SystemC hardware in the loop simulation scheme

Hongyun Zheng¹,*, Xianghu Wu² and Yongchao Tao³
¹Shenzhen Academy of Aerospace Technology, Shenzhen, China
²Shenzhen Academy of Aerospace Technology, Shenzhen, China
³Shenzhen Academy of Aerospace Technology, Shenzhen, China

*Corresponding author e-mail: qq158754845@outlook.com

Abstract. SystemC is the standard in the field of virtual modeling. With the continuous expansion of semiconductor industry, test verification is a very important part in the whole process of semiconductor. Our research is based on the standard SystemC platform, which is very helpful for the early planning and architecture modeling of SOC architecture, as well as for the later regression testing and upper software development testing. SystemC is a pure software solution, but with more and more customized customer needs, we need semi physical simulation. The so-called hardware in the loop simulation is the integration of software and hardware in a platform for simulation. For example, in the previous pure software simulation tests, all of them are virtual, and the simulation engine is implemented by pure software. It has nothing to do with real world time. There are also real device tests. All devices are pure hardware. If a problem is found, it may affect the overall results. The cost of building hardware again is too high. Therefore, the requirement of hardware in the loop simulation is put forward.

1. Introduction

Due to the vigorous expansion of modern semiconductor industry, the scale of semiconductor integration is becoming more and more complex and huge; for example, the latest Intel CPU, which integrates more than 1000000 semiconductor devices; with the current booming industry, the speed of semiconductor development, verification and testing is crucial to the development of semiconductor enterprises. So the basic mode of semiconductor development is to develop and simulate at the same time. Among them, SystemC is a development and test framework that can support the whole life cycle. From the logic verification of digital circuit in the early stage, to the architecture analysis, algorithm analysis in the later stage, as well as the transplantation and adaptation of upper application software such as operating system, all can be supported. So at present, SystemC has been widely used. In the current 5g technology competition, Huawei has adopted the fastmodels and semi physical simulation scheme to speed up the research and development progress. The export interface standard of fastmodels is SystemC standard.
Figure 1. Location of SystemC.

SystemC is a class library developed by C++ programming language, and provides an event driven simulation framework to schedule the execution sequence of various events. The scheduling algorithm idea of SystemC is to divide the continuous simulation time into multiple discrete simulation times, and then divide a simulation time into multiple delta cycles, update the channel value after a delta cycle or multiple delta cycles, so that the sequential execution programming language can be used in these delta cycles to simulate the parallel behavior of hardware. It enables the designer of simulation system to simulate parallel processes with C++ syntax rules. This not only increases the convenience of development, but also reduces the threshold for software engineers to develop and test hardware.

Figure 2. Basic principle of SystemC scheduler.
At present, SystemC has become the standard of simulation industry, and the products of EDA manufacturers are compatible with SystemC standard. Therefore, our goal now is to explore some problems in the field of SystemC simulation, and propose original solutions.

2. Principle analysis

SystemC is pure software. Although it can completely simulate the existing hardware behavior, the current semiconductor development has not started from scratch. Many EDA companies or R & D teams have very mature modules, which have passed many strict tests. For example, communication baseband, which is a very mature module, is a high-frequency device. People often hope that this module can cooperate with software to simulate, instead of using software to write another simulation module with the same complexity. At this time, the need for hardware in the loop simulation arises. In the same way, some modules may not be too complex in the whole system, and the design is not too complex. At this time, we can use pure software or even TLM to write the implementation. On the one hand, we can integrate mature modules, on the other hand, we will not affect the overall test and verification progress due to some modules that are not very critical. Here is an example of semi-physical simulation: the box on the left represents the real hardware, and the right represents the simulation program running on the PC. They communicate with each other in some way.

Figure 3. Hardware in the loop simulation diagram.

Figure 4. Modeling abstraction of SystemC.
In the field of hardware in the loop simulation, there is a crucial problem, that is time synchronization. What is time synchronization? Because software is simulating real hardware, there is also a concept of time in the logic of software. This time is not the time we display in the world, but the virtual logical time. If the software simulation needs to run logic for one minute, if the operation logic is very complex and your computer's performance is general, then it may take 10 minutes for the software to run. So in the simulated world, the software still runs for 1 minute, but in the real world, it takes 10 minutes. If your computer is powerful enough, the software may only run for one second, and the logic will run for one minute. Therefore, the current problem is that in different simulation environments and different computer performance, the simulation speed of software is difficult to be unified. So in the actual work of hardware software co simulation, this problem must be the first to be solved. Because of a mature peripheral module, its performance is fixed; for example, a mature baseband module hardware, its performance is fixed, and the time it takes to decode a piece of radio information is basically the same. So our problem is to ensure the synchronization of the two modules, because one is fixed time, the other is not fixed time.

![Figure 5. Comparison of real time and system C time.](image)

![Figure 6. Problems we face.](image)

3. System framework

Our common digital circuits need a clock oscillator. Our solution is to do special processing in the part of clock oscillation source, and make a clock source generator. This generator can generate the corresponding clock source according to the specified value. This clock source is the clock input source of the actual hardware in the hardware in the loop simulation. This clock source generator is actually a logic chip. On the one hand, this chip is to test the hardware Generate clock output, on the other hand, communicate with PC through USB bus, and then virtual simulation software accesses the chip through USB driver, which can send different control commands to the chip to generate different
clocks, and count the clock times, which can be used as the basis for PC synchronous clock cycle; the schematic diagram is as follows:

![System framework diagram](image)

**Figure 7.** System framework diagram.

In the aspect of SystemC software program, we need to write a clock synchronization module. The implementation of this module is roughly as follows: create a thread in SystemC, and then design a clock cycle for the thread. The logical clock cycle corresponds to the real hardware clock cycle to be synchronized, and then our core purpose is to keep the two clock cycles the same Step.

**Table 1.** Example code of clock sync module.

```cpp
class clock_sync : sc_module
{
public:
  SC_HAS_PROCESS(clock_sync);
  clock_sync(sc_module_name name)
  {
    SC_THREAD(run);
  }
  void run(void)
  {
    //no quantum, speed low
    while (0)
    {
      wait(cycle);
      //simulation software have run one cycle, then generate a hardware wave one cycle
      function_to_gengrate_hardware_clock_wave(1);
      wait_for_hardware_finish();
    }
    //have quantum, speed fast
    while (1)
    {
      wait(quantum);
      //simulation software have run quantum of cycle, then generate quantum of hardware wave cycle
      function_to_gengrate_hardware_clock_wave(quantum / cycle);
      wait_for_hardware_finish();
    }
  }
public:
  bool set_cycle(sc_time t)
  {
    cycle = t;
  }
private:
  sc_time cycle = sc_time(1, SC_US);
  sc_time quantum = sc_time(1, SC_MS);
};
```
First of all, the clock input of hardware will not be generated spontaneously, but passively generate different times of clock pulses through the instructions issued by PC host; if there is no instruction, the output of clock oscillation will remain the original level. Then every time the PC machine sends the instruction to generate the clock, the clock synchronization chip will generate the clock output to the actual hardware after receiving the instruction, and then the hardware will show its own logical behavior to achieve the purpose of simulation. Then the clock synchronization chip sends a signal back to the PC, indicating that the clock action has been completed, and the same is true for the next cycle. When the logic of the software simulation is very complex, the software may need to run longer to reach the next cycle. At this time, the synchronization chip has not yet received the instruction to generate the clock, so it will not generate the clock output. The actual hardware behavior is also equivalent to temporarily frozen, waiting for the next synchronization of the software. Such a running mode can ensure the timing synchronization between the software clock and the actual hardware, and meet the basic needs of software and hardware collaborative simulation development.

4. Conclusion

From the experimental results, this scheme is feasible and simple to deploy. When it is used on a large scale, it can be made into a special ASIC for processing. SystemC hardware in the loop simulation has a strong performance, and it will be more conducive to the hardware simulation test.

Table 2. Statistical table of experimental results.

|       | Clock deviation | Scale of time, software:hil |
|-------|-----------------|-----------------------------|
| test 1 | 1               | 1003                        |
| test 2 | 0               | 995                         |
| test 3 | 2               | 1231                        |
| test 4 | 0               | 980                         |
| test 5 | 0               | 1100                        |
| test 6 | 0               | 995                         |
| test 7 | 0               | 1006                        |
| test 8 | 1               | 1003                        |
| test 9 | 0               | 987                         |
| test 10| 0               | 1102                        |

5. Acknowledgments

This work was financially supported by Shenzhen Special Fund for Strategic Emerging Industry Development JCYJ20170816151055158.

References

[1] Chen Xi, Xu Ningyi, SystemC System-on-Chip Design [M], Science Press, 2003.
[2] Wang Jinming, Verilog HDL Programming Tutorial [M], People Post Press, 2004.
[3] Qiao Lufeng, Verilog HDL Digital System Design and Verification [M], Electronics Industry Press, 2009.
[4] Jiang Guoqiang, New digital logic circuit [M], Beijing University of Posts and Telecommunications Press, 2013.