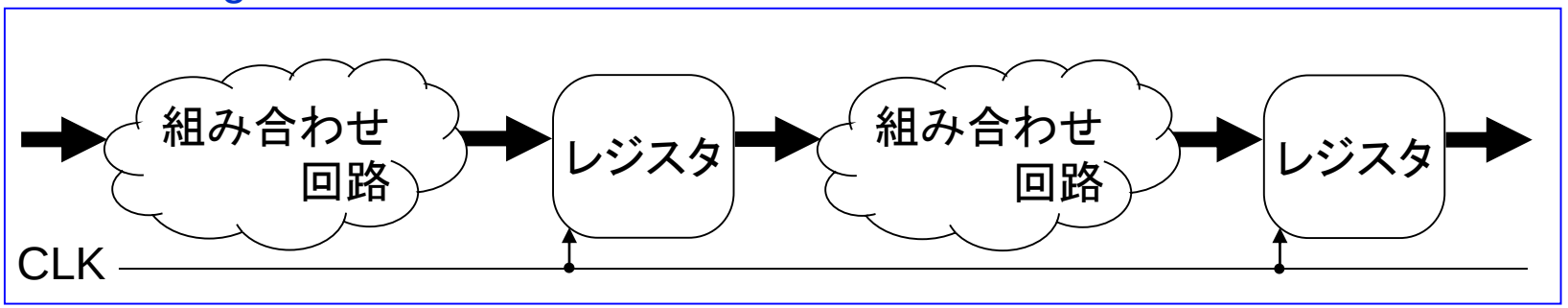
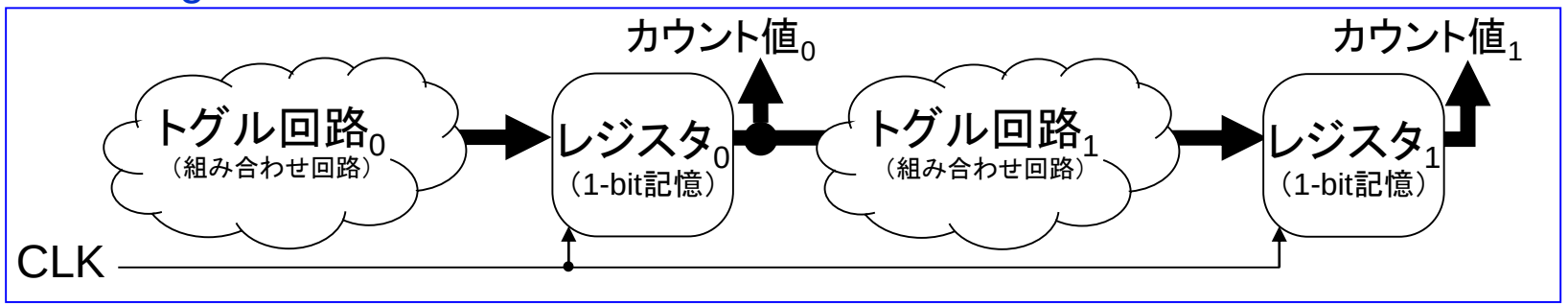
カウンタの設計　～RTLでの機能図～

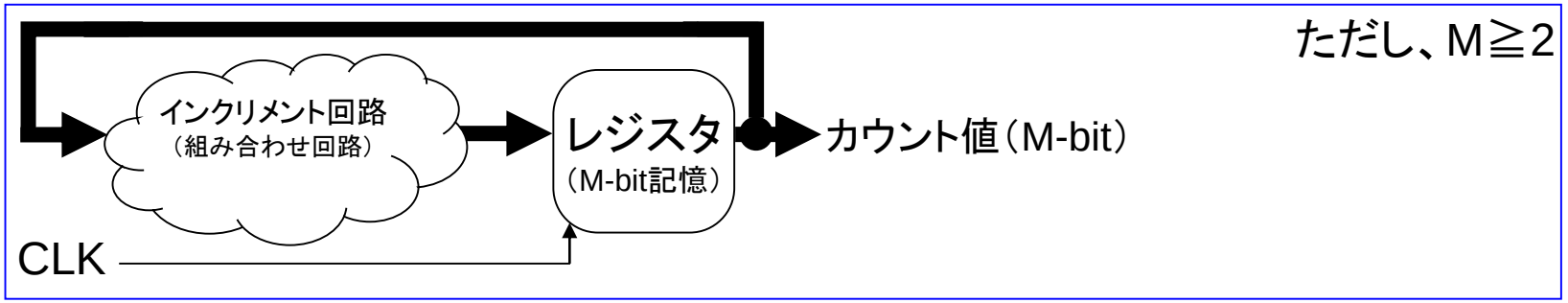
RTL（Register Transfer Level:レジスタ転送レベル）での一般的回路機能図



RTL（Register Transfer Level:レジスタ転送レベル）でのカウンタの機能図その1



RTL（Register Transfer Level:レジスタ転送レベル）でのカウンタの機能図その2



（例）カウンタの機能記述①　基本カウンタ

●カウンタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- カウントビット数
- 付加機能

基本カウンタ

- ポジティブエッジトリガ
- 1-bitカウント
- 付加機能なし

のカウンタ

入出力ポートの記述

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

各種宣言の記述

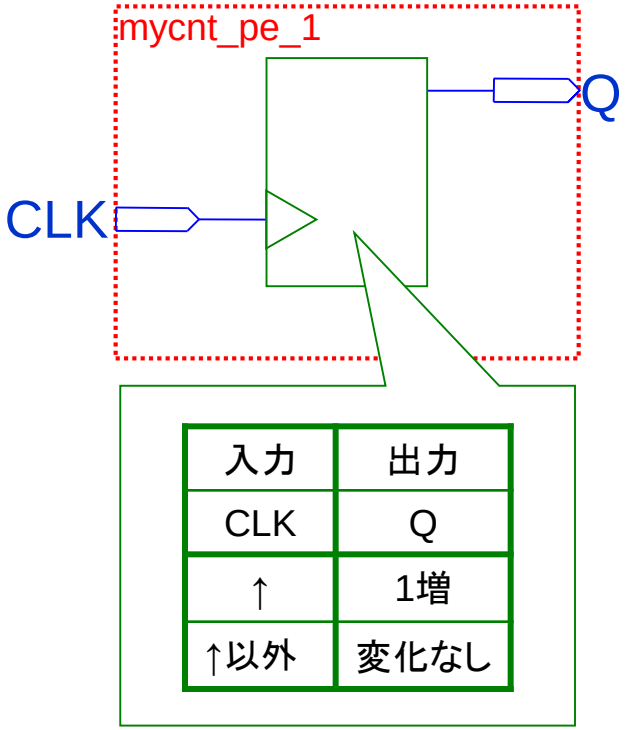
```
signal Y : std_logic;
```

回路機能の記述

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

2プロセス

mycnt_pe_1.vhd



入力	出力
CLK	Q
↑	1増
↑以外	変化なし

【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カウントを意図している

（例）カウンタの機能記述②　多ビットカウンタ

●カウンタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- カウントビット数
- 付加機能

カウントビット数：M

- ポジティブエッジトリガ
- M-bitカウント
- 付加機能なし

のカウンタ

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;
  Q : out std_logic_vector(M-1 downto 0)
);
```

各種宣言の記述

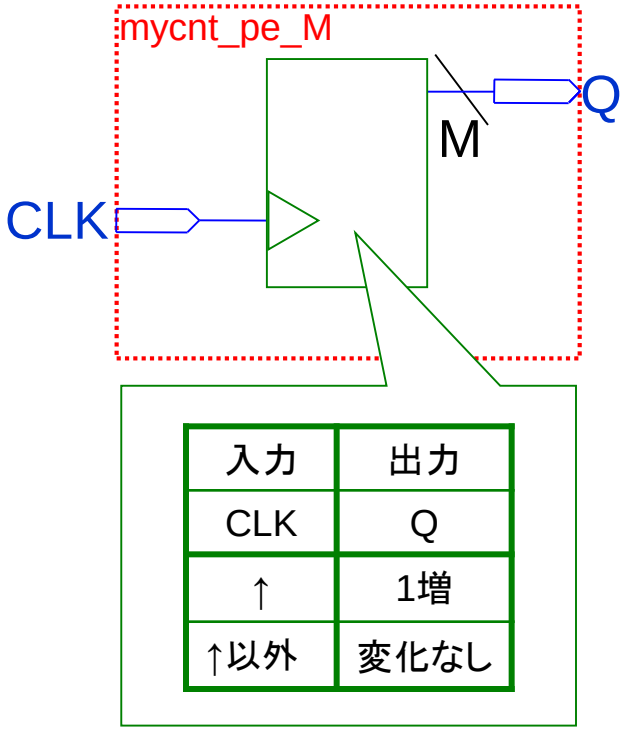
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

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

2プロセス

mycnt_pe_M.vhd



入力	出力
CLK	Q
↑	1増
↑以外	変化なし

【注】エンティティ名の
「_M」はM-bitカウントを意図している

（例）カウンタの機能記述③　1ビット→多ビット

1ビットカウンタ

入出力ポートの記述

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

回路機能の記述

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

多ビットカウンタ

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;
  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
    Y <= Y + '1';
  end if;
end process;
-------------------------------
Q <= Y;
```

- ✓ 変更の容易性を鑑みて、カウントビット数をジェネリック宣言で用意し、カウント値出力ポートをstd_logic_vector型に変更。
- ✓ 回路機能は、トグル動作では対応できないので、インクリメントに変更。

（例）カウンタの機能記述④　付加機能一覧

●カウンタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- カウントビット数
- 付加機能

14種類の付加機能

- ポジティブエッジトリガ
- M-bitカウント
- 付加機能?????

のカウンタ

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にする。	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	
Load (Set) 機能	出力を任意の値にする。	ALDN	0	クロック非同期	
		SLDN		クロック同期	
		ALD	1	クロック非同期	
		SLD		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にする。≫	OCN	0		
		OC	1		〔例4〕
Up/**D**own切替機能	カウント増／減を切り替える。	**UND**	0⇒**U**,1⇒**D**	クロック同期	
		UDN	1⇒**U**,0⇒**D**		〔例5〕

※
多ビットカウントの場合、出力の全ビット
集合体で指定

〔例1〕～〔例5〕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。

（例）カウンタの機能記述⑤　付加機能〔例１〕：ACLRN

付加機能一覧へ▲

●カウンタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- カウントビット数
- 付加機能

clock Async. CLeaR Negative
ACLRN

- ポジティブエッジトリガ
- M-bitカウント
- 付加機能：ACLRN

のカウンタ

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;
  Q : out std_logic_vector(M-1 downto 0)
);
```

各種宣言の記述

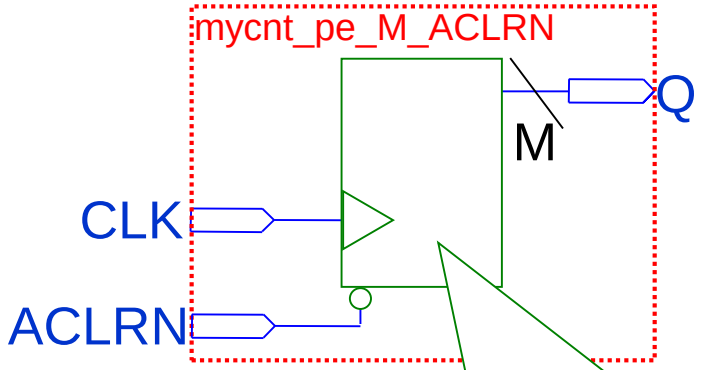
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

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

2プロセス

mycnt_pe_M_ACLRN.vhd



入力		出力
ACLRN	CLK	Q
0	－	0クリア
1	↑	1増
	↑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）カウンタの機能記述⑥　付加機能〔例2〕：SLD

付加機能一覧へ▲

●カウンタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- カウントビット数
- 付加機能

clock Synchronous LoaD
SLD

- ポジティブエッジトリガ
- M-bitカウント
- 付加機能：SLD

のカウンタ

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, SLD : 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

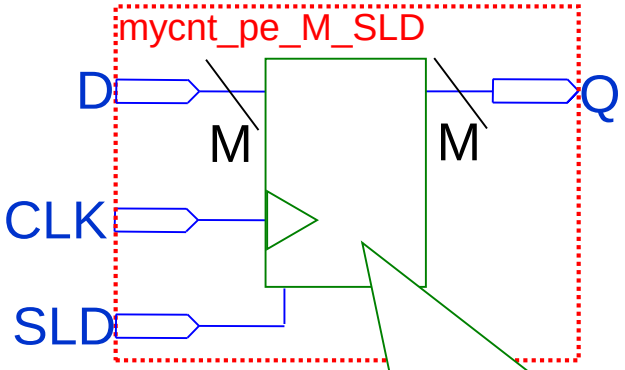
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

```
process(CLK,SLD)
begin
  if(CLK'event and CLK='1')then
    if(SLD='1')then
      Y <= D;
    else
      Y <= Y + '1';
    end if;
  end if;
end process;
Q <= Y;
```

2プロセス

mycnt_pe_M_SLD.vhd



入力			出力
CLK	SLD	D	Q
↑	1	－	D
	0	－	1増
↑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）カウンタの機能記述⑦　付加機能〔例3〕:CEN

付加機能一覧へ▲

●カウンタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- カウントビット数
- 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カウント
- 付加機能:CEN
- 追加出力なし

のカウンタ

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;
  Q : out std_logic_vector(M-1 downto 0)
);
```

各種宣言の記述

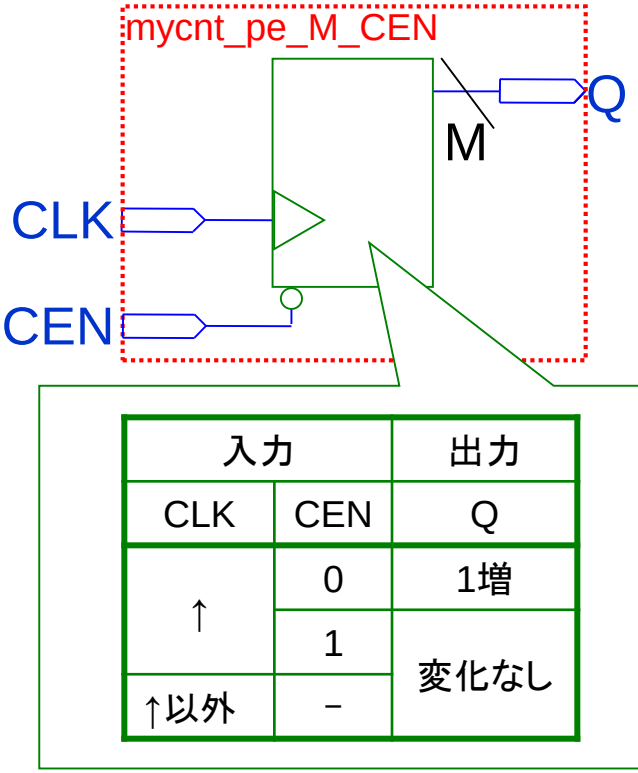
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

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

2プロセス

mycnt_pe_M_CEN.vhd



入力		出力
CLK	CEN	Q
↑	0	1増
	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）カウンタの機能記述⑧　付加機能〔例4〕:OC

付加機能一覧へ▲

●カウンタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- カウントビット数
- 付加機能

Output Control
OC

- ポジティブエッジトリガ
- M-bitカウント
- 付加機能:OC

のカウンタ

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;
  Q : out std_logic_vector(M-1 downto 0)
);
```

各種宣言の記述

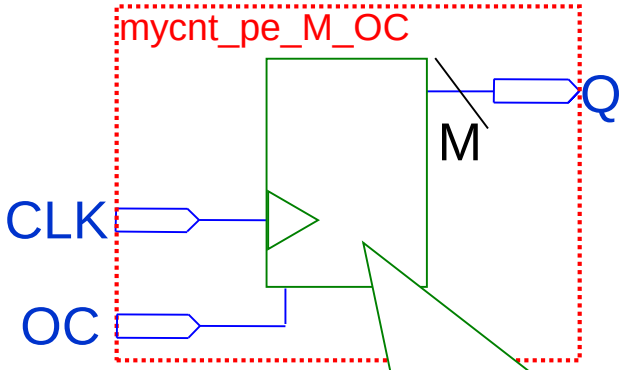
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

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

2プロセス

mycnt_pe_M_OC.vhd



入力		出力
CLK	OC	Q
↑	0	1増
↑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）カウンタの機能記述⑨　付加機能〔例5〕:UDN

付加機能一覧へ▲

●カウンタの機能記述
- ポジティブ／ネガティブ　エッジトリガ
- カウントビット数
- 付加機能

Up count/ Down count Neg.
UDN

- ポジティブエッジトリガ
- M-bitカウント
- 付加機能:UDN

のカウンタ

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, UDN : in std_logic;
  Q : out std_logic_vector(M-1 downto 0)
);
```

各種宣言の記述

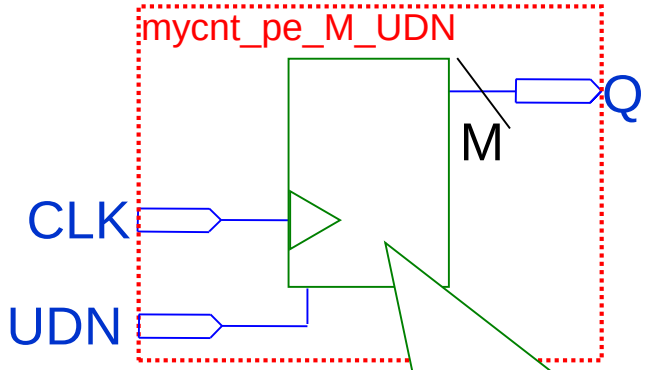
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

```
process(CLK,UDN)
begin
  if(CLK'event and CLK='1')then
    if(UDN='1')then
      Y <= Y + '1';
    else
      Y <= Y + (M-1 downto 0=>'1');
    end if;
  end if;
end process;
Q <= Y;
```

2プロセス

mycnt_pe_M_UDN.vhd



入力		出力
CLK	UDN	Q
↑	1	1増
	0	1減
↑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■

以下のカウンタのプロジェクトについての設計をせよ。
■ mycnt_pe_M_ACLR_CE_SLD

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カウントの意味である。
✓Mはジェネリック宣言で指定できるようにすること。
✓ _ACLR, _CE, _SLD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▲

以下のカウンタのプロジェクトについての設計をせよ。
▲ mycnt_pe_M_ACLR_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カウントの意味である。
✓Mはジェネリック宣言で指定できるようにすること。
✓ _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
mycnt_***	VHDL-06	順序	○(if)	信号代入 条件付信号代入	算術(+)	○	○	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済

適宜応用する