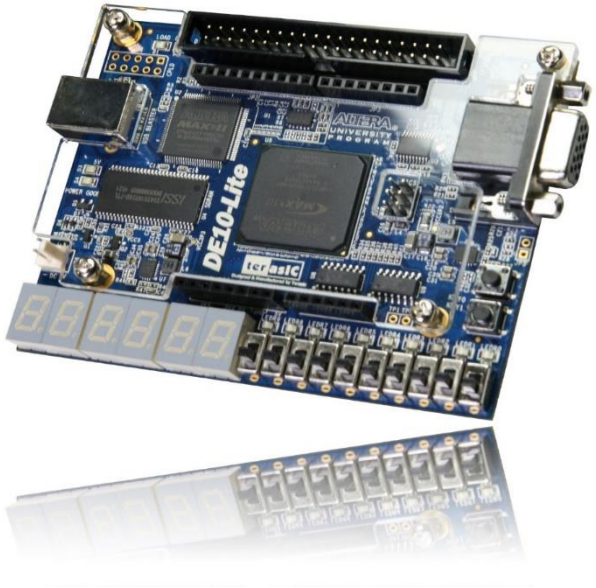
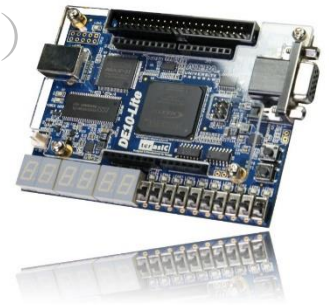


VHDL與循序邏輯電路設計

CSIE MIAT Lab.

Outline

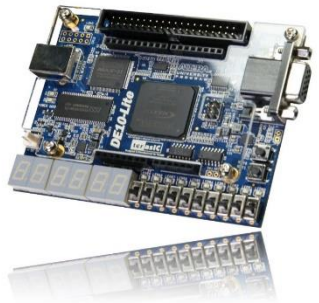
- Process結構
- If-Then-Else 敘述
  - 基本語法格式
  - 多重判斷條件
  - 巢狀結構敘述
- 時脈觸發
- Case-When 敘述
- Loop 敘述
  - For-Loop敘述
  - While-Loop敘述
  - Exit When
- Exit 敘述
- 正反器設計
  - RS 正反器
  - JK 正反器
  - T 型正反器
  - D 型正反器
- Generic 敘述
- 計數器設計
  - 二進位計數器設計(隨堂練習)
  - 二進位上/下數計數器設計(隨堂練習)
  - 除N計數器設計(隨堂練習)
  - BCD計數器設計(隨堂練習)
  - BCD加法器設計(隨堂練習)
  - 移位暫存器設計(隨堂練習)



Process結構(1/5)

- 時序邏輯設計的重要介面。
- 可使用一個或多個Process結構。
- 每個Process結構都是同時動作的。

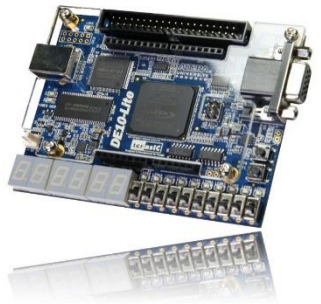
```
[標籤:] process(敏感信號列)
    宣告變數;
Begin
    描述電路功能
    (時序性描述;)
End process [標籤];
```



Process結構(1/5)

- 標籤:
  - 標籤用來標示電路，通常是使用有意義的文字，例如counter10、clock_div..等
  - End Process後的標籤可以省略，如果有使用標前，則必須與開頭的標籤一致。
  - 整個電路設計若只有一個Procee結構，可不放置標籤。

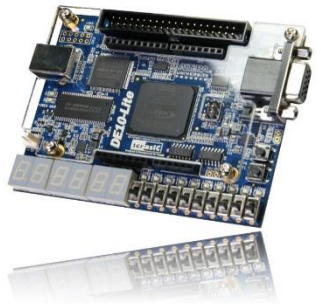
```
[標籤:] process(敏感信號列)
    宣告變數;
Begin
    描述電路功能
    (時序性描述;)
End process [標籤];
```



Process結構(1/5)

- 敏感信號列(Sensitivity List)
  - 可**觸發**此Process 動作的信號。
  - 若超過一個敏感信號，信號與信號之間須以**逗點**分隔。
  - Process 結構類似微處理器裡的**中斷**(Interrupt)，並非隨時都在工作，而是在任一個敏感信號有所變化時，才會執行Processs 內描述的動作。

```
[標籤:] process(敏感信號列)
    宣告變數;
Begin
    描述電路功能
    (時序性描述;)
End process [標籤];
```

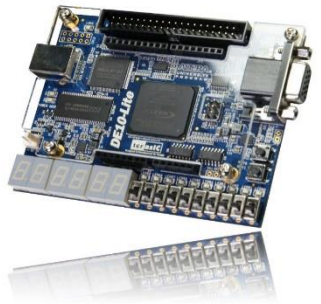


Process結構(1/5)

- 變數的宣告
  - 在 Process與Begin之間，用來宣告此Process所使用的信號及變數。而宣告的格式如下：

```
Variable 變數名稱1, 變數名稱2,… : 資料型態 [:=初值];
        :
        :
```

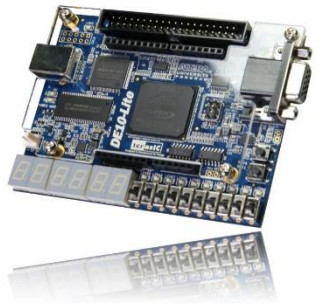
```
[標籤:] process(敏感信號列)
    宣告變數;
Begin
    描述電路功能
    (時序性描述;)
End process [標籤];
```



Process結構(1/5)

- 描述電路功能
  - 在 Begin與End之間，為此Process 主要的部分，可利用時序性語法，描述電路的動作，而其動作是按描述的順序。

```
[標籤:] process(敏感信號列)
    宣告變數;
Begin
    描述電路功能
    (時序性描述;)
End process [標籤];
```



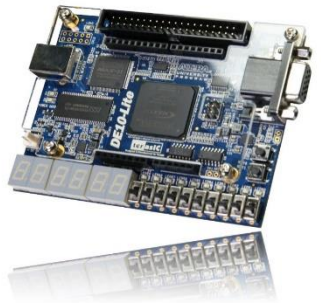
If-Then-Else 敘述(1/4)

- 基本語法格式

```
If 判斷條件Then
  電路描述 1;
Else
  電路描述 2;
End If;
```

- 若其中有處理程序為不做人和處理時處理任何，應使用「null;」。

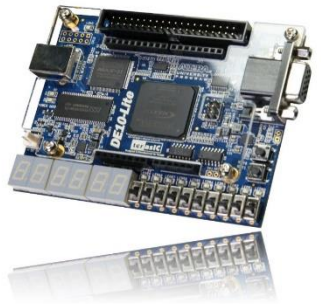
```
If 判斷條件Then
  電路描述;
Else
  null;
End If;
```



If-Then-Else 敘述(1/4)

- 基本語法

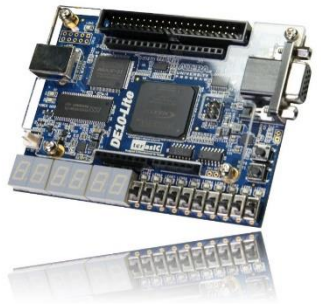
```
Library IEEE;
Use IEEE.std_logic_1164.all;
Use IEEE.std_logic_unsigned.all;
Entity ADDSUB is
    Port(A, B:in std_logic_vector(7 downto 0);
        Cin,ADDSUB:in std_logic;
        Co:out std_logic;
        Results:out std_logic_vector(7 downto 0));
End ADDSUB;
Architecture ARCH of ADDSUB is
Begin
    Process(A,B,Cin,ADDSUB)
        Variable TMP: std_logic_vector(8 downto 0);
    Begin
        If ADDSUB = '1' Then
            TMP := '0' & A + B + Cin;
        Else
            TMP := '0' & A - B - Cin;
        End If;
        Results <= TMP(7 downto 0);
        Co <= TMP(8);
     End Process;
End ARCH;
```



If-Then-Else 敘述(1/4)

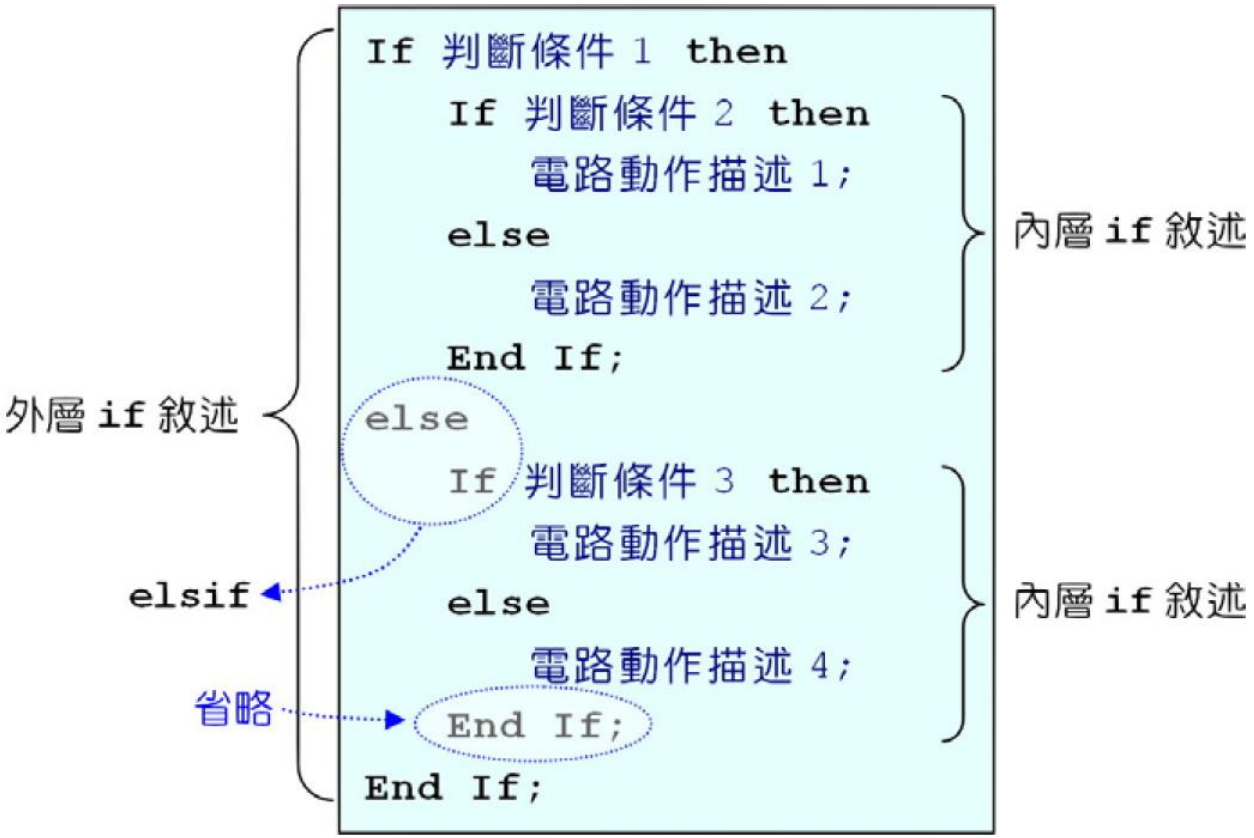
- 多重判斷條件

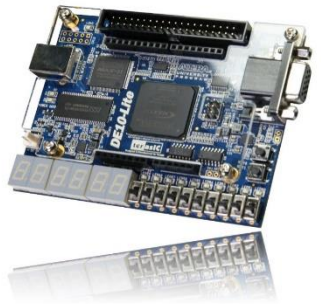
```
If 判斷條件1 Then
  電路描述 1;
Elsif 判斷條件2 Then
  電路描述 2;
    : :
Else
  電路描述 n;
End If;
```



If-Then-Else 敘述(1/4)

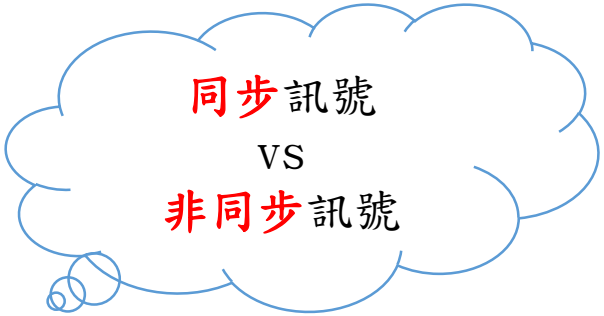
- 巢狀結構敘述





時脈觸發

- 時鐘脈波(簡稱時脈)
- **正緣**觸發與**負緣**觸發

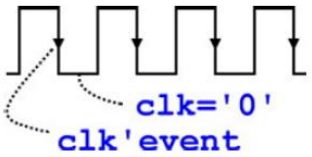


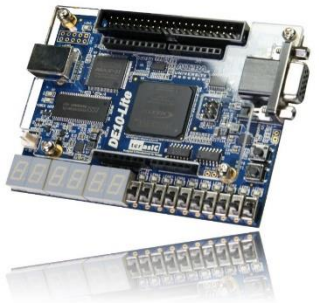
正緣觸發

```
Clk0:Process (clk)
Begin
    If clk'event and clk='1' Then
        電路動作描述;
    End If;
End Process;
```

負緣觸發

```
Clk0:Process (clk)
Begin
    If clk'event and clk='0' Then
        電路動作描述;
    End If;
End Process;
```

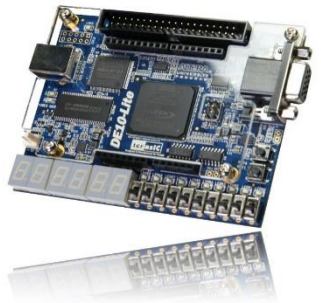




Case-When 敘述

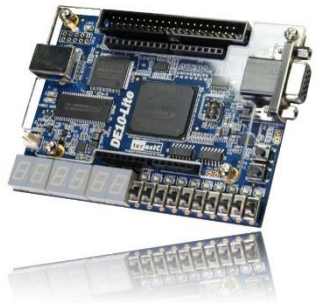
- 屬於循序性敘述
- 常被使用在狀態機

```
Case 控制項目is
    When 值1 或 信號1 =>
        電路描述 1;
            : :
    When 值2 或 信號2 =>
        電路描述 2;
            : :
    When others =>
        電路描述 n;
            : :
End Case;
```



Loop 敘述

- 控制迴圈結構，用於描述重複的電路動作，分為:
  - For-Loop
  - While-Loop
  - Loop
- 使用Exit When敘述，或結合If-Then-Else與Exit跳出迴圈。

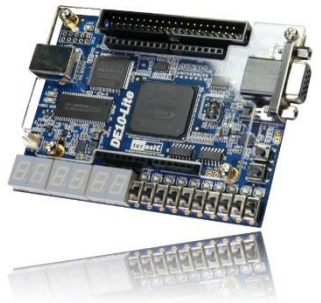


Loop 敘述

- For-Loop 敘述

```
[標籤:] For 控制變數in 範圍 Loop
        電路描述 1;
                  : :
End Loop [標籤];
```

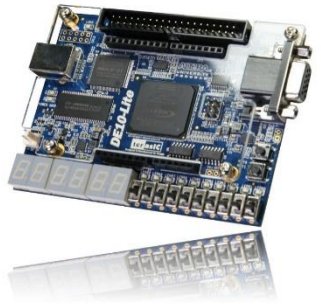
  - 標籤 ： 增加電路描述的可讀性，可有可無。
  - 控制變數 ： 不須宣告(ex. i、j、k、cnt …)
  - 範圍 :重複執行迴圈
    - 數值範圍:例如「0 to 7」
    - 列舉資料型態



Loop 敘述

- For-Loop 敘述範例

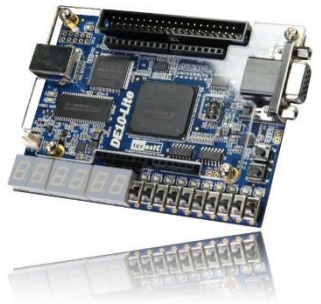
```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity Parity11 is
    Port( Din:in std_logic_vector(0 to 10);
          ODD:out std_logic);
End Parity11;
Architecture ARCH of Parity11 is
Begin
    Process(Din)
    Variable TMP: Boolean;
Begin
    TMP := false;
    For I in 0 to Din'length -1 Loop
        If Din(I) = '1' Then
        TMP := not TMP;
        End If;
    End Loop;
    If TMP Then
        ODD <= '1';
    Else
    ODD <= '0';
    End If;
End Process;
End ARCH;
```



Loop 敘述

- For-Loop 敘述範例一

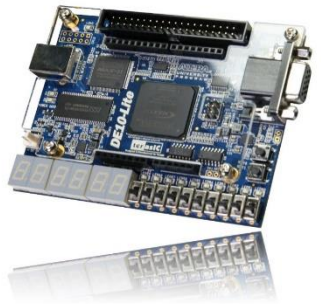
```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity Parity11 is
    Port( Din:in std_logic_vector(0 to 10);
          ODD:out std_logic);
End Parity11;
Architecture ARCH of Parity11 is
Begin
    Process(Din)
    Variable TMP: Boolean;
Begin
    TMP := false;
    For I in 0 to Din'length -1 Loop
        If Din(I) = '1' Then
        TMP := not TMP;
        End If;
    End Loop;
    If TMP Then
        ODD <= '1';
    Else
    ODD <= '0';
    End If;
End Process;
End ARCH;
```



Loop 敘述

- For-Loop 敘述範例二

```
Architecture ARCH of Loop_EX
    Type week is Sunday,Monday,Tuesday,Thursday,Friday,Saturday;
Begin
    : :
    Process (A)
Begin
    For day in week Loop
    Case day is
        When Sunday =>
            電路描述 1;
                : :
        When Monday =>
            電路描述 2;
                : :
```



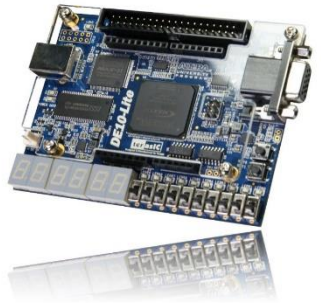
Loop 敘述

- While-Loop 敘述

  提供先決條件判斷的迴圈敘述，只要判斷條件成立，則執行迴圈內的電路描述，其語法格式如下：

```
[標籤:] While 判斷條件 Loop
          電路描述 1;
          : :
End Loop [標籤];
```

基本上，While-Loop 敘述是用在撰寫測試平台(Test Bench)，產生激勵信號，以進行電路模擬，而不適用於合成電路。

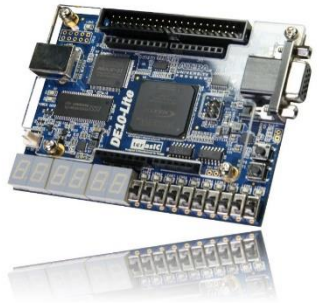


Loop 敘述

- Exit When 敘述

  在 Loop 迴圈裡，可利用Exit When 敘述跳出迴圈，其語法格式如下：

```
[標籤:] For 控制變數 in 範圍 Loop
          電路描述 1;
                : :
          Exit When 判斷條件;
End Loop [標籤];
```

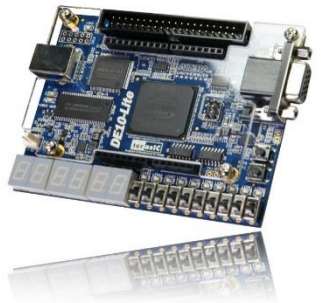


Loop 敘述

- Exit 敘述

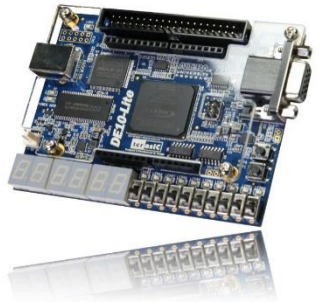
  對於無窮盡的 Loop 迴圈，可利用If-Then 敘述與Exit 敘述跳出迴圈，其語法格式如下：

```
Loop
     電路描述 1;
      : :
    If 判斷條件Then Exit;
End Loop;
```



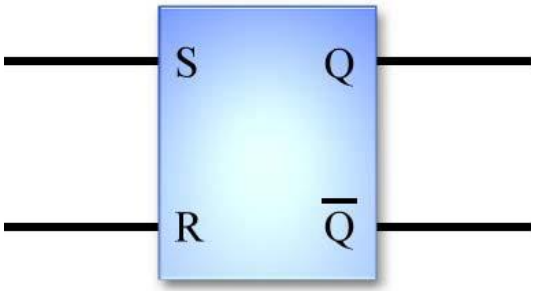
正反器設計(Flip-Flop)

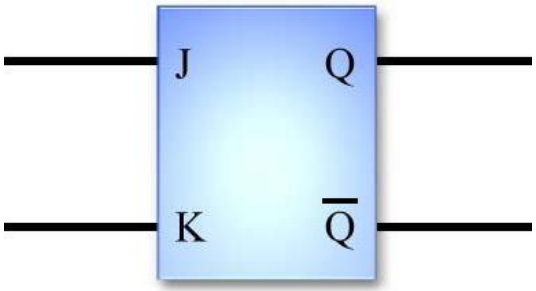
- 正反器(Flip-Flop)常被使用在各種循序邏輯的電路設計上，分為以下4種:
  - RS 正反器
  - JK 正反器
  - D 型正反器
  - T 型正反器



正反器設計(Flip-Flop)

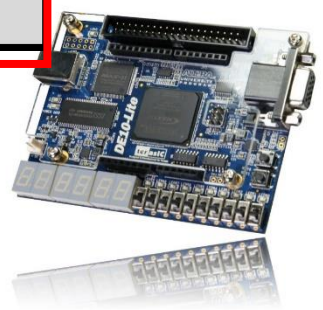
- RS正反器與JK正反器(簡單版本)





S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	不允許	不允許

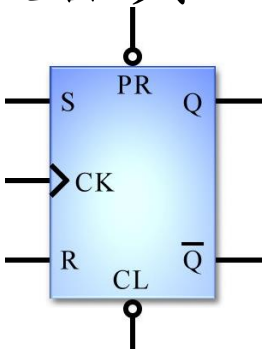
J	K	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	$\overline{Q}$	Q



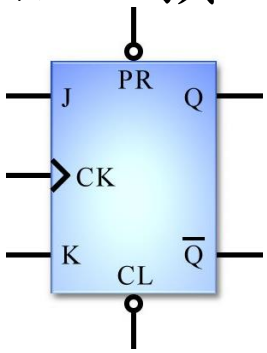
- JK正反器幾乎可以取代RS正反器。

正反器設計(Flip-Flop)

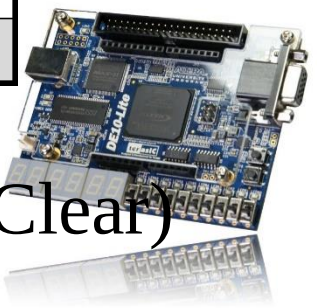
- RS正反器與JK正反器(加入時脈、預設、清除)



PR	CL	CK	S	R	Q	$\overline{Q}$
0	1	-	-	-	1	0
1	0	-	-	-	0	1
1	1	↑	0	0	Q	$\overline{Q}$
1	1	↑	0	1	0	1
1	1	↑	1	0	1	0
1	1	↑	1	1	X	X



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



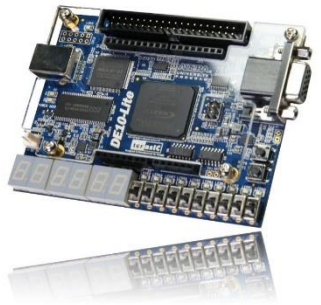
- CK:時脈(Clock) / PR:預設(Preset)/ CL:清除(Clear)

正反器設計(Flip-Flop)

- VHDL for RS正反器

```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity RSFF1 is
    Port( PR,CL,CK,R,S:in std_logic;
          Q,Qbar:out std_logic);
End RSFF1;
Architecture ARCH of RSFF1 is
Begin
    Process(PR,CL,CK)
    Variable TMP:std_logic;
    Begin
        If PR='0' Then TMP := '1';
        Elsif CL='0' Then TMP := '0';
        Elsif Rising_edge(CK) Then
            If S='0' and R='1' Then TMP := '0';
            Elsif S='1' and R='0' Then TMP := '1';
            Elsif S='1' and R='1' Then TMP := 'X';
            Else null;
            End If;
        End If;
        Q <= TMP;
        Qbar <= not TMP;
    End Process;
End ARCH;
```

PR	CL	CK	S	R	Q	$\overline{Q}$
0	1	-	-	-	1	0
1	0	-	-	-	0	1
1	1	↑	0	0	Q	$\overline{Q}$
1	1	↑	0	1	0	1
1	1	↑	1	0	1	0
1	1	↑	1	1	X	X

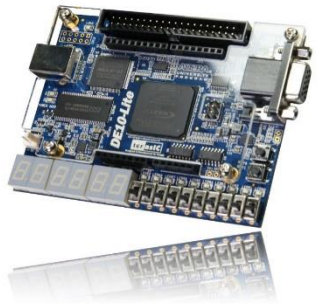


正反器設計(Flip-Flop)

- VHDL for JK正反器

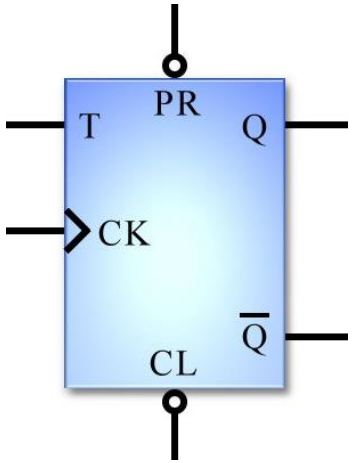
```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity JKFF1 is
    Port( PR,CL,CK,J,K:in std_logic;
          Q,Qbar:out std_logic);
End JKFF1;
Architecture ARCH of JKFF1 is
Begin
    Process(PR,CL,CK)
    Variable TMP:std_logic;
    Begin
        If PR='0' Then TMP := '1';
        Elsif CL='0' Then TMP := '0';
        Elsif Rising_edge(CK) Then
            If J='0' and K='1' Then TMP := '0';
            Elsif J='1' and K='0' Then TMP := '1';
            Elsif J='1' and K='1' Then TMP := not TMP;
            Else null;
            End If;
        End If;
        Q <= TMP;
        Qbar <= not TMP;
    End Process;
End ARCH;
```

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

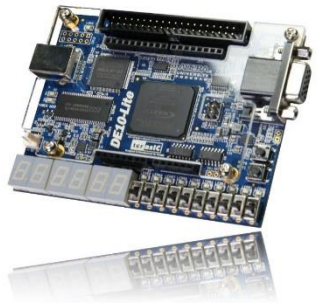


正反器設計(Flip-Flop)

- T型正反器



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

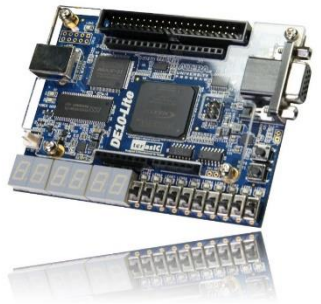


正反器設計(Flip-Flop)

- VHDL for T型正反器

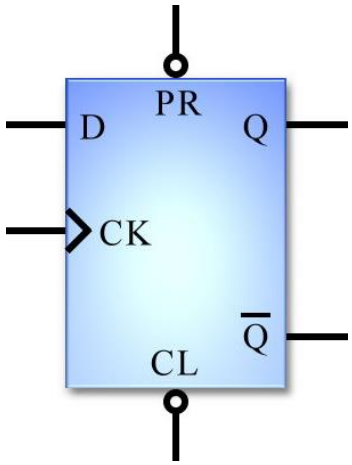
PR	CL	CK	T	Q	$\overline{Q}$
0	1	-	-	1	0
1	0	-	-	0	1
1	1	↑	0	Q	$\overline{Q}$
1	1	↑	1	$\overline{Q}$	Q

```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity TFF1 is
    Port( PR,CL,CK,T:in std_logic;
          Q,Qbar:out std_logic);
End TFF1;
Architecture ARCH of TFF1 is
Begin
    Process(PR,CL,CK)
        Variable TMP:std_logic;
    Begin
        If PR='0' Then TMP := '1';
        Elsif CL='0' Then TMP := '0';
        Elsif Rising_edge(CK) Then
            If T='1' Then TMP := not TMP;
            Else null;
            End If;
        End If;
        Q <= TMP;
        Qbar <= not TMP;
    End Process;
End ARCH;
```

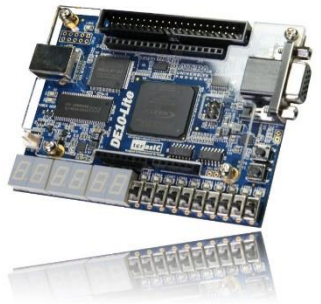


正反器設計(Flip-Flop)

- **D型正反器**



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

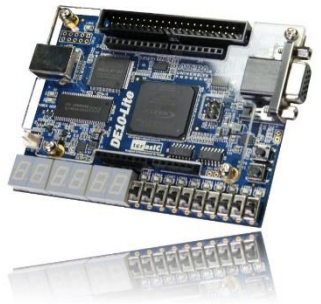


正反器設計(Flip-Flop)

- VHDL for D型正反器

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

```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity DFF1 is
     Port( PR,CL,CK,D:in std_logic;
           Q,Qbar:out std_logic);
End DFF1;
Architecture ARCH of DFF1 is
Begin
     Process(PR,CL,CK)
     Variable TMP:std_logic;
     Begin
          If PR='0' Then TMP := '1';
          Elsif CL='0' Then TMP := '0';
          Elsif Rising_edge(CK) Then
               If D='1' Then TMP := '1';
               Else TMP := '0';
               End If;
          End If;
          Q <= TMP;
          Qbar <= not TMP;
     End Process;
End ARCH;
```



正反器設計(Flip-Flop)

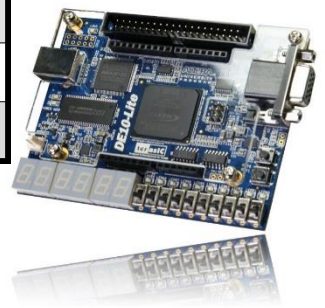
- 各種正反器比較

PR	CL	CK	S	R	Q	$\overline{Q}$
0	1	-	-	-	1	0
1	0	-	-	-	0	1
1	1	↑	0	0	Q	$\overline{Q}$
1	1	↑	0	1	0	1
1	1	↑	1	0	1	0
1	1	↑	1	1	X	X

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

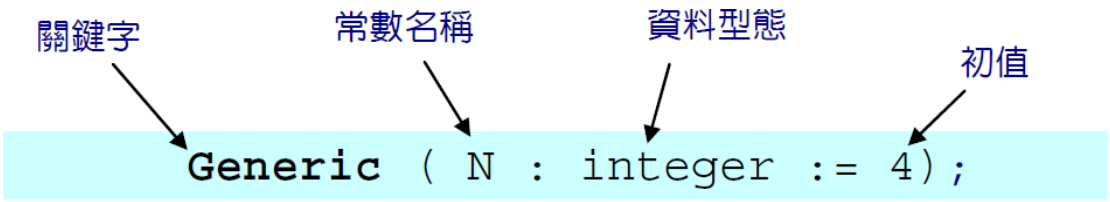
PR	CL	CK	T	Q	$\overline{Q}$
0	1	-	-	1	0
1	0	-	-	0	1
1	1	↑	0	Q	$\overline{Q}$
1	1	↑	1	$\overline{Q}$	Q

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



Generic 敘述

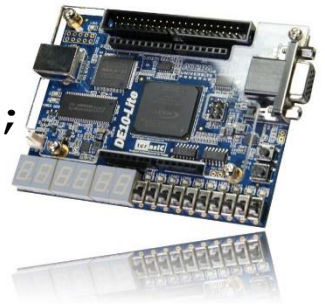
- Generic 敘述的功能是將參數傳入設計裡，其格式如下：



- Generic範例:

  Q為4bit匯流排，只要修改N，就可以修改匯流排寬度。

```
Entity MOD1 is
     Generic(N :integer :=4);
     Port( CLR,CK,UD:in std_logic;
           Q:out std_logic_vector(N-1 downto 0));
End MOD1;
```



Generic 敘述

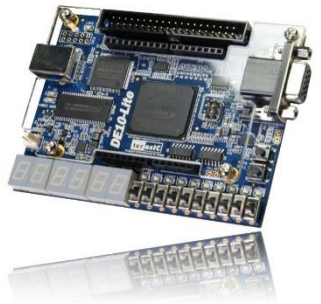
- 當作為Component電路時，需在Port Map敘述前使用Generic Map敘述將參數引入Component其格式如下：

Generic Map (參數)

- ex:

  欲設定上升時間為1ns、下降時間為1ns，如下：

```
Generic Map(tRise => 1 ns, tFall => 1 ns)
```

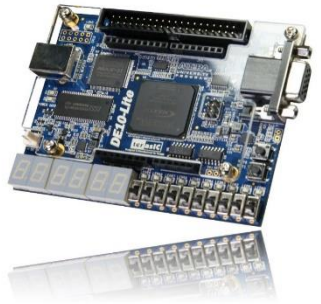


Generic 敘述

- Generic 敘述範例

DEF1電路

```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity DFF1 is
    Generic (wid:positive);
    Port( CL,CK:in std_logic;
          D:in std_logic_vector( wid-1 downto 0);
          Q:out std_logic_vector( wid-1 downto 0));
    End DFF1 ;
Architecture ARCH of DFF1 is
Begin
    Process(CL,CK)
    Variable TMP : std_logic_vector( wid-1 downto 0);
    Begin
        If CL='1' Then
            TMP := (others => '0') ;
        Elsif Rising_Edge(CK) Then
            For I in TMP'range Loop
                TMP(I) := D(I);
            End Loop;
        End If;
        Q <= TMP;
    End Process;
End ARCH;
```

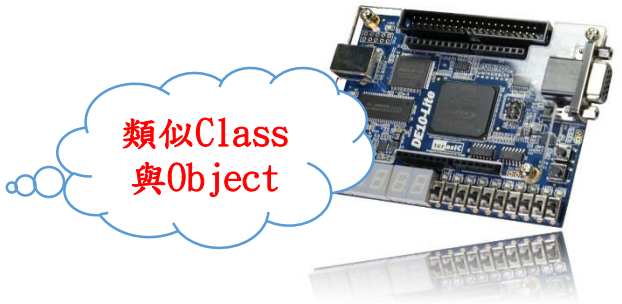


Generic 敘述

- Generic 敘述範例(續)

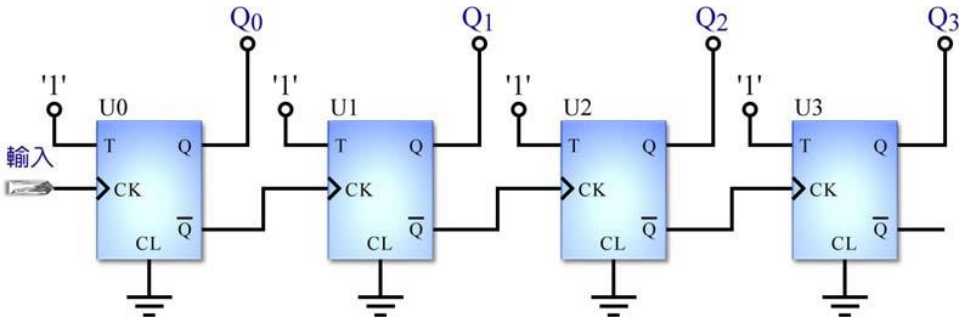
引用DEF1零件

```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity NewDesign is
    Port( CL,CK:in std_logic;
          Din:in std_logic_vector(7 downto 0);
          Dout:out std_logic_vector(7 downto 0));
End NewDesign;
Architecture ARCH of NewDesign is
    Component DFF1
        Generic (wid:positive);
        Port( CL,CK:in std_logic;
              D:in std_logic_vector( wid-1 downto 0);
              Q:out std_logic_vector( wid-1 downto 0) );
        End Component ;
    Constant wid8: positive := 8;
Begin
    DFF8: DFF1 Generic Map(wid8)
    Port Map (CL,CK,Din,Dout);
End ARCH;
```



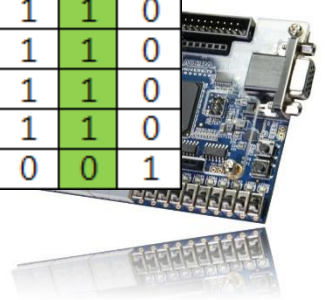
計數器設計

- 二進位計數器設計(隨堂練習)



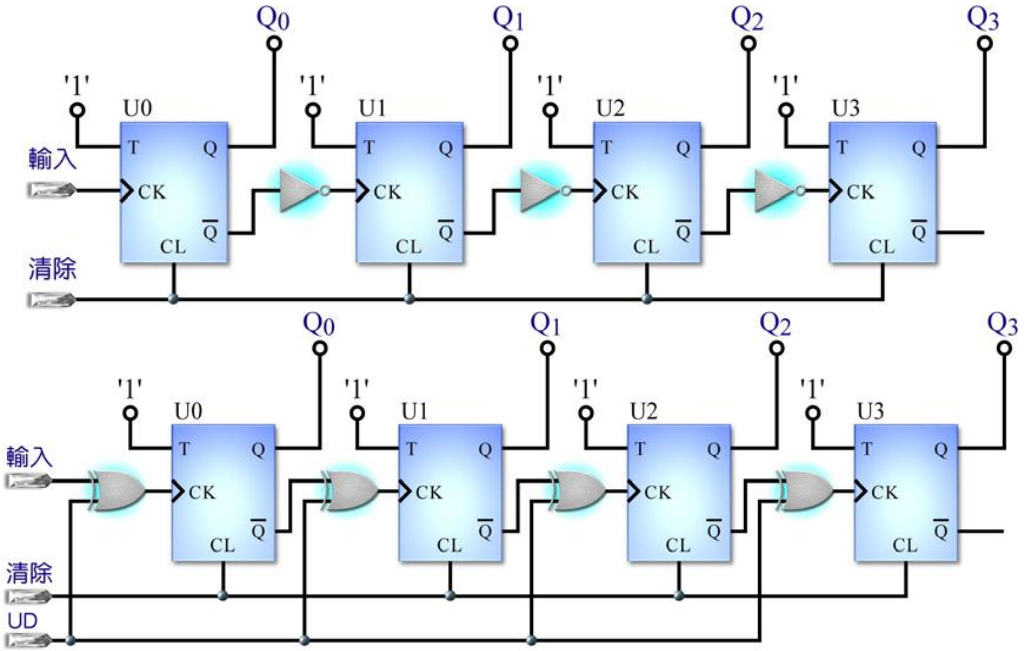
PR	CL	CK	T	Q	$\overline{Q}$
0	1	-	-	1	0
1	0	-	-	0	1
1	1	↑	0	Q	$\overline{Q}$
1	1	↑	1	$\overline{Q}$	Q

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	0
1	0	1	0	1	1	0	0	1

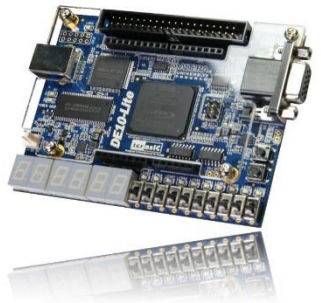


計數器設計

- 二進位上/下數計數器設計(隨堂練習)

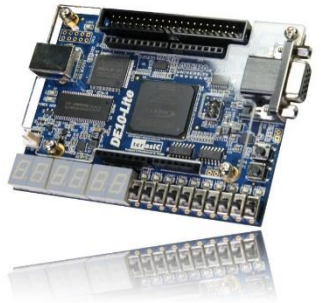


PR	CL	CK	T	Q	$\overline{Q}$
0	1	-	-	1	0
1	0	-	-	0	1
1	1	↑	0	Q	$\overline{Q}$
1	1	↑	1	$\overline{Q}$	Q



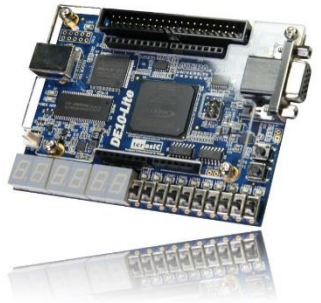
計數器設計

- 除N計數器設計(隨堂練習)



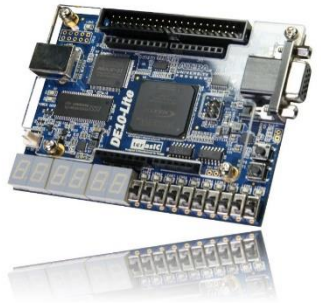
計數器設計

- BCD計數器設計(隨堂練習)



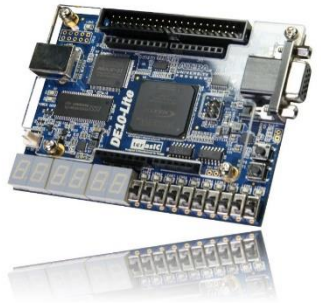
計數器設計

- BCD加法器設計(隨堂練習)



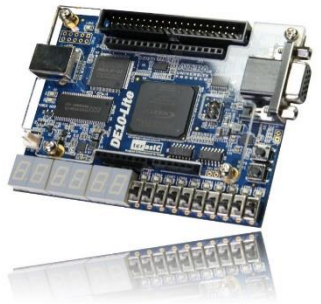
計數器設計

- 移位暫存器設計(隨堂練習)



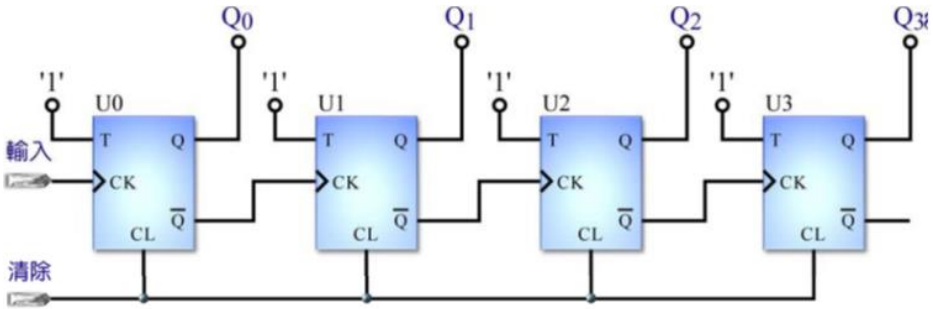
隨堂練習

- 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）二進位計數器設計
  - b）二進位上/下數計數器設計
  - c）除N計數器設計
  - d）BCD計數器設計
  - e）BCD加法器設計
  - f）移位暫存器設計
- 本次實驗完成後需助教確認正確，全部完成後，將專案與報告壓縮上傳EE-Class。
- Lecture8_組別XX.ZIP



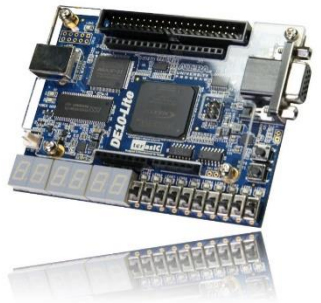
隨堂練習(一)

- 二進位計數器設計(1/2)



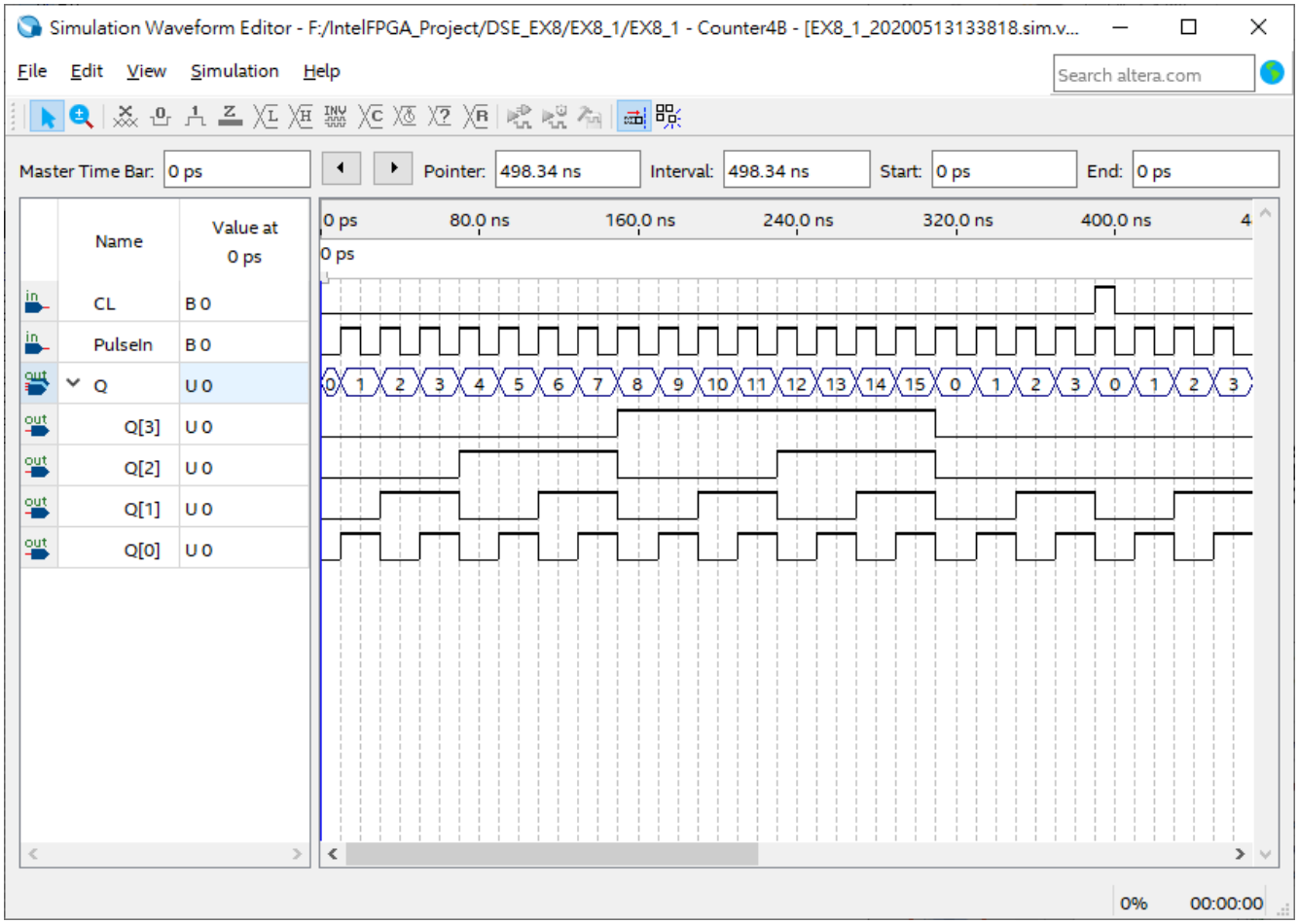

```
Library IEEE;
Use IEEE.std_logic_1164.all;

Entity Counter4B is
   Port( CL, PulseIn:in std_logic;
         Q:out std_logic_vector(3 downto 0));
End Counter4B;

Architecture ARCH of Counter4B is
   Component DFF1
      Port( CL,CK,T:in std_logic;
            Q,Qbar:out std_logic);
   End Component ;
   Signal TMP: std_logic_vector(4 downto 0) ;
   Begin
      TMP(0) <= PulseIn;
      LP1: For I in 0 to 3 Generate
         U : DFF1 Port Map (CL, TMP(I), '1', Q(I), TMP(I+1));
      End Generate;
End ARCH;
```

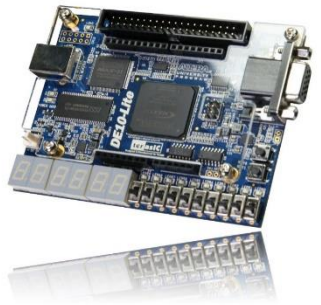
```
Library IEEE;
Use IEEE.std_logic_1164.all;

Entity DFF1 is
   Port( CL,CK,T:in std_logic;
         Q,Qbar:out std_logic );
End DFF1;

Architecture ARCH of DFF1 is
Begin
   Process(CL,CK)
      Variable TMP:std_logic;
      Begin
         If CL='1' Then TMP := '0';
         Elsif Rising_Edge(CK) Then
            If T='1' Then TMP := not TMP;
            Else null;
            End If;
         End If;
         Q <= TMP;
         Qbar <= not TMP;
   End Process;
End ARCH;
```



隨堂練習(一)

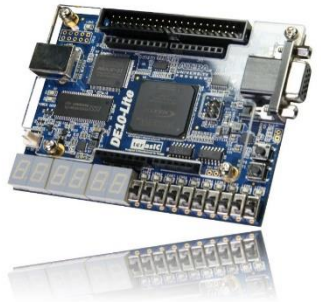
- 二進位計數器設計(2/2)





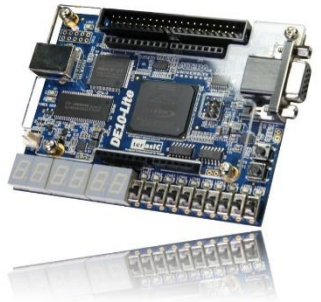
隨堂練習(二)

- 二進位上/下數計數器設計(1/2)



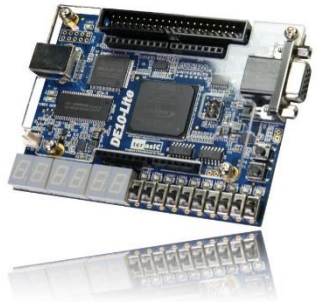
隨堂練習(二)

- 二進位上/下數計數器設計(1/2)



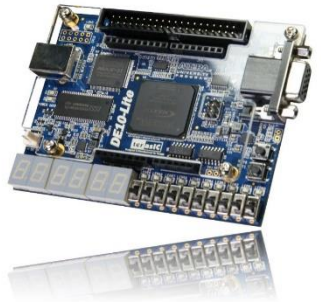
隨堂練習(三)

- 除N計數器設計(1/2)



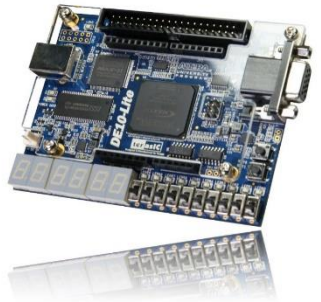
隨堂練習(三)

- 除N計數器設計(2/2)



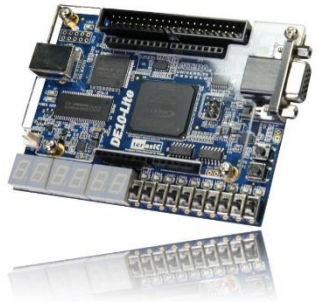
隨堂練習(四)

- BCD計數器設計(1/2)



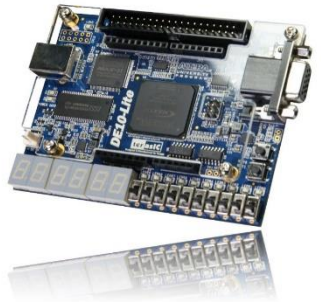
隨堂練習(四)

- BCD計數器設計(2/2)



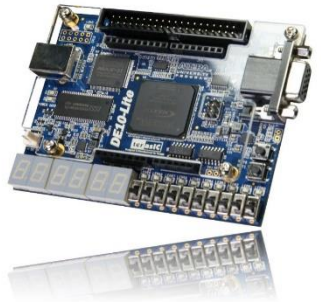
隨堂練習(五)

- BCD加法器設計(1/2)



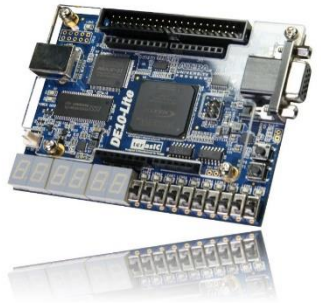
隨堂練習(五)

- BCD加法器設計(2/2)



隨堂練習(六)

- 移位暫存器設計(1/2)



隨堂練習(六)

- 移位暫存器設計(2/2)