DIPEN KUMAR (2018CS50098)
UMESH PARMAR (2018CS50424)

COL216 ASSIGNMENT-4

16-02-2020

In this Assignment we have created a block memory generator which has a ip.coe file as an input. ip.coe contain set instructions. It is an array of 32 bit std_logic_vector of length 4096. The remaining uninitialized address of the array is initialized to zero vector automatically. We then use this block memory in our processor.vhd to get the instruction and process the instruction in processor.vhd. We port mapped the block memory generator component to processor.vhd. We now specify the address to fetch data from block memory. The address is updated in next cycle and hence the block memory then send back the data which is further updated in next cycle. Similarly for data to write in memory, we specify the address and data which is updated in next cycle. It is then written at the specified address.

Using the concept of pipelining we achieved to perform add, sub, sll, srl in one clock cycle whereas lw and sw in three clock cycles for single port block memory.

We have seven state in our implementation i.e. {-2,-1,0,1,2,3,4}.

1. We start from state -2 where we request to address to 0.
2. Request address to 1, address=0 (state=-1)
3. Request address to 2, address=1, data=0 (state=0)
4. Request address to 3, address=2, data=1, executing=0 (state=1)
5. If load/store and executing=i then request address=i+1 (state=3)
6. If load/store and executing=I then request address=i+2, address=i+1 (state=4)
7. State 2 is idle state (doing nothing all instruction has been executed)

We have two signals- k (cycle counter) and value (16 bit of output register in the last instruction). We have displayed k when the switch is '1' and value when switch is 0 on seven segment display in hexadecimal form because in order to display 16 bit vector in binary we need hexadecimal to express all values in 4 bit vector (4-seven segment display). (2^16=16^4).
In order to make life easy we again used hexadecimal to display count of clock cycle. In VHDL it is not so trivial to access individual digits of an integer to print it on seven-segment display. So we converted it in binary which is straight forward simple by using inbuilt command. Now since we have std_logic_vector we can now access any range of bits from the vector. 4 most significant bit of 16 bit vector in binary will be for sure the most significant digit of 4-digit hexadecimal number. (2^4=16^1)