SYSTEMC-AMS MODEL OF A DYNAMIC LARGE-SCALE SATELLITE-BASED AIS-LIKE NETWORK

Mu Zhou and René van Leuken

Delft University of Technology, Dept. of Elec. Eng., Mekelweg 4, 2628 CD Delft, The Netherlands.
Email: m.zhou@tudelft.nl; t.g.r.m.vanleuken@tudelft.nl.

ABSTRACT

In an automatic identification system (AIS), ground base stations can only provide services within a very limited range off the coast. Recent analyses show that low earth orbit (LEO) satellites can expand the service of AIS to a global range. Multi-antenna satellite receivers are suggested to be used to cancel the interference between messages in heavily trafficked areas. In this paper, we present a SystemC-AMS model for a dynamic large-scale satellite-based AIS-like network as the test environment for such algorithms. Complex data type is used in the signal flow. Many effects such as Doppler shifts, path loss, arrival angles, and the real-time transmission and collisions of messages are modeled. This model is further tested with a multi-user detection algorithm in the satellite receiver and validated with receivers in matlab.

Index Terms— SystemC-AMS, automatic identification system, multi-antenna satellite receiver, Doppler shift.

1 INTRODUCTION

The automatic identification system (AIS) [1] was developed to facilitate the efficient exchange of navigational data between ships and between ships and shore stations. The current AIS system uses time division multiple access (TDMA) techniques in the VHF maritime mobile band. Recently, a need has evolved for the capability to detect and track ships at distances from coastlines that are larger than can be accomplished by normal terrestrial VHF communications. Requirements inherent in these long-range applications such as better handing of hazardous cargo, improved security and countering illegal operations require detecting ships at very large distances from shores.

Recent analyses [2], [3], [4], [5] show that low earth orbit (LEO) satellites can expand the service of AIS to a global range but the service is only reliable in sparsely trafficked areas. The additional technical challenges LEO satellites faces are collisions of messages in the same time slot from different self-organized time-division multiple access (SOTDMA) cells, collisions of messages between adjacent time slots caused by longer relative propagation channel delay, and relative high variance of path loss and high carrier Doppler.

New receiver algorithms are requested to mitigate collisions of messages in the uplink. To develop new algorithms, it requires a test environment for those algorithms. However, the heavy load for the real-time simulation of the simultaneous transmission between tens of thousands of ships and the needs for flexible reconfiguration and extension make the implementation of this kind system difficult under matlab. With the help of SystemC-AMS [6], we can easily model mixed-signal systems more efficiently by using C++ language only [7, 8, 9, 10, 11, 12, 13, 14]. In this paper, we implement a SystemC-AMS model for a dynamic large-scale satellite-based AIS-like network including transmitters, receivers and the channel. Many effects such as Doppler shifts, path loss, arrival angles, and the real-time transmission and collisions of messages are modeled.

The remainder of this paper is organized as follows. Section 2 introduces the mathematical model. Section 3 presents the structure of the SystemC-AMS model. Section 4 shows the simulation setup and results. Section 5 concludes this paper.

This work was supported in part by the project BDREAMS at Delft University of Technology in Netherlands, and China Scholarship Council from P.R.China.

2 MATHEMATICAL MODEL

In this section, we introduce the mathematical model of the system.

2.1 SCENARIO

Consider a single satellite in the low earth orbit at altitude h_s (see Fig. 1). In Fig. 1, the view is taken on the observed plane defined by the observed ship, the satellite and the nadir. The red dotted line denotes the satellite orbit which is in another plane intersecting the observed one. The satellite coordinate system is set to a right-handed system:

1. X-axis parallel to the satellite velocity vector.
2. Y-axis parallel to the satellite field of view (FoV) and orthogonal to the satellite velocity vector.
3. Z-axis pointed to the nadir.

Ships in the FoV (has radius r_{FoV}) send out message packets to the satellite frequently to report their navigational information. The i-th ship has its own azimuth seen from the satellite ϕ_i and ground range away from the nadir r_i (see Fig. 2), which determines its own parameters related to the effects modeled.

One of the typical effects in the AIS-like system is the different carrier Doppler shifts for ships inside FoV. The ship Doppler shift $\Delta f_i(\phi_i, r_i)$ is defined by

$$\Delta f_i(\phi_i, r_i) = \frac{v_r}{\lambda}, \tag{1}$$

where v_r is the relative velocity between the observed ship and the satellite, and λ is the carrier wavelength given by

$$\lambda = \frac{c}{f}, \tag{2}$$

where c is the speed of light, f is the carrier frequency. Then the ship carrier frequency seen from the satellite is given by

$$f_i(\phi_i, r_i) = f + \Delta f_i(\phi_i, r_i). \tag{3}$$

The ship Doppler phase shift seen at the satellite receiver φ_i is given by

$$\varphi_i(\phi_i, r_i) = e^{j2\pi\Delta f_i(\phi_i, r_i)T_s}, \tag{4}$$

where $T_s = 1/f_s$ is the sampling period for the satellite receiver and f_s is the corresponding sampling frequency. For a satellite at

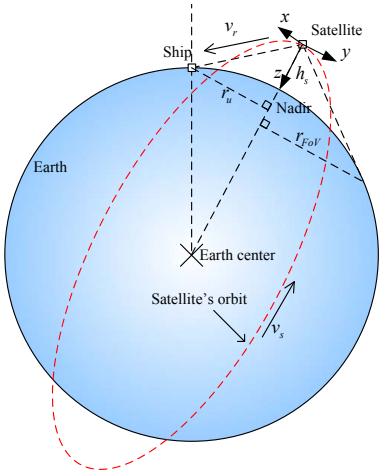

Fig. 1. The satellite orbit and the geometric relation between the satellite and the observed ship.

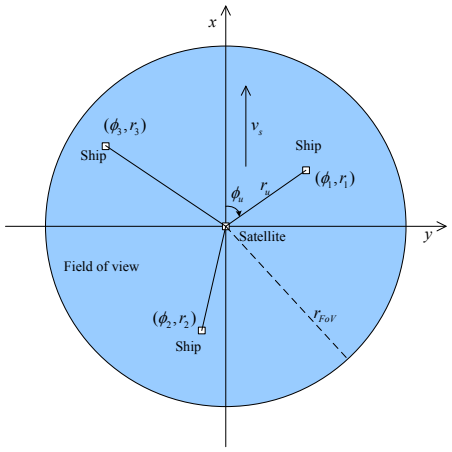

Fig. 2. The top view of the satellite field of view.

altitude $h_s = 600\text{km}$ and carrier frequency $f = 162.025\text{MHz}$, the maximum ship Doppler shift seen by the satellite is $\pm 3.7\text{kHz}$.

A perpendicular half-wave dipole antenna is used as the ship transmit antenna. The received power is given by

$$P_i(r_i) = P_t(r_i) + L(r_i) + G_a, \tag{5}$$

where $P_t(r_i)$ is the ship transmit power, $L(r_i)$ is the path loss, G_a is the satellite antenna gain. The received power is virtually gained by a value G_{AGC} in the satellite receiver to bring the value close to numerical "1.0". Since the online computation of the received power is not acceptable due to its heavy load, we use a look-up table precomputed from the matlab routine provided by [2] instead, where the index is r_i and the resolution of the index is 1 nautical mile.

Assume the satellite is equipped with a uniform linear array positioned parallel to FoV. The arrival angle of the incident wave from ships on the satellite antenna array is $\theta_i(\phi_i, r_i)$. We ignore the detailed formula for $\theta_i(\phi_i, r_i)$ here.

2.2 Baseband Channel Model

In this paper, a baseband channel model is constructed. The effects of the RF parts of transmitters and receivers are integrated into the baseband channel model because the large amount of samples of the carrier makes the simulation load unacceptable. We model the multiple-input and multiple-output (MIMO) channel at a high level where interference, path loss, arrival angles and Doppler effects are added to the transmitted signals. The channel delay is not explicitly modeled and absorbed into the starting time of ship transmission. QPSK modulation instead of GMSK modulation currently used in AIS system is selected. In the ship transmitter, source signals are mapped into a alphabet $\frac{\sqrt{2}}{2}\{1+j, -1+j, -1-j, 1-j\}$. Assume p ground ships and m satellite antennas with spacing Δ wavelengths. For finite n observations, the baseband channel model is defined as

$$\mathbf{X} = \mathbf{AB}\left(\mathbf{F} \odot \mathbf{S}\right) + \mathbf{N}, \tag{6}$$

where $\odot$ is the Schur-Hadamard (pointwise multiplication) operator. $\mathbf{X}$ is the received data matrix. $\mathbf{A}$ is the steering matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \varphi_a^1(\theta_1) & \varphi_a^1(\theta_2) & \cdots & \varphi_a^1(\theta_p) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_a^{m-1}(\theta_1) & \varphi_a^{m-1}(\theta_2) & \cdots & \varphi_a^{m-1}(\theta_p) \end{bmatrix}, \tag{7}$$

where $\varphi_a^k(\theta_i) = e^{j2\pi k\Delta \sin\theta_i}$ and $\theta_i = \theta_i(\phi_i, r_i)$. $\mathbf{B}$ is the scaling matrix given by

$$\mathbf{B} = diag\left\{b_1, b_2, \cdots, b_p\right\}, \tag{8}$$

where $b_i = \sqrt{10^{(P_i+G_{AGC})/10}}$ and $P_i = P_i(r_i)$. $\mathbf{F}$ is the Doppler phase shift matrix given by

$$\mathbf{F} = \begin{bmatrix} 1 & \varphi_1^1 & \cdots & \varphi_1^{n-1} \\ 1 & \varphi_2^1 & \cdots & \varphi_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \varphi_p^1 & \cdots & \varphi_p^{n-1} \end{bmatrix}, \tag{9}$$

where $\varphi_i = \varphi_i(\phi_i, r_i)$. $\mathbf{S}$ is the source matrix given by

$$\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ s_{21} & s_{22} & \cdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{p1} & s_{p2} & \cdots & s_{pn} \end{bmatrix}, \tag{10}$$

where s_{ij} denotes the j-th transmitted symbol of the i-th ship, and $\mathbf{N}$ is a $m \times n$ noise matrix constructed by i.i.d. Gaussian random variables.

3 C++ Model

Based on the mathematical basis in section II, a C++ system model is established in this section. The system is divided into four main parts (see Fig. 3), a virtual transmitter, a satellite receiver, a channel and a system controller. In Fig. 3, the solid flow line denotes the signal flow and the dash flow line denotes the control flow.

3.1 Configuration Data

A general C++ class *syspara* is defined to help manage the entire system (See Listing 1). It stores parameters and the status of the system, and provides functions to update it. The instance of this class is declared globally and initialized in the beginning of the main function of SystemC. The lookup table is put into a file and read into the system when *syspara* is initialized.

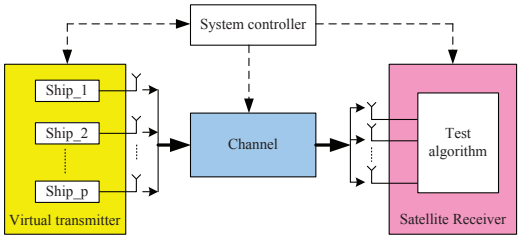

Fig. 3. The schematic diagram of the C++ system model.

Listing 1. The *syspara* class.

```
class syspara
{
public:
 // System size
 int m_num_user, m_num_trans, m_num_rec;
 double m_oversampling, m_antenspacing;
 // System time
 double m_global_time, m_global_time_l;
 // Ship config. data
 double *p_Cafreq_u, *p_lamda_u;
 double *p_Pathloss_u, *p_ar_angle_u;
 ...
 // Satellite config. data
 double m_hight_sat, m_velo_sat;
 double m_T_sat, m_earth_r;
 ...
 // Path loss
 double *p_Pr;
 ...
 syspara();
 ~syspara();
 void initpara();
 double doppler_sat(double r_u, double phi_u);
 void initPathloss();
 void genuserdata();
 void gen_randpos_u(int o);
 void update_pos_u(double delta_T, int o);
 void update_spatialpara_u(double delta_T, int o);
 void checktrans_u(int o);
 void shutdowntrans_u(int o);
};
// Global declaration
extern syspara *g_syspara;
```

3.2 Virtual Transmitter

The virtual transmitter (See Listing 3) is modeled as a SystemC module which consists of p single ship transmitters (See Listing 2). Fig. 4 shows the structure of a single ship transmitter. Inside a ship transmitter, a message encoder produces binary signals, which is then separated into two parallel I/Q paths (See S2P in Fig. 4) fed into a QPSK modulator. The output of the QPSK modulator is modulated signals of a predefined complex data type, *st_complex* (See Listing 4). Because SystemC-AMS does not support tracing a self-defined data type, we implement the output stream function "<<" and use a module *trace* to output the data into a file, which is later read out and analyzed.

The rate of the output baseband complex signal is 4.8k symbol per second (sps), which is half of the source rate, 9.6k bit per second (bps). The frequency values of ports denote the sampling frequency. The transmission control module is used to control the start and the end of the ship transmission by exchanging information with the system controller module.

Listing 2. The *user_b* class.

```
SC_MODULE(user_b)
{
 sc_in<bool> i_clk;
 sc_in<bool> i_reset;
 sc_out<st_complex> o_QPSK_trans;
 sc_signal<double> s_sourcesig;
 sc_signal<int> s_status;
 sc_signal<bool> s_reset;
 sc_signal<int> s_cnt, s_cnt_in;
 ...
 void comb_proc_user_b()
 {
  if (i_reset.read()==1 || g_syspara->p_status_u[o]==0)
  {...}
  else
  { ...
   if (s_cnt.read() < g_syspara->p_packlen_u[o])
   { ...
    s_sourcesig.write(packet[(int)s_cnt.read()]);
   }else{...}
  }
 }
 void seq_proc_user_b();
 SC_HAS_PROCESS(user_b);
 s2p_QPSK_b *p_QPSK;
 explicit user_b(::sc_core::sc_module_name, int o_in=0)
  :k(sqrt(2.0)/2.0), o(o_in)
 {
  p_QPSK = new s2p_QPSK_b("QPSK_mod", k);
  p_QPSK->i_clk(i_clk);
  p_QPSK->i_reset(s_reset);
  p_QPSK->i_sig(s_sourcesig);
  p_QPSK->o_QPSK(o_QPSK_trans);
  SC_METHOD(comb_proc_user_b);
  sensitive<<i_reset<<s_cnt<<i_clk;
  SC_METHOD(seq_proc_user_b);
  sensitive_pos<<i_clk;
 }
 ~user_b();
};
```

Listing 3. The *multi_users_trans_b* class.

```
template<int NUMUSERS>
SC_MODULE(multi_users_trans_b)
{
 sc_in<bool> i_clk;
 sc_in<bool> i_reset;
 sc_out<st_complex> o_trans[NUMUSERS];
 // Single ship transmitter
 user_b **p_user_trans;
 SC_CTOR(multi_users_trans_b)
 {
  int i;
  char stbuf[20];
  p_user_trans = new user_b*[NUMUSERS];
  for (i = 0; i < NUMUSERS; i++)
  {
   sprintf(stbuf, "user_trans(%d)", i);
   p_user_trans[i] = new user_b(stbuf, i);
   p_user_trans[i]->i_clk(i_clk);
   p_user_trans[i]->i_reset(i_reset);
   p_user_trans[i]->o_QPSK_trans(o_trans[i]);
  }
 }
 ~multi_users_trans_b();
};
```

Listing 4. The *st_complex* class.

```
class st_complex
```

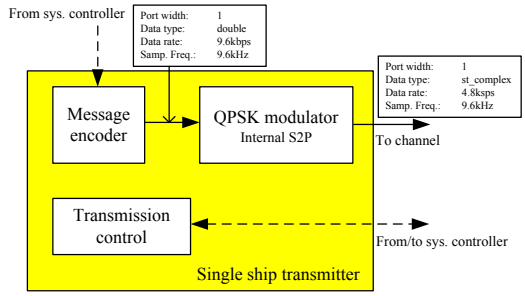

Fig. 4. The schematic diagram of a single ship transmitter module.

```
{
public:
 double real, imag;
 st_complex();
 st_complex(double real_in, double imag_in);
 st_complex(const st_complex &r);
 inline st_complex &operator=(const st_complex &r);
 inline bool operator==(const st_complex &r) const;
 inline friend st_complex operator+(const st_complex &
      r1, const st_complex &r2);
 inline friend st_complex operator-(const st_complex &
      r1, const st_complex &r2);
 inline friend st_complex operator*(const st_complex &
      r1, const st_complex &r2);
 inline friend st_complex operator*(const st_complex &r
      , const double s);
 inline friend st_complex operator/(const st_complex &r
      , const double s);
 inline friend st_complex operator/(const st_complex &
      r1, const st_complex &r2);
 inline double norm_s() const;
 inline double norm() const;
 inline st_complex conj() const;
 // Output stream
 inline friend ostream& operator << (ostream& os,
      st_complex const &r)
 {
  os << "(" << r.real <<",_"<< r.imag << ")";
  return os;
 }
 // Trace function for SystemC
 inline friend void sc_trace(sc_trace_file *tf, const
      st_complex &r, const std::string & NAME)
 {
  sc_trace(tf,r.real, NAME + ".real");
  sc_trace(tf,r.imag, NAME + ".imag");
 }
 // This function is not available in SystemC-AMS 1.0
      BETA1
 inline friend void sca_trace(sca_util::sca_trace_file
      *tf, const st_complex &r, const std::string &
      NAME)
 {
  sca_util::sca_trace(tf, r.real, NAME + ".real");
  sca_util::sca_trace(tf, r.imag, NAME + ".imag");
 }
};
```

3.3 Channel

The channel is modeled as a SystemC-AMS model. Fig. 5 shows the processing flow inside the channel module, where the p paths complex data from the virtual transmitter is multiplied by scaling factors, rotated by Doppler phase shifts and array phase shifts, and added together to form m ports output separately to the m antennas in the satellite receiver. These processing are the implementation of (6) (See Listing 5). The necessary information for these processing comes from the system controller module. Note that the oversampling ratio P is used in ports.

Listing 5. The *channel_b* class.

```
template<int NUMUSERS, int NUMTRECS>
SCA_TDF_MODULE(channel_b)
{
 sca_tdf::sc_in<st_complex> i_sig[NUMUSERS];
 sca_tdf::sc_out<st_complex> o_sig[NUMTRECS];
 void set_attributes()
 {
  double T = 1.0/(OVERSAMPLINGRATIO*1.0*USER_RATE);
  int i;
  for (i = 0; i < NUMUSERS; i++)
  {
   i_sig[i].set_timestep(T,sc_core::SC_SEC);//1.0beta2
  }
 }
 void processing()
 {
  int i,j;
  st_complex signalout[NUMTRECS];
  st_complex array_phase, doppler_phase;
  double phi;
  // Process data from every ship
  for (i = 0; i < NUMUSERS; i++)
  {
   if (g_syspara->p_status_u[i] == 1)
   {
    // Doppler phase shift
    phi = 2 * M_PI * (g_syspara->p_Cafreq_u[i] -
         g_syspara->m_centralfreq) * sc_time_stamp().
         to_seconds();
    doppler_phase = st_complex(cos(phi), sin(phi));
    // Add the output to antennas
    for (j = 0; j < NUMTRECS; j++)
    {
     phi = 2 * M_PI * g_syspara->m_antenspacing * j *
          sin(g_syspara->p_ar_angle_u[i]) * g_syspara->
          m_lamda / g_syspara->p_lamda_u[i];
     // Array phase shift
     array_phase = st_complex(cos(phi), sin(phi));
     signalout[j] = signalout[j] + i_sig[i].read() *
          g_syspara->p_Pathloss_u[i] * array_phase *
          doppler_phase;
    }
   }
  }
  // Add noise
  double noise_mag = 0.707 * sqrt(pow(10.0, (g_syspara
       ->m_noise_p + AGCGAIN)/10.0));
  st_complex Antenna_noise[NUMTRECS];
  for (i = 0; i < NUMTRECS; i++)
  {
   Antenna_noise[i] = st_complex(gaussian(0.0,
        noise_mag), gaussian(0.0, noise_mag));
   o_sig[i].write(signalout[i] + Antenna_noise[i]);
  }
 }
 SCA_CTOR(channel_b){}
};
```

3.4 Satellite Receiver

The satellite receiver is modeled as a SystemC module (See Listing 6). Fig. 6 shows the structure of the satellite receiver, where the m paths complex data are fed into the detection algorithm (See Listing 7) to detect the number of signals while the m paths complex

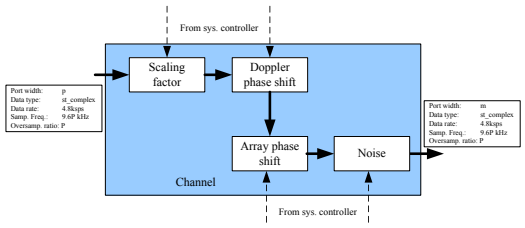

Fig. 5. The schematic diagram of the channel module.

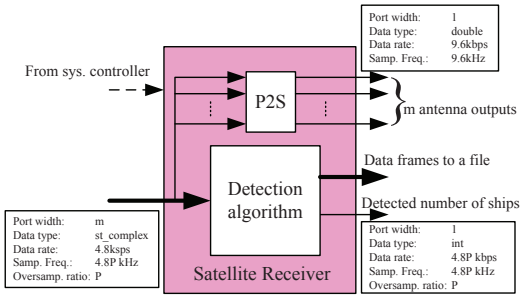

Fig. 6. The schematic diagram of the satellite receiver module.

data are also fed into the parallel to serial (P2S) module to form m paths real antenna output. In the P2S module, we simply separate the real and imaginary part of the complex data to a single data stream where the rate of data is doubled. The oversampling ratio P is used in ports.

Listing 6. The *multi_antennas_fe_b* class.

```
template<int NUMTRECS>
SC_MODULE(multi_antennas_fe_b)
{
 sc_in<bool> i_clk;
 sc_in<bool> i_reset;
 sc_in<st_complex> i_sig[NUMTRECS];
 sc_out<double> o_sig[NUMTRECS];
 // Parallel to serial
 p2s_QPSK_b **p_rec;
 SC_CTOR(multi_antennas_fe_b)
 {
  int i;
  char stbuf[20];
  p_rec = new p2s_QPSK_b*[NUMTRECS];
  for (i = 0; i < NUMTRECS; i++)
  {
   sprintf(stbuf, "p2s_QPSK_b(%d)", i);
   p_rec[i] = new p2s_QPSK_b(stbuf);
   p_rec[i]->i_clk(i_clk);
   p_rec[i]->i_reset(i_reset);
   p_rec[i]->i_sig(i_sig[i]);
   p_rec[i]->o_sig(o_sig[i]);
  }
 }
 ~multi_antennas_fe_b();
};
```

Listing 7. The *detection* class.

```
template<int NUMTRECS>
SC_MODULE(detection)
{
 sc_in<bool> i_clk;
 sc_in<bool> i_reset;
 sc_in<st_complex> i_sig[NUMTRECS];
 sc_out<int> o_d;
 // Internal counter
 sc_signal<int> s_cnt, s_cnt_in;
 // Data matrix for the detection algorithm
 queue<double *> *winquere, *winqueim;
 double **Qre, **Qim, **Rre, **Rim;
 ...
 void comb_proc_detection(){// Control counters}
 void seq_proc_detection()
 {// Ignore the processing
  o_d.write(d);
 }
 SC_CTOR(detection)
 { // Ignore initialization data
  SC_METHOD(comb_proc_detection);
  sensitive<<i_reset<<s_cnt;
  SC_METHOD(seq_proc_detection);
  sensitive_pos<<i_clk;
 }
 ~detection();
};
```

3.5 System Controller

The system controller is the key part in the model to make the system dynamic (see Fig. 7). The system controller is modeled as a SystemC module consisting of two internal processes regularly refreshing the system status (See Listing 8), where one process refreshes the ship position (ϕ_i, r_i) and related data while the satellite is flying over them, and the other process checks the ship transmission time T_u and triggers the real-time transmission of ships.

Listing 8. The *refresh_sys* class.

```
SC_MODULE(refresh_sys)
{
 sc_in<bool> i_clk1;
 sc_in<bool> i_clk2;
 sc_out<int> o_num_sigs;
 // The first refreshing process
 void seq_proc_refresh_sys1()
 {
  if (i_clk1.read() == 1)
  {
   int i;
   g_syspara->m_global_time = sc_time_stamp().
        to_seconds();
   double delta_time = g_syspara->m_global_time -
        g_syspara->m_global_time_l;
   for (i = 0; i < g_syspara->m_num_user; i++)
   { // Update global data, ship position
    g_syspara->update_spatialpara_u(delta_time, i);
    g_syspara->checktrans_u(i);
   }
   g_syspara->m_global_time_l = g_syspara->
        m_global_time;
  }
 }
 // The second refreshing process
 void seq_proc_refresh_sys2()
 {
  int count = 0;
  if (i_clk2.read() == 1)
  {
   int i;
   g_syspara->m_global_time = sc_time_stamp().
        to_seconds();
   for (i = 0; i < g_syspara->m_num_user; i++)
   { // Check ship transmission time
    g_syspara->checktrans_u(i);
    // Count the number of active ships
```

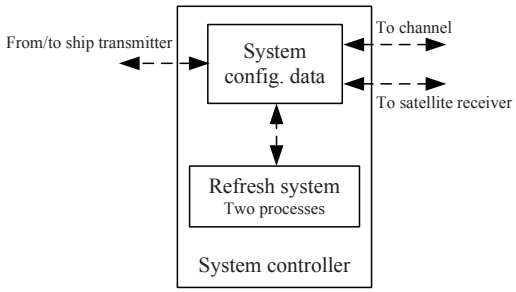

Fig. 7. The schematic diagram of the system controller module.

```
34        if (g_syspara->p_status_u[i] == 1)
35        {
36          count++;
37        }
38      }
39    }
40    o_num_sigs.write(count);
41  }
42  SC_CTOR(refresh_sys)
43  {
44    SC_METHOD(seq_proc_refresh_sys1);
45    sensitive_pos<<i_clk1;
46    SC_METHOD(seq_proc_refresh_sys2);
47    sensitive_pos<<i_clk2;
48  }
49 };
```

In the initialization phase of the system, ships are randomly positioned in FoV and other parameters of the system are generated based on the ships' positions. All the ships in FoV are moving backward opposite to the velocity vector of the satellite. When the ground range between one ship and the satellite exceeds the radius of FoV, the position of this ship will be replaced by a randomly generated new one in FoV. To simply produce the collision of messages, the transmission time of ships are randomly generated within a given time interval. The transmission time is assumed to be integer offsets on the time line which is the multiple of the clock time period of the transmitters. Higher resolution on the transmission time is achievable by increasing the clock of transmitters as well as the updating rate of the other module in the system.

Fig. 8 shows the updating time period (or clock time period) of each module in the system. The updating time period of the first refreshing process is $10s$. The updating time period of the second refreshing process as well as the clock time period for the transmitters is $1/9600s$. The clock time period of the receiver is also $1/(4800P)s$, where P is the oversampling ratio. The channel is sampled at the highest rate with an updating time period $1/(9600P)s$.

Fig. 9 shows the flow chart of the ship transmission control which is managed by the interactive communication between the system controller and the ship transmission controller. The system controller regularly determines if the system time T_g exceeds the ship transmission time T_u, i.e., $T_g > T_u$ and when this condition fulfills it generates a new transmission time T'_u for this ship and sets the status of this ship to "transmission", after which the transmission controller inside this ship gets its status and starts transmission. The ship transmitter regularly checks if $T_g > T_u + NT_{sym}$, where N is the fixed message length for every ship and T_{sym} is the symbol time period, and when this condition fulfills it assigns $T_u = T'_u$ and sets the status of this ship to "standby".

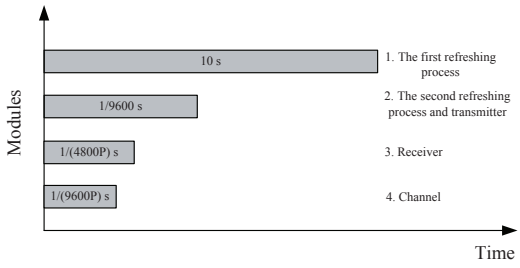

Fig. 8. The updating time period of each module.

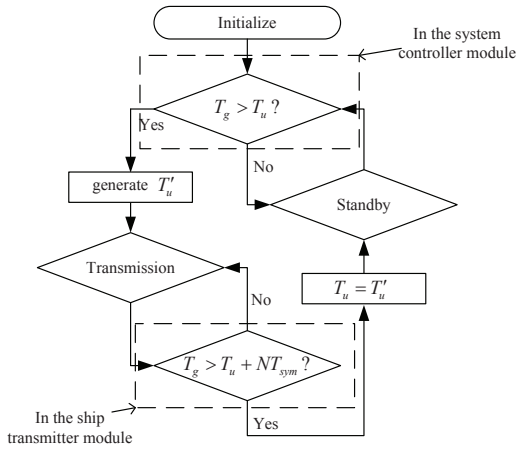

Fig. 9. The flow chart of the ship transmission control.

4 Simulation Results

This C++ system model is configured as listed in Table I. A multi-user detection algorithm is implemented to test the output of this model. The code is compiled and simulated under cygwin 1.7 and windows XP with Intel Duo CPU 3.0 GHz and 4 GB ram using libraries of SystemC 2.2.0 and SystemC-AMS 1.0BETA2.

Simulation time cost is one of the important factors that influences our modeling. We need to check if it is acceptable. The measure is the time cost ratio defined as the ratio between the time cost and the simulated system time period.

$$r_{tcr} = \frac{T_{simu}}{T_{sys}}, \tag{11}$$

where T_{simu} is the simulated time cost in the computer and T_{sys} is the system time period. We shut down all output streams in order to leave the results accurate. Fig. 10 shows the time cost ratio for a real-time simulation of a system time period of 10 seconds with 10 to 10,000 ships. The ratio increases nonlinearly by the number of ships, which was expected to have a linear growth in time. The reason for this nonlinearity is the undetermined loss of efficiency in the growth of stack as we manually increase the size of stack for 10,000 ships.

To verify the modeling of collision of messages, we trace the antenna output (real data in single streams) to a VCD file which is then converted to a WLF file. We input the WLF file into Modelsim to view the collision of messages. Fig. 11 shows the waveform at the time interval with 3 active ships out of 100 ships. In Fig. 11, ships transmit signals with different attenuation at random time points and

Table 1. System configuration.

Carrier frequency	162.025MHz
Channel bandwidth	25kHz
Modulation	9.6kbps QPSK
Oversampling ratio P	8
Channel sampling frequency	$9.6P$ kHz
Satellite altitude	600km
Number of ships	10 to 10,000
Ship transmission interval	Random time in every 6s after the last transmission
Ship position refresh interval	10s
Message length	256bits (128 QPSK symbols)
Ship transmitter	
Transmitter power	12.5W
Antenna type	Half-wave dipole
Antenna gain	Look-up table
Channel	
Free space path loss at 162.025MHz	Look-up table
Satellite receiver	
Number of antennas	4
Antenna spacing	0.5λ
Antenna gain	0dB
G_{AGC}	97dB
Uplink noise power	-121.30dBm

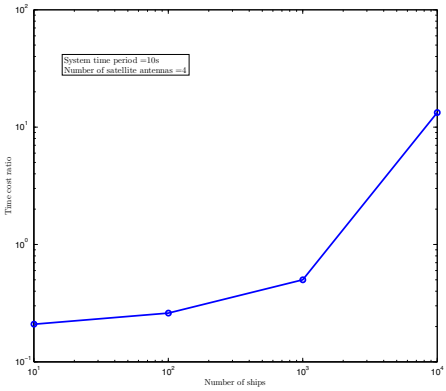

Fig. 10. The time cost ratio vs. the number of ships.

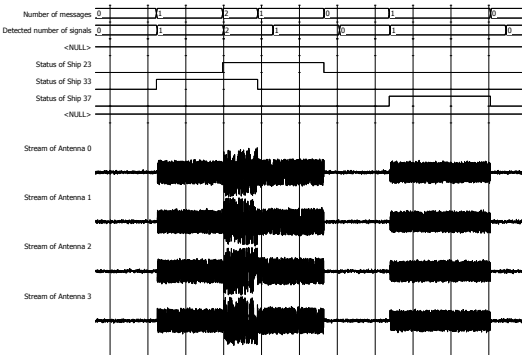

Fig. 11. The output waveform of the satellite receiver at the time interval with 3 active ships out of 100 ships.

the output of every antenna on the satellite is the addition of these signals after the channel. During the simulation, the detection algorithm [15] gives a correct estimation of the number of transmitted signals at the same time.

To test the modeling of other effects, we output the frames collected at the satellite receiver to files and use receivers [16] implemented in matlab to process these frames, estimate the parameters of the ships and decode the messages (See Fig. 12). Frames have a fixed length and are continuously overlapped along the time line. The receivers have been tested together with transmitters and a noisy channel in matlab, where the routine mainly deals with randomly generated frame data. Table 2 shows the comparison between the ship configuration data and the corresponding received and estimated data. It is evident that the output frames can be successfully decoded by the receivers in matlab, which validates our C++ model.

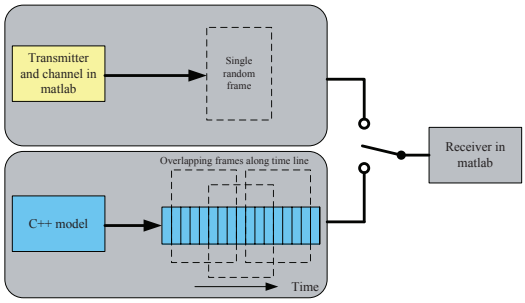

Fig. 12. The setup of the verification scheme.

Table 2. Comparison between the configured and the received data.

ID	X (nm)	Y (nm)	Δf(Hz)	DOA(degree)	b
Ship Configuration Data					
0	-952	-413	-3397.94	-40.4248	0.706584
1	-738	803	-2509.47	58.0452	0.677247
2	-1019	744	-3012.63	52.86397	0.596408
3	1219	592	3359.174	44.52525	0.558618
4	664	-77	3507.657	-11.1587	0.966476
5	-507	-758	-2040.49	-59.4616	0.782711
6	-1258	-449	-3517.03	-37.0874	0.566311
7	-475	129	-3183.06	19.83745	1.114353
8	-626	-592	-2652.24	-53.775	0.816643
9	-141	-1253	-414.932	-65.7895	0.597258
Received and Estimated Data					
0	-952	-413	-3401.04	-40.3895	0.704346
1	-738	803	-2515.99	57.8846	0.675193
2	-1019	744	-3015.61	52.87989	0.595452
3	1219	592	3359.287	44.52076	0.558163
4	664	-77	3500.967	-11.261	0.96334
5	-507	-758	-2044.91	-59.3996	0.782012
6	-1258	-449	-3516.2	-37.0776	0.565496
7	-475	129	-3186.91	19.89696	1.111181
8	-626	-592	-2658.74	-53.7418	0.813988
9	-141	-1253	-417.546	-65.8802	0.596844

Subsequently, we use our validated C++ model to further compare the performance of different receivers. We did a long time simulation for a system time period of 10 minutes and 100 ships in FoV. Fig. 13 and Fig. 14 show the performance of the multi-user receiver and single user receiver, respectively, where the big blue circle at the outer edge denotes FoV, the red points denote the reported ship positions relative to the satellite in FoV, and the blue small circles denote the mark of the successfully decoded position data (Only the messages that pass the CRC check are shown). The numbers beside the circles and points denote the ID number of ships. The relative movement of ships draws the dotted lines. The visible ID number can be larger than 100 because ships are moving out of FoV while new ones are generated to keep the number of ships in FoV constant. It is obvious that the more red points are circled, the better the performance will be. Fig. 15 shows the comparison of the number of successfully decoded messages of the two receivers. It is sufficient to conclude that the multi-user receiver greatly improves the performance of ship

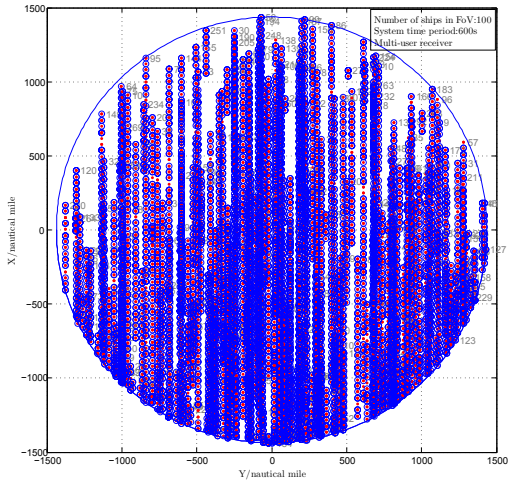

Fig. 13. The ship positions detected by the multi-user receiver.

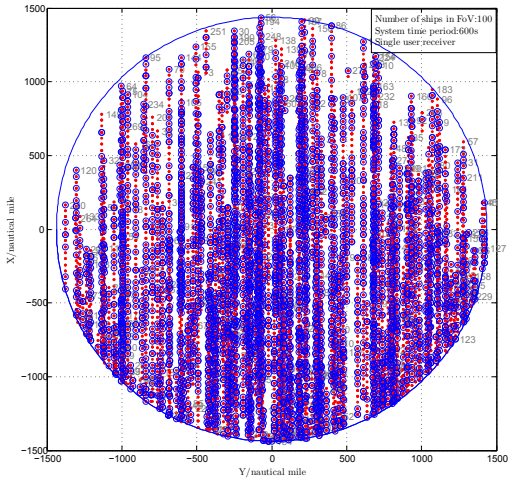

Fig. 14. The ship positions detected by the single-user receiver.

detection against the single user receiver.

5 Conclusion

In this paper, we presented and validated a SystemC-AMS model for a dynamic large-scale network. This model combined SystemC-AMS, SystemC and the general C++ classes and showed the attractive ability of SystemC-AMS to handle such large-scale mixed signal network. This model will be extended to a multi-satellite system.

6 References

[1] *RECOMMENDATION ITU-R M.1371-2: Technical characteristics for a universal shipborne automatic identification system using time division multiple access in the VHF maritime mobile band*, ITU Std., 2006.

[2] O. F. H. Dahl, "Space-based AIS receiver for maritime traffic monitoring using interference cancellation," Master's thesis, Norwegian University of Science and Technology, Department of Electronics and Telecommunications, 2006.

[3] *REPORT ITU-R M.2084: Satellite detection of automatic identification system messages*, ITU Std., 2006.

[4] M. A. Cervera and A. Ginesi, "On the performance analysis of a satellite-based AIS system," in *Signal Process. for Space Commun., 2008. SPSC 2008. 10th Int. Workshop on*, Oct. 2008, pp. 1–8.

[5] M. A. Cervera, A. Ginesi, and K. Eckstein, "Satellite-based vessel automatic identification system: A feasibility and performance analysis," *Int. J. Satell. Commun. Network.*, 2009.

[6] *SystemC AMS extensions Users Guide*, OSCI Std., 2010.

[7] K. Caluwaerts and D. Galayko, "SystemC-AMS modeling of an electromechanical harvester of vibration energy," in *Specification, Verification and Design Languages, 2008. FDL 2008. Forum on*, Sep. 2008, pp. 99–104.

[8] E. Markert, M. Dienel, G. Herrmann, and U. Heinkel, "SystemC-AMS assisted design of an inertial navigation system," *Sensors Journal, IEEE*, vol. 7, no. 5, pp. 770–777, May 2007.

[9] M. Zhou, R. van Leuken, and H. J. L. Arriëns, "Modeling a configurable resistive touch screen system using SystemC and SystemC-AMS," in *20th annual workshop on circuits, systems and signal processing-ProRISC*, Nov. 2009, pp. 393–398.

[10] E. Markert, M. Dienel, G. Herrmann, D. Mueller, and U. Heinkel, "Modeling of a new 2D acceleration sensor array using SystemC-AMS," *Journal of Physics: Conference Series*, vol. 34, no. 1, p. 253, 2006.

[11] P. Hartmann, P. Reinkemeier, A. Rettberg, and W. Nebel, "Modelling control systems in SystemC AMS-benefits and limitations," in *SOC Conference, 2009. SOCC 2009. IEEE International*, Sep. 2009, pp. 263–266.

[12] M. Vasilevski, N. Beilleau, H. Aboushady, and F. Pecheux, "Efficient and refined modeling of wireless sensor network nodes using SystemC-AMS," in *Research in Microelectronics and Electronics, 2008. PRIME 2008. Ph.D.*, Apr. 2008, pp. 81–84.

[13] M. Vasilevski, F. Pecheux, N. Beilleau, H. Aboushady, and K. Einwich, "Modeling refining heterogeneous systems with systemc-ams: application to wsn," in *Proceedings of the conference on Design, automation and test in Europe*, ser. DATE '08, 2008, pp. 134–139.

[14] A. Vachoux, C. Grimm, and K. Einwich, "Analog and mixed signal modelling with SystemC-AMS," in *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, vol. 3, May 2003, pp. III–914–III–917.

[15] M. Zhou and A.-J. van der Veen, "Stable subspace tracking algorithm based on signed URV decomposition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Prague, Czech Republic, May 2011, pp. 2720–2723.

[16] ——, "Improved subspace intersection based on signed URV decomposition," in *Proc. of the Asilomar Conf. on Signals, Syst., and Comput.*, Pacific Grove, California, USA, Nov. 2011.

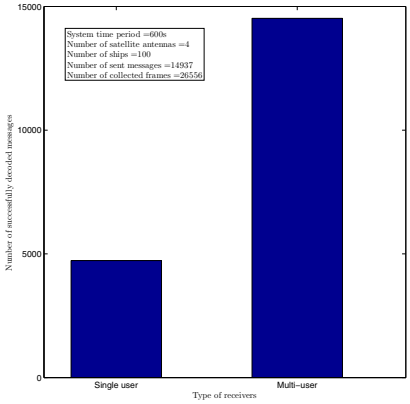

Fig. 15. Comparison of the number of successfully decoded messages.