International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 5 262 – 264

Verification Strategy for 8-bit Processor

P.V.Bhandarkar[1]
[1]Research scholar
Shri. Ramdeobaba COE & Management,
RTMN University, Nagpur, Maharashtra,
INDIA
Puja.khangar04@gmail.com

Dr.S.S.Limaye[2]
[2]HOD
Oberoi Centre of Excellence,
RTMN University, Nagpur, Maharashtra,
INDIA
Shyam_limaye@hotmail.com

***Abstract*—**Verification of processors is one the bottle neck in designing of processor's lifecycle it consumes about 70-80 % of design's cycle. At present verification has become the wide area of research for reducing the verification time of processors and developing various techniques for verifying the processor. Much paper has been published on how to reduce the verification time and how efficiently we can verify any processor. This paper focuses on verification of processor with higher abstraction level using LISA as a reference model. The proposed methodology verifies any processor independent of technology used.

Keywords: *LISA, Architecture exploration, machine description languages, Microprocessor 8085.*

*****

I. INTRODUCTION

As per the Moore's law the number of transistor on board doubles every 18 months the complexity of the circuit has increased tremendously with every reducing size of the circuits. Hence, verification has become bottle neck in circuit's lifecycle. Major issues involved are the hardware-software co-verification, timing issues as clocking of each module differs, power dissipation some are the main issues which we can't neglect while deigning & verifying the circuits or processors. Testing and verification are important aspects of today's successful operations of circuits & processors. Our aim is to design the powerful test bench based on higher abstraction level which will hide the processor architecture as well as the infrastructure. At present we decided to do reverse engineering as LISA compiler is not freeware. We decided to prepare the LISA model of Microprocessor 8085 which we have taken as case study and prepare the VHDL test bench of 8085 from which we will pick the instructions randomly and compared it with LISA model which will act as reference model it should be same .

II. LITERATURE SURVEY

Verification is a process used to demonstrate the functional correctness of a design also called logic verification or simulation. Verification consumes about 70-80% of design cycles. To reduce verification time following methods we adopt.

a.**Parallelism**: Digging a hole in the ground can be parallelized by providing more workers armed with shovels. In verification, it is necessary to be able to write and debug testbenches in parallel with each other as well as in parallel with the design implementation. But remember, there is a limit on the number of resources to apply

b.**Abstraction:** Instead of adding additional resources with shovels, invest in a backhoe to dig the hole.

Caution: Using a backhoe to dig a hole suffers from some loss-of-control. Worker no longer directly interacts with the dirt. Digging happens much faster, but with less precision. Also consider a novice vs.expert operating the backhoe.

c.Automation: Holes must be dug in a variety of shapes, sizes, depths, locations, and in a variety of soil types.

Verification faces similar challenges. Variety of functions, interfaces, protocols, and transformations must be verified, so it is not possible to provide general purpose automation. But there are areas that can be automated. For example, there exists trench digging machines to lay cable.

Purpose of verification is to ensure that the result of some transformation is as intended or as expected.Following types of verification all have different origin and reconvergence points:

- Formal Verification
- Model Checking
- Functional Verification
- Testbench Generators.

Formal Verification: once the end points of formal verification reconvergence paths are understood, then you know exactly what is being verified.Types of Formal: Equivalence checking and Model Checking. In Equivalence checking

It compares two models to see if they are equivalent. It proves mathematically that the origin and output are logically equivalent.In formal verification characteristics of a design are formally proven or disproved. In functional verification

Verifies design intent. Without one must trust that the transformation of a specification to RTL was performed correctly. Prove presence of bugs, but cannot prove their absence.Test bench Generators is a tool to generate stimulus to exercise code or expose bugs.Designer input is still required in fact RTL code is the origin and there is no reconvergence point Verification engineer is left to determine if the test bench applies valid stimulus

Functional Verification Approaches: Black-Box Approach, White-Box Approach, Grey-Box Approach.

The black box has inputs, outputs, and performs some function. The function may be well documented or not. To verify a black box, you need to understand the function and be able to

predict the outputs based on the inputs.The black box can be a full system, a chip, a unit of a chip, or a single macro. White box verification means that the internal facilities are visible and utilized by the test bench stimulus.Eg. Unit/Module level verification.Grey box verification means that a limited number of facilities are utilized in a mostly black-box environment.Eg: Most environments. Prediction of correct results on the interface is occasionally impossible without viewing an internal signal.In our paper we are concentrating on Functional verification we are using systemverilog language for including the verification constructs.

III. SIMILAR WORK

Many papers has been published on LISA as well as on verification In [3] author has presented the machine description language LISA for generation of bit-and cycle-accurate models of DSP processors. Based on a behavioural operation description, the architectural details and pipeline operations of modern DSP processors can be covered. Beyond the behavioural model, LISA descriptions include other architecture-related information like the instruction set. The information provided by LISA models enables automatic generation of simulators and assemblers which are essential elements of DSP software development environments. In order to proof the applicability of their approach, a realized model of the Texas Instruments TMS320C6201 DSP is presented and derived LISA code examples are given in these paper.In [4] author has presented retarget able framework for ASIP design which is based on machine descriptions in the LISA language. From that, software development tools can be generated automatically including high-level language C compiler, assembler, linker, simulator, and debugger frontend. Moreover, for architecture implementation, synthesizable hardware description language code can be derived, which can then be processed by standard synthesis tools. Implementation results for a low-power ASIP for digital video broadcasting terrestrial acquisition and tracking algorithms designed with the presented methodology will be given. To show the quality of the generated software development tools, they are compared in speed and functionality with commercially available tools of state-of-the-art digital signal processor and C architectures.

IV. LISA LANGUAGE

The language LISA is aimed to formalized description of programmable architectures, their peripherals and interfaces. It was developed to close the gap between purely structural oriented languages (VHDL, Verilog) and instruction set languages for architecture exploration purposes as well as for verification. The language syntax provides a high flexibility to describe the instruction set of various processors, such as SIMD, MIMD and VLIW-type architectures and mixed styles also. Moreover, processors with complex pipelines can be easily modelled and designed.
The LISA machine has following model:

1)The **memory model** lists the registers and memories of the system with their respective bit widths, ranges, and aliasing all together.

2) The **resource model** describes the available hardware resources, eg. registers or functional units and the resource requirements of operations. Resources reproduce properties of hardware structures which can be accessed only by a given number of operations at a time.

3) The **instruction set** model identifies valid combinations of hardware operations and admissible operandsfor the instructions. It is expressed by the assembly syntax, instruction word coding, and the specification of legal operands and addressing modes for each instruction.

4) The **behavioural model** abstracts the activities of hardware structures to operations changing the state of the processor for simulation purposes. The abstraction level of this model can range widely between the hardware implementation level and the level of high level language (HLL) statements eg. VHDL

5) The **timing model** specifies the activation sequence of hardware operations and units all together.

6) The **micro-architecture model** allows grouping of hardware operations to functional units and contains the exact micro-architecture implementation of structural components such as adders, multipliers, delay elements etc. These various model components are sufficient for generating software tools as well as a HDL representation each with their particular requirements from its LISA description in detail. Furthermore, LISA models may cover a wide range of abstraction levels hiding all infrastructure from the user. This comprises all levels starting at a pure functional sight, modeling the data path of the architecture, to register transfer level (RTL) accurate models to gate level netlist. Besides a proper description of the structure, RTL models include detailed information about the micro-architecture model. Therefore, these models can be used to generate a HDL representation of the architecture, using the languages VHDL,Verilog or SystemC, SystemVerilog. Certainly a working set of software tools can be generated from all levels of abstraction. LISA can be used to model any processor that is defined by an instruction set, such as an SRC or a DSP processor or any other processor. The LISA tool suite can also be used to develop new application specific processors, study the effect of different computer architectures on an instruction set, as well as develop replacements for legacy processors which exist at present. The LISA language allows for the easy representation of pipelined (cycle-accurate) and VLIW processors at higher level of abstractions.

V. MICROPROCESSOR 8085

To begin with we have taken 8-bit Microprocessor 8085 which has 8 bit data bus & 16 bit address bus which is time multiplexed for reducing the no. Of pins and hence the hardwareof the circuit.There are toatal 6 general purpose registers of 8 bit B,C,D,E,H,Lwhich can be combined as pair to use as 16 bit registers.B-C,D-E,H-L,also there 2 ,16 bit registers Stack pointer & program counter SP,PC. There are two temporary registers in up 8085 they are W & Z registers they are used to store intermediate results. Not accessible to users. There is one 8-bit register used for arithmetic & logical operation result storage it is Accumulator ' A'. We are mainly concern with registers set & instructions of processor. 8085 offers 246 op-codes for its 80 instructions which is devided into five major groups

1) Data transfer(10 instructions)
2) Arithmetic(17 instructions).
3) Logical(12 instructions)
4) Programme Branching (29 instructions)
5) Machine control (12 instructions)

VI. METHODOLOGY

To begin with the verification we have designed the 8085 processor in VHDL and written the VHDL Testbench in which we have taken the CPU 8085 as one component & RAM as other component we have port mapped both of them feed the RAM with some instructions which we are using as programme block memory. This instructions we are feeding manually next step is to design the systemverilog testbench to automate the random selection of instructions together.

VII. SIMULATION RESULTS

Following fig.1 is the simulation with ModelSim 10.0b in which we have portmap CPU8085 with RAM of 256 kb memory and feed the instruction 1,2,3 bytes of different category.

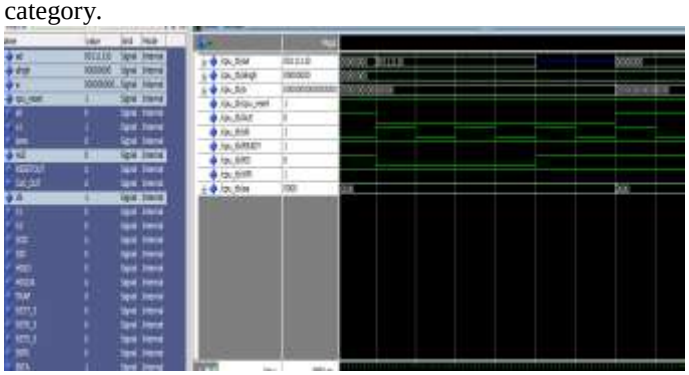

Fig 1.Simulation of 8085CPU connected to RAM

Fig.2 is the sytemverilog interface module for converting the VHDL module into systemverilog we have tried for Interface module since we were facing error in multi driver error for bidirectional signal. We prepare the testbench using Interface construct of systemverilog.
Fig 3 is the systemverilog test bench output using interface.

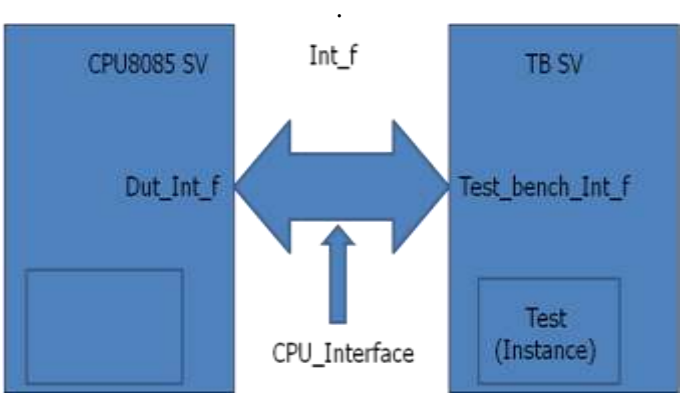

Fig.2 sytemverilog interface

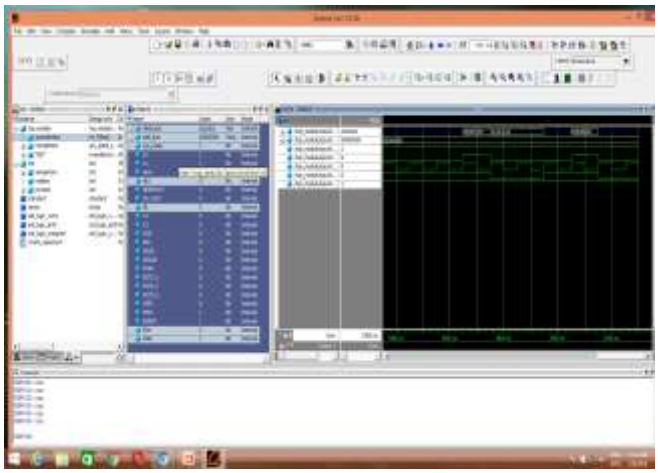

Fig.3 Systemverilog test bench output

VIII. CONCLUSIONs AND RESULTS

At present we prepared the DUT model we now apply the randomized construct to the systemverilog testbench to find the constrained randomized verification.we are presently working on it.

REFERENCES

[1] A. Hoffmann, F. Fiedler, A. Nohl, and Surender Parupalli, A Methodology and Tooling Enabling Application Specific Processor Design, IEEEConference on VLSI Design.

[2] A. Hoffmann, H. Meyr, and R. Leupers, Architecture Exploration for Embedded Processors with LISA, Kluwer Academic Publishers, Boston, 1 ed., 2002.

[3] S. Pees, A. Hoffmann, V. Zivojnovic, and H. Meyr. LISA – Machine Description Language for Cycle-Accurate Models of Programmable DSP Architectures. In Proc. of the Design Automation Conference (DAC), New Orleans, June 1999

[4] A. Hoffmann, A. Nohl, G. Braun, O. Schliebusch, T. Kogel, and H. Meyr. A Novel Methodology for the Design of Application Specific Instruction Set Processors (ASIP) Using a Machine Description Language. IEEE Transactions on Computers-Aided Design, Nov. 2001

[5] Zivojnovi, S.Pees & H.Meyr;LISA–machine description language and generic machine model for HW/SW codesign in Proceedings of the IEEE Workshop on VLSI Signal Processing,San Francisco,Oct 1996

[6] http:www .ertrwth_aachende_lisa_lisahtml

[7] a. halambi, p. grun, v. ganesh, a. khare, n. dutt, and a. nicolau, expression:language for architecture exploration through compiler/simulator retargetability, in proc. of the conference on design,automation & test in europe, mar. 1999

[8] S. Pees, V. Zivojnovic, A. Ropers, and H. Meyr, \Fast Simulation of the TI TMS 320C54x DSP," in Proc. Int. Conf. on Signal Processing Application and Technology (IC-SPAT), (San Diego), pp. 995{999, Sep. 1997.

[9] A. Hoffmann, A. Nohl, G. Braun, and H. Meyr, A Survey on Modeling Issues Using the Machine Description Language LISA, , 2001

[10] S. Yang, Y. Qian, H. Tie-Jun, S. Rui, and H. Chao-Huan, A New HW/SW Co- design Methodology to Generate a System Level Platform Based on LISA, 2005