Advanced Topics in Electronic System Level Design

Final Report

N26094859 蔡宗穎

Abstract: From the 1 semester ESL course, it is important that we should summarize all the ideas and knowledge we've learned, and then combined with our experience to learn and use those concepts and techniques in the future. In this report, I illustrate the ideas of why we need to use ESL, the relationship between knowledge, action and communication, and what is ESL in details combined with the experience from the assignments and the term project.

Part 1. Introduction

Moore's Law is a famous law that mentioned in many courses and articles related to semiconductor industry and electronic systems, which stated that the number of transistors on a chip, roughly doubles every 18 months…, while the chip price remains the same[1]. As a result of Moore's Law, the number of transistors and the complexity of the electronic systems are surely to increase tremendously. As the time goes on, after the multi-core processors being produced, some critical problems emerged and caused the failure of many VLSI system designs. Thus, some people came up with an idea of designing a system-level in an easier and low cost approach that can integrate hardware and software and verify in early stages of the VLSI system design, which is ESL that I will discuss in this report.

The rest of the report is organized as follows. Reasons for using ESL is mentioned in Part 2, the relationship between knowledge, action, and communication is stated in Part 3, and the concrete ideas of ESL is illustrated in Part 4. Some conclusions are presented in Pare 5.

Part 2. Why use ESL

As mentioned earlier, there are two main reasons for using ESL. One is the increasing design complexity of hardware, and another is the shortened design cycle[2].

2.1 The increasing design complexity

From Moore's Law we know that the number of transistors growing rapidly, as well as the complexity of the VLSI system design. This results in many problems in a period of time. First, traditional design flow of a VLSI system design didn't have verification in system-level in early stages, which leads to some crucial design defects

not noticed while designing and then found after the chips were taped out and lost lots of money. Second, the software design can only be verified after the hardware is finished. Hence, it's possible that the performance is not suitable for the target software developed. Third, with old design flow, it's also possible that the cost of the system is much higher than their competitors, so does the price of the product.

2.2 The shortened design cycle

Customers are always eager to get a new device for their own reasons, but the need for new electronic systems in the marketplace leads to the shortened design cycle. Thus, an effective modeling/verification tool for the overall system which can speed up their design flow is desired. ESL is such a tool the industries needed.

Hence, system level modeling such as ESL is very important, it can help us analyzing the algorithm, dataflow and memory access, which leads to a low power design. At the same time, it can let the software design begins as early as the hardware, which reduced the design cycle. Thus, using ESL increase the rate of success of a product. These are the reasons why we should use ESL.

Part 3. The knowledge, action, communication, and synchronization in a multicore system

In this part, the relationship between the knowledge, action, and communication is illustrated in the form of "Cheating Husband Puzzle" originated from the article [3]. In this paper, the women are the processors in the distributed system, whether there are cheating husbands, whom are cheating husbands (except for their own husbands) that everyone knows and the date the wives get the information is the knowledge, whether the husband is killed is the action made by the women (processors), and the means that the queen giving the message that there are cheating husbands is the communication in either the form of synchronous and asynchronous.

3.1 Cheating Husbands Puzzle

The story of Henrietta I told us that the all processors could act based on the knowledge they received.

3.2 Results of asynchronous communication

The story of Henrietta II told us the result of asynchronous communication. It only works if there is only one cheating husband, but if there are more than one cheating husband, then no wives could be sure that their husbands are faithful or not, thus no cheating husbands were harmed because they didn't know the exact date the mail was sent.

3.3 Results of synchronous communication

Henrietta III's story is an example of weakly synchronous system that the letter should be received within b days, which corresponding to the concept of b-common knowledge. Although this assured that many cheating husband would be killed, but some wives may be confused because of the lack of the information about the date other wives received the mail. Also, if they waited for d days after they found that their husbands were cheating, then they can determine the faithfulness of their husbands once the k * (b + d) silent night pass after she received the letter. Furthermore, if $d \geq b - 1$, then all cheating husbands would be killed. Additionally, if a wife bribed the mailperson and got the initial date, she would finally figure out the faithfulness of her husband.

Henrietta IV's story is an example of strongly synchronous system that the initial date was on the mail, and it also mentioned about the situation if some wives were disobedient. If there's only one cheated wife and she didn't kill her husband, then other wives wouldn't shoot their husbands. However, if everyone knows that there is at least one obedient wife, then all obedient wives' cheating husbands would be killed.

3.4 Results of ring-based communication

The story of the queen of Mamaringa told us that in a ring-based communication, if the system is asynchronous, then the last wife received the mail would kill her husband and other wives would not because they can't figure out whether their husbands were cheating, so k-1 nights after the last wife received the mail was silent. For weakly/strongly synchronous system, some cheating husbands were killed and some are not because some wives were confused.

3.5 Quick elimination

Queen Margaret's story gave us an efficient way shooting all cheating husbands. The wives were grouped into 3 groups, group 1 with $k_0 \% 3 = 0$, group 2 with $k_1 \% 3 = 1$, and group 3 with $k_2 \% 3 = 2$. Noticed that there are only two possible number of cheating husbands, i.e. k and k-1. So it should be 0/1, 1/2, or 2/0 groups, with the formal (0 in 0/1, 1 in 1/2, 2 in 2/0) having cheating husbands. The protocol is as follows. A wife knowing $k_0 \% 3 = 0$ cheating husbands should shoot into the air and shoot her husband only if $k_0 = 0$ in the first night. In the second night, if it's silent in the first night, then a wife knowing $k_1 \% 3 = 1$ killed her husbands; if it's not silent in the first night, then a wife knowing $k_2 \% 3 = 2$ killed her husbands. In the third night, if both the first and the second night are silent, then all wives need to shoot their husbands; if there's shoot in the first night and silent for the second night, then a wife knowing $k_0 \% 3 = 0$ (who shoot in the first night) killed their husbands if they were still alive.

The above stories gave us many concrete examples how the processors act based on their knowledge from either type of communications in the distributed computing systems, with the explanation why some processors act what we want them to do but some don't, and why the problems occur.

Part 4. What is ESL and how it is used

In this part, I will explain the fundamental concepts of ESL and how it is used in the assignments and the term projects in this semester.

4.1 What is ESL

As mentioned earlier, ESL is the bridge between the hardware and software, and it is a very useful methodology for doing verification of the overall system with both hardware and software, and shorten the design cycle.

In the following sections, I will explain the ESL design methodology first, then talk about the transaction level modeling, and how the ESL platform is built. Additionally, model-based design and hardware dependent software are also mentioned.

4.1.1 ESL design methodology

Fig. 1[4] is a concrete flow of ESL design methodology. In system level design, we need to start from the specification of the system, then create functional models using C/C++, estimating the amount of data used, which is also called data-flow analysis, and the amount of computation needed. After those parameters are determined, check for shared resource conflicts, i.e. reduce the hardware resource if the timing of the system will not be influenced to have lower costs. The last part in system-level design is HW/SW partitioning. In this part, if the algorithm, or function, is not heavily-computed, or it can be executed in real-time using software (CPU), then this specific part will be assigned to software part. On the other hand, if the algorithm is not suitable or can't achieve the timing in the SPEC in software, then it will have a specified hardware to run this algorithm.

After the HW/SW portioning, the hardware team will continue to explore the architecture, and the software will develop software applications after some hardware prototype is ready in the traditional flow. However, in the ESL design flow, some software can be developed as early at the stage of architecture exploration, and the software team can give feedback to the hardware team after doing verification based on the hardware models written in C/C++ with both HW and SW using the virtual ESL platform.

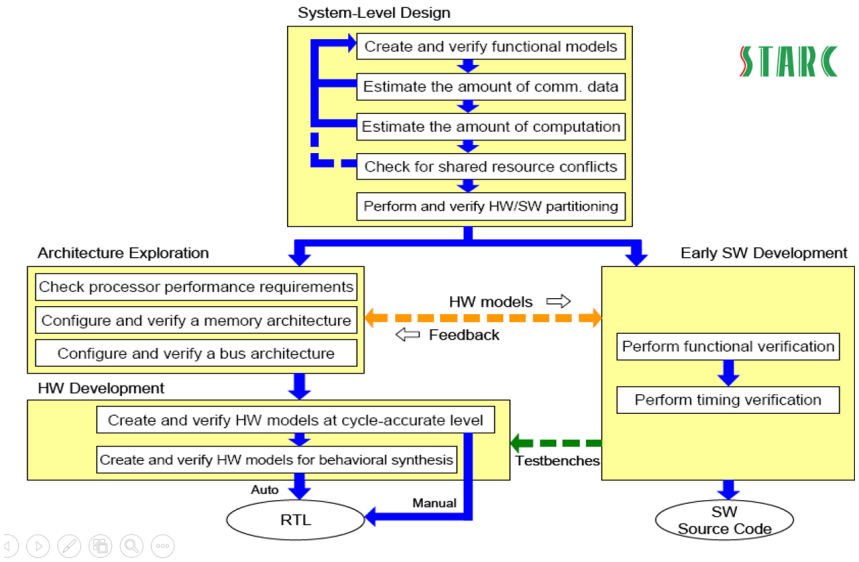

Fig. 1 The ESL Design Methodology, Japan from source[4]

Thus, if one architecture is not suitable for the software, then the hardware team only have to change the architecture (and even change the algorithm sometimes.) In the stage of HW development, the software team could also give testbenches used in functional verification, which actually verified the functionality of the hardware design. Note that ESL is not time-accurate.

4.1.2 Transaction Level Modeling

Transaction Level Modeling (TLM) is modeling using bus transactions, defined as addressed data reads and writes[5], and verifying the functionalities of the bus. Thus, transaction level is token based, and token is an object which represents the right to perform some operation[6].

The main reason for using TLM is its faster simulation time than RTL, and it is the bridge between software development and hardware verification. There are several cases to use it. It can represent some components in a hardware platform, participating in architectural exploration and modeling performance of a system, being part of the virtual hardware model to execute some software, and being the golden model to do hardware verification[5].

There are two kinds of coding styles in TLM 2.0. One is loosely-timed and another is approximately-timed. The former provides less timing information and it is good at verifying the functionalities for components; the later provides more timing information and locked to the simulator scheduler, and it is suitable for architecture exploration and performance modeling.

There are also two kinds of transport interfaces in TLM 2.0, i.e. blocking and non-blocking interfaces. The blocking interface is often used with loosely-timed style because its basic time unit is called “quantum,” which is the time equal to the basic unit of execution cycle of the longest process. Hence, if process 1 was executed

before process 2 in the current quantum, and process 2 have transactions pass to process 1, then process 1 will receive it in the next quantum, which is not accurate for timing (note that it is the accurate data for the functionalities.) The non-blocking interface is often used with approximately-timed style because architectural exploration and performance modeling may include arbitration, which have to specify detailed timing.

4.1.3 Building the ESL platform

A virtual (ESL) platform is a truthfully mimic target system's address mapping and IP functionalities[4]. We use the platform to do HW/SW co-design, co-verification, and co-debug[7]. The hardware in the virtual platform are models written in C/C++ and thus can be simulated with the software. For verification, the ESL way is to verify functionalities first, then deal with timing. One advantage of the ESL way is that the C++ vectors can be used in FPGA emulation and Verilog test vectors. We can also debug while performing the FPGA emulation.

4.1.4 Model-based design

The basic idea of model-based design is to use some specific mathematical formula in the form of "models" to represent the controller or devices being controlled by the controller, like Fig. 2[8] shown below. It is viewed as the cost-effective and provably correct solutions for exploration and synthesis, and it allows to advance the analysis, validation, and verification[8]. Some usages of model-based design are Finite State Machine (FSM), dataflow of computation, synchronous data-flow, modeling of periodic schedule and consistency. Examples of the models are FSM, Petri net (PN), Kahn Process networks (KPN), Communicating Sequential Process (CSP), and Synchronous Data Flow(SDF)[8].

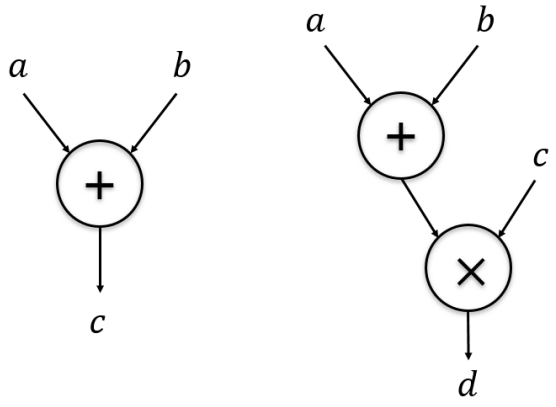

Fig.2 An example of Model-based language[8]

4.1.5 Hardware-dependent software

The hardware-dependent software is actually the firmware or driver. As time goes on, the costs of hardware-dependent software and validation have already more than the costs of implementing the hardware. Thus, it is crucial to reduce the

change of firmware results from the change of hardware IPs, and it is achieved by a layer between hardware and hardware-dependent software, called hardware abstraction layer, includes many APIs for user to simply change the API corresponding to the hardware IPs and hardware-dependent software rather than develop a whole new hardware-dependent software corresponding to the hardware with some IPs changed.

4.2 How ESL is used

I will use some experiences from the assignments and the term project to illustrate how ESL is used.

At first, we have to finish the SystemC model of a DMA. The SPEC of the DMA include the I/O port names and their width, the 4 control registers and their size, and the DMA behavior which corresponds to a simple command from the CPU to DMA, and the DMA performs data reads and writes, and sent the interrupt back to the CPU and the CPU will send the clear signal (combined with start signal) to the DMA. It is the basics for creating a model.

Then, we have to combined the DMA with TLM 2.0 as so called “wrapped DMA,” and we finished this assignment to get familiar with TLM 2.0 and how it is used, i.e. simply called some API like “b_transport(tlm::tlm_generic_payload&, sc_time&);.”

Finally, we integrate the DMA model into the virtual platform to learn the steps for system-level verification, like which processors and memories are better to choose from those models in the OVP IP library and perform a specific application. Debug with ISS is also included in the project, but since I didn’t finish the project yet, I have no experience to share about.

Part 5. Conclusions

In this final report, I illustrate the basic ideas of ESL, i.e. why use ESL, the relationship between knowledge, action, and communication in multi-core systems, and what is ESL and how to use it. Two reasons for use ESL are the increasing complexity of hardware and the shortened design cycle. The processor take actions based on the knowledge they have from synchronous/asynchronous communication and it’s explained using the “cheating husbands” stories. I also mentioned the ESL design methodology, transaction modeling, the virtual ESL platform, model-based design and hardware-dependent software to talk about what is ESL, and I explained how to use ESL based on my experience in the assignments and the term projects.

Reference

[1] Alan P. Su, “Why ESL,” *Chapter 1 Slides of Advanced Topics in Electronic System Level Design Course.*

[2] Black, David C., et al. SystemC: From the ground up. Vol. 71. Springer Science & Business Media, 2009.

[3] Moses, Yoram, Danny Dolev, and Joseph Y. Halpern. "Cheating husbands and other stories: a case study of knowledge, action, and communication." Distributed computing 1.3 (1986): 167-176.

[4] Alan P. Su, “System Design,” *Chapter 2 Slides of Advanced Topics in Electronic System Level Design Course.*

[5] Alan P. Su, “System Design,” *Chapter 3 Slides of Advanced Topics in Electronic System Level Design Course.*

[6] Wikipedia: Token. (2021, May 18). FL: Wikimedia Foundation, Inc. Retrieved June 24, 2021, from https://en.wikipedia.org/wiki/Token

[7] Alan P. Su, “System Design,” *Chapter 4 Slides of Advanced Topics in Electronic System Level Design Course.*

[8] Alan P. Su, “System Design,” *Chapter 5 Slides of Advanced Topics in Electronic System Level Design Course.*