CO224 - COMPUTER ARCHITECTURE

REPORT ON THE COMPARISON OF PERFORMANCE BETWEEN CACHED AND CACHE-LESS IMPLEMENTATION OF SINGLE CYCLE PROCESSOR

DE SILVA K.R
E/16/068

In part 01 of lab 06, a memory module was implemented for the single cycle processor without using a cache memory. Because of nature this main memory module, it requires about 5 clock cycles for both read and write access. This is a considerable amount of time compared to the total runtime of the CPU. To reduce this latency and make CPU efficient, a cache memory module was introduced to the system in part 02 of lab 06.

Cache memory stores recently accessed data based on two locality principles. Because of this nature of Cache memory, it requires less time to access and get the required data. However, if the requested data word is not in the current cache table, then cache memory needs to be updated and this requires accessing the main memory. Cache memory is associated with a tag, valid bit and a dirty bit that are used to determine the availability of a data word in the cache table. Therefore when cache needs to access main memory to fetch or update any data block, all these additional signals too have to be updated. This whole operation requires about 40 clock cycles since cache and main memory deal with data blocks, not data words.

```
loadi 0 0xA
loadi 1 0x14
loadi 2 0x32
add 3 0 1
sub 4 2 1
j 0x0
swi 4 0x03
swd 3 2
lwi 5 0x03
lwd 6 2
```

The instructions mentioned above are the set of instructions used for the testing purpose of both implementation cached and cache-less. This instruction set takes about 248 time units (figure 01) to execute in the cache-less environment. But in the cached environment, it takes about 430 ns time units (figure 02) . It appears that the cached environment takes longer time to execute. That is because cache memory needs to access main memory to fetch/write data into the memory. But this happens only when the requested data word is not in the current cache table, i.e tags do not match. Then, the required memory block is fetched/written from the memory. Because of the locality principles, once a block is fetched from the memory to the cache table, the CPU does not get stalled when it requests data word near spatial locality or when repeatedly accessed near locations, resulting in increased performance by a considerable amount.
Whereas in cache-less implementation, everytime CPU requests a word to be fetched or written to the memory, the CPU is forcefully stalled by the memory in order to complete fetching/writing operation on memory, causing a decrease in performance by a considerable amount.

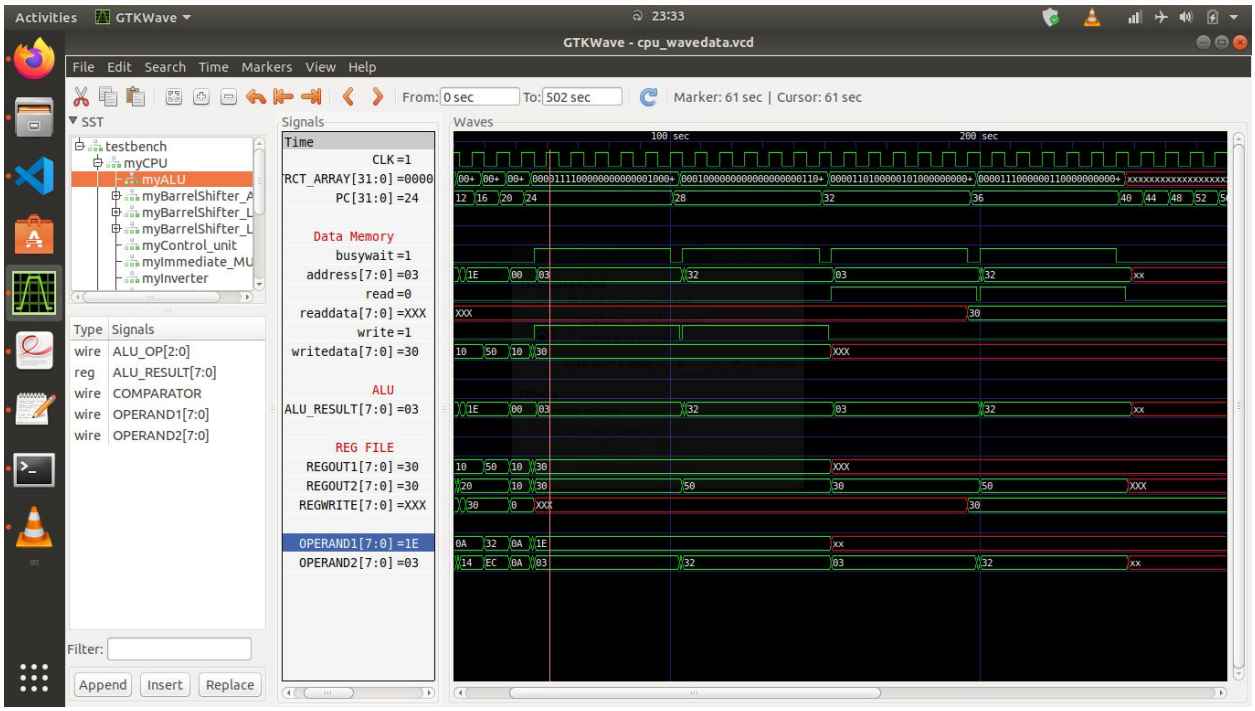

Figure 01 - Timing diagram with cache-less implementation

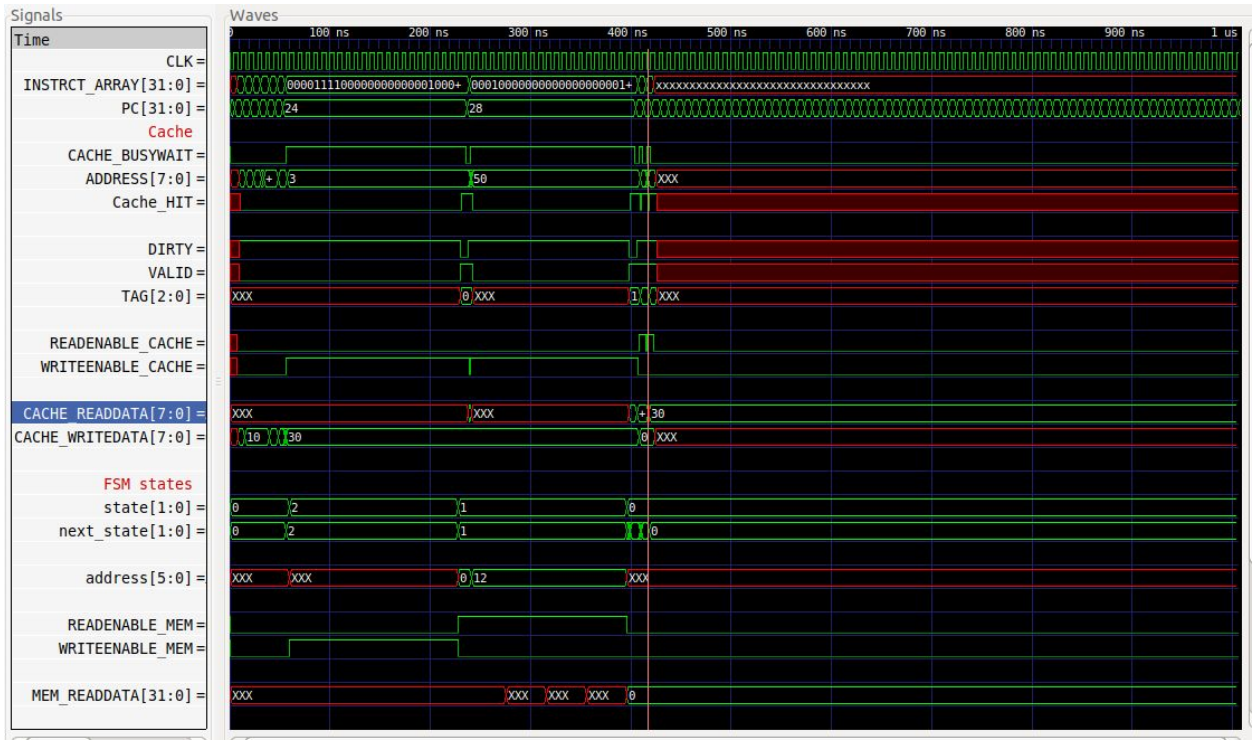

Figure 02 - Timing diagram with cached implementation