CO224 –LAB 06
PART 2 – DATA CACHE

GROUP 09
DISSANAYAKE D.M.D.R (E/17/072)
DEVINDI G.A.I (E/17/058)

CACHE IMPLEMENTATION

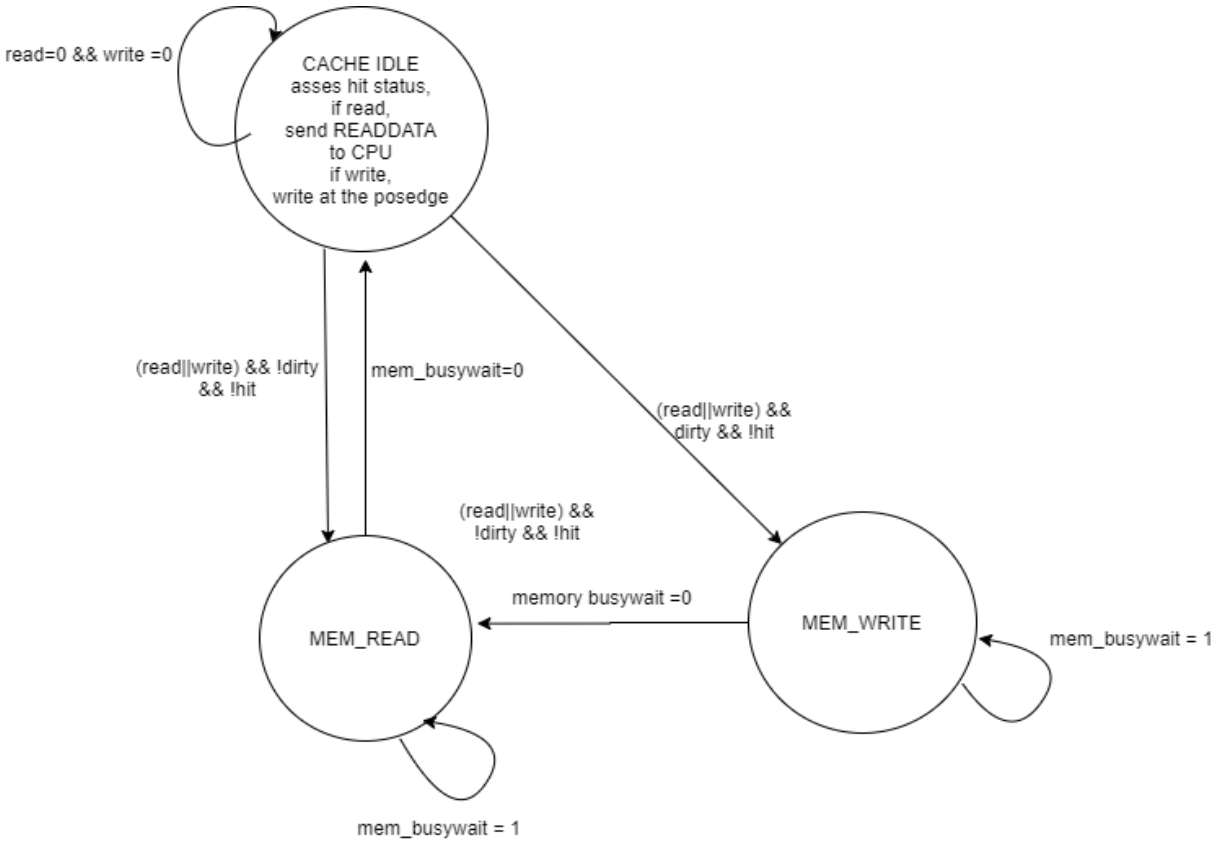


Figure 1 : Finite State Machine of the Cache

Three states were used in the cache to perform the memory accessing and writing.

1. **IDLE** :-
    - When read & write signals coming from the CPU's control unit are 0, the cache will be in the idle state.
    - When read or write signals are set to high, the address will be split to tag, index and offset and the cache block corresponding to the index will be extracted after 1 time unit delay.
    - Then, the hit signal is asserted after 0.9 time unit delay.
    - Depending on the hit, read, write, dirty bit signals the following steps will happen.

i. hit =1, read=0,write=1 - the data given in writedata will be written to the relevant cache block depending on the address at the next positive edge of the clock (write hit)

ii. hit=1, read=1,write=0 - the data present in the cache at the given address will be directed to read data within one clock cycle (read hit)

iii. hit=0, (read or write =1), dirty bit =0 - Change the state to MEM_READ

iv. hit=0, (read or write =1), dirty bit =1 - Change the state to MEM_WRITE

2. **MEM_READ** :-

- Read the relevant data blocks from the memory according to the tag and index of the address. (must consume 20 clock cycles but due to a race condition occurred as a result of setting the memory read signal to high at the positive clock edge, consumes 21 clock cycles to read a whole block(4 bytes) from memory.)

- After the reading is completed, the block read from memory must be written to the relevant cache block which will have a latency of another 1 time unit.

- Once the memory reading is completed, the cache state will go back to the IDLE state, in which the hit status is asserted again and the original read or write operation is performed after the hit status is asserted.

3. **MEM_WRITE** :-

- Since the dirty bit is 1, the data block in the cache must be written to the memory before replacing the block with another set of data.(must consume 20 clock cycles but due a race condition occurred as a result of setting the memory read signal to high at the positive clock edge, consumes 21 clock cycles to read a whole block(4 bytes) from memory.)

- Once the data block is written to the memory and the memory busywait signal is asserted to 0, the state will be directed to MEM_READ.

COMPARING THE OUTCOME WITH THE CACHE-LESS IMPLEMENTATION

The outcome will be discussed under 6 cases.

1. CASE 1

Cache miss occurs when write =1 and read = 0 (write miss) and the dirty bit =0

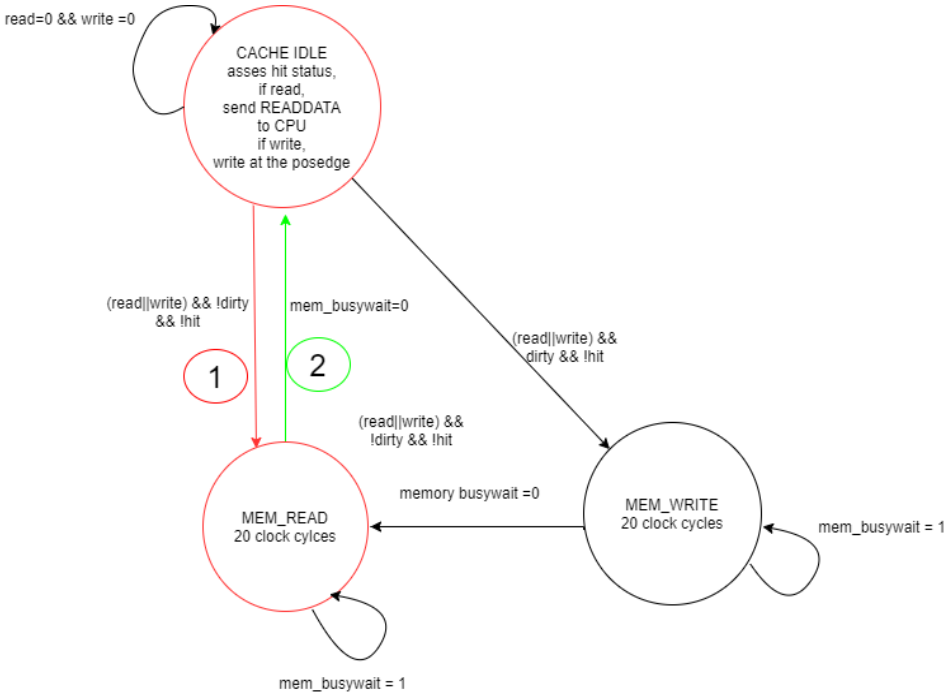


Figure 2: Write miss state transition when dirty bit is 0

- Since the dirty bit is 0, a write back to the memory won't happen.
- But, the memory block related to the write address must be fetched from the memory to the cache.
- This fetching operation consumes 20 clock cycles because 4 bytes are fetched from memory (5 cycles per byte).
- After the relevant block of memory is fetched, the hit signal is asserted at the 21st clock cycle, and the data can be written to the relevant cache address at the next positive clock edge after 1 time unit delay.
- Due to the race condition an extra clock cycle is consumed for reading, therefore the total clock cycles consumed after the instruction fetching is 22.

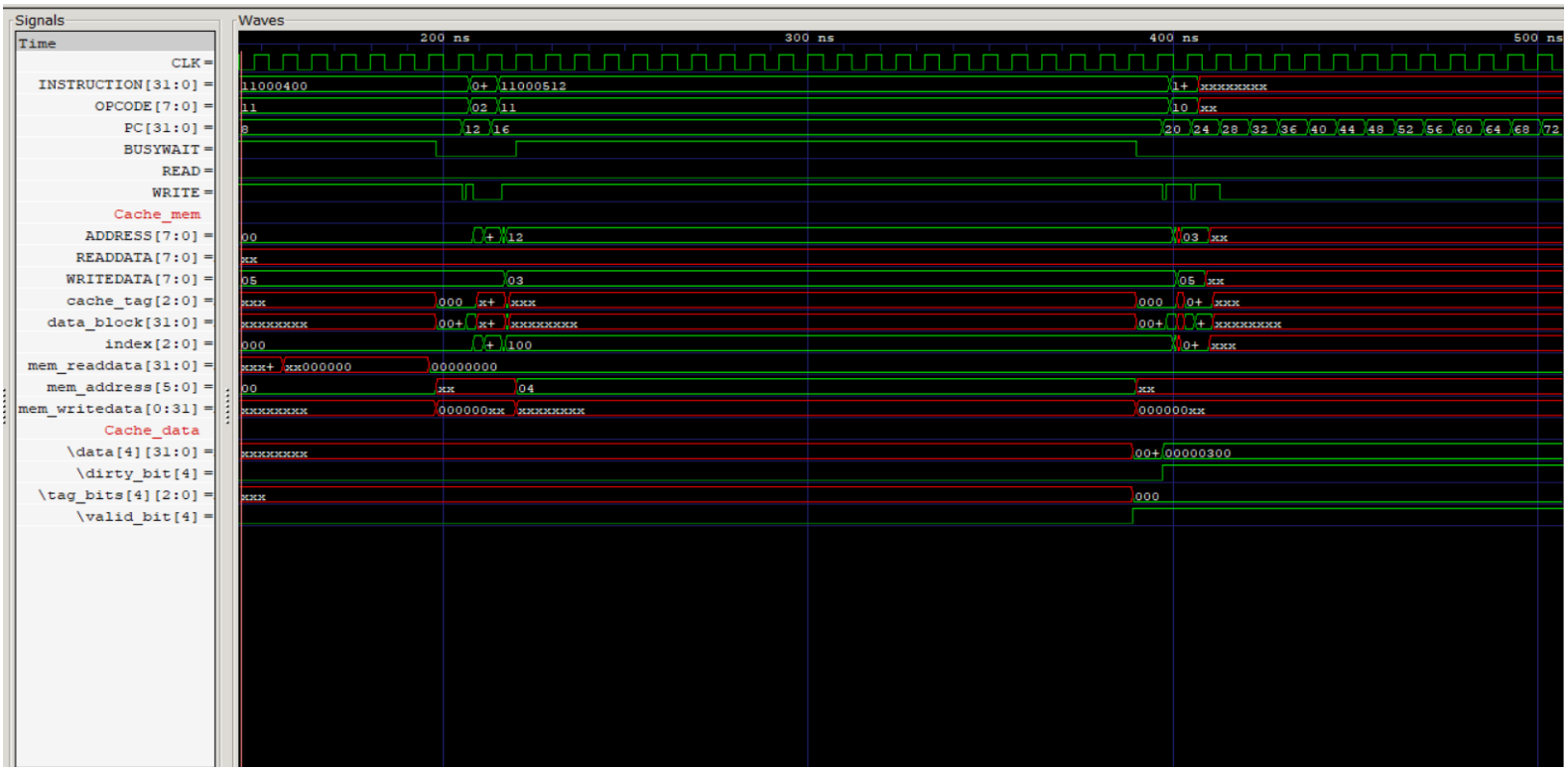
Cache-less implementation - in the cache less implementation, writing to a memory address will always consume 5 clock cycles. In this case only 1 byte is considered, therefore, total operation will conclude after 5 clock cycles.

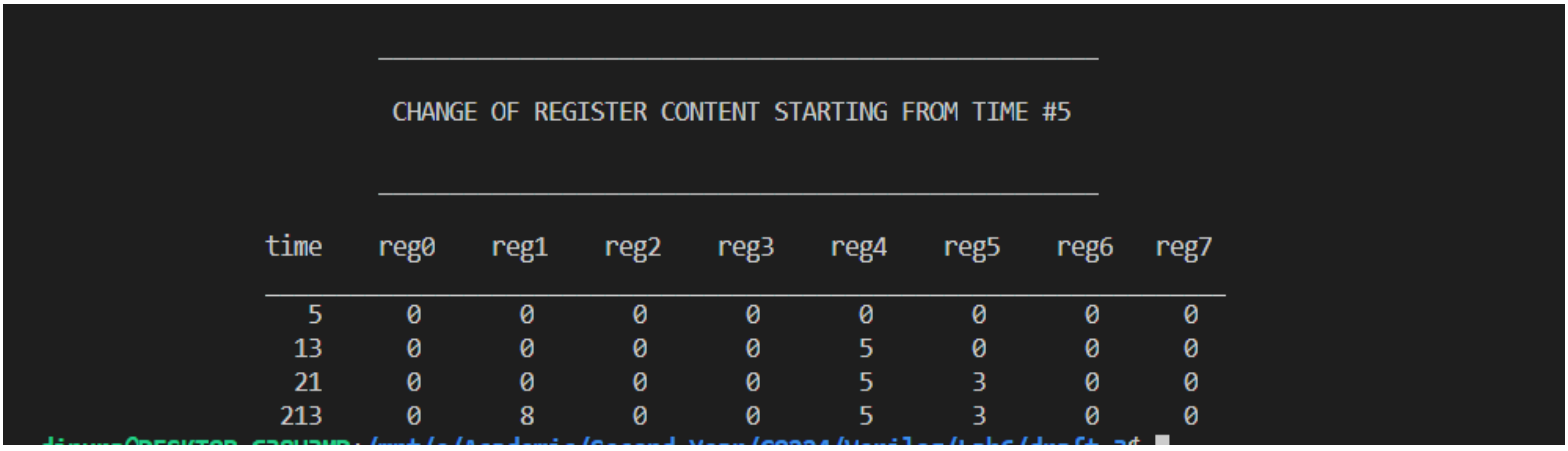
GTKWave timing wave form for an example for CASE 1

Instruction set

loadi 4 0x05
loadi 5 0x03
swi 4 0x00
add 1 4 5
swi 5 0x12 //cache miss when write=1, dirty = 0
swd 4 5

Cached Implementation





CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
213	0	8	0	0	5	3	0	0

Figure 3: GTKWave Timing diagram and register file outputs for cached implementation of CASE 1

Cache less implementation

CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
213	0	8	0	0	5	3	0	0

Figure 4: GTKWave Timing diagram and register file outputs for cache less implementation of CASE 1

2. CASE 2

Cache miss occurs when write =1 and read = 0 (write miss) and the dirty bit=1(write back)

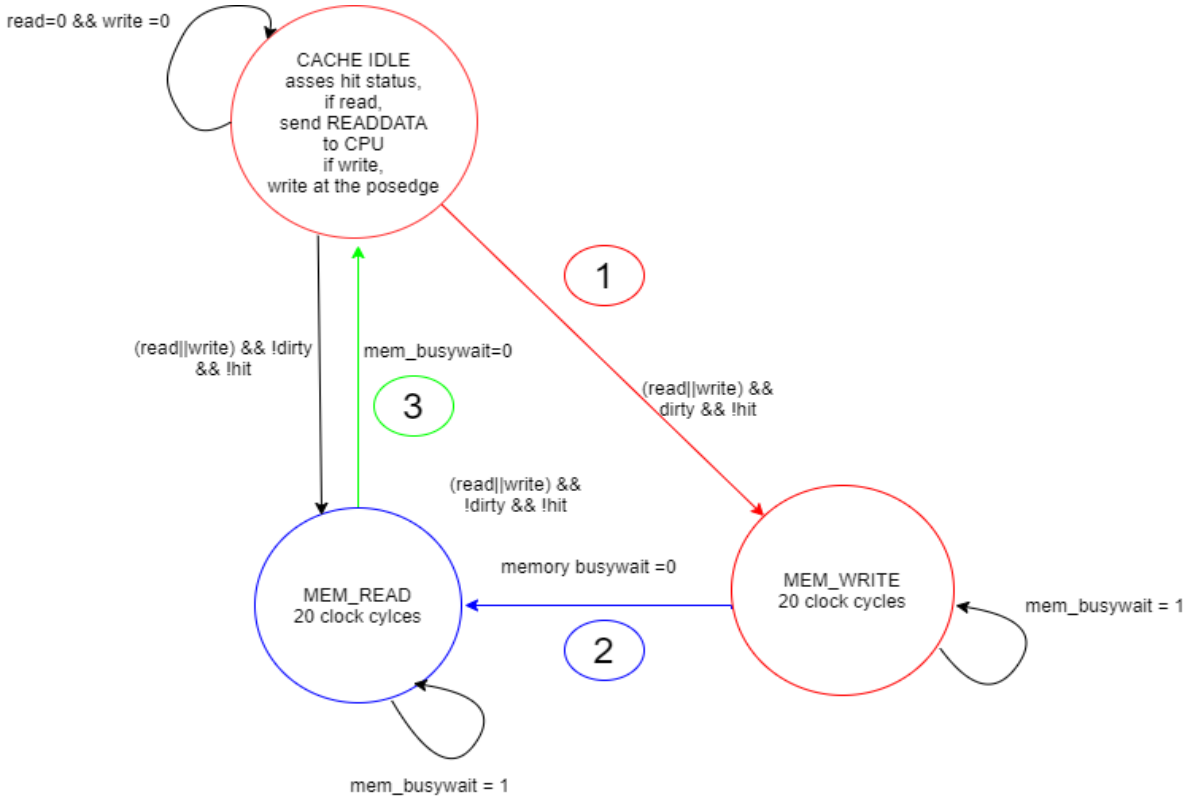


Figure 5: Write miss state transition when dirty bit is 1

- Since the dirty bit is 1, a write back to the memory will happen which will consume 20 clock cycles (5 clock cycles per byte).
- After the data in the cache block is written to the memory, the memory block related to the write address must be fetched from the memory to the cache.
- This fetching operation consumes 20 clock cycles because 4 bytes are fetched from memory (5 cycles per byte).
- After the relevant block of memory is fetched, the hit signal is asserted at the 41st clock cycle, and the data can be written to the relevant cache address at the next positive clock edge after 1 time unit delay.
- Due to the race condition an extra clock cycle is consumed for writing to memory and reading from memory. Therefore, the total clock cycles consumed after the instruction fetching is 42.

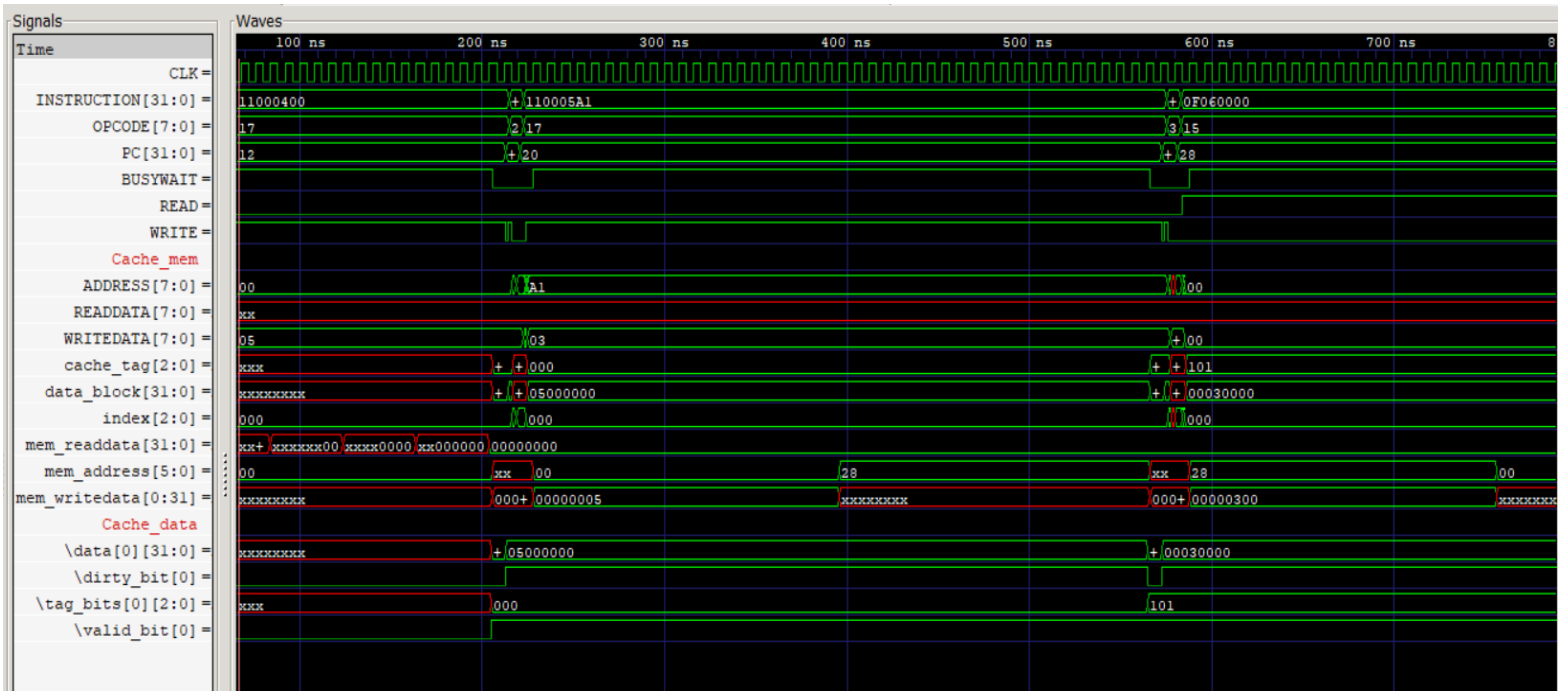
Cache-less implementation - in the cache less implementation, writing to a memory address will always consume 5 clock cycles. In this case only 1 byte is considered, therefore, total operation will conclude after 5 clock cycles.

GTKWave timing wave form for an example for CASE 2

Instruction set

loadi 4 0x05
loadi 5 0x03
loadi 7 0x07
swi 4 0x00
add 4 4 5
swi 5 0xA1 //Cache miss when write = 1 and dirty = 1
sub 5 4 5
lwi 6 0x00

Cached Implementation

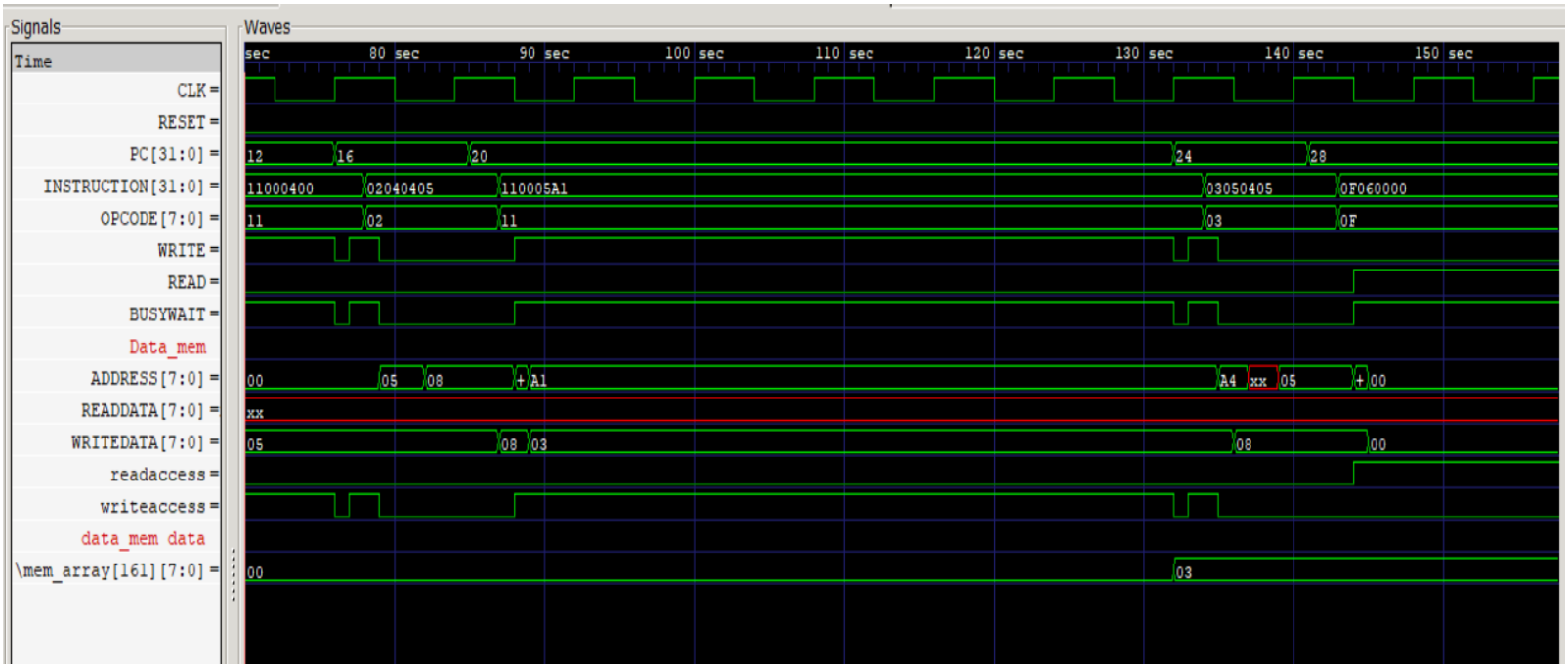


CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
221	0	0	0	0	8	3	0	7
581	0	0	0	0	8	5	0	7
933	0	0	0	0	8	5	5	7

Figure 6: GTKWave Timing diagram and register file outputs for cached implementation of CASE 2

Cache less Implementation



CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
85	0	0	0	0	8	3	0	7
141	0	0	0	0	8	5	0	7
189	0	0	0	0	8	5	5	7

Figure 7: GTKWave Timing diagram and register file outputs for cache less implementation of CASE 2

3. CASE 3

Cache hit when write = 1 and read = 0 (write hit)

- No state transitions will occur from the IDLE state because the hit signal will be asserted 2 time units after the ADDRESS is received from the CPU.
- Since the hit signal is high, the data available in WRITEDATA will be written to the memory at the next positive edge of the clock after 1 time unit delay.
- As there's no need to stall the CPU to write data to the cache, the CPU can move on to the next instruction at the next positive clock edge in which data will be written to the cache.
- Therefore, if a write hit is encountered, writing to the cache can be completed within 1 clock cycle.

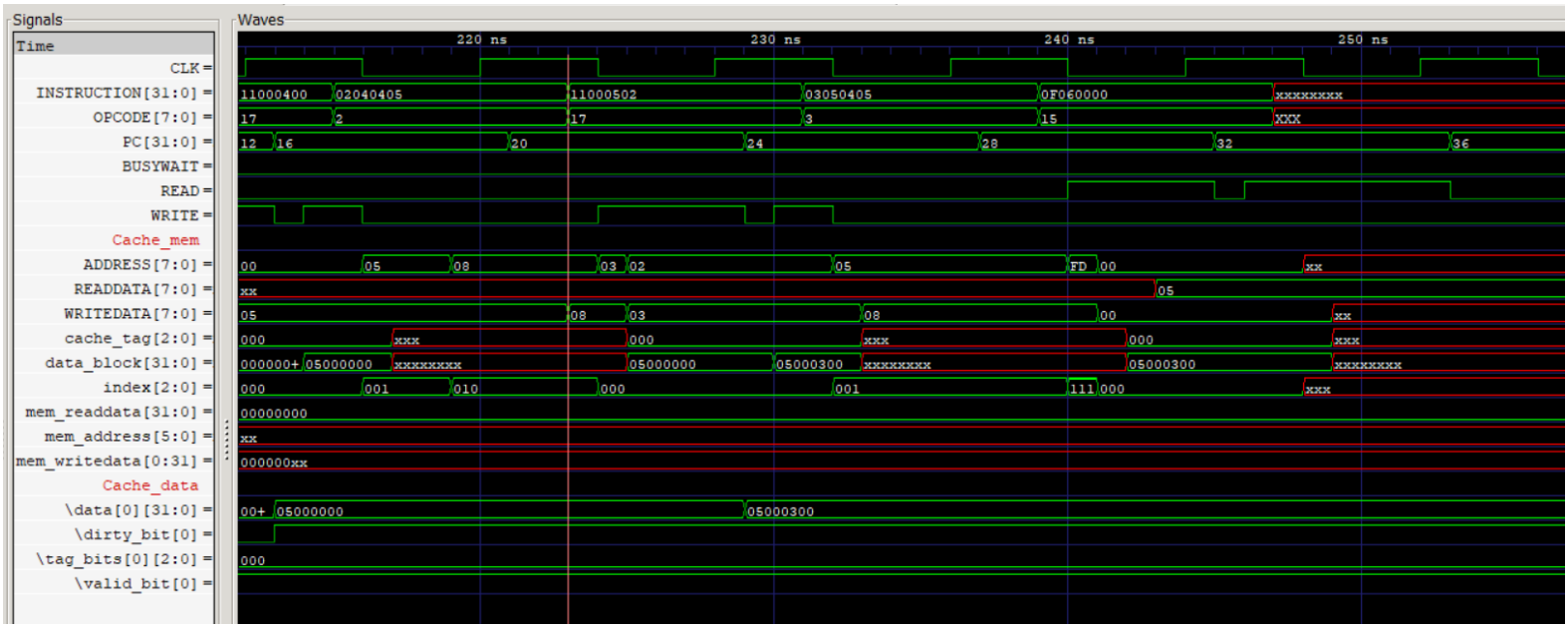
Cache-less implementation - in the cache less implementation, writing to a memory address will always consume 5 clock cycles. In this case only 1 byte is considered, therefore, total operation will conclude after 5 clock cycles.

GTKWave timing wave form for an example for CASE 3

Instruction set

```
loadi 4 0x05
loadi 5 0x03
loadi 7 0x07
swi 4 0x00
add 4 4 5
swi 5 0x02        //Cache hit when write = 1
sub 5 4 5
lwi 6 0x00
```

Cached Implementation

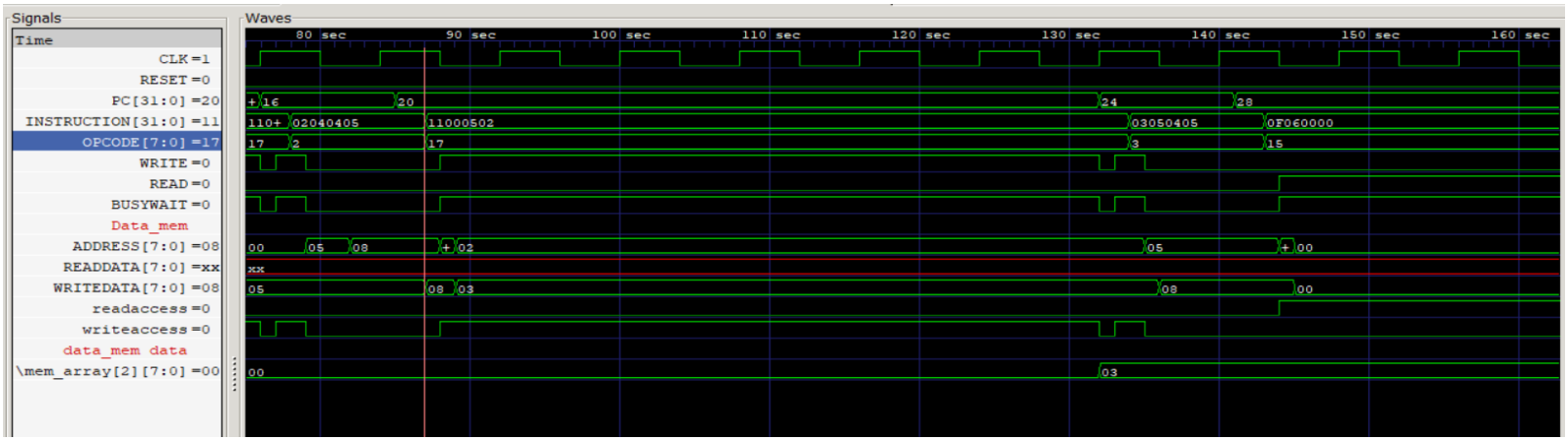


CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
221	0	0	0	0	8	3	0	7
237	0	0	0	0	8	5	0	7
245	0	0	0	0	8	5	5	7

Figure 8: GTKWave Timing diagram and register file outputs for cached implementation of CASE 3

Cache less Implementation



CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
85	0	0	0	0	8	3	0	7
141	0	0	0	0	8	5	0	7
189	0	0	0	0	8	5	5	7

Figure 9: GTKWave Timing diagram and register file outputs for cache less implementation of CASE 3

4. CASE 4

Cache miss occurs when read =1 and write = 0 (read miss) and the dirty bit =0

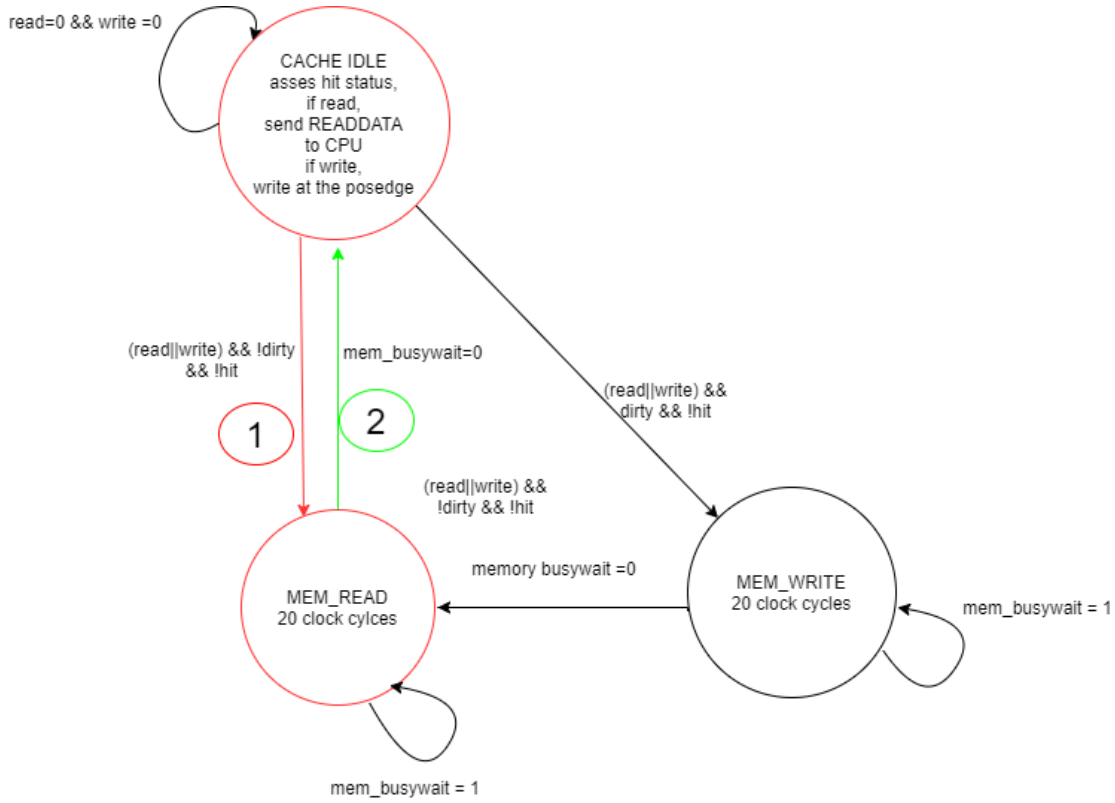


Figure 10: Read miss state transition when dirty bit is 0

- Since the dirty bit is 0, a write back to the memory won't happen.
- But, the memory block related to the read address must be fetched from the memory to the cache.
- This fetching operation consumes 20 clock cycles because 4 bytes are fetched from memory (5 cycles per byte).
- After the relevant block of memory is fetched, the hit signal is asserted at the 21st clock cycle, and the data can be read from the relevant cache address and sent to the CPU which will be then written to a register at the next positive edge of the clock.
- Due to the race condition, an extra clock cycle is consumed for memory reading, therefore the total clock cycles consumed after the instruction fetching is 22.

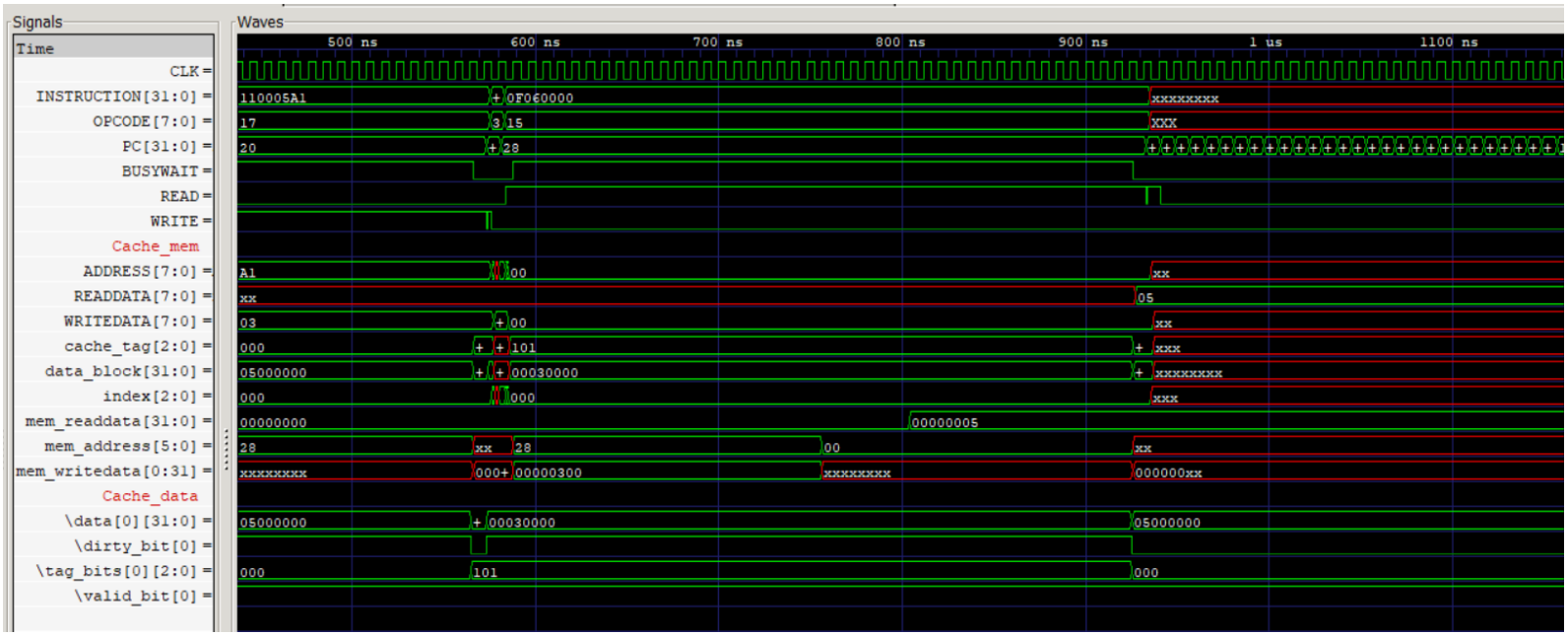
Cache-less implementation - in the cache less implementation, reading from a memory address will always consume 5 clock cycles. In this case only 1 byte is considered, therefore, total operation will conclude after 5 clock cycles.

GTKWave timing wave form for an example for CASE 4

Instruction set

loadi 4 0x05
loadi 5 0x03
loadi 7 0x07
swi 4 0x00
add 4 4 5
swi 5 0xA1
sub 5 4 5
lwi 6 0x00 //Read miss when dirty bit = 0;

Cached Implementation

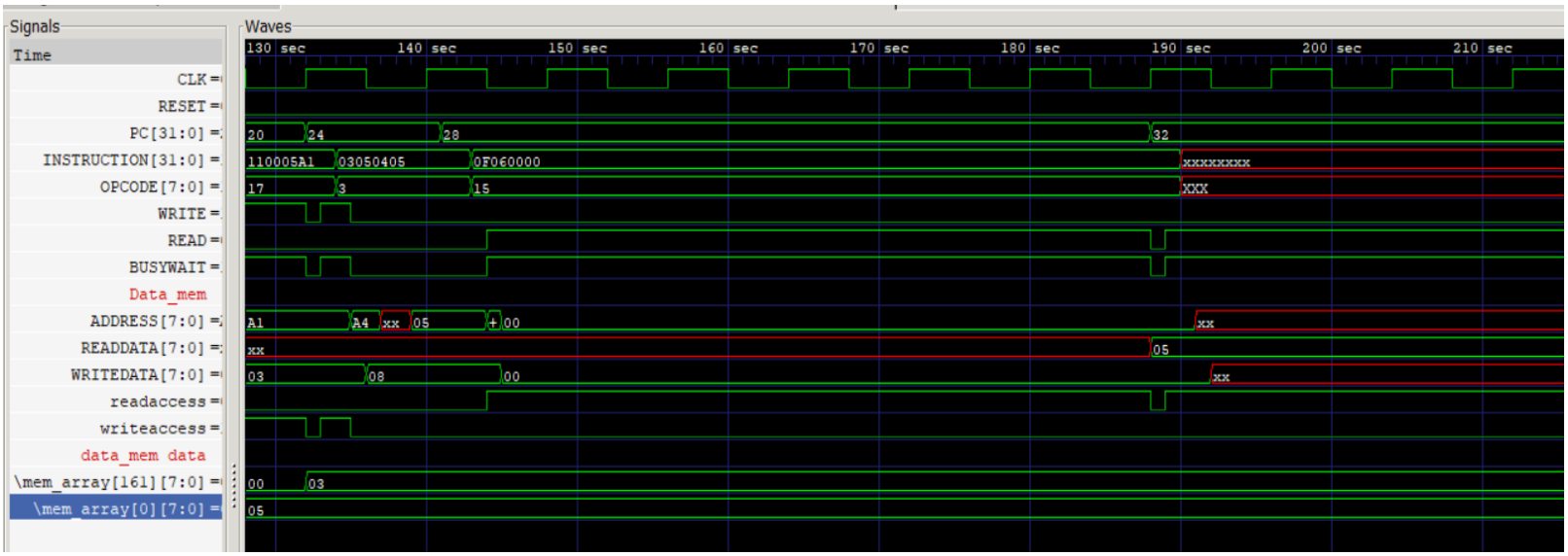


CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
221	0	0	0	0	8	3	0	7
581	0	0	0	0	8	5	0	7
933	0	0	0	0	8	5	5	7

Figure 11: GTKWave Timing diagram and register file outputs for cached implementation of CASE 4

Cache less Implementation



CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
85	0	0	0	0	8	3	0	7
141	0	0	0	0	8	5	0	7
189	0	0	0	0	8	5	5	7

Figure 12: GTKWave Timing diagram and register file outputs for cache less implementation of CASE 4

5. CASE 5

Cache miss occurs when read =1 and write = 0 (read miss) and the dirty bit =1 write back)

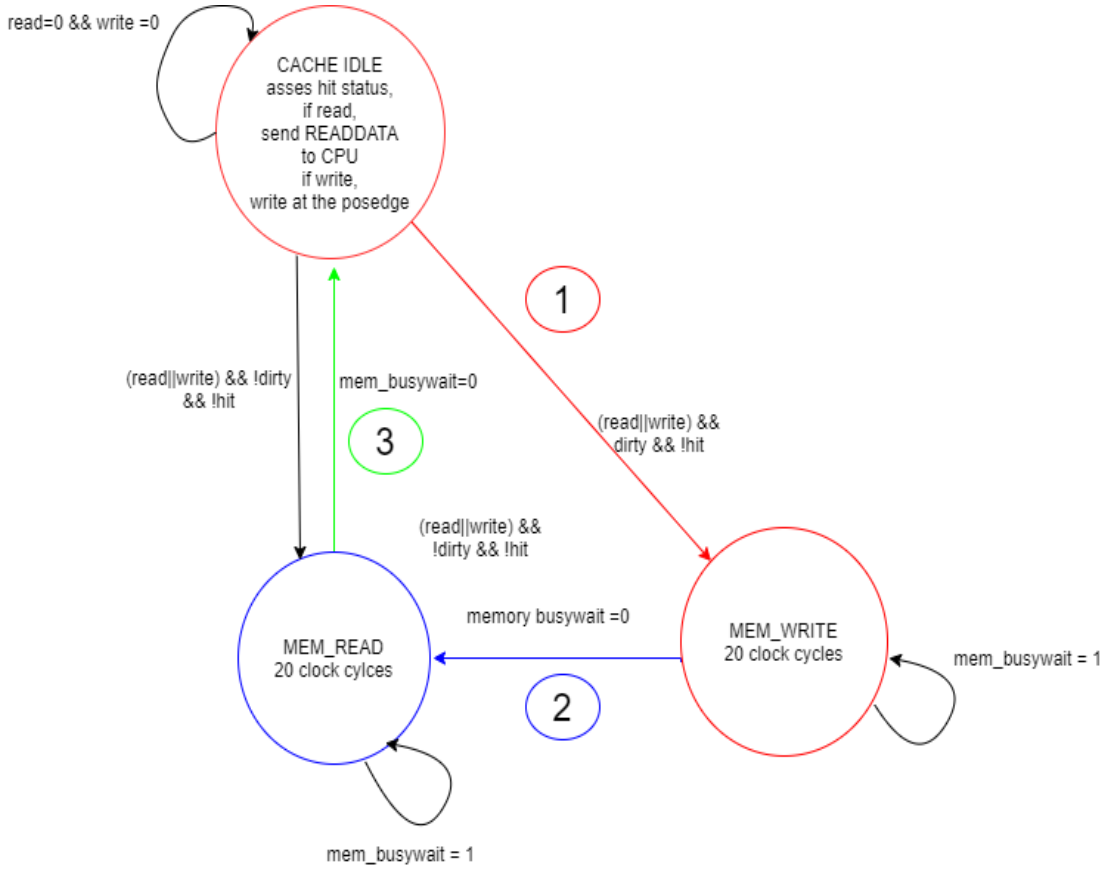


Figure 13: Read miss state transition when dirty bit is 1

- Since the dirty bit is 1, a write back to the memory will happen which will consume 20 clock cycles (5 clock cycles per byte).
- After the data in the cache block is written to the memory, the memory block related to the read address must be fetched from the memory to the cache.
- This fetching operation consumes 20 clock cycles because 4 bytes are fetched from memory (5 cycles per byte).
- After the relevant block of memory is fetched, the hit signal is asserted at the 41st clock cycle, and the data can be read from the relevant cache address and directed to the CPU which will be then written to a register at the next positive edge of the clock.
- Due to the race condition an extra clock cycle is consumed for writing to memory and reading from memory. Therefore, the total clock cycles consumed after the instruction fetching is 42.

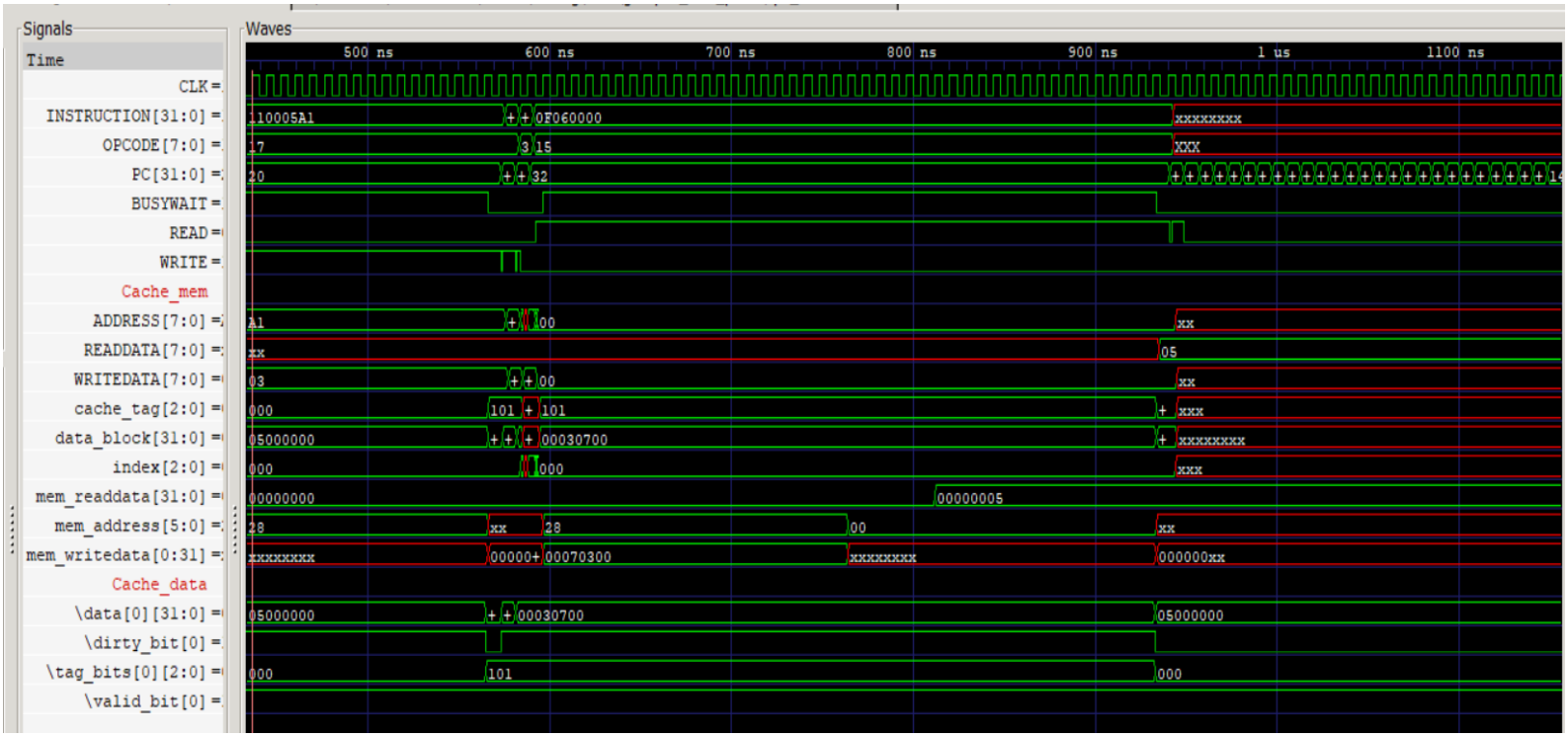
Cache-less implementation - in the cache less implementation, reading from a memory address will always consume 5 clock cycles. In this case only 1 byte is considered, therefore, total operation will conclude after 5 clock cycles.

GTKWave timing wave form for an example for CASE 5

Instruction set

loadi 4 0x05
loadi 5 0x03
loadi 7 0x07
swi 4 0x00
add 4 4 5
swi 5 0xA1
swi 7 0xA2
sub 5 4 5
lwi 6 0x00 //Cache miss when dirty bit = 1;

Cached Implementation

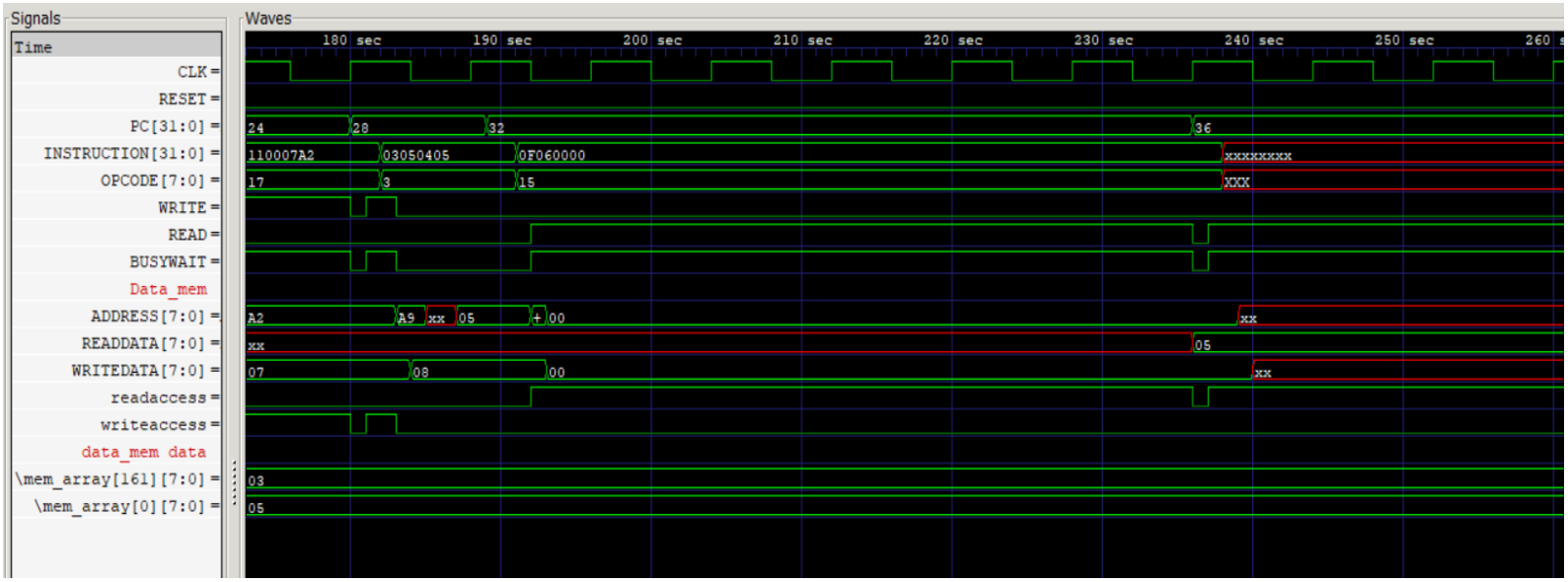


CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
221	0	0	0	0	8	3	0	7
589	0	0	0	0	8	5	0	7
941	0	0	0	0	8	5	5	7

Figure 14: GTKWave Timing diagram and register file outputs for cached implementation of CASE 5

Cache less Implementation



CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
85	0	0	0	0	8	3	0	7
189	0	0	0	0	8	5	0	7
237	0	0	0	0	8	5	5	7

Figure 15: GTKWave Timing diagram and register file outputs for cache less implementation of CASE 5

6. CASE 6

Cache hit when read =1 and write = 0 (read hit)

- No state transitions will occur from the IDLE state because the hit signal will be asserted 2 time units after the ADDRESS is received from the CPU.
- Since the hit signal is high, the data available in READDATA will be directed to the CPU and written to the register at the next positive edge of the clock.
- Therefore, if a read hit is encountered, reading from the cache can be completed within 1 clock cycle.

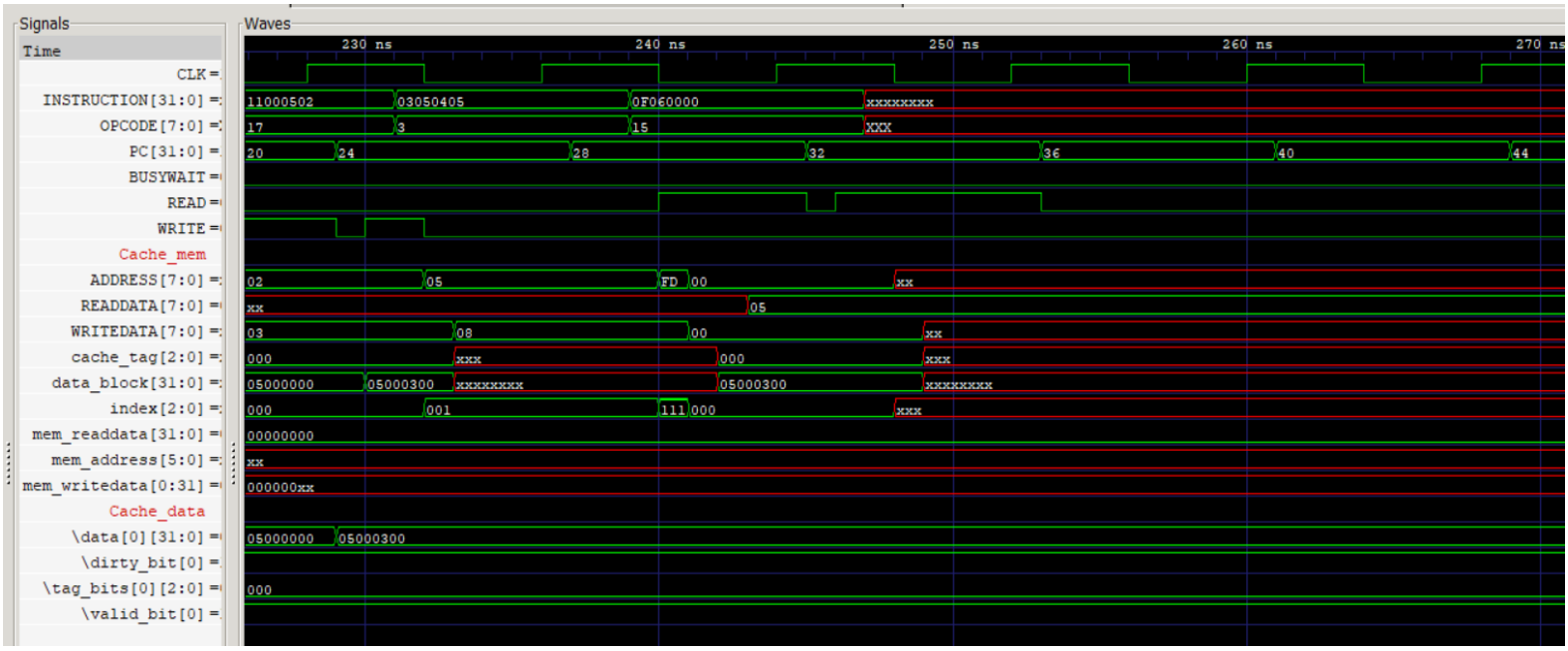
Cache-less implementation - in the cache less implementation, reading from a memory address will always consume 5 clock cycles. In this case only 1 byte is considered, therefore, total operation will conclude after 5 clock cycles.

GTKWave timing wave form for an example for CASE 6

Instruction set

```
loadi 4 0x05
loadi 5 0x03
loadi 7 0x07
swi 4 0x00
add 4 4 5
swi 5 0x02
sub 5 4 5
lwi 6 0x00    //Read hit
```

Cached Implementation

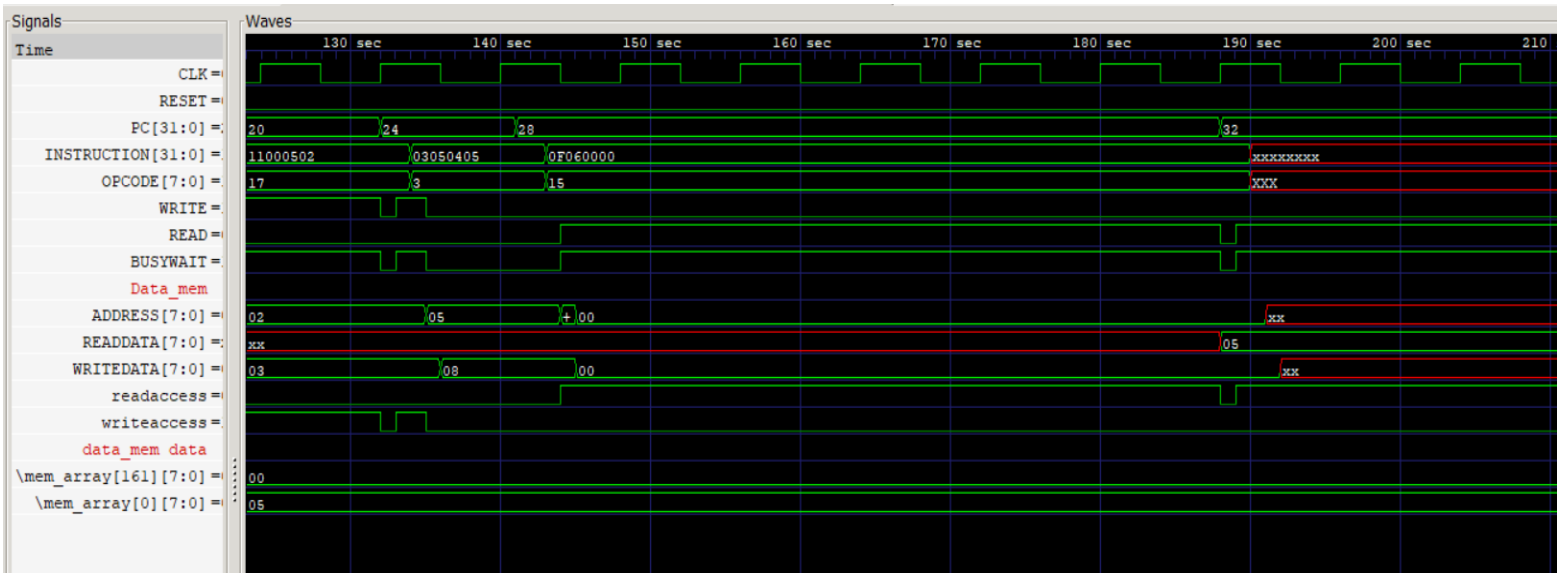


CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
221	0	0	0	0	8	3	0	7
237	0	0	0	0	8	5	0	7
245	0	0	0	0	8	5	5	7

Figure 16: GTKWave Timing diagram and register file outputs for cached implementation of CASE 6

Cache less Implementation



CHANGE OF REGISTER CONTENT STARTING FROM TIME #5

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
85	0	0	0	0	8	3	0	7
141	0	0	0	0	8	5	0	7
189	0	0	0	0	8	5	5	7

Figure 17: GTKWave Timing diagram and register file outputs for cache less implementation of CASE 6

CONCLUSION

The cache implementation is more efficient than the cache-less implementation when the cache encounters more hits than misses.

Since this cache is built considering the two principles of locality,

- recently accessed data will be available in the cache (temporal locality) and on the other hand
- data located closer to the recently accessed data will also be available in the cache since blocks of data are fetched from the memory (spatial locality).

Therefore, if the hit rate is high, implementation of the cache will drastically reduce the time consumed to execute a program because memory reading and writing can be completed within 1 clock cycle, which would otherwise always take 5 clock cycles to complete without a cache in between the data memory and the CPU.