Bringing Accuracy to Open Virtual Platforms (OVP): A Safari from High-Level Tools to Low-Level Microarchitectures

G. Shalina Percy Delicia
M.Tech Student,
Department of ECE,
Karunya University.

Thomas Bruckschloegl, Peter Figuli, Carsten Tradowsky, Gabriel Marchesan Almeida, Juergen Becker
Karlsruhe Institute of Technology – KIT, Germany.

ABSTRACT
The aggressive technology scaling in the feature size has propelled the designers to integrate millions of transistors in a single die. Thus Multi-Processor System on Chip (MPSoC) has become the irrefutable elucidation to meet the demands of parallel computing in the domain of embedded systems. The gap between software development and actual hardware model has led to the emergence of virtual platforms so that the performance status can be improved even before the Register Transfer Logic (RTL) of the hardware is actualized. This paper presents a framework to bring accuracy to Open Virtual Platforms (OVP). Several architectures are modeled using this functional simulator and they are profiled to achieve a good accuracy/speed tradeoff. The accuracy of the simulation results is further enhanced by tuning profiling parameters and introducing an empirical correction factor which compensates the imprecisions of OVP that arise e.g. from missing simulated bus- and memory access times.

Keywords
MPSoC, Virtual Platforms, OVP, Power Estimation, ARM7, ARM Cortex-M3, OR1K, and MIPS32.

1. INTRODUCTION
The exponential increase in the number of transistors that can be accumulated in a single die has led to aggressive technology scaling in the feature size and this has uncovered new stakes for high speed, low power and energy efficient models [1]. The advancing power voracious applications have already started to spark the technology industry to fleetingly develop complex and potent architectures which can make a concentrated tradeoff between performance, efficiency, and low power consumption [2]. The distinct performance demands of the autonomous applications has led to a fashionable computing platform designated as MPSoC which can accommodate multiple processors, different types of Processing Elements (PEs), Embedded Field Programmable Gate Array (eFPGA), communication architectures, and memory hierarchy on a single chip.

In the early years of the new millennium, with Central Processing Unit (CPU) clock speeds finally reaching past the 1 GHz mark, PC enthusiasts looked forward to a new world where CPU clocks kept increasing at an accelerating pace. But physics doesn't allow for exponential increase in clock rate without exponential increase in heat, especially since the core voltages are already close to the threshold voltage of the transistors and can hardly be reduced anymore. Though the fastest commercial CPUs have been hovering between 3 GHz and 4 GHz for a number of years now, there were a number of other challenges to consider, such as manufacturing technology. The saturation in clock frequency has impacted the Moore's law as shown in Figure 1.

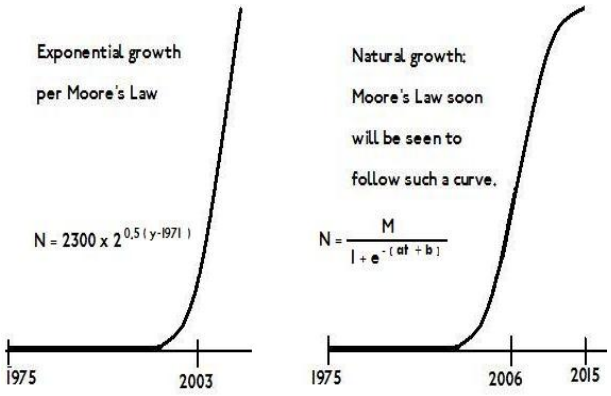


Figure 1: Exponential Vs Natural Growth

This has led to a new regime called multi-core technology which combines both multi-tasking and hyperthreading [4].

In today's mobile computers, like e.g. Tablets and Smart phones, Power Performance Area (PPA) is the major constraints in defining cost-efficient consumer products. One approach to optimize General Purpose Processors for a given task is Application Specific Instruction-set Processors (ASIP), which enables a more efficient realization towards several goals [5]. One of the motivations is that different applications have different resource requirements during their execution in terms of power and performance. Some applications may have large amount of Instruction Level Power (ILP), which can be cooped by a core which executes multiple instructions per cycle. The same core, however, might be wasted on an application with little ILP [6]. Due to huge set of variables, which are possible on a multi-core system, a Design Space Exploration (DSE) should be carried out during design time. These simulations can be executed in a very fast abstract manner or in a more accurate way that may result in very slow simulations.

2. VIRTUAL PLATFORMS
The hardware and software components of an embedded system were designed consecutively or in other words sequentially until the complexity of the MPSoC blew up. The software engineers started to develop their operating system, device drivers, and interprocessor communication protocol stacks only after getting a very solid hardware prototype for their work. The yearn for a technology which can fill in the

gap between the software development and the actual hardware model is satisfied by the emergence of virtual platforms as the software developers start their work earlier even before the RTL of the hardware is finalized [7].

OVP from Imperas encourages the developers to promote this concept of virtual platforms for SoC and multiprocessor SoC platforms. This simulation environment includes fast simulation (OVPsim) free open source models, and it is easy to use. It is instruction accurate as it enables the platform to run at hundreds of MIPS. They are hierarchical and modular as they model not only the platforms of the processors but also their bus architecture, memory model and the peripherals. They have their own library functions with four C interfaces, as shown in Figure 2.

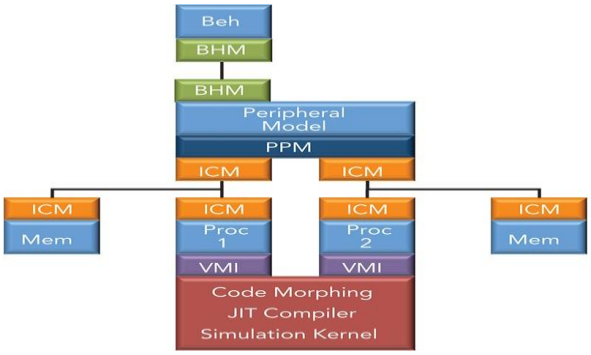


Figure 2: Interfaces in OVP [8]

The Innovative CPU Manager (ICM) C interface is used for platform modeling by tying together the system blocks such as processors, memory subsystems, peripherals, and other hardware blocks. The Virtual Machine Interface (VMI) is used for the communication between the processor models and the kernel. The Peripheral Programming Model (PPM) is for understanding buses and networks. The Behavioral Hardware Modeling (BHM) separates the address space for each model and allows communication only with those mechanisms provided by API [8]. OVP helps in the expansion of DSE as they support innumerable complex architectures in a single platform with different configurations.

2.1 Tracing

The OVP simulator executes instructions in time slices and these results in simulated instructions-per-second which is nothing but the number of instructions that are executed in one second. One of the ways of application analysis using the default instructions-per-second parameter is tracing. It works by outputting the destination address of branches. During simulation, the execution results are written onto a trace which displays the sequence of all the executed instructions optionally with embedded source code and allows backtracing of the execution flow. Instruction tracing is slower than the normal execution as the debugger runs the current thread step by step to retrieve all the required register values. This is not more efficient as the trace has to be further processed to get the instruction count for each individual instruction. It is time consuming and also has to deal with unmanageable information when used for larger applications. The instruction tracing for the MicroBlaze processor when running Bubble Sort Benchmark application is shown in the Figure 3.

```
Info 16711739: 'cpu0', 0x0000000000000ed54: 230
Info 16711740: 'cpu0', 0x000000000000007c: 91
Info 16711741: 'cpu0', 0x0000000000000084: 87
Info 16711742: 'cpu0', 0x0000000000000088: 105
Info 16711743: 'cpu0', 0x000000000000008c: 102
Info 16711744: 'cpu0', 0x00000000000000a8: 93
Info 16711745: 'cpu0', 0x00000000000000b0: 93
Info 16711746: 'cpu0', 0x00000000000000b4: 115
Info 16711747: 'cpu0', 0x00000000000000b8: 87
Info 16711748: 'cpu0', 0x00000000000000bc: 87
Info 16711749: 'cpu0', 0x00000000000000c4: 104
Info 16711750: 'cpu0', 0x00000000000000d8: 87
Info 16711751: 'cpu0', 0x00000000000000dc: 100
Info 16711752: 'cpu0', 0x00000000000000e4: 93
Info 16711753: 'cpu0', 0x00000000000000e8: 132
Info 16711754: 'cpu0', 0x00000000000000ec: 87
Info 16711755: 'cpu0', 0x0000000000000ed58: 62
Info 16711756: 'cpu0', 0x0000000000000ed5c: 132
Info 16711757: 'cpu0', 0x0000000000000ed60: 87
Info 16711758: 'cpu0', 0x00000000000001e0: 121
Info 16711759: 'cpu0', 0x00000000000001e4: 230
Info 16711760: 'cpu0', 0x00000000000006b8: 132
Info 16711761: 'cpu0', 0x00000000000006bc: 230
Info 16711762: 'cpu0', 0x00000000000001e8: 62
Info 16711763: 'cpu0', 0x00000000000001ec: 87
Info 16711764: 'cpu0', 0x00000000000001f0: 132
Info 16711765: 'cpu0', 0x00000000000001f4: 85
Info 16711766: 'cpu0', 0x0000000000000070: 121
Info 16711767: 'cpu0', 0x0000000000000074: 87
Info 16711768: 'cpu0', 0x00000000000006c8: *** INTERCEPT *** (exit)
Info
Info ------------------------------------------------
Info CPU '/cpu0' STATISTICS
Info   Type                 : microblaze (V8_20)
Info   Nominal MIPS         : 100
Info   Final program counter : 0x78
Info   Simulated instructions: 16,711,768
Info   Simulated MIPS       : 0.0
Info ------------------------------------------------
Info
Info ------------------------------------------------
Info SIMULATION TIME STATISTICS
Info   Simulated time       : 0.17 seconds
Info   User time            : 34.84 seconds
Info   System time          : 42.22 seconds
Info   Elapsed time         : 1459.34 seconds
Info ------------------------------------------------
```

Figure 3: Tracing for Bubble Sort in MicroBlaze

From Figure 3, it can be seen that the trace line for each executed instruction gives the instruction address and the instruction id.

2.2 Profiling

Once the requirements for modeling a processor are completed, various benchmarks like Bubble sort, Merge sort, and Heap sort, Dhrystone, Fibonacci, Linkpack, Peakspeed are run on top of them to confirm the testing of the modeled processor. Profiling is done at this stage to measure the number of times each instruction is executed in a particular benchmark for different processors. Profiling in simple words is a dynamic program analysis which computes or estimates some unknown qualities like complexity of program or usage of particular instructions from the known qualities. Instruction level profiling helps in evaluating the behavior of the complex CPUs, as they run faster than simulation in several orders of magnitude [9]. The processor model can be profiled by instrumenting the program source code. Tracing has engendered the simulator to find out the hot spots from the information obtained about the executed instructions and their register values whereas profiling makes it user readable.

2.2.1 Sampling

The default scheduling algorithm executes 100,000 instructions for a nominal MIPS rate of 100 with a time slice of 1 ms. For instruction level profiling in order to get the information about each instruction, scheduling algorithm has to be modified. Since 100,000 instructions are executed in a time slice, profiling them will give information only about the last instruction leaving the rest of the 99,999 instructions. Though it gives fast simulation, it leads to poor performance as much information cannot be extracted from the profiling of the processor. For this reason, the instructions are sampled by defining the number of instructions that has to be executed in

a time slice. The size of the time slice is chosen in such a way that only one instruction will be executed per time slice having a MIPS rate of 100. As a result, profiling the processor with the modified time slice will give insight about the instruction set of that processor.

2.2.2 Stepping

Stepping is the dynamic way of profiling. Stepping in processors can be achieved through time slicing which is a short interval of time allotted to each user or program in a multitasking or timesharing system. By default the time slices are in milliseconds. Time slice also known as time quantum is a period of time for which a process is allowed to run uninterrupted in a system. Simulation time is broken into time slices. For Instruction-Level Profiling as shown in Figure 4, the time slice has been set to 100 us so that only one instruction will be executed by the processor per time slice. The simulator selects the first processor and simulates it for one time slice. It calculates the number of instructions that has to be executed by the processor in a time slice and then simulates for that number of instructions. When the first processor simulates the required number of instructions, it is then suspended and the next processor is simulated for that time slice. When all the processors have simulated the time slice, the simulated time is moved on and the next time slice starts. Stepping makes the model instruction accurate.

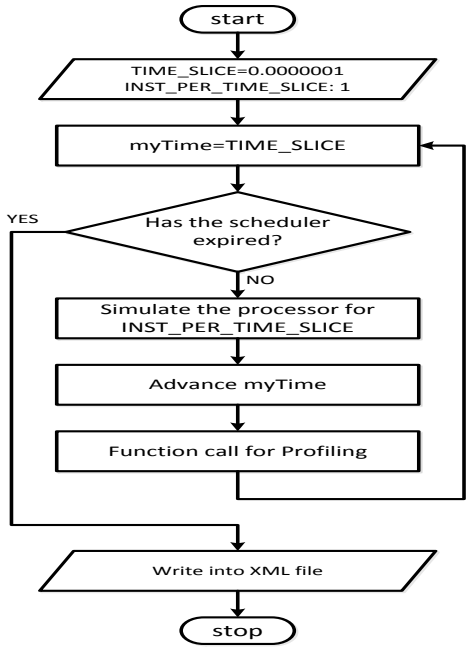


Figure 4: Instruction-Level Profiling

2.2.3 Instrumentation

The information derived from profiling like number of times each instruction is executed, number of cycles per instruction, and the base cost of each instruction are then written to an XML file which is nothing but a textual data format which finds its way in many APIs. This modification in the scheduling algorithm will give realistic simulation results without degrading the performance of the simulator.

2.3 Custom Functions

Endianness is an attribute of the system that deals with the multi-byte numbers and the order they should be stored, either as most significant first (Big Endian) or least significant first (Little Endian). To be precise it usually refers to the byte orderings in memory and not to the individual bit orderings contained within the bytes. Endianness makes sense only when breaking a multi byte data and trying to store the bytes at consecutive memory locations. It is not significant in bitwise or bitshift operations but it matters when using a type cast operation.

3. PROPOSED METHODOLOGY

The SoC design is a fusion of complex hardware and software components [10]. Prototype hardware containing multi-core processors either homogeneous or heterogeneous is usually handed over to software developers too late in design cycle causing project delays or in worst case missing its market window. For this reason, software developers work with virtual models of SoC hardware which are always have to compromise between speed and accuracy. The block diagram of the proposed methodology is shown in the Figure 5.

A hardware prototype, which can be either a processor board or an FPGA board with soft- or embedded hard processor core, is used for executing benchmarks. The electrical current profile during the execution is recorded with a digital oscilloscope over a shunt current monitor and exported as CSV (comma separated value) file. Then this data is imported to a numerical computing environment and the energy and power profiles are derived as well as their effective values per benchmark and per instruction. The effective power and energy per instruction are fed back to OVP. With this information which is obtained for every instruction type, the power and energy for executing any application can be estimated in simulation by applying these values to tracing and profiling. These steps are described more into detailed in the following sub sections.

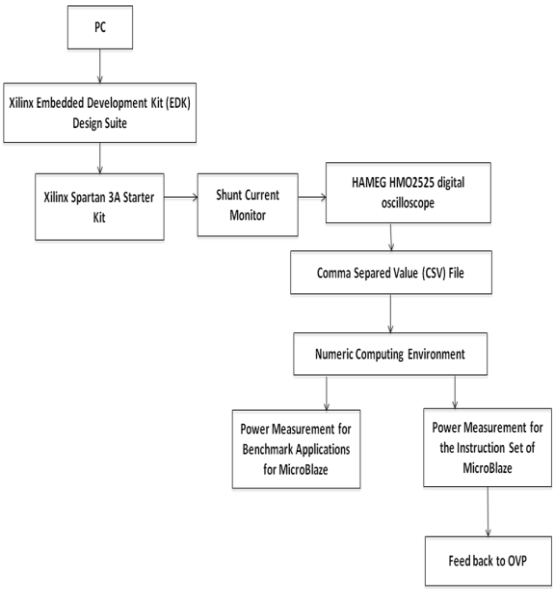


Figure 5: Proposed Methodology

3.1 Hardware Prototypes

The power consumption for different benchmark applications of MicroBlaze is done measured on the Xilinx Spartan-3A Starter Kit. This board was chosen since it has easy access to the core supply voltage by jumpers. The MicroBlaze processor along with its instruction profiling is further made more instruction accurate by profiling the power consumption of each instruction, so that the total cost of each application can be evaluated.

3.1.1 Xilinx Spartan-3A Starter Kit

The Xilinx Spartan-3A Starter Kit uses Xilinx 700K-gate XC3S700A Spartan-3A non-volatile FPGA with a soft core MicroBlaze. It gives instantaneous access to SUSPEND power-saving mode, high-speed I/O options, Double Data Rate synchronous Dynamic Random-access Memory (DDR2 SRAM) interface, and flash support. Rapid data transfer is achieved when connected to DDR2 memory through the memory interface controller. It has a 50 MHz clock oscillator. It communicates through Ethernet and JTAG USB port and supports Xilinx Embedded Development Kit (EDK) Design Suite to flash the benchmark applications to the FPGA [11]. It has a core voltage of 1.5 V and has an external 4 Mbit Flash PROM where the applications are flashed.

3.2 Power Profile

The modern embedded systems are more sophisticated and complex that power consumption has become one of the most demanding critical design constraints. There are two ways to pilot the power consumption estimation, either at the cycle or instruction level [12].

Cycle accurate methodology implemented by tools such as SimplePower and Wattch are time consuming and also require complete information about the microarchitecture which is not feasible for ready-made processors.

The Instruction Level Power (ILP) analysis model proposed in [13] is based on the hypothesis that if a given instruction or an instruction sequence is executed repeatedly, then the power cost of that instruction will be equal to the power cost of the processor. Each instruction in the instruction set of the processor is assigned a fixed power cost called base power cost which varies for each instruction depending on their operands, register used, immediate values and memory address. More the number of 1's in the immediate value, lesser the cost. The position of 1's in the binary representation of the address also affects the base cost, so an average base cost value is used. Figure 6 below, shows the current measured for merge sort application in Xilinx Platform for various iterations.

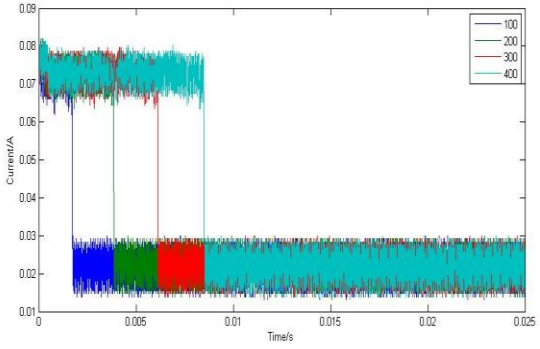


Figure 6: Current Measurements for Merge Sort in Xilinx Platform for Different Iterations

The assembly code for each instruction is made to run in infinite loops in the Xilinx custom board and the average current drawn by the processor core during the execution of this loop is measured by HAMEG HMO2524 oscilloscope. There is instability in the current measurement because of the varying loads. The measuring environment is made accurate and freed from noise disturbances by using an integrated high-performance shunt current monitor.

The Comma Separated Values (CSV) file obtained from the oscilloscope is then processed in MATLAB numerical computing environment to extract the effective power of each instruction and energy cost of different applications. This information captured from the real prototype is then fed back to the profiled MicroBlaze processor in OVP to calibrate it. The summation of the calculated power multiplied by the executed count value of each instruction will give the total cost of the application. Similarly the power consumption of different applications for ARM Cortex-M3 is measured from the Actel SmartFusion Evaluation Board.

3.3 Power Feed in OVP

The MicroBlaze processor model is already profiled to determine the number of times each instruction is executed. Profiling it further with the information derived from the Xilinx platform leads to the conclusion of understanding which instruction is executed the most and the total power it draws in each application. The power measured for different benchmark applications of MicroBlaze form Xilinx Platfom is then compared against the OVP's results to arrive at the relative error percentage. In the first iteration, the evaluated relative error percentage was in the range of 50 %, which might have the following reasons:

(1) When executing real-time applications in OVP, since it is instruction accurate, it will not consider the hit and miss of the caches, the branching effect and the inter-instruction effects which usually happens in a real hardware prototype.

(2) The speed of the processors varies with respect to the inputs and the register values.

(3) Due to the simulation in various systems with different configurations.

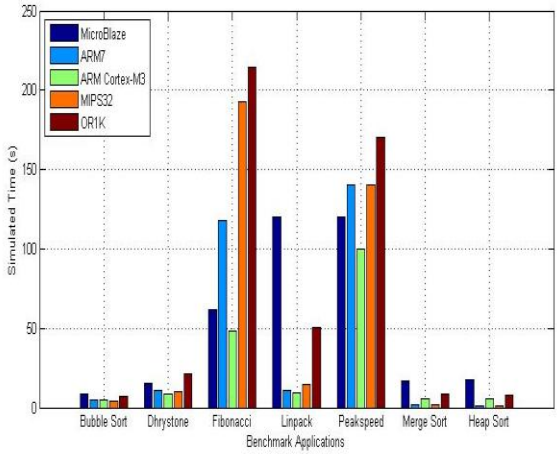
In order to improve the situation and deal with the effects which are not natively covered by the OVP models, a correction factor could be determined empirically over a number of benchmarks. This improves the relative error remarkably.

4. EXPERIMENTAL RESULTS

When describing the characteristics of processors, architecture must be distinguished from the hardware implementation of that architecture. Architecture refers to the instruction set, registers, the exception model, memory management, virtual and physical address layout, and other features that all hardware executes. These system level functional descriptions from the architectures are used to model the processors in virtual platforms. Four different processors MIPS32 [14], OR1K [15], ARM7TDMI and ARM Cortex-M3 [16] are modeled in OVP with Instruction tracing, Endianness features and are profiled based on their corresponding instruction set.

4.1 Measured Power and Energy

The power consumed by MicroBlaze modeled processor for different benchmarks is assessed with the help of Spartan-3A Starter Kit which has a soft core MicroBlaze FPGA. The simulated time in seconds and the simulated instructions for the modeled processors when simulated by OVPsim for different benchmark applications is shown in Figure 7 and Figure 8 respectively.



The simulated time in seconds and the simulated instructions for the modeled processors when simulated by OVPsim for different benchmark applications is shown in Figure 7 and Figure 8 respectively.

Figure 7: Comparison of Simulated Time of Different Applications for the Modeled Processors

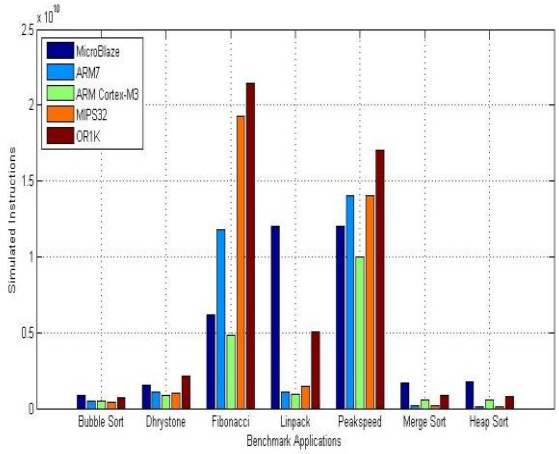


Figure 8: Comparison of Simulated Instructions of Different Applications for the Modeled Processors

It can be seen that MiroBlaze Processor executes the highest number of instructions when compared to ARM Cortex-M3, ARM7, and OR1K in all the sorting algorithms and Linpack. OR1K tops in Fibonacci, Dhrystone, and Peakspeed applications. MIPS32 has the lowest simulation time for all the sorting algorithms whereas ARM Cortex-M3 for Fibonacci, Dhrystone, Linpack and Peakspeed. This explains the reason why an application consumes different amount of power when run in different cores. It depends on the ILP of each application and the instruction set of the processors as some processors support Single Instruction Multiple Data (SIMD), and some MIMD. The Effective power measurement for different benchmark applications for MicroBlaze from the Xilinx Spartan-3A Starter is tabulated in the Table 1.

Table 1: Energy Estimation for Different Benchmark Applications of MicroBlaze from Xilinx Platform

MicroBlaze	Time (ms)	Power (mWatt)	Energy (mJoule
Peakspeed	1.4	71.4	0.099
Bubble Sort	530.8	87.8	46.7
Heap Sort	16	86.5	14
Merge Sort	16	82.1	0.131
Dhrystone	1041.6	87.7	91.4

4.2 Estimated power and Energy

The power estimated for each instruction in the instruction set of MicroBlaze processor with the help of Spartan-3A starter kit is then fed back to the modeled MicroBlaze processor in OVP. Since the processor is already instruction profiled, adding the base cost of each instruction to it will results in the total power cost of all the instructions which are executed in that particular benchmark application. A comparison between the estimated power from OVP and the measured power are shown in the Table 3. The Relative Error% of the MicroBlaze processor for the power consumption of different benchmark applications is calculated. Since the Relative Error% is almost the same for all the benchmark applications, an empirical correction factor (K) is introduced.

$$K = 1/[\text{sum}(\text{simulatedPower}(n)/\text{realPower}(n)]/N \quad (1)$$

Where 'n' denotes different benchmark applications and N is the number of benchmarks. With this formula the correction factor K=2.28 was obtain. The inclusion of this correction factor reduces the Relative Error% by 7.422%. It is to be noted that the correction factor should be calculated for every processor model separately.

4.3 Accuracy

The following Table 4 shows the relative error percentage of Fibonacci application when different time slices of 100 us, 100 ms and 1 ms (Default) are used and how accuracy is achieved in OVP.

Table 2: Bringing Accuracy to OVP

Time slice	100 us	100 ms	1 ms (Default)
Instructions	407,426	408	5
Cycles	602,303	617	99
Power (mWattt)	37.671	37.833	35.136
Relative error %	57.72	57.53	60.56
Accuracy		0.18	-2.845

When the default time of 1 ms is used it increases the error by 2.845 percent thereby greatly affecting the accuracy as it executes only 5 instructions whereas when a time slice of 100 ms is used, it gives 0.18% improvement in the accuracy but still the number of instructions executed is relatively poor. Therefore bigger the time slice, the accuracy gets marred whereas the performance (simulation speed) gets improved. By setting the time slice to 100 us, a tradeoff is made between accuracy and performance.

Table 3: Comparison between OVP's MicroBlaze Estimated Power and Xilinx's Measured Power

Application	Bubble	Merge	Heap	Dhrystone	Peakspeed	Fibonacci
Instructions	6,859,786	39,556	658,007	3,750,450	12,143	3,405
Cycle	9,616,514	50,221	746,970	5,470,698	13,206	5,045
Power (mWatt)-OVP	37.19	34.75	33.34	36.89	32.64	37.81
Power (mWatt)-H/W	87.9	82.1	85.6	87.6	71.4	73
Relative Error %	57.69	57.66	61.051	57.88	54.27	48.19
Corrected Power (mWatt) - OVP	84.79	79.23	76.02	84.11	74.42	86.21
Relative Error% after correction	3.534	3.496	11.197	3.984	-4.229	-18.092

5. CONCLUSION

In this paper, a conscientious DSE for MPSoC having the necessity for rapid and accurate tools to compute performance and power consumption is exploited as the targeted platform for modeling processors are validated by real applications. A modified scheduling algorithm is proposed to elicit information about each instruction which relinquishes prudent simulation results without performance degradation. The power devoured by distinct benchmark applications and each instruction in the instruction set explored by the real hardware prototypes when fed back to OVP makes it an early power consumption estimator with low simulation overhead. From the investigation between the measured and estimated power, the relative error is high due to effects like memory and bus access times which are not considered by OVP. But since for all the benchmarks, the relative error is almost the same, it can be improved remarkably by a correction factor and this will bring a high accuracy per simulation time as it reduces the Relative Error % to 7.422% average.

6. REFERENCES

[1] O.S. Unsal., J.W. Tschanz., K. Bowman and et al. (2006) "Impact of parameter variations on circuits and microarchitecture" Micro, IEEE, 26(6), 30–39.

[2] Gabriel Marchesan Almeida. (March 14, 2012) "Adaptive multiprocessor systems-on-chip architectures: Principles, methods and tools", Lap Lambert Academic Publishing.

[3] Fabrice Lemonnier, Philippe Millet, Gabriel Marchesan Almeida and et al. (2012) "Towards future adaptive multiprocessor systems-on-chip: an innovative approach for flexible architectures" International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XII).

[4] ARM7TDMI technical reference manual. January 2008.

[5] Jürgen Teich., Jörg Henkel., Andreas Herkersdorf and et.al. (2011) "Invasive computing: An Overview", Springer New York, 241–268.

[6] R. Kumar., K.I. Farkas., N.P. Jouppi and et al. (December, 2003) "Single-isa heterogeneous multi-core architectures: the potential for processor power reduction" 81–92.

[7] B. Bailey and G. Martin. (2009) "Esl models and their application: Electronic system level design and verification in practice embedded systems", Springer.

[8] Available from: [Online].Available:http://www.ovpworld.org.

[9] Chriss Stephens., Bryce Cogswell., John Heinlein and et.al. (May, 1991) "Instruction level profiling and evaluation of the IBM/6000" SIGARCH Comput. Archit. News, 19(3), 180-189.

[10] Rabie Ben Atitallah., Smail Niar., Alain Greiner and et al. (2006) "Estimating energy consumption for an MPSoC architectural exploration", Proceedings of the 19th international on Architecture of Computing Systems, Springer-Verlag, 298–310.

[11] Modelsim â advanced verificationand debugging. Xilinx Tutorial, September, 2004.

[12] N. Julien., J. Laurent., E. Senn and et.al. (2003) "Power consumption modeling and characterization of the TI C6201", Micro, IEEE, 23(5), 40–49.

[13] V. Tiwari., S. Malik., and A. Wolfe. (December, 1994) "Power analysis of embedded software: a first step towards software power minimization", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2(4), 437–445.

[14] Available from: [Online].Available:http://http://www.mips.com.

[15] Available from: [Online].Available:http://www.opencores.com.

[16] "ARM7DI Data Sheet". Document number arm ddi0027d. Issued: December 1994.