Sequential Binary Multiplier

第18組 A1053308 蘇詠盛、A1055547 王公志

# Abstract

使用Verilog做出Sequential Binary Multiplier，將教材ppt乘法器的DATAPATH以及CONTROL獨立為兩個MODULE。

# Content

VHDL程式碼

* **Sequential Binary Multiplier：**

module Sequential\_Binary\_Multiplier(Product, Ready, Multiplicand, Multiplier, Start, clock, reset\_b);

parameter dp\_width = 5;

output [2\*dp\_width - 1: 0] Product;

output Ready;

input [dp\_width - 1: 0] Multiplicand, Multiplier;

input Start, clock, reset\_b;

wire Load\_regs, Decr\_P, Add\_regs, Shift\_regs;

control (Ready, Load\_regs, Add\_regs, Shift\_regs, Decr\_P, Zero, Q0, clock, reset\_b, Start);

datapath (Product, Q0, Zero, Multiplicand, Multiplier, Load\_regs, Add\_regs, Shift\_regs, Decr\_P, clock);

endmodule

* **Control Unit：**

module control(Ready, Load\_regs,Add\_regs, Shift\_regs, Decr\_P, Zero , Q0, clock, reset\_b, Start);

output Ready;

input clock, Q0, Zero, reset\_b, Start;

output Load\_regs, Decr\_P, Add\_regs, Shift\_regs;

reg Load\_regs, Decr\_P, Add\_regs, Shift\_regs;

reg [2:0] state, next\_state;

parameter S\_idle = 3'b001,

S\_add = 3'b010,

S\_shift = 3'b100;

wire Ready = (state == S\_idle);

always@(posedge clock, negedge reset\_b)begin

if(~reset\_b)

state <= S\_idle;

else

state <= next\_state;

end

always@(state, Start, Q0, Zero)begin

next\_state <= S\_idle;

Load\_regs <= 0;

Decr\_P <= 0;

Add\_regs <= 0;

Shift\_regs <= 0;

case(state)

S\_idle: begin if (Start) next\_state <= S\_add;

Load\_regs <= 1;

end

S\_add: begin next\_state <= S\_shift;

Decr\_P <= 1;

if(Q0) Add\_regs <= 1;

end

S\_shift: begin Shift\_regs <= 1;

if(Zero) next\_state <= S\_idle;

else next\_state <= S\_add;

end

endcase

end

endmodule

* **Datapath Unit：**

module datapath (Product, Q0,Zero, Multiplicand, Multiplier, Load\_regs, Add\_regs, Shift\_regs, Decr\_P, clock);

output Zero, Q0;

output [2 \* dp\_width - 1:0] Product;

input [dp\_width-1:0] Multiplicand,Multiplier;

input clock, Load\_regs, Add\_regs, Shift\_regs, Decr\_P;

parameter BC\_size = 3;

parameter dp\_width = 5;

assign Product = {A, Q};

reg [dp\_width - 1: 0] A, B, Q;

reg C;

reg [BC\_size - 1:0] P;

wire Q0 = Q[0];

wire Zero = (P == 0);

always@(posedge clock) begin

if(Load\_regs) begin

P <= dp\_width;

A <= 0;

C <= 0;

B <= Multiplicand;

Q <= Multiplier;

end

if(Add\_regs){C, A} <= A + B;

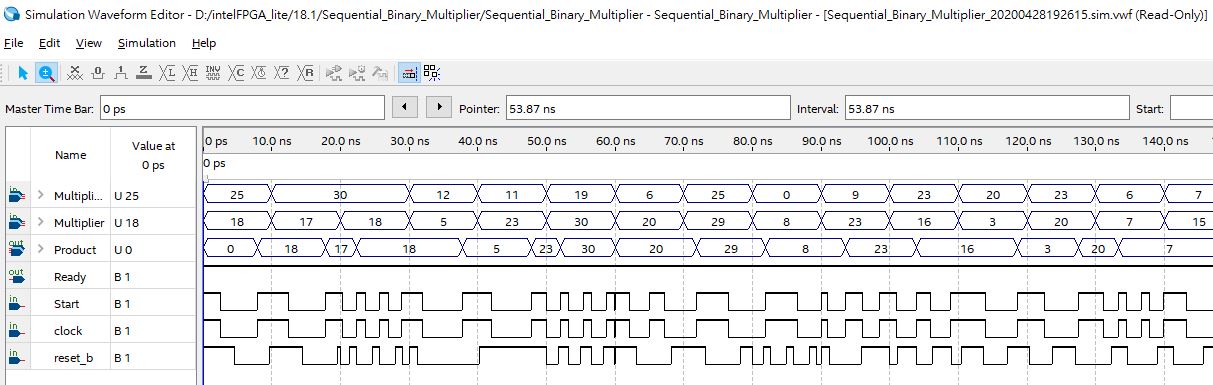
if(Shift\_regs){C ,A, Q} <= {C, A, Q} >> 1;

if(Decr\_P)P <= P-1;

end

endmodule

**模擬結果：**



# Discussions

本次作業基本可以由各堂的PPT上得到作法，其餘部分則必須注意到reg和wire等等資料型態的差別，根據不同用途在不同module中來宣告。

**參考資料：課堂上課PPT講義**