

1: ALU

a FULL_ADDER

Description

To implement my FULL_ADDER, I connect two half_adders. The full_adder has three one bit inputs "A", "B", "Cin", and two one bit outputs "Sum" and "Carry". "Sum" is the result of two bits addition, and "Carry" is the carry out.

Waveform

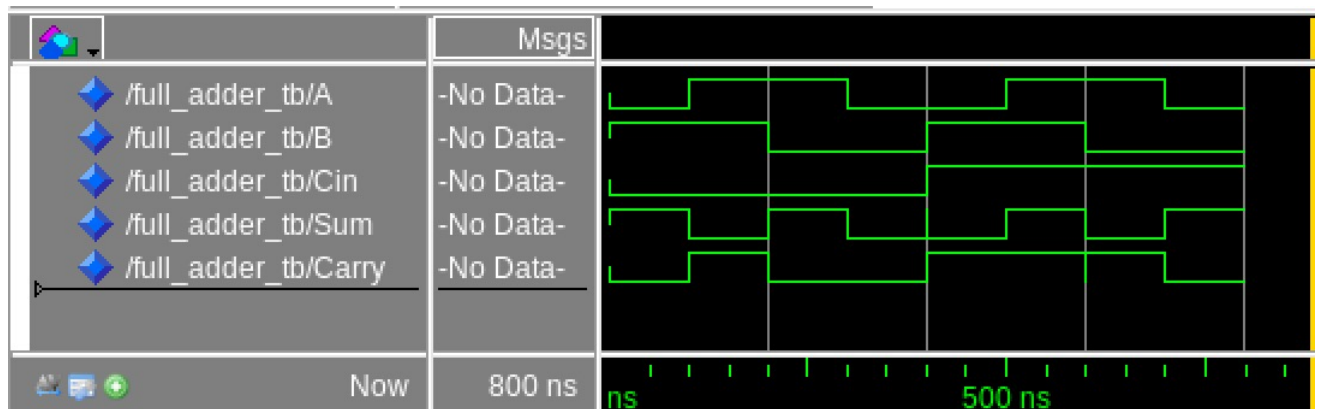


Figure 1: Simulation Waveform of AND2

| A | B | Cin | Sum | Carry |
|---|---|-----|-----|-------|
| 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |

Table 1: FULL_ADDER output table

From the above figure and table, we can see all the addition operations are working correctly.

b 64 bits adder

Description

To implement my 64 bits adder, I connect 64 full adders into a 64 bits carry ripple adder. The carry ripple adder has an one bit input "Cin", two 64 bits inputs "A" and "B", an one bit output "Cout", and one 64 bits output "Sum". "Sum" is the result of our addition, and "Cout" is the carry out.

Waveform

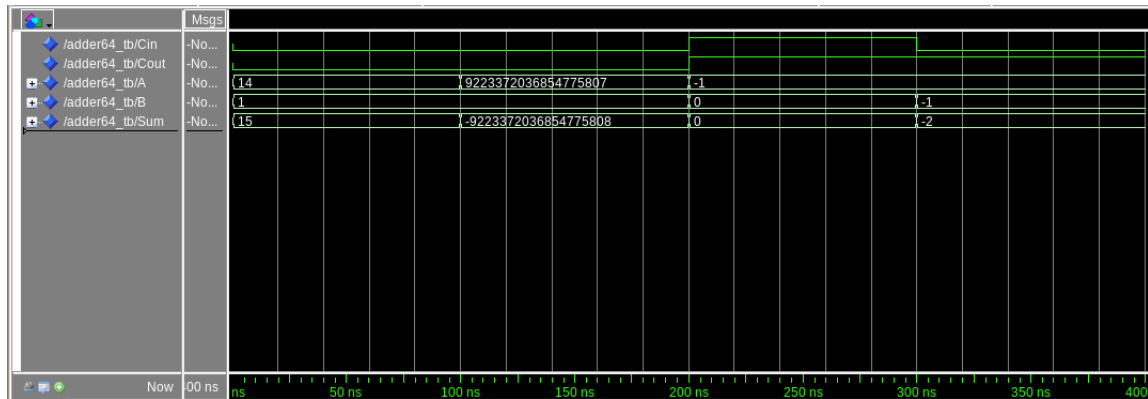


Figure 2: Simulation Waveform of 64bits carry ripple adder

In the waveform, I did 4 additions. 64 bits signals are shown in decimal radix. For those cases do not have overflow issue, all the results are correct. Although we do not care carry out in these cases, the carry out signals are all correct if we treat these signals as binary.

c ALU

Description

ALU runs different arithmetic logic operation according to the operation code. ALU has two 64 bits inputs, when 'operation' = "0000", it do "A and B". When 'operation' = "0001", it does "A or B". When 'operation' = "0010", it does signed addition "A + B". And I use my 64 bits carry ripple adder to implement this addition operation. When 'operation' = "0110", it does signed subtraction "A - B".

If 'operation' equals to other value, currently ALU does nothing.

'zero' is used to check the result, if result equals to '0', then 'zero' is set to '1', otherwise 'zero' is '0'. 'overflow' is '1' when overflow happens.

'AND' 'OR' Waveform

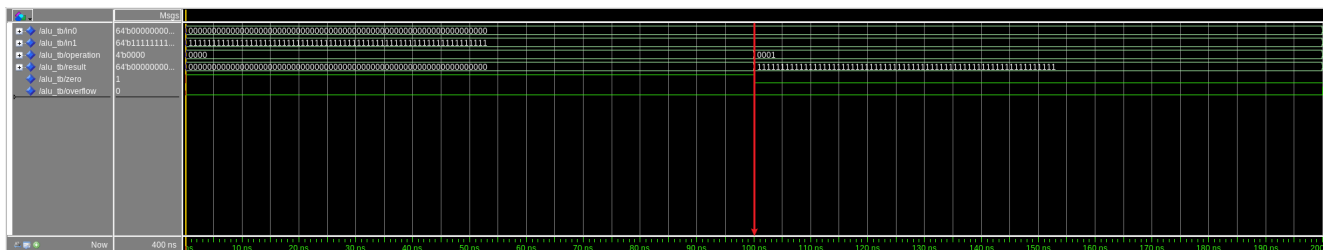


Figure 3: Simulation Waveform of 'AND' and 'OR'(include one zero case)

In the above waveform, I set A to 64 bits '0', B to 64 bits '1'. During 0 to 100 ns, the operation is "0000" which means "AND" function. The output of "A AND B" is 64 bits '0'. Since the output is zero, the "zero" is set to '1'. During 100 to 200 ns, the operation is "0001" which means "OR" function. The output of "A OR B" is 64 bits '1'.

'add' 'subtract' Waveform

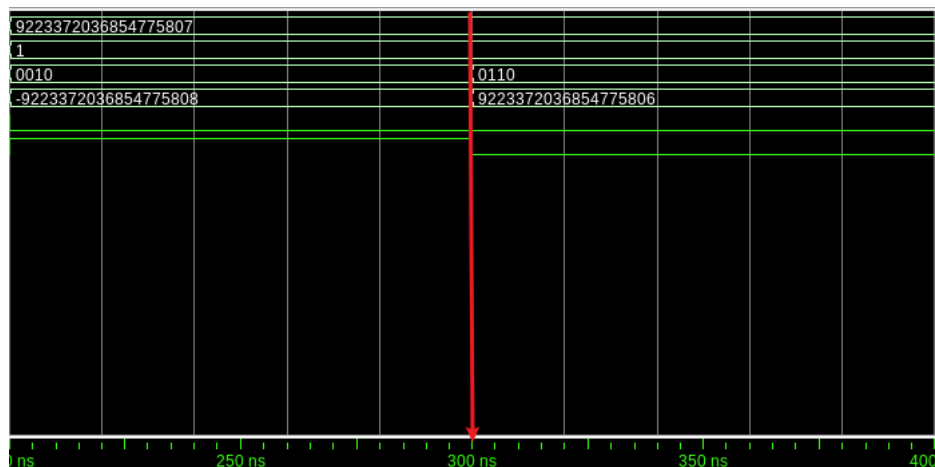


Figure 4: Simulation Waveform of 'ADD' and 'SUB'

In the above waveform, I set A to '1'(63 bits '0')=9223372036854775807, B to (63 bits '0')'1'=1. During 200 to 300 ns, the operation is "0010" which means "ADD" function. The output of "A + B" is -9223372036854775808. Overflow occurs here, and the "overflow" signal is '1'. During 300 to 400 ns, the operation is "0110" which means "SUBTRACT" function. The output of "A - B" is 9223372036854775806. All the outputs are correct.

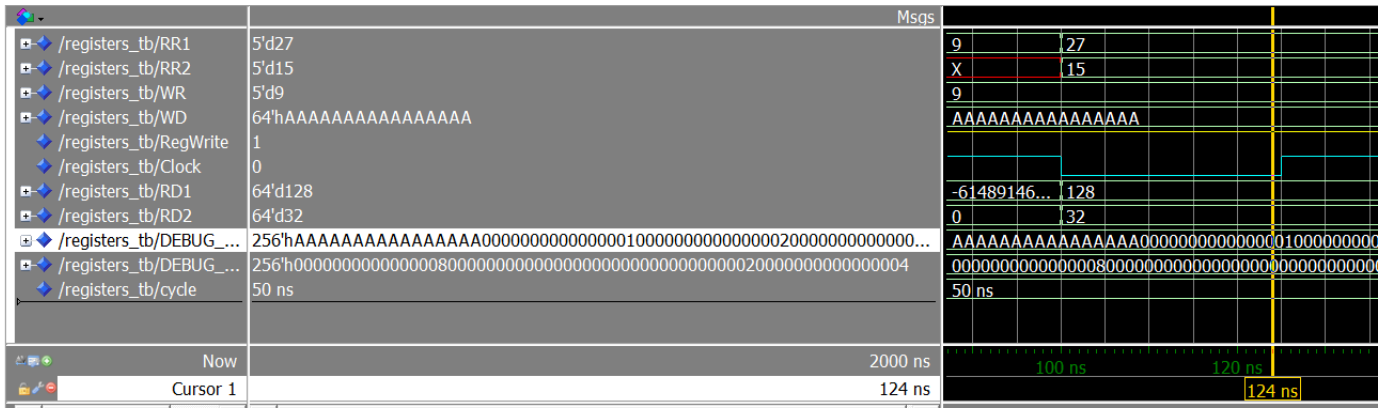


Figure 6: Simulation Waveform of register read

At 100 ns, I set RR1 = "11011", RR2 = "01111". And we get RD1 = \$X27 = 128, RD2 = \$X15 = 32 instantly.

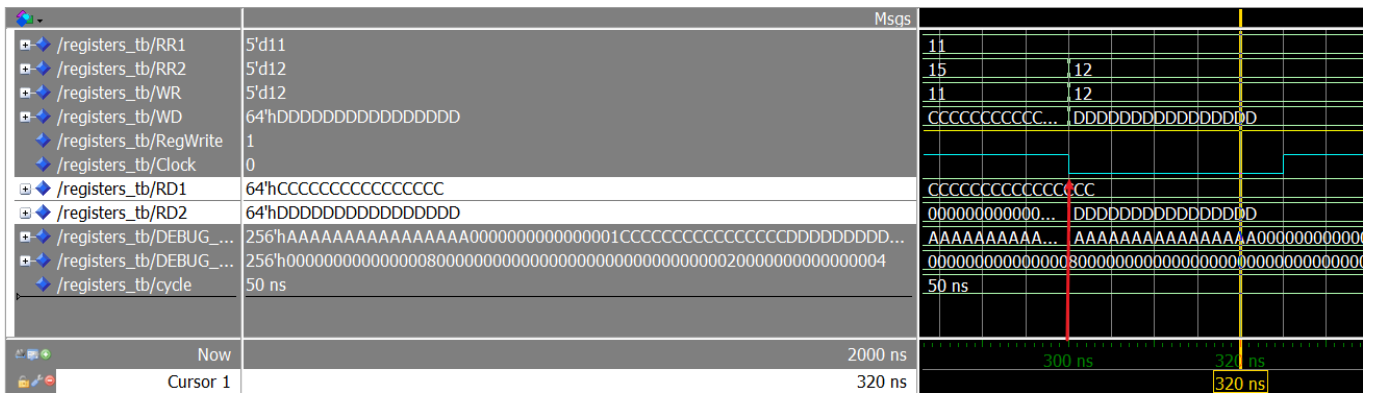


Figure 7: Simulation Waveform of register write

At 300 ns, this is a falling edge of 'Clock', and RegWrite = '1'. I set WR = "01100" = 12, RR2 = "01100" = 12, WD = all 'D's. And we can see the value of \$X12 is changed to "DDDDDDDDDDDDDDDDDD" at this negative edge of 'Clock'.

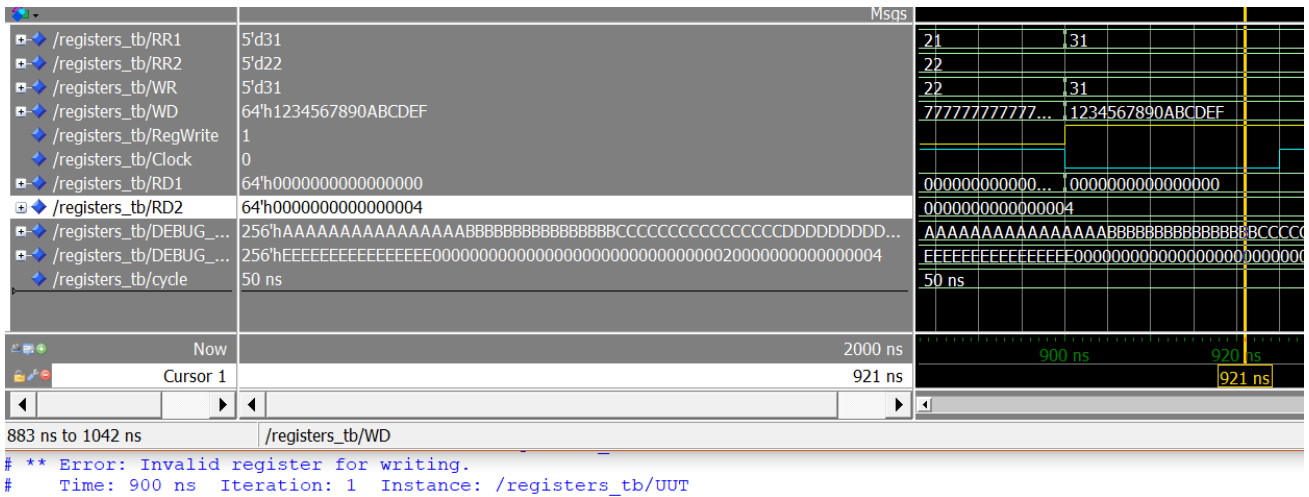


Figure 8: Simulation Waveform of register write to XZR

At 900 ns, I tried to write "0x12345667890ABCDEF" to \$X31, and this is a falling edge of 'Clock', and RegWrite = '1'. But the value of \$X31 stay at 0. And there is an error report "Invalid register for writing" at 900 ns.

3: IMEM

Description

Instruction memory continuously reads instructions, and outputs the read data. IMEM reads from 'Address' and 'ReadData' shows the four bytes data start from that address. I've set `imemBytes(0) = "00000010"`, `imemBytes(16) = "00000011"`, `imemBytes(32) = "00000110"`, and all the other bytes are 8 bits '0'.

Waveforms

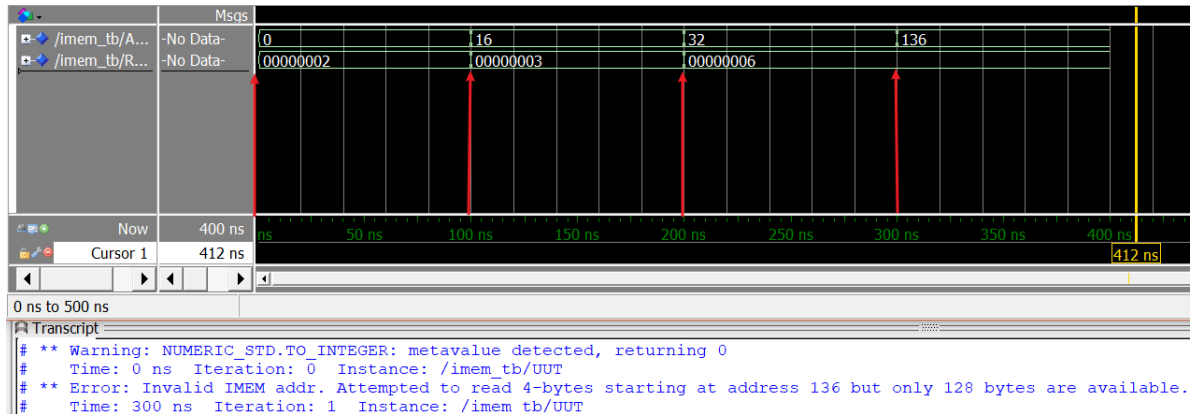


Figure 9: Simulation Waveform of instruction memory

At 0 ns, IMEM reads 4 bytes start from byte(0). The 'ReadData' is "0x00000002". At 100 ns, IMEM reads 4 bytes start from byte(16). The 'ReadData' is "0x00000003". At 200 ns, IMEM reads 4 bytes start from byte(32). The 'ReadData' is "0x00000006". These three cases match my initialization. At 300 ns, IMEM tries to read 4 bytes start from byte(136), we got an error. It fails, because the length of my IMEM is 128 bytes, and byte(136) is not available.

4: D MEM

Description

Data memory can do two operations, read and write. The size of my data memory is 1024KB. It does read operation when 'MemRead' = '1', 'MemWrite' = '0'. It does write operation when 'MemRead' = '0', 'MemWrite' = '1', and rising edge of 'Clock'. For initialization, only Byte(0) = '1', Byte(41) = '1', and all the other Bytes are '0'.

Waveforms

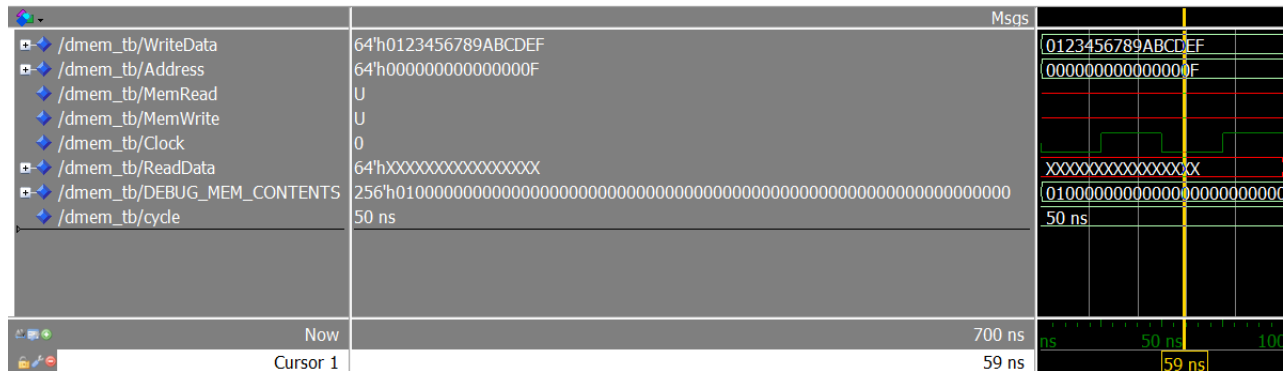


Figure 10: Simulation Waveform of data memory initialization

Data memory is initialized, 'MemRead' and 'MemWrite' have no value, DMEM didn't do anything. The DEBUG_MEM_CONTENTS matches the initialization.

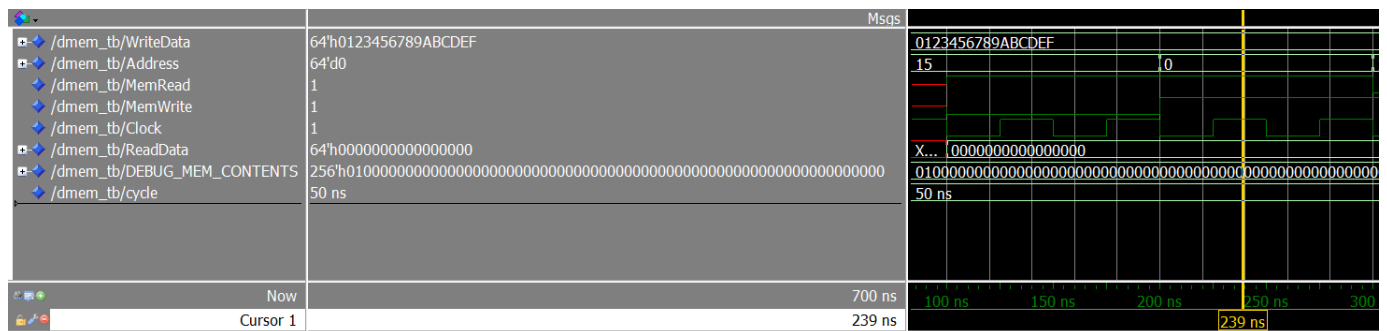


Figure 11: Simulation Waveform of data memory read

At 100 ns, 'MemRead' = '1', 'MemWrite' = '0', 'Address' = 15. DMEM reads data from Bytes(15). At 200 ns, 'MemRead' = '1', 'MemWrite' = '1', 'Address' = 0. If read operation is successful, we should get "0x0000000000000001" from 'ReadData'. DMEM correctly does nothing when both 'MemRead' and 'MemWrite' = '1'.

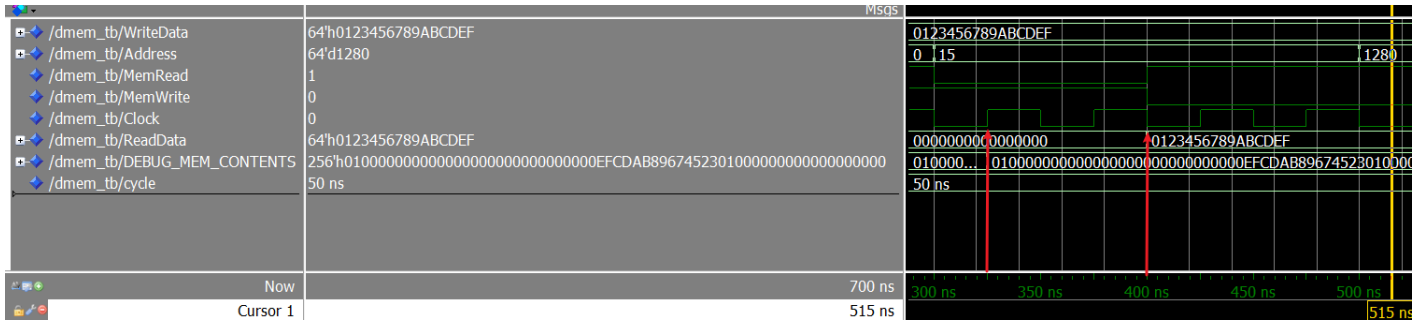


Figure 12: Simulation Waveform of data memory write

At 350 ns, 'MemRead' = '0', 'MemWrite' = '1', 'Address' = 15, and there is a rising edge of Clock. From 'DEBUG_MEM_CONTENTS', we can tell data are correctly written into Byte(15) instantly. At 400 ns, 'MemRead' = '1', 'MemWrite' = '0', 'Address' = 15, 'ReadData' correctly give us the data we just wrote in.

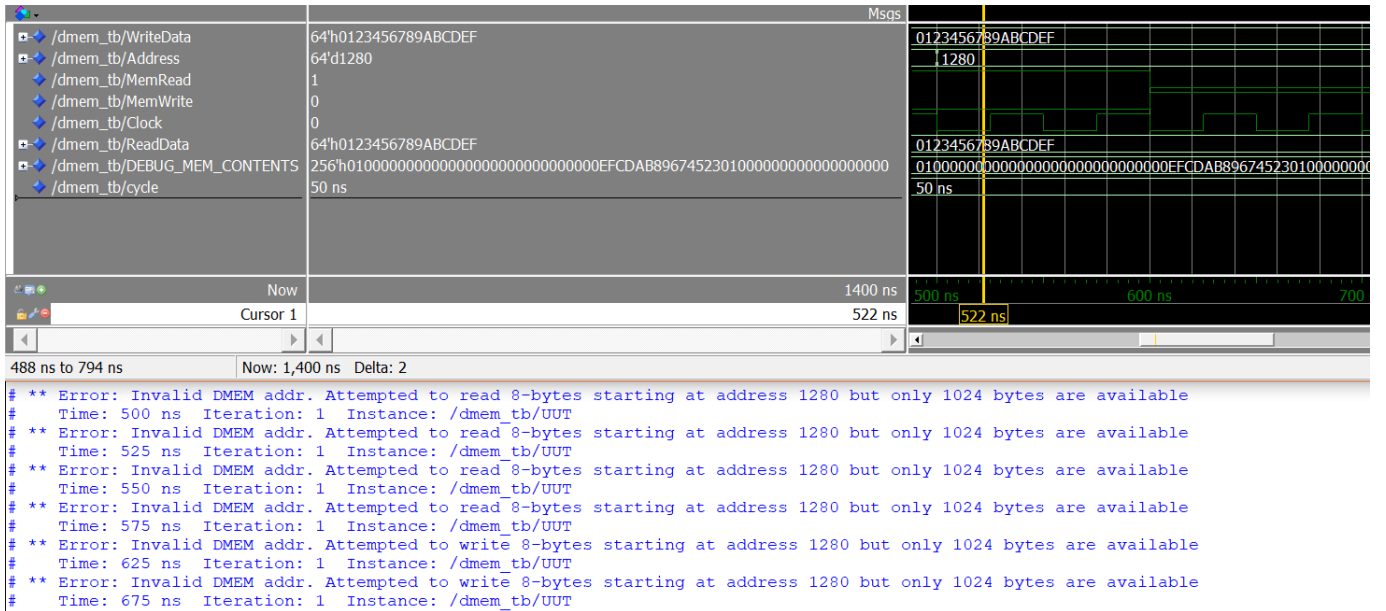


Figure 13: Simulation Waveform of data memory invalid address

At 500 ns, 'MemRead' = '1', 'MemWrite' = '0', 'Address' = 1280. We tried to read data from Byte(1280), which is not available. And we got error information, and the 'ReadData' didn't change. During 600 to 700 ns, 'MemRead' = '0', 'MemWrite' = '1', 'Address' = 1280, and there are rising edges at 625 ns and 675 ns. We tried to write data into Byte(1280), which is not available. And we got error information at the rising edge of 'Clock'.