection engine and the code-synchronization algorithm. Simulation results of both parts showed results that validate their usage in WISSCE.

The data detection and code-synchronization algorithm together form the main parts of the baseband processing of WISSCE. The algorithms presented in this chapter were implemented in a C-description to allow for simulations. This C-description will also be used as an input to the hardware software partitioning stage in the next chapter.

Chapter 6

HARDWARE/SOFTWARE PARTITIONING

Contents

6.1 Introduction 107
6.2 System description 109
6.2.1 Deriving the SIR-graph and profiling information . . . 109
6.2.2 Deriving cost-functions 113
6.2.3 Constraints 116
6.3 Partitioning process 116
6.4 Conclusions 119

6.1 Introduction

While the previous chapters focussed on system design issues and algorithms, this chapter describes the hardware/software partitioning process, i.e.: giving an answer to the question: "what functionality to put in hardware and what functionality to put in software?". This description is based on results gained from the previous chapters.

In the previous chapter (page 82) it was mentioned that a C-model of the receiver was used to enable system-level simulations. This C-model also serves another goal: it can be used as an input to the HW/SW-partitioning process as was described in chapter 2 on page 17. By using the same code for both the system-level simulations and the partitioning we ensure that there are no discrepancies between simulation and implementation at this point.

Providing an algorithmic description of the receiver is unfortunately only the first stage of the partitioning process. It was already explained in chapter 2 that to efficiently search through the whole HW/SW-partitioning design-space it is very much desirable to be guided by an automatic tool. Beside requirements for the partitioning results which were addressed in chapter 2, also user-demands towards usage of such a tool exist. A number of important requirements are:

- *Ability to cope with hierarchy*
  When looking at an algorithmic input description, it is clear that granularity should not be at a single operation. Before the partitioning process starts operations must be clustered into "functions" for which the partitioning question is a reasonable one. Nowadays it is still hard to automatically determine this kind of "functions". As a compromise an automatic tool should enable the designer to apply his/her ideas about clustering (hierarchy).

- *Verification of the input-description*
  As writing flawless code is rather hard in general, it is important to have some means to verify whether the input-description does not contain errors or inefficiencies. The best way to do this is by presenting the description to the designer in an easy understandable format, for instance via a graphical representation.

- *Ability to cope with uncertainties*
  Functionality is only implemented at the time a decision is made whether that function will be put in hardware or in software. As a result it is not possible to provide exact data on the cost-functions at the time the HW/SW-partitioning process starts. To solve this problem a designer will usually "guess" cost-data. A disadvantage of this approach is that using "wrong-guessed" numbers might give non-optimal results or even results that are outside the specifications. It is therefore important that an automatic partitioning tool can cope with this kind of uncertainties.

- *Putting the designer in control*
  To prevent the HW/SW-partitioning process from going into wrong directions and consequently arrive at undesired solutions, it is important that the designer stays in control of this process.

A tool-set that meets these requirements to a large extent is STONE [Cap95]. In STONE a number of CASTLE-tools [TSV94] is combined with a HW/SW-partitioning-tool called HSPART [Kar95] . In this chapter we will describe how the partitioning process using STONE works out for WISSCE.

Next section deals with the input data to the partitioning process. Here it is shown how the designer is responsible for clustering and how inefficiencies is the design can be coped with. Also the issue of obtaining the cost-data on all implementation alternatives is covered in this section.

After the description stage we will concentrate on applying HSPART and discussing its results. This is the subject of section 6.3. At the end a number of conclusions will be stated concerning the used HW/SW-partitioning strategy.

6.2 System description

6.2.1 Deriving the SIR-graph and profiling information

In the introduction it was stated that STONE enables a designer to control the HW/SW-partitioning-process. One way to influence this process is by clustering the operations into functions for which the HW/SW-partitioning-question makes sense. To do so, the designer has to use function-calls in the input algorithmic description. The function-calls that appear in the *main*-function of the description are included in the partitioning process. Operations in the *main*-function which are not function-calls will not appear in the graph. These operations are supposed to be implemented in software as part of the "supervisor". It is important to keep this in mind before specifying timing-constraints as these operations will also take time. In conclusion: by structuring the C-code in a certain way the designer decides which functions take part in the partitioning process.

Another way in which the designer can control the structure of the graph is by choosing a way to deal with coarse grain parallelism like for instance pipe-lining. Also the "timing-schedule" will influence the "structure". For instance the choice: what functionality to perform in-line (triggered by the incoming data stream), and what functionality can be performed triggered by the system clock (off-line)?

The fact that the structure of the input description heavily influences the HW/SW-partitioning results has two sides: If a designer does not know exactly how to tackle the problem, it is possible that a sub-optimal solution will be the result [Gla95, WDW94]. An advantage is however that, the designer gets the opportunity to direct the partitioning tool into a sensible direction. The latter results in a faster partitioning process.

Another feature that STONE is said to have is to enable the designer to "verify" the design. This property is implemented by translating the input C-description into a graph which is then, in a graphical way, shown to the designer. In such a graphical representation inefficiencies can be tracked easily and the designer is able to correct the input description before starting the partitioning process.

To perform the translation from C to a graph, a number of CASTLE-tools [TSV94] are used. As a result the internal representation is in the form of a so-called SIR-graph. The SIR-graph representing the digital receiving process in WISSCE is shown in figure 6.1. The "ovals" represent the functions while the "rectangles" represent the variables. As the graph only contains the function calls from the *main*-function with their dependencies, the translation from the C-description to the SIR-graph is not reversible.

To ease the understanding of the receiving algorithm, the structure of this graph is shown in figure 6.2. In this figure it is shown what operations can be performed in parallel.

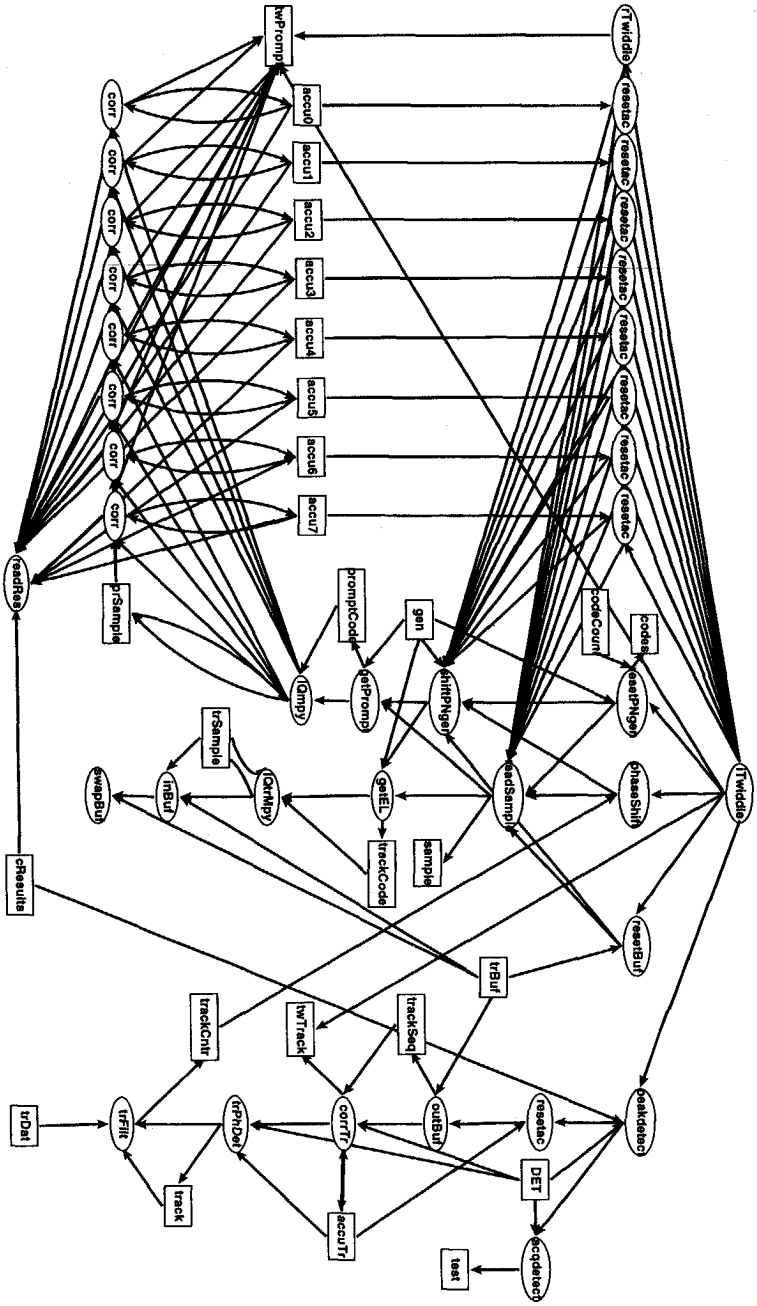


Figure 6.1. SIR-representation of WISSCE's receiving process

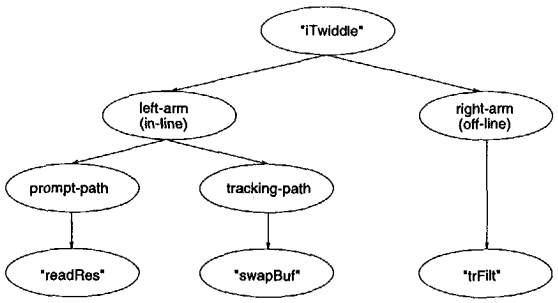


Figure 6.2. Parallelism in the WISSCE-receiver

The function on the top: *iTwiddle* is performed on reset. The operation takes care of the initialization of the TWIDDLE-FACTORS used for data-detection (see section 5.2). All other functions are in a loop and are called at least once every symbol-time. In the left-arm every symbol-time starts with an initialization, these functions can be processed in parallel:

reset_ac is applied to reset the data-detection accumulators (*accu*) to their initial value.

resetPNgen resets the PN-CODE-generator (*gen*) to its all-zero state and loads a new code. The codes to be used are stored in *codes* while *codeCount* is an index to the actual code.

phaseShift tells the local time reference (LTR) to make a phase-step, the size of this step is read from *trackCntr*. This function takes care about code-tracking.

Thereafter a loop starts, which is carried out N times. N is the number of samples in a symbol-time, which is fixed to be 260 (see (5.1) on page 69). These functions are performed "in-line" which means that the loop is triggered by the sample-clock.

readSample takes care about reading a new input-sample and putting it in *sample*. Such a symbol contains a in-phase and quadrature sample, both in a one-bit representation.

shiftPNgen shifts the PN-CODE-generator one "sample"-time.

After reading an input-sample and shifting the PN-CODE-generator, two activities start in parallel: the prompt-path processing which collects data for the data-detection, and the tracking-path processing which does the early-late despreading which is needed to perform code-tracking. In the prompt-path processing the following functions are performed:

getPrompt reads the the prompt-code (*promptCode*) from the PN-generator, called after shifting the generator.

IQmpy multiplies an in-phase/quadrature sample with a 2-leveled code-chip, the result is put in *promptSample*.

corr is an operation which performs the prompt-path data detection. The correlation goes in-line while the results are accumulated in *accu*. During this operation the TWIDDLE-FACTORS are read from *twPrompt*. The correlation is done in eight independent channels and can therefore be performed in parallel. A single *corr*-operation consists of the calculation of four ACCUMULATION-FACTORS (for two MFSK-CHANNELS).

readRes collects after N sample-times the correlation-data (*accu*) and puts it in *cResults*.

The tracking-path processing contains the following functions:

getEL reads the early-late code, this is a 3-leveled signal which results from subtracting the late-code from the early-code. The code chip is put in *trackCode*.

IQtrMpy performs the same functions as *IQmpy* but now for a 3-leveled code-chip (which is the case for the early-late code). The result is put in *trackSample*.

inBuf puts a 3-leveled in-phase/quadrature sample in *trBuf*. During the tracking-stage (*corrTr*) samples are read from the buffer.

swapBuf takes care of "swapping" the tracking-buffer (*trBuf*) after receiving all inputs belonging to a symbol-period. This buffer forms the connection between the two pipe-line stages. If a new symbol starts, the samples which are read during the previous symbol are swapped to the output buffer. The input-buffer is reset so that new samples can be read.

The right-arm represents the second pipe-line stage. The first operation is *peakdetect* which converts the data from *corrResults* into powers per channel. Thereafter it selects that channel with the highest energy contents. The data-symbol corresponding with this channel is considered to be the transmitted symbol. All data involved with this decision is put in *DET*. After performing the data-detection, four operations are carried out in parallel:

acqdetect finds out whether the system is still in lock, its operation is described in section 5.3.3. A test-bit (*test*) represents its decision.

reset_ac starts a sequence of functions used to perform code-tracking. The functions resets the accumulator used for the tracking-correlation (*accuTr*).

outBuf reads tracking-samples from *trBuf* and puts the requested sample in *trackSeq*.

corrTr performs the same operation as *corr* in the prompt-processing path. However this function deals with 3-leveled input-signals. Results are accumulated in *accuTr*. The TWIDDLE-FACTORS are stored in *twTrack*, Different buffers for the prompt and tracking TWIDDLE-FACTORS are required as they are read at different times.

trPhDet performs the phase-detection in the code-tracking loop (see section 5.3.4). The results goes to the variable *track*.

trFilt filters the output of *trPhDet* (loop-filter). The filter itself is represented by *trDat* its output goes to *trackCntr* which is used by *phaseShift*.

Not all operations in WISSCE's digital baseband processing appear in the graph. The frequency-hopping synthesizers used for FH-spreading and despreading is not included as it was obvious that the high clock-speed required in this circuit demands a hardware implementation.

Except for the combined data and control flow from figure 6.1, the partitioning software also needs to know how often the various functions are called. To obtain this data, we use profiling results after processing a single symbol, results are given in table 6.1. The first column gives the function-name, the second shows the number of times the function is called during a single symbol time. The third and fourth column show whether the function is called in the right-arm or the left-arm. As *initTwiddle* is only called once at the moment the receiver is switched on, it does not appear at all.

6.2.2 Deriving cost-functions

The third feature STONE was said to have is the ability to cope with uncertainties. The problem is that as exact cost-data is usually not known at the moment the HW/SW-partitioning process starts, the designer has to "guess" about this data. This approach has the undesirable consequence that partitioning results can become inefficient or even get out of specification.

The partitioning tool HSPART [Kar95] takes care of this problem by allowing the designer to supply imprecise data in the form of triangular fuzzy numbers (possibilistic data). All costs can now be represented using three numbers:

$$X = (x^m, x^l, x^u).$$

Here x^m represents the most-possible value, x^l the lower-bound value and x^u the upper-bound value of variable X. The most-possible value is equal to the designers "guess". As upper-bound values and a lower-bound values are also taken into

function name	# calls	in-line	off-line
corr	2080	x	
inBuf	520	x	
IQmpy	260	x	
IQtrMpy	260		x
corrTr	260		x
getEL	260	x	
getPrompt	260	x	
inBuf	260	x	
outBuf	260		x
readSample	260	x	
shiftPNgen	260	x	
reset_ac	2	x	x
WriteToBus	1		x
acqdetect	1		x
carrTr	1		x
peakdetect	1		x
phaseShift	1	x	
readRes	1	x	
resetPNgen	1	x	
swapBuf	1	x	
trFilt	1		x
trPhDet	1		x

Table 6.1. profiling data of WISSCE

account, the risk of getting an inefficient or out-of-specification partitioning can be much smaller compared to usual approaches. Upper-bound and lower-bound values are usually rather easy to find. An area upper-bound value can for instance be found by performing a fast placement/routing step. A lower-bound on this parameter could be found by only counting the required gates and leaving out any wiring.

In conclusion: by supplying fuzzy numbers in stead of crisp-numbers the risk of getting a result which is inefficient or out of specification is reduced. On the other hand the designer has to supply extra data on the lower-bound and upper-bound values. This data is usually easy to obtain.

Although the introduction of fuzzy input data can reduce the risk of getting inefficient results, deriving the required cost-estimation data is still tedious work. In the following we will concentrate on the process of cost-derivation.

As the cost-data is to a large extent dependent on the available resources, they will be recalled here. For hardware implementations there is a semi-custom ic fabrication process available which is based on the *fishbone* image: a gate-isola-

tion image in a 1.6μ CMOS process with 2-level metallization. The digital part of the system (both general purpose and dedicated hardware) should be realized, if possible, on a single chip which contains 100,000 n/p transistor pairs. The sea-of-gates design system OCEAN [GS93] is being used for prototyping. Concerning the software-cost, we will assume to have a TTA-processor (see chapter 2, page 19) clocked at a speed of 41.6 MHz.

Let us now make a distinction between data and functions. For data, timing does not make sense, the only time that plays a role is the interfacing time which is specified separately. The estimate on the size of the data is rather easy. The hardware costs can be expressed as a cost per bit-storage, which is to be multiplied by the number of required memory elements. Uncertainties in this context are: does the flipflop need a reset? can dynamic logic be applied? how "clever" is the design? On the basis of experience we define the cost of a single bit-storage element to be (in n/p-transistor pairs):

$$C_{\text{area, flipflop}} = (16, 12, 21).$$

So the most-possible flipflop-size is 16 pairs, the minimum size is 12 and the maximum size 21 transistor pairs. On the basis of this number the hardware-costs of all variables can be determined. The software costs are expressed by crisp numbers equal to the number of bytes used.

The cost-determination of the functions is more complicated. The estimations can be based on previous designs, automatically generated designs or just experience. The estimates used in this example are based on data from previous designs adjusted for the changes introduced.

The costs of software functionalities are derived from profiling the code for the target architecture. To do this the code-generation software belonging to the MOVE-framework is used [Hoo96, Cor95]. Uncertainties in this sense are the amount of parallelism possible. To derive proper cost values, we chose a small MOVE-configuration (2 busses, 2 ALUs, 1 MULTIPLIER and a load-store unit) and had the scheduler do its job. The sequential code-sizes/latencies are used as maximum values, while the parallel code-sizes/latencies are used as most possible values. The minimum values are based on the scheduled numbers which are adjusted for the possibility of a larger MOVE-configuration and manual optimization of the assembly code.

The third set of costs contains the interfacing cost, for this cost there are three possibilities:

1. *hardware to hardware* costs will be assumed to be cheap: once the signal is available it takes only a connection to pass it to another functional block.

2. *software to software* costs include putting data on the bus and reading it from the bus (via sockets).

3. *mixed software to hardware* costs include reading or writing to the bus, before or after that operation the data is available in hardware.

6.2.3 Constraints

HSPART can be configured in such a way that it optimizes the used ic-area under the condition of timing constraints. The area available for the system is a single SOG-chip that contains about 100.000 n/p transistor pairs. This chip should accommodate both the hardware functionality and the software functionality. The MOVE-configuration used to obtain profiling data had a size of about 60% of the chip-area. This leaves 40% or 40.000 transistor pairs for dedicated hardware.

The constraints used in the partitioning process are timing-constraints. For the receiver we define three maximum path latencies: two in the in-line path (left-arm) and one for the off-line path (right-arm). In the in-line path we split the prompt-path from the early/late-path. By choosing these latencies, the critical paths of the receiver are captured in three different path-latencies.

One global time limit is the symbol-period, the processing in all three paths must be completed within such a time-frame of 50 μs. However within this time-frame also supervising tasks (operations in the *main*-function, not appearing in the graph) must be carried out. As there will also be uncertainties in the supervising process, the timing constraints in all three paths will also be represented by possibilistic values. For the supervising tasks the timing cost-estimate is:

$$C_{\text{timing, supervisor}} = (12, 10, 16)\ \mu s$$

which corresponds to a minimum of 20% of the time-frame spent in supervising tasks. The typical value is 24% while the maximum value is 32%. This results in the following timing constraint:

$$\text{Constr}_{\text{time}} = (38, 34, 40)\ \mu s.$$

By now numbers on the cost-data are found as well as timing constraints. The partitioning process can start.

6.3 Partitioning process

After a first partitioning run we decided to lock the tracking-buffer (*trBuf*) and the twiddle-factors (data sequences used by the DFT-CE) (*twPrompt* and *twTrack*)

in external hardware. The reason for this is that the amount of data to be stored in these variables is too much to put on the chip. By "locking" and "unlocking" certain implementation alternatives and by initiating new partitioning runs the designer stays in control of the process.

The final partitioning result is shown in figure 6.3. The resulting costs (in possibilistic format) are given in table 6.2. That table shows that for the chosen configuration the timing-constraints are met, while also the ic-area is reasonable (maximally 30% of a SOG-chip). It should be noted though that these numbers are only indications, the real costs are only known after implementation.

HW chip-area	(20995,19744,29129)	n/p transistor pairs
prompt-path latency	(16640,8728,24819)	ns
track-path latency	(13829,10700,16956)	ns
offline-path latency	(34167,28350,39669)	ns

Table 6.2. Final Costs

In the top left corner of figure 6.3 we see the three path-latencies and the one available processor (MOVE). The dark blocks are selected to put in hardware while for the light-blocks a software implementation is chosen.

What we see from this picture is that the in-line processing path is almost completely put in hardware. This seems to be a sensible choice for the following reasons:

- The algorithm behind the correlators that form the largest part of the in-line processing path, is optimized for efficient hardware realization. For example: the processing is done with 2 and 3 leveled signals instead of 16 bit signals. In hardware this saves space while in a software implementation it would still use complete bytes.
- Some functionality in the system is clustered in hardware. The PN-CODE GENERATOR for instance: if one of the functions dealing with this generator is assigned to hardware, the generator will be on the chip. The other functionality of the generator is then available as well and that functionality will be automatically put in hardware too.
- Except for the correlation, the in-line path does not contain much processing. Simple operations on the input samples are executed, these operations can efficiently be mapped on hardware.

On the other hand, the off-line processing path contains more signal-processing tasks (*peakdetect*, *trPhDet* etc.). Except for the PN-CODE GENERATOR which was

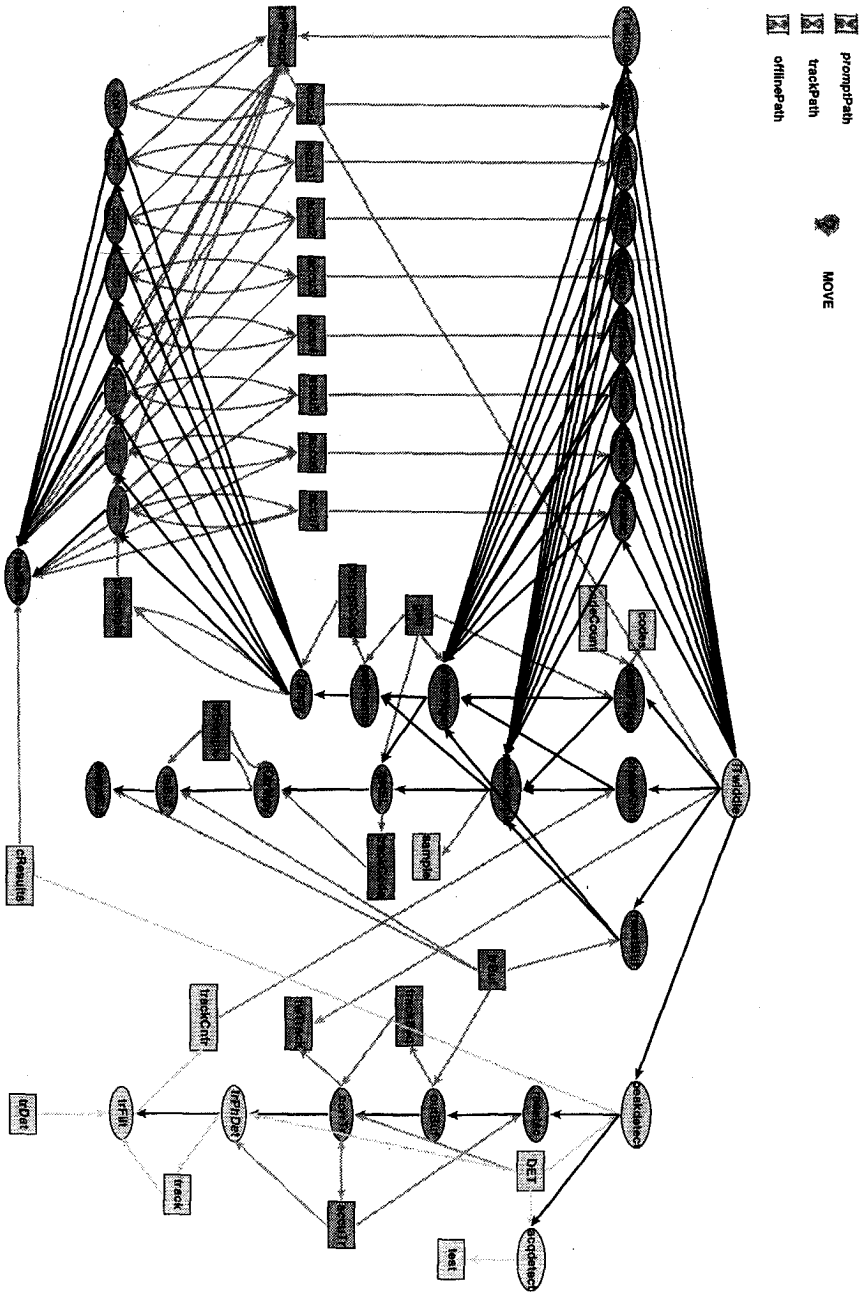


Figure 6.3. HW/SW-partitioning result

already in hardware, those operations were put in software. Also the compute-intensive correlation operation used for tracking (*corrTr*) is put in hardware, just like its equivalent in the in-line path.

6.4 Conclusions

In this chapter the HW/SW-partitioning-process of the WISSCE-receiver is described. We found that using a visualization of the input description was very helpful in detecting flaws in the design. It also appeared that user-interaction is still important. A good example of this is the "clustering": combining operations into a set of functions for which the HW/SW-partitioning-question makes sense. Structuring the input description gives the designer another way to control the process.

The partitioning process itself was eased by the use of HSPART. As the input-data is generally not exactly known at the moment of partitioning, a designer will "guess" about this data with the risk to get a result which is far from optimal or even outside the design specifications. HSPART is a tool that can cope with uncertainties in input-data, in this way the risk of getting inefficient results can be controlled.

The partitioning results from the previous section can now be used to configure a processor framework and to design the hardware and software parts. This part of the design-process will be addressed in the next chapter.

Chapter 7

WISSCE ON THE MOVE

Contents

7.1 Introduction 121
7.2 Hardware design 122
7.2.1 Processor framework 122
7.2.2 Application specific hardware 123
7.2.3 Processor configuration 128
7.3 Firmware design 128
7.4 Co-Simulation 131
7.4.1 A co-simulation tool for the MOVE-processor 131
7.4.2 Simulation level 132
7.4.3 Co-simulation 134
7.4.4 Conclusions 136
7.5 Conclusions 139

7.1 Introduction

In previous chapters all necessary data was collected to enable the design of a processor framework and firmware to implement the baseband processing of the WISSCE transceiver. This chapter will make the design more concrete. It will show whereto the design trajectory described in previous chapters can lead.

First the hardware issues will be addressed. The processor framework existing of both standard-functionality and application specific functionality will be described. Also a processor outline will be shown. As an important part of the functionality is implemented in software, the firmware design will be addressed separately in section 7.3. After discussing both the hardware and software parts, the combination of them can be evaluated. To this end co-simulations will be performed, a subject addressed in section 7.4. At the end we will briefly evaluate the issues addressed in this chapter.

7.2 Hardware design

The hardware used to implement WISSCE can be divided into two parts: standard functionality that can also be found in general purpose processors and application specific functionality dedicated for performing special operations. After discussion these two parts, the complete processor configuration is shown.

7.2.1 Processor framework

From chapter 2 we recall that a transport triggered architecture (TTA) was selected as a processor concept on which the implementation of WISSCE is based. For convenience reasons the figure showing this processor architecture is again shown in figure 7.1.

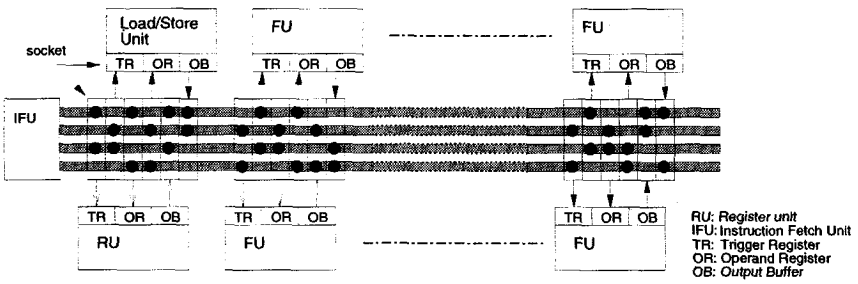


Figure 7.1. Structure of a transport triggered architecture with four busses

An example of a TTA is the MOVE-architecture proposed by Corporaal and Mulder [CM91, Cor95]. Properties of this architecture that make it suitable for implementing an embedded system like WISSCE are:

- *flexibility*
  A designer can select his/her own MOVE-configuration. By the addition or removal of functional units, the processor can be tuned to the receiver's algorithm. Not only "standard" functional units can be included, application-specific functionality can be included as well.

- *simple processor organization*
  The operation of a processor following this concept is simple. Actually the only operation supported by the processor is a "move"-operation. The "real"-functionality of the processor is implemented in functional units (FUs) that can be described independently.

- *ability to cope with various latencies*
  Scheduling is done during compile-time. The scheduler [Hoo96] can cope with different latencies for different FUs. In this way all FUs can have their own latency.

- *instruction level parallelism*
  While a certain FU is performing its job, data can be moved to another FU. This means that more instructions can be processed simultaneously which leads to instruction level parallelism and interleaving.

The remainder of this section is about a processor configuration implementing standard functionality, this configuration will be referred to as MINI-MOVE2, the successor of the MINI-MOVE architecture introduced in [Str94]. Later on the application specific functionality will be added.

The MINI-MOVE2 architecture is a result from refining the MINI-MOVE configuration after interpreting the profiling-data resulting from simulation runs. This configuration was also used to obtain cost-data on software implementation alternatives during the HW/SW-partitioning-stage (see also [Nie96]).

Standard functionality that is included in the MINI-MOVE2 contains the following functional units (FUs):

1. **1 registerfile (GPR)** existing of 16 registers.
2. **2 integer-units (INT1/2)** for addition and subtraction.
3. **1 compare-unit (CMP)** for comparing 2 signed integers and generating "squash requests" depending on the guard of the instruction. A squash cancels a move. In this way conditional execution is enabled.
4. **1 Instruction-fetch unit (PC)** which fetches the instruction, takes care of interrupts and keeps track of the program-counter.
5. **1 logic-unit (LOG)** to perform the boolean operations: SHL, SHR, SHRU, AND, IOR and XOR.
6. **1 immediate-unit (IMM)** to perform immediate addressing. The standard way of performing these operations is not satisfactory. For this reason such a unit is included. In practical realizations this unit will be integrated in the instruction fetch unit.

This "standard" functionality enables the usage of this processor for general purpose applications. The application specific functionality enables faster execution of specific operations.

7.2.2 Application specific hardware

Following the partitioning result from the previous chapter (see figure 6.3) the following application specific functionality has to be implemented:

- MFSK *data-detection*
  In the prompt-path the MFSK data-detection takes place (functions *rTwiddle*, *resetac*, *corr* and *readRes*). From the figure it is clear that these operations are implemented in hardware.

- PN-CODE-*generation*
  All operations related with the PN-CODE-generation (functions *resetPNgen*, *shiftPNgen*, *getPrompt*, *getEL* and *phaseShift*) are implemented in hardware as well.

By observing figure 6.3 in more detail it can be concluded that not all "dark" (= hardware) blocks are mentioned. Operations related with the despreading operation and the tracking-buffer can be more efficiently mapped on hardware outside the processor-framework (not as functional units). They will be implemented accordingly.

Evaluating WISSCE's operation and considering the issues above leads to the following set of application specific functional units:

1. A DFT-CE which performs the MFSK data-detection. The functions to be implemented on such a unit are: *corr*, *corrTr*, *resetac* and *readRes*.

2. A PN-CODE GENERATOR which implements all functionality related to the PN-CODE-generation.

   To enable also transmission of data, this FU also includes a second PN-CODE GENERATOR. As the transmission task is less computationally intensive than the receiving task, it will not be a problem including this task in this FU as well.

   Furthermore this application specific FU is also responsible for the generation of the "start-symbol" signal. This signal is an important control signal, it initiates the processing of a new data-symbol and is used by various other functions implemented in both hardware and software.

3. The third application specific functional unit contains the frequency-hopping synthesizers needed for frequency-hopping spreading and despreading. The functionality of these synthesizers did not appear in the graph as motivated in section 6.2.1 on page 113.

4. A multiplier/accumulate circuit which is used in the calculation of the powers in the different MFSK-CHANNELS. While deriving software costs it was assumed that a multiplier was available in the processor architecture. As WISSCE uses square operations wherever a multiplication appears, a multiplier/accumulate circuit was considered as being a more efficient choice.

5. Finally an input-data conversion FU is required to convert the serial input data-stream to a symbol representation.

7.2.2.1 MFSK data-detection FU

Interfacing with the DFT-CE-FU is illustrated in figure 7.2. The unit consists of nine correlation-engines, eight to be used during data-detection and one for code-tracking purposes. All correlation engines are built from four counters which are responsible for counting the real and imaginary parts of the frequency-representation in two (mirrored) MFSK-CHANNELS (see section 5.2). The design of this unit is described in [Tek96].

At a symbol-period reset (*Corr-Reset*) the contents of the 8 data-detection correlators (32 counters) is moved into buffers which can be read from the MOVE-bus. At that moment also the contents of the counters is preset to its initial value. These 32 counters operate in-line at the sample-frequency of 5.2 MHz (*Corr-CK*). Internal clock speeds however reach 20.8 MHz. After the first symbol-period these correlation results are valid.

Writing an address to the *corr-req* trigger register results in putting the contents of the addressed buffer in the output-buffer one cycle (latency=1) later.

If a number is written in trigger-register *corr-track*, a correlation-operation starts in the tracking-correlator at processor-speed. The TWIDDLE-FACTORS corresponding with the requested number are used for this correlation, in this way the appropriate MFSK-CHANNEL can be selected. When the correlation is finished, a flag is set to notify the processor that the results can be read. The value of the flag can also be read via the *corr-req* register.

Reading the count-values from the tracking-correlator is done in the same way as reading the contents of the other counters.

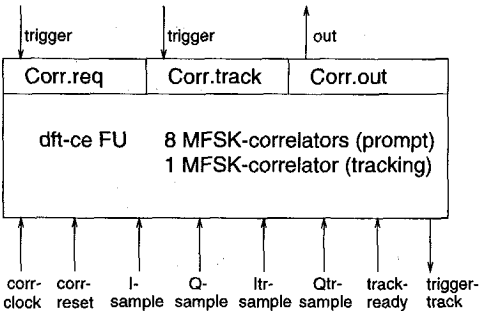


Figure 7.2. Interfacing with the DFT-CE-FU

There are 2 more control-signals present: *trigger-track* and *track-ready*. The first signal is an FU-output to the tracking-buffer that indicates that a tracking-correlation run can start. The second signal is an input signal to the FU that indicates that all tracking-data is sent.

7.2.2.2 Code generation FU

The interfacing with the PN-CODE GENERATOR-FU is shown in figure 7.3 whereas a description of the design can be found in [NC96]. The unit consists of three parts:

1. A PN-CODE GENERATOR (Kasami-code generator Rx) to perform the de-spreading (in the receiver chain). This unit generates a prompt-code signal and an early-late signal.
2. A PN-CODE GENERATOR (Kasami-code generator Tx) to perform spreading (in the transmitter chain). The prompt-code is generated with the speed of the transmitter chip-frequency.
3. A clock-circuit (see also section 5.3.2), that can be controlled to enable code-tracking.

All three parts have their own trigger-register. Once a value arrives at the trigger-registers PN.Tx, PN.Rx or PN.cntr, this value is stored in the socket to be used during the next symbol-period. There is also a fourth trigger-register (PN.req). If a number is written to this register, one cycle later the status of "start-symbol" flag appears in the output-register: if a new symbol started since the previous request, the status-bit is 1 otherwise 0.

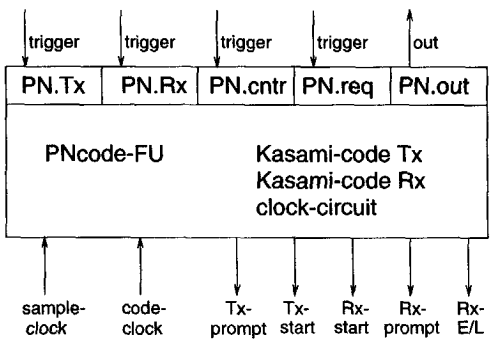


Figure 7.3. Interfacing with the PN-CODE GENERATOR-FU

7.2.2.3 DDS-FU

An outline of the interfacing with the FU that contains the Direct Digital Synthesizers (DDS) is given in figure 7.4. The operation of this particular DDS and its implementation can be found in [Hol94, Puc94].

The interfacing is as with the PN-CODE-FU: For both synthesizers there is a trigger-register in which the new frequency setting word (FSW) can be stored. Except for

a clock-input which is used to generate the output-frequencies, also two symbol-start inputs are available: one initiates a new carrier-frequency for the receiver, while the other does the same for the transmitter. A transmit-code input enables DS-spreading in this FU.

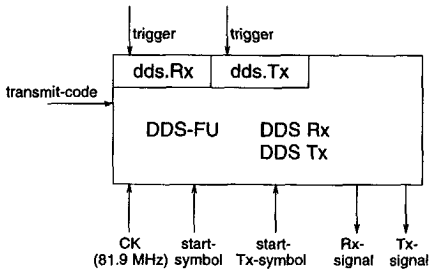


Figure 7.4. Interfacing with the DDS-FU

7.2.2.4 Squaring-FU

The Squaring-FU takes care of calculating energy (see section 5.2). To this end two numbers must be squared and accumulated. There are two trigger-registers: *sqr.mul* and *sqr.m_a*. The first register starts a square operation while the second also adds its result to the previous result (square and accumulate). The FU responsible for this operation is shown in figure 7.5, the design can be found in [Kli96].

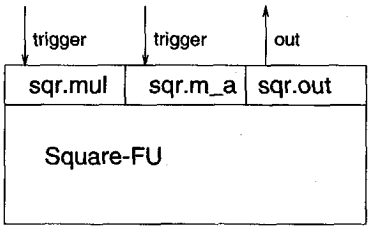


Figure 7.5. Interfacing with the Squaring-FU

7.2.2.5 Data-in FU

The data-in FU converts the serial input bit-sequence to a command for the transmit FH frequency synthesizer. The frequency setting word (FSW) that is actually moved to the transmit DDS using the dds.Tx register, is a combination of both the FH-frequency and the MFSK-frequency that corresponds to the symbol to be transmitted.

The interfacing of the data-in FU is outlined in figure 7.6. The FU contains a buffer that collects the incoming data-stream. On request, four bits are combined into a symbol representation that is put in the output-buffer.

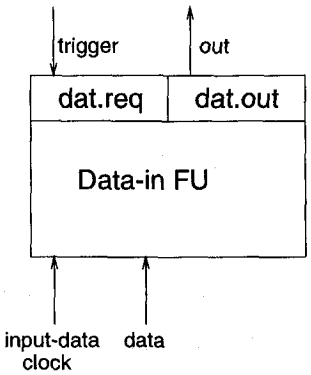


Figure 7.6. Interfacing with the data-in FU

7.2.3 Processor configuration

The complete processor configuration is shown in figure 7.7. In this figure the immediate-unit is included in the instruction fetch unit.

When comparing this MOVE-processor with figure 7.1 that showed an example of a transport triggered architecture, a number of differences can be observed. These are due to the fixing of a number of properties. We for instance selected a move-configuration with 2 busses as a compromise between area-usage and potential parallelism. Another issue is that the processor is "fully connected": all FUs are connected to all busses, this eases instruction scheduling. The bus-width is chosen equal to 16 because this number of bits is sufficient to represent all occurring data in the WISSCE-transceiver. An extra socket provides a way to output the recovered data-signal (4-bits in parallel).

7.3 Firmware design

An important property of an embedded system is the fact that it executes a single program over and over again. This program is usually stored in a kind of read only memory (ROM) like an eprom and will be referred to as "firmware". The previous chapter resulted in a split of the receiver-algorithm into a software and a hardware part. As a result it is known what functionality is implemented in the firmware.

For normal operation, a high-level firmware description is given in figure 7.8. The right hand side of this figure represents the firmware whereas the left hand side

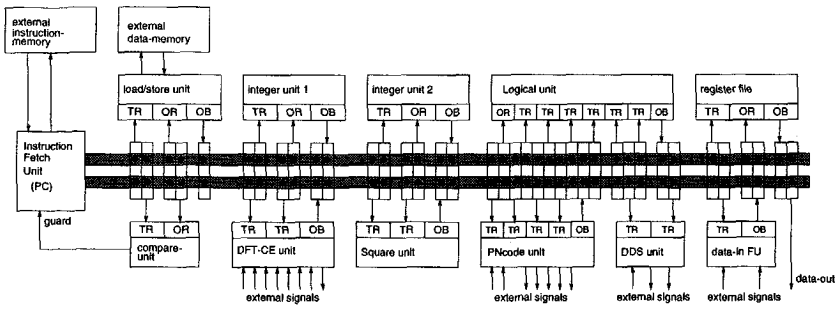


Figure 7.7. WISSCE-MOVE processor configuration

shows the operations performed in hardware. The sequence of operations is from top to bottom.

The figure shows the execution loop of a single symbol-period. A new symbol-period is initiated by the "start-symbol" signal generated in the PN-CODE-FU. This signal is used by both hardware and software.

After the "start-symbol" signal is generated, the tracking control signal is loaded from the socket into the local time reference (LTR). This leads to a code-phase step if required. At the same moment also the receive PN-CODE generator is switched to the desired PN-CODE and the contents of the tracking-buffer is swapped. The tracking buffer contains the samples of the previous symbol despread with the early-late code-signal.

The operations described in the previous paragraph can mainly be executed in parallel and they should be finished within a sample-period: There is no guard-time between the symbols to enable slowly processing of the data. Directly after the 260^{th} sample of symbol n, the first sample of symbol $n+1$ arrives.

After this, the DFT-FU will start a new correlation-stage. At the same time, correlation data from the previous symbol is read to calculate the energy in all MFSK-CHANNELS and their signs. During the calculation of the power-levels the firmware keeps track of the highest level and its position. In this way no extra stage is required to select the highest.

After deciding upon the received symbol, the tracking correlation stage can start in the DFT-FU. As an input the tracking-symbols from the previous symbol and the TWIDDLE-FACTORS corresponding to the detected symbol are used. In the meantime the firmware executes an acquisition-test to check whether the system is still in synchronization. The exact contents of this operation depends on the "acquisition state" at that moment (see figure 5.12 on page 89).

If the system is in synchronization, new data can be written to the PN-CODE-FU: the next code-words for the transmit and receive PN-CODE generator. Also new data can be moved to the DDS-FU: new frequency setting words (FSWs) for both

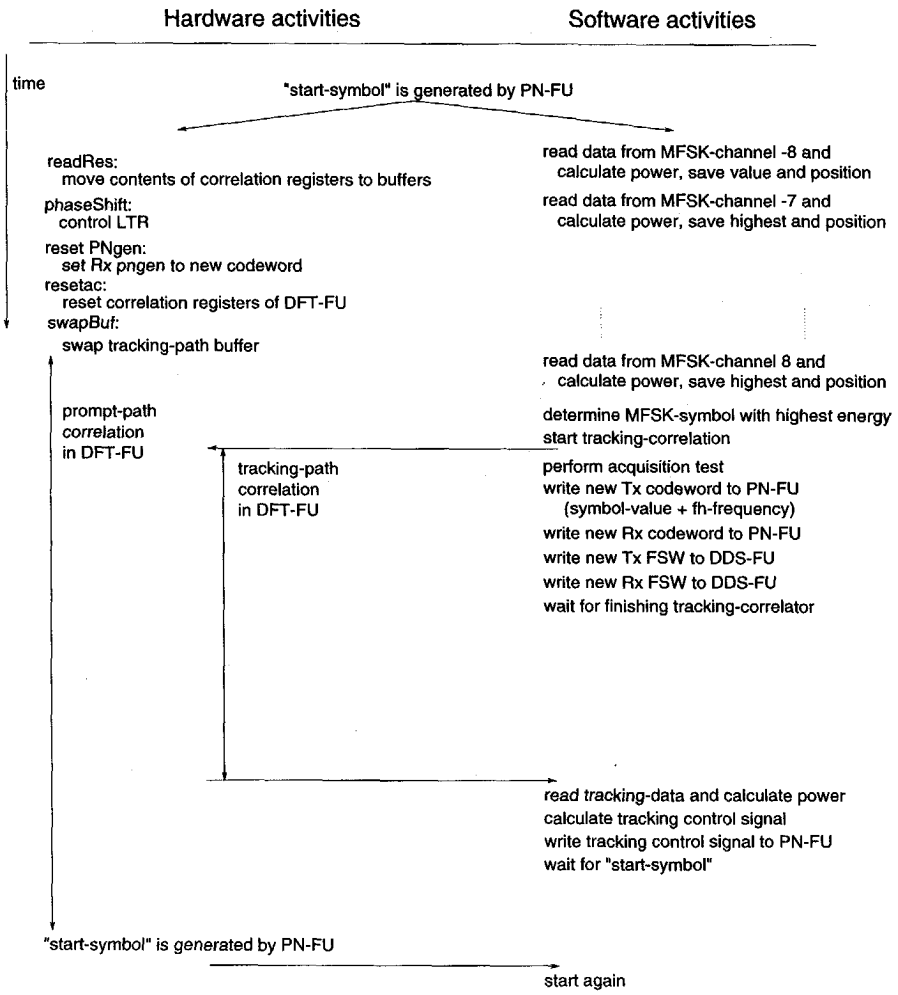


Figure 7.8. Firmware operation during normal operation

transmitter and receiver. After that, the firmware will go to an idle-state, waiting for the DFT-FU to finish.

The next task is to read the correlation results from the DFT-FU, and to calculate the power-level in the tracking-path. Using this answer it is possible to calculate the tracking-control data and to write that number to the PN-CODE-FU. When all operations are fulfilled, the processor goes to an idle-state and waits until a new "start-symbol" signal is generated.

If it turns out that the system is no longer in synchronization, operation will be different from next symbol on. An acquisition search is started. During this search which is described in section 5.3.3, operation is different: the tracking-path operations are not performed and the detected power-levels are used as decision variables in the acquisition process.

7.4 Co-Simulation

7.4.1 A co-simulation tool for the MOVE-processor

Co-simulation of hardware and software is the process of simulating the transport triggered architecture in its environment and running its firmware. The goal is to verify whether the hardware and software can work together without failure.

In this sense the goal of performing a co-simulation is different from the system-simulations that were discussed earlier. In system-simulation the target is to find out whether system-requirements are met: does the system meet the BER-requirements? does the synchronization have satisfactory results? etc. etc. Co-simulation should posses the following features:

- A check to see whether the combination of software and hardware provides functionality that meets its constraints. As a result, a co-simulation tool should have the means to simulate hardware, software and its interaction.
- When the operation of the embedded-system is not yet as desired, co-simulation runs should provide the means to perform user-friendly debugging. The consequence is that it must be possible to change both hardware and software easily.

Simulating both hardware and software leads to the requirement that the simulator must be able to handle different levels of abstraction. For instance, some parts of a processor are known to function correctly. For these parts simulations can be performed on a high level. The correctness of other parts of the system may be uncertain. For those parts a low-level simulation is required.

For simulation of an embedded-system based on a MOVE-architecture no appropriate standard tools existed. Consequently there was a need for a simulation platform that enabled co-simulation and was also extensible, as the MOVE-architecture had to be included.

PTOLEMY [BHLM94] provides a simulation environment that enables heterogeneous simulations while extensions are possible as the distribution package includes well-documented source code. Beside these facts, PTOLEMY was known to support co-simulation-like applications [KL95]. These were the reasons for choosing PTOLEMY as a platform to implement MOVE-based co-simulations [Nie96].

7.4.2 Simulation level

It was already stated that a co-simulation tools should be able to handle different levels of simulation. In building a co-simulation tool it is important to determine which part of an embedded system must be simulated at what level. Concerning this choice a number of requirements can be given:

1. The simulation-speed is an important parameter. The faster the simulation runs, the easier it is for the designer to obtain feedback. A simulation must be performed at a level as high as possible.
2. "standard" processor blocks that appear in all reasonable processor configurations can be simulated at a high level.
3. The exact configuration of a processor (functional units that are used, number of busses etc.) is dependent on the application and can also change during the design process. It is therefore important that this configuration can be changed in the simulator.
4. Application specific functionality should be simulated at a level that the designer considers to be appropriate. At different stages of the design process, a designer can have different opinions towards this simulation level.
5. It must also be possible to simulate the environment. In this way it becomes possible to test the system in its proper context.
6. The simulator must have notion of time: it should count the processor cycles.

Several documents exist that give a good overview of PTOLEMY and its operation [BHLM94, Dep96a, Dep]. For this reason we will not go into detail here. However, two important properties are worth mentioning:

1. Hierarchy is supported. A simulation run is always performed on a *universe*. A *universe* can be built out of *galaxies* and *stars*. A *star* is a "block"

containing C++-code that describes its operation, while a *galaxy* can exist of other *galaxies* and/or *stars*.

To summarize: A *universe* takes the top in the hierarchy while a *star* is the lowest in the hierarchy.

2. Different models of computation can be combined to enable heterogeneous simulations. Models are referred to as *domains*, we will use the synchronous data-flow (SDF) domain to perform rather high level simulations without notion of time, while the discrete-event (DE) domain will be used to perform simulations with notion of time.

Depending on the desired level of simulation, blocks can be implemented in:

- *DE-domain*
  To obtain low-level simulation results, with notion of time.
- *SDF-domain*
  To simulate at higher level where time does not play a role of importance.
- *C++*
  High-level blocks can be written as stars in C++. Simulating such a block is fast, on the cost of less flexibility.

Below we enlist the used simulation-levels and what simulation blocks are simulated on that level. A complete description of the implementation of a co-simulation tool in PTOLEMY can be found in [Nie96].

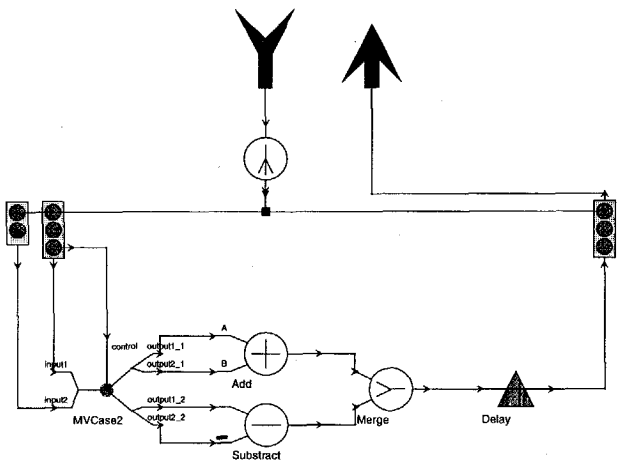


Figure 7.9. PTOLEMY representation of the integer-unit as a *galaxy*

- *DE-domain*
  As the complete simulation itself needs a notion of time, the universe containing the simulation model is built in the DE-domain. The same consideration holds for the MOVE-processor, this processor is implemented as a DE-galaxy. Also the functional-units are modelled as *galaxies* in the *DE-domain*. However within such a galaxy, stars or galaxies of an arbitrary domain can occur. An example is given in figure 7.9, this picture shows a galaxy implementing an integer functional-unit. The "add" and "subtract" blocks are *stars*.

- *SDF-domain*
  Some *galaxies* might be built in the *SDF-domain*. This domain can be used for fast prototyping as there is a large library of *SDF*-blocks available.

- *Implementation as Stars*
  Stars are written following special conventions which are based on the C++ programming language [Dep96b]. The advantage of stars is that the simulation goes fast, an disadvantage however is that the flexibility is low as stars have to be compiled and statically or dynamically linked into PTOLEMY. For these reasons, only "fixed" blocks are written as stars. Examples are the program counter (PC, see figure 7.11) and the sockets (connections to the bus).

7.4.3 Co-simulation

Now we discussed the co-simulation concept, we can concentrate on the simulation of the WISSCE receiver itself. The *universe* representing the "test-system" is given in figure 7.10.

At the left side the central clock is located, this is a 41.6 MHz clock. This clock is divided by 2 in the upper-block to obtain the simulation CPU-clock which is 20.8 MHz. The clock-signal is also divided by 8 to obtain the sample clocks, a delay-element takes care of a phase-shift between the I-sample clock and the Q-sample clock.

The sample-clocks are inputs to a transmitter-block. This transmitter generates an I-sample at the moment a trigger appears on the I-clock input. Analogously a Q-sample is created at the moment a trigger appears on the Q-clock input. Two samplers located after the transmitter take care of re-synchronizing the I and Q samples.

As the objective of a co-simulation tool is to check the cooperation of hardware and software, system-level issues like channel-properties are of no concern during these co-simulation runs. This is the reason that the transmitter block only implements the transmitter and not the channel.

After re-synchronization of the I and Q samples (for simulation purposes), despreading takes place. The upper-path is the "prompt-path" while the lower-path

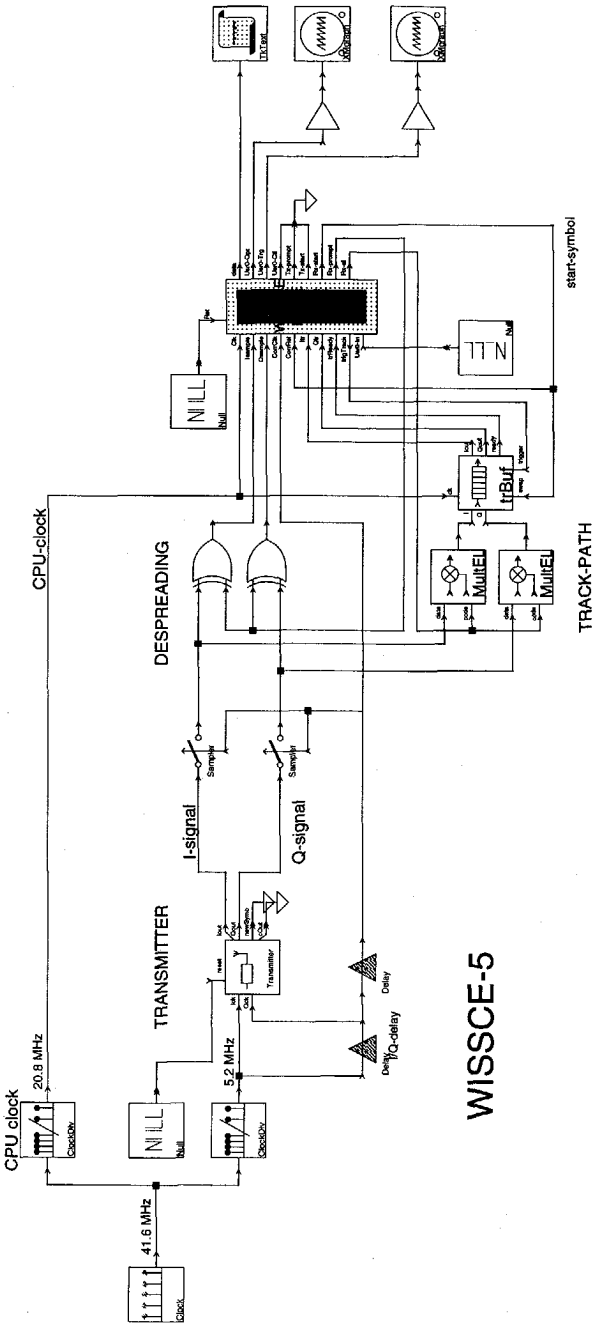


Figure 7.10. WISSCE-model in PTOLEMY

implements the "tracking-path" despreading. The code-sequences that are used in both paths are generated by the PN-CODE-FU in the processor. The despread prompt I/Q-samples are fed into the processor while the despread samples in the tracking-path are stored in the tracking-buffer. This buffer is controlled by the processor, if the processor is ready to perform the tracking-path processing a trigger signal is sent to the buffer to start transmission of the tracking-samples into the processor.

An important control-signal generated by the processor is the "start-symbol" signal. This signal resets the prompt-path processing (located in the processor but routed via an external line: *Corr-Reset*) and gives the command *swapBuf* to the tracking-buffer. This signal is also available in software.

As data-outputs the processor generates the "detected-symbol" and the "measured power-contents" during MFSK-detection. Those outputs are visualized using standard PTOLEMY blocks.

The contents of the MOVE-processor is shown in figure 7.11. At first sight the two vertical lines in the middle represent two busses. This is however not true. Where busses are bidirectional, these lines represent the input and output of all busses. A parameter associated with this bus-object specifies the bus-width.

The main block appearing in the processor is the instruction fetch unit, referred to as "PC" (program counter). This star is responsible of reading the firmware from a file and controlling the moves on the bus. The firmware is written in terms of these moves. Every line of code specifies source and destination addresses for both busses. Also guard signals can be used. Jumps in the program can be implemented by moving an address to the program counter.

From the figure we can also see that 3 application specific units are added: the PN-CODE-FU that generates the code-sequences, the MFSK-detector to recover the transmitted data-message and the Square unit which is used to calculate powers.

Additional sockets are available for an extra FU (usr0) and an output-socket is provided to pass the detected data to the outside-world.

7.4.4 Conclusions

In this section we described the way co-simulations are performed to evaluate the cooperation of hardware and software functionality. An important issue in building a co-simulation tool was the abstraction level on which simulations should be performed. By choosing PTOLEMY as a simulation platform it became possible to combine different levels of simulation to obtain both a fast and flexible tool.

The co-simulator was built with the objective to be used in this particular MOVE-architecture, therefore was tuned to this framework. As a result the simulator behaved as desired and was successfully applied in the design of WISSCE. To conclude this section an illustration of the simulator "in operation" is shown in figure 7.12

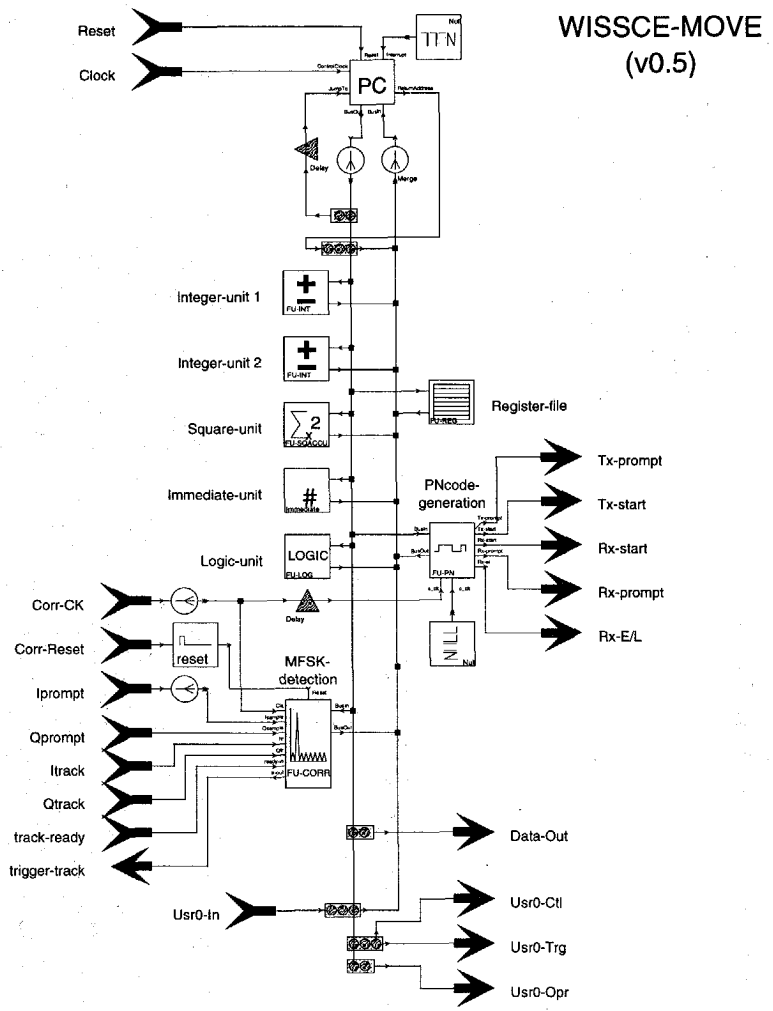


Figure 7.11. WISSCE-MOVE processor in PTOLEMY

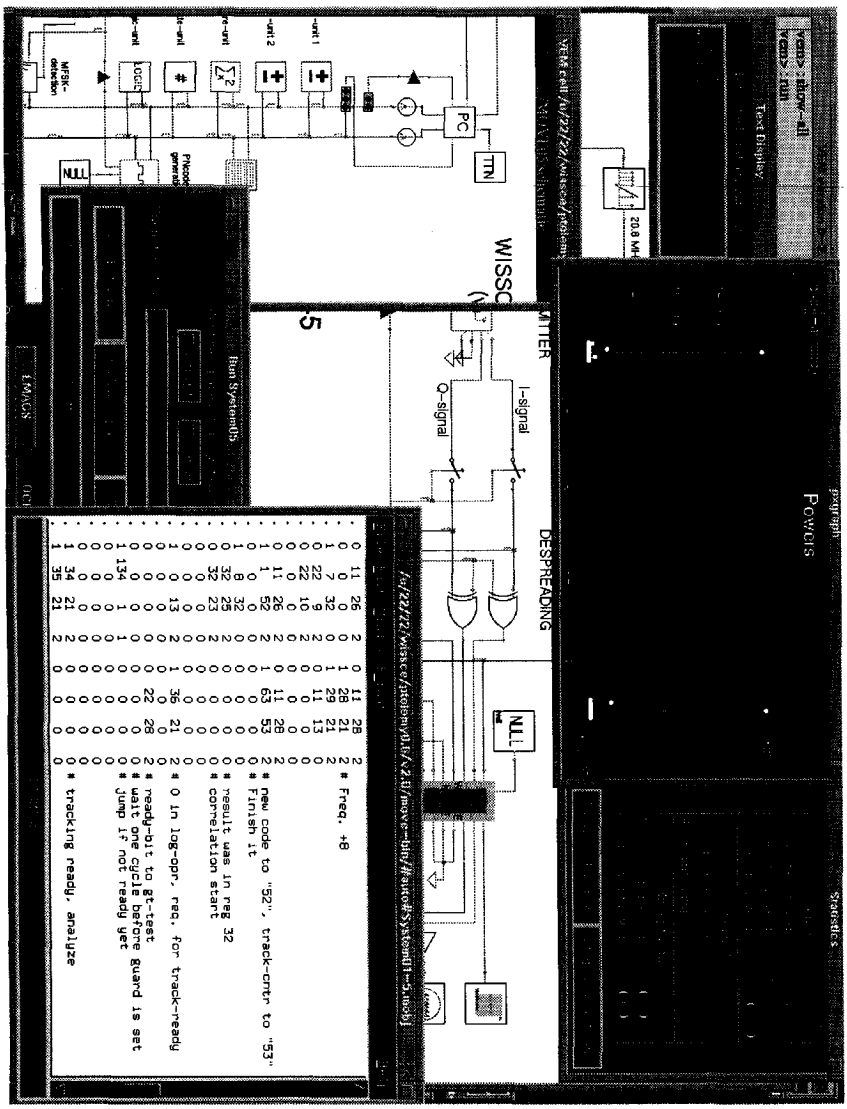


Figure 7.12. Co-simulation in process

7.5 Conclusions

On the basis of results gained from the previous chapters, this chapter addressed the design of the processor environment and the firmware to implement the baseband processing required for WISSCE.

We showed how the mapping of the hardware/software partition results on an transport triggered architecture can be done. For WISSCE, five application specific functional units are required. As the cooperation of hardware and software is an important issue, we also illustrated how the synchronization between the two parts is done.

After designing both the hardware and firmware, it is important to check whether the cooperation of the two parts is as desired. Co-simulations were performed using the simulation and code-generation framework PTOLEMY [BHLM94] was extended to enable co-simulations with a MOVE-processor.

Chapter 8

CONCLUSIONS

From the results presented in this thesis we conclude that systems like the digital baseband processing of WISSCE can be efficiently implemented using an embedded design methodology. The available resources were adequate and the transport triggered architecture provided enough flexibility to incorporate both general purpose functionality and dedicated hardware. To show how this conclusion follows from the thesis, we will stepwise go trough the design trajectory.

To keep the design space manageable, design decisions have to be fixed at the appropriate design stage. Decisions to be made at early design stages were the following:

- A multiple-access scheme forms the basis for the whole implementation. A specification of this technique at an early stage is therefore required. We showed that by combining two of the more common CDMA techniques, the high processing gain of direct-sequence is retained while near-far effects are effectively reduced due to frequency-hopping.
- After fixing the multiple-access scheme, this scheme should be properly embedded in a complete system specification. During the system definition however, numerous trade-offs situations presented itself. Market potential formulated as user demands and their relative importance were used as a guidance to arrive at sensible compromises.
- Another important consideration early in the system specification are the available resources to implement the system. The resources that were available to implement WISSCE's baseband processing clearly show that a complete software solution is precluded. To meet hard timing constraints, an embedded design methodology was therefore selected.
- The seamless cooperation of hardware and software implies a processor framework configurable in both its general purpose capability and dedicated functionality. An excellent option in this sense is the transport triggered architecture.

After fixing the multiple-access scheme, the system specification and the implementation concept to realize the digital processing of this system, the implementation stage can start. This is however far from straightforward! Standard solutions namely, do not always match with the available resources. "Intelligent algorithms" have to be found that combine simple implementations with acceptable losses. In this thesis this process was illustrated using two examples:

- A data-detection algorithm which mainly implements a simplified discrete fourier transform, is shown to be a good compromise between implementation complexity and SNR-loss.
- For code-synchronization a satisfactory synchronization performance can be obtained by applying an algorithm with low complexity.

The adequateness of both algorithms was confirmed by applying system-level simulation. The "program" used for these simulation runs was also used as an input to the hardware/software partitioning stage: the design stage in which the functionality of a system is partitioned in a hardware and a software part. During this partitioning stage we concluded the following:

- To search the enormous design space efficiently, a designer should be guided by an automatic tool that provides profiles on samples in the design space.
- Partitioning is based on cost-data of the various implementation alternatives. Finding precise cost-data however appears to be impossible: at the moment HW/SW-partitioning takes place this data is just not available. Providing guesses on this cost-data could easily lead to inefficient designs or even results that are outside the specifications. This problem was avoided by applying HSPART: a HW/SW partitioning tool able to cope with imprecise input-data.

Mapping the hardware functionality onto functional units in a transport triggered architecture is the next step in the design process. For WISSCE, five application specific functional units have been designed.

Before realizing the actual hardware, a check is required to find whether the hardware and software parts cooperate as desired. For this purpose PTOLEMY was extended to enable co-simulation of a transport triggered architecture. Applying the tool for WISSCE's baseband processing showed that valuable feedback can be obtained from co-simulation runs.

After obtaining satisfactory co-simulation results, the transport triggered architecture, including application specific parts, can be realized. For WISSCE, the baseband processing has been allocated on the target Sea-of-Gates chip. Software

is put in an eprom next to the WISSCE-chip. Some hardware functionality had to be implemented outside the processor framework as well.

The overall conclusion is that building the digital baseband processing of a transceiver for a non-cellular short distance wireless communication system as an embedded system is possible on the available resources without violating specifications.

Appendix A

PSEUDO-RANDOM NOISE SEQUENCES

This appendix provides a list with code-sequences used in WISSCE together with a description of the translation step necessary for obtaining these codes.

A.1 Selected Code-set

As described in section 4.4, the code-set to be used is a subset of the large set of Kasami-codes with length 63. The Kasami-code set used is orders in increasing order of $R_i^{(K)}$, see (4.37). The Kasami-codes are build of 3 M-sequences: the u-code: M(6,1), the v-code: M(6,5,2,1) and the w-code: M(3,2). The notation is as defined in (4.36).

There are 520 sequences in the large set of Kasami-codes. From these sequences there are 241 balanced, those are ordered in increasing order of $R_i^{(K)}$ in table A.1.

Table A.1. Selected Kasami-sequences

index	sequence	$R_i^{(K)}$	index	sequence	$R_i^{(K)}$	index	sequence	$R_i^{(K)}$
0	(6,28,7)	40958	1	(6,17,6)	41077	2	(6,15,4)	41166
3	(6,8,3)	41285	4	(6,30,7)	41293	5	(6,26,1)	41295
6	(6,4,6)	41315	7	(6,31,5)	41379	8	(6,39,6)	41429
9	(6,3,5)	41472	10	(6,27,7)	41602	11	(6,1,4)	41690
12	(6,42,7)	41873	13	(6,58,7)	41924	14	(6,46,0)	41940
15	(6,34,7)	42059	16	(6,26,0)	42170	17	(6,60,0)	42208
18	(6,41,5)	43104	19	(6,55,5)	43807	20	(6,53,3)	43906
21	(6,21,1)	44084	22	(6,22,0)	44197	23	(6,51,3)	44208
24	(6,37,1)	44265	25	(6,18,5)	44319	26	(6,2,3)	44333
27	(6,49,6)	44335	28	(6,56,6)	44335	29	(6,35,1)	44341
30	(6,44,1)	44348	31	(6,15,2)	44509	32	(6,14,1)	44544
33	(6,13,0)	44555	34	(6,52,2)	44565	35	(6,20,5)	44688
36	(6,16,3)	44699	37	(6,6,5)	44814	38	(6,14,6)	44847

continued on next page

continued from previous page

index	sequence	$R_i^{(K)}$	index	sequence	$R_i^{(K)}$	index	sequence	$R_i^{(K)}$
39	(6,62,0)	44881	40	(6,61,6)	44917	41	(6,32,5)	44925
42	(6,59,4)	44953	43	(6,57,0)	44954	44	(6,17,4)	44977
45	(6,62,5)	45026	46	(6,8,2)	45052	47	(6,50,0)	45055
48	(6,11,5)	45062	49	(6,51,1)	45074	50	(6,11,3)	45077
51	(6,38,4)	45108	52	(6,45,2)	45129	53	(6,3,2)	45131
54	(6,0,6)	45147	55	(6,23,3)	45153	56	(6,9,3)	45188
57	(6,56,1)	45189	58	(6,49,1)	45245	59	(6,54,6)	45260
60	(6,25,5)	45289	61	(6,21,6)	45296	62	(6,29,0)	45304
63	(6,32,3)	45358	64	(6,40,4)	45420	65	(6,1,2)	45469
66	(6,3,7)	46197	67	(6,59,6)	46739	68	(6,21,2)	46778
69	(6,63,7)	46840	70	(6,19,7)	46851	71	(6,53,0)	46877
72	(6,37,5)	46888	73	(6,12,0)	46916	74	(6,0,2)	47012
75	(6,54,1)	47039	76	(6,4,0)	47124	77	(6,23,4)	47163
78	(6,60,7)	47165	79	(6,21,7)	47169	80	(6,54,7)	47191
81	(6,6,2)	47193	82	(6,9,4)	47236	83	(6,48,2)	47248
84	(6,3,6)	47251	85	(6,47,1)	47261	86	(6,2,5)	47267
87	(6,14,7)	47310	88	(6,0,7)	47310	89	(6,61,7)	47318
90	(6,10,7)	47337	91	(6,44,4)	47366	92	(6,36,3)	47377
93	(6,24,6)	47389	94	(6,28,2)	47394	95	(6,31,6)	47407
96	(6,46,7)	47412	97	(6,43,3)	47415	98	(6,30,4)	47458
99	(6,47,0)	47473	100	(6,55,2)	47476	101	(6,62,2)	47482
102	(6,48,1)	47483	103	(6,19,1)	47485	104	(6,58,5)	47486
105	(6,18,6)	47489	106	(6,45,6)	47496	107	(6,40,0)	47498
108	(6,33,0)	47512	109	(6,29,7)	47518	110	(6,5,0)	47520
111	(6,35,2)	47521	112	(6,57,7)	47522	113	(6,32,7)	47530
114	(6,33,1)	47541	115	(6,27,2)	47580	116	(6,38,6)	47592
117	(6,50,4)	47600	118	(6,56,3)	47606	119	(6,50,7)	47631
120	(6,42,3)	47651	121	(6,6,7)	47686	122	(6,47,7)	47702
123	(6,20,1)	47719	124	(6,37,4)	47741	125	(6,5,7)	47767
126	(6,14,3)	47824	127	(6,12,1)	47835	128	(6,21,3)	47842
129	(6,31,7)	47874	130	(6,5,1)	47950	131	(6,53,7)	47970
132	(6,29,3)	47975	133	(6,23,5)	48014	134	(6,34,1)	48019
135	(6,16,4)	48025	136	(6,50,3)	48030	137	(6,13,2)	48051
138	(6,54,0)	48064	139	(6,2,4)	48095	140	(6,10,6)	48109
141	(6,13,7)	48113	142	(6,2,7)	48126	143	(6,12,7)	48137
144	(6,40,1)	48142	145	(6,40,7)	48163	146	(6,51,4)	48169
147	(6,55,7)	48175	148	(6,19,0)	48201	149	(6,39,0)	48237

continued on next page

continued from previous page

index	sequence	$R_i^{(K)}$	index	sequence	$R_i^{(K)}$	index	sequence	$R_i^{(K)}$
150	(6,56,7)	48249	151	(6,30,5)	48255	152	(6,51,7)	48275
153	(6,22,4)	48277	154	(6,42,2)	48286	155	(6,10,5)	48292
156	(6,36,7)	48305	157	(6,49,7)	48307	158	(6,15,3)	48315
159	(6,57,3)	48315	160	(6,11,6)	48320	161	(6,49,2)	48353
162	(6,35,3)	48361	163	(6,38,7)	48370	164	(6,17,7)	48384
165	(6,18,7)	48409	166	(6,64,2)	48434	167	(6,37,7)	48473
168	(6,32,6)	48500	169	(6,41,7)	48507	170	(6,58,4)	48549
171	(6,24,7)	48582	172	(6,22,7)	48685	173	(6,0,3)	48686
174	(6,64,7)	48776	175	(6,43,0)	49674	176	(6,29,2)	49730
177	(6,19,4)	49806	178	(6,31,4)	49833	179	(6,5,6)	49851
180	(6,23,1)	50054	181	(6,9,1)	50221	182	(6,64,6)	50342
183	(6,18,3)	50441	184	(6,26,6)	50490	185	(6,57,2)	50500
186	(6,48,5)	50515	187	(6,0,1)	50517	188	(6,54,4)	50517
189	(6,42,6)	50611	190	(6,55,0)	50629	191	(6,28,1)	50639
192	(6,53,5)	50674	193	(6,60,3)	50682	194	(6,1,0)	50706
195	(6,8,0)	50741	196	(6,13,5)	50746	197	(6,24,4)	50751
198	(6,4,5)	50779	199	(6,36,0)	50805	200	(6,10,4)	50810
201	(6,60,5)	50874	202	(6,6,0)	50879	203	(6,64,1)	50917
204	(6,59,2)	50980	205	(6,5,4)	50984	206	(6,11,0)	51012
207	(6,22,2)	51095	208	(6,45,4)	51101	209	(6,15,0)	51104
210	(6,46,5)	51122	211	(6,39,3)	51158	212	(6,35,6)	51218
213	(6,25,3)	51317	214	(6,28,6)	51346	215	(6,33,6)	51347
216	(6,27,5)	51396	217	(6,58,1)	51399	218	(6,30,3)	51401
219	(6,38,2)	51426	220	(6,39,5)	51448	221	(6,3,4)	51491
222	(6,42,1)	51509	223	(6,44,3)	51624	224	(6,62,7)	52189
225	(6,52,7)	52889	226	(6,44,7)	53365	227	(6,14,2)	53376
228	(6,35,7)	53381	229	(6,9,5)	53385	230	(6,6,1)	53578
231	(6,20,2)	53713	232	(6,62,1)	53714	233	(6,38,5)	53834
234	(6,36,4)	53850	235	(6,41,2)	53900	236	(6,46,6)	53991
237	(6,1,7)	54028	238	(6,8,4)	54073	239	(6,4,7)	54224
240	(6,56,2)	54385						

A.2 Implementing the relative delays

In this section we will provide a translation table which can be used to convert a relative delay of the v or w sequence to a tap-selection word. The theory behind this translation was described in section 4.4.1.3, and illustrated in figure 4.7.

A.2.1 M(6,5,2,1)-sequence

In table A.2 is shown how relative delays in the v-code can be obtained. All shifts are relative to the all-ones state. A "1" indicates that the tap-output is used, a "0" indicates the opposite.

Table A.2. Tap-selection of u-sequence M(6,5,2,1)

shift	tap: 5	4	3	2	1	0	shift	tap: 5	4	3	2	1	0
0	0	0	0	0	0	1	1	0	0	0	0	1	0
2	0	0	0	1	0	0	3	0	0	1	0	0	0
4	0	1	0	0	0	0	5	1	0	0	0	0	0
6	1	1	0	0	1	1	7	0	1	0	1	0	1
8	1	0	1	0	1	0	9	1	0	0	1	1	1
10	1	1	1	1	0	1	11	0	0	1	0	0	1
12	0	1	0	0	1	0	13	1	0	0	1	0	0
14	1	1	1	0	1	1	15	0	0	0	1	0	1
16	0	0	1	0	1	0	17	0	1	0	1	0	0
18	1	0	1	0	0	0	19	1	0	0	0	1	1
20	1	1	0	1	0	1	21	0	1	1	0	0	1
22	1	1	0	0	1	0	23	0	1	0	1	1	1
24	1	0	1	1	1	0	25	1	0	1	1	1	1
26	1	0	1	1	0	1	27	1	0	1	0	0	1
28	1	0	0	0	0	1	29	1	1	0	0	0	1
30	0	1	0	0	0	1	31	1	0	0	0	1	0
32	1	1	0	1	1	1	33	0	1	1	1	0	1
34	1	1	1	0	1	0	35	0	0	0	1	1	1
36	0	0	1	1	1	0	37	0	1	1	1	0	0
38	1	1	1	0	0	0	39	0	0	0	0	1	1
40	0	0	0	1	1	0	41	0	0	1	1	0	0
42	0	1	1	0	0	0	43	1	1	0	0	0	0
44	0	1	0	0	1	1	45	1	0	0	1	1	0
46	1	1	1	1	1	1	47	0	0	1	1	0	1
48	0	1	1	0	1	0	49	1	1	0	1	0	0
50	0	1	1	0	1	1	51	1	1	0	1	1	0
52	0	1	1	1	1	1	53	1	1	1	1	1	0
54	0	0	1	1	1	1	55	0	1	1	1	1	0
56	1	1	1	1	0	0	57	0	0	1	0	1	1
58	0	1	0	1	1	0	59	1	0	1	1	0	0
60	1	0	1	0	1	1	61	1	0	0	1	0	1
62	1	1	1	0	0	1							

A.2.2 M(3,2)-sequence

Table A.3 shows how relative delays in the *w*-code can be obtained. The length of the code is 7, while the number of shiftregisters (and so the number of available taps) is equal to 3.

Table A.3. Tap-selection of *w*-sequence M(3,2)

shift	tap: 2	1	0
0	0	0	1
1	0	1	0
2	1	0	0
3	0	1	1
4	1	1	0
5	1	1	1
6	1	0	1

Bibliography

[Bai95] Rupert Baines. The dsp bottleneck. *IEEE Communications Magazine*, 33(5):46–54, May 1995.

[Bea75] K.G. Beauchamp. *Walsh Functions and their Applications*. London: Acadamic Press, 1975.

[BHLM94] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogeneous systems. *Int. Journal of Computer Simulation*, pages 155–182, April 1994.

[BISH96] Nguyen Ngoc Bình, Masaharu Imai, Akichika Shiomi, and Nobuyuki Hikichi. A hardware/software partitioning algorithm for designing pipelined asips with least gate counts. In *Proceedings of the Design Automation Conference*, pages 527–532, Las Vegas, USA, June 1996.

[BMH89] Martin Bloch, Marvin Meirs, and John Ho. The microprocessor compensated crystal oscillator (MCXO). In *Proceedings of the frequency-control symposium*, page 16, 1989. index: 6-43-16.

[BMS89] Robert J.C. Bultitude, Samy A. Mahmoud, and William A. Sullivan. A comparison of indoor radio propagation characteristics at 910 mhz and 1.75 ghz. *IEEE Journal on Selected areas in Communications*, 7(1):20–30, Januari 1989.

[Bul87] Robert J. C. Bultitude. Measurement, characterization and modeling of indoor 800/900 mhz radio channels for digital communications. *IEEE Communications Magazine*, 25(6):5–12, June 1987.

[Cap95] P. Cappelletti. *The STONE user's manual*. Delft University of Technology, 1995.

[CM91] Henk Corporaal and Hans J. M. Mulder. Move: A framework for high-performance design. In *the Proceedings of Supercomputing '91*, pages 692–701, Albaquerque, November 1991.

[Com96] Peter Combee. Front-end for WISSCE. Master's thesis, Delft University of Technology, Delft, The Netherlands, 1996. number: 96-A?

[Cor95] Henk Corporaal. *Transport Triggered Architectures, Design and Evaluation.* PhD thesis, Delft University of Technology, September 1995. ISBN: 90-9008662-5.

[Dep] Department of EECS, University of California, Berkeley. Welcome to the ptolemy project. `http://ptolemy.eecs.berkeley.edu`.

[Dep96a] Department of EECS, University of California, Berkeley. *The Almagest: Vol.I - Ptolemy 0.6 User's Manual*, 1996.

[Dep96b] Department of EECS, University of California, Berkeley. *The Almagest: Vol.II - Ptolemy 0.6 Programmer's Manual*, June 1996.

[Dix84] R. C. Dixon. *Spread Spectrum Systems.* Wiley, New York, second edition, 1984.

[EHB93] Rolf Ernst, Jörg Henkel, and Thomas Benner. Hardware-software cosynthesis for microcontrollers. *IEEE Design & Test of Computers*, pages 64–75, December 1993.

[EKB87] Said E. El-Khamy and Ahmed S. Balamesh. Selection of gold and kasami code sets for spread spectrum cdma systems of limited number of users. *International Journal of Satellite Communications*, 5:23–32, 1987.

[FdD86] M. Führen and R. C. den Dulk. A new despreading method based on sub-nyquist sampling. In *Signal Processing III: Theory and Applications*, page A2.4. Elsevier Science Publishers, 1986.

[GCD94] Rajesh K. Gupta, Claudionor N. Coelho Jr., and Giovanni De Micheli. Program implementation schemes for hardware-software systems. *Computer*, pages 48–55, January 1994.

[Ger85] E. A. Geraniotis. Performance of noncoherent direct-sequence spread-spectrum multiple-access communications. *IEEE Transactions on Selected Areas in Communications*, SAC-3(5):687–694, September 1985.

[Ger86] E. A. Geraniotis. Noncoherent hybrid ds-sfh spread spectrum multiple access communications. *IEEE Transactions on Communications*, COM-34(9):862–872, September 1986.

[Gla] Jack P.F. Glas. The principles of spread spectrum communication. WWW-page: `http://cas.et.tudelft.nl/~glas/ssc/techn`.

[Gla92] Jack P.F. Glas. Definitie en synchronisatie van een ds ffh spread spectrum communicatie systeem voor de korte afstand. Master's thesis, Delft University of Technology, Faculty of Electrical Engineering, Delft, The Netherlands, June 1992. in dutch.

[Gla94] Jack P. F. Glas. On multiple access interference in a ds/ffh spread spectrum communication system. In *the Proceedings of the Third IEEE International Symposium on Spread Spectrum Techniques and Applications*, pages 3–2, Oulu, Finland, July 1994.

[Gla95] Jack P.F. Glas. Codesign in a cdma-receiver. In *The proceedings of the workshop on High Level Synthesis Algorithms, Tools and Design (Hiles)*, Stanford University, November 1995.

[Gli91] Savo G. Glisic. Automatic decision threshold level control in direct-sequence spread-spectrum systems. *IEEE Transactions on Communications*, COM-39(2):187–92, February 1991.

[Gol67] S. W. Golomb. *Shift Register Sequences*. Holden-Day Inc., San Francisco, California, 1967.

[GS93] Patrick Groeneveld and Paul Stravers. *OCEAN: The Sea-of-Gates Design System*. Delft University of Technology, 1993.

[GS94] J. P. F. Glas and S. E. Skolnik. Fourier transform based ds/fh spread spectrum receiver. In *the Proceedings of the International Conference on Computer Design '94*, Cambridge, Massachusetts, October 1994.

[Has93] Homoyoun Hashemi. The indoor radio propagation channel. *Proceedings of the IEEE*, 81(7):943–968, July 1993.

[HdV71] G. Hoffmann de Visme. *Binary Sequences*. The English Universities Press Ltd., 1971.

[Hen84] C. G. Henning. Spread spectrum test criteria and methods. In *Proceedings of AUTOTESTCON '84*, pages 48–55, Washington D.C., November 1984. AUTOTESTCON.

[Hol82] J. K. Holmes. *Coherent Spread Spectrum Systems*. Wiley, New York, 1982.

[Hol94] Mark Holtkamp. Direct digital synthesizers voor fast frequency hop spread spectrum communicatie. Master's thesis, Delft University of Technology, Faculty of Electrical Engineering, Electronic Engineering Group, Delft, The Netherlands, April 1994. in dutch, number: 94-A4.

[Hoo96] Jan Hoogerbrugge. *Code Generation for Transport Triggered Architectures*. PhD thesis, Delft University of Technology, February 1996. ISBN: 90-9009002-9.

[HP90] S.J. Howard and K. Pahlavan. Performance of a dfe modem evaluated from measured indoor radio multipath profiles. In *Proceedings of the ICC'90*, volume 4, page 335.2, Atlanta, April 1990.

[HT94] Homayoun Hashemi and David Tholl. Statistical modeling and simulation of the rms delay spread of indoor radio propagation channels. *IEEE Transactions on Vehicular Technology*, 43(1):110–120, February 1994.

[JP92] Gerard J.M. Janssen and Ramjee Prasad. Propagation measurements in an indoor radio environment at 2.4 ghz, 4.75 ghz and 11.5 ghz. In *Proceedings of the 42nd VTS conference frontiers of Technology*, pages 617–620, Denver, Colorado, May 1992.

[Kar95] I. Karkowski. *Performance Driven Synthesis of Digital Systems*. PhD thesis, Delft University of Technology, December 1995. ISBN: 90-5326-022-6.

[KL95] Asawaree Kalavade and Edward A. Lee. Hardware/software codesign using ptolemy. In Jerzy Rozenblit and Klaus Buchenrieder, editors, *Codesign, computer-aided software/hardware engineering*, chapter 19, pages 397–413. IEEE Press, 1995.

[Kli96] A.H. van Klinken. A move functional unit for calculating signal powers by squaring and accumulating numbers. taakverslag, Delft University of Technology, Circuits and Systems group, 1996.

[KO95] I. Karkowski and R.H.J.M. Otten. Uncertainties of hardware-software co-synthesis of embedded systems. In *The proceedings of the workshop on High Level Synthesis Algorithms, Tools and Design (Hiles)*, Stanford University, November 1995.

[KS95] Joseph Kennedy and Mark C. Sullivan. Direction finding and "smart antennas" using software radio architectures. *IEEE Communications Magazine*, 33(5):62–68, May 1995.

[Lin72] W.C. Lindsey. *Synchronization Systems in Communication and Control*. Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[Mil75] K.S. Miller. *Multivariate Distributions*. R.E. Krieger Publishing Co., Inc., 1975.

[Mit95] Joe Mitola. The software radio architecture. *IEEE Communications Magazine*, 33(5):26–38, May 1995.

[MT92] Svetislav V. Marić and Edward L. Titlebaum. A class of frequency hop codes with nearly ideal characteristics for use in multiple-access a spread-spectrum communications and radar and sonar systems. *IEEE Transactions on Communications*, COM-40(9):1442–1447, September 1992.

[NC96] Mark Nitters and Peter Combee. Vhdl-beschrijving van pncode generation. Technical report, Delft University of Technology, Circuits and Systems group, August 1996. in dutch.

[Nie96] Jan M.G. Nieuwstad. Hardware-software co-simulation of move processors using ptolemy. Master's thesis, Delft University of Technology, Delft, The Netherlands, 1996. number: 96-A1.

[Nik95] Homayoun Nikookar. *Wireless Channel Modeling and Code Division Multiple Access for Indoor Communications*. PhD thesis, Delft University of Technology, October 1995. ISBN: 90-9008709-5.

[PG94] Andreas Polydoros and Savo Glisic. Code synchronization: A review of principles and techniques. In *Proceedings of the ISSSTA '94*, pages 115–137, Oulu, Finland, July 1994. IEEE.

[PMK90] R Prasad, H. S. Misser, and A Kegel. Performance analysis of direct-sequence spread-spectrum multiple-access communication in an indoor rician-fading channel with dpsk modulation. *Electronics Letters*, 26(17):1366–7, August 1990.

[Pro89] J.G. Proakis. *Digital Communications*. McGraw-Hill Book Company, New York, second edition, 1989.

[PS77] M. B. Pursley and D. V. Sarwate. Performance evaluation for phase coded spread spectrum multiple acces communication - part ii: Code sequence analysis. *IEEE Transactions on Communications*, 25(8):800–3, August 1977.

[PSM82] Raymond L. Pickholtz, Donald L. Schilling, and Laurence B. Milstein. Theory of spread spectrum communications - a tutorial. *IEEE Transactions on Communications*, COM-30(5):855–884, May 1982.

[Puc94] Giacomo Puccio. Layout design of a direct digital frequency synthesizer as a frequency dehopper for a spread spectrum communication system on sea-of-gates. Master's thesis, Delft University of Technology, Delft, The Netherlands, July 1994. number: 94A5.

[Pur77] M. B. Pursley. Performance evalutation for phase coded spread spectrum multiple acces communication - part i: System analysis. *IEEE Transactions on Communications*, 25(8):795–9, August 1977.

[Q-T94] Q-Tech Corporation. Microcomputer compensated crystal oscillators, 1994.

[Qua92] An overview of the application of code division multiple acess (cdma) to digital cellular systems and personal cellular networks. Qualcomm Inc., 1992.

[Reg] Loek K. Regenbogen. private communication.

[Roe77] H. F. A. Roefs. *Binary Sequences for Spread-Spectrum Multiple-Access Communications*. PhD thesis, University of Illionois, Urbana, Illinois, 1977.

[Sch94] Robert A. Scholtz. The evolution of spread spectrum multiple-access communications. In *the Proceedings of the Third IEEE International Symposium on Spread Spectrum Techniques and Applications*, pages 4–13, Oulu, Finland, July 1994.

[SH85] R. Skaug and J. F. Hjelmstad. *Spread Spectrum in Communication*. IEE Telecommunications series. Peter Peregrinus Ltd., London, UK., 1985.

[SOSL85a] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt. *Spread Spectrum Communications*, volume I of *Electrical Engineering Communications and Signal Processing*. Computer science press, Inc., Rockvill, Maryland 20850, 1 edition, 1985.

[SOSL85b] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt. *Spread Spectrum Communications*, volume III of *Electrical Engineering Communications and Signal Processing*. Computer science press, Inc., Rockvill, Maryland 20850, 1 edition, 1985.

[SP80] D. V. Sarwate and M. B. Pursley. Crosscorrelation properties of pseudorandom and related sequences. In *Proceedings of the IEEE, Vol.68*, pages 593–619. IEEE, May 1980.

[Str94] Paul Stravers. *Embedded System Design*. PhD thesis, Delft University of Technology, December 1994. ISBN: 90-9007-879-7.

[Tek96] Cenk Tekin. Design and implementation of an fsk-correlation engine for data detection and tracking for WISSCE. Twaio-report, Delft University of Technology, Delft, The Netherlands, January 1996.

[TSV94] Markus Theißinger, Paul Stravers, and Holger Veit. Castle: An interactive environment for hw-sw co-design. In *Proceedings of the International Workshop on hardware-Software Codesign*, pages 203–209, Grenoble, September 1994.

[Tur84] G. L. Turin. The effects of multipath and fading on the performance of direct-sequence cdma systems. *IEEE Transactions on Vehicular Technology*, VT-33(3):213–9, August 1984.

[Wan91] J. Wang. *Spread Spectrum Multiple Access with Diversity and Coding for Indoor Radio*. PhD thesis, State University of Ghent, Communications Engineering Laboratory, 1991.

[WDW94] Nam S. Woo, Alfred E. Dunlop, and Wayne Wolf. Codesign from cospecification. *Computer*, pages 42–46, January 1994.

[WIS] WISSCE-team, Circuits and Systems group. WISSCE home-page. WWW-page: `http://cas.et.tudelft.nl/~wissce`.

[WM92] Jiangzhou Wang and Marc Moeneclaey. Hybrid ds/sfh-ssma with predetection diversity and coding over indoor radio multipath rician-fading channels. *IEEE Transactions on communications*, COM-40(10):1654–1662, October 1992.

[YB82] Richard A. Yost and Robert W. Boyd. A modified pn code tracking loop: Its performance analysis and comparative evaluation. *IEEE Transactions on Communications*, COM-30(5):1027–36, may 1982.

[YLF94] Nathan Yee, Jean-Paul M.G. Linnartz, and Gerhard. Fettweis. Multi-carrier-cdma in indoor wireless network. *IEICE Transaction on Communications, Japan*, E77-B(7):900–904, July 1994.

[Zwa96] Wim Zwart. Clock for WISSCE. Master's thesis, Delft University of Technology, Delft, The Netherlands, September 1996. number: 96A6.

GLOSSARY

Symbols

α_k	suppression-factor of k^{th} zone
$(\Delta f)_c$	coherence bandwidth
$(\Delta t)_c$	coherence time
δ_{mod}	extra bandwidth-usage due to modulation
Δ_{FH}	FH-spacing in radial frequency
Δ_{FSK}	MFSK-spacing in radial frequency
γ	detection SNR
γ_p	pre-detection SNR
$\bar{\gamma}$	mean-γ value
ε_n	attenuation factor of n^{th} signal-path
ω_c	carrier radial frequency
$\rho_{k,i}$	measure for aperiodic cross correlation between codes k and i.
σ_n^2	noise power
$\sigma_{k,i;m}^2$	MA-interference-term, reference user i is interfered by user k in the m^{th} MFSK-CHANNEL
τ_n	propagation delay of n^{th} signal-path
$\theta(t)$	arbitrary phase
a	weigth-factor used to obtain 3-leveled TWIDDLE-FACTORS
B_{input}	input bandwidth (DS-spread signal)
BW_i	information bandwidth
BW_t	transmission bandwidth (DS-spread signal)
BW_{total}	total transmission bandwidth in one direction
B_{total}	Total Transmission bandwidth in one direction
$C_{k,l}(l)$	aperiodic cross correlation function of codes k and i with a discrete relative time delay of l chips
$c(t)$	binary pseudo-random noise sequence

CK_{DDS} DDS clock frequency

$CK_{processor}$ main processor clock

$CK_{sampling}$ sampling clock

$d(t)$ data symbol $\in \{-8,-7,\ldots,7,8\}\setminus\{0\}$

D disired power-term

E_b energy per bit

E_s energy per symbol

$F[n]$ frequency-representation of sampled signal.

$f_{3\text{-overtone}}$ 3^{rd} overtone frequency (used in MCXO)

f_c center-frequency

f_{beat} beat frequency (used in MCXO)

f_{FSK} FSK-channel

$f_{fundamental}$ fundamental frequency of crystal

f_i^n frequency-hopping code for user i during period n

f_{LO} local-oscillator frequency

f_{ref} referency frequency from which all clocks in transceiver are derived.

$f_{sampler\text{-}in}$ center-frequency of input signal to sampler

G_p processing gain

$\Im[n]$ Imaginary part of $F[n]$

$I_{k,i}(n)$ MA-interference term in the n^{th} timeslot if user k interferes reference user i.

$\mathrm{Ics}[n]$ ACCUMULATION-FACTOR, inproduct of in-phase signal component with cosine-TWIDDLE-FACTOR for frequency n

$\mathrm{Isn}[n]$ ACCUMULATION-FACTOR, inproduct of in-phase signal component with sine-TWIDDLE-FACTOR for frequency n

k k-shift value of Kasami-code (w-code shift)

k' k-shift value in tap-selection format

K number of active users in the system

K_{far} number of far-interferers

K_{near} number of near-interferers

K_{set} size of code-set

L_f loss-factor due to input-filtering

$L_t(\tau)$ correction-factor due to timing-misalingment

m m-shift value of Kasami-code (v-code shift)

m' m-shift value in tap-selection format

M number of MFSK channels

$N_0/2$ double sided noise density

N_{DS} direct-sequence (DS) code-length

N_{FH} frequency-hopping (FH) sequence-length

$n(t)$ noise signal with a two-sided noise spectral density of $N_0/2$

$P_{c,\,symbol}$ probability on correct detection of symbol

P_{hit} probability on two users using the same FH-channel

$P_{e,\,bit}$ probability on wrong detection of bit

$P_{h;\,FH}$ probability of 2 users having the same FH-SEQUENCE

P_{hit} over-all hit-probability

$p_{texthit}(k, K)$ probability that k interferers hit with the reference user if there are K active users.

$p_{\bar{s}}(u)$ PDF of the energy in channel that does not contain any deterministic signals

$p(\gamma)$ PDF of γ-value (in case of fading)

$p_s(u)$ PDF of the energy in channel that does contain a deterministic signal

$p_{hit}(k, K_{far})$ PDF of number of far-users that have a "hit" with the reference user.

q_{acq} total number of acquisition-cells to be searched

Qcs[n] ACCUMULATION-FACTOR, inproduct of quadrature signal component cosine-TWIDDLE-FACTOR for frequency n

Qsn[n] ACCUMULATION-FACTOR, inproduct of quadrature signal component with sine-TWIDDLE-FACTOR for frequency n

R Rice-factor

$R_i^{(}K_{set})$ cross-correlation cost function of code-set

$R_u(\tau)$ auto-correlation function

$\Re[n]$ real part of $F[n]$

$R_{k,i}(\tau)$ partial cross-correlation term

$\widehat{R}_{k,i}(\tau)$ partial cross-correlation term

$R'_{k,i}(\tau)$ partial cross-correlation term

$r(t)$ received signal

r_{symb} symbol-rate

S received signal power

$S(f)$ frequency-representation of received signal

T_{rms} RMS-delay spread

$TH_{textacq}$ threshold used in code-acquisition scheme

T_c time corresponding to a chip-period

T_s time corresponding to a symbol-period

t_{dwell} dwell-time

$tw_{\cos}[n,k]$ cosine TWIDDLE-FACTOR representing frequency n at time-step k

$tw_{\sin}[n,k]$ sine TWIDDLE-FACTOR representing frequency n at time-step k

$v_m(t)$ expected MFSK-signal in channel m

$Z_m(n)$ decision variable in the m^{th} MFSK-CHANNEL during the n^{th} symbol

Abbreviations

ASP	application-specific processor
ASP-FU	application-specific functional unit
BER	bit-error-rate
CELL	code-acquisition search position
CDMA	Code Division Multiple Access
DDS	Direct Digital Synthesizer
DIMES	Delft Institute of Micro Electronics and Submicron Technology
DS	direct-sequence
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
FDMA	Frequency Division Multiple Access
FH	frequency-hopping
FIFO	first-in first-out register
FSK	frequency shift keying
FSW	frequency setting word
FU	functional-unit
IC	integrated circuit
I&D-filter	integrate and dump filter
IFU	instruction fetch unit
LOS	line of sight
LTR	local time reference
MA	multi-access
MCTL	Modified Code Tracking Loop
MCXO	Microprocessor Controlled Crystal Oscillator
MFSK	Multiple Frequency Shift Keying
M-sequence	maximum-length sequence
MFSK-CE	MFSK correlation engine
PDF	probability density function
PN-CODE	pseudo-random noise code
RF	radio-frequency
RAM	random access memory
ROM	read only memory
RMS	root mean square
SYNC-CELL	acquisition search-CELL that corresponds to a situation in which the receiver is synchronized.
SOG	Sea-of-Gates

spd	spectral power density
SNR	signal to noise ratio
TDMA	Time Division Multiple Access
TTA	Transport Triggered Architecture
WISSCE	Wireless Indoor Spread-Spectrum Communication Equipment

Index

A/D-converter, 73
access time, 26
ACCUMULATION-FACTOR, 70, 71
acquisition search-space, 86
acquisition-trajectory, 93
ad-hoc communication links, 1
address-bus, 19
algorithmic description, 15, 16, 107, 109
amplification, 103
application specific hardware, 20, 123
area coverage, 25
ASP-FU, 123
autocorrelation, 60, 88

base-station, 1, 30
baseband processing, 14, 20, 33, 121
beat frequency, 36
BER, 2, 32, 33, 43, 53, 55, 72, 81, 82
bus width, 19, 128

C++, 133
C-description, 105, 107, 109
CASTLE, 20, 108, 109
CDMA, 3, 4, 103
CELL, 86
cellular systems, 1, 30
channel selection, 103
chi-square distribution
    central, 52, 55
    non-central, 52
clock-control, 35, 85
co-processor, 15
co-simulation, 16, 131, 134
code-misalignment, 44
code-selection, 57, 145
coherence bandwidth, 54
coherence time, 54
COSYMA, 20

data-detection, 32, 54, 69, 80, 81, 100, 124
data-speed, 2, 28
DDS, 37, 104
DDS-FU, 126
DECT, 3
design process, 16
DFT, 69
DFT-CE, 125
direct digital synthesizer, 124
domains, 133
    discrete-event, 133
    synchronous data-flow, 133
doppler-spread, 54
DS, 5, 30
DS-FH, 30
dwell-time, 86

early-path, 95
embedded system, 14

fading, 33, 53
    Rayleigh, 55
    Rician, 55
far-interference, 46, 49
FDMA, 3
FFT, 69

FH, 5, 7, 30
  fast, 8
  slow, 8
FH-sequence, 30
FH-SEQUENCE, 7, 63, 85
filtering, 104
firmware, 128
fishbone, 21
flexibility, 26
FOURIER-TRANSFORM, 69
frequency bands, 27
frequency translation, 103
frequency-plan, 28
front-end, 103
FSK, 32
FSW, 35, 126
FU, 18, 123
fuzzy number, 113

galaxy, 133
GSM, 3

hand-over, 1
hardware/software partitioning, 13, 17, 107, 116
hit-probability, 63
HSPART, 20, 108, 113, 116, 119

immediate-unit, 123
instruction parallelism, 18
instruction-fetch unit, 123
instruction-memory, 19
integer-unit, 123
Interference limited operation, 3, 45
IS-95, 3, 59
ISM-band, 27

late-path, 95
latency, 18, 122
logic-unit, 123
LTR, 95, 99, 129

MC-CDMA, 5
MCTL, 96
MCXO, 36, 41, 85
MFSK, 32, 84
MFSK-CHANNEL, 51, 52, 69
MINI-MOVE, 123
MINI-MOVE2, 123
mobility, 26
modulation scheme, 69
MOVE, 122, 136
multi-access, 2
multi-access interference, 45
multi-path, 33, 53

near-far effect, 6, 30, 63
near-interference, 50
non-cellular systems, 1, 24, 30

OCEAN, 21, 115
operation costs, 26
output power, 32

parallel search, 87
parallelism, 123
PN-CODES, 5, 30, 58
  Gold-codes, 5, 60
  Kasami-codes, 5, 60
  M-sequences, 5, 59
  Walsh Hadamard codes, 58
PN-CODE-generator, 124, 129, 136
PN-CODE GENERATOR-FU, 126
possibilistic data, 20, 113
power-control, 30
preferred pair, 60
processing gain, 4, 30
processor framework, 13, 18, 122
prompt-path, 95
protocol, 1, 24, 88
pseudo random noise sequence, 5
PTOLEMY, 132, 136
purchasing costs, 26

quadrature sampling, 34

random-access, 2
real-time systems, 14
register
  operand, 18

output, 18
trigger, 18
register-file, 19, 123
reliable operation, 25
Rice-factor, 55
RMS-delay spread, 54

sampling, 33
scheduler, 122
Sea-of-Gates, 21
select-highest search, 88
sequential search, 87
shift-and-add property, 60, 62
Shift-Register sequences, 59
simulation, 93, 94, 100, 131
SNR, 43
SOCKET, 18, 134
software radio, 33
spread spectrum, 4, 13, 30
square-law detection, 69
squash requests, 123
star, 132
supervisor, 109, 116
synchronization, 84
carrier, 84
code, 30, 85
acquisition, 86
tracking, 95

TDMA, 3
TH, 5
threshold-search, 87
timing constraints, 14
timing-misalignment, 44
tracking-curve, 96
transmission capacity, 25, 28
TTA, 18, 122
TWIDDLE-FACTOR, 70

universe, 132
user capacity, 25
user demands, 24

VULCAN, 20

WISSCE, 24, 57, 69, 73, 124

SAMENVATTING

Niet-Cellulaire Draadloze Communicatie Systemen

De vraag naar mobiele communicatiemiddelen stijgt met de mogelijkheden die door de technologie geboden worden. Dankzij moderne middelen is mobiele communicatie nu in vele situaties. Belangrijke componenten in deze mobiele communicatie vormen de zgn. niet-cellulaire draadloze communicatie systemen.

Dit proefschrift laat zien hoe moderne middelen en ontwerpmethoden kunnen worden toegepast in het ontwerp van dergelijke systemen. De nadruk ligt hierbij op de digitale basis-band bewerkingen van een transceiver. Hierbij wordt de "embedded system" ontwerpmethode behandeld. Deze methode wordt gebruikt om de traditionele barrière tussen hardware en software te verkleinen. De consequenties van de systeem-specificatie voor het front-end worden echter wel genoemd.

Tijdens de realisatie van een communicatiesysteem moeten vele beslissingen worden genomen. Een probleem hierbij is dat alle beslissingen op een of andere manier gerelateerd zijn. Het eerst verzamelen van alle afwegingen om daarna een optimaal systeem te configureren is daarom niet mogelijk. Keuzes moeten gemaakt worden op geschikte plaatsen in het ontwerptraject.

De keuze van een *multiple-access* techniek beïnvloedt alle hierop volgende stappen van het ontwerptraject en moet daarom in een vroeg stadium worden gemaakt. Het is bekend dat *Code Division Multiple Access* (CDMA) technieken goed passen bij de aard van niet-cellulaire communicatie systemen, iedere CDMA-techniek heeft echter specifieke nadelen. We laten zien dat door het combineren van een tweetal bekende technieken: *direct-sequence* en *frequency-hopping*, een hoge *processing gain* kan worden gecombineerd met een effectieve vermindering van het *near-far effect*.

De *multiple-access* techniek vormt de basis voor een volledige systeemspecificatie. Gedurende de systeemdefinitie verschijnen er steeds nieuwe afwegingen. Om hierbij tot zinvolle compromissen te komen, wordt markpotentieel in de vorm van gebruikerswensen geëvalueerd. We beschrijven de systeemspecificatie aan de hand van een praktijkvoorbeeld. Op deze manier wordt er duidelijk waar in

het ontwerp problemen bestaan. Dit voorbeeld heeft de naam WISSCE: "Wireless Indoor Spread Spectrum Communication Equipment".

Ook de beschikbare middelen om het systeem te implementeren vormen een belangrijke overweging. In vele tijd-kritische systemen zoals communicatie apparatuur, zijn software-implementaties voordelig vanwege hun flexibiliteit en het feit dat ze nauw aansluiten bij de operaties die uitgevoerd moeten worden. Er bestaan over het algemeen echter strenge eisen t.a.v. het tijdgebruik die een dergelijke implementatie uitsluiten. Een logische oplossing is dan het verplaatsen van tijd-kritische functionaliteit van software naar hardware; dit leidt tot een zgn. *embedded system*. Om een efficiënte implementatie mogelijk te maken moeten de software en hardware delen echter naadloos samenwerken. Dit betekent dat er een *processor framework* moet worden toegepast dat configureerbaar is met zowel standaard als toepassingsspecifieke functionaliteit. Een goed voorbeeld van een dergelijk *processor framework* is de *transport triggered architecture*.

Nu zowel het (communicatie) systeem als het implementatieconcept bekend is, gaan andere implementatie-aspecten een rol spelen. Standaard implementaties zijn gezien de beschikbare middelen niet mogelijk. "Slimme oplossingen" waarbij een reductie in complexiteit samengaat met een acceptabel verlies moeten daarom gevonden worden. Voor de basisbandbewerkingen in WISSCE wordt dit toegelicht aan de hand van een datadetectie en synchronisatie algoritme. Simulatie resultaten laten zien dat het toepassen van deze algoritmes verantwoord is.

Simulaties zijn noodzakelijk om de werking van een systeem te controleren, ze kunnen echter ook voor een ander doel gebruikt worden. Het "programma", bijvoorbeeld geschreven in "C", kan samen met eisen t.a.v. tijd- en oppervlaktegebruik en informatie betreffende implementatie alternatieven gebruikt worden als invoer voor de ontwerpstap waarin het systeem in hardware en software componenten wordt verdeeld, de zgn. *hardware/software partitioning*. Om snel alle ontwerpmogelijkheden af te zoeken is het gebruik van automatische gereedschappen onontbeerlijk. We laten zien hoe het programma HSPART kan worden toegepast om de basisbandbewerkingen van WISSCE te verdelen over hardware en software.

Het laatste deel van het ontwerp is het afbeelden van de hardware en software delen op de *transport triggered architecture*. Voor WISSCE bleek dat er een vijftal speciale *functional units* benodigd waren. Na het specificeren van zowel het "standaard" en het toepassingsspecifieke deel van de processor, wordt hun onderlinge samenwerking gecontroleerd d.m.v. co-simulatie. Het simulatie-programma PTOLEMY is voor dit doel uitgebreid. Na bevredigende simulatie-resultaten kan de processor op "echte" hardware worden gerealiseerd.

We concluderen dat de *embedded system* ontwerpmethodologie voldoet voor systemen zoals beschreven in dit proefschrift. De beschikbare middelen zijn voldoende en de *transport triggered architecture* beschikt over voldoende flexibiliteit om alle gewenste specifieke hardware toe te kunnen voegen.

ABOUT THE AUTHOR

Jack Glas was born on July 7, 1969 in Nieuw Vennep (Haarlemmermeer), The Netherlands. From 1981 to 1987 he attended the Rijksscholengemeenschap in Brielle where he received his Atheneum-B diploma. In 1987 he started his study Electrical Engineering in Delft. In 1992 he received his masters degree "cum laude" from this institute. During the last 3 years of his studies he assisted at a microprocessor course for undergraduates.

In the following years he continued his work in the group of Ralph Otten at Delft University of Technology as a PhD student. In the embedded system design project he participated in the research effort towards the realization of embedded CDMA transceiver. His current research interests lay on the crossing of mobile communications and embedded system design. In February 1997 he will join Bell Laboratories in New Jersey, USA.