Journal of Engineering and Applied Sciences 13 (7): 1596-1600, 2018
ISSN: 1816-949X

Profiling Based Functional Verification on a Heterogeneous Architecture

[1]S. Karthik and [2]S. Saravana Kumar
[1]Department of ECE, SRM University, Vadapalani, Chennai, India
[2]Department of CSE, Karpagam College of Engineering, Coimbatore, India

Abstract: This study research involves analyze and explore the study of the current state-of-the-art in parallel logic simulation including simultaneous simulation techniques. Functional verification, a tortuous and tedious task, refers to the verification of any logic design according to the specifications. It's basically a cumbersome process due to the presence of an enormous number of test cases even for a simple logic design. Logic simulation is used to predict the behavior of digital circuits where the system components can vary from transistor level through the gate level to the behavioral level. Due to increased complexity, large designs, the time taken to test will also be long. To overcome this problem, parallel simulation was employed. Simulation based verification due to its ease in performing is widely used. As the name suggests it allows verifying the performance of various operations simultaneously, there by reducing the time to a considerable extent. So, this method is restored to achieve our goal. To verify our design, we chose a hardware platform named Zybo board (Zynq Z-7010-heterogeneous architecture). As the name suggests, it comprises of FPGA and a 650 MHz cortex-A9 ARM processor which expedites the testing process. We use Vivado tool for profiling, a process which helps one to know which portion of the function takes more time compared to other during the simulation. The portion which takes large time for simulation is port mapped to the FPGA while the rest is assigned to the processor.

Key words: Profiling, functional verification, acceleration, heterogeneous architecture, Zybo, Zed board

INTRODUCTION

Simulation (Li *et al*., 2004; Zhu *et al*., 2005) forms an integral part of any field of engineering-software or hardware. The process for simulating the same could be tedious or simple depending upon the design to be tested. If the simulation process (Kim *et al*., 2013) alone consumes a large amount of time, then correcting the bugs if any will also take a considerable amount of time. Through this research we try to formulate a methodology which envisages reducing the testing time (Karthik *et al*., 2015) which in turn reduces the cost involved in the testing process. Design verification methodologies have been developed in a multi-flavored spectrum ranging from manual verification to formal verification techniques. Verification methodologies (Lam, 2005) can be formal or simulation based. Formal based methodologies (Lam, 2005) use comprehensive mathematical techniques to prove circuit responses to all possible inputs and all possible reachable states of the circuit conform to specification. These methods do not rely on the generation of input to verify the design. Simulation based methodologies aims at uncovering design errors by thoroughly exercising the current model of the circuit.

To begin with we need to be familiar with certain terms which stand part of our project title. Functional verification in a simple language, refers to the act of validating the design including proving whether the design has fulfilled the specifications mentioned by the customer. It is a required step in the development of today's multifaceted digital designs. Hardware complication growth continues to follow Moore's law but the verification complexity is even more challenging. In fact, the verification difficulty rises exponentially with the hardware complexity, doubling up exponentially with time (Ha *et al*., 2006). It is recognized as a bottle neck in design methodology, up to 70% of the design development time and resources are spent completely on the functional verification.

The next term which would be introduced as a part of our research is 'Profiling'. Profiling plays a pertinent role in this research. It is the profiling that provides impetus to move ahead in our implementation. Profiler gathers

Corresponding Author: S. Karthik, Department of ECE, SRM University, Vadapalani, Chennai, India

statistics by measuring an executing program on a standard CPU. There are two major ways to implement the profiling, PC sampling, the first method, uses timer to interrupt the program to be profiled periodically and record program's PC value. The second method inserts a code into program to be profiled that increments counters each time the program's execution passes certain points. A separate program can be used as the profiled program finishes displaying the statistics gathered from execution. With the help of profiling results, one can find the part of the design that extracts an exorbitant time and that which takes the normal delay time. To reduce the delay or in other ways to increase the execution, we intend to use Zybo board that comprises FPGA and an ARM processor (Karthik *et al*., 2015a, b) on the same board. By applying the part of the logic design that takes longer time to FPGA, one can the output from the same at a reduced time level. The other logic design which takes normal time can be given to the processor.

Insight to Vivado tool and Zybo board: Vivado design suite software (Xilinx, 2008) aimed to increase productivity with respect to designing integrating and implementing. It is produced by Xilinx for mainly for synthesis and performs analysis of RTL design. It replaces Xilinx ISE with additional features for SOC development and high-level synthesis. With the Vivado design suite, you can accelerate the design, methodically optimize for multiple and parallel design metrics. Vivado permits the developers compile their designs, execute timing analysis, observe RTL diagrams, simulate a design according to a number of constraints and configure the target device with the programmer. The main aim of this tool is to integrate more system functions with less number of parts there by increasing the overall system performance and lessen system power consumption. This tool has been successfully used in the verification. After the explanation of this method one can find the effective and potential usage of this method in testing.

There are various components in Vivado design suite, a high level compiler permits C, C++ and system C programs which can be directly targeted into Xilinx devices without converting or creating RTL. A compiled-language simulator that supports diverse language, TCL scripts, encrypted IP and improved verification. A IP Integratorlet's engineer to rapidly integrate and configure IP from the huge Xilinx IP library.

MATERIALS AND METHODS

Zynq architecture: The Z-7010 is based on the Xilinx all programmable System-on-Chip (AP SoC) architecture shown in Fig. 1 (Anonymous, 2014) which tightly integrate a dual-core ARM cortex-A9 processor with Xilinx 7-series Field Programmable Gate Array (FPGA) logic. When coupled with the rich set of multimedia and connectivity peripherals available on the ZYBO, the Zynq

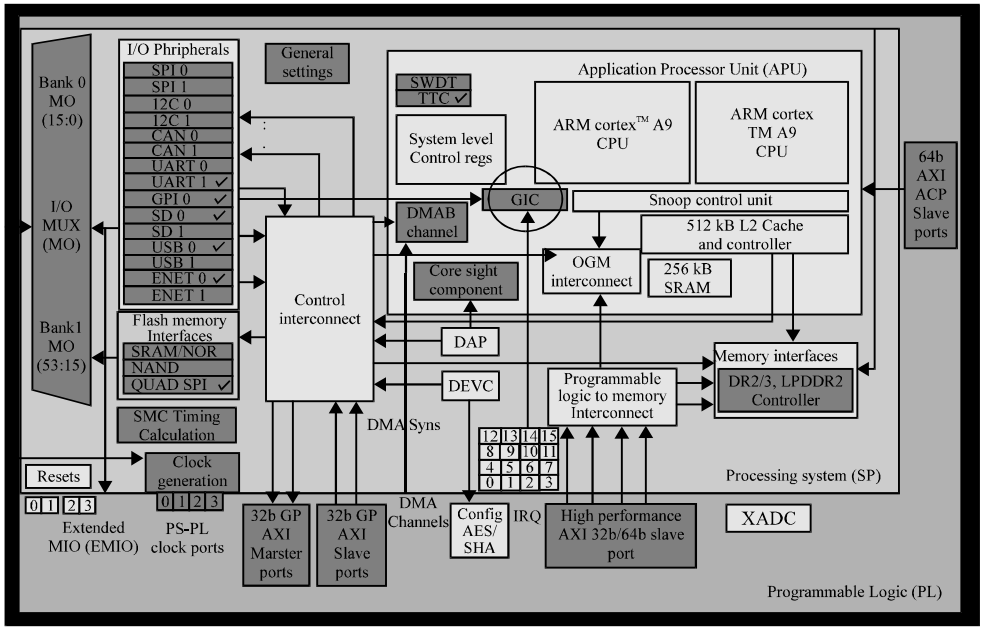


Fig. 1: Zynq architecture (Xilinx minual)

Z-7010 can host a whole system design. The on board memories, video and audio input/output, USB with dual role, ethernet and SD slot will have your design fueled up with no extra hardware needed. Additionally, six pmod connectors are accessible to put any design on an easy development path.

Profiling based functional verification: Profiling is achieved by instrumentingeither the programsource codeor its binary executable form using a tool called a profiler (or code profiler). Profilers may use a number of different techniques such as event-based, statistical instrumented and simulation methods. Program analysis tools are extremely important for understanding program behavior. Computer architects need such tools to evaluate how well programs will perform on new architectures. Software writers require tools to analyze their programs and identify critical sections of code. Compiler writers often use such tools to find out how well the instruction scheduling orbranch prediction algorithm is performing and progressing. The steps involved in profiling are shown in Fig. 2.

Suitable application project with necessary IP is created and the environment is built which is shown in Fig. 3 and then targeted on the board and after choosing the appropriate C/C++ compiler the application is run with appropriate settings. Then, we obtain the profiling results stored in gmon out in debug column which is shown in Fig. 4 and 5. With the results obtained one can decided the function that has to be hardware accelerated.

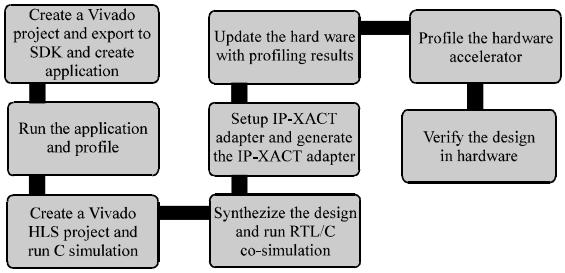


Fig. 2: Profiling flow

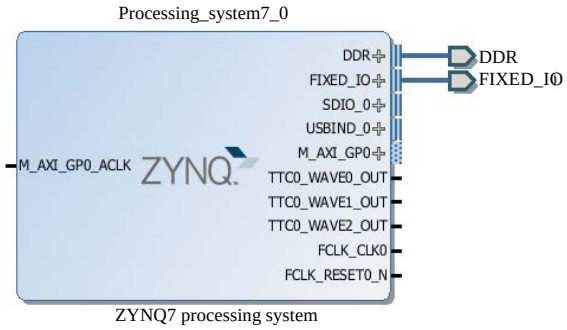
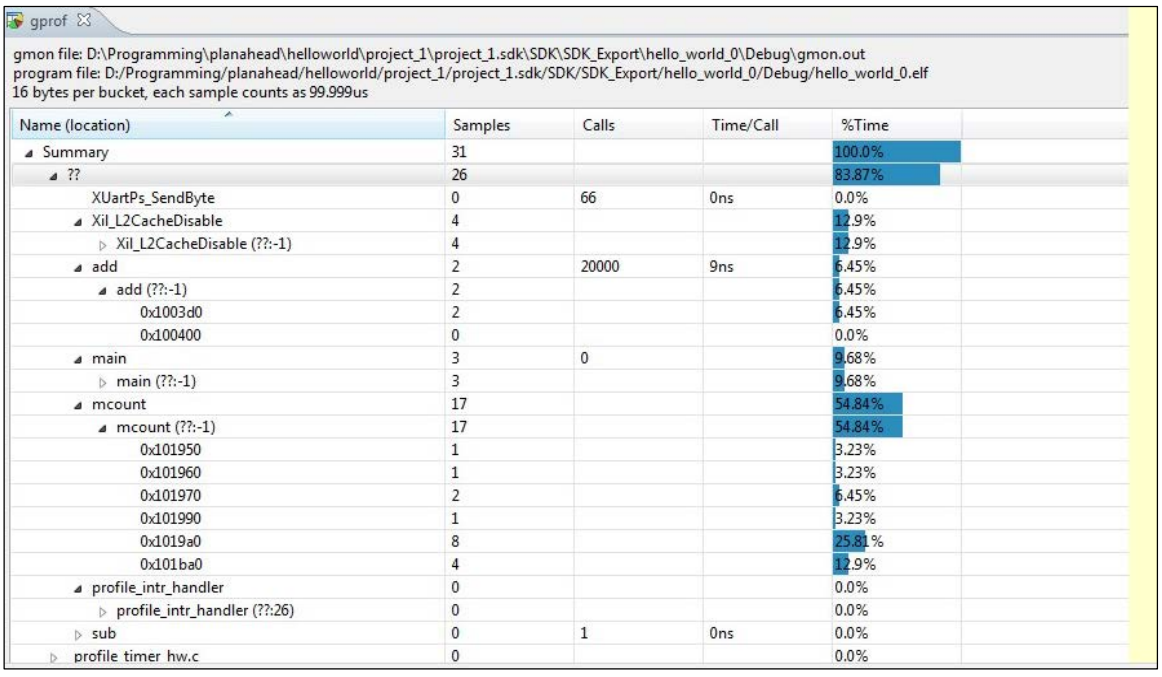


Fig. 3: Block design for connections made for the Zybo board


gprof

gmon file: D:\Programming\planahead\helloworld\project_1\project_1.sdk\SDK\SDK_Export\hello_world_0\Debug\gmon.out
program file: D:/Programming/planahead/helloworld/project_1/project_1.sdk/SDK/SDK_Export/hello_world_0/Debug/hello_world_0.elf
16 bytes per bucket, each sample counts as 99.999us

Name (location)	Samples	Calls	Time/Call	%Time
Summary	31			100.0%
??	26			83.87%
XUartPs_SendByte	0	66	0ns	0.0%
Xil_L2CacheDisable	4			12.9%
Xil_L2CacheDisable (??:-1)	4			12.9%
add	2	20000	9ns	6.45%
add (??:-1)	2			6.45%
0x1003d0	2			6.45%
0x100400	0			0.0%
main	3	0		9.68%
main (??:-1)	3			9.68%
mcount	17			54.84%
mcount (??:-1)	17			54.84%
0x101950	1			3.23%
0x101960	1			3.23%
0x101970	2			6.45%
0x101990	1			3.23%
0x1019a0	8			25.81%
0x101ba0	4			12.9%
profile_intr_handler	0			0.0%
profile_intr_handler (??:26)	0			0.0%
sub	0	1	0ns	0.0%
profile_timer_hw.c	0			0.0%

Fig. 4: Profiling results of a sample application

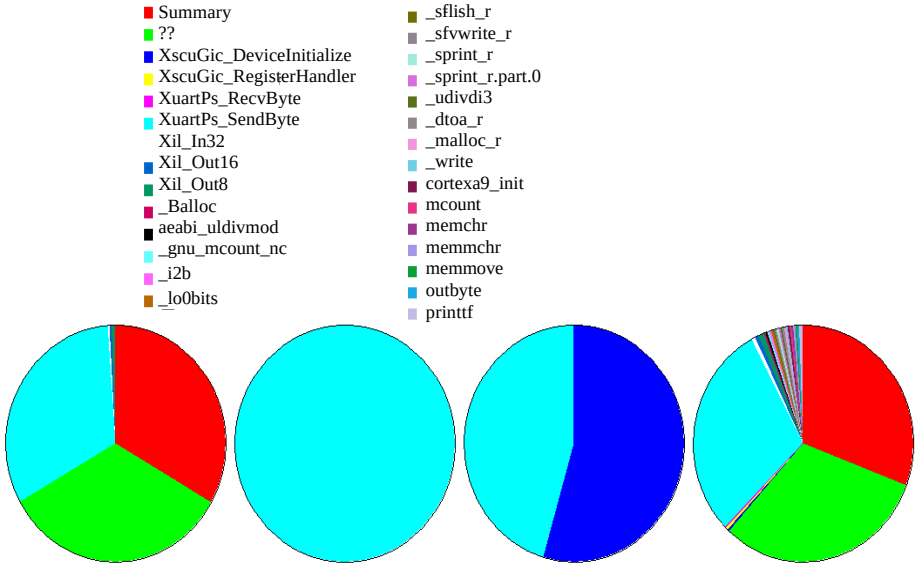


Fig. 5: Pie chart representation of time consumed by each function

Summary

Performance estimates for 'main' function

SW-only (Measured cycles)	438104516
HW accelerated (Estimated cycles)	289703822
Estimated speedup	1.51

Details

Performance estimates for functions 'madd and mmult'

SW-only (Measured cycles)	189994
HW accelerated (Estimated cycles)	45073
Estimated speedup	4.22

Resource utilization estimates for hardware accelerators

Resource	Used	Total	% Utilization
DSP	42	80	52.5
BRAM	16	60	26.67
LUT	8838	17600	50.22
FF	5353	35200	15.21

Fig. 6: Results obtained from TCL profiler of MAC design

RESULTS AND DISCUSSION

After the profiling is being done, we can see the following results are observed under gmon.out file. This gives an insight about the functions that extract a longer time. To have a deeper view of the time taken by the functions, debug option can be chosen and the application can be run in the Debug format. In the Debug format, we can see clearly the inclusive and exclusive timing, i.e., the time taken by the main function with its child functions and the time taken by the child functions alone, respectively. Table 1 shows the simulation speed up for various circuit designs and corresponding simulation time before acceleration and after hardware acceleration. The other methodology in obtaining the results is to identify the portion of the design consuming large simulation time using TCF (Target Communication Framework) profiler. Results obtained from TCF profiler is shown in Fig. 6.

Table 1: Speed up results of various designs

Circuit design name	Function identified using profiling and mapped onto FPGA as IP	Simulation time before acceleration (μsec)	Simulation time after hardware acceleration (μsec)	Speed up factor
Hamming code	Min-sendbyte ()	8.030	5.360	1.498134
MAC unit	Mull_vt ()	34.067	15.223	2.237864
FIR filter	Fir_soft ()	12.120	5.970	2.030151
AES	Aesrec ()	14.250	3.270	4.357798
ALU	Uarmes ()	44.340	22.330	1.985670

CONCLUSION

From the above results one conclude that the total time taken by each function can be analyzed following by the transfer of the timing consuming function on the hardware and accelerating the verification. Not only this, several flexibility is extended for the purpose of testing. Vivado tool with its variety of environments made the process of testing very simple and cost effective.

Another salient feature of this method is that as per the current requirements, the method utilizes the existing tools and its associated hardware. This tool is board compatible, i.e. by adding the libraries needed by the particular board, the same design can be run on any boards. But for this research, we utilized the Zybo board due to its extraordinary feature, presence of ARM A9 Cortex processor and FPGA. For our research, we applied the same methods on several filters and hamming code but the same can be extended to intricating and time consuming designs. Fulfilling our objective this method happens to be cost effective since the boards and the softwares are available at an affordable rate. Also, this method enhances the system performance by reducing the functions that run of the processor in other words the processor is relieved of some of its functions. The processor can utilize this time for performing its usual routines and interrupts, efficiency of the system is improved and the total time taken is reduced, i.e., the simulation and the testing process has been accelerated, thereby obtaining the results at a much lowered time. Mainly, through this work every aspect of the Vivado tool right from the wide range of tools available and also the various platforms too have been explored. The usage of design sources in low level or high level languages in this process is a welcoming fact of this tool. Another major advantage of this research is both the processor and the FPGA are made use of there by resulting in the utilization of the resources to the maximum.

REFERENCES

Anonymous, 2014. 9 Reasons why the Vivado design suite accelerates design productivity. Xilinx, San Jose, California, USA. https://www.xilinx.com/publications/prod_mktg/vivado/Vivado_9_Reasons_Backgrounder.pdf.

Ha, S., C. Lee, Y. Yi, S. Kwon and Y.P. Joo, 2006. Hardware-software co-design of multimedia embedded systems: The peaCE. Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, August 16-18, 2006, IEEE, Sydney, Australia, ISBN:0-7695-2676-4, pp: 207-214.

Karthik, S., S. Saravanakumar, G Murali and Gokulnath, 2015a. Parallel HDL simulation using heterogeneous FPGA architectures. Intl. J. Appl. Eng. Res., 10: 18219-18222.

Karthik, S., S.S. Kumar and K. Priyadarsini, 2015b. Comparative study of homogeneous and heterogeneous processor in FPGA for functional verification. Intl. J. Appl. Eng. Res., 10: 38358-38361.

Kim, D., M. Ciesielski and S. Yang, 2013. MULTES: Multilevel temporal-parallel event-driven simulation. IEEE. Trans. Comput. Aided Des. Integr. Circuits Syst., 32: 845-857.

Lam, W.K.C., 2005. Hardware Design Verification: Simulation and Formal Method-based Approaches. Prentice Hall, Upper Saddle River, New Jersey, USA., ISBN:9780131433472, Pages: 585.

Li, T., S. Li, F. Ao and G. Li, 2004. Parallel verilog simulation: Architecture and circuit partition. Proceedings of the 2004 International Conference on Asia and South Pacific Design Automation, January 27-30, 2004, IEEE, Piscataway, New Jersey, ISBN:0-7803-8175-0, pp: 644-646.

Xilinx, 2008. Embedded System Tools Reference Manual: Embedded Development Kit. Xilinx, San Jose, California, USA, Pages: 294.

Zhu, L., G. Chen, B.K. Szymanski, C. Tropper and T. Zhang, 2005. Parallel logic simulation of million-gate VLSI circuits. Proceedings of the 2005 13th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, September 27-29, 2005, IEEE, Atlanta, Georgia, ISBN:0-7695-2458-3, pp: 521-524.