Quick Points

- Midterm course evaluation form available on WebCT
- HW #4 – VHDL for synthesis
  - Due Thursday, November 1
  - Work with your project group
  - If in a single person group, work with another single group and submit separately

Recap – PROCESS Block

- List of signals to which the process is sensitive
- Whenever there is an event on any of the signals in the sensitivity list, the process fires
- Every time the process fires, it will run in its entirety
- WAIT statements are NOT allowed in a processes with sensitivity list

Processes in VHDL

- Processes describe sequential behavior
- Processes in VHDL are very powerful statements
  - Allow to define an arbitrary behavior that may be difficult to represent by a real circuit
  - Not every process can be synthesized
  - Use processes with caution in the code to be synthesized
  - Use processes freely in testbenches

Mixed Style Modeling

Design Exercise

- Create the entity declaration for some component of your final project
  - Names, ports, signal types
  - Just the structure

```vhdl
ENTITY entity_name IS
    PORT ( ...
END entity_name;
```
Outline

- Recap
- Dataflow Style
  - Logic Gates
  - Decoders / Encoders
  - Arithmetic Functions
- A Structural Example
- Behavioral Style
  - Registers
  - Counters

Dataflow VHDL

- All concurrent statements
- Major instructions:
  - Concurrent signal assignment (\(\leftarrow\))
  - Conditional concurrent signal assignment (when-else)
  - Selected concurrent signal assignment (with-select-when)
  - Generate scheme for equations (for-generate)

Dataflow Example – Full Adder

```vhdl
ENTITY fulladd IS
  PORT ( x : IN STD_LOGIC ;
         y : IN STD_LOGIC ;
         cin : IN STD_LOGIC ;
         s : OUT STD_LOGIC ;
         cout : OUT STD_LOGIC ) ;
END fulladd ;
ARCHITECTURE dataflow OF fulladd IS
BEGIN
  s <= x XOR y XOR cin ;
  cout <= (x AND y) OR (cin AND x) OR (cin AND y) ;
END dataflow ;
```

Logical Operators

- AND, OR, NAND, NOR, XOR, NOT, XNOR
- Only NOT has order of precedence
- Otherwise, no implied precedence
- Example: \(y = ab + cd\)
  - \(y \leftarrow a \text{ AND } b \text{ or } c \text{ AND } d\); -- Equivalent to
  - \(y \leftarrow ((a \text{ AND } b) \text{ OR } c) \text{ AND } d\); -- Equivalent to
  - \(y = (ab + c)d\)

Arithmetic Operators

- For basic arithmetic operators on std_logic types, use the IEEE libraries
- Standard addition, subtraction, multiplication

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all ;
signal A :  STD_LOGIC_VECTOR(3 downto 0);
signal B :  STD_LOGIC_VECTOR(3 downto 0);
signal C :  STD_LOGIC_VECTOR(3 downto 0);
.....
C <= A + B ;
```

16-bit Addition

```vhdl
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;
ENTITY adder16 IS
  PORT ( Cin : IN STD_LOGIC ;
         X, Y : IN STD_LOGIC_VECTOR(15 DOWNTO 0) ;
         S : OUT STD_LOGIC_VECTOR(15 DOWNTO 0) ;
         Cout : OUT STD_LOGIC ) ;
END adder16 ;
ARCHITECTURE Dataflow OF adder16 IS
BEGIN
  SIGNAL Sum : STD_LOGIC_VECTOR(16 DOWNTO 0) ;
  Sum <= ('0' & X) + Y + Cin ;
  S <= Sum(15 DOWNTO 0) ;
  Cout <= Sum(16) ;
END Dataflow ;
```
### Conditional Signal Assignment

**When - Else**

```
target_signal <= value1 when condition1 else value2 when condition2 else 
                    . . . valueN-1 when conditionN-1 else valueN;
```

### 2:1 Multiplexer

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY mux2to1 IS
    PORT (w0, w1, s : IN STD_LOGIC;
             f : OUT STD_LOGIC);
END mux2to1;

ARCHITECTURE dataflow OF mux2to1 IS
BEGIN
    f <= w0 WHEN s = '0' ELSE w1;
END dataflow;
```

### Selected Signal Assignment

**With - Select-When**

```
with choice_expression select
    target_signal <= expression1 when choices_1,
                    expression2 when choices_2,
                    . . . expressionN when choices_N;
```

### Priority Encoder

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY priority IS
    PORT (w : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
          z : OUT STD_LOGIC);
END priority;

ARCHITECTURE dataflow OF priority IS
BEGIN
    y <= "11" WHEN w(3) = '1' ELSE 
         "10" WHEN w(2) = '1' ELSE 
         "01" WHEN w(1) = '1' ELSE 
         "00" ;
    z <= '0' WHEN w = "0000" ELSE '1';
END dataflow;
```

### Generate Statements

- A way to simplify a pattern of concurrent statements
- Can’t do regular FOR…LOOP in dataflow

```
For - Generate

label: FOR identifier IN range GENERATE
    {Concurrent Statements}
END GENERATE;
```

### 4:1 Multiplexer

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY mux4to1 IS
    PORT (w0, w1, w2, w3 : IN STD_LOGIC;
          s : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
          f : OUT STD_LOGIC);
END mux4to1;

ARCHITECTURE dataflow OF mux4to1 IS
BEGIN
    WITH s SELECT
        f <= w0 WHEN "00",
            w1 WHEN "01",
            w2 WHEN "10",
            w3 WHEN OTHERS;
END dataflow;
```
Parity Example

ARCHITECTURE parity_dataflow OF parity IS
SIGNAL xor_out: STD_LOGIC_VECTOR (7 DOWNTO 0);
BEGIN
xor_out(0) <= parity_in(0);
G2: FOR i IN 0 TO 6 GENERATE
    xor_out(i+1) <= xor_out(i) XOR parity_in(i+1);
end generate G2;
parity_out <= xor_out(7);
END parity_dataflow;

Structural Mapping Example

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY priority_resolver IS
PORT (r : IN STD_LOGIC_VECTOR(5 DOWNTO 0) ;
      s : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
      z : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
END priority_resolver;
ARCHITECTURE structural OF priorityResolver IS
SIGNAL  p : STD_LOGIC_VECTOR (3 DOWNTO 0) ;
SIGNAL  q : STD_LOGIC_VECTOR (1  DOWNTO 0) ;
SIGNAL  ena : STD_LOGIC ;
COMPONENT mux2to1
    PORT (w0, w1, s : IN STD_LOGIC ;
          f : OUT STD_LOGIC ) ;
END COMPONENT ;
COMPONENT priority
    PORT (w : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
      y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0) ;
      z : OUT STD_LOGIC ) ;
END COMPONENT ;
COMPONENT dec2to4
    PORT (w : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
      En : IN STD_LOGIC ;
      y : OUT STD_LOGIC_VECTOR(0 TO 3)) ;
END COMPONENT ;
BEGIN
u1: mux2to1 PORT MAP (w0 => r(0) ,
                     w1 => r(1),
                     s => s(0),
                     f => p(0));
p(1) <= r(2);
p(1) <= r(3);
u2: mux2to1 PORT MAP (w0 => r(4) ,
                     w1 => r(5),
                     s => s(1),
                     f => p(3));
u3: priority PORT MAP (w => p,
                        y => q,
                        z => ena);
u4: dec2to4 PORT MAP (w => q,
                       En => ena,
                       y => z);
END structural;

Behavioral Latch

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY latch IS
PORT ( D, Clock : IN STD_LOGIC ;
       Q : OUT STD_LOGIC ) ;
END latch ;
ARCHITECTURE Behavior OF latch IS
BEGIN
    PROCESS ( D, Clock )
    BEGIN
        IF Clock = '1' THEN
            Q <= D ;
        END IF ;
        END PROCESS ;
END Behavior ;

Behavioral Flip Flop

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY flipflop IS
PORT ( D, Clock : IN STD_LOGIC;
Q : OUT STD_LOGIC);
END flipflop;

ARCHITECTURE Behavior_1 OF flipflop IS
BEGIN
PROCESS ( Clock )
BEGIN
IF Clock'EVENT AND Clock = '1' THEN
Q <= D;
ELSEIF Clock'EVENT AND Clock = '1' THEN
Q <= Q;
END IF;
END PROCESS;
END Behavior_1;

Truth table

<table>
<thead>
<tr>
<th>D</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

N-bit Register with Reset

ENTITY regn IS
GENERIC ( N : INTEGER := 16 );
PORT ( D                     : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
Resetn, Clock : IN STD_LOGIC;
Q : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0) ) ;
END regn ;

ARCHITECTURE Behavior OF regn IS
BEGIN
PROCESS ( Resetn, Clock )
BEGIN
IF Resetn = '0' THEN
Q <= (OTHERS => '0') ;
ELSIF Clock'EVENT AND Clock = '1' THEN
Q <= D ;
END IF;
END PROCESS;
END Behavior ;

4-bit Up-Counter with Reset

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY upcount IS
PORT ( Clock, Resetn, Enable : IN STD_LOGIC;
Q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)) ;
END upcount ;

ARCHITECTURE Behavior OF upcount IS
BEGIN
SIGNAL Count : STD_LOGIC_VECTOR(3 DOWNTO 0) ;
PROCESS ( Clock, Resetn )
BEGIN
IF Resetn = '0' THEN
Count <= "0000" ;
ELSIF ( Clock'EVENT AND Clock = '1' ) THEN
IF Enable = '1' THEN
Count <= Count + 1 ;
ELSEIF Clock'EVENT AND Clock = '1' THEN
Q <= Q;
ELSIF Count <= "0000" THEN
Q <= D;
ELSIF Count <= "0111" THEN
Q <= D;
ELSIF Count <= "1000" THEN
Q <= D;
ELSIF Count <= "1001" THEN
Q <= D;
ELSIF Count <= "1010" THEN
Q <= D;
ELSIF Count <= "1011" THEN
Q <= D;
ELSIF Count <= "1100" THEN
Q <= D;
ELSIF Count <= "1101" THEN
Q <= D;
ELSIF Count <= "1110" THEN
Q <= D;
ELSIF Count <= "1111" THEN
Q <= D;
ELSIF Count <= "0001" THEN
Q <= D;
ELSIF Count <= "0010" THEN
Q <= D;
ELSIF Count <= "0011" THEN
Q <= D;
ELSIF Count <= "0100" THEN
Q <= D;
ELSIF Count <= "0101" THEN
Q <= D;
ELSIF Count <= "0110" THEN
Q <= D;
ELSIF Count <= "0111" THEN
Q <= D;
ELSIF Count <= "1000" THEN
Q <= D;
ELSIF Count <= "1001" THEN
Q <= D;
ELSIF Count <= "1010" THEN
Q <= D;
ELSIF Count <= "1011" THEN
Q <= D;
ELSIF Count <= "1100" THEN
Q <= D;
ELSIF Count <= "1101" THEN
Q <= D;
ELSIF Count <= "1110" THEN
Q <= D;
ELSIF Count <= "1111" THEN
Q <= D;
END IF;
END IF;
END PROCESS;
Q <= Count ;
END Behavior ;

Shift Register With Parallel Load

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY shift4 IS
PORT ( D : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
Enable : IN STD_LOGIC;
Load : IN STD_LOGIC;
Sin : IN STD_LOGIC;
Clock : IN STD_LOGIC;
Q : BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
END shift4 ;

END shift4 ;
ARCHITECTURE Behavior_1 OF shift4 IS
BEGIN
  PROCESS (Clock)
  BEGIN
    IF Clock'EVENT AND Clock = '1' THEN
      IF Load = '1' THEN
        Q <= D;
      ELSIF Enable = '1' THEN
        Q(0) <= Q(1);
        Q(1) <= Q(2);
        Q(2) <= Q(3);
        Q(3) <= Sin;
      END IF;
    END IF;
  END PROCESS;
END Behavior_1;