



VHDL除頻電路與狀態機電路



Outline

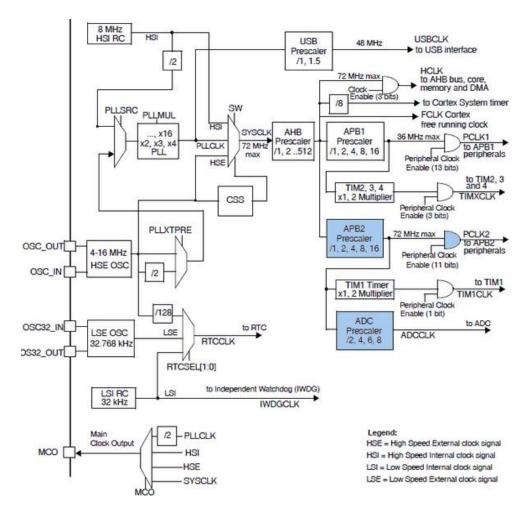
- 除頻電路
 - 非同步計數器
 - 同步計數器
 - 補充:RTL Schematic
- 延遲電路
 - 累積時脈延時
 - 基本延時電路延時
- 有限狀態機設計
 - 米利機(Mealy Machine)
 - 莫爾機(Moore Machine)



除頻器



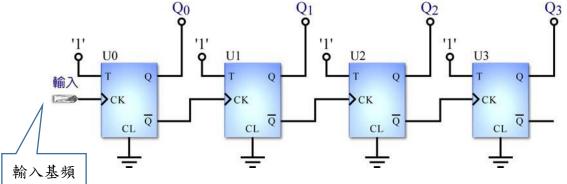
• 為什麼需要除頻器





除頻器

- 非同步計數器:
 - 延遲誤差(Propagation Delay)會累積
- 基頻來源
 - 有意義的頻率

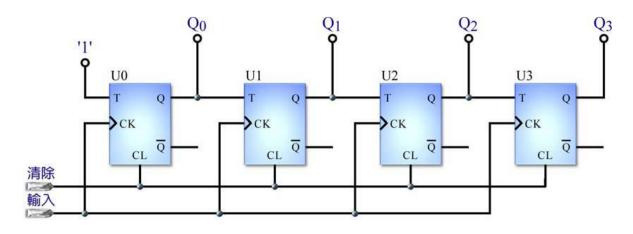


PR	CL	СК	T	Q	$\overline{\overline{Q}}$
0	1	-	-	1	0
1	0	-	-	0	1
1	1	↑	0	Q	$\overline{\overline{Q}}$
1	1	↑	1	\overline{Q}	Q

CK	Q0	Q0'	Q1	Q1'	Q2	Q2'	Q3	Q3'
0	0	1	0	1	0	1	0	1
1	1	0	0	1	0	1	0	1
0	1	0	0	1	0	1	0	1
1	0	1	1	0	0	1	0	1
0	0	1	1	0	0	1	0	1
1	1	0	1	0	0	1	0	1
0	1	0	1	0	0	1	0	1
1	0	1	0	1	1	0	0	1
0	0	1	0	1	1	0	0	1
1	1	0	0	1	1	0	0	1
0	1	0	0	1	1	0	0	1
1	0	1	1	0	1	0	0	1
0	0	1	1	0	1	0	0	1
1	1	0	1	0	1	0	0	1
0	1	0	1	0	1	0	0	1
1	0	1	0	1	0	1	1	0
0	0	1	0	1	0	1	1	0
1	1	0	0	1	0	1	1	0
0	1	0	0	1	0	1	1	0
1	0	1	1	0	0	1	1	0
0	0	1	1	0	0	1	1	0
1	1	0	1	0	0	1	1	0
0	1	0	1	0	0	1	1	
1	0	1	0	1	1	0	0	1

除頻器

• 同步計數器



		Name	0 ps 0 ps	2.0 us	4.0 us	6.0 us	8.0 us ^
1MHz	in_	CLK				<mark>4 1us</mark>	→
500KHz	out	Q[0]				205	→
250KHz	out	Q[1]			4	4 us	→
250KHz	out	Q[2]			-	4us	-
125KHz	out	Q[3]	—		8us		-
	<	>	<		İ		> \

PR	CL	CK	T	Q	$\overline{\overline{Q}}$
0	1	-	-	1	0
1	0	-	-	0	1
1	1	1	0	Q	Q
1	1	↑	1	\bar{Q}	Q

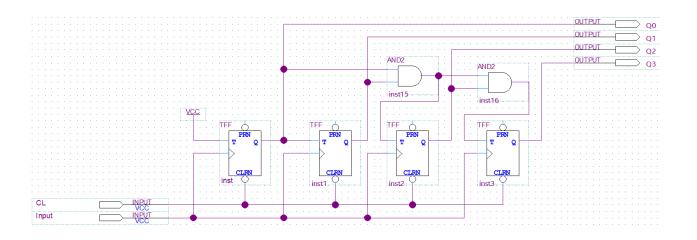
4位元同步下數計數器

CK	T1	Q0	Q1	Q2	Q3	
0	1	0	0	0	0	0
1	1	1	1	1	1	0 15
0	1	1	1	1	1	
1	1	0	1	0	1	10
0	1	0	1	0	1	
1	1	1	0	0	1	9
0	1	1	0	0	1	
1	1	0	0	0	1	8
0	1	0	0	0	1	
1	1	1	1	1	0	7
0	1	1	1	1	0	
1	1	0	1	0	0	2
0	1	0	1	0	0	
1	1	1	0	0	0	1
0	1	1	0	0	0	
1	1	0	0	0	0	0
0	1	0	0	0	0	
1	1	1	1	1	1	15
0	1	1	1	1	1	10
1	1	0	1	0	1	10
0	1	0	1	0	1	
1	1	1	0	0	1	9
0	1	1	0	0	1	
1	1	0	0	0	1	8
0	1	0	0	0	1	
1	1	1	1	1	0	7
0	1	1	1	1	0	
1	1	0	1	0	0	2
CK 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Q0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0	Q1 0 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0	Q2 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Q3 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0	1
1	1	1	0	0	0	1
0	1	1	0	0	0	
1	1	Λ	Λ	Λ	Λ	Λ

結論同步計數器也可以是:

- 1. 四位二進制加法計數器
- 2. (一位)十六進制計數器或模十六計數器
- 3. 分頻器
- 同步計數器(修正)

除頻器



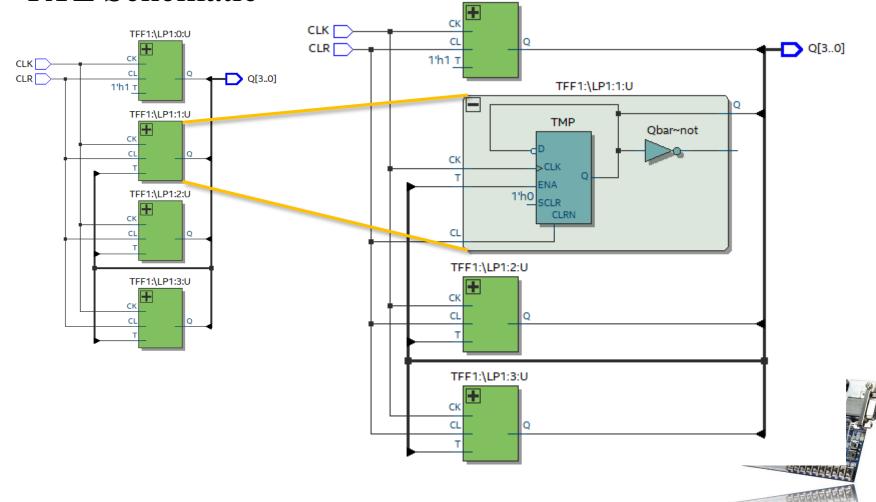
2.0 us 4.0 us 8.0 us 10.0 us 12.0 us 14.0 us 16.0 6.0 us Name CL 1MHz Input 500KHz Q0 250KHz Q1 125KHz 02 62.5KHz Q3 > V

4位:	元同:	步卜	双計署	双器(1	隊止)	
CK	T1	Q0	Q1	Q2	Q3	
0	1	0	0	0	0	0
1	1	1	1	1	1	15
0	1	1	1	1	1	
1	1	0	1	1	1	14
0	1	0	1	1	1	
1	1	1	0	1	1	13
0	1	1	0	1	1	
1	1	0	0	1	1	12
0	1	0	0	1	1	
1	1	1	1	0	1	11
0	1	1	1	0	1	11 10
1	1	0	1	0	1	10
0	1	0	1	0	1	
1	1	1	0	0	1	9
0	1	1	0	0	1	
1	1	0	0	0	1	8
0	1	0	0	0	1	7
1	1	1	1	1	0	7
0	1	1	1	1	0	
1	1	0	1	1	0	6 5
0	1	0	1	1	0	
1	1	1	0	1	0	5
0	1	1	0	1	0	4
1	1	0	0	1	0	4
0	1	0	0	1	0	
1	1	1	1	0	0	3
0	1	1	1	0	0	
1	1	0	1	0	Q3 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0	2 1
0	1	0	1	0	0	
1	1	1	0	0	0	1
0	1	1	0	0	0	1
1	1	0	0	0	0	0

A位于同步下断针断界/修订\

補充

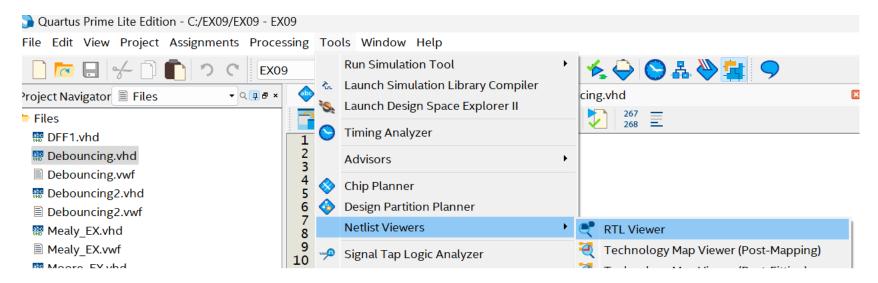
• RTL Schematic



TFF1:\LP1:0:U

補充

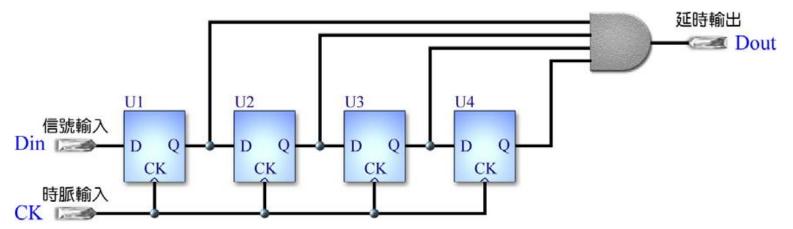
- RTL Schematic
- Tool-Netlist Viewers-RTL Viewer





- 累積時脈延時(1/4)
 - 延時輸出
 - 確保穩定輸入訊號
 - 防彈跳電路

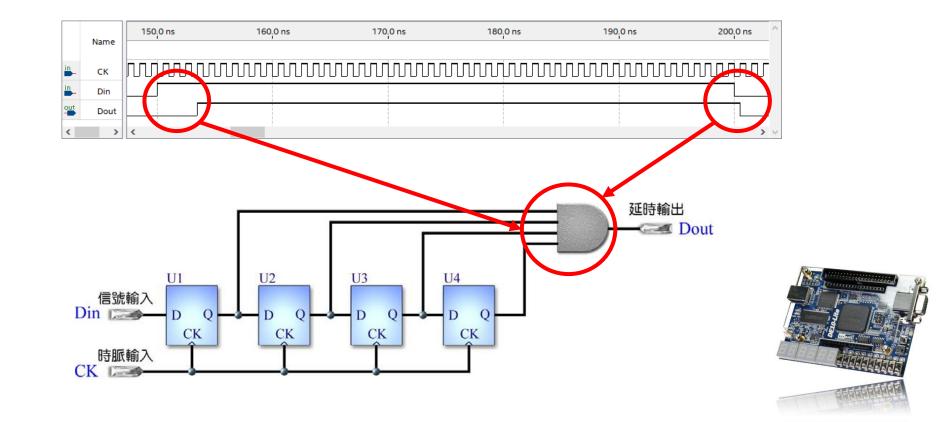
PR	CL	СК	D	Q	$\overline{\overline{Q}}$
0	1	-	-	1	0
1	0	-	-	0	1
1	1	\uparrow	0	0	1
1	1	↑	1	1	0



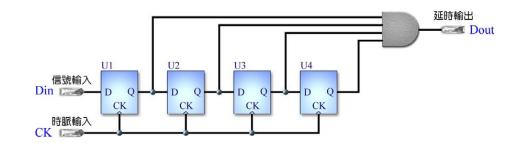


• 累積時脈延時(2/4)

PR	CL	СК	D	Q	$\overline{\overline{Q}}$
0	1	-	-	1	0
1	0	-	-	0	1
1	1	↑	0	0	1
1	1	↑	1	1	0



• 累積時脈延時(3/4)



```
Library IEEE;
 Use IEEE.std_logic_1164.all;

□ Entity Debouncing is

    Port( Din,CK:in std_logic;
           Dout:out std_logic);
LEnd Debouncing:
□ Architecture ARCH of Debouncing is
Component DFF1
       Port( D,CK:in std_logic;
              Q:out std_logic);
    End Component;
    Signal TMP: std_logic_vector(4 downto 0);
 Begin
    TMP(0) \le Din;
    Lp1:For I in 1 to 4 Generate
       U1:DFF1 Port Map(TMP(I-1), CK,TMP(I));
    End Generate;
    Dout \leftarrow TMP(4) and TMP(3) and TMP(2) and TMP(1);
 End ARCH;
```

```
Library IEEE;
USE IEEE.std_logic_1164.all;

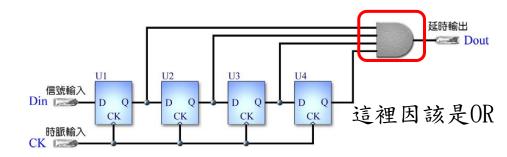
Entity DFF1 is
Port( D,CK:in std_logic;
Q:out std_logic);
End DFF1;

Architecture ARCH of DFF1 is
Begin
Process(CK)
Begin
If Rising_Edge(CK) Then Q <= D;
End If;
End Process;
End ARCH;
```



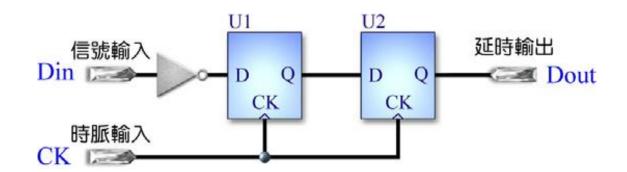
- 累積時脈延時(4/4)
 - 不使用DFF,利用VHDL直接敘述計時器

```
Library IEEE;
 Use IEÉE.std_logic_1164.all;
□Entity Debouncing2 is
     Port( Din,CK:in std_logic;
            Dout:out std_logic);
 End Debouncing2;
□Architecture ARCH of Debouncing2 is
⊟Begin
     Process(CK)
        Variable TMP:integer range 0 to 9;
        If Rising_Edge(CK) Then
If Din = '1' Then
               Din =
TMP := 0;
'= '1';
            Else
               TMP := TMP + 1;
               If TMP = 4 Then
                   TMP := 0;
Dout <= '0';
               End If:
            End If;
        End If;
     End Process:
 End ARCH;
```



- 基本延時電路延時
 - 延時輸出

PR	CL	CK	D	Q	$\overline{\overline{Q}}$
0	1	-	-	1	0
1	0	-	-	0	1
1	1	↑	0	0	1
1	1	↑	1	1	0





有限狀態機設計

· 狀態機(State Machine)

- 依據循序邏輯電路的各種狀態繪製其狀態圖或狀態表來設計電路。
- 狀態越多,電路越複雜,由於無法實作出無限多的狀態數,所以稱為有限狀態機(Finite State Machine)。
- 傳統數位邏輯電路設計裡,有限狀態機的設計程序相當複雜,而採用VHDL電路設計,有限狀態機的設計程序變得相對簡單。
- 有限狀態機可有多個輸入、多個輸出,而依其輸出與輸入是否相關,可區分為下列兩種:
 - 米利機(Mealy Machine)
 - 莫爾機(Moore Machine)

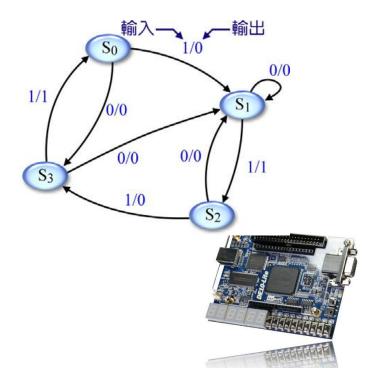


有限狀態機設計

- · 米利機(Mealy Machine)
 - 輸出狀態與目前狀態與輸入狀態有關。
 - Q: I×PS→O(I:輸入, O:輸出, PS:目前狀態Present State)

右圖中包括So~S3等4個狀態,如下說明:

- 1. So狀態時,若輸入1,則輸出0且跳至S1狀態。 若輸入0,則輸出0且跳至S3。
- 2. S1狀態時,若輸入1,則輸出1且跳至S2狀態。 若輸入0,則輸出0且維持在S1狀態。
- 3. S2狀態時,若輸入1,則輸出0且跳至S3狀態。 若輸入0,則輸出0且跳回S1狀態。
- 4. S3狀態時,若輸入1,則輸出1且跳至S0狀態。 若輸入0,則輸出0且跳至S1狀態。

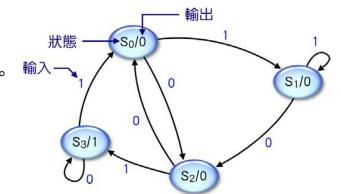


有限狀態機設計

- · 莫爾機(Moore Machine)
 - 輸出狀態只受目前狀態影響,與輸入狀態無關。
 - Q: PS→O(I: 輸入, O: 輸出, PS:目前狀態Present State)

右圖中包括So~S3等4個狀態,如下說明:

- 1. So狀態時輸出0,若輸入1,則跳至S1狀態。 若輸入0,則跳至S2狀態。
- 2. S1狀態時輸出0,若輸入1,則保持在S1狀態。 若輸入0,則跳至S2狀態。
- 3. S2狀態時輸出0,若輸入1,則跳至S3狀態。 若輸入0,則跳回So狀態。
- 4. S3狀態時輸出1,若輸入1,則跳至S0狀態。 若輸入0,則保持在S3狀態。

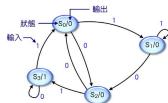




隨堂練習

- ·請使用VHDL完成下列電路,並完成紀錄,包括 VHDL Source Code、模擬波形圖。
 - a) 累積時脈延時電路(以DFF為基礎設計電路)並觀察RTL
 - b) 累積時脈延時電路(使用VHDL直接描述DFF電路)並觀察RTL
 - c) 米利機電路設計
- 1/1 0/0 S₁ 0/0 1/1 1/1 S₂ 1/0 S₂

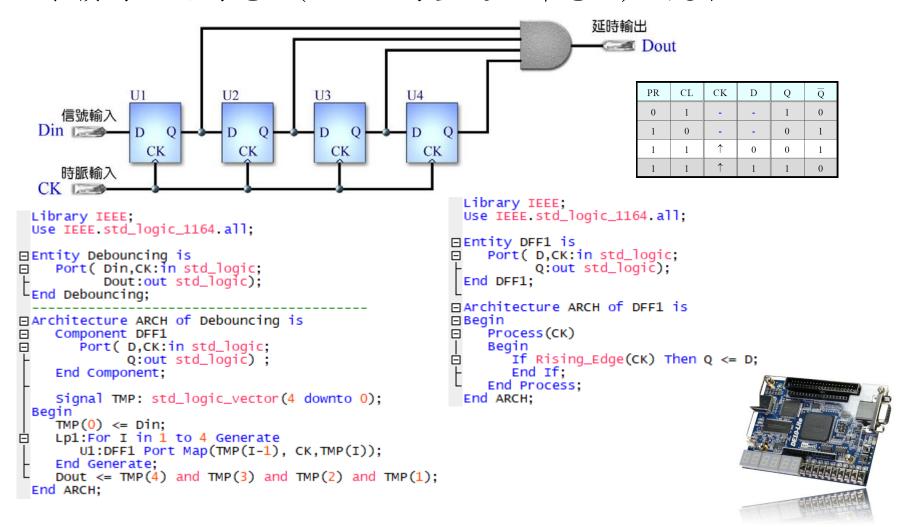
a) 莫爾機電路設計



- · 本次實驗完成後需助教確認止確,全部完成後,將專案與報告壓縮上傳EE-Class。
- · 作業X 第X組 例如:作業9 第一組

隨堂練習(一)

• 累積時脈延時電路(以DFF為基礎設計電路)並觀察RTL



隨堂練習(一)

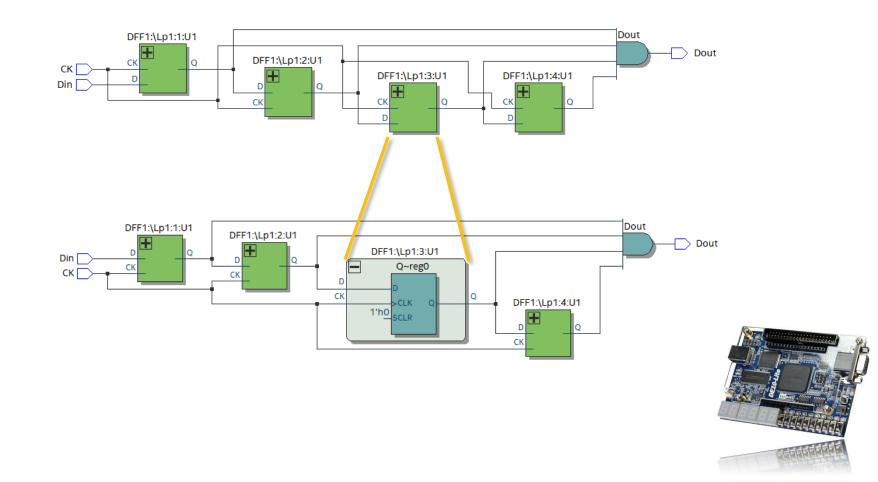
· 累積時脈延時電路(以DFF為基礎設計電路)並觀察RTL

<u></u> S	imulation W	aveform Editor -	F:/IntelFPGA_Project/DSE_EX9/EX	(9_2/EX9_2 - Debounci	ng - [EX9_2_20200518231818	3.sim.vwf (Read-Only)]		×
<u>F</u> ile	<u>E</u> dit <u>V</u> iew	<u>S</u> imulation <u>I</u>	<u>H</u> elp				Search altera.com	•
	<u>4</u> 💥 .0	- 1 <u>z</u> / 1	I W XC X X X X R ₩ ₩ ₩ 2	ia □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □				
Mast	er Time Bar:	0 ps	Pointer: 81.2	'ns Inten	val: 81.27 ns S	tart:	End:	
	Name	50.0 ns	60.0 ns	70.0 ns	80.0 ns	90.0 ns	100 _. 0 ns	^
in	ск	www	nnnynnn	www	mmmmm	mmimm	nnnnin	Π
in_	Din		 					_
out	Dout							
<	>	<		<u>'</u>		<u>'</u>	<u>'</u>	> ∨
							0% 00:0	00:00



隨堂練習(一)

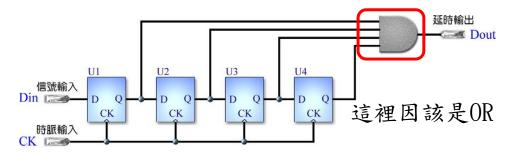
· 累積時脈延時電路(以DFF為基礎設計電路)並觀察RTL



隨堂練習(二)

• 累積時脈延時電路(使用VHDL直接描述DFF電路)並觀察RTL

```
Library IEEE;
 Use IEEE std_logic_1164.all;
□Entity Debouncing2 is
     Port( Din,CK: in std_logic;
            Dout:out std_logic);
 End Debouncing2;
□Architecture ARCH of Debouncing2 is
⊟Begin
     Process(CK)
        Variable TMP:integer range 0 to 9;
        If Rising_Edge(CK) Then
    If Din = '1' Then
               TMP := 0;
Dout <= '1';</pre>
            Else
                TMP := TMP + 1;
                If TMP = 4 Then
                   TMP := 0;
Dout <= '0';
                End If:
            End If;
        End If;
     End Process;
  End ARCH:
```

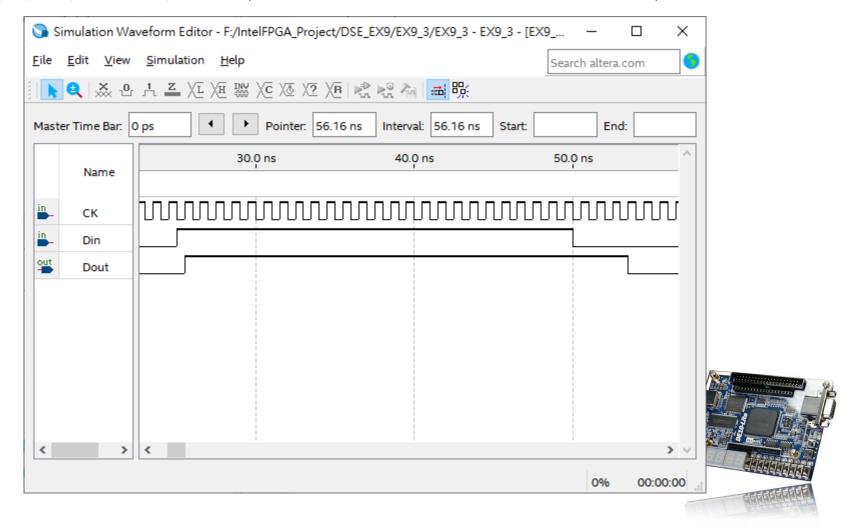


PR	CL	CK	D	Q	$\overline{\overline{Q}}$
0	1	-	-	1	0
1	0	-	-	0	1
1	1	↑	0	0	1
1	1	↑	1	1	0



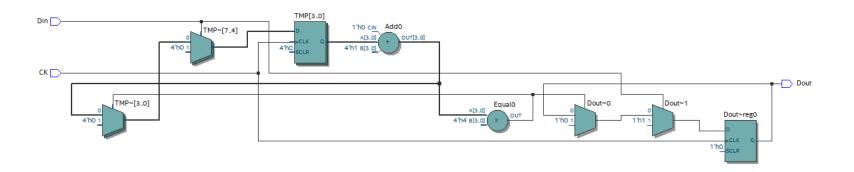
隨堂練習(二)

• 累積時脈延時電路(使用VHDL直接描述DFF電路)並觀察RTL



隨堂練習(二)

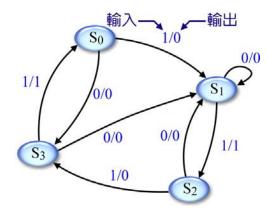
• 累積時脈延時電路(使用VHDL直接描述DFF電路)並觀察RTL





隨堂練習(三)

• 米利機電路設計(1/2)

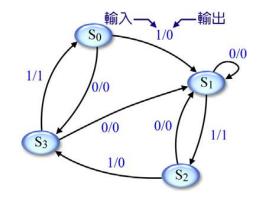


右圖中包括S0~S3等4個狀態,如下說明:

- 1. So狀態時,若輸入1,則輸出0且跳至S1狀態。 若輸入0,則輸出0且跳至S3。
- 2. S1狀態時,若輸入1,則輸出1且跳至S2狀態。 若輸入0,則輸出0且維持在S1狀態。
- 3. S2狀態時,若輸入1,則輸出0且跳至S3狀態。 若輸入0,則輸出0且跳回S1狀態。
- 4. S3狀態時,若輸入1,則輸出1且跳至S0狀態。 若輸入0,則輸出0且跳至S1狀態。

```
Library IEEE;
 Use IEEE.std_logic_1164.all;
⊟Entity Mealy_Ex is
    Port( Input, CK: in std_logic;
           PState:out integer range 0 to 9; -- 觀察用
          Output:out std Togic):
 End Mealv_Ex :
□Architecture FSM of Mealy_Ex is
    Type states is (SO, S1, S2, S3);
    Signal PS: states:=S0;
Begin 
    Process(CK)
       Variable NS: states:=S0;
       「If Rising_Edge(CK)Then -- 以CK時脈之升緣同步觸發
          Case PS is
                                 -- SO狀態的處理
              When SO =>
                 If Input='1' Then
                    NS := S1;
                    PState <= 1;
                    Output <= '0';
                    NS := S3:
                    PState <= 3;
Output <= '0';
                 End If:
                                 -- S1狀態的處理
              When S1 =>
                 If Input='1' Then
                    NS := S2:
                    PState <= 2;
Output <= '1';
                    NS := S1;
                    PState <= 1;
Output <= '0';
                 End If:
                                 -- S2狀態的處理
                 If Input='1' Then
                    NS := S3:
                    PState <= 3;
                    Output <= '0';
                    NS := S1;
                    PState <= 1;
                    Output <= '0':
                    End If:
              When S3 =>
                                 -- S3狀態的處理
                 If Input='1' Then
                    NS := S0:
                    PState <= 0:
                    Output <=
                   PState <= 3:
                    Output <= '0':
                 End If;
              When others => null:
           End Case;
       End If:
                          這裡因該是1
       PS <= NS:
    End Process:
 end FSM:
```

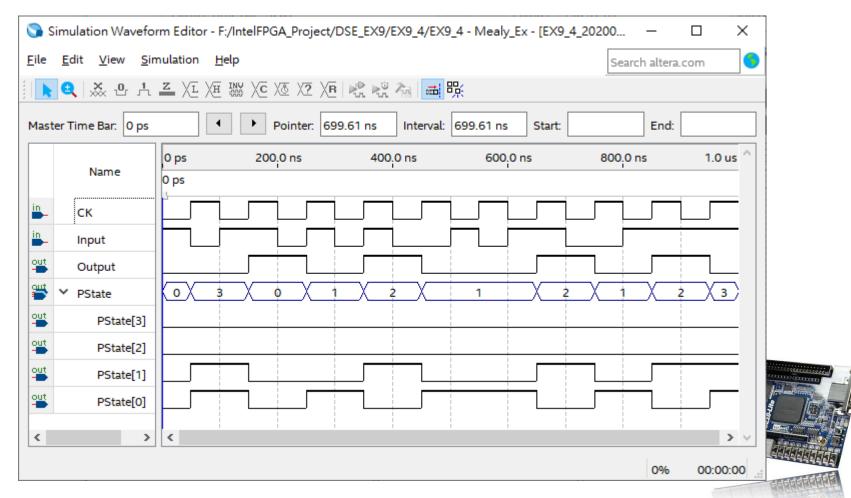
隨堂練習(三)



End Time:1us Grid Size:100ns CK週期:100ns

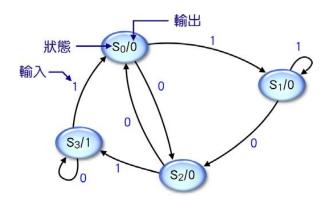
Input:Every half grid interval

• 米利機電路設計(2/2)



隨堂練習(四)

• 莫爾機電路設計(1/2)



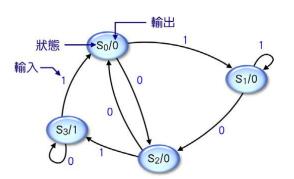
右圖中包括So~S3等4個狀態,如下說明:

- S₀狀態時輸出0,若輸入1,則跳至S₁狀態。
 若輸入0,則跳至S₂狀態。
- 2. S1狀態時輸出0,若輸入1,則保持在S1狀態若輸入0,則跳至S2狀態。
- 3. S2狀態時輸出0,若輸入1,則跳至S3狀態。 若輸入0,則跳回So狀態。
- 4. S₃狀態時輸出1,若輸入1,則跳至S₀狀態。 若輸入0,則保持在S₃狀態。

```
Library IEEE;
 Use IEEE.std_logic_1164.all;
□Entity Moore_Ex is
    Port( Input,CK:in std_logic;
          PState:out integer range 0 to 9; -- 觀察用
          Output:out std_logic);
 End Moore Ex:
□ Architecture FSM of Moore_Ex is
    Type states is (SO, S1, S2, S3);
    Signal PS: states:=S0;
    Output <='1' when PS=S3 Else '0':
    Process(CK)
       Variable NS: states:=S0;
       Case PS is
             when SO => -- SO狀態的處理
If Input='1' Then
                    NS := S1;
                    PState <= 1:
                 Else
                    NS := S2:
                    PState <= 2:
                 End If:
             when S1 => -- S1狀態的處理
If Input='1' Then
                    NS := S1:
                    PState <= 1:
                 Else
                    NS := S2;
                    PState <= 2:
                 End If:
              When S2 => -- S2狀態的處理
                 If Input='1' Then
                    NS := S3:
                    PState <= 3;
                    NS := S0;
                    PState <= 0:
                 End If:
             when S3 => -- S3狀態的處理
If Input='1' Then
                    NS := SO:
                    PState <= 0;
                 Else
                    NS := S3;
                    PState <= 3;
                 End If;
              When others => null;
          End Case:
       End If:
       PS <= NS:
    End Process:
 End FSM:
```

隨堂練習(四)

• 莫爾機電路設計(2/2)



End Time:1us Grid Size:100ns CK週期:100ns

Input:Every half grid interval

