Modeling A Configurable Resistive Touch Screen System Using SystemC and SystemC-AMS

Mu Zhou, René van Leuken, and Huib J. Lincklaen Arriëns

***Abstract*—A configurable resistive 4-wire touch screen system is modeled in this paper using SystemC and SystemC-AMS. The touch screen system is divided into four parts, including a human touch imitator, a 4-wire screen circuit network, successive approximation analog-to-digital converters (ADC), and a system controller. The linear electrical network and the synchronous data flow from SystemC-AMS are used for describing circuit networks and ADCs in the system. The other parts are described in SystemC. The connection, synchronization, and control between SystemC and SystemC-AMS modules are demonstrated. The methods for making the model configurable are introduced. A testbench imitating a human touch is input to the system. Simulation results of the system are shown. It proves the functionality of the system.**

***Index Terms*—Resistive touch screen, SystemC-AMS, analog mixed-signal, human touch, ADC.**

I. Introduction

SystemC [1], [2] is a standardized modeling language based on C++ that is intended to enable system level design and IP exchange. It facilitates the engineering programming in a unit, very fast executable specifying, and flexible language to find the solution for representing functionality, communication, software and hardware at various system levels of abstraction.

However, SystemC lacks a standard support for modeling and simulation of analog (continuous-time (CT)) and mixed-signal (mixed continuous-time/discrete-event (DE)) systems. To bridge the gap, SystemC-AMS [3], an analog and mixed-signal (AMS) extension to SystemC, is developed to provide an efficient means for modeling and simulation of heterogeneous systems. The continuous-time systems can be seamlessly integrated with discrete-event models of computation (MoC) in SystemC and SystemC-AMS.

Modeling and simulation of a resistive 4-wire touch screen system [4], as an example on how to model mixed-signal systems using SystemC and SystemC-AMS, are presented in this paper. The resistive 4-wire touch screen system belongs to the most popular and most common touch screen technologies. This type of screen system is recommended for applications in homes, offices, hospitals, and etc. The screen consists of a glass or acrylic panel that is coated with electrically conductive and resistive layers separated by invisible separator dots. A electrical connection between the two layers is established when pressure, usually a human or stylus touch, is applied to the screen causing a change in the electrical current flowing through the screen and a touch event to occur. Then the voltage of some parts of the resistive screen are measured for the calculation of the touch position on the screen surface.

Manuscript received July 5, 2009; revised October 12, 2009. This work was supported in part by the project BDREAMS at Delft University of Technology in Netherlands.

Mu Zhou, René van Leuken and Huib J. Lincklaen Arriëns are with Circuits and Systems Group in the Department of Electrical Engineering, Mathematics and Computer Sciences, Delft University of Technology, Delft, Netherlands (e-mail: M.Zhou@tudelft.nl, t.g.r.m.vanleuken@tudelft.nl, H.J.LincklaenArriens@tudelft.nl).

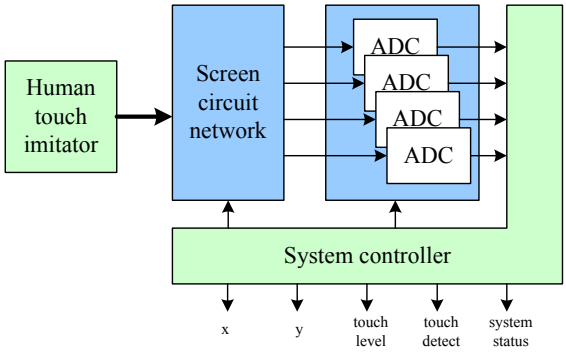

Fig. 1. The schematic diagram of the 4-wire resistive touch screen system.

The system is divided into four parts in this paper, including a human touch imitator (HTI), a 4-wire screen circuit network (SCN), successive approximation analog-to-digital converters (ADC), and a system controller (see Fig. 1). The resistive screen is viewed as an analog circuit network, which is suitable to be modeled in linear electrical network (LEN) from SystemC-AMS. A touch event is viewed as a change to some elements in SCN, which is modeled as an digital-to-analog input to a LEN (from a DE module to a CT module). The detection and measurement for a touch event, are formed by checking the pin status, converting the voltage of four parts of the two layers divided by the electrical connection caused by the touch into digital values using ADCs, and based on these values calculating the resistance of the four parts then further into screen coordinates by using a system controller. The ADC used in this paper is also a mixed-signal subsystem, containing synchronous data flows (SDF) and DE MoCs. The system controller is purely a DE module and modeled in SystemC.

The remainder of this paper is organized as follows. In Section II, the modeling of every part of the screen system is discussed. In Section III, the simulation of the screen system is shown. Section IV concludes the paper.

II. Resistive 4-wire touch screen system

A. Screen circuit network

Fig. 2 shows the structure of SCN modeled in SystemC-AMS. The network is modeled as a four-port resistor network, with output ports located at edges in direction Y- and Y+ on the top layer, X+ and X- on the bottom layer, marked as

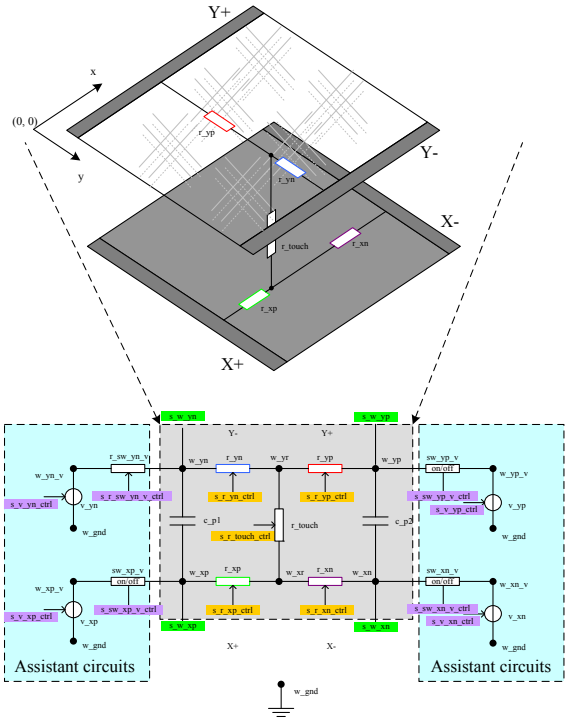

Fig. 2. The screen circuit network.

s_w_yn, *s_w_yp*, *s_w_xp*, and *s_w_xn*, separately. The four ports are all connected to ADCs. Each layer is divided by a touch point into two controlled resistors (totally four resistors, *r_yn*, *r_yp*, *r_xp*, and *r_xn*) with a controlled resistor *r_touch* as a touch resistor connecting the two layers at the touch point. Control signals for these resistors, named *s_r_yn_ctrl*, *s_r_yp_ctrl*, *s_r_xp_ctrl*, *s_r_xn_ctrl*, and *s_r_touch_ctrl*, are channels (ports) connected to the human touch imitator. Parasitic capacities between the two layers are modeled into two parallel capacitors, *c_p1* and *c_p2*, each with a capacitance of 40nF.

In order to measure the voltage at the ports of the network to get information about the touch position, assistant circuits are connected to the four ports so that the network can be put into a certain status for each measure by setting these assistant circuits. Each assistant circuit consists of a controlled voltage source, and a controlled switch or a controlled resistor. The control signals for assistant circuits are channels connected from the system controller. The whole circuit network takes *w_gnd* as the reference node.

Listing 1 shows how the SCN model in Fig. 2 can be expressed in connections of LEN elements provided by SystemC-AMS.

Listing 1. The source code for screen circuit network.

```
#pragma once
#include "systemc-ams.h"

SC_MODULE(touchscreen4wires)
{
 sca_scsdf_in<double> i_r_yn_ctrl, i_r_yp_ctrl,
     i_r_xp_ctrl, i_r_xn_ctrl, i_r_touch_ctrl;
 sca_sdf_in<double> i_v_yn_ctrl, i_v_yp_ctrl,
     i_v_xp_ctrl, i_v_xn_ctrl, i_r_sw_yn_v_ctrl;
 sca_sdf_in<bool> i_sw_yp_v_ctrl, i_sw_xp_v_ctrl,
     i_sw_xn_v_ctrl;

 sca_sdf_out<double> o_w_yn, o_w_yp, o_w_xp,
     o_w_xn;

 sca_elec_node w_yn, w_yp, w_xp, w_xn, w_yr, w_xr,
     w_yn_v, w_yp_v, w_xp_v, w_xn_v;
 sca_elec_ref w_gnd;
 sca_sc2r *r_yn, *r_yp, *r_xp, *r_xn, *r_touch;
 sca_sdf2r *r_sw_yn_v;
 sca_sdf_rswitch *sw_yp_v, *sw_xp_v, *sw_xn_v;

 sca_c *c_p1, *c_p2;
 sca_sdf2v *v_yn, *v_yp, *v_xp, *v_xn;

 sca_v2sdf *vconv_w_yn, *vconv_w_yp, *vconv_w_xp,
     *vconv_w_xn;

 ~touchscreen4wires()
 {
  delete r_yn; r_yn = NULL;
  delete r_yp; r_yp = NULL;
  delete r_xp; r_xp = NULL;
  delete r_xn; r_xn = NULL;
  delete r_touch; r_touch = NULL;
  ...
 }

 SC_CTOR(touchscreen4wires)
 {
  r_yn = new sca_sc2r("resistor_y-");
  r_yn->p(w_yn);
  r_yn->n(w_yr);
  r_yn->ctrl(i_r_yn_ctrl);

  r_yp = new sca_sc2r("resistor_y+");
  r_yp->p(w_yr);
  r_yp->n(w_yp);
  r_yp->ctrl(i_r_yp_ctrl);

  r_xp = new sca_sc2r("resistor_x+");
  r_xp->p(w_xp);
  r_xp->n(w_xr);
  r_xp->ctrl(i_r_xp_ctrl);

  r_xn = new sca_sc2r("resistor_x-");
  r_xn->p(w_xr);
  r_xn->n(w_xn);
  r_xn->ctrl(i_r_xn_ctrl);

  r_touch = new sca_sc2r("resistor_touch");
  r_touch->p(w_yr);
  r_touch->n(w_xr);
  r_touch->ctrl(i_r_touch_ctrl);

  c_p1 = new sca_c("parasitic_capacitor_1");
  c_p1->p(w_yn);
  c_p1->n(w_xp);
  c_p1->value = 40e-9;

  c_p2 = new sca_c("parasitic_capacitor_2");
  c_p2->p(w_yp);
  c_p2->n(w_xn);
  c_p2->value = 40e-9;

  r_sw_yn_v = new sca_sdf2r("
      resistor_switch_r_sw_yn_v");
```

```
71      r_sw_yn_v->p(w_yn_v);
72      r_sw_yn_v->n(w_yn);
73      r_sw_yn_v->ctrl(i_r_sw_yn_v_ctrl);
74
75      sw_yp_v = new sca_sdf_rswitch("
            resistor_switch_sw_yp_v");
76      sw_yp_v->p(w_yp);
77      sw_yp_v->n(w_yp_v);
78      sw_yp_v->ctrl(i_sw_yp_v_ctrl);
79      sw_yp_v->off_val = false;
80      sw_yp_v->ron = 1.0e-6;
81      sw_yp_v->roff = 1.0e12;
82
83      sw_xp_v = new sca_sdf_rswitch("
            resistor_switch_sw_xp_v");
84      sw_xp_v->p(w_xp_v);
85      sw_xp_v->n(w_xp);
86      sw_xp_v->ctrl(i_sw_xp_v_ctrl);
87      sw_xp_v->off_val = false;
88      sw_xp_v->ron = 1.0e-6;
89      sw_xp_v->roff = 1.0e12;
90
91      sw_xn_v = new sca_sdf_rswitch("
            resistor_switch_sw_xn_v");
92      sw_xn_v->p(w_xn);
93      sw_xn_v->n(w_xn_v);
94      sw_xn_v->ctrl(i_sw_xn_v_ctrl);
95      sw_xn_v->off_val = false;
96      sw_xn_v->ron = 1.0e-6;
97      sw_xn_v->roff = 1.0e12;
98
99      v_yn = new sca_sdf2v("voltage_source_v_yn");
100     v_yn->p(w_yn_v);
101     v_yn->n(w_gnd);
102     v_yn->ctrl(i_v_yn_ctrl);
103     v_yn->gain = 1.0;
104
105     v_yp = new sca_sdf2v("voltage_sourve_v_yp");
106     v_yp->p(w_yp_v);
107     v_yp->n(w_gnd);
108     v_yp->ctrl(i_v_yp_ctrl);
109     v_yp->gain = 1.0;
110
111     v_xp = new sca_sdf2v("voltage_source_v_xp");
112     v_xp->p(w_xp_v);
113     v_xp->n(w_gnd);
114     v_xp->ctrl(i_v_xp_ctrl);
115     v_xp->gain = 1.0;
116
117     v_xn = new sca_sdf2v("voltage_source_v_xn");
118     v_xn->p(w_xn_v);
119     v_xn->n(w_gnd);
120     v_xn->ctrl(i_v_xn_ctrl);
121     v_xn->gain = 1.0;
122
123     vconv_w_yn = new sca_v2sdf("
            voltage_converter_vconv_w_yn");
124     vconv_w_yn->p(w_yn);
125     vconv_w_yn->sdf_voltage(o_w_yn);
126     vconv_w_yn->scale = 1.0;
127
128     vconv_w_yp = new sca_v2sdf("
            voltage_converter_vconv_w_yp");
129     vconv_w_yp->p(w_yp);
130     vconv_w_yp->sdf_voltage(o_w_yp);
131     vconv_w_yp->scale = 1.0;
132
133     vconv_w_xp = new sca_v2sdf("
            voltage_converter_vconv_w_xp");
134     vconv_w_xp->p(w_xp);
135     vconv_w_xp->sdf_voltage(o_w_xp);
136     vconv_w_xp->scale = 1.0;
137
138     vconv_w_xn = new sca_v2sdf("
            voltage_converter_vconv_w_xn");
139     vconv_w_xn->p(w_xn);
140     vconv_w_xn->sdf_voltage(o_w_xn);
141     vconv_w_xn->scale = 1.0;
142   }
143 };
```

B. *Human touch imitator*

The human touch imitator controls and sets resistance values of the controlled resistors in SCN through channels connected between SCN and it according to screen coordinates of the touch point given by the testbench. Since the moving speed of the touch point is limited to 50 points/s, the imitator is modeled as a synchronous DE module with a cycle frequency at most 50Hz. The touch resistance is less than 20 ohm, relatively small compared to the resistance of the two layers, each of which are around 200 ohm typically.

C. *Analog-to-digital converter*

A type of successive approximation ADC documented in [7] is chosen to convert voltages at the output ports of SCN into digital values. This ADC uses a delta-sigma digital-to-analog converter (DAC) documented in [8] to generate a reference voltage for the negative input to the voltage comparator [10], while the sampled input analog signal feeds the positive input of the comparator. A voltage divider is put in front of the comparator to convert the the voltage level of the analog input into the voltage range limited by the common mode of the comparator. The ADC conversion operation is defined as doing a dichotomy search to the voltage level of the input analog signal through comparing the input signal to a reference, generated by an internal DAC, equal to the voltage mid-range determined by the last comparison. This DAC is driven by a successive approximation register (SAR), the most significant bit (MSB) of which is initialized to "1", while others to "0". The process is repeated with each bit from MSB to the least significant bit (LSB) in SAR in turn. Every current bit in SAR to be determined is initialized to "1", then remained or reset (to "0") when the voltage level of the input signal is higher or lower than the reference.

Fig. 3 shows that the ADC consists of a CT part (the left dotted box) and a DE part (the right dotted box). The CT part is a SDF cluster with a feedback input from the DE part, an input from the external analog source, and an output to the DE part. The CT and DE parts form a negative feedback loop. At least one sample delay is set to the loop. The comparator has two parameters, a offset voltage and a hysteresis. The low-pass filter (LPF) in the loop is modeled as an analog linear behavior model, which is a first-order butterworth low-pass filter using a transfer function provided by SystemC-AMS. The cut-off frequency of LPF is defined as

$$f_c = (W_{adc} + 1) \times N_s \times T_{fstm} \times f_h, \qquad (1)$$

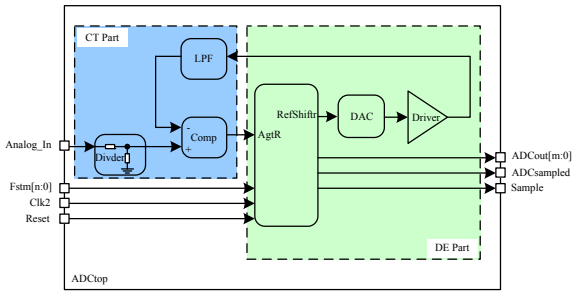

Fig. 3. The schematic diagram of ADC.

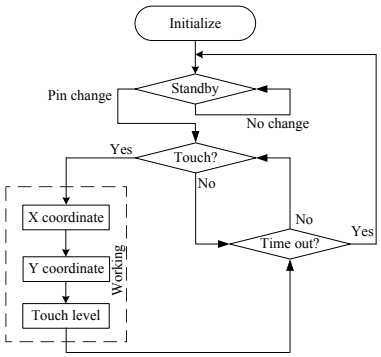

Fig. 4. The system control flow chart.

where W_{adc} is the digitalized output data width of ADC, N_s is the sampling frequency ratio, T_{fstm} is the filter settle time multiplier (FSTM), and f_h is the highest frequency of signals that can be converted by ADC. The clock frequency of internal DAC is at least

$$clk_{dac} = 2^{W_{adc}+1}, \tag{2}$$

where $W_{adc}+1$ means the input width of internal DAC is 1-bit wider than the output width of ADC in order to generate more precise reference voltage. The clock frequency of ADC is at least

$$f_{adc} = f_c \times clk_{dac}. \tag{3}$$

The sample period of the CT part is less than

$$T_{CT} = \frac{1}{2f_c}. \tag{4}$$

D. System controller

The system controller sets the status of the whole system and schedules the computation flow. The system has two statuses, *Standby* and *Working*. In *Standby*, the port *s_w_yn* of SCN is pulled up to a positive voltage (VCC). If there is a touch, the port is pulled down to *w_gnd*. This change can be detected as an input event to the system controller. When this change occurs, the system status switches to *Working*. In *Working*, every full cycle contains three subcycles, *X coordinate* (x) *measurement*, *Y coordinate* (y) *measurement*, and *touch level* (t) *measurement*. Every subcycle has two cycles, first one for initializing the screen circuit network, second one for obtaining and calculating data. Fig. 4 shows the system flow chart. Table I shows the signal values for setting assistant circuits in every status. X coordinate and Y coordinate are calculated as,

$$x = \frac{Z_{Y-}}{2^{W_{adc}}} \times W_{scr}, y = \frac{Z_{X-}}{2^{W_{adc}}} \times H_{scr}. \tag{5}$$

where Z_{Y-} is the ADC value converted from port *s_w_yn*, Z_{X-} is the ADC value converted from *s_w_xn*, W_{scr} is the width of the screen, and H_{scr} is the height of the screen. Determining the touch level needs two ADC values, Z_1 and Z_2, from port *s_w_xn* and port *s_w_yp*, separately. Touch level is defined as the difference between them,

$$t = Z_2 - Z_1. \tag{6}$$

TABLE I
ASSISTANT CIRCUIT SETTINGS FOR SCREEN CIRCUIT NETWORK IN *standby* AND *working* STATUSES.

Signal	Standby	X	Y	touch level
s_sw_xp_v_ctrl (bool)	1	1	0	1
s_v_xp_ctrl (V)	0.0	0.0	0.0	0.0
s_sw_xn_v_ctrl (bool)	0	1	0	0
s_v_xn_ctrl (V)	0.0	VCC	0.0	0.0
s_sw_yp_v_ctrl (bool)	0	0	1	0
s_v_yp_ctrl (V)	0.0	0.0	0.0	0.0
s_r_sw_yn_v_ctrl (ohm)	1000	1.0e12	1.0e-6	1.0e-6
s_v_yn_ctrl (V)	VCC	0.0	VCC	VCC

The precision of t depends on the ADC's output width. The wider the output is, the more accurate t is. The touch resistance value is not calculated here for the complexity reason. The difference of the ADC values shows the change of touch resistance too, which provides enough information we want to obtain. To make sure the conversion speed and precision fit the application request, the ADC conversion speed is set at least two times faster than request and the ADC output width is set as wide as possible (typically $5 \leq W_{adc} \leq 10$). For 8 cycles needed for status *working*, the clock speed of the system controller is at least 8 times of that of HTI (at least 400Hz).

Listing 2 shows the source code in SystemC-AMS for the control of assistant circuits according to Table I.

Listing 2. The source code for controlling assistant circuits.

```
#pragma once
#include "systemc-ams.h"
#include "config.h"

SCA_SDF_MODULE(ts4wctrl)
{
 sca_scsdf_in<sc_uint<3> > i_sel;
 sca_sdf_out<double> o_v_yn_ctrl, o_v_yp_ctrl,
      o_v_xp_ctrl, o_v_xn_ctrl, o_r_sw_yn_v_ctrl;
 sca_sdf_out<bool> o_sw_yp_v_ctrl, o_sw_xp_v_ctrl,
       o_sw_xn_v_ctrl;

 double vcc;
 void sig_proc()
 {
  if (i_sel.read() == 0)//standby
  {
   //X+ Gnd
   o_sw_xp_v_ctrl.write(1);
   o_v_xp_ctrl.write(0.0);
```

```
19
20      //X- Hi-Z
21      o_sw_xn_v_ctrl.write(0);
22      o_v_xn_ctrl.write(0.0);
23
24      //Y+ Hi-Z
25      o_sw_yp_v_ctrl.write(0);
26      o_v_yp_ctrl.write(0.0);
27
28      //Y- Pull up/Int
29      o_r_sw_yn_v_ctrl.write(1000);
30      o_v_yn_ctrl.write(vcc);
31    }
32    else if (i_sel.read() == 1)//X-coordinate
33    {
34      //X+ Gnd
35      o_sw_xp_v_ctrl.write(1);
36      o_v_xp_ctrl.write(0.0);
37
38      //X- Vcc
39      o_sw_xn_v_ctrl.write(1);
40      o_v_xn_ctrl.write(vcc);
41
42      //Y+ Hi-Z
43      o_sw_yp_v_ctrl.write(0);
44      o_v_yp_ctrl.write(0.0);
45
46      //Y- Hi-Z/ADC
47      o_r_sw_yn_v_ctrl.write(1.0e12);
48      o_v_yn_ctrl.write(0.0);
49    }
50    else if (i_sel.read() == 2)//Y-coordinate
51    {
52      //X+ Hi-Z
53      o_sw_xp_v_ctrl.write(0);
54      o_v_xp_ctrl.write(0.0);
55
56      //X- Hi-Z/ADC
57      o_sw_xn_v_ctrl.write(0);
58      o_v_xn_ctrl.write(0.0);
59
60      //Y+ Gnd
61      o_sw_yp_v_ctrl.write(1);
62      o_v_yp_ctrl.write(0.0);
63
64      //Y- Vcc
65      o_r_sw_yn_v_ctrl.write(1.0e-6);
66      o_v_yn_ctrl.write(vcc);
67    }
68    else if (i_sel.read() == 3 || i_sel.read() == 4)
69    //Z1 Z2
70    {
71      //X+ Gnd
72      o_sw_xp_v_ctrl.write(1);
73      o_v_xp_ctrl.write(0.0);
74
75      //X- Hi-Z/ADC
76      o_sw_xn_v_ctrl.write(0);
77      o_v_xn_ctrl.write(0.0);
78
79      //Y+ Hi-Z
80      o_sw_yp_v_ctrl.write(0);
81      o_v_yp_ctrl.write(0.0);
82
83      //Y- Vcc
84      o_r_sw_yn_v_ctrl.write(1.0e-6);
85      o_v_yn_ctrl.write(vcc);
86    }
87  }
```

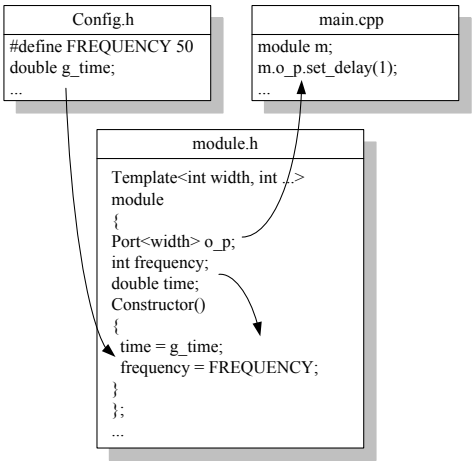

Fig. 5. Configuration methods for modules in the system.

```
88
89    SCA_CTOR(ts4wctrl)
90    {
91      vcc = VCC;
92    }
93  };
```

E. Configurable modules

When debugging and modifying the system, configurable parameters for modules are of great importance. Integer values as template parameters is suitable for defining the port, signal, and variable width. Other type of parameters, for example double values, are declared public inside modules and assigned, in the constructor *SC_CTOR()* or *SCA_CTOR()*, values of corresponding global variables declared in a single configuration head file. One global configuration head file is very clear and easy for debugging and modifying parameters. All the DE modules are written in a structural style with a sequential process and a combinational part. Registers in a DE module are put in a single C++ structure, which is declared inside this module as a signal of a custom type. The advantage of this style is similar to structural VHDL [11]. For SDF clusters, *sample delay*, *sample period*, and *port rate* are set in topper modules usually in *sc_main* for convenience. Fig. 5 shows configuration methods for parameters in modules.

III. Simulation Results

The simulation of the system is performed on a 16x16 resistive 4-wire touch screen. The highest frequency of the input signal for ADCs is 100Hz. The width and FSTM of ADCs are set to 6-bit and 2, separately. The moving speed of touch position is assumed within 50 points/s. The sampling frequency of ADCs is 100 times of the original analog signal. The testbench is created as moving pressed touch point on the screen with a certain resistance value for the touch resistor from x-y position (0, 0) to (15, 15) along the diagonal at speed 50 points/s with a no touch break at (6, 6) for 160ms. HTI works at 50Hz. ADCs work at 17.92MHz. The system controller works at 1kHz.

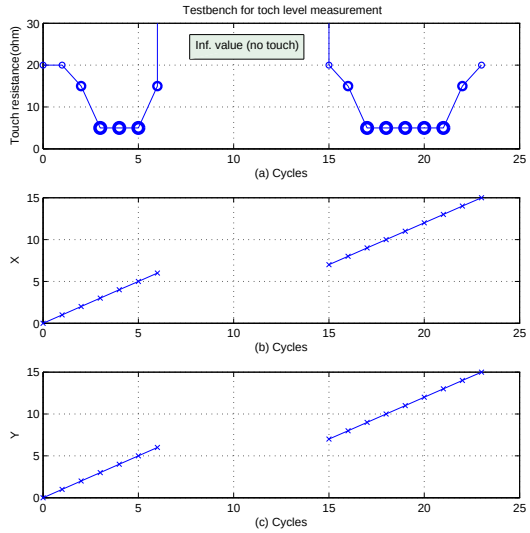

Fig. 6. The testbench input to HTI for the system.

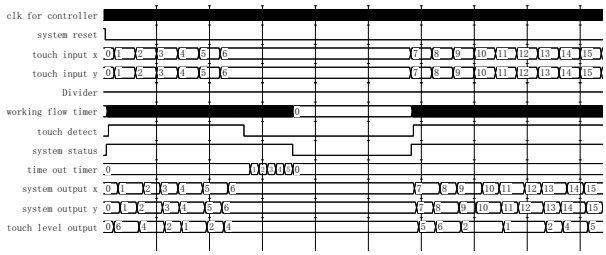

Fig. 7. The waveform of the simulation result of the resistive 4-wire touch screen system.

Fig. 6 shows the testbench input to HTI. The circles in subfigure (a) denotes the pressure of a human touch. The line width and the diameter of the circle increase, while the touch pressure goes up. However, the touch resistance is inversely proportional to the touch pressure. The timeout is set to 6 *working* cycles.

Fig. 7 shows the waveform of the above system output. The x coordinate and y coordinate obtained correctly. The touch level changes proportionally to the testbench value. The system status successfully changes from *working* (system status = 1 in the waveform) to *standby* (system status = 0) at 6 *working* cycles after no touch detected (touch detect = 0), then back to *working* after a touch interruption (touch detect = 1).

IV. Conclusion

The powerful method for modeling mixed-signal systems and subsystems in SystemC and SystemC-AMS has been demonstrated in this paper. A resistive 4-wire touch screen system has been modeled as an example. Different types of modules were chosen to model different parts of the system, for example, SCN as a LEN, LPF as a behavior module, and digital parts as DE modules. Different types of connections between modules were set up to construct the system, including bidirectional connections and feed back loops between CT and DE parts. Different rates were used for modules. Further, the methods for making modules configurable were introduced. Simulation results verified the functionality of the system model.

Acknowledgment

This work was supported in part by the project BDREAMS at Delft University of Technology in Netherlands.

References

[1] S. Swan, "An introduction to system level modeling in SystemC 2.0",*Open SystemC Initiative (OSCI)*, pp. 1, 2001.
[2] "SystemC version 2.0 user's guide," *http://www.systemc.org*, 2002.
[3] "SystemC SystemC-AMS 0.15RC4," *Fraunhofer IIS/EAS Dresden*, Feb. 2007.
[4] "Four and five-wire touch screen controller (rev. 8091A-AVR-07/07)," *ATMEL Application Note AVR341*, 2007.
[5] N. Brenner and S. Sullivan, "4-wire and 8-wire resistive touch-screen controller using the MSP430," *Texas Instruments Microcontroller Field Applications*, Feb. 2008.
[6] S. Paliy, "Four-wire, resistive-type touch screen with USB interface," *CYPRESS Application Note AN2376*, 17 Oct. 2006.
[7] J. Logue, "Virtex analog to digital converter (version 1.1)," *Xilinx Application Note XAPP155*, 23 Sep. 1999.
[8] J. Logue, "Virtex^{TM} synthesizable delta-sigma DAC (version 1.1)," *Xilinx Application Note XAPP154*, 23 Sep. 1999.
[9] "XPS delta-sigma analog to digital converter (ADC) (v1.00a)," *Xilinx Product Specification DS587*, 4 May 2007.
[10] "Selecting the right comparator," *MAXIM Application Note 886*, 13 Dec. 2001.
[11] J. Gaisler, "A structured VHDL design method," *CTH / Gaisler Research*.
[12] J. Bhasker, "A SystemC primer," *Star Galaxy Publishing*, 2007.
[13] C. Grimm, K. Einwich, and A. Vachoux, "Analog and mixed-signal system design with SystemC," *FDL04 Tutorial*, 16 Sep. 2004.
[14] M. Vasilevski, H. Aboushady, F. Pecheux, and L. de Lamarre, "Modeling wireless sensor network nodes using SystemC-AMS," *2007 Int. Conf. on Microelectronics*, pp. 53-56, 29-31 Dec. 2007.
[15] D. T. V. Pawluk and R. D. Howe, "Dynamic contact of the human fingerpad against a flat surface," *Journal of biomechanical engineering*, vol. 121, no. 6, pp. 605-611, Dec. 1999.

Mu Zhou received the B.S. and M.S. degrees from the School of Information Science and Engineering, Southeast University, Nanjing, P.R.China, in 2005 and 2008, respectively, all in electrical engineering.

Since 2008, he has been a PhD student in Circuits and Systems Group in the Department of Electrical Engineering, Mathematics and Computer Sciences, Delft University of Technology, Delft, Netherlands. His current research interests include receiver design for nano-satellite communication and mixed signal system modeling.