DESIGN OF A PIPELINED FIR FILTER USING APPLICATION DESCRIPTION LANGUAGE

Umakanta Nanda
Electronics and Communication Engineering
National Institute of Technology
Rourkela, India
uk_nanda@yahoo.co.in

Kamalakanta Mahapatra
Electronics and Communication Engineering
National Institute of Technology
Rourkela, India
kmaha2@rediffmail.com

Abstract—**Usually there are two approaches to design an application specific instruction set processor (ASIP). One of them is at Register Transfer Level (RTL) and another is at just higher level than RTL. Application Description Languages (ADLs) are becoming popular recently because of its quick and optimal design convergence achievement capability during the design of ASIPs. It comprises several stages to design a processor using ADLs. These are architecture design implementation, software development, instruction and system verification. Verification of such ASIPs at various design stages is a tedious job to do. This paper presents the architecture description of a FIR filter using ADL based instruction set description. Here the design process is more consistent while allowing maximum flexibility. Furthermore, it enables the design process in both instruction and cycle accurate modes. The design process of a FIR filter with pipelining is demonstrated here. The main advantage is that the FIR filter model can be optimized with respect to resources by changing the LISA code written in Coware platform.**

Keywords- LISA, ASIP, RTL, FIR filter, HDL, CoWare, Profiling

I. INTRODUCTION

Now a days digital signal processors are considered to be an important member in the processor family because of its flexibility and portability. The flexibility of these processors can be achieved by many ADLs [1-2] like LISA, EXPRESSION, MIMOLA etc. Different phases of design of the processor are distributed among different designers in there respective fields. There should be some type of communication between the groups of design engineers or between the phases of the design. Out of the above languages, LISA [3-4] is more preferable because of it's software development and HDL generation capability.

VHDL and Verilog languages are widely used to design and simulate a processor keeping in mind to implement in hardware. But these models can not be used for architecture exploration and optimization especially to design cycle based or instruction level processor simulation. Because the hardware implementation details are very high which are not required for performance evaluation, cycle based simulation and software verification[5-6].

In this paper we have implemented the architecture of a FIR filter using LISA where the description for each instruction of the instruction set (of that specific architecture) is described properly in CoWare platform. A brief description of LISA is specified in the next section.

II. LISA

Language for Instruction Set Architecture is very much helpful to reduce the gap between the traditional design of a processor using VHDL or Verilog and instruction set languages for architecture exploration. The syntax of this language is having so much flexibility to describe the processor (RISC, VLIW, DSP, ASIPs, Special purpose co processors) instruction set which have complex pipelining.

Generally the processor model that include LISA consists of two sections. Those are Resource and Operation section [6]. Instruction resource is a register that is referred as the instruction register. But the instruction resource can be a memory location, an input pin array or a concatenation of a multiple storage elements. Operation section describe the complete transition function of the processor including pipelining stages such as fetch, decode, execute and write back. This section generally consists of three sub sections. Those are behavior, syntax and coding. Behavior describes the transition function of the processor. Coding section describes the binary image of the instruction word and the syntax section describe the syntax of that particular instruction in assembly programs.

This language is more suitable with the processor designer tool called CoWare [5] for its advanced and flexible features such as,

- Automatic generation of synthesizable RTL with both control and datapath.

- Accurate profiling capabilities for high speed instruction set simulator.
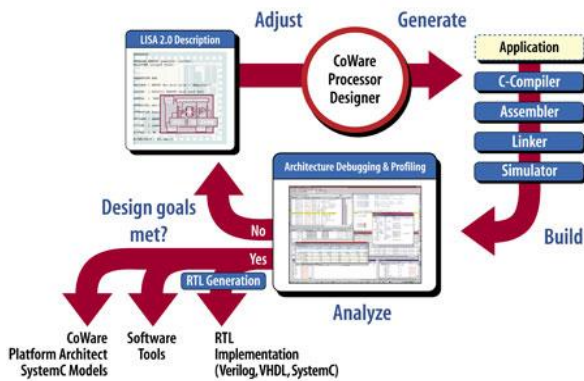- Compatible with extensively used synthesis tools like SYNOPSYS, MAGMA.

Figure 1. Design tool flow of CoWare

- Software development tool generation like assembler, linker, debugger, C- compiler.
- Integrated profiling [5,6] helps to optimize instructions for the target architecture.
- Enables the design team to develop flexible and reusable ASIPs rapidly.

III. ARCHITECTURE DESIGN

Two fields are used for the development of development tools are realized using a high level language to describe the target architecture, and for implementation purpose hardware description languages [3] are used to model the underlying hardware. It is advantage to combine both the development process and the HDL description. Here the LISA compiler can generate both the of these.
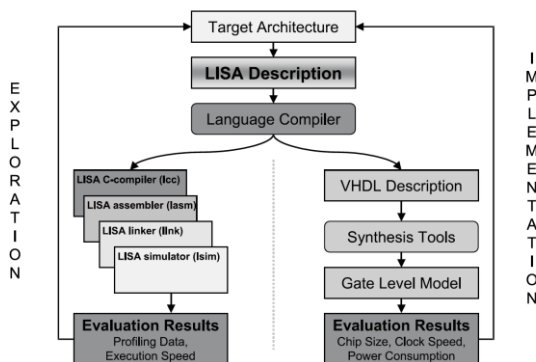

Figure 2. Exploration and implementation

After design exploration and application design the target architecture needs to be implemented which is discussed in next part of this paper.

IV. ARCHITECTURE IMPLEMENTATION

The LISA compiler should derive all the necessary information from the given LISA description since the generated HDL model does not have any predefined components. Then the generated HDL model can be compared to the LISA model components as shown in the figure below.

- LISA memory model derives the memory configuration which summarizes the registers and the memory sets.
- Resource models [3] gives the idea about the structure of the architecture such as pipeline stages and pipeline registers.
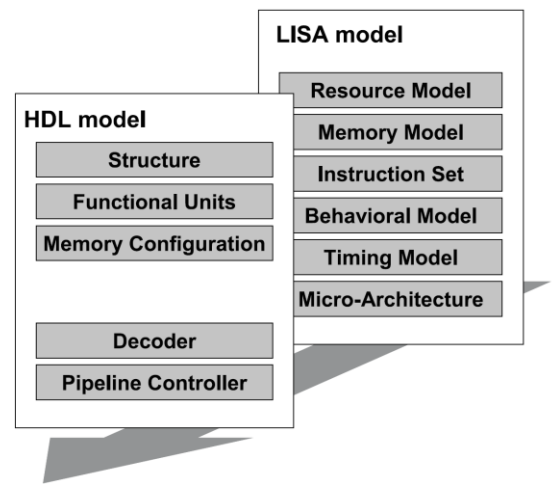

Figure 3. Comparision of HDL And LISA model

- Functional units are either generated as empty frames or with fully functionality depending on the HDL language used.
- Coding information in the instruction set model and the timing model results the decoders.
- Pipeline controller is also generated from the above.

The designer will have full control over the generated HDL model with all its components. The generated HDL model can be analyzed with respect to power, area and time constraints and the optimized HDL model can be replaced with the handwritten HDL code written by the experienced designers which will be done in future work.

V. IMPLEMENTATION RESULT

Here we have taken the case study of a low power FIR filter designed using LISA. Then the resource section of this

model has been optimized. A major decrease in total architecture design time can be seen, as the LISA model results from the design exploration phase.

The software development tool suit includes assembler, linker and simulator as well as a graphical debugger frontend. The tools are the enhanced version of those tools used for architecture exploration. The enhancements for the software simulate the ability to graphically visualize the debugging process of the application under test. The LISA debugger frontend is a generic graphical user interface for the generated LISA simulator as shown in the figure.

It visualizes the internal state of simulation process. Here the C source code, the disassembly of the application as well as all the configured memories and registers (pipeline) are displayed. In this frontend all contents can be changed at the run time of the application. Tools like assembler and linker can be enhanced in functionality as well. More than 30 assembler directives, labels and symbols are supported by the assembler.
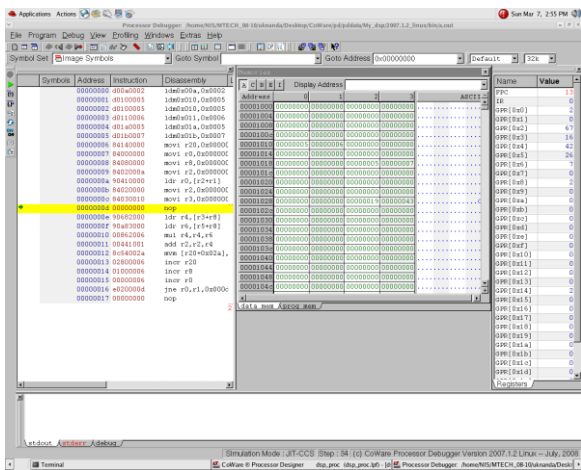

Figure 4. LISA debugger window

The processor debugger provides extensive hardware and software profiling capabilities. Operation profiling gives us the information about Calls/Total which shows the proportion of operation executions for a specific operation to all executed operations.
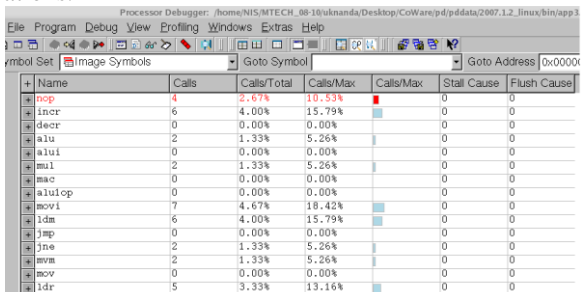

Figure 5. Operation profiling window

$$\frac{Calls}{Total} = \frac{Number\ of\ specific\ operation\ executions}{Number\ of\ all\ operation\ executions}$$

Calls/Max contains information containing the proportion of the execution of a specific operation to the execution of the LISA operation which has been executed the highest number of times.

$$\frac{Calls}{Max} = \frac{Number\ of\ specific\ operation\ executions}{Max\ Number\ of\ specific\ operation\ executions}$$

These information can be shown graphically also.

Memory profiling tells about the access statistics for the memories contained in the processor model. Similarly resource profiling shows the access statistics for all resources modeled with the resource specifier as one of register, program counter and control register in the LISA model as shown in the figure 6.
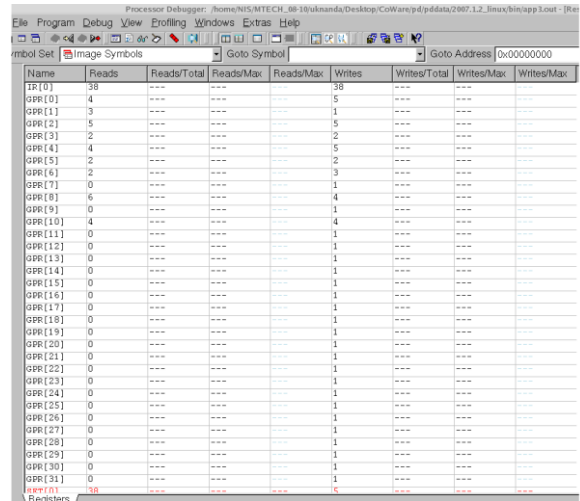

Figure 6. General Purpose Register window

These profiling information is very much required to optimize our design. This architecture was designed on the respective abstraction level with LISA and software development tools were generated successfully.

VI. OPTIMIZED IMPLEMENTATION RESULT

Now we have optimized the FIR filter with respect to the resources we used like,

- Data and program memory
- Instruction set
- Number of general purpose registers

In the operation profiling we can see that the instructions or operations like decr, alui, mac, alu1op, jmp, sub, and, or and mov have not been called in our specific application. So writing the behavioral code for these operations is not required and the result will be unchanged. To reduce the resource

section further we can take 16 general purpose registers (GPR) instead of 32 which will reduce the area of our model.
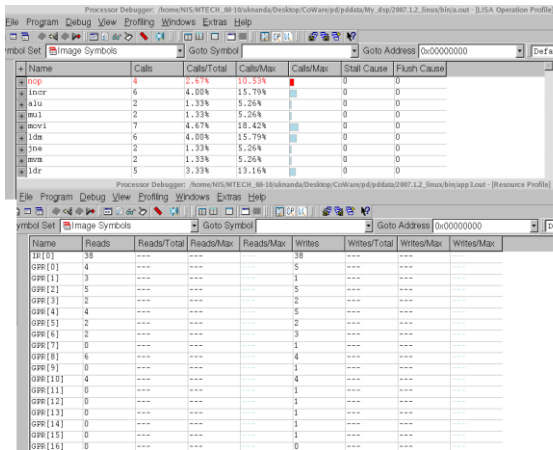

Figure 7. Optimized profiling window

In the optimized model we have less space allocated for data and program memory. Program memory starts from 0x0000 to 0x0015 and Data memory starts from 0x0016 to 0x0042 reducing the area further.

VII. CONCLUSION

ADLs are growing in demand for design of ASIPs because we can change our model according to the instruction set provided for the specific application. In this paper we implemented FIR filter architecture using LISA. Then the same model was optimized with respect to resources like data memory, program memory, instruction set and number of general purpose registers.

Our future work will focus on the generation of the RTL file from which we can get the HDL model of the same architecture. The automatic generation of pipelined functional units of the ASIP with optimization in data path and resources will be another interesting research work. Furthermore we can analyze and compare the area, power and timing issues of our generated HDL model and the hand written model for the same architecture.

REFERENCES

[1] Anupam Chattopadhaya, Arnab Sinha, Dandian Zhang, Rainer Leupers, Gerd Ascheid, Henrich Meyr, "Integrated Verification approach during ADL driven processor design", Microelectronics journal 40(2009), page 1111-1123.

[2] Welhua Sheng, Jianjiang Ceng, Manuel Hohenauer, Hanno Scharwachter, Rainer Leupers, Henrich Meyr, Institute for Integrated systems, Achen, Germany, "A novel approach for fexible and consistent ADL driven ASIP design", DAC'04, June 7-11, 2004, San Diego, California, USA.

[3] Andreas Hoffman, Member IEEE, Tim Kogel, Achim Nohl, Gunnar Braun, Oliver Schliebush, Oliver Wahlen, Andreas Wieferink and Henrich Meyr, Fello, IEEE, "A novel methodology for the design of application specific instruction set processors (ASIPs) using a machine description language". IEEE transaction on Computer Aided Design of integrated circuits and systems, vol-20, number 11, Nov.-2001.

[4] O. Schliebusch, A. Chattopadhayay, E M Witte, D Kammler, G. Ascheid, R Leupers, H Meyr, "Optimization techniques for ADL driven RTL processor synthesis" in IEEE workshop on rapid system prototyping(RSP), Montreal, Canada, June 2005.

[5] CoWare, The ESL design Leader reference manuals, Product version V2007.1.2, June-2008.

[6] CoWare,inc,http://www.coware.com.