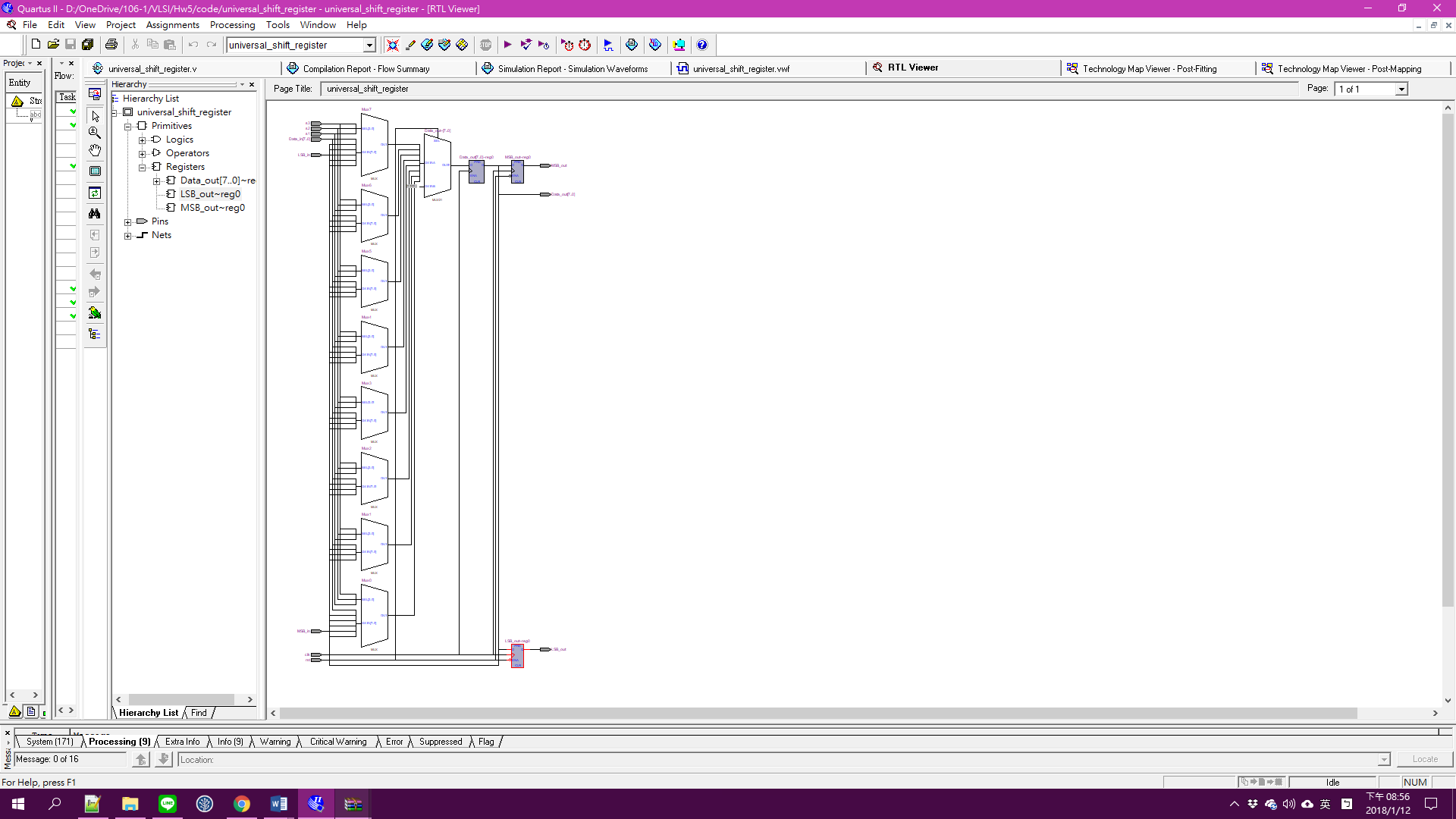
Hw 5 40243118S 王擎天

1. 設計電路說明(20%):

此設計將Data\_out, MSB\_out, LSB\_out設定為D-type FF.輸入訊號由MSB\_in, LSB\_in, Data\_in決定.同時利用s1, s2, s3等訊號來決定multiplexer對輸入訊號和D-type FF輸入的選擇.最後將結果輸出到Data\_out,其中MSB的訊號輸出到MSB\_out, LSB的訊號輸出到LSB\_out.rst則決定D-type FF輸出與否(Enable).



2. 程式碼說明(30%):

在rst為一時會將Data\_out初始化成0.在clk為rising edge時,會執行下列功能:

|  |  |
| --- | --- |
| {s3, s2, s1} | 功能 |
| 000 | 輸出保留在LSB\_out中的值 |
| 001 | Right shift(MSB comes from MSB\_in) |
| 010 | Circular Right shift |
| 011 | Right shift(MSB comes from MSB of LSB\_out) |
| 100 | Left shift(LSB comes from LSB\_in) |
| 101 | Left shift(LSB comes from LSB of LSB\_out) |
| 110 | Circular Left shift |
| 111 | 輸出LSB\_in中的值 |

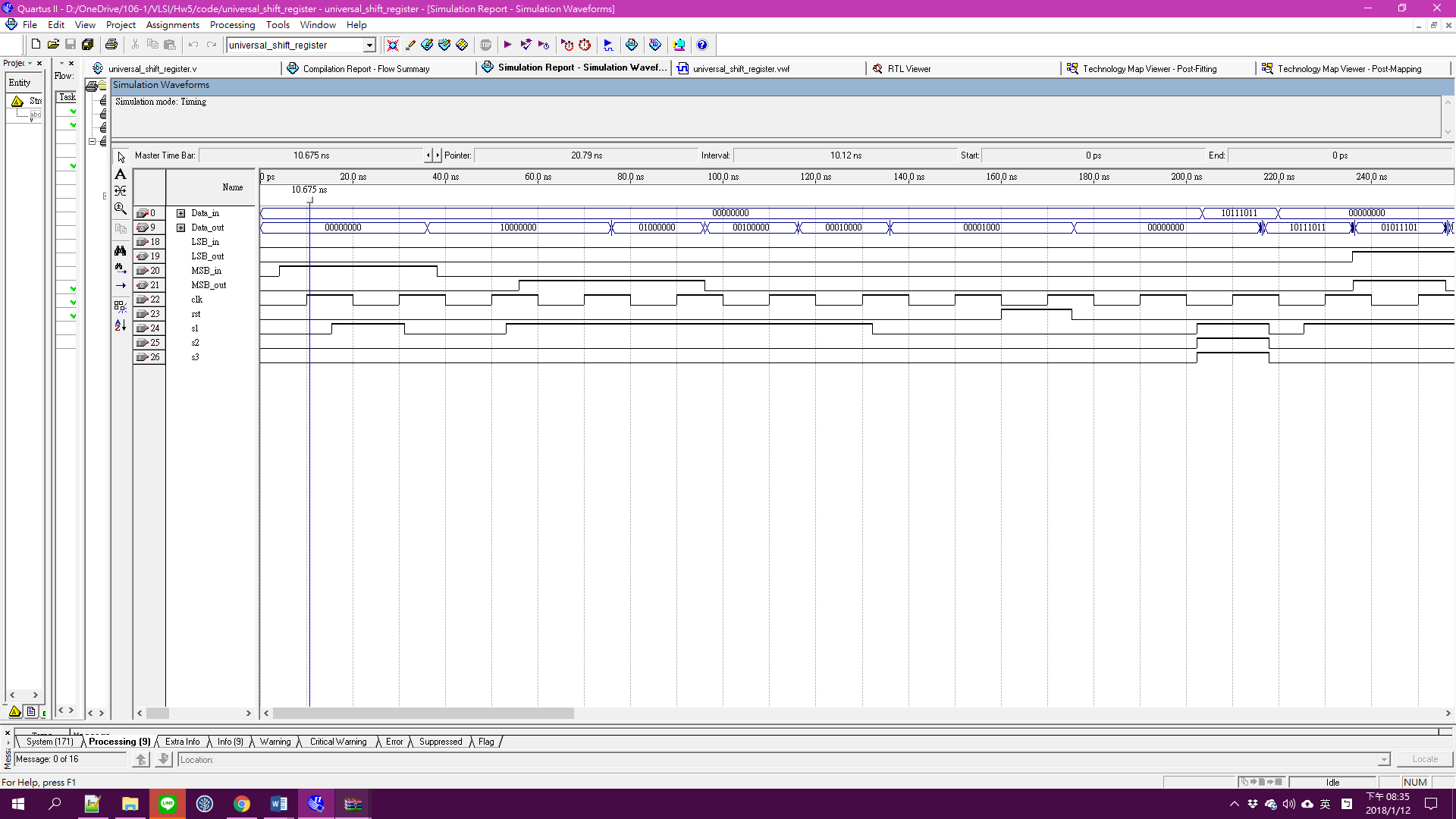
MSB\_out則輸出LSB\_out的Most Significant Bit, LSB\_out則輸出Least Significant Bit.

|  |
| --- |
| module universal\_shift\_register(MSB\_out,LSB\_out,Data\_out[7:0],clk, rst,s1,s2,s3,MSB\_in,LSB\_in,Data\_in [7:0]);  output reg MSB\_out;  output reg LSB\_out;  output reg[7:0]Data\_out ;  input clk;  input rst;  input s1;  input s2;  input s3;  input MSB\_in;  input LSB\_in;  input [7:0] Data\_in ;  always @(posedge clk)  begin  if (rst) begin Data\_out<=8'b0; end  else begin  case({s3, s2, s1})  3'b000:Data\_out<=Data\_out;  3'b001:Data\_out<={MSB\_in, Data\_out[7:1]};  3'b010:Data\_out<={Data\_out[0], Data\_out[7:1]};  3'b011:Data\_out<={Data\_out[7], Data\_out[7:1]};  3'b100:Data\_out<={Data\_out[6:0], LSB\_in};  3'b101:Data\_out<={Data\_out[6:0], Data\_out[0]};  3'b110:Data\_out<={Data\_out[6:0], Data\_out[7]};  default:Data\_out<=Data\_in;  endcase  MSB\_out<=Data\_out[7];  LSB\_out<=Data\_out[0];  end  end  endmodule |

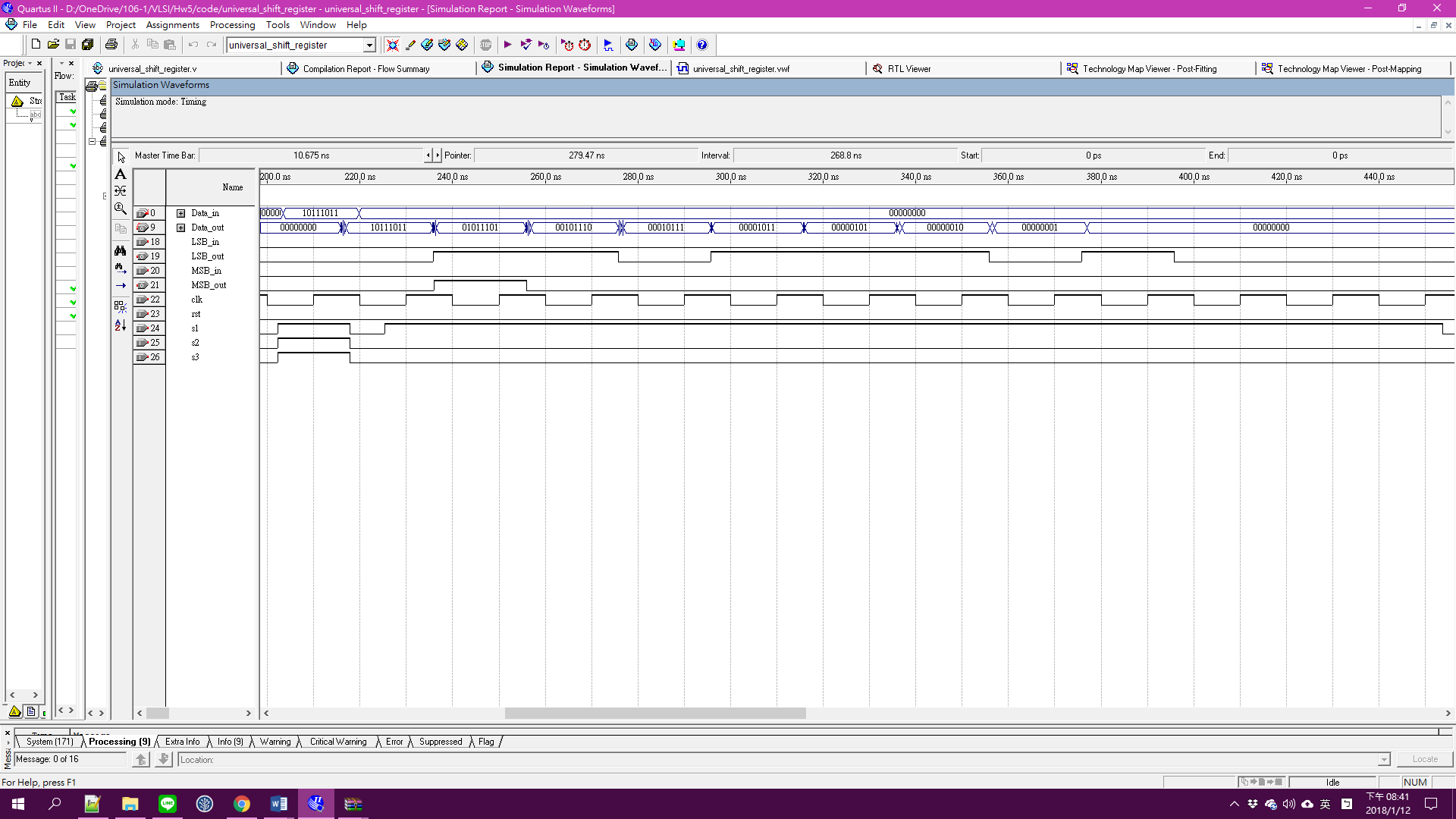
1. 模擬結果說明(30%):

(a)

在{s3,s2,s1}為{0,0,1}和MSB\_in為1時,Data\_out中的值right shift並將Most Significant Bit設定為MSB\_in,最後將結果輸出到Data\_out.接下來,在{s3,s2,s1}為{0,0,1}和MSB\_in為0時,不斷進行right shift達到serial-in parallel-out (SIPO) register的效果.

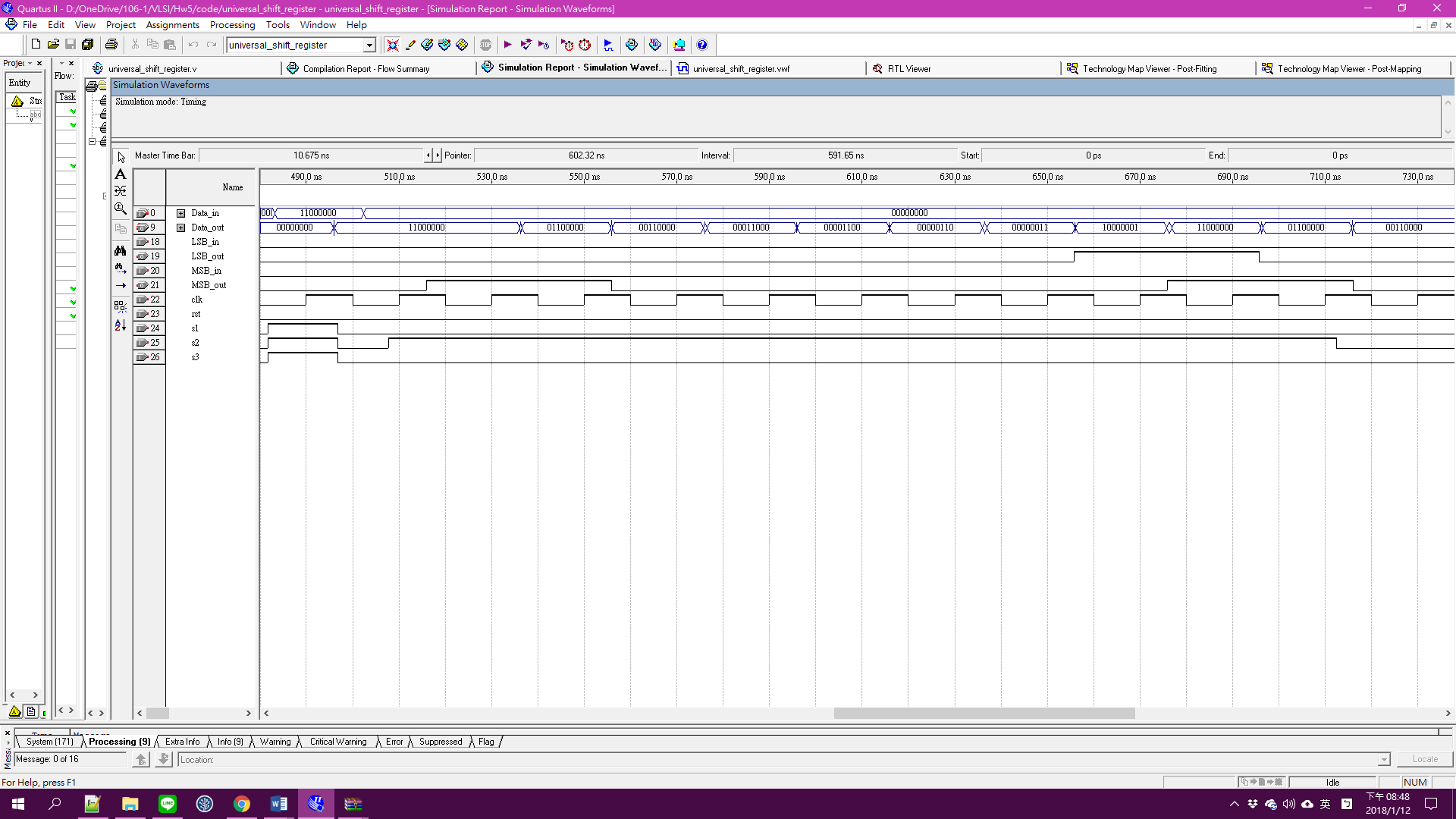
接下來,rst設成一後, Data\_out會變成零.

(b)

在{s3,s2,s1}為{1,1,1}和Data\_in為10000000時,Data\_out中的值設定成Data\_in. 在{s3,s2,s1}為{0,1,0},Data\_out會不斷進行right shift,同時在LSB\_out可以看出Data\_out中LSB的狀態,達到parallel-in-serial-out (PISO) register的效果.

(c)

在{s3,s2,s1}為{1,1,1}和Data\_in為10111011時,Data\_out中的值設定成Data\_in. 在{s3,s2,s1}為{0,0,1},Data\_out會不斷進行Circular right shift,以下為Data\_out中數值的變化,呈現ring counter的效果:

11000000🡪01100000🡪00110000🡪0001100🡪…🡪00000011🡪10000001🡪11000000🡪…

1. 結論及心得(20%):

從上述的模擬圖可得知,此程式可以正常運作,透過不同指令集的排列,可以達到universal shift register的效果,其中從RTL viewer的圖可以看出其universal shift register的運作原理.這個程式是以Verilog語言寫出來的,這次的練習可以體會到Verilog和VHDL的差異,並熟習Verilog的語法.同時也可以學習到要如何用程式寫出universal shift register 的效果.