EE434
ASIC & Digital Systems

VHDL

Practice (Sequential Logic & Sequential Processing)

Spring 2016

Dae Hyun Kim
daehyun@eecs.wsu.edu

1. Clock Generator

- Entity name
  - myOSC (oscillator)
- Inputs
- Outputs
  - Clk
- Function
  - Generate a clock signal of period 2ns.

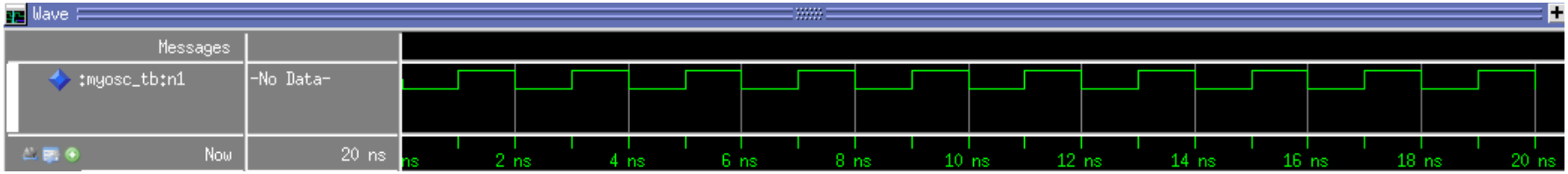
1. Clock Generator

- Entity name
  - myOSC
- Inputs
- Outputs
  - Clk
- Function
  - Generate a clock signal of period 2ns.

```
ENTITY myOSC IS
   PORT ( Clk : OUT std_logic );
END myOSC;

ARCHITECTURE myOSC_arch OF myOSC IS
BEGIN
   PROCESS
   BEGIN
      Clk <= '0';
      WAIT FOR 1 ns;
      Clk <= '1';
      WAIT FOR 1 ns;
   END PROCESS;
END myOSC_arch;
```

1. Clock Generator



2. Positive-Edge-Triggered D-FF

- Entity name
  - myDFF
- Inputs
  - D, Clk
- Outputs
  - Q, QN
- Function
  - $Q = D$ when Clk $\uparrow$
  - $QN = \overline{D}$

2. Positive-Edge-Triggered D-FF

- Entity name
  - myDFF
- Inputs
  - D, Clk
- Outputs
  - Q, QN
- Function
  - $Q = D$ when Clk ↑
  - $QN = \overline{D}$

```
ENTITY  myDFF  IS
   PORT  (  D, Clk : IN std_logic;
            Q, QN : OUT std_logic );
END  myDFF;

ARCHITECTURE  myDFF_arch  OF  myDFF  IS
BEGIN
   PROCESS  ( Clk )
   BEGIN
      IF  (Clk'EVENT )  AND  (Clk = '1')  THEN
         Q <=  D;
         QN  <=  NOT  D;
      END  IF;
   END  PROCESS;
END  myDFF_arch;
```

‘EVENT

- A signal has attributes.
  - ‘EVENT
    - returns TRUE if an event occurred (FALSE otherwise).
  - ‘LAST_VALUE
    - returns the last value before the last event.
  - ...

2. Positive-Edge-Triggered D-FF

- Entity name
  - myDFF
- Inputs
  - D, Clk
- Outputs
  - Q, QN
- Function
  - $Q = D$ when Clk ↑
  - $QN = \overline{D}$

```
ENTITY  myDFF  IS
   PORT  (  D, Clk : IN std_logic;
            Q, QN : OUT std_logic );
END  myDFF;

ARCHITECTURE  myDFF_arch  OF  myDFF  IS
BEGIN
  PROCESS  ( Clk )
   BEGIN
     IF  (Clk'EVENT )  AND  (Clk = '1')  AND  (Clk'LAST_VALUE = '0')  THEN
        Q <=  D;
        QN  <=  NOT  D;
     END  IF;
   END  PROCESS;
END  myDFF_arch;
```

2. Positive-Edge-Triggered D-FF

- Entity name
  - myDFF
- Inputs
  - D, Clk
- Outputs
  - Q, QN
- Function
  - $Q = D$ when Clk ↑
  - $QN = \overline{D}$

```
ENTITY  myDFF  IS
   PORT  (  D, Clk : IN std_logic;
            Q, QN : OUT std_logic );
END  myDFF;

ARCHITECTURE  myDFF_arch  OF  myDFF  IS
BEGIN
   PROCESS  ( Clk )
   BEGIN
      IF  rising_edge (Clk) THEN
         Q <=  D;
         QN  <=  NOT  D;
      END  IF;
   END  PROCESS;
END  myDFF_arch;
```

2. Positive-Edge-Triggered D-FF

```
ENTITY  myDFF  IS
   PORT  (  D, Clk : IN std_logic;
            Q, QN : OUT std_logic );
END  myDFF;

ARCHITECTURE  myDFF_arch  OF  myDFF  IS
BEGIN
  PROCESS
   BEGIN
     WAIT  UNTIL  (Clk'EVENT )  AND  (Clk = '1')  AND  (Clk'LAST_VALUE = '0')  THEN
        Q <=  D;
        QN  <=  NOT  D;
     END  IF;
   END  PROCESS;
END  myDFF_arch;
```

3. T-FF

- Entity name
  - myTFF
- Inputs
  - T, Clk
- Outputs
  - Q, QN
- Function
  - $Q^+ = \bar{Q}$ when T=1 and Clk ↑
  - $QN^+ = Q$

3. T-FF

- Entity name
  - myTFF
- Inputs
  - T, Clk
- Outputs
  - Q, QN
- Function
  - $Q^+ = \bar{Q}$ when T=1 and Clk ↑
  - $QN^+ = Q$

```
ENTITY  myTFF  IS
   PORT  (  T, Clk : IN std_logic;
            Q, QN : OUT std_logic );
END  myTFF;

ARCHITECTURE  myTFF_arch  OF  myTFF  IS
BEGIN
   PROCESS  ( Clk )
   BEGIN
      IF  rising_edge (Clk)  AND  T = '1' THEN
         Q <= NOT  Q;
         QN  <=  Q;
      END  IF;
   END  PROCESS;
END  myTFF_arch;
```

Output signals cannot be assigned in this way!

3. T-FF

```
ENTITY  myTFF  IS
   PORT  (  T, Clk : IN std_logic;
            Q, QN : OUT std_logic );
END  myTFF;

ARCHITECTURE  myTFF_arch  OF  myTFF  IS
BEGIN
   PROCESS  ( Clk )
   BEGIN
      IF  rising_edge (Clk)  AND  T='1'  THEN
         IF  Q = '1'  THEN
            Q <= '0';
            QN <= '1';
         ELSE
            Q <= '1';
            QN <= '0';
         END  IF;
      END  IF;
   END  PROCESS;
END  myTFF_arch;
```

Output signals cannot be used in IF.

3. T-FF

```
ARCHITECTURE myTFF_arch OF myTFF IS
   SIGNAL n1 : std_logic;
BEGIN
   PROCESS ( Clk )
   BEGIN
      IF rising_edge (Clk) AND T='1' THEN
         IF n1 = '1' THEN
            n1 <= '0';
            Q <= '0';
            QN <= '1';
         ELSE
            n1 <= '1';
            Q <= '1';
            QN <= '0';
         END IF;
      END IF;
   END PROCESS;
END myTFF_arch;
```

4. 16-bit XOR

- Entity name
  - myXOR16
- Inputs
  - A, B (std_logic_vector (15 downto 0))
- Outputs
  - Y (std_logic_vector (15 downto 0))
- Function
  - Y(n) = A(n) XOR B(n)

4. 16-bit XOR

- Entity name
  - myXOR16
- Inputs
  - A, B
- Outputs
  - Y
- Function
  - Y(n) = A(n) XOR B(n)

```
ENTITY myXOR16 IS
   PORT ( A, B : IN std_logic_vector (15 DOWNTO 0);
                 Y : OUT std_logic_vector (15 DOWNTO 0) );
END myXOR16;

ARCHITECTURE myXOR16_arch OF myXOR16 IS
BEGIN
   PROCESS ( A, B )
   BEGIN
      FOR i IN 0 TO 15 LOOP
         Y(i) <= A(i) XOR B(i);
      END LOOP;
   END PROCESS;
END myXOR16_arch;
```

4. 16-bit XOR

- Entity name
  - myXOR16
- Inputs
  - A, B
- Outputs
  - Y
- Function
  - Y(n) = A(n) XOR B(n)

```
ENTITY myXOR16 IS
   PORT ( A, B : IN std_logic_vector (15 DOWNTO 0);
              Y : OUT std_logic_vector (15 DOWNTO 0) );
END myXOR16;

ARCHITECTURE myXOR16_arch OF myXOR16 IS
BEGIN
   Y <= A XOR B;
END myXOR16_arch;
```

5. D-Latch

- How to implement