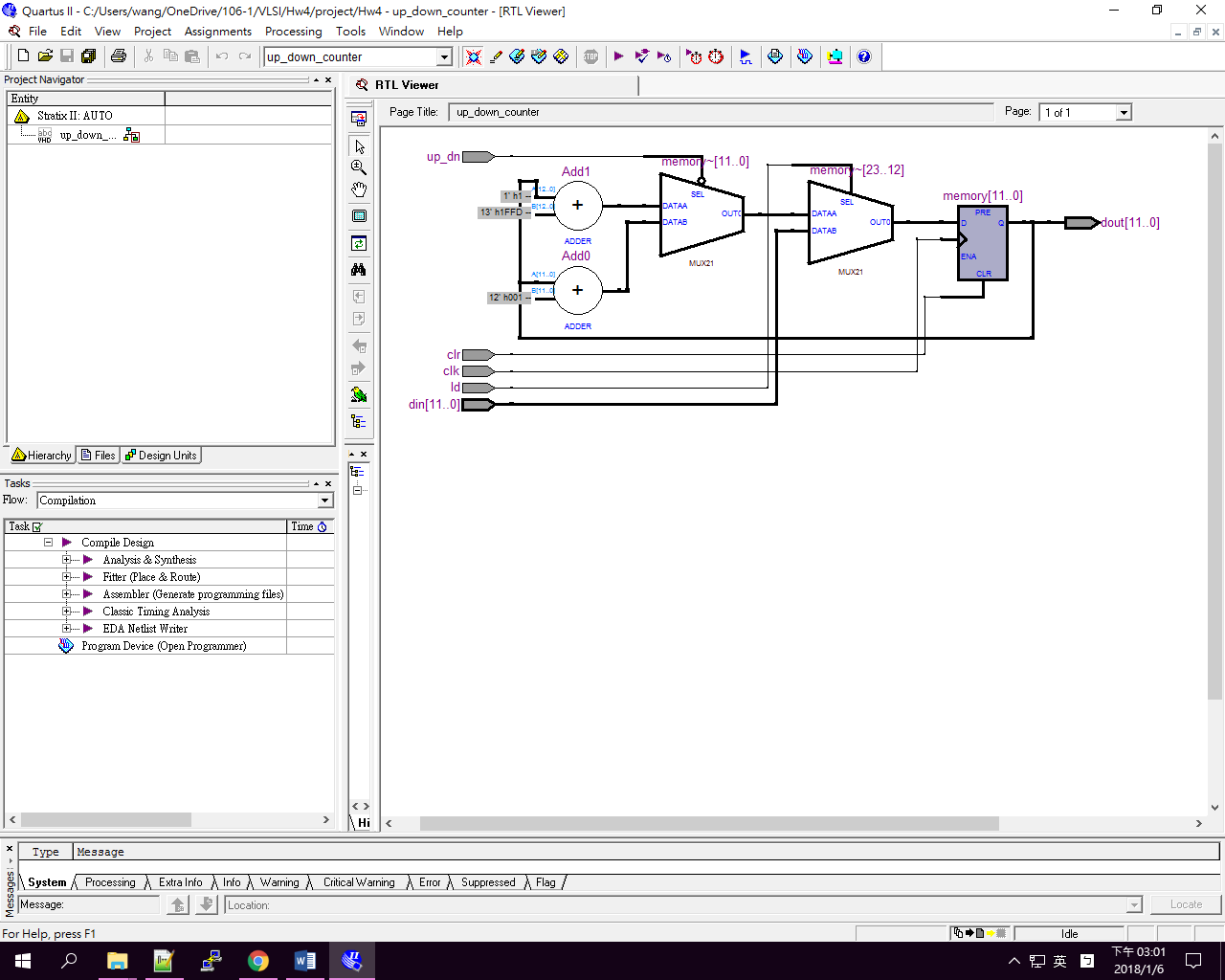
Hw 4 40243118S 王擎天

1. 設計電路說明(20%):

此程式藉由一個D type Flip-flop來儲存記憶,並透過訊號的輸入up\_dn來決定將D type Flip-flop中的值加一或減一,並存回D type Flip-flop.透過clr的訊號將D type Flip-flop中的值初始化為零.透過ld的訊號將din存入D type Flip-flop.所有在D type Flip-flop的值都會輸出到dout.

2. 程式碼說明(30%):

Up\_down\_counter:

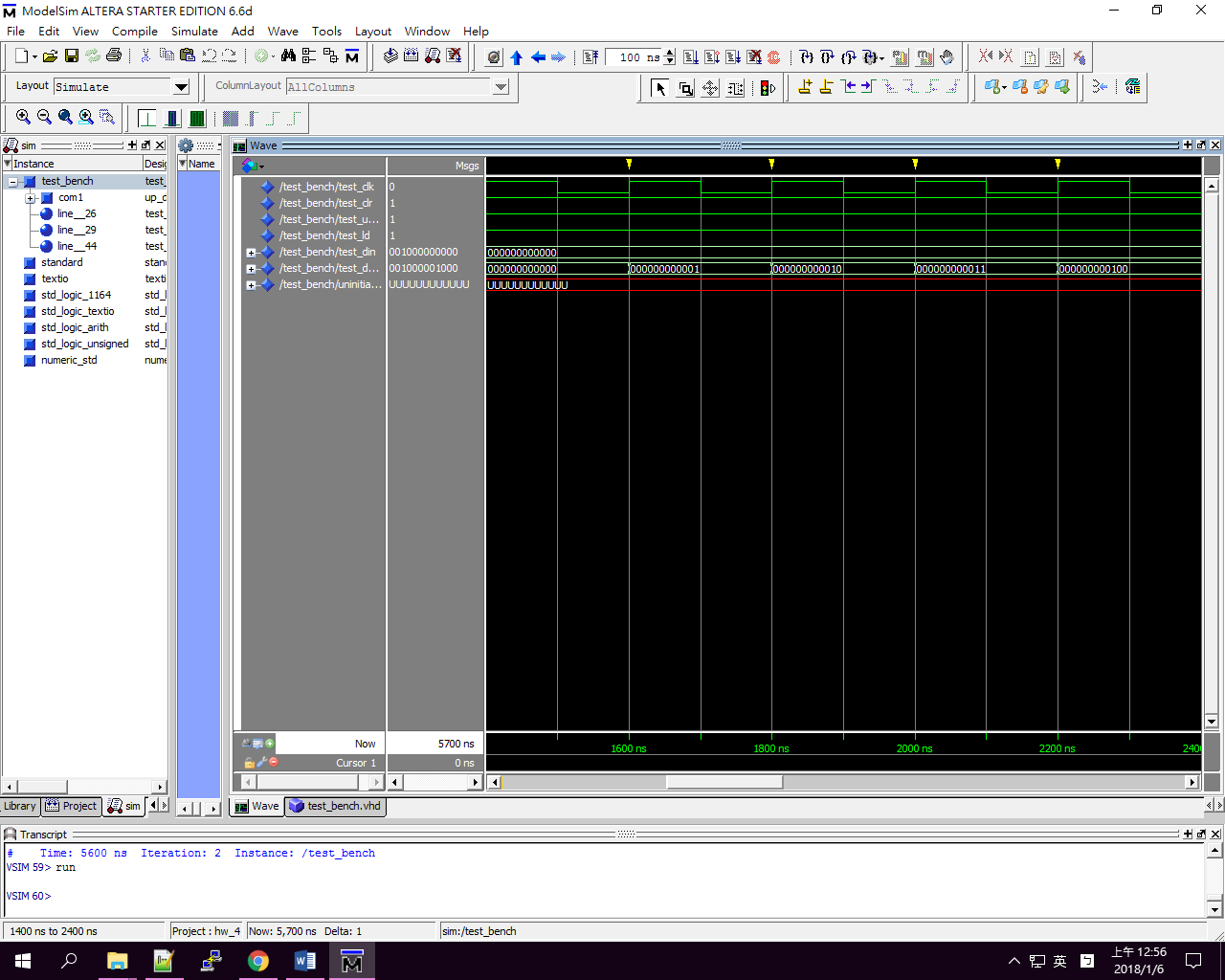
在clr為零時會將memory初始化成0.在clock為rising edge時,如果load為0,則將din存入memory.假如up\_dn為1,則memory加一後存入memory,否則memory減一後存入memory.每當memory改變時則將結果輸出到dout.

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  use ieee.numeric\_std.all;  entity up\_down\_counter is  generic(K:integer:=12);  port(clk,clr,up\_dn,ld :in std\_logic;  din:in std\_logic\_vector(K-1 downto 0);  dout:out std\_logic\_vector(K-1 downto 0)  );  end entity up\_down\_counter;  architecture up\_down\_counter\_arch of up\_down\_counter is  signal memory:std\_logic\_vector(K-1 downto 0):=std\_logic\_vector(to\_unsigned(0, K));  begin  process(clk,clr)  begin  if clr='0' then  memory<=std\_logic\_vector(to\_unsigned(0, K));  else  if clk'EVENT and clk='1' then  if (ld='0') then  memory<=din;  else  if (up\_dn='1') then  memory<=memory+'1';  else  memory<=memory-'1';  end if;  end if;    end if;  end if;  end process;  process(memory)  begin  dout<=memory;  end process;  end architecture up\_down\_counter\_arch ; |

testbench file:

模擬訊號clock,週期為100ns.藉由"input\_data.txt"中的資料輸入訊號到up\_down\_counter,並將up\_down\_counter的輸出寫到檔案"output\_data.txt".輸入的訊號為binary,從右算起第一個為load,第二個為up\_dn,第三個為clr,其餘為din訊號.輸出也為binary訊號(dout). uninitialValue不會被初始化,它被用來判斷輸出資料是否初始化過,如果沒有則不將訊號輸出.

|  |
| --- |
| library ieee;  library std;  use std.textio.all;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_textio.all;  use ieee.std\_logic\_unsigned.all;  use ieee.numeric\_std.all;  entity test\_bench is  generic(test\_K:integer:=12);  end entity test\_bench;  architecture test\_bench\_arch of test\_bench is  signal test\_clk,test\_clr,test\_up\_dn,test\_ld:std\_logic;  signal test\_din:std\_logic\_vector(test\_K-1 downto 0);  signal test\_dout:std\_logic\_vector(test\_K-1 downto 0);  signal uninitialValue:std\_logic\_vector(test\_K-1 downto 0);  component up\_down\_counter is  generic(K:integer:=12);  port(clk,clr,up\_dn,ld :in std\_logic;  din:in std\_logic\_vector(K-1 downto 0);  dout:out std\_logic\_vector(K-1 downto 0)  );  end component up\_down\_counter;  begin  test\_clk <= '1' after 50 ns when test\_clk='0' else '0' after 50 ns;  COM1:up\_down\_counter generic map(K=>test\_K) port map(clk=>test\_clk,clr=>test\_clr,up\_dn=>test\_up\_dn,ld=>test\_ld,din=>test\_din,dout=>test\_dout);  process  file input\_file : text is in "input\_data.txt";  variable input\_line : line;  variable temp : std\_logic\_vector(test\_K+2 downto 0);  begin  while not endfile(input\_file) loop  wait until test\_clk'EVENT and test\_clk='1';  readline(input\_file,input\_line);  read(input\_line,temp);  test\_din <= temp(test\_K+2 downto 3);  test\_clr <= temp(2);  test\_up\_dn <= temp(1);  test\_ld <= temp(0);  end loop;  wait;  end process;  process(test\_dout)  file output\_file : text is out "output\_data.txt";  variable output\_line : line;  begin  if(test\_dout /= uninitialValue)then  write(output\_line,test\_dout);  writeline(output\_file,output\_line);  end if;  end process;  end architecture test\_bench\_arch; |

3. 模擬結果說明(30%):

|  |  |  |
| --- | --- | --- |
| 說明 | Input | Output |
| Up-down=1:原數字加一  Up-down=1:原數字加一  Up-down=1:原數字加一  Up-down=1:原數字加一  Up-down=1:原數字加一  Clr 0:初始化  Up-down=1:原數字加一  Up-down=1:原數字加一  Up-down=1:原數字加一  Up-down=1:原數字加一  Up-down=1:原數字加一  Up-down=1:原數字加一  Up-down=0:原數字減一  Up-down=0:原數字減一  Up-down=0:原數字減一  Up-down=1:原數字加一  Up-down=1:原數字加一  Up-down=1:原數字加一  Load=0:  Load number:1000000000  Up-down=1:原數字加一  Up-down=1:原數字加一  Up-down=1:原數字加一 | 000000000000111  000000000000111  000000000000111  000000000000111  000000000000111  000000000000011  000000000000111  000000000000111  000000000000111  000000000000111  000000000000111  000000000000111  000000000000101  000000000000101  000000000000101  000000000000111  001000000000111  001000000000111  001000000000110  001000000000111  001000000000111  001000000000111 | 000000000000  111111111111  000000000000  000000000001  000000000010  000000000011  000000000100  000000000000  000000000001  000000000010  000000000011  000000000100  000000000101  000000000110  000000000101  000000000100  000000000011  000000000100  000000000101  000000000110  001000000000  001000000001  001000000010  001000000011 |

4. 結論及心得(20%):

輸出訊號可以得知此testbench確實可以正確的模擬clock signal並將輸入檔的訊號輸入up\_down\_counter裡,經過正常的執行後,將它的輸出寫到外部檔案去.同時可以看到輸出檔中並無uninitialized value,可見它可以判斷只將非uninitialized value的訊號寫到外部檔案.透過這次的練習我可以將我設計的線路放入ModelSim去做測試,這次的作業也讓我學到最大的收穫是學會如何用VHDL去讀文件中的資料,並將訊息寫到文件中.

資料說明:

1. input\_data.txt: 輸入訊號
2. output\_data.txt: 輸出訊號
3. up\_down\_counter.vhd: up down counter的程式碼
4. test\_bench.vhd: testbench的程式碼
5. hw\_4.mpf:ModelSim的project file
6. work: ModelSim的project directory