ECE437 Final Report

Name: Cong Ku
Zhiyang Lin

TA: Pranav Laxmanan

Date: April 24, 2015

Overview

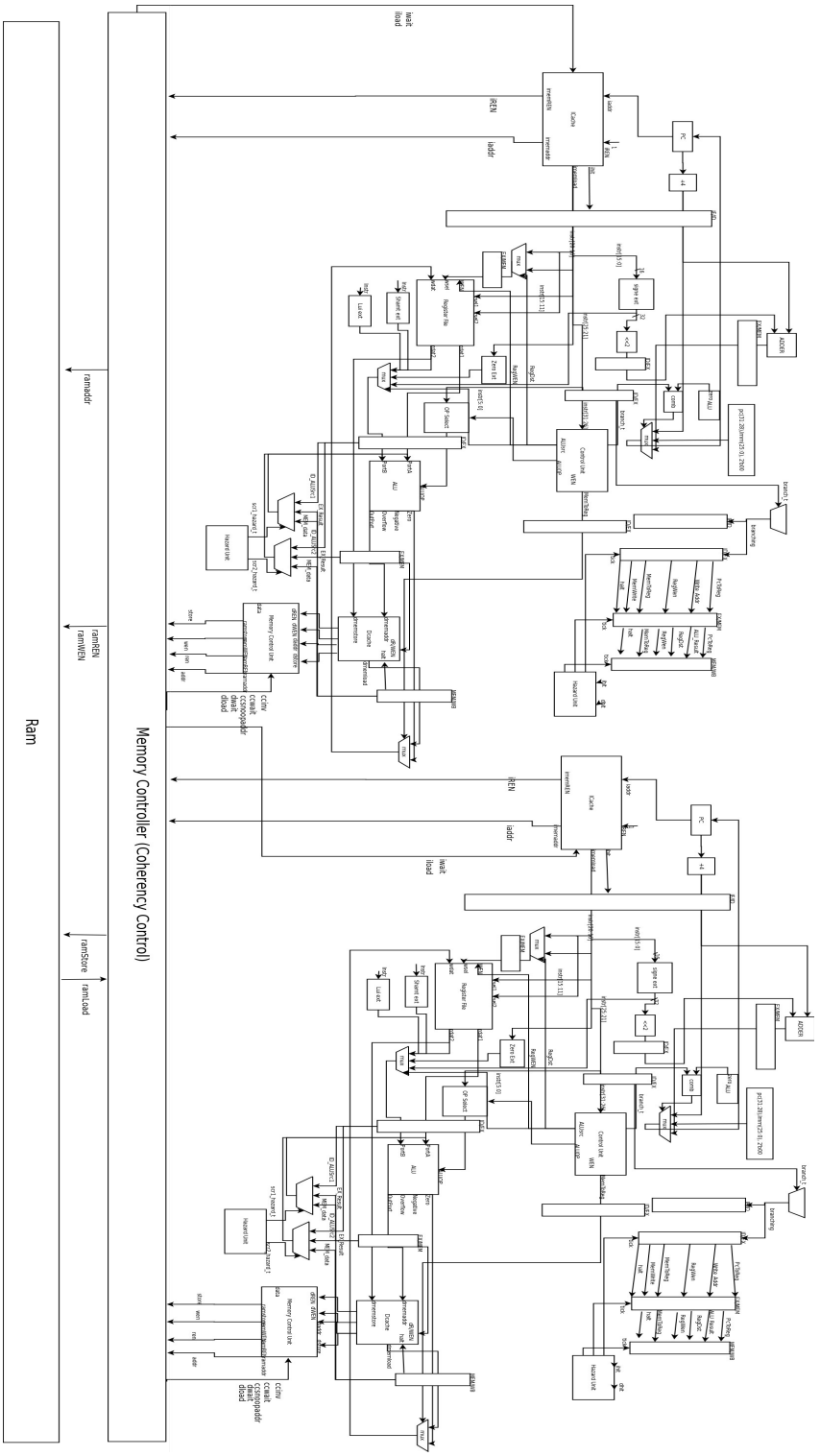
In the past few weeks, We have successfully implemented multi-core and cache with our current existing pipeline processor.

The processor had a performance boost each time these new designed have been implemented. After the Instruction cache and data cache was implemented, a huge performance can be felt on some heavy looped and data contiguous cases like mergesort.asm, since LW and SW instructions will take huge advantage of cache. LW and SW will always read or write to cache if possible, in that way, CPI is a lot better than an actual memory access. For advantages and disadvantages, the cached version is better in performance in any respect, but require extra logic and state machine.

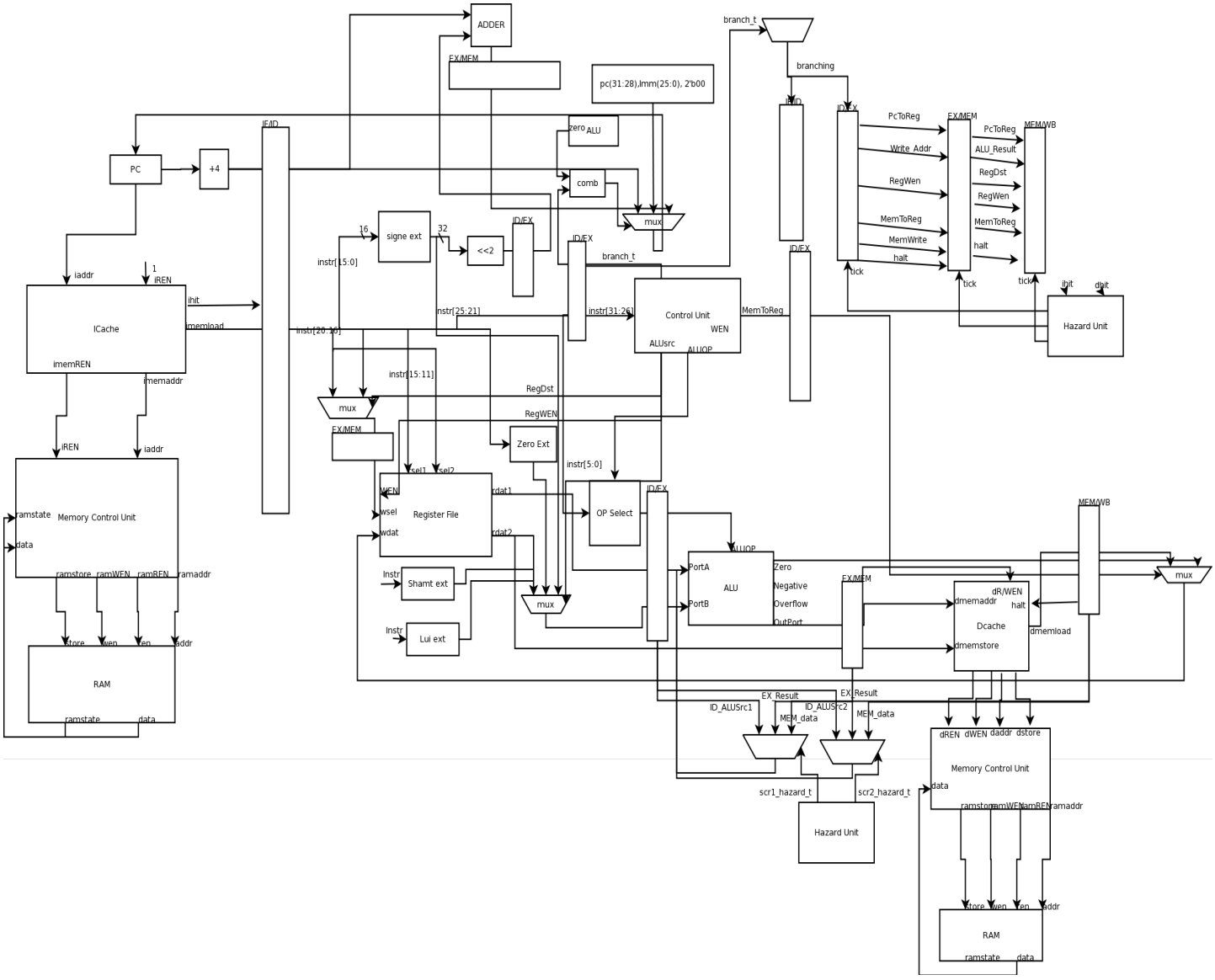
The multi-core design takes advantage of parallel programming to achieve better performance. Two processor have different address pointer and they can run different instruction at same time if there is no coherency issue between two processors. The multi-core disign give us better CPI performance since it still take advantages from Cache and more instructions can run at same cycle. For advantages and disadvantages for this design, the better CPI can be achieved in most of cases but it still require complex logic and more cores lead to higher power consumption.

For performance analysis of this report, we will use mergesort.asm and dual_mergesort.asm as our test files. For each design, we will collect the following data: Synthesize Frequency, average instructions per clock cycle(1/CPI), Latency, FPGA resources required and Speedup. The expected test result is pipeline, pipeline-cache, pipeline-multicore ranked from worst to best for each of these measurements.

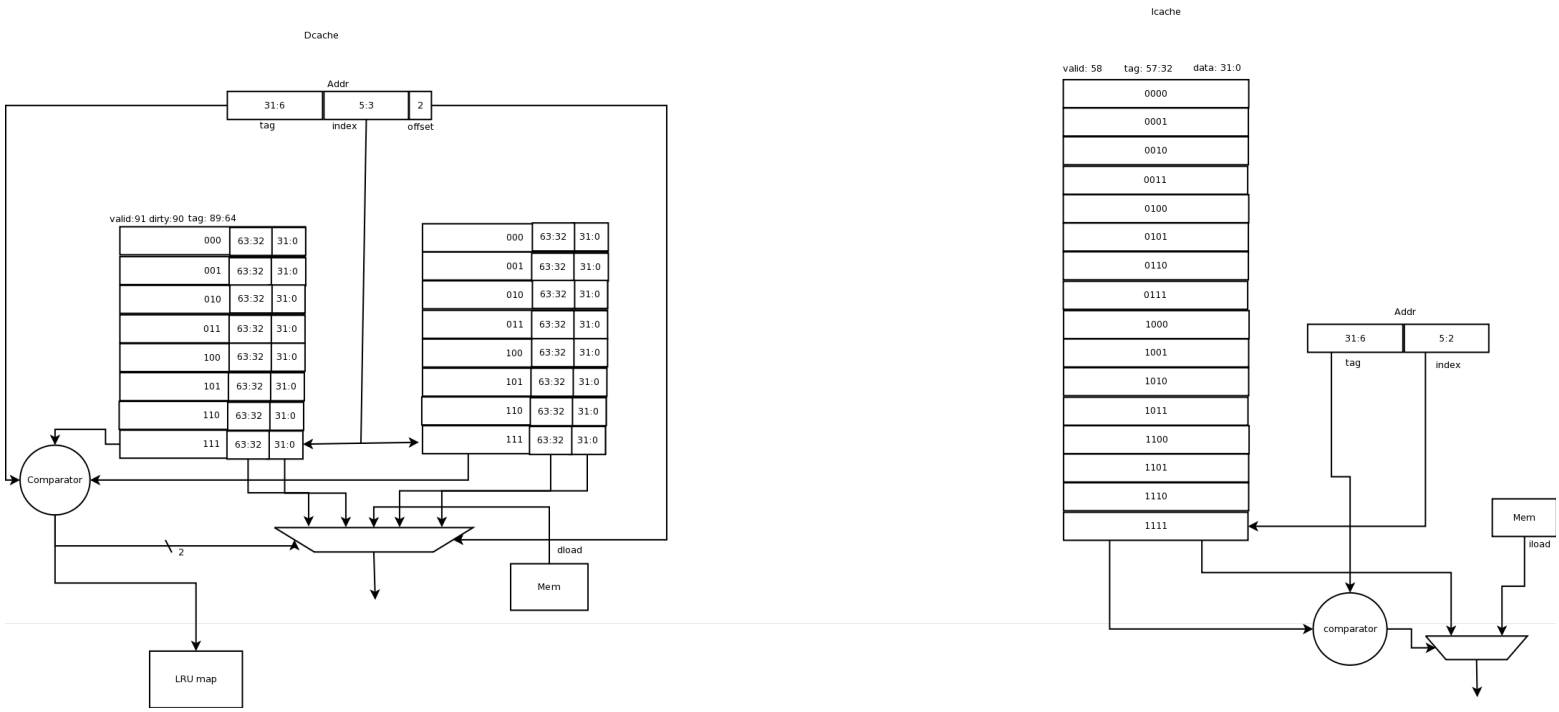
Design



Figue. 1 Multicore CPU Overall Block Diagram

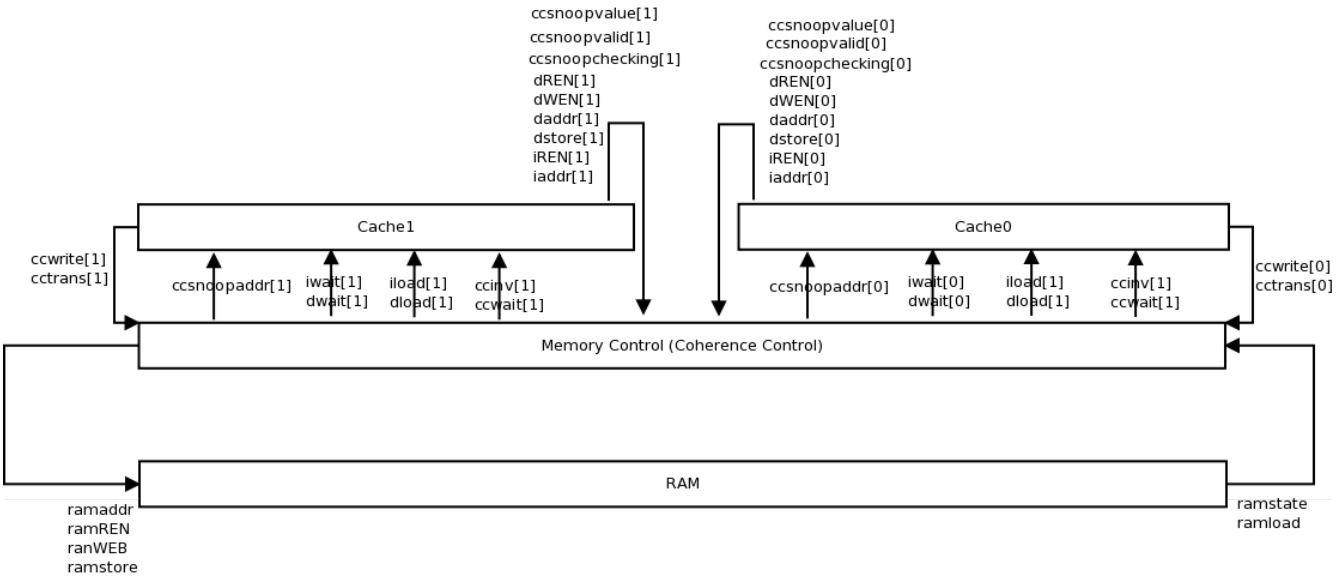


Figue 2. Pipeline CUP with Cache Block Diagram

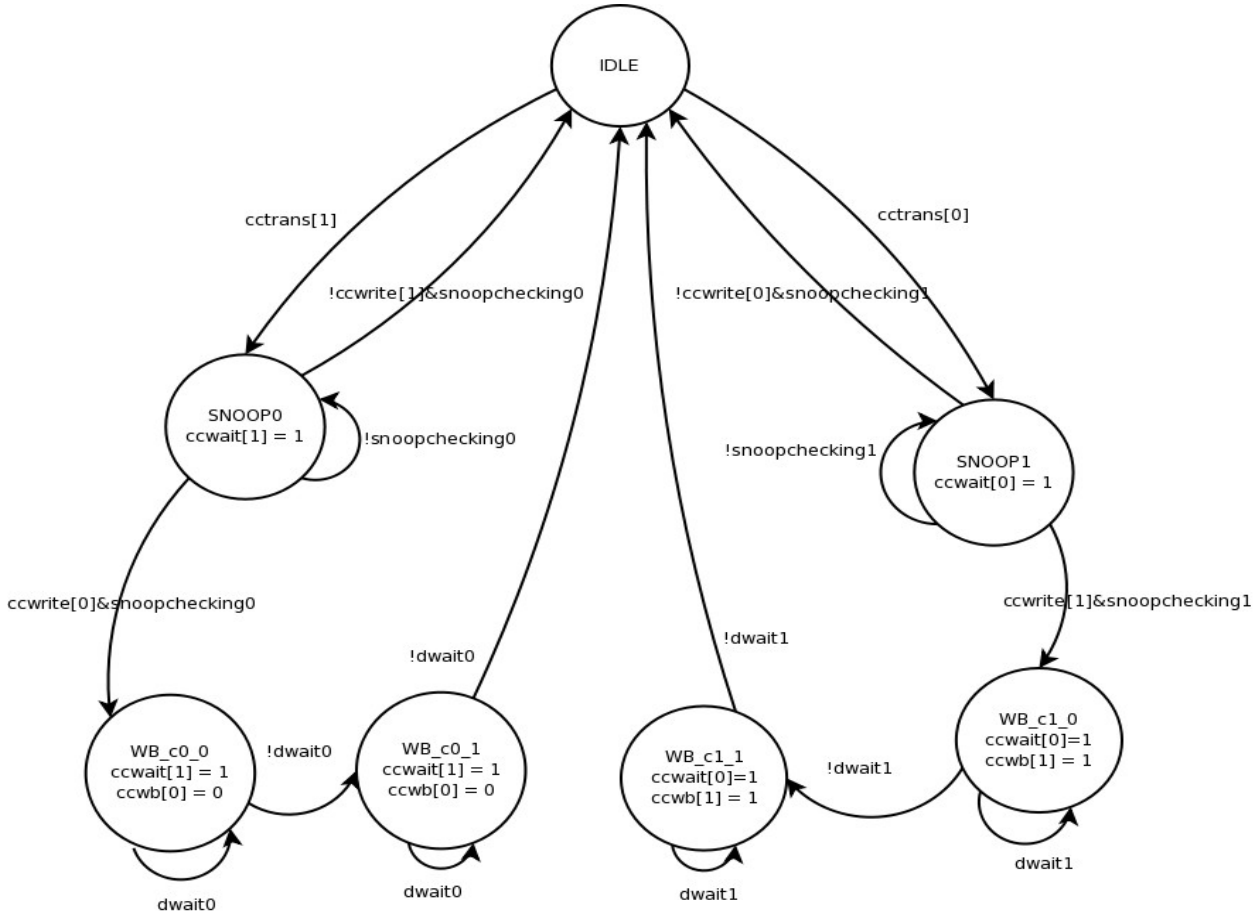


Figue 3. Cache block diagram

Figue 4. Dcache state transition diagram



Figue 5. Coherency Block diagram



Figue 6.Coherency state transition diagram

Results

	Pipeline(p1)	Cache(P2)	Mulicore(P3)
FMAX (MHz)	41.67	39.22	33.41
Latency(ns)	255.46	246.33	214.75
IPC	0.136	0.419	0.621
MIPS	2.943	8.741	10.913
FPGA Res (Registers)	1624	4118	8571
Speedup(comp with p1)	1	2.97	3.71

Conclusions

As the results shows, the processor performance have significant improve each time a new design is added. There is a huge performance boost between p1 to p2 from previous result. As MIPS shows, P2 is 2.97 tims faster than p1. There is also a performance boost from p2 to p3 due to parallel programming capacity. P3 is about 1.3 times faster than p2.

As for space usage, more advanced/complex design take significantly more space. P2 uses more than 2 times registers p1 use and p3 registers uses just doubled from p2. But again, the FPGA board is large enough to hold those registers.