A Wild Multi Core Processor Appeared

ECE 437

Cole Reinhold & Samodya Abeysiriwardane

TA: Adam Hendrickson

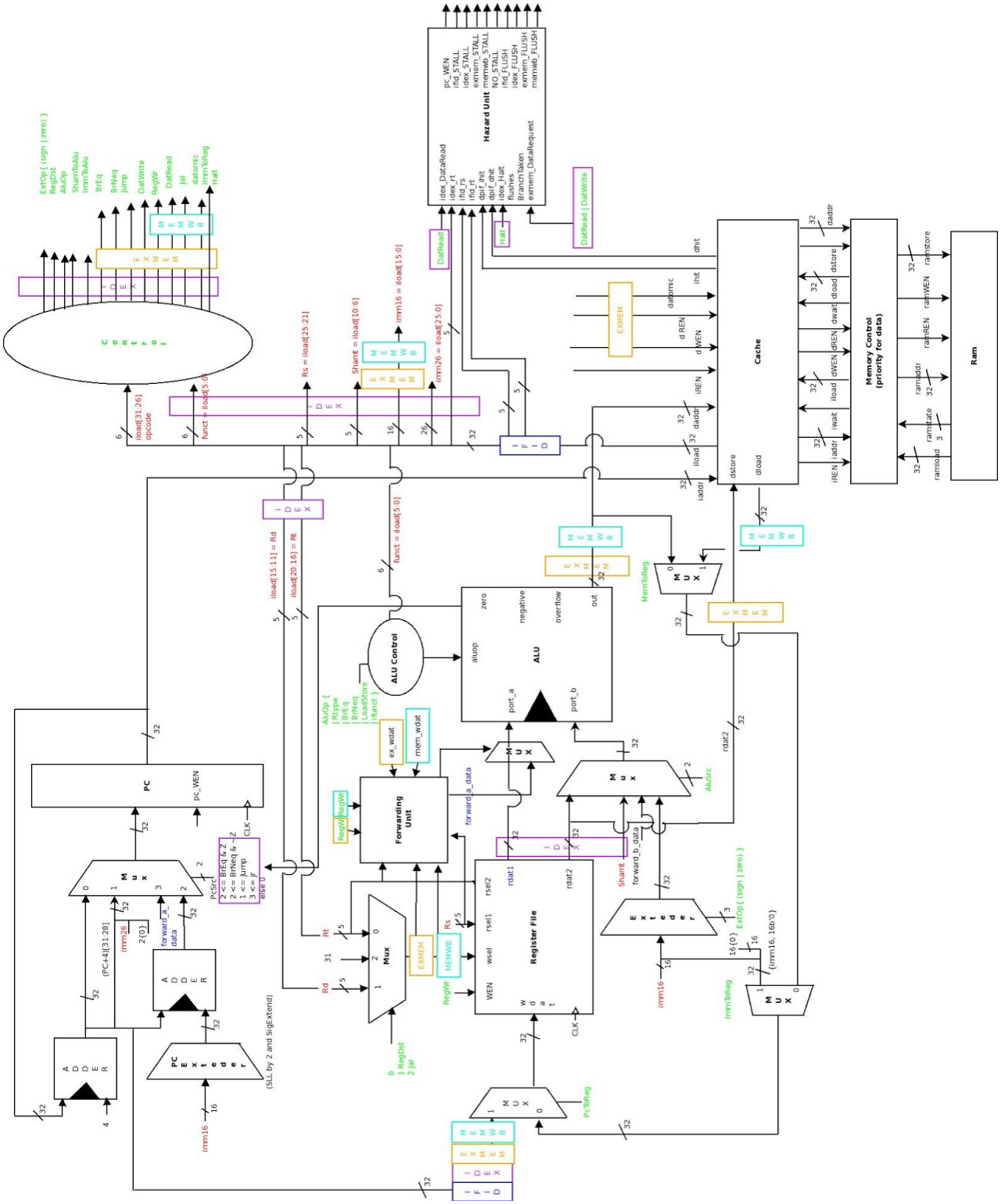
Due: 12/12/14

Overview

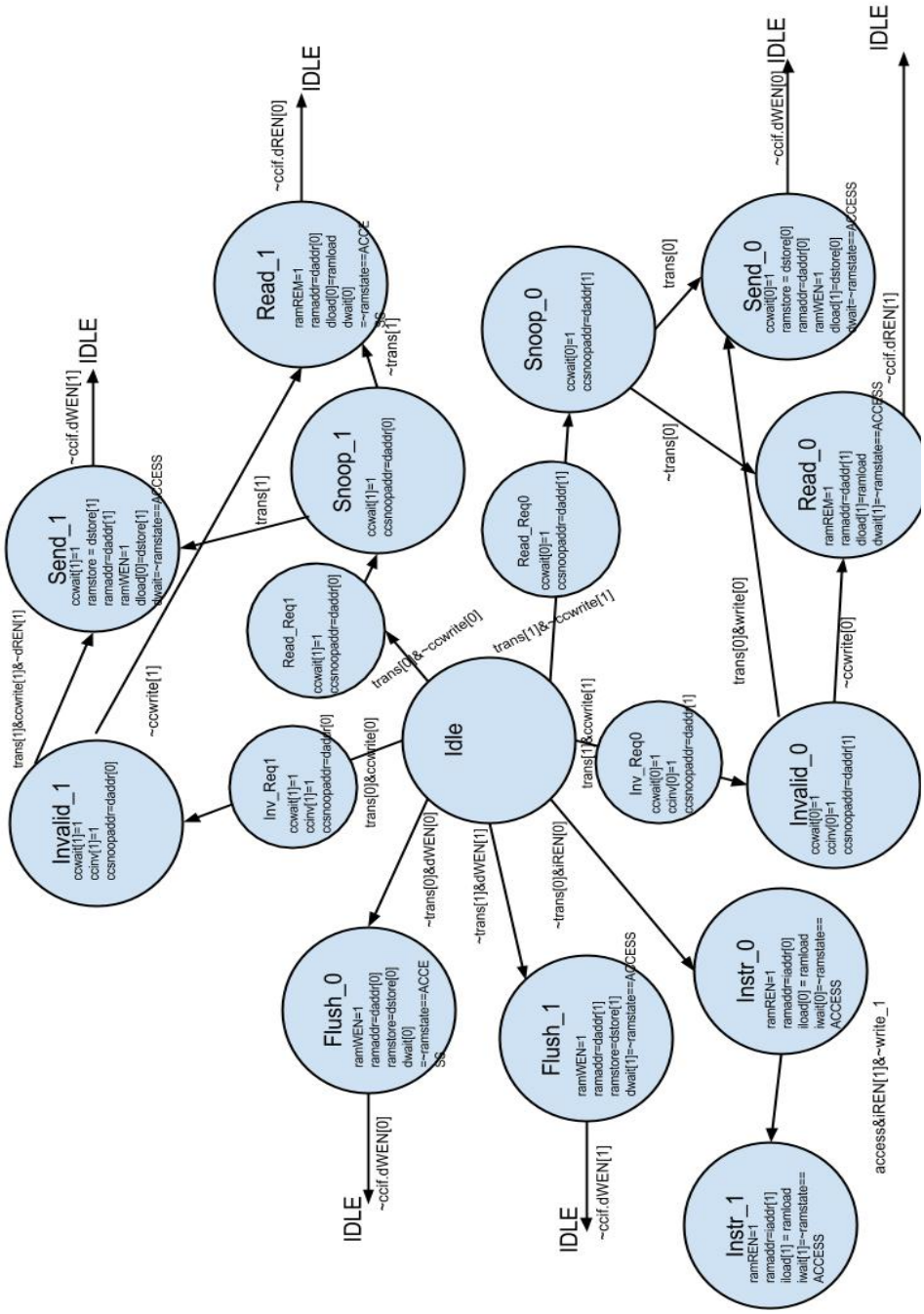
This report focuses on the comparison of three processor designs, a pipeline processor with cache, pipeline processor without cache, and a multicore processor with cache. For reference, block diagrams of the designs can be found in the processor design section. In the results section, performance metrics are discussed after running a mergesort program on both pipelined processors. This mergesort program was modified to take advantage of the multicore processor as well. We thought that mergesort was a great program to test the functionality of our designs. It is made up of 17 different instructions for a total of 5394 instructions. This large scale test will take care of many corner cases and stress the design. It was also important to find a program that could be modified to run on a dual core processor.

The pipelined and multicore designs have their own advantages and disadvantages. A multicore processor is able to take advantage of parallelism and split workloads. Unfortunately, many programs have not been optimized to take advantage of this. Therefore, a pipelined design would be better suited for purely sequential programs that have not been optimized to run on more than one core. We expect mergesort to run about 2x faster on the multicore design. Also between the pipeline designs, we expect the design with caches to be much faster.

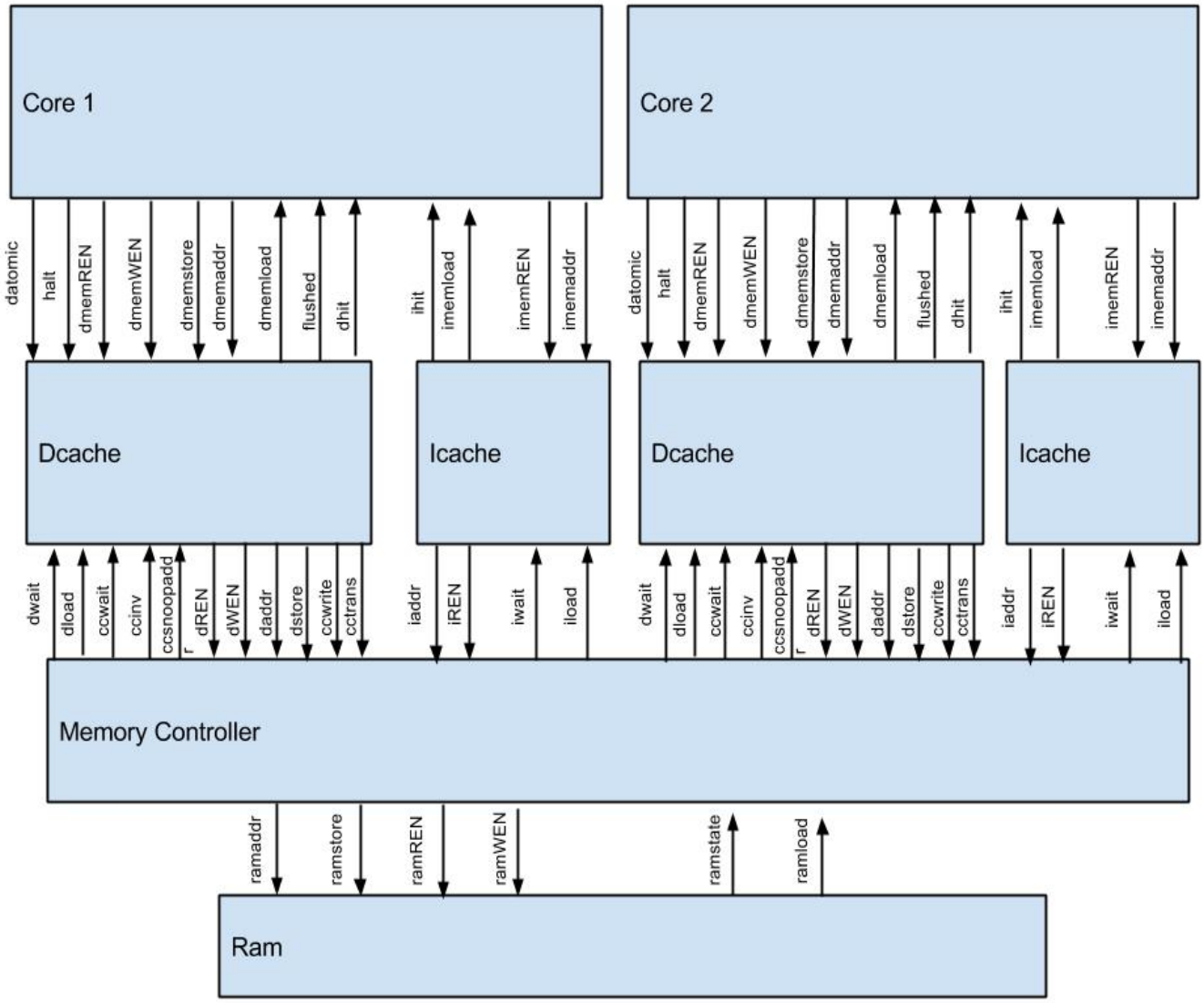
Processor Design - Single core datapath



Memory control state diagram with cache coherency

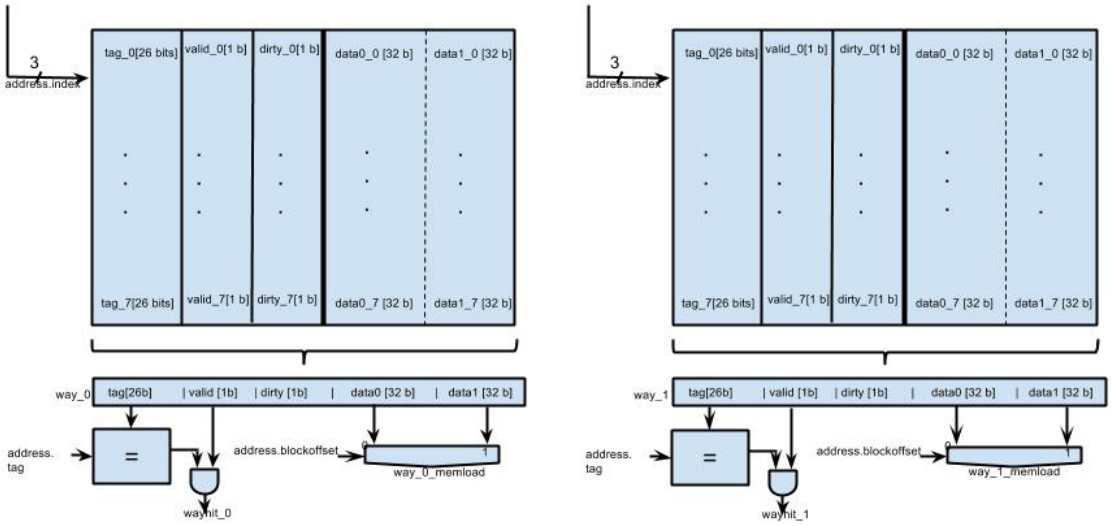


Multicore cache hierarchy block diagram

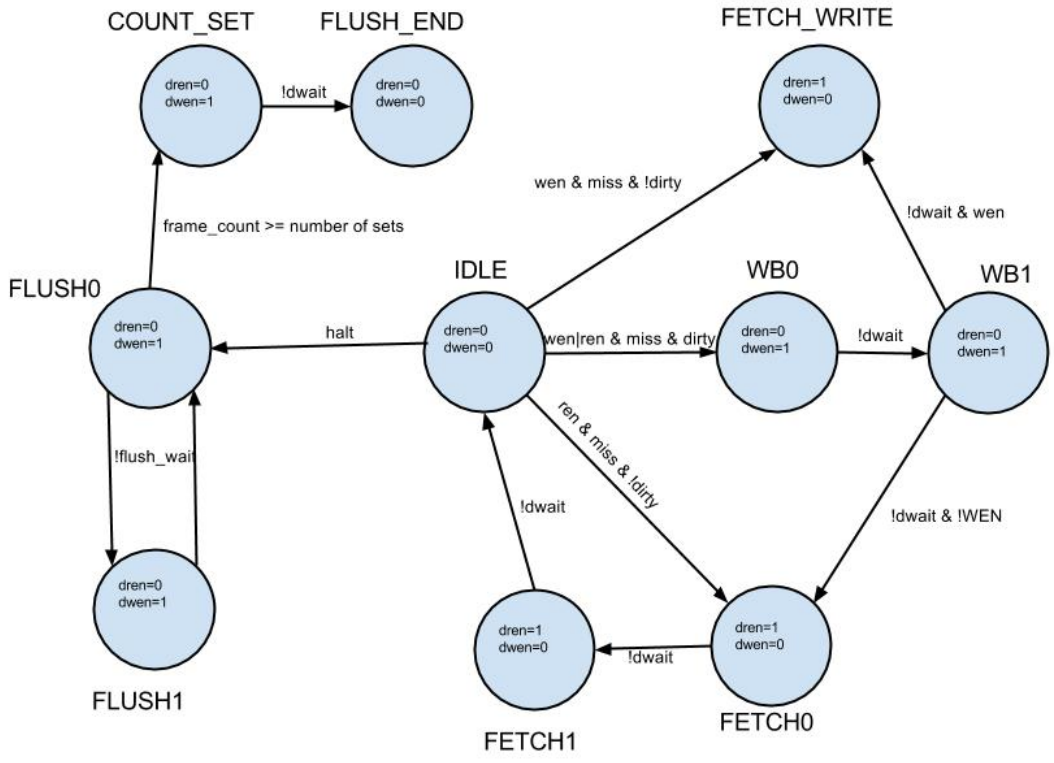


dcache

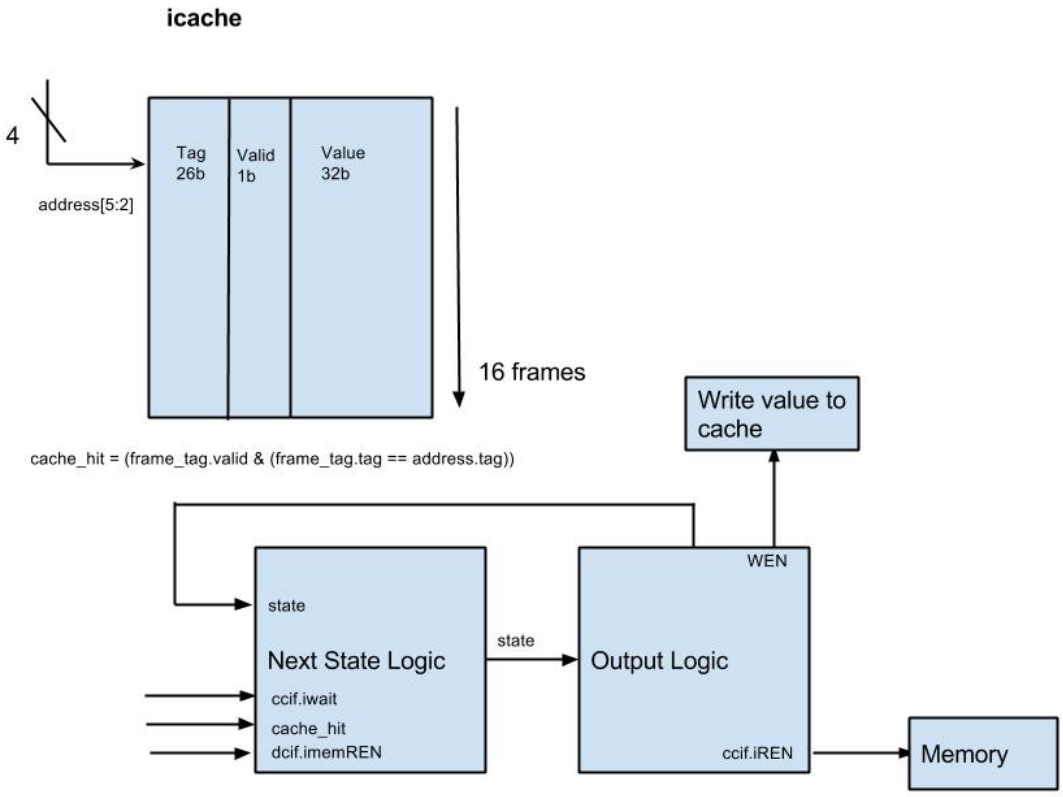
address (32 bit word)
address[31:6] => address.tag
address[5:3] => address.index
address[2] => address.
blockoffset

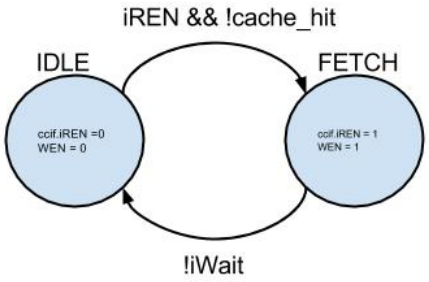


cache_hit = wayhit_0 | wayhit _1
dmemload = wayhit_0 ? way_0_memload : way_1_memload



icache block diagram and state diagram





Results

RAM latency was set to 8 clock cycles for the following test results.

Multicore: +Pipeline +Cache +2 cores
Pipeline cache: +Pipeline +Cache
Pipeline no cache: +Pipeline

	Multicore	Pipeline Cache	Pipeline No Cache
Max Frequency (Mhz)	27.77	27.77	50
Latency (ns)	180	180	100
IPC	1/(14205/5414) = 0.381	1/(18119/5399) = 0.298	1/(76481/5399) = 0.071
MIPS	10.58	8.28	3.55
Total logic elements	15,271 / 114,480 (13%)	7,610 / 114,480 (7 %)	3,448 / 114,480 (3%)
Total registers	8073	4118	3081
Total Pins	102 / 529 (19%)	102 / 529 (19%)	102 / 529 (19%)
Speedup vs Pipeline No Cache	10.58/3.55 = 2.98	8.28/3.55 = 2.33	1
Speedup vs Pipeline Cache	10.58/8.28 = 1.28	1	3.55/8.28 = 0.42

Conclusion

The maximum synthesizable frequency of the processor design of pipeline with no cache was higher than the multicore and pipeline with cache as we would expect due to the extra hardware logic added in between the MEM and WB pipeline stages increasing the critical path time. Due to the extra hardware for the cache we can observe the increase of total logic elements, registers. It doesn't come as a surprise that in multicore processor design there is double the amount of total logic elements and registers of pipeline cache design because we have two cores of pipeline cache in multicore.

We use MIPS as the performance metric because we are testing on the same type of program across the different processor designs. As we can see in the results above processor designs with caching enabled has a significant performance edge compared to the processor design with no cache. Although this performance improvement can only be seen when testing the design under programs that has temporal and spatial locality, which we can expect it to be the common case for most of useful programs. As we expected the multicore design was faster than the pipeline with cache processor design because the multicore design is able to exploit the parallelism of mergesort algorithm. We may have been able to attain close to the 2x theoretical speedup by duplicating cache tag on to a snoop tag store so to reduce the cycles stalled due to structural hazard during snooping.

In conclusion we can say that processor designs with cache has higher performance than a design with out cache. As well as properly parallelized algorithms would perform better on a multicore design than their counterpart single threaded algorithms perform on just a singlecore processor design.