Synthesis of Embedded SystemC Design: A Case Study of Digital Neural Networks *

Djones Lettnin, Axel Braun, Martin Bodgan, Joachim Gerlach, Wolfgang Rosenstiel
Department of Computer Engineering
Tübingen University
Sand 13
72076 Tübingen – Germany
{lettnin,abraun,bodgan,gerlach,rosenstiel}@informatik.uni-tuebingen.de

Abstract

*This work presents the whole System-on-Silicon design flow using **SystemC** system specification language. In this study, SystemC is used to design a multilayer perceptron neural network, which is applied to an electrocardiogram pattern recognition system. The objective of this work is to exemplify the synthesis of **RTL- and behavioral integrated systems**. To achieve this, a preprocessing methodology was used to optimize the three main constraints of hardware neural network (HNN) design: accuracy, space and processing speed. This allows a complex HNN to be implemented on a single Field Programmable Gate Array (FPGA). The high level SystemC synthesis allows the straightforward translation of system level into hardware level, avoiding the error prone and the time consuming translation into another hardware description language.*

Keywords: SystemC Synthesis; Rapid Prototyping; Embedded Systems; Digital System Design; Hardware Neural Network (HNN); Electrocardiogram (ECG).

1. Introduction

The number of applications based on Rapid Prototyping for Embedded Systems and System-on-a-Chip (SoC) has been impulsed by the increase of technological advance and the *time-to-market*. New approaches have been proposed to improve the System-on-Silicon design flow, where a high-level description of circuits is the goal (for instance, behavioral model). To achieve this, it is mandatory to use a system-level description language. *SystemC*, a library based on C++ , allows creating models of software algorithms and hardware architectures for embedded systems as well as for SoCs, at different levels of abstraction [1]. Furthermore, the EDA market has been promoting new design tools, for instance *CoCentric SystemC Compiler* [2], to improve the design process and the prototyping of SystemC. Hence it is possible to define and simulate a system at a higher level and then translate it to a lower level using only SystemC in all phases of the design. Based on the aspects related above, the SystemC is used in this work to specify, simulate and synthesize the design to gate level.

Dedicated neurochips have been explored successfully by industry market, especially for real-time and embedded applications, such as pattern recognition, image processing, and speech processing [4]. The system under development is a Digital Neural Network applied to a complex application involving the Electrocardiogram Pattern Recognition [3].

To exemplify the whole System-on-Silicon Design Flow of SystemC, firstly, a *Hardware Neural Network* is designed in SystemC by means of RTL and Behavioral Models Integration. Secondly, the *CoCentric SystemC Compiler* and *FixTool Tübingen* [5] are used to refine and adapt the system to a synthesizable form. Thirdly, the system is synthesized into a XILINX VirtexE rapid prototyping board [6].

The article is structured as follows: *Section 2* presents an overview of SystemC. *Section 3* gives the basic concepts involving hardware neural networks, multilayer perceptron and electrocardiogram processing. *Section 4* describes the SystemC modeling. *Section 5* covers the conversion of the system into a synthesizable form. *Section 6* presents the HNN synthesized to the XILINX Virtex-E rapid prototyping board. *Section 7* concludes this work.

2. SystemC

New methodologies have been proposed due to the recent development of technology at all stages of the design process [7]. SystemC is a C++ class library and a methodology to design (cycle-accurate models) of software algorithms, hardware architectures and interfaces of SoCs.

SystemC reduces the *time-to-market* due to the linking of the system and the hardware designers' work. Initially, it is possible to describe the design in abstract level using a variety of powerful types of high level abstract models (well known transaction level), which allow to simulate and evaluate the design in a very fast form in an earlier phase [8]. Other important aspects are the code and the vertical test bench reuse [9]. The code reuse is intrinsic to the object-oriented properties used

* This work is partially supported by DAAD and FAPERGS.

by the designers at specification level. The vertical test bench reuse has a powerful capacity to validate the system, due to the reusing of the test bench in all stages of the design. At the final stage, the system can be translated into hardware level, avoiding the error prone and the time consuming translation into other HW description languages [8].

Therefore, considering these important aspects, SystemC is able to increase the productivity and the quality of the results, as well as to reduce the design costs. The main constructors of SystemC are consisted in the core language, which allows describing the functionality and communication. As shown in the Figure 1, SystemC has also a layered approach which enables the introduction of new high-level constructors [8].

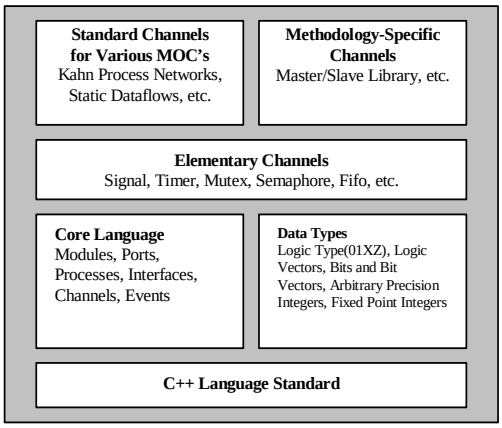

Figure 1: SystemC 2.0 Language Architecture [8].

2.1. Synthesis of SystemC

The synthesis of SystemC behavioral and RTL models is a new emerging possibility from EDA at the moment. Some evaluation of SystemC synthesis has been presented by other groups [10].

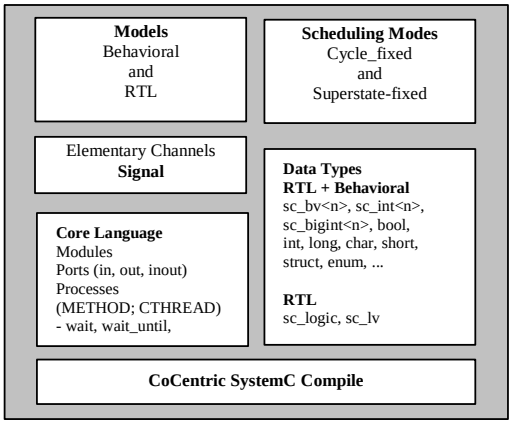

Figure 2: Synthesizable subset of SystemC.

In this work, the whole top-down design flow of SystemC synthesis is presented, while evaluating its restrictions and proposing solutions.

As shown in the Figure 1, SystemC is composed of rich architecture options, which however are not possible to synthesize, though CoCentric SystemC Compiler can successfully synthesize a subset of SystemC language constructors, summarized in Figure 2.

Two important restrictions of the synthesizable subset are the coding rules for superstate-fixed and cycle-fixed modes [2], and the data types. For instance, fixed point types can not be synthesized and for this reason, a tool is used to translate fixed point arithmetic into integer arithmetic. A detailed explanation is given in section 5.

3. Benchmark System Specification

3.1. Hardware Neural Networks (HNN)

Artificial neural networks (ANN) are based on the neural brain structure. They are powerful tools in many areas like pattern classification, function approximation, forecasting, fault tolerance and accomplishment in VLSI (Very Large-Scale Integrated) [4]. Most of the neural systems are implemented in simulators, where the fundamental drawback is the lost of the spatial-temporal parallelism [11]. HNN are capable to take advantage of the system performance. However, the hardware neural networks have to optimize three main constraints: *accuracy*, *space and processing speed*. The next section presents some approaches to optimize the design of digital neural network systems. This work uses the *multilayer perceptron (MLP)*, which is one of the most popular ANNs used as classifier [4].

3.2. Electrocardiogram Processing

The increasing number of people with cardiovascular diseases stimulates new research in pre-monitoring and pre-diagnostic of arrhythmia using the electrocardiogram (ECG) [3]. We have validated our specification based on the MIT/BIH Arrhythmia Database [12]. The records were sampled with 360 samples per second and a resolution of 11 bits. We have chosen patients (records) with three types of signals: *normal*, *fusion*, and *premature ventricular contraction* (PVC). However, the classification is not efficient applying the whole ECG segment. Minimizing the structure of the neural network, as much as possible, makes the system more efficient. To do this, the *Karhunen-Lòeve Transform (KLT)* method is used in the pre-processing phase, to extract some important characteristics of the ECG segments. The aim of KLT is to reduce the dimensionality of the data set, where the

importance of the variables is statistically evaluated [4]. In this way, it is possible to reduce the weight memory positions (WMP) and consequently the weights memory space (WMS) area. Thus, one electrocardiogram segment with 180 samples is reduced to 5 principal components. The compression ratio (CR) between the total number of data on the original form and the number of data using the KLT approach is 36 times smaller. *In hardware it means a reduction of 36 times the amount of memory/register used.* Therefore it can be possible to reduce drastically the weight memory space using principal components and keeping the high accuracy. The principal components are obtained by means of Generalized Hebbian Algorithms [4] and downloaded into the FPGA memory.

4. SystemC Modeling and Simulation

The neural network system was designed using five behavioral modules and one top-level (RTL) module. A behavioral model is an algorithmic description of the block's behavior. The *SC_CTHREAD*, a clocked thread process, is sensitive to one edge of a clock. This macro registers the member function declared previously as a behavioral process. The global reset is defined by means of statement *watching* within the constructor. The internal signals are used as communication between processes within the same modules. The following code presents the neuron designed by the standard SystemC behavioral modeling style.

```
// model_neuron_hid.h: header file
#include <systemc.h>
#include "usr_define.h"
class model_neuron_hid
: public sc_module
{
 public:
  // ports
  sc_in<sc_fixed<20,6> > X;   //Input
  sc_out<sc_fixed<20,5> > Y;  //Output
  sc_in_clk CLK;              //Clock
  sc_in<bool> RESET;          //RESET
  ...
  // Member Functions        //Forward Phase
  void Summup();             // Summup
  void ActFunction();        // Activation
  void AdjustWeights();      // Learning Phase
  sc_fixed<20,5> AF_derivate();
  ...
  void neuron_hid();         //Main Process

  // Variables
  sc_fixed<20,6> input[7];
  sc_fixed<25,6> wg[7];
  sc_fixed<20,5> delta_old[7];
  ...
  //default constructor
  SC_CTOR( model_neuron_hid)
    {
      // process declarations
      SC_CTHREAD(neuron_hid,CLK.pos());
      watching(RESET.delayed() == true);
    }
}; // end module model_neuron_out
```

The module structure of HNN is shown in Figure 3. The highest degree of parallelism can be reached if all the interconnections between neurons are simultaneous. However, if the number of connections is high, the constraint *space* will be also expensive. Therefore, a ***hybrid approach*** is proposed. Thus, between the input and hidden layer the *broadcasting principle* [13] is used: the input layer sends one data word to all the neurons of the hidden layer. Between the hidden and the output layer the *parallelism principle* is applied, and consequently all the data transfer will happen in parallel form (as shown Figure 3.b).

To simulate and evaluate the system, a well known method in ANN called "open test" [4] is used, which is a powerful way to analyze the pattern recognition system.

The open test means that the database used to test the neural network efficiency, was not consisted in the train database. In this work, we use one third of the database to perform the open test.

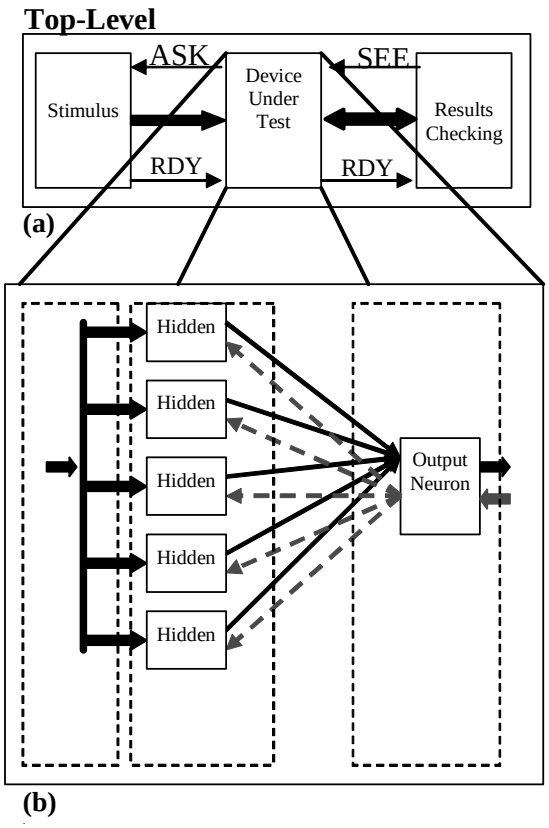

Figure 3: (a) Top-level RTL; (b) Behavioral Neural Network

The Table 1 shows the open test results, where we can observe the capacity of the HNN as a beat classifier. The ANN can classify correctly more then 95% of the

PVC signals, which is very important to pre-diagnose the fibrillation arrhythmia. We can also observe that the worst misclassification occur between the Fusion and Normal signals (error of 14.05%) because of the similarity between these signals. The mean accuracy of this simulation was **91.35**%.

Table 1: Open test results.

		MIT – DATA BASE		
	%	NORMAL	FUSION	PVC
HNN	NORMAL	**92.97**	14.05	2.16
	FUSION	7.02	**85.40**	2.16
	PVC	0	0.54	**95.67**

5. Converting to a Synthesizable Form

After defining and simulating the system, it is mandatory to refine and convert it to a synthesizable form. This section presents some issues to be considered for the synthesis phase.

5.1. SystemC Fixed-Point to Integer-Based Conversion

The use of floating point data types to define and simulate a system is an usual form to validate if the system is working properly. However floating point arithmetic has extremely high costs when implemented in hardware. One solution is the conversion of float point types to fixed point types based on bit widths and accuracy analysis. The tools FRIDGE [5] and CoCentric Fixed Point Designer [14] are two well known examples for this evaluation process. Most of the synthesis tools however, are not able to synthesize fixed point data types and their arithmetic operations and thus, we used the FixTool [5] to convert the fixed point data types and their arithmetic operations into integer-based system, as shown in the code below.

```
/* Fixed Point Format*/
...
sc_fixed<20,6> input[7];
sc_fixed<25,6> wg[7];
sc_fixed<25,6> sum;
...
sum = sum + input[i]*wg[i];
---------------------------------------
/* Integer Format */
...
sc_int<20> input[7];
sc_int<25> wg[7];
sc_int<25> sum;
...
sum=sum+((sc_int<25>)(((((sc_int<45>)(input[
i])))*(((sc_int<45>)(wg[i])))))>>14));
```

This automated conversion avoids the expensive time-consuming and error-prone usual manual work.

5.2. Scheduling Modes

The behavioral synthesis has two basic schedule modes [3]: the cycle-fixed, in which the SystemC Compiler keeps the cycle-to-cycle I/O behavior in the synthesized design as the previous behavioral description, and the superstate-fixed, where SystemC Compiler keeps the I/O behavior and adapts the number of cycles between the I/O operations. For this reason, it is necessary to implement an handshake protocol between the circuit and the testbench.

In this system, the superstate-fixed mode is used. It is more flexible to optimize the system through the SystemC Compiler. A two-way handshake protocol [2] is used between the circuit and the testbench and thus it is possible to use the same testbench in the design and synthesis. The main module requests (ASK) to receive or to send (X_RDY) data. Waiting until an ac knowledge is received from the transmitter (Y_RDY) or from the receiver (SEEN), as shown in Figure 4.

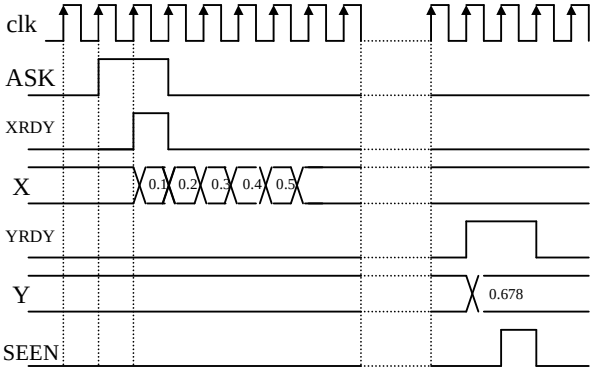

Figure 4: Two-Way Protocol.

The superstate-fixed mode has some coding rules for placing clock cycles, thus it was necessary to modify the initial specification. In the following examples, the *wait()* statements are pointed out to exemplify the coding rules of super-fixed mode. The system had to be adapted for four of six general rules [2].

1. *Place at least one wait statement in every loop except unrolled for loops*.

```
for(int i = 0; i < N_IN_WG; i++){
sum=sum+(sc_int<25>)(((((sc_int<45>)(input
[i])))*(((sc_int<45>)(wg[i])))))>>14));
  wait();
}
```

2. *Place at least one wait statement after the reset action and before the main infinite loop.*

```
…
//initialize the Ports
  SEND_INPUT.write(false);
  OUTPUT_RDY.write(false);
  wait();
  while(true){
```

```
    in_ptl();
    Summup();
…
  }
```

3. *Place at least one wait statement between successive writes to the same output.*

```
…
GRAD_SEEN.write(true);
wait();
GRAD_SEEN.write(false);
wait();
…
```

4. *If one branch of a conditional (if..else, …) has at least one wait statement in each of the other branches, including the default branch and implicit else conditions.*

```
…
if (LEARN.read() == true){
      out_ptl(); // wait() within method
      AdjustWeights();
    }else{ wait(); }
…
```

6. HNN synthesized to the XILINX VirtexE board

The first implementation of HNN described in SystemC is validated on a SPYDER-VIRTEX-X2E board. This board has one XILINX VirtexE-Series FPGA of the XCV2000E family (2.000.000 equivalent logic gates) [6]. The block diagram and the board picture is show in the Figure 5.

The CoCentric SystemC Compiler is able to synthesize integrated RTL and behavioral modules into gate-level netlists or HDL RTL descriptions [2]. In our case, a HDL RTL description was obtained and was applied to the Xilinx low-level synthesis tools. This synthesis phase is composed of mapping and fitting the design into a specific device (for instance, XCV2000E). The automatic synthesis is completed after the successful routing and bit stream generation.

The use of preprocessing path allows the integration of weights in a single-chip. Thus it is possible to achieve a higher real-time throughput without the bandwidth problem of weights external memory communication.

HNN was applied to the Electrocardiogram Pattern Classification, and a second simpler XOR Problem was additionally used to evaluate and to compare the results.

The benchmarks are composed of two complex non-linearly separable systems, meaning that the HNN requires at least one hidden layer. The applied neural architecture was:

- ECG: 7 inputs / 5 hidden neurons / 1 output neuron;
- XOR: 2 inputs / 2 hidden neurons / 1 output neuron;

The implementation results of the Xilinx board are exposed in Table 2. The first two columns present the elements and the maximal number of them on Xilinx board. The last two columns present the synthesis results of benchmarks and testbench together. It is possible to observe that the duplication of neurons cause one increment of almost five times in Slices size. However the clock speed is reduced to the half.

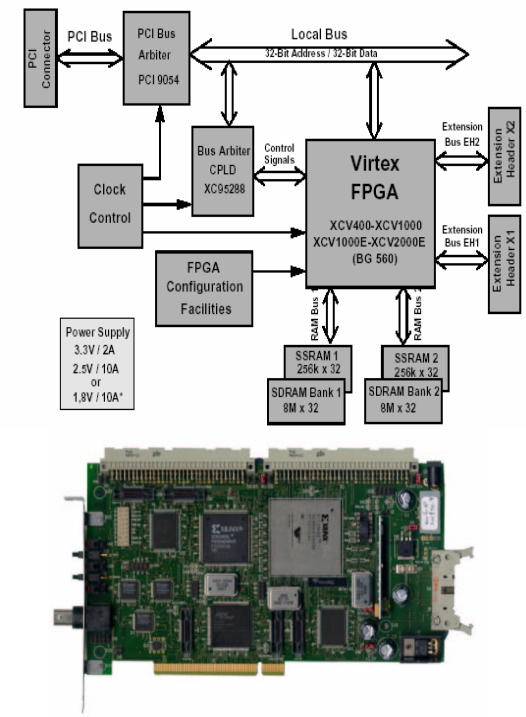

Figure 5: Block Diagram and board picture [6]

Table 2. Processing Elements

XCV2000E	**Max**	**ECG****	**XOR**
Slices*	19200	14826	3009
Slice Flip-Flop	38400	5236	1440
4 input LUT	38400	27529	5403
IOB	404	3	3
GCLKs	4	1	1
GCLKIOBs	4	1	1
CLK Freq. MHz	33	7	16

* One CLB is compound of two slices.

** The result does not consider the training and test data set memory.

7. Conclusion

This paper presented the design and synthesis flow of a SystemC design. As a case study, a digital neural network is applied to the electrocardiogram pattern recognition problem. Firstly, the system is designed and simulated, achieving the expressive results of **95.67%** of PVC accuracy, and then it is adapted to synthesizable form. Finally, the HNNs examples were synthesized into a Xilinx hardware target.

As shown in this investigation, the CoCentric SystemC Compiler supplies good results to automatically synthesize RTL and behavioral integrated systems. Our solutions for the fixed point data type conversion (FixTool) and scheduling mode (coding rules) were pointed out. Therefore, the synthesis of SystemC designs offer a new possibility for commercial market to synthesize high level hardware design in an efficient and economic form.

8. Bibliography

[1] SystemC Version 2.0 User Guide.
http://www.systemc.org

[2] Synopsys, *CoCentric SystemC Compiler*. Behavioral User and Modeling Guide,2002.

[3] Vargas, F., Lettnin, D., Felippetto de Castro, M. C., Macarthy, M. "Electrocardiogram Pattern Recognition by Means of MLP Network and PCA: A Case Study on Equal Amount of Input Signal Types". SBRN2002.

[4] Haykin, S. *Neural Networks.* Editor Prentice Hall, 1999.

[5] Braun A. G., Freuer Jan B., Gerlach J., Rosenstiel, W. "Automated Conversion of SystemC Fixed-Point Data Types for Hardware Synthesis" IFIP VLSI SOC 2003;

[6] Weiss, K., Steckstor, T.,Oetker, C., Esser, K. Embedded Systems Design Group. User's Manual. Spyder-Virtex-X2E.

[7] Mueller, W., Rosenstiel, W., Ruf, J. SystemC Methodologies and Applications. Kluwer Academic Publishers, 2003.

[8] Groetker, T., Stan Liao, Grant Martin, Stuart Swan, *System Design with SystemC*, Kluwer Academic Publishers, The Netherlands, 2002.

[9] Sayinta, A., G. Canverdi, M. Pauwels, A. Alshawa, W. Dehaene,"A Mixed Abstraction Level Co-Simulation Case Study Using SystemC for System on Chip Verification", Designers' Forum, DATE03.

[10] Bruschi, F., F. Ferrandi, "Synthesis of complex control structures from behavioral SystemC models", Designers' Forum, DATE03.

[11] Ramacher, U., Ulrich Rückert, *VLSI Design of Neural Networks*, Kluwer Academic Publishers, The Netherlands, 1991.

[12] MIT/BIH database distributor, Beth Israel Hospital, Biomedical Engineering, Division, available in the address: http://ecg.mit.edu/, USA, 1979.

[13] Ossoinig, H., E. Reisinger, C. Steger, R. Weiss, "Design and FPGA-Implementation of a Neural Network", available in the address: www.icspat.com/papers/493mfi.pdf

[14] Synopsys, Inc. CoCentric Fixed-Point Designer User Guide, version 2002.05 edition, June 2002.