VHDL文法の習得と、これまで・今回・今後の内容

Quartus IIの資料

VHDL文法の習得

Quartus II プロジェクト

VHDL記述
（デザインファイル）

工程

基本構文

簡単な例

論理式記述-組み合わせ回路
機能記述-組み合わせ回路
機能記述-レジスタ
複合回路

ゲートレベル②

RTL

これまでの内容

今回の内容

今後の内容

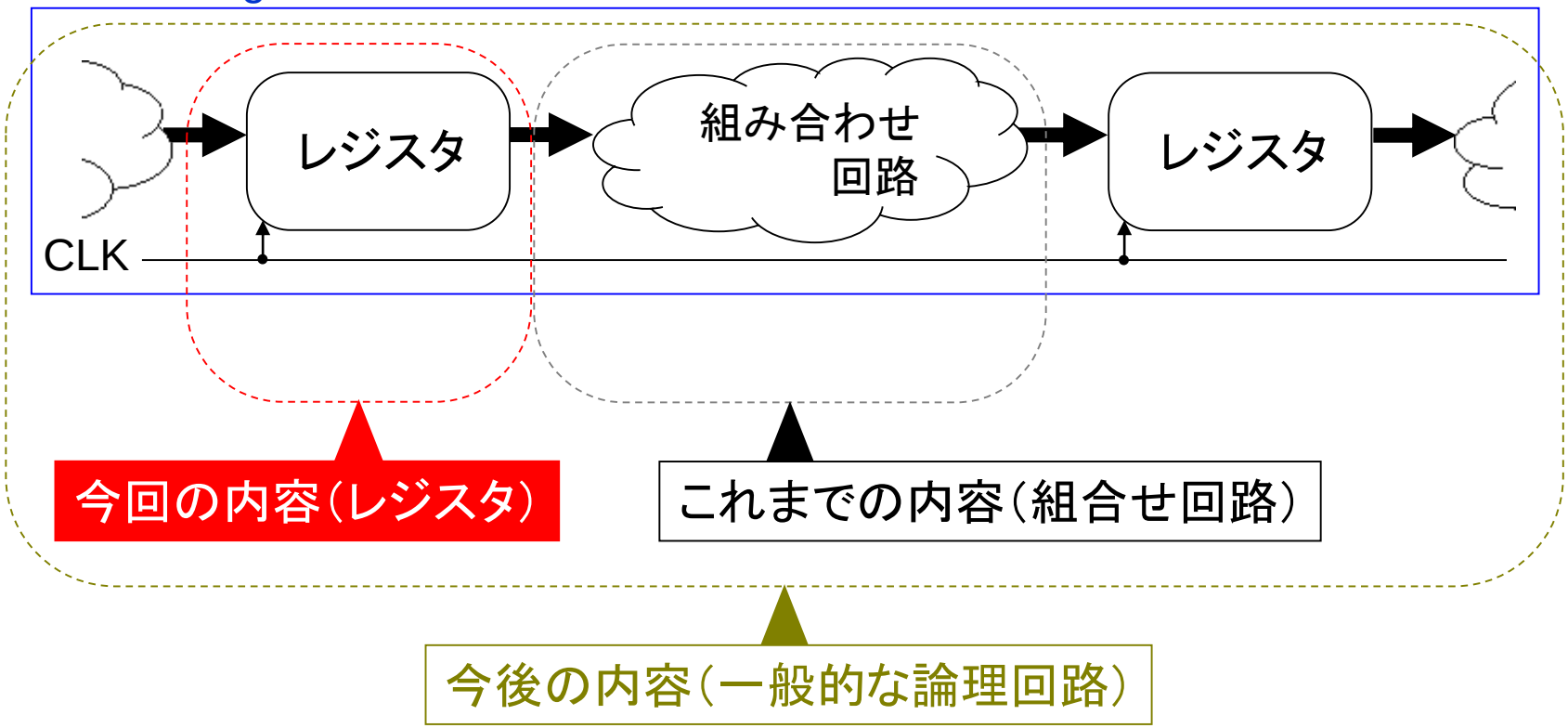
高度な例

課題を設定して
進めていく。

（これまでの）組み合わせ回路
＋
（今回の）レジスタ（記憶要素、ある種の順序回路）
↓
（今後の）一般的な論理回路（一般的な順序回路）

RTLでの一般的回路機能図と、これまで・今回・今後の内容

RTL（Register Transfer Level:レジスタ転送レベル）での一般的回路機能図



（例）レジスタの機能記述①　基本レジスタ

●レジスタの機能記述
- ・ポジティブ／ネガティブ　エッジトリガ
- ・記憶ビット数
- ・付加機能
- ・追加出力

基本レジスタ

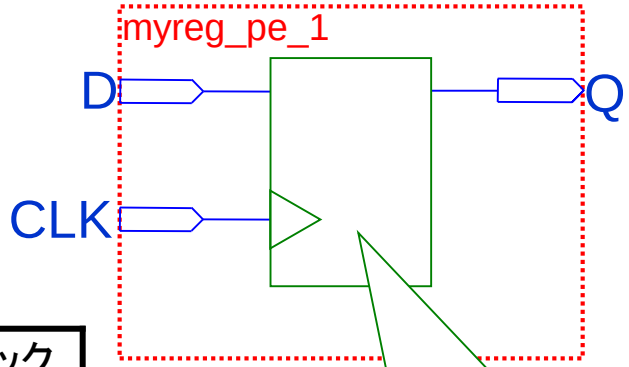
- ・ポジティブエッジトリガ
- ・1-bit記憶
- ・付加機能なし
- ・追加出力なし

のレジスタ

入出力ポートの記述

```
port(
   CLK : in std_logic;
   D : in std_logic;
   Q : out std_logic
);
```

myreg_pe_1.vhd



回路機能の記述

```
process(CLK)
begin
  if(CLK'event and CLK='1')then
     Q <= D;
  end if;
end process;
```

要・条件チェック
→if文

チェックする信号
→CLK

条件内容
→ポジティブ
　エッジかどうか

入力		出力
CLK	D	Q
↑	0	0
	1	1
↑以外	0	変化なし
	1	

【注】エンティティ名の
「**_pe**」はポジティブエッジトリガを、「**_1**」は1-bit記憶を意図している

if文 （順序回路における場合）

●if文

条件を順次チェックし、対応する信号代入文に従って、出力を決定する。

書式

```
if(条件1)then
   信号代入文;
elsif(条件2)then
   信号代入文;
    ⋮
elsif(条件n)then
   信号代入文;
else
   信号代入文;
end if;
```

組み合わせ回路	関係する全ての入力・内部信号を列挙
順序回路	同期信号や制御信号を列挙

```
process(センシティビティリスト)
begin
  ：
  ：
end process;
```

●演算子

【※】信号代入の記号ではない

優先順位別					
	not	*	+	=	and
	abs	/	-	/=	or
	**(冪)	mod	&	<	xor
		rem		>	nand
				<=【※】	nor
				=>	

高 ←― 優先順位 ―→ 低

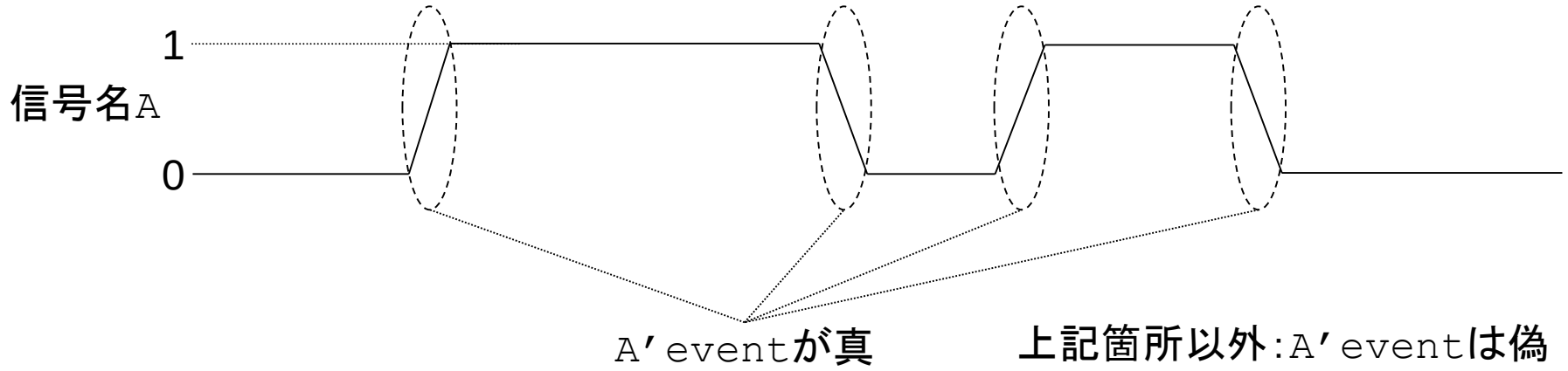
演算機能別	論理	関係	連接	算術
	not	=	&	+
	and	/=		-
	or	<		*
	xor	>		**(冪)
	nand	<=【※】		/
	nor	=>		mod
				rem
				abs

項数別	単項	2項
	not	and
	abs	or
	+	xor
	-	+
		など
		その他
	書式例	
	not A	B and C

event属性

●event属性：信号の変化を明示的に捉えるための属性

書式 | 信号名'event |



《利用例》

◆信号Aのポジティブエッジを捉えたい場合
「A'eventが真でかつA='1'が真」であるかどうかを判定すればよい。

◆信号Aのネガティブエッジを捉えたい場合
「A'eventが真でかつA='0'が真」であるかどうかを判定すればよい。

（例）レジスタの機能記述②　多ビット記憶

●レジスタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- 記憶ビット数
- 付加機能
- 追加出力

記憶ビット数：M

- ポジティブエッジトリガ
- M-bit記憶
- 付加機能なし
- 追加出力なし

のレジスタ

ジェネリック宣言の記述

```
generic(M:integer:=4);
```

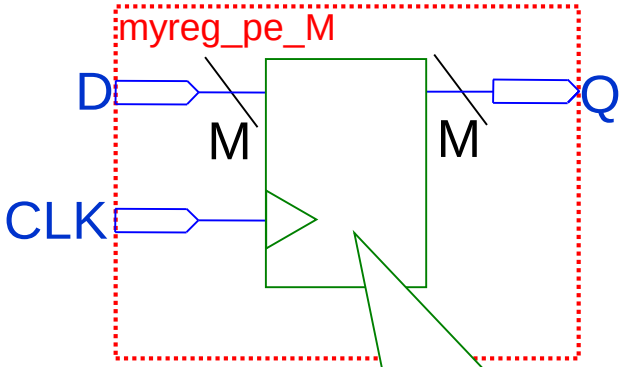
入出力ポートの記述

```
port(
   CLK : in std_logic;
   D : in std_logic_vector(M-1 downto 0);
   Q : out std_logic_vector(M-1 downto 0)
);
```

回路機能の記述

```
process(CLK)
begin
   if(CLK'event and CLK='1')then
      Q <= D;
   end if;
end process;
```

myreg_pe_M.vhd



入力		出力
CLK	D	Q
↑	－	D
↑以外	－	変化なし

【注】エンティティ名の
「**_M**」はM-bit記憶を意図している

（例）レジスタの機能記述③　1ビット→多ビット

1ビットレジスタ

入出力ポートの記述

```
port(
   CLK : in std_logic;
   D : in std_logic;
   Q : out std_logic
);
```

多ビットレジスタ

ジェネリック宣言の記述

```
generic(M:integer:=4);
```

入出力ポートの記述

```
port(
   CLK : in std_logic;
   D : in std_logic_vector(M-1 downto 0);
   Q : out std_logic_vector(M-1 downto 0)
);
```

回路機能の記述

```
process(CLK)
begin
   if(CLK'event and CLK='1')then
      Q <= D;
   end if;
end process;
```

- ✓ 変更の容易性を鑑みて、記憶ビット数をジェネリック宣言で用意し、データの入出力ポートをstd_logic_vector型に変更。
- ✓ 回路機能の記述は変更なし。

（例）レジスタの機能記述④　付加機能一覧

●レジスタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- 記憶ビット数
- 付加機能
- 追加出力

12種類の付加機能

- ポジティブエッジトリガ
- M-bit記憶
- 付加機能?????
- 追加出力なし

のレジスタ

複数の機能を同時に付加することもできる

付加機能	機能の意味	機能のための入力ポート名の例	機能を有効にする入力値	機能の作用とクロック信号との関係	
Clear (Reset) 機能	出力※を常に0とする。	ACLRN	0	クロック非同期	〔例1〕
		SCLRN		クロック同期	
		ACLR	1	クロック非同期	
		SCLR		クロック同期	
Preset (Set) 機能	出力※を常に1とする。	APRN	0	クロック非同期	
		SPRN		クロック同期	
		APR	1	クロック非同期	
		SPR		クロック同期	〔例2〕
Clock Enable 機能	クロック入力を許可する。	CEN	0		〔例3〕
		CE	1		
Output Control 機能	出力※をHigh Impedanceとする。≫	OCN	0		
		OC	1		〔例4〕

※
多ビット記憶の場合、出力の全ビット
集合体で指定

〔例1〕～〔例4〕は以降で具体的に例示する。
それ以外は、例を参考に記述してみよ。

集合体

●集合体:ビット(std_logic型の信号)要素を結合する

書式 (要素,要素,・・・,要素)

- 信号名または定数
  - 記述順が位置に対応する
  - または
- 位置=>信号名または定数
  - ビット位置添字
  - ビット範囲
    - downto/to
    - others

【注】・othersを用いた要素の後に、更なる要素は書けない。
・othersを用いた場合、被演算子として記述できない。

《利用例》Aがstd_logic、 Tがstd_logic_vector(3 downto 0)であるとき、

◆「Tの全ビットにAを代入する」

①	T <= A&A&A&A;	連接演算子を使用			
②	T <= (A,A,A,A);	集合体を使用	位置指定なし記述		
③	T <= (3=>A,2=>A,1=>A,0=>A);		位置指定つき記述	ビット毎に添字指定	
④	T <= (3 downto 0=>A);			ビット範囲指定	downto使用
⑤	T <= (others=>A);				others使用

◆「TのLSBに'1'、残りの全ビットにAを代入する」

⑥	T <= A&A&A&'1';	連接演算子を使用			
⑦	T <= (A,A,A,'1');	集合体を使用	位置指定なし記述		
⑧	T <= (3=>A,2=>A,1=>A,0=>'1');		位置指定つき記述	ビット毎に添字指定	
⑨	T <= (3 downto 1=>A, 0=>'1');			ビット範囲指定	downto使用
⑩	T <= (0=>'1', others=>A);				others使用

（例）レジスタの機能記述⑤　付加機能〔例１〕:ACLRN

付加機能一覧へ▲

●レジスタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- 記憶ビット数
- 付加機能
- 追加出力

clock Async. CLeaR Negative
ACLRN

- ポジティブエッジトリガ
- M-bit記憶
- 付加機能:ACLRN
- 追加出力なし

のレジスタ

ジェネリック宣言の記述

```
generic(M:integer:=4);
```

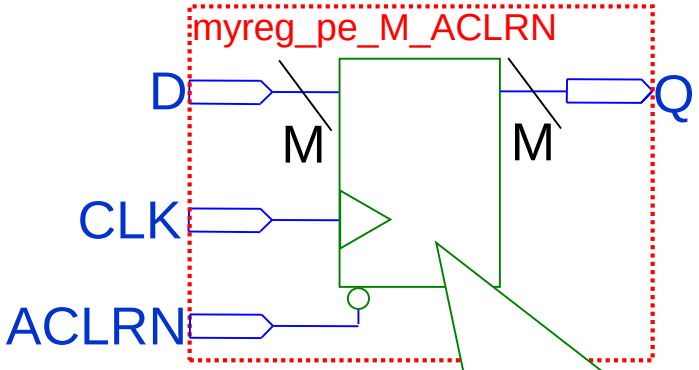
入出力ポートの記述

```
port(
  CLK, ACLRN : in std_logic;
  D : in std_logic_vector(M-1 downto 0);
  Q : out std_logic_vector(M-1 downto 0)
);
```

回路機能の記述

```
process(ACLRN,CLK)
begin
  if(ACLRN='0')then
    Q <= (others=>'0');
  elsif(CLK'event and CLK='1')then
    Q <= D;
  end if;
end process;
```

myreg_pe_M_ACLRN.vhd



入力			出力
ACLRN	CLK	D	Q
0	－	－	0クリア
1	↑	－	D
	↑以外	－	変化なし

【注】エンティティ名に付加機能の意を含めている

（例）レジスタの機能記述⑥　付加機能〔例2〕：SPR

付加機能一覧へ▲

●レジスタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- 記憶ビット数
- 付加機能
- 追加出力

clock Synchronous PReset
SPR

- ポジティブエッジトリガ
- M-bit記憶
- 付加機能：SPR
- 追加出力なし

のレジスタ

ジェネリック宣言の記述

```
generic(M:integer:=4);
```

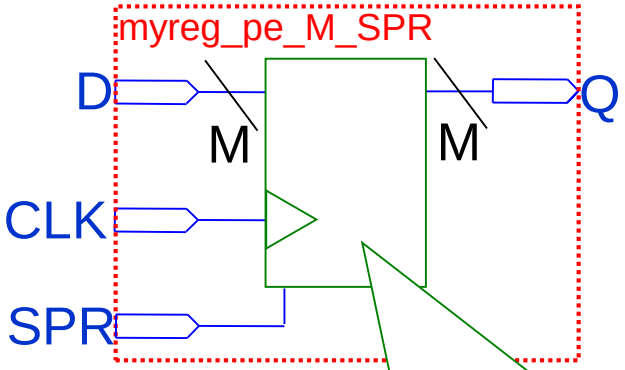
入出力ポートの記述

```
port(
  CLK, SPR : in std_logic;
  D : in std_logic_vector(M-1 downto 0);
  Q : out std_logic_vector(M-1 downto 0)
);
```

回路機能の記述

```
process(CLK,SPR)
begin
  if(CLK'event and CLK='1')then
    if(SPR='1')then
      Q <= (others=>'1');
    else
      Q <= D;
    end if;
  end if;
end process;
```

myreg_pe_M_SPR.vhd



入力			出力
CLK	SPR	D	Q
↑	1	－	プリセット
	0	－	D
↑以外	－	－	変化なし

【注】エンティティ名に付加機能の意を含めている

（例）レジスタの機能記述⑦　付加機能〔例3〕:CEN

付加機能一覧へ▲

●レジスタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- 記憶ビット数
- 付加機能
- 追加出力

Clock Enable Negative
CEN

- ポジティブエッジトリガ
- M-bit記憶
- 付加機能:CEN
- 追加出力なし

のレジスタ

ジェネリック宣言の記述

```
generic(M:integer:=4);
```

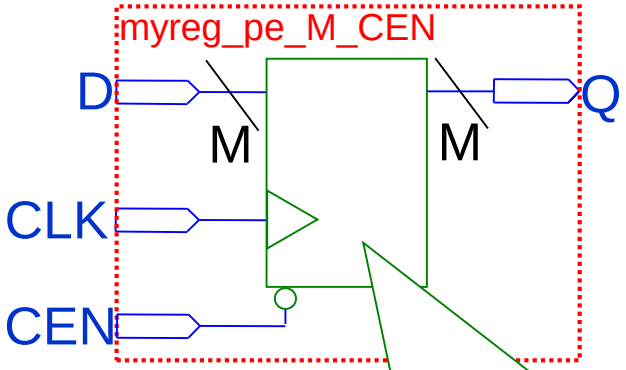
入出力ポートの記述

```
port(
  CLK, CEN : in std_logic;
  D : in std_logic_vector(M-1 downto 0);
  Q : out std_logic_vector(M-1 downto 0)
);
```

回路機能の記述

```
process(CLK,CEN)
begin
  if(CLK'event and CLK='1')then
    if(CEN='0')then
      Q <= D;
    end if;
  end if;
end process;
```

myreg_pe_M_CEN.vhd



入力			出力
CLK	CEN	D	Q
↑	0	－	D
	1	－	変化なし
↑以外	－	－	

【注】エンティティ名に付加機能の意を含めている

（例）レジスタの機能記述⑧　付加機能〔例4〕:OC

付加機能一覧へ▲

●レジスタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- 記憶ビット数
- 付加機能
- 追加出力

Output Control
OC

- ポジティブエッジトリガ
- M-bit記憶
- 付加機能:OC
- 追加出力なし

のレジスタ

ジェネリック宣言の記述

```
generic(M:integer:=4);
```

入出力ポートの記述

```
port(
  CLK, OC : in std_logic;
  D : in std_logic_vector(M-1 downto 0);
  Q : out std_logic_vector(M-1 downto 0)
);
```

各種宣言の記述

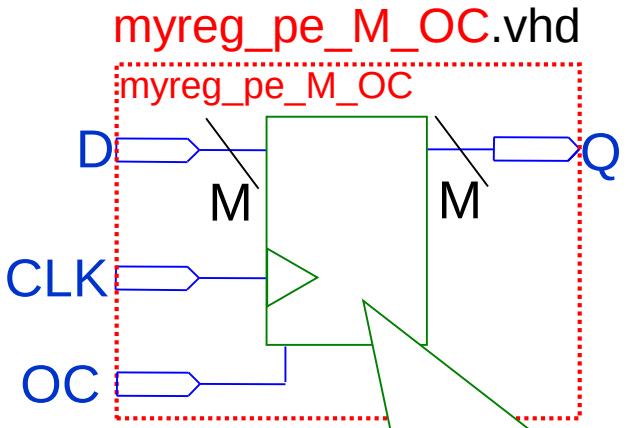
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

```
process(CLK)
begin
  if(CLK'event and CLK='1')then
    Y <= D;
  end if;
end process;
Q <= (others=>'Z') when(OC='1') else Y;
```

2プロセス

myreg_pe_M_OC.vhd



入力			出力
CLK	D	OC	Q
↑	–	0	D
↑以外	–		変化なし
–	–	1	High Impedance

【注】エンティティ名に付加機能の意を含めている

（例）レジスタの機能記述⑨　追加出力

●レジスタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- 記憶ビット数
- 付加機能
- 追加出力

Qの反転出力 →

- ポジティブエッジトリガ
- M-bit記憶
- 付加機能なし
- 追加出力：QN

のレジスタ

ジェネリック宣言の記述

```
generic(M:integer:=4);
```

入出力ポートの記述

```
port(
  CLK : in std_logic;
  D : in std_logic_vector(M-1 downto 0);
  Q, QN : out std_logic_vector(M-1 downto 0)
);
```

各種宣言の記述

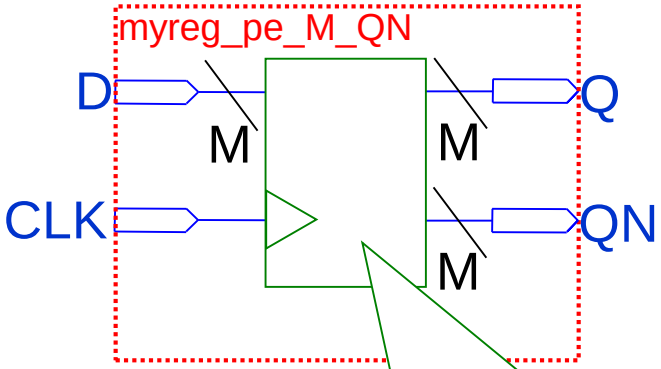
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

```
process(CLK)
begin
  if(CLK'event and CLK='1')then
    Y <= D;
  end if;
end process;
Q <= Y;
QN <= not Y;
```

3プロセス

myreg_pe_M_QN.vhd



入力		出力	
CLK	D	Q	QN
↑	－	D	not D
↑以外	－	変化なし	

【注】エンティティ名に付加機能の意を含めている

■必須課題■

以下の2つのレジスタのプロジェクトについての設計をせよ。
■プロジェクト「myreg_pe_M_CE」
■プロジェクト「myreg_pe_M_CE_OCN」

なお、
✓_peは、ポジティブエッジトリガの意味である。
✓_Mは、M-bit記憶の意味である。
✓Mはジェネリック宣言で指定できるようにすること。
✓_CE, _OCNは、
付加機能一覧の表の「機能のための入力ポート名の例」と同じとして考えよ。≫

▲追加課題▲

以下のレジスタのプロジェクトについての設計をせよ。
▲プロジェクト「myreg_pe_M_ACLR_CE_SCLR」

なお、
✓_peは、ポジティブエッジトリガの意味である。
✓_Mは、M-bit記憶の意味である。
✓Mはジェネリック宣言で指定できるようにすること。
✓_CE, _ACLR, _SCLRは、
付加機能一覧の表の「機能のための入力ポート名の例」と同じとして考えよ。≫

これまでの例の整理

Top-Level Entity名	スライド	論理回路種別	Process文	代入文	演算子	内部信号	generic宣言		モジュール化デザイン
mynand2	VHDL-01	組み合わせ	ー	信号代入	論理	ー	ー		ー
myor3	VHDL-02	組み合わせ	ー	信号代入	論理	○	ー		ー
mynand2xM	VHDL-02	組み合わせ	ー	信号代入	論理	ー	○	値渡しなし	ー
mynand2_cmp_inext	VHDL-02	組み合わせ	ー	信号代入 （下位Level ）	論理	ー	ー		○（その1）
mynand2xM_cmp_or3	VHDL-02	組み合わせ	ー	信号代入 （Top/下位Level ）	論理	○	○	値渡しあり ・Top-Level (signal宣言文) ・下位Level (generic map)	○（その2）
mynand2_cmp_fg	VHDL-02	組み合わせ	ー	信号代入 （下位Level ）	論理	ー	○	値渡しあり ・Top-Level (for-generate文)	○（その3）
myadderM	VHDL-03	組み合わせ	ー	信号代入	算術(+)	ー	ー		ー
myshift1	VHDL-03	組み合わせ	ー	信号代入	連接(&)	ー	ー		ー
mydec1to2	VHDL-03	組み合わせ	ー	条件付信号代入	関係	ー	ー		ー
myenc4to2	VHDL-04	組み合わせ	○(if)	信号代入	関係	ー	ー		ー
mydec1to2	VHDL-04	組み合わせ	○(case)	信号代入	なし	ー	ー		ー
myandM	VHDL-04	組み合わせ	○(for-loop)	変数代入	論理	ー	○	値渡しあり (for-loop文)	ー
myreg_***	VHDL-05	順序(記憶要素)	○(if)	信号代入 条件付信号代入	論理	○	○	値渡しあり (signal宣言文、集合体)	ー

さまざまなバリエーション
で学習した

以降、適宜応用する

さまざまなバリエーション
で学習済

適宜応用する