ASIC design with AHPL

Sadiq M. Sait, Muhammad S. T. Benten, and Asjad M.T. Khan
Department of Computer Engineering
King Fahd University of Petroleum and Minerals
Dhahran-31261, Saudi Arabia.
e-mail: facy009@saupm00.bitnet

1 Introduction

One of the greatest challenges facing the electronics industry today is to reduce the *"time to market"* new products. This time includes design time, manufacturing time and testing time. Design time is the sum total of time taken to design the architecture and to perform layout. Reducing design time in fact is more important than optimizing the area or performance of the IC. This is especially true in the case of Application Specific Integrated Circuits (ASICs) since, unlike a microprocesor, they are not programmable, and have a limited market.

A high degree of automation is required to reduce the design time. Automation can be achieved by standardizing the design process. In this paper we present a design automation environment developed at KFUPM which helps in reducing the design time and effort both at the archtiectural level as well as at the layout level.

The KFUPM DA system is a blend of tools developed locally and those developed at the University of Arizona, University of California at Berkeley, University of Washington and the Microelectronics Center of North Carolina (MCNC).

This DA system is suitable for design of any synchronous digital systems in VLSI. The systems has been successfully used to synthesize designs such as data compression chips, protocol processors, programmable CRC checkers, digital controllers, computer arithmetic algorithms and small microprocessors. In addition to providing workable designs in short turn-around time the KFUPM DA system is an excellent educational tool to convey concepts related to digital system design, synthesis, and VLSI design automation. The system takes input at the register transfer level. This specification undergoes *logic synthesis* to give a gate level specification in terms of a netlist of standard gates (NAND, NOR etc.) and flip flops. Simulation is done both at the register transfer and the gate level and results are compared to verify the translation process. This finishes the synthesis task. The netlist is given to physical design subsystem which has placement, routing and graphic layout manipulation tools along with a standard library. The layout produced is checked for design rules and circuit is extracted. The extracted circuit is again simulated and the results compared with the results of the functional simulation at the register transfer level. The designs can be fabricated by a number of foundries worldwide. One of the designs was fabricated by Orbit Semiconductor Inc. of Caliifornia, USA.

This work is an extension of the ongoing AHPL based design automation work at KFUPM [4, 5, 9]. A detailed schematic diagram of the AHPL DA system is shown in Figure 1. The sequence involved is discussed in the following section.

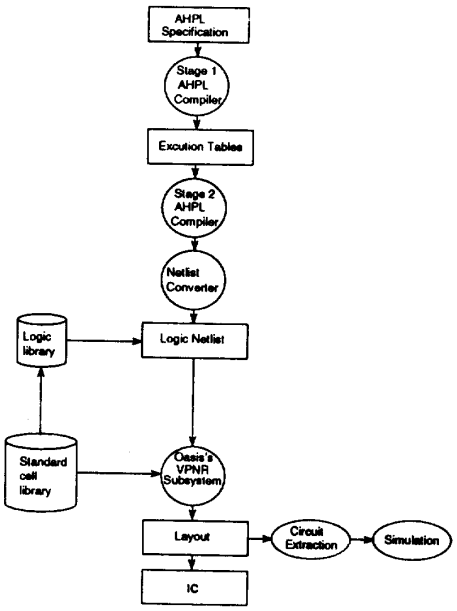

Figure 1: The AHPL based design automation system.

0-7803-1772-6/94/$3.00

2 AHPL DA System

Universal AHPL, an extension of A Hardware Programming Language (AHPL), is used as the register transfer language for specification of the design [1]. It has been used at KFUPM for the earlier automation system design [4, 5, 9]. It is a simple language yet is sufficient to model highly complex digital systems such as parallel processors and dataflow machines. The system is described by sequential automatons termed as MODULES, which contain the procedural part, and combinational circuit part called CLUs. A compiler for converting the system specification into an intermediate representation in form of tables after performing syntax and semantic analysis is available [2].

To aid in the construction of an efficient model, exercise hardware tradeoffs, and verify the logic of design, the environment is supported by a functional AHPL simulator [3]. The task of logic synthesis is done by the *stage-2* compiler which takes the internal representation of the system and produces a gate interconnection list using standard NAND, NOR, AND etc., gates and flip flops. It extracts the controller from the system description and realizes it using the one-hot encoding. It realizes register and memories using flip flops in the data part. The output produced by this stage is a netlist in the form of a linked list representation giving the gate number and its input and output gates. These two stages complete the synthesis task.

The netlist produced by the AHPL hardware compiler is technology independent. It may be logically correct, but may not be suitable for implementation. An example is of an 8-input AND gate used in the logic design. In a cell based system such a circuit/layout may not be available in the cell-library. The Oasis standard cell library is used in this implementation [7]. The Stage-2 AHPL compiler generates a logic netlist using combinational gates (AND, OR, NOT, NAND, NOR, XOR etc.,) and three types of D flip-flops, with enable, and asynchronous set and reset inputs. In Oasis, corresponding to each layout cell of the cell library, there exists a logical/switch level model that can be simulated using the RNL Logic level simulator [6]. An AHPL logic library using the Oasis models of the cell library elements has been designed. This library contains the RNL models of all the combinational gates and the three types of flip flops required for implementing the Stage-2 compiler logic netlist. A netlist converter has been designed to translate the netlist produced to the RNL netlist using AHPL logic library. This netlist is verified at the switch level by simulating it using RNL and then it is mapped to a layout.

One of the important constituents of the physical design subsystem is the cell library. It contains the design rule correct layouts of the standard gates and flip flops. The standard cell library consists of scalable CMOS cells compatable with 2μ MOSIS SCMOS technology. It consists of combinational logic gates (AND, OR, INVERT cells with different fan in) and sequential logic (D type flip flop with reset and Tristate buffer). Apart from these a number of useful combinational gates like Xor and And-Or-Invert (AOI) are also provided. All of these cells are tailored to support scan based testabilty in the design. The netlist converter program modifies the netlist generated by the AHPL compiler to gates which are present in the library.

The logic level netlist is translated into a standard cell layout using the **VPNR**, Vanilla Place and Route subsystem of **Oasis**. This system also supports the inclusion of scan path based testing circuitry and consistency checks during the placement and routing phases of the design. The VPNR system uses quadrisection algorithm for placement [10]. It recursively partitions the input logic netlist into quadrants until the partition contains one cell row. This is accompanied by approximate global routing phase. Once the cells are placed the scan path is threaded through all the flip flops. Then the process of detailed global routing is done by constructing a minimum spanning tree for each net, finding exact location of nets crossing cell rows, and inserting feed through channels. The task of routing global signals (e.g. clock) is done using a fixed routing scheme. The actual routing uses a greedy/left edge based router [11, 12]. The final task of layout assembly is done using the **Magic** layout editing system.

Once the entire system layout is ready, it is time to simulate it again to make sure that the design would work when fabricated. Simulation is necessary as at higher level of design a number of important specifications like timing information are left out and this may cause the system to malfunction. The circuit is extracted using the Magic's hierarchical circuit extractor. This extracted circuit can be converted into three different formats to feed switch level simulator **esim**, timing analyzer **crystal** and circuit level simulator **spice** [6] or Magic's own simulator **irsim**. Simulation can be carried out at different levels of detail according to the system complexity. If the simulation results agree with the functional simulation results then there are strong chances that the chip would work when it is fabricated.

3 Extensions

The work carried out so far has targeted semicustom design, SLAs, Gate Arrays, PPLAs, and standard cell methodologies. The frontend of the system, upto the netlist, is independent of technology and

architecture. The output of this stage can be conveniently mapped to programmable devices (PLD's) and field programmable gate arrays (FPGAs). Current research on the extensibility includes the task of synthesis from algorithmic specification. The approaches under consideration include the use of high level programming languages eg. Pascal, C subsets for input specification. The use of a VHDL subset is also under investigation. The results from these studies can be easily integrated into this DA system to convert it from an RTL level DA system to an algorithmic level DA system. Other work includes investigation of formal synthesis option that will do away with the comparative simulation approach to verification.

4 Example

In this section we present an example of a programmable CRC generator circuit that illustrates the complete translation of an AHPL model to a VLSI layout. This circuit was fabricated as a tinychip project. We begin by presenting a verbal description of the digital system to be implemented. The verbal description is translated to an algorithmic description, which is then translated to AHPL. This AHPL model is then synthesized in VLSI. The synthesized layout is then extracted and simulated to verify if it represents the function modeled in AHPL.

Refering to the AHPL model, the initialization is done in the 1^{st} STATE. In the 2^{nd} STATE, a 64-bit message is supplied sequentially on line MESIN and a 16-bit CRC pattern is simultaneously generated and stored in a register, CREG. When all the message bits have been processed, the CRC pattern is ready by that time and it can be serially appended to the message data stream for transmission. The CRC pattern is serially appended to the message in 3^{rd} STATE. The generator then goes to the initial state to perform the same task described for another message, if any.

The 16-bit CRC pattern generated is available in 66^{th} clock pulse. The transmission of 16^{th} bit takes place in 81^{st} clock pulse. Thus, the number of clock cycles required to generate and transmit any CRC in our implementation, is the sum of the size of the message and the degree of the generator polynomial used.

```
MODULE:CRCGENERATOR.
MEMORY:CRCREG{16};COUNT{6}.
EXBUSES: C;Z.
BUSES: X{6};Y;ZOUT;CRCRDY.
EXINPUTS:CLK;RESET;START.
EXINPUTS:MESIN;A.
CLUNITS:INC{6} <: INCR <. 6 .>.
BODY SEQUENCE: CLK.
  1 COUNT<=6$0;
    CRCREG<=16$0;
    =>~(START)/(1).
  2 ZOUT=MESIN;
    Y=MESIN@CRCREG{15};
    COUNT<=X;
    CRCREG<=(Y,CRCREG{0:3},CRCREG{4}@Y,CRCREG{5:10},
             CRCREG{11}@Y,CRCREG{12:14}) !
           (Y,CRCREG{0},Y@CRCREG{1},CRCREG{2:13},
             CRCREG{14}@Y)*(A,~A);
    =>~(&/COUNT)/(2).
  3 COUNT<=X;CRCRDY=\1\;
    CRCREG<=\0\,CRCREG{0:14};
    ZOUT=CRCREG{15};
    =>(~(&/COUNT{2:5}),COUNT{1})/(3,1).
  ENDSEQUENCE
CONTROLRESET(RESET)/(1);
    X=INC(COUNT);
    Z=ZOUT;
    C=CRCRDY.
END.
CLU:INCR(X) <. I .>.
 "I-BIT INCREMENTER CONSTRUCTED WITH EXCLUSIVE OR GATES"
 "AND GATES AND AN INVERTER"
 INPUTS: X{I}.
 OUTPUTS: Y{I}.
     BODY
      FOR J=(I-1) TO 0 STEP -1
       CONSTRUCT
        IF J=I-1 THEN Y{J}=~X{J}
          ELSE Y{J}=X{J}@(&/X{J+1:I-1})
        FI
      ROF.
  END. "INCR"
```

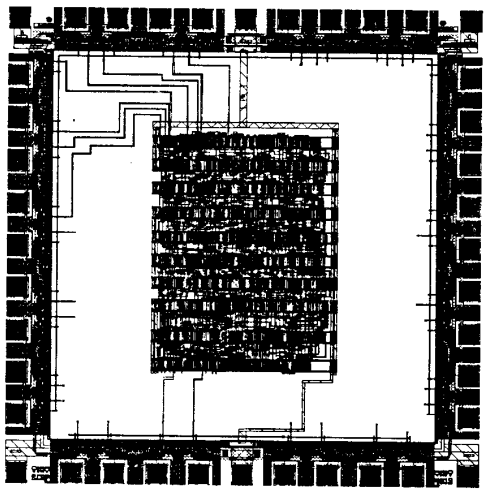

Figure 2: Layout of programmable CRC chip.

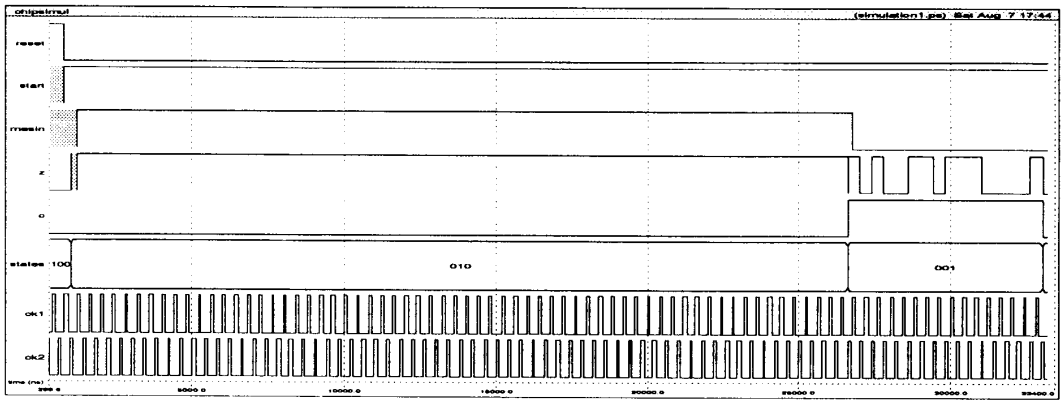

Figure 3: Simulation of the extracted programmable CRC circuit.

5 Conclusions

This paper presented the progress made at KFUPM in putting together an AHPL based design automation system using indigenous effort, software tools developed locally and in US universities into a cohesive system. The AHPL language is used as the frontend specification medium due to its closeness to hardware implementation issues. The system is modular and technology independent so that future extensions and specific implementation issues can be added/modified.

Acknowledgements

The authors acknowledge support by King Abdul Aziz City for Science and Technology (KACST) in the form of research grant AR 11-21 and the King Fahd University of Petroleum and Minerals.

References

[1] Fredrick. J. Hill and G. R. Peterson, "Digital System: Hardware Organization and Design", Second edition, John Wiley and Sons, New York, 1978.

[2] M. Masud, "Modular Implementation of a Digital Hardware Design Automation System", Ph.D Dissertation, University of Arizona, 1981.

[3] M. M. Al-Sharif, "Functional Level Simulator for Universal AHPL," M.S. Thesis, University of Arizona, 1983.

[4] Sadiq M. Sait, "VLSI Mask Descriptions from Register Transfer Level Descriptions: An Automated Approach", Ph.D Dissertation, KFUPM 1987.

[5] M. Masud and Sadiq M. Sait, *Universal AHPL — A language for VLSI Design Automation*, IEEE Circuits and Devices Magazine, September 1986.

[6] VLSI Design Tools Reference Manual, Release 3.1, NW Laboratory for Integrated Systems, FR-35, University of Washington, February 1987.

[7] K. Kozminski, Ed. , *"OASIS Users Guide"*, MCNC, Research Triangle Park,North Carlolina, October 1992.

[8] Robert N. Mayo et.al, 1990 DECWRL Livermore Magic Release, Digital Western Research Laboratory, September 1990.

[9] Sadiq M. Sait, "Integrating UAHPL-DA system with VLSI Design Tools to support VLSI DA courses", *IEEE Transactions on Education.* , Vol-35, No.4, pp321-330, 1992.

[10] P. R. Suaris and G. Kedem, "A new Approach to Standard Cell Layout", 1987 International Conference on Computer Aided Design, pp 474-477, November 1987.

[11] R. E. Rose, "Greedy Algorithms for wiring in VLSI", Masters Thesis, Department of Computer Science, North Carolina State University,1985.

[12] M. J. Lorenzetti, M. S. Nifong and J. E. Rose, "Channel Routing for Compaction ", Proceedings of the International Workshop on Placement and Routing, May 1988.