Analysis of "SystemC" design flow for FPGA implementation

Kartika S.
Research Scholar, Indonesian University
karitkaindo@gmail.com

Article History

Article Submission
19 November 2015
Revised Submission
17 January 2016
Article Accepted
1 March 2016
Article Published
31st March 2016

Abstract

High level language termed as SystemC language is recently gaining popularity in VLSI industries especially in Hardware-Software co-design. Using SystemC, Hardware IPs can be modeled at system level which helps to reduce the time to market for SOCs. In most applications SystemC is utilized to verify functionality of the design. However there has been relatively less work done on the synthesis of equivalent hardware from SystemC. In this paper, Finite Impulse Response Filter and Greatest Common divisor are designed as examples in SystemC language and their corresponding synthesis flow from SystemC to FPGA is proposed. The proposed method of synthesis would be time saving than the conventional design and synthesis using HDL in RTL perspective.

Keywords: *SystemC, System level Synthesis, Questasim, Xilinx ISE, Hardware Software Codesign, Simulation.*

I. Introduction

System-level design methodology involves attributes to system specifications, verification of design functionalities and determination of optimal architecture through guided evaluation alterrnatives [1][3]. Modern day complex architectures integrate software into chip and termed as system-on-chip (SoC). These system level designs have many hindrance and one possible solution for this is the use of SystemC[2][8]. SystemC comes from C++ which is familiar with hardware architects; system oriented designers and advanced software design engineers. In order to model the hardware systems at many levels of abstractions, required constructs are added to C++ by SystemC classes. So, it enables the system level design by its object oriented programming approach [4][5].

SystemC find its importance in performing verification of system functionalities through executable constructs, provided there exists design tools and IP vendors who can provide SystemC models. The challenges faced in hardware software codesign in the advanced ASIC technologies are solved through a common platform. "Time to market" issue is to be solved by the standard modelling platform and it is desirable by SOC designers to reuse the Intellectual Properties for multiple designs. Earlier, in order to have IP creation and IP integration in high level SoC design, Synthesis tools required Verilog/VHDL only as an entry language [1]. SystemC Compiler – a Synthesis tool for SystemC which uses synthesizable subset modules was introduced by Synopsys in June 2000. After which SystemC has been widely accepted by Hardware design community [6][7].

The behavioural level synthesis of SystemC models have been presented in [3][4]. The paper [3] presented traditional compiler techniques for synthesis of SystemC, which ultimately gave a start to a closer relationship between hardware and software descriptions and development tools. The cycle accurate timing behaviour of synthesized PCI bus model in SystemC has been analysed in [4]. SystemC^{FL}, a classical process algebra, which is based on an algebraic theory have been developed to specify and analyse SystemC based designs. The semantics of SystemC^{FL} were given by deducing rules in a standard structured operational semantics style which connect a labelled transition system to a SystemC^{FL} process [6].

ISSN: 2250-0839

The mapping of memory arrays in a high-level SystemC description to hardware was done [7]. In Paper [7], assignment of access to read or write ports was done with an algorithm provided. A merged combination of C/C++ and SystemC approaches known as SystemC synthesizable superset was given in [8]. OSCI's synthesis working group (SWG) has defined the suitable synthesis subset for SystemC language for High Level Synthesis [9]. Authenticated verification of SystemC by sophisticated software model is explored in [13]. A design outline and methodology for scrubbers based on SystemC modelling is described in [9]. The given framework includes Models such as Scrubber, Supervisor, FPGA and a Fault model.It was observed that most of the work done with respect to SystemC are related to verification environment and enhancing the modelling at different levels of hierarchy. However, there were little work done with respect to design of behavioural level IP components and respective synthesis of SystemC Threads and Modules and equivalent HDL generation for the target FPGA. Hence in the following chapters, system level design of Finite Impulse Response (FIR) Filter and Greatest Common Divisor (GCD) modules in SystemC, is explained with respect to hardware software co-design perspective. The proposed synthesis flow is demonstrated using flow chart in which synthesis of SystemC modules is done by converting SystemC processes to equivalent HDLs using it as an intermediate step. Next, this HDL can be further synthesized and optimized using Xilinx ISE Tool [10].

II. Related Work

SystemC is a common platform for system architect designers, software critic engineers, and hardware based designers. SystemC is a C++ library with class definitions. Compiling (using a C++ compiler) and running it will perform the simulation. This is shown in figure 1.

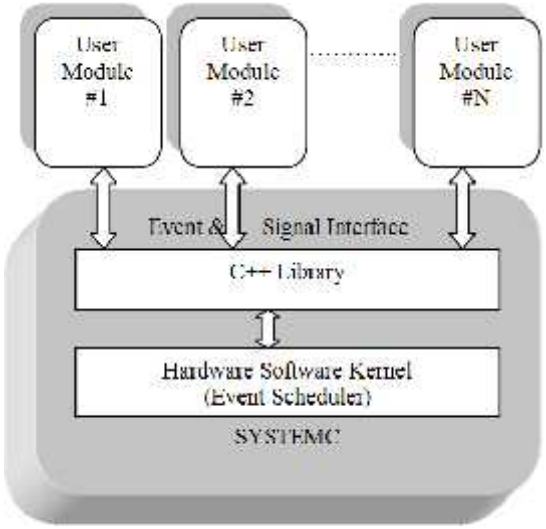

Fig 1: SystemC and user modules.

Demonstration of writing SystemC program is discussed in this section. SystemC is just a set of connected modules. A module can later be realized in hardware or software. The general steps for the design of digital circuit which contain several connected gates are given below.

a) Description of a module
b) Instantiations and connections between multiple gates.
c) Simulation of this circuit module.
d) Visualisation of the output of the simulator

A module is the basic unit for describing structure in SystemC and for partitioning a design. A module is described via a module class SC_MODULE. A module can be hierarchical in that it can have processes and

ISSN: 2250-0839

other modules instantiated within it. A process is used to describe functionality. Processes appear inside modules. There are three kinds of processes that can model different process abstractions. They are SC_METHOD, SC_THREAD & SC_CTHREAD. A process is used to describe the functionality of the system and allows the expression of concurrent behavior. Furthermore, processes are not hierarchical. So, Hierarchy is implemented in SystemC by using the module, linked through ports. This is shown in figure 2.

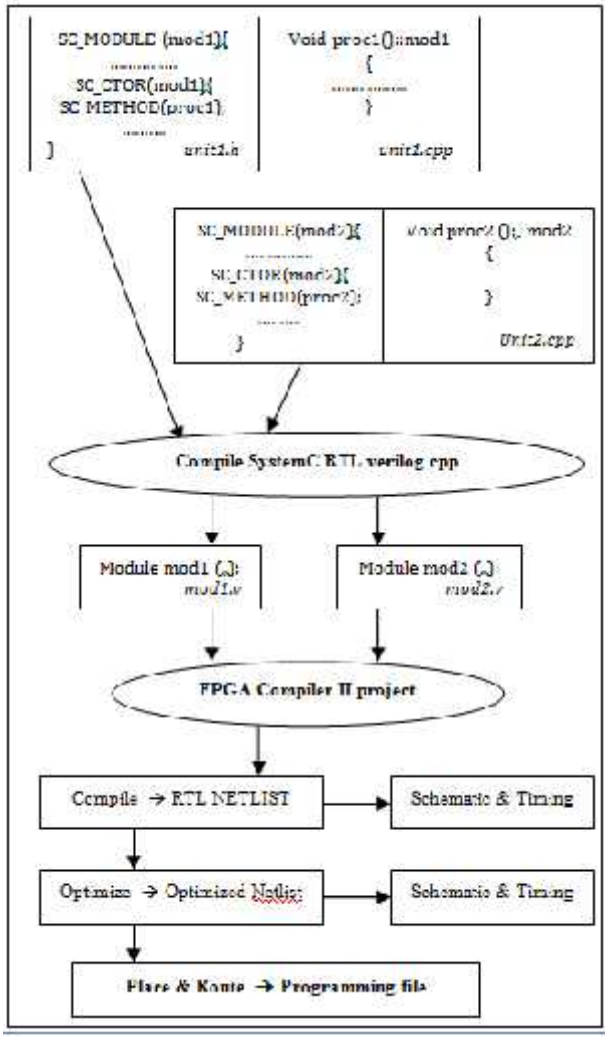

Fig 2: Existing SystemC synthesis flow from Synopsys

III. Proposed Systemc Synthesis Flow

The proposed synthesis flow is time saving and has less efforts and also cost effective, can be targeted for most of the FPGA boards. The basic advantage of this proposed method is, it can use any of the C++ compiler files that are associated with windows/Unix system. As shown in Fig 3, The Verification of the SystemC code and design is done through Mentor graphics Questasim tool.

The design along with test bench is then exported to The Systemcrafter to convert the SystemC code to equivalent HDL. Here, C++ compiler of the system is used to check code standard and for compilation. The generated HDL (Verilog/VHDL) is synthesized by Xilinx ISE for the targeted FPGA board. For example, SPARTAN 3E board.

ISSN: 2250-0839

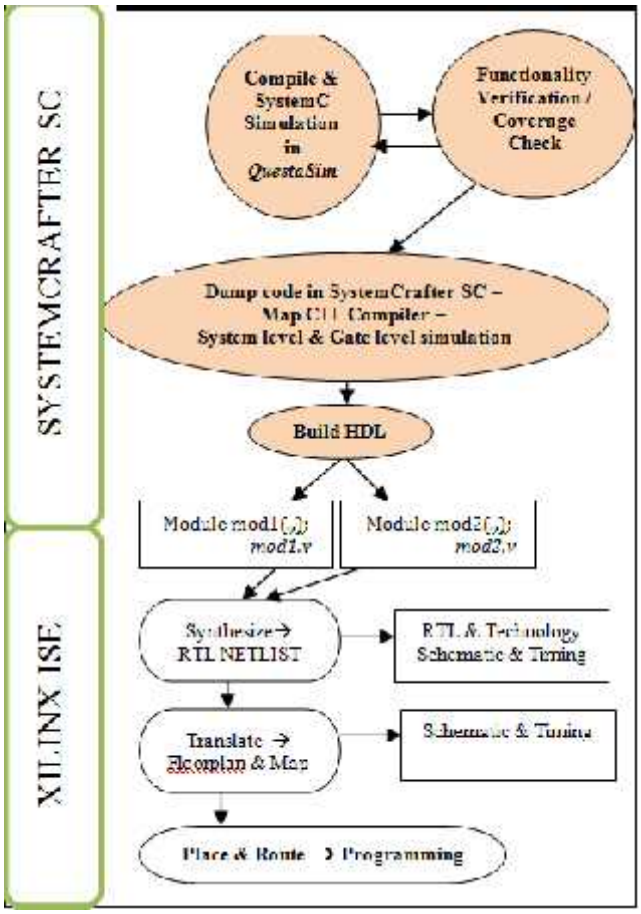

Fig 3: Proposed SystemC synthesis flow using Questasim.

There are two simulations performed by SystemCrafter tool before producing HDL.

a) The System-level simulation: simulates the SystemC code that was written (the input to SystemCrafter).
b) The Gate-level Simulation: simulates a SystemC description of the HDL that SystemCrafter has synthesized (the output from SystemCrafter).

By simulating these descriptions, the circuit that has been written and the circuit that SystemCrafter has synthesized can be verified. The SystemC and HDL that SystemCrafter generates are alternative descriptions of the same hardware. The FIR design in SystemC is based on sc_buffer object to act as delay element. This filter comprises of five delay components clocked with respective illustration frequency. The weighted sum is obtained from the output of each delay element. The FIR filter can be modeled in SystemC in several different ways. There are many choices to make, some of which are listed below. Each SG operator node can be used in a separate assignment statement or several operators can be combined in a compound expression.

```
SC_MODULE(FIR) {
 sc_in<sc_logic> clk;
 sc_in<double> sample;
 sc_out<double> result;
 SC_CTOR(FIR) {
 SC_METHOD(behavior);
 sensitive << clk.reg();
 }
private:

 sc_buffer<double> i1, i2, i3, i4, i5;
void behavior() {
 double i0 = sample.read();

// double v0 = -0.07556556070608 * i0;
// double v1 = 0.09129209297815 * i1.read();
// double v2 = 0.47697917208036 * i2.read();
// double v3 = 0.47697917208036 * i3.read();
// double v4 = 0.09129209297815 * i4.read();
// double v5 = -0.07556556070608 * i5.read();
 double v0 = 1 * i0;
 double v1 = 1 * i1.read();
 double v2 = 1 * i2.read();
 double v3 = 1 * i3.read();
 double v4 = 1 * i4.read();
 double v5 = 1 * i5.read();
 double s1 = v0 + v1;
 double s2 = v2 + v3;
 double s3 = v4 + v5;
 double s4 = s1 + s2;
 double s5 = s4 + s3;
 result.write(s5);
 i1.write(i0);
 i2.write(i1.read());
 i3.write(i2.read());
 i4.write(i3.read());
 i5.write(i4.read());
 }
};
```

Fig 4: FIR FILTER – SystemC code

Figure 6 shows a Vedic 4-bit architecture which is proposed here. Dependent on selection lines inputs A and B executes ALU functions. We here employ 8:1 and two 2:1 multiplexer. Input B, 2's complement is received by 2:1 multiplexer. If all the selection lines are zero, A is added with B directly, or else, A and input of 2's complement together added to produce output subtraction. All the arithmetic functions at the output are received by 8:1, employing selection lines logical functions are produced at the output. Logical functions output may be excess 4-bit that is of two state. Logical operations attained will observe in output of LSB. Arithmetic functions output may be also excess 4-bit and is observed in MSB.

IV. Simulation Results

The behavior of the FIR filter in SystemC is described in a SC_METHOD, the delay elements are modeled explicitly by using sc_buffer objects as shown in Fig 5. The sample input sequence x(n)=[0 1 2 3 1 0 -1 -2 -3 -4 -5] with weights as w(n)=[1 1 1 1 1 1 1 1 ……] is given to the FIR Filter design. The simulation results using Questasim is shown in Fig 6. The functionality is verified.

ISSN: 2250-0839

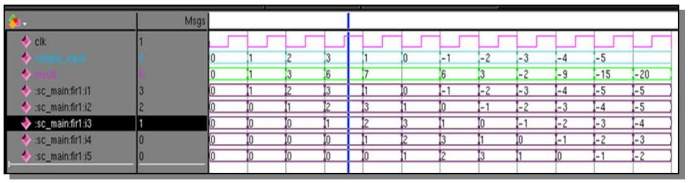

Fig 5: FIR FILTER – Simulation Results.

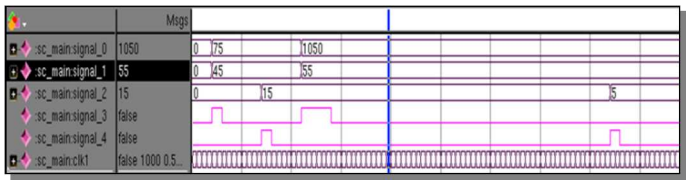

Fig 6: GCD – Simulation Results.

The equivalent RTL schematic generated for GCD is shown in Fig 7 for target SPARTAN 3E Board.

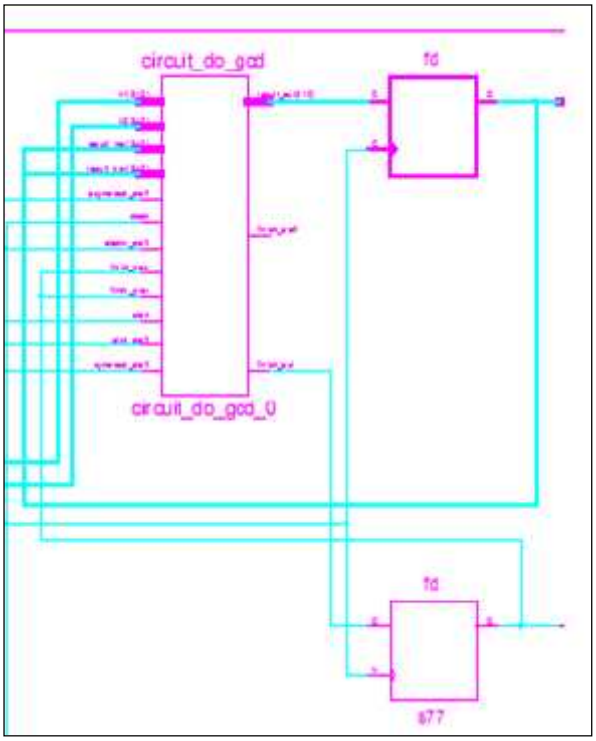

Fig 7: GCD – Part of RTL Schematic from Xilinx ISE

The HDL synthesis report summary of number of hardware components and the logic utilization for FPGA implementation is given in Table 1.

TABLE 1: GCD – Verilog – Synthesis report from Xilinx

Logic utilization	Used	Available	utilization
Number of Slices	152	960	15%
Number of Slice Flio flops	108	1920	5%
Number of 4 input LUTs	260	1920	13%
Number of bonded IOBs	99	108	91%
Number of GCLKs	1	24	4%

ISSN: 2250-0839

Hardware Utilization	
Number of Adders/Subtractors	2
Number of FlipFlops/Latches	108
Number of Comparators (32 bit)	2
Maximum Frequency of operation	80.702 MHz

V. Conclusion

The proposed synthesis flow is tested with FIR and GCD, taken as behavioral design examples. Modeling at high level in SystemC is simple and time saving. Hence such models which are coded in SystemC, by following the subset rules can be converted to HDLs by the tools given in proposed synthesis flow and then follow the traditional synthesis from HDL. This ultimately reduces the time to design and synthesis complex systems and ends up with faster Time to market.

References

[1] Schlebusch, H.-J. ; Euromicro DSD, Maastricht, Netherlands, "SystemC based hardware synthesis becomes reality," Euromicro Conference, 2000. Proceedings of the 26th, vol. 1, Sep 2000.

[2] Synopsys, "Describing Synthesizable RTL in SystemC", Version 1.0, May 2001

[3] Economakos, G; Oikonomakos, P. ; Panagopoulos, I. ; Poulakis, I. ; Papakonstantinou, G. "Behavioral synthesis with SystemC", in Proc., Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001, pp. 21 – 25.

[4] Bruschi, F. ; Politecnico di Milano, Italy ; Ferrandi, F., "Synthesis of complex control structures from behavioral SystemC models", Published in Design, Automation and Test in Europe Conference and Exhibition, 2003, pp. 112-117 suppl.

[5] Grimpe, E. ; Inst. of Res., OFFIS, Oldenburg, Germany ; Oppenheimer, F., "Extending the SystemC synthesis subset by object-oriented features", Published in Hardware/Software Codesign and System Synthesis, 2003. First IEEE/ACM/IFIP International Conference, Newport Beach, CA, USA, Oct 2003.

[6] Man, K.L. ; Dept. of Math. & Comput. Sci., Eindhoven Univ. of Technol., Netherlands, "An overview of SystemC" , Research in Microelectronics and Electronics, vol. 1,pp. 145 - 148 , no. 1 , July 2005.

[7] Ditmar, J. ; Celoxica, Abingdon ; McKeever, S., "Array Synthesis in Systemc Hardware Compilation", Published in Field Programmable Logic and Applications, 2007. FPL 2007. International Conference, pp. 23-28, Amsterdam, Aug 2007.

[8] Smirnov, M. ; Mentor Graphics Corp., Wilsonville, OR, USA ; Takach, A., "A SystemC superset for high-level synthesis", Published in Specification & Design Languages, 2009. FDL 2009. Forum, pp. 1-6, Sophia Antipolis, Sept. 2009.

[9] D. Kulakovskis and D. Navakauskas, "Automation of metabolic P system implementation in FPGA: A case study," 2015 IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), Riga, 2015, pp. 1-4, doi: 10.1109/AIEEE.2015.7367289.

[10] S. Boukaka, H. Teiar and P. Sicard, "FPGA implementation of a library of adaptive control laws for PMSM," 2015 Conference on Design of Circuits and Integrated Systems (DCIS), Estoril, 2015, pp. 1-6, doi: 10.1109/DCIS.2015.7388573.