

whole thing into many small pieces. We wrote verilog module for each piece and test them separately. After that, we created a top level file to assemble all the pieces together and do the simulations.

2.4 Memory Hierarchy

While producing caches for load unit, we discovered that in order to avoid much pipe stalls, we would need Synchronous Dynamic Random Access Memory (SDRAM) outside of the chip. This will assure issueing of requested datas every cycle except for first data request. Since caches are non-blocking caches, queue in size of SDRAM pipeline stages are implemented as well.

3 Synthesis

The behavior Models cannot be mapped directly to synthesis library. Some of the language structures are not fully supported by the tool, such as for-loop. Therefore we have to modify the models and iterate the simulations to match the results of the verilog.

3.1 Tools

The tool we used to synthesize the behavior model to structural netlist is Cadence Synergy. Our goal was to make the netlist function exactly the same as the verilog behavior model, therefore, we didn't put in aggressive constrains on the area and timing.

3.2 Library

We utilize the generic Cadence library for our design. And we have a 0.5um design rule.

4 Layout

The register file is custom drawn by layout. It is a combination of arrays of standard 14T-cells and have sense amplifiers connected to their bit lines.

4.1 Register File

One of the largest layout we did was register file as you can see in figure 3, there are two register file cells connected in parallel with one decoder and two sense amplifier strips.

4.2 Conversion

The register file make by the verilog file will be much bigger than it should be. In order minimize the cost, we customly design our own standard 14T-cell.

4.3 Transistors

We use minimum sizing for the 14T-cells. The correct sizing for the transistors of the sense amplifier is throughly tested by hspice.

5 Floorplan

The place and route of our design.

5.1 Modules

The main modules are the "add" and "load" pipes which we cell ensembled. Then we do a high level place and route of all the modules to complete the full chip.

5.2 Interconnections

Optimize the wire length and increase the routing density are always our highest priorities. We also try to align all our internal blocks to create rectangular channels for a puzzle like fit.

5.3 Power and Ground

Power and ground lines have the highest net criticality. We first creat an IO ring for the power and gnd around the chip, then we stretch the rails into the chip to facilitate better distribution.

6 Final Chip

Our final chip size is 13990.70 um x 17103.33um. With 256 total pins, 126 of those are vdds and gnds. The i/o signals are i_cache and d_cache's addresses and data and two clock signals.

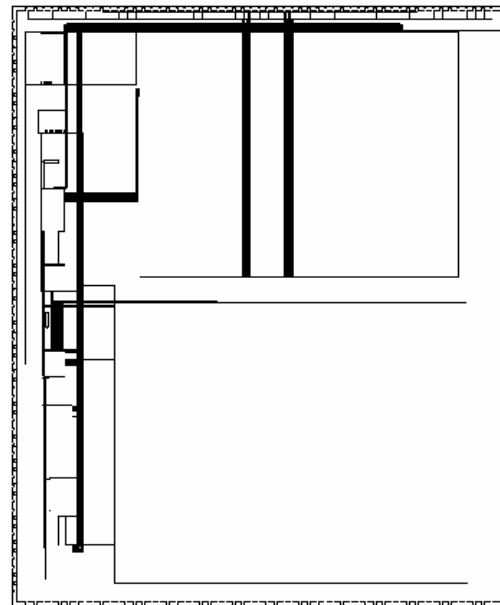


Figure 4: full chip layout

9 Conclusion

Overall, we use a top down design methodology and with all the cad tools, we are able to see through the whole ASIC design process - a prime goal of this course.

VLIW - RISC Processor

Computer Science 250 - Final Project - May 6, 1996

by Johnson Kin, Leo Lee, William Tsu, and Young Cho

Instructor: Howard Sachs

Department of Computer Sciences
University of California at Berkeley
Berkeley, Ca 94720

Abstract

Architecture of this Very Long Instruction Word (VLIW) processor executes six word instructions through five stage pipeline running on two phase clocks. Design consists of two load unit, two arithmetic units, a store unit, a branch unit, an instruction cache, a data cache, a register file, and a reservation station. Instruction cache issues six words into appropriate units at the same time. Then each unit checks reservation station for dependency and usage. If any registers needed by a unit is busy in another unit, dependent unit stalls until requested registers become free. Two load units load words from data cache into register file. In parallel, two arithmetic units add, subtract, shift arithmetically, or logically values in specific registers. Then results are written back into another register. Store unit and branch unit, though not implemented in our logic design, need to store register values into data cache and calculate appropriate value for program counter (PC). Data caches are non-blocking, thus require synchronous DRAMs. Since all these components are connected to register file, register file need to have five write ports and nine read ports. Whole chip is implemented from behavior model, then synthesized using a half micron technology, and floorplanned.

1. Introduction

Idea for VLIW processor was first examined as early as 1970's. Due to limited technology and complex implementation, the idea was not popular at that time. When Reduced Instruction Set Computing (RISC) processors came out with multi-stage pipeline, designing VLIW processors took a different turn. Since there is increasing demand on processor power, it is necessary to re-evaluate the performance and design decisions for fabrication of VLIW - RISC processor.

2 Architecture

The whole architecture and instruction set is defined by Instructor Howard Sachs. We make some minimum modification to the specs when it seems fit.

2.1 Two phase clocking approach

Two phase clocking is probably best to our design after considering a hardware complexity trade-off. As the two

phase clocking strategy allows us to pass slack, and still give us reasonably flexibility in designing the hardware in each of the stages as it doesn't require us to know the delay values of each of the components, as in the 1 phase clocking approach.

2.2 Load and Arithmetic pipe

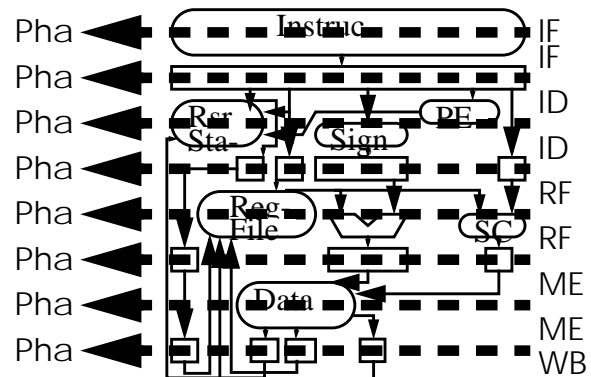


Figure 1: Load Pipe

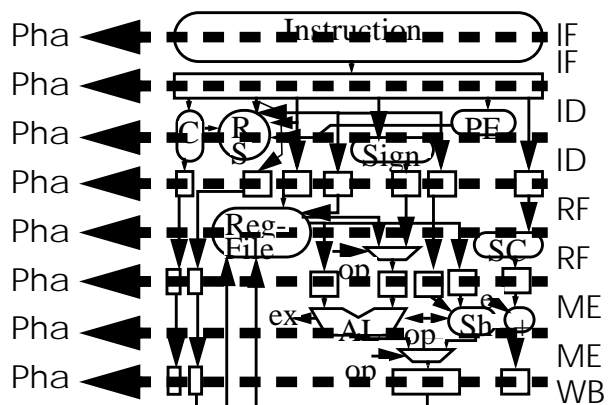


Figure 2: Arithmetic Pipe

As you can see in figure 1 and figure 2, we have designed a full load and arithmetic pipes. The behavioral models were created in verilog and tested exhaustively to verify the pipelines.

2.2 Behavior Model

We use the bottom up design methodology. To make everything modular and easy to test, we divided the