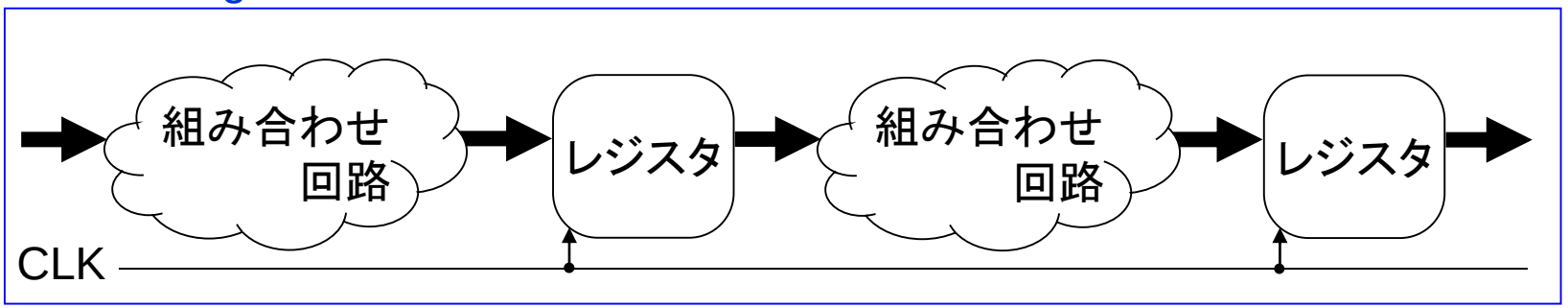
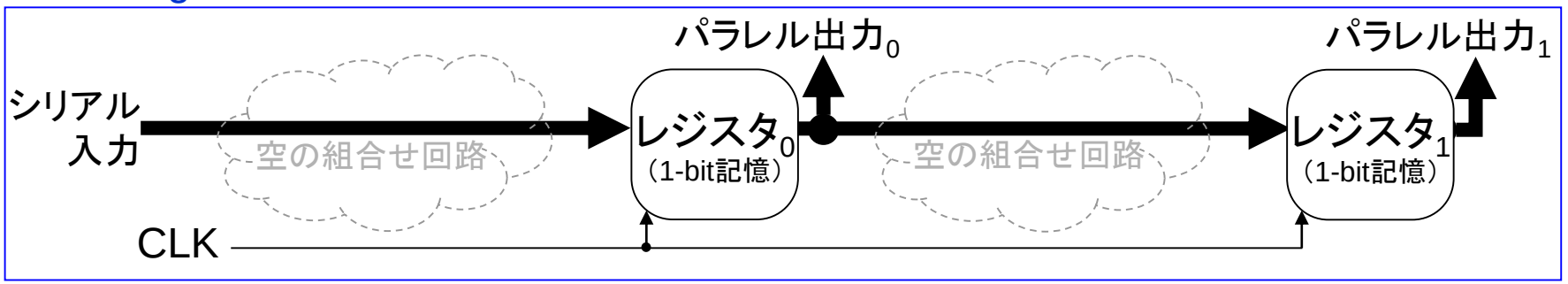
シフトレジスタの設計　～RTLでの機能図～

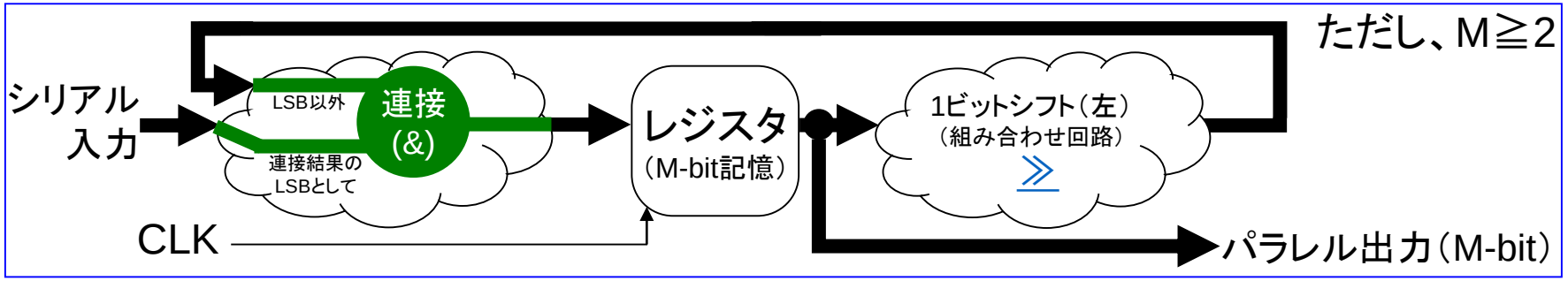
RTL（Register Transfer Level:レジスタ転送レベル）での一般的回路機能図



RTL（Register Transfer Level:レジスタ転送レベル）でのシフトレジスタの機能図その1

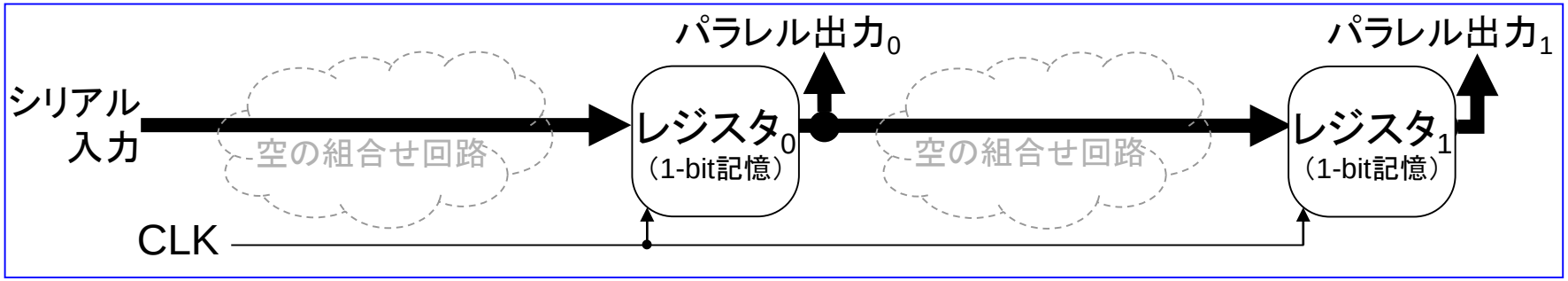


RTL（Register Transfer Level:レジスタ転送レベル）でのシフトレジスタの機能図その2

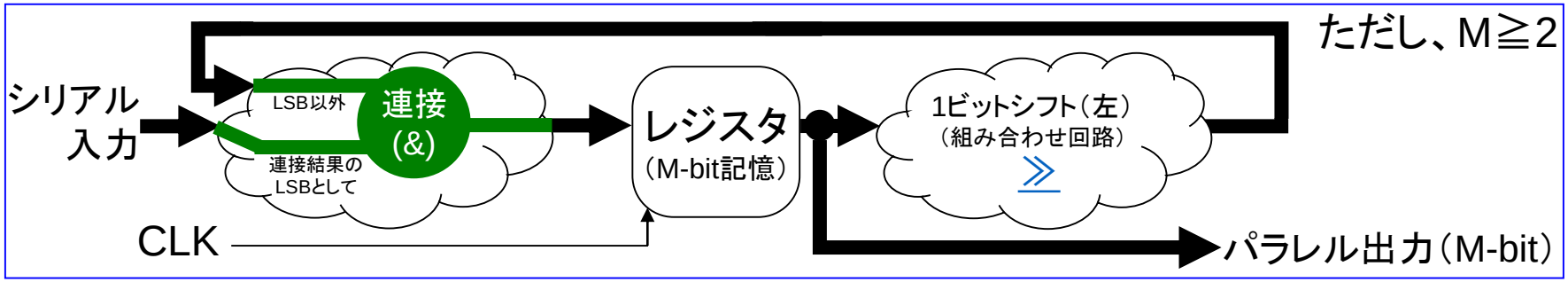


シフトレジスタの設計　～シフト動作の向き～

RTL（Register Transfer Level:レジスタ転送レベル）でのシフトレジスタの機能図その1



RTL（Register Transfer Level:レジスタ転送レベル）でのシフトレジスタの機能図その2



一般に、シフトレジスタにおいては、シフト動作の向きを「左／右」で表す。
この「左／右」の表現を、ここでは、downto指定での添え字順序を基準として、以下のように定める。
✓添え字の大きい方への向きを左
✓添え字の小さい方への向きを右

これより、
➢機能図その1は、図をそのまま見ると、右シフトに見えるが、
　パラレル出力の添え字をdownto指定での順序に照らし合わせると、左シフトのシフトレジスタである。
➢機能図その2は、1ビットシフト回路≫ 及び、LSBの位置をdownto指定での順序に照らし合わせると、
　左シフトのシフトレジスタである。

（例）シフトレジスタの機能記述①　基本シフトレジスタ

●シフト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
- シフト方向
- 付加機能

基本シフトレジスタ

- ポジティブエッジトリガ
- 1-bit記憶
- シフト方向無意味
- 付加機能なし

のシフトレジスタ

【注】
基本レジスタ≫ と
同じであることに注意せよ。

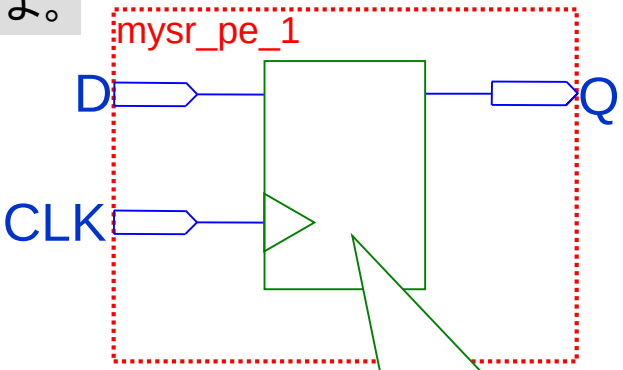
入出力ポートの記述

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

回路機能の記述

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

mysr_pe_1.vhd



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

（例）シフトレジスタの機能記述②　多ビット記憶・左シフト

●シフト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
- ・シフト方向
- ・付加機能

記憶ビット数：M
シフト方向：左

- ・ポジティブエッジトリガ
- ・M-bit記憶
- ・左シフト
- ・付加機能なし

のシフトレジスタ

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

各種宣言の記述

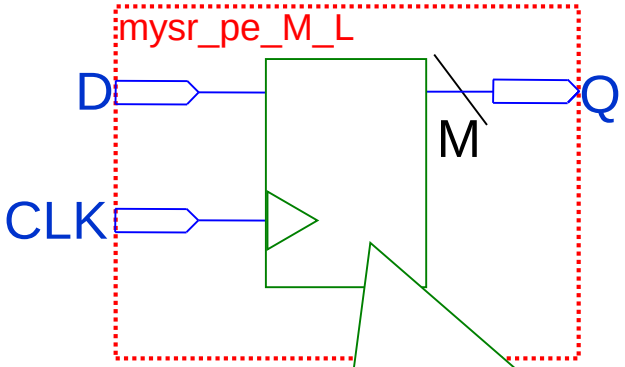
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

```
process(CLK)
begin
  if(CLK'event and CLK='1')then
    Y <= Y(M-2 downto 0)&D;
  end if;
end process;
----------------------------------
Q <= Y;
```

2プロセス

mysr_pe_M_L.vhd



入力		出力					動作
CLK	D	Q					
		$Q_{(M-1)}$	$Q_{(M-2)}$	~	$Q_{(1)}$	$Q_{(0)}$	
↑	0	$Q^p_{(M-2)}$	$Q^p_{(M-3)}$	~	$Q^p_{(0)}$	0	左シフト
	1	$Q^p_{(M-2)}$	$Q^p_{(M-3)}$	~	$Q^p_{(0)}$	1	
↑以外	0	$Q^p_{(M-1)}$	$Q^p_{(M-2)}$	~	$Q^p_{(1)}$	$Q^p_{(0)}$	変化なし（Q^p）
	1						

【注】エンティティ名の「_M」はM-bit記憶を、「_L」はシフト方向を、意図している

（例）シフトレジスタの機能記述③　多ビット記憶・右シフト

●シフト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
- シフト方向
- 付加機能

記憶ビット数：M
シフト方向：右

- ポジティブエッジトリガ
- M-bit記憶
- 右シフト
- 付加機能なし

のシフトレジスタ

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

各種宣言の記述

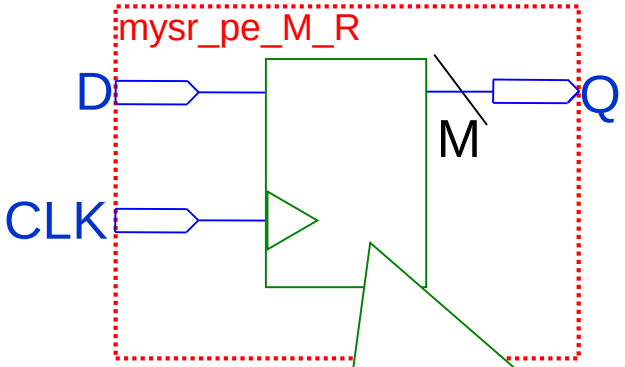
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

```
process(CLK)
begin
  if(CLK'event and CLK='1')then
    Y <= D&Y(M-1 downto 1);
  end if;
end process;
-------------------------------
Q <= Y;
```

2プロセス

mysr_pe_M_R.vhd



入力		出力					動作
CLK	D	Q					
		$Q_{(M-1)}$	$Q_{(M-2)}$	~	$Q_{(1)}$	$Q_{(0)}$	
↑	0	0	$Q^p_{(M-1)}$	~	$Q^p_{(2)}$	$Q^p_{(1)}$	右シフト
	1	1	$Q^p_{(M-1)}$	~	$Q^p_{(2)}$	$Q^p_{(1)}$	
↑以外	0	$Q^p_{(M-1)}$	$Q^p_{(M-2)}$	~	$Q^p_{(1)}$	$Q^p_{(0)}$	変化なし（Q^p）
	1						

【注】エンティティ名の「_M」はM-bit記憶を、「_R」はシフト方向を、意図している

（例）シフトレジスタの機能記述④　1ビット→多ビット・左（／右）シフト

1ビットシフトレジスタ

入出力ポートの記述

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

回路機能の記述

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

【注】基本レジスタ≫と同

多ビット・左（／右）シフトレジスタ

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;
  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(M-2 downto 0)&D; --Left
    --(Y <= D&Y(M-1 downto 1); Right)
  end if;
end process;
-------------------------------------
Q <= Y;
```

- ✓ 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、シフトレジスタ出力ポートをstd_logic_vector型に変更。
- ✓ 回路機能は、連接演算子を用いた信号代入に変更。

（例）シフトレジスタの機能記述⑤　付加機能一覧

●シフト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
- シフト方向
- 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記憶
- 左（／右）シフト
- 付加機能?????

のシフトレジスタ

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	
Parallel Load 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〕
Left/**R**ight切替機能	シフト方向を切り替える。	**L**N**R**	0⇒**L**,1⇒**R**	クロック同期	
		LRN	1⇒**L**,0⇒**R**		〔例5〕

※
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）シフトレジスタの機能記述⑥　付加機能〔例1〕:ACLRN

付加機能一覧へ▲

●シフト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
- シフト方向
- 付加機能

clock Async. CLeaR Negative
ACLRN

- ポジティブエッジトリガ
- M-bit記憶
- 左シフト
- 付加機能:ACLRN

のシフトレジスタ

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

各種宣言の記述

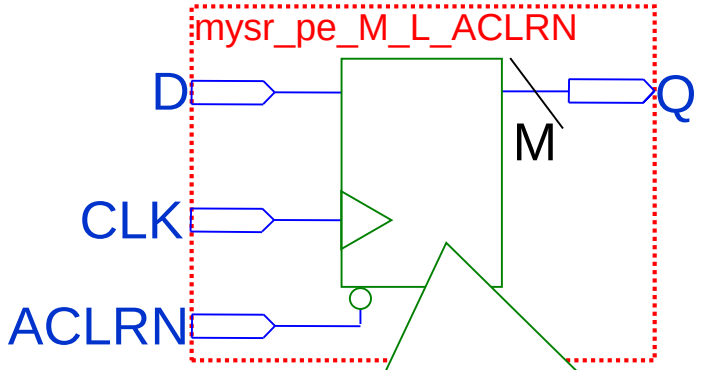
```
signal Y : std_logic_vector(M-1 downto 0);
```

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(M-2 downto 0)&D;
  end if;
end process;
-------------------------------
Q <= Y;
```

2プロセス

mysr_pe_M_L_ACLRN.vhd



入力			出力					動作
			Q					
ACLRN	CLK	D	$Q_{(M-1)}$	$Q_{(M-2)}$	~	$Q_{(1)}$	$Q_{(0)}$	
0	–	–	0	0	~	0	0	0クリア
1	↑	0	$Q^{p}_{(M-2)}$	$Q^{p}_{(M-3)}$	~	$Q^{p}_{(0)}$	0	左シフト
		1	$Q^{p}_{(M-2)}$	$Q^{p}_{(M-3)}$	~	$Q^{p}_{(0)}$	1	
	↑以外	0	$Q^{p}_{(M-1)}$	$Q^{p}_{(M-2)}$	~	$Q^{p}_{(1)}$	$Q^{p}_{(0)}$	変化なし（Q^{p}）
		1						

【注】エンティティ名に付加機能の意を含めている

（例）シフトレジスタの機能記述⑦　付加機能〔例2〕：SLD

付加機能一覧へ▲

●シフト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
- シフト方向
- 付加機能

clock Synchronous LoaD
SLD

- ポジティブエッジトリガ
- M-bit記憶
- 左シフト
- 付加機能：SLD

のシフトレジスタ

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

各種宣言の記述

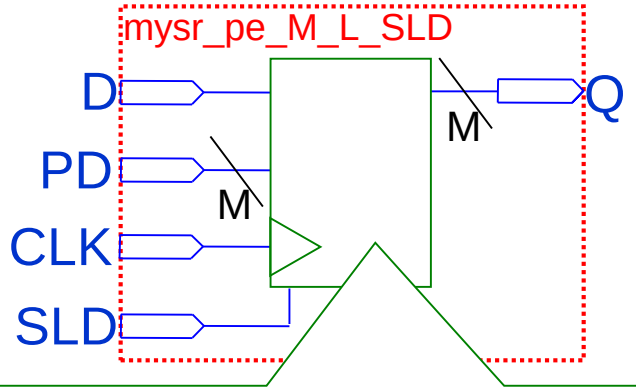
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

```
process(CLK,SLD)
begin
  if(CLK'event and CLK='1')then
    if(SLD='1')then
      Y <= PD;
    else
      Y <= Y(M-2 downto 0)&D;
    end if;
  end if;
end process;
Q <= Y;
```

2プロセス

mysr_pe_M_L_SLD.vhd



入力				出力					動作
CLK	SLD	D	PD	Q					
				$Q_{(M-1)}$	$Q_{(M-2)}$	~	$Q_{(1)}$	$Q_{(0)}$	
↑	1	–	PD	$PD_{(M-1)}$	$PD_{(M-2)}$	~	$PD_{(1)}$	$PD_{(0)}$	ロード
	0	0	–	$Q^p_{(M-2)}$	$Q^p_{(M-3)}$	~	$Q^p_{(0)}$	0	左シフト
		1	–	$Q^p_{(M-2)}$	$Q^p_{(M-3)}$	~	$Q^p_{(0)}$	1	
↑以外	–	0	–	$Q^p_{(M-1)}$	$Q^p_{(M-2)}$	~	$Q^p_{(1)}$	$Q^p_{(0)}$	変化なし（Q^p）
		1	–						

【注】エンティティ名に付加機能の意を含めている

（例）シフトレジスタの機能記述⑧　付加機能〔例3〕：CEN

付加機能一覧へ▲

●シフト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
- シフト方向
- 付加機能

Clock Enable Negative
CEN

- ポジティブエッジトリガ
- M-bit記憶
- 左シフト
- 付加機能：CEN

のシフトレジスタ

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

各種宣言の記述

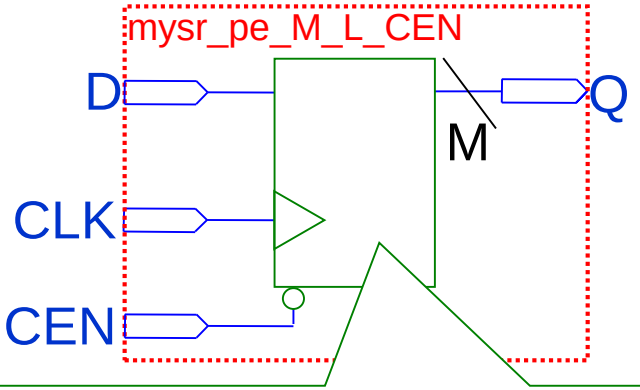
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

```
process(CLK,CEN)
begin
  if(CLK'event and CLK='1')then
    if(CEN='0')then
      Y <= Y(M-2 downto 0)&D;
    end if;
  end if;
end process;
-----------------------------
Q <= Y;
```

2プロセス

mysr_pe_M_L_CEN.vhd



入力			出力					動作
			Q					
CLK	CEN	D	$Q_{(M-1)}$	$Q_{(M-2)}$	~	$Q_{(1)}$	$Q_{(0)}$	
↑	0	0	$Q^{p}_{(M-2)}$	$Q^{p}_{(M-3)}$	~	$Q^{p}_{(0)}$	0	左シフト
		1	$Q^{p}_{(M-2)}$	$Q^{p}_{(M-3)}$	~	$Q^{p}_{(0)}$	1	
	1	0	$Q^{p}_{(M-1)}$	$Q^{p}_{(M-2)}$	~	$Q^{p}_{(1)}$	$Q^{p}_{(0)}$	変化なし（Q^{p}）
		1						
↑以外	–	0						
		1						

【注】エンティティ名に付加機能の意を含めている

（例）シフトレジスタの機能記述⑨　付加機能〔例4〕:OC

付加機能一覧へ▲

●シフト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
- シフト方向
- 付加機能

Output Control
OC

- ポジティブエッジトリガ
- M-bit記憶
- 左シフト
- 付加機能:OC

のシフトレジスタ

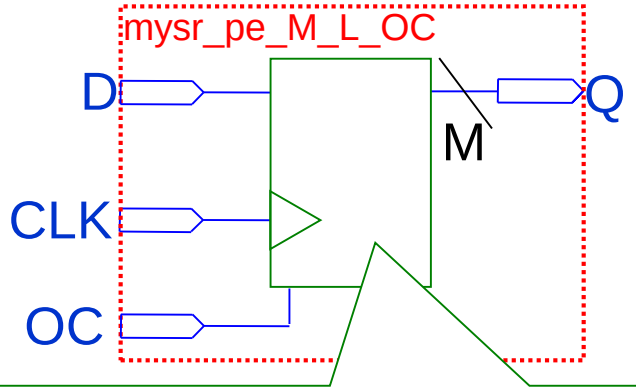
ジェネリック宣言の記述

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

mysr_pe_M_L_OC.vhd

入出力ポートの記述

```
port(
  CLK, OC : in std_logic;
  D : 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(M-2 downto 0)&D;
  end if;
end process;
-------------------------------
Q <= (others=>'Z') when(OC='1') else Y;
```

2プロセス

入力			出力					動作
			Q					
CLK	D	OC	$Q_{(M-1)}$	$Q_{(M-2)}$	~	$Q_{(1)}$	$Q_{(0)}$	
↑	0	0	$Q^p_{(M-2)}$	$Q^p_{(M-3)}$	~	$Q^p_{(0)}$	0	左シフト
	1		$Q^p_{(M-2)}$	$Q^p_{(M-3)}$	~	$Q^p_{(0)}$	1	
↑以外	0		$Q^p_{(M-1)}$	$Q^p_{(M-2)}$	~	$Q^p_{(1)}$	$Q^p_{(0)}$	変化なし（Q^p）
	1							
–	–	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）シフトレジスタの機能記述⑩　付加機能〔例5〕:LRN

付加機能一覧へ▲

●シフト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
- シフト方向
- 付加機能

Left shift/ Right shift　Neg.
LRN

- ポジティブエッジトリガ
- M-bit記憶
- 左右両シフト
- 付加機能:LRN

のシフトレジスタ

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

各種宣言の記述

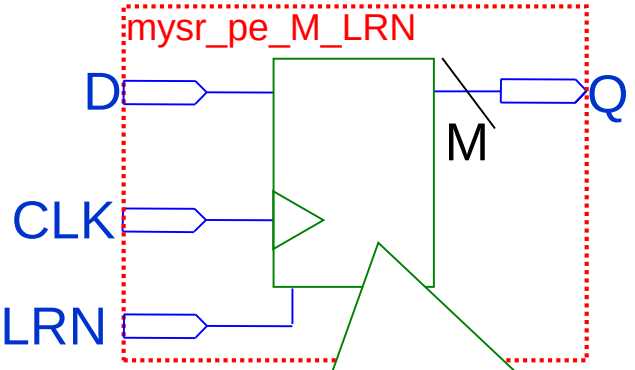
```
signal Y : std_logic_vector(M-1 downto 0);
```

回路機能の記述

```
process(CLK,LRN)
begin
  if(CLK'event and CLK='1')then
    if(LRN='1')then
      Y <= Y(M-2 downto 0)&D;
    else
      Y <= D&Y(M-1 downto 1);
    end if;
  end if;
end process;
-------------------------------
Q <= Y;
```

2プロセス

mysr_pe_M_LRN.vhd



入力			出力					動作
CLK	LRN	D	Q					
			$Q_{(M-1)}$	$Q_{(M-2)}$	~	$Q_{(1)}$	$Q_{(0)}$	
↑	1	0	$Q^p_{(M-2)}$	$Q^p_{(M-3)}$	~	$Q^p_{(0)}$	0	左シフト
		1	$Q^p_{(M-2)}$	$Q^p_{(M-3)}$	~	$Q^p_{(0)}$	1	
	0	0	0	$Q^p_{(M-1)}$		$Q^p_{(2)}$	$Q^p_{(1)}$	右シフト
		1	1	$Q^p_{(M-1)}$		$Q^p_{(2)}$	$Q^p_{(1)}$	
↑以外	–	0	$Q^p_{(M-1)}$	$Q^p_{(M-2)}$	~	$Q^p_{(1)}$	$Q^p_{(0)}$	変化なし（Q^p）
		1						

【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■

■シフトレジスタのプロジェクト「mysr_pe_M_LNR_CE」について回路を設計せよ。

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。
✓_LNR, _CE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
mysr_***	VHDL-07	順序	○(if)	信号代入 条件付信号代入	連接(&)	○	○	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