

SOFTWARE CONTROLLED HARDWARE DESIGN VERIFICATION USING ZYNQ SoC

Karthik.S

Assistant Professor (Sr.G), Dept of ECE
SRM IST, Vadapalni
skarthikvit@gmail.com

P.Muthukrishnan

Assistant Professor , Dept of ECE
SRM IST, Vadapalni
muthu_eng05@yahoo.co.in

Abstract— Verification of today's Systems-on-Chip (SoC) occurs at low abstraction-levels, typically at register-transfer level (RTL). As the complexity of SoC designs grows, it is increasingly important to move verification to higher abstraction-levels. Hardware/software co-simulation is a step in this direction. System-level monitoring commonly used for event-based software debugging, provides information about task scheduling events, inter-task communication and synchronization, semaphores/resources and I/O interrupts. ZynQ FPGA implementation is selected because FPGA can process the information faster, controller architecture can be optimized for space or speed. Zynq has good level of productivity, it can create own ASSP (Application Specific Standard Product) and it has an ability to create truly application specific single chip solution that will enable the design to market paradigm. The Vivado tool is selected due to its robust performance, low system power and comprehensive hardware debug. We have configured user design in the Processing Logic SoC of ZynQ and the test vectors are given using Processing System. In this paper we have used hardware/software co-design approach in verifying a design. Here the test bench will be running in software side (PS) and design will run in FPGA (PL).

Index Terms— Field programmable gate array, Application-specific standard product, Processing system, Programmable logic

I. INTRODUCTION

The development of today's multidimensional digital designs is a mandatory step. The way the project team is put together is really vital. The amount of design and verification engineers are typically same in classic project team. To validate a design, one must first understand the specifications, knowledge of the design and more important, develop a different design approach from the specifications. Hence time to time the number of verification engineers will be more than the design engineers [1]. The methodologies espoused by verification engineer will not be the similar as that of the design engineer. If the verification engineer shadows the same design style as the design engineer then both would obligate the same faults and nothing would be validated. The verification difficulty is challenging. The efforts of DV (design verification) can be appreciated during the development cycle. In verification based technique, the test bench is available in the Processing System which generates

the test vectors. The test vectors is then used to simulate the design present in the FPGA. When entire development cycle is considered then the history shows that around 70% of time is spent on DV.

The simultaneous development of both hardware and Software sides of the system is called Hardware-Software co-design. To execute on a general purpose processor in FPGAs or ASICs the hardware parts are used while to translate to a low-level programming language the Software parts are used. The primary goal of hardware-software co-design here is the processing system (PS) generates the test vectors and test vectors are used to stimulate design in FPGA (Programmable Logic). The Hardware/Software (HW/SW) design flow is shown in the figure.

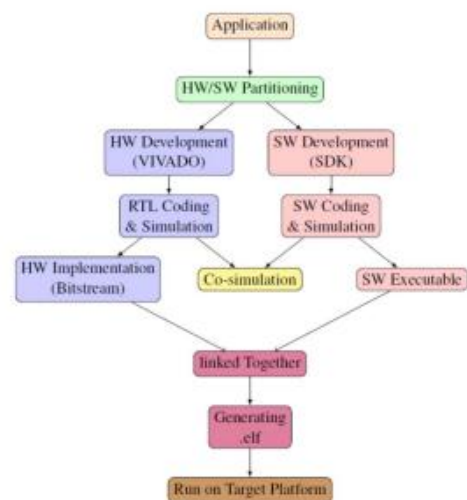


Fig.1. Hardware/Software Design flow

Field Programmable Gate Arrays (FPGA) are used for high-performance computing applications which provides configurable computing platform. It consumes significantly less energy even though it provides notable performance improvement over conventional computing platforms. FPGA has many resource usages such as Flip-Flop (FF), Look-Up

Table (LUT), Digital Signal Processor (DSP) and Block RAM (BRAM). One data bit between cycles are stored by flip-flop. N-bit bench of pre-defined rejoinders for each exclusive set of inputs is called LUT. DSPs are a block with manufactured computational units such as the adder, subtractor and multiplier. BRAM is a block of single/dual port RAM memory actual affinity to the FPGA fabric.

II. LIMITATIONS OF FORMAL VERIFICATION

An alternative to simulation based verification is formal verification and static-timing-analysis (STA). To improve their efficiency these optionally use simulation internally. Formal-Verification technique proves a design without the stimulus/test vectors. So it gives formal verification technique a vast advantage as it entirely eliminates the use of a test-bench. Formal verification can be divided into equivalence-checking and property checking or model checking. Equivalence-checking (EC). It is used when we need to find whether two designs are equivalent in their implementation. For instance, equivalence-checking is used to find whether synthesized gate level, netlist and Register Transfer Language are functionally similar. Property (or model) checking takes a design or circuits and proves or disproves a set of properties given as specs of the design. Two forms of a design/circuit are needed for Verification. Therefore, an error-bug in the specifications creates an incomplete the reference model which may escape verification. Missing specifications is a classic type of specification error. Errors in formal-verification software will omit errors in the design and will provide wrong validation.

III. LITERATURE SURVEY

Various methods are prevalent which does parallel & simultaneous working of hardware and software. The work which already exists is event-driven HDL simulation which is a prevailing method for efficient and timing simulation. Although, ancient event-driven simulation undergoes negligible act due to its intrinsically consecutive characteristics and a necessity for event synchronization. Next, a recently proposed, competing parallel simulation technique known as time parallel HDL simulation is presented and compared against the spatially distributed parallel HDL simulation [2]. The literature on parallel simulation is effluent. In the ancient parallel simulation, the design is distributed, into modules to distinct simulators. The simulation depends on physical separation of the strategy in spatial domain known as spatial parallelism. Parallel Discrete Event Simulation (PDES) simulation concept is known since late 1980s [3].

There are certain factors which affect the performance of parallel HDL system like timing granularity, design structure, target architecture, partitioning, and synchronization algorithm.

The time granularity: The design-dependent factor over which simulation has no control. Increasing timing resolution can increase the amount of processing, which in turn decreases simulation performance. In general, simulation performance varies dramatically from one design structure to another. The design structure at higher level of abstraction, for example C++, simulates faster than the design structure at the lower level like gate-level.

Hardware Architecture: The target platform or execution machine also impacts parallel simulation performance. Multitasking/Multiprocessing comes under the hardware architecture. A method in which multiple tasks or process run on a CPU. It is the responsibility of the Operating System (OS) to switch between the tasks to give an impression of multitasking [3]. Under this condition, the computer with only one core will work. Multi-tasking solves this issue by setting up which task may work at any given time and when another waiting task takes a turn. Many tasks execute simultaneously while running on multi-core system & multitasking OS.

Design Partitioning: The major difficulty in partitioning is that the simulation load of a LP is determined at run time. Hence, workload requirements are not known prior to simulation. The simulation which runs for a short time interval or even full simulation is run to profile the simulation for this the idea of pre-simulation was proposed [3]. However, it adds an extra processing step, unless it can be done as part of a complete simulation-based flow.

Time synchronization: Conservative algorithm is an asynchronous algorithm which permits different simulation times among LPs. To protect from any kind of causality it processes messages in non-decreasing order. Optimistic algorithm is also known as Time Warp algorithm. In this approach, events are immediately processed at LPs until an event with time stamp earlier (straggler event) than the local simulated time arrives. This causes LP to roll-back to a previous time so that the straggler event could be processed. The state must be saved at all LPs to allow roll-back [3]. The conservative and optimistic approaches differ in the method units of the segregated strategy interconnect during simulation to coordinate facts. With the design and partition strategy their enactment varies. The way they handle inter-simulation synchronization differ. The shared memory is a time warp used in order to decrease message passing.

Thus we aim for software control hardware verification of a design using ZynQ SoC (FPGA) with better performance and which overcome the limitation of previous system.

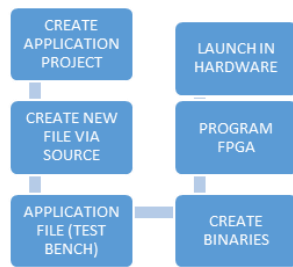


Fig.6. SDK flow

The design is implemented in the PL (Programmable Logic) section of ZynQ SoC (FPGA). The FPGA is the Programmable Logic section. The test bench is available in the Processing System Section where the test vectors are generated and which simulates the FPGA [2]. The connection between the processing system and programmable logic (FPGA) is done by AXI interconnect. The AXI Interconnect connects one or more processing system to one or more programmable logic.

VII. RESULTS

For simulation purpose various industrial bench mark design were taken and simulation was done according to our methodology. For various design the simulation noted and also compared with simulation run time of other designs.

TABLE I. SIMULATION SPEED UP FOR VARIOUS CIRCUIT DESIGNS

Circuit Design name	Simulation Time using Existing Technique in sec	Simulation Time using our methodology	Speed up factor
Hamming Code	12421	8100	1.533
ALU	8763	5234	1.674
AES	23442	14535	1.612
PCI	34452	26098	1.320

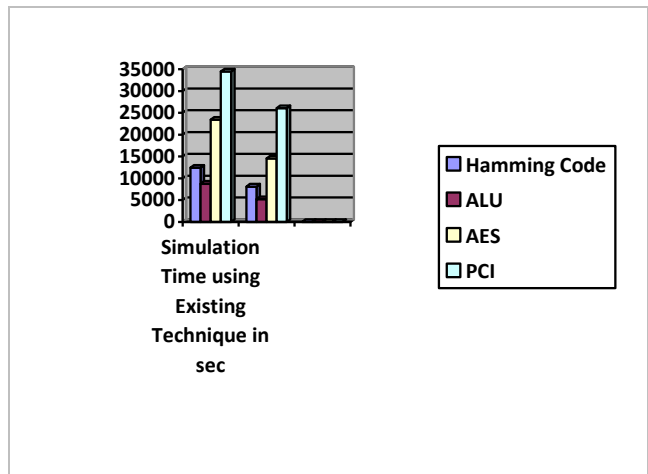


Fig.6. Graph showing the simulation speed up for various circuit designs

VIII. 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, much flexibility is extended for the purpose of Verification. Vivado tool with its variety of environments made the process of verification 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 work, we utilized the Zybo board due to its extraordinary feature, presence of ARM A9 Cortex processor and FPGA.

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 work is both the processor and the FPGA are made use of, thereby resulting in the utilization of the resources to the maximum.

REFERENCES

- [1] <http://ce.et.tudelft.nl/>

[2]Ahmad, Tariq Bashir, Namdo Kim, Byeong Min, Apurva Kalia, Maciej Ciesielski, and Seiyang Yang. "Scalable parallel event-driven HDL simulation for multi-cores" , 2012 International Conference on Synthesis Modeling Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2012

[3] www.ecs.umass.edu

[4] Hardware acceleration of an evolutionary algorithm on Xilinx Zynq-7000- Hussain Javaid Butt.

[5]Ahmad, Tariq Bashir, and Maciej Ciesielski. "An Approach to Multi-core Functional Gate-Level Simulation Minimizing Synchronization and Communication Overheads" , 2013 14th International Workshop on Microprocessor Test and Verification, 2013.

