THE UNIVERSITY OF TEXAS AT DALLAS

CS4341 Digital Logic & Computer Design

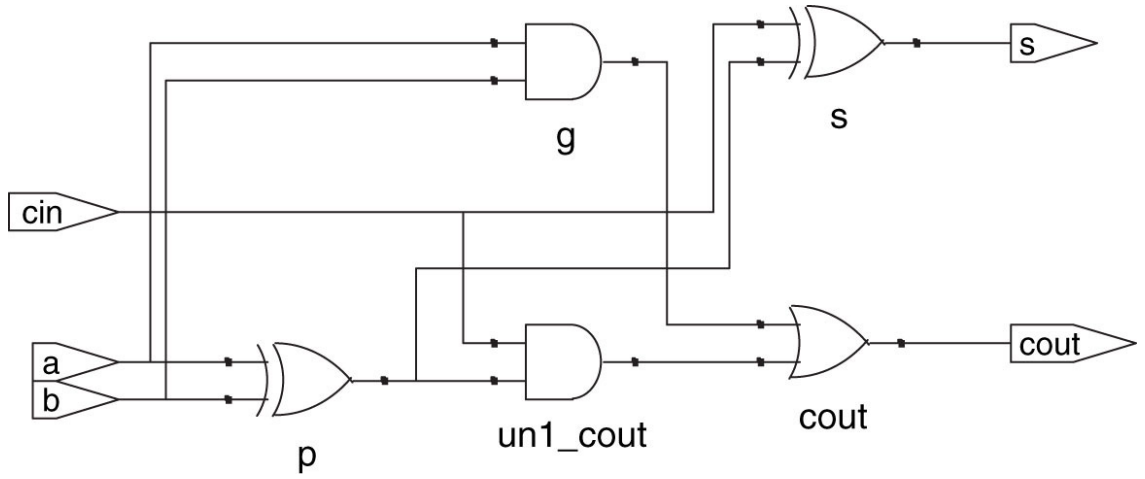
Lecture Notes 20

Omar Hamdy
Assistant Professor
Department of Computer Science

VHDL – C. Logic: Internal Variables

- Internal variables are useful when breaking statements into intermediate steps.
- Internal variables are neither input nor output signals

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity fulladder is
  port(a, b, cin: in STD_LOGIC;
       s, cout: out STD_LOGIC);
end;
architecture synth of fulladder is
  signal p, g: STD_LOGIC;
begin
  p <= a xor b;
  g <= a and b;
  s <= p xor cin;
  cout <= g or (p and cin);
end;
```



VHDL – C. Logic: Precedence

- Operation precedencies are HDL language specific. Hence, it is a good practice to use parenthesis to ensure universality

Verilog

Table 4.1 Verilog operator precedence

	Op	Meaning
Highest	~	NOT
	*, /, %	MUL, DIV, MOD
	+, −	PLUS, MINUS
	<<, >>	Logical Left/Right Shift
	<<<, >>>	Arithmetic Left/Right Shift
	<, <=, >, >=	Relative Comparison
	==, !=	Equality Comparison
	&, ~&	AND, NAND
	^, ~^	XOR, XNOR
	|, ~|	OR, NOR
Lowest	?:	Conditional

VHDL

Table 4.2 VHDL operator precedence

	Op	Meaning
Highest	not	NOT
	*, /, mod, rem	MUL, DIV, MOD, REM
	+, −, &	PLUS, MINUS, CONCATENATE
	rol, ror, srl, sll, sra, sla	Rotate, Shift logical, Shift arithmetic
	=, /=, <, <=, >, >=	Comparison
Lowest	and, or, nand, nor, xor	Logical Operations

VHDL – C. Logic: Z's and X's

- HDL uses Z to indicate a floating value (neither 1 nor 0). Example tristate buffers, when output floats (high impedance Z) if the enable is 0.
- VHDL uses x to indicate an invalid logic level. Example, If a bus is simultaneously driven to 0 and 1 by two enabled tristate buffers.
- At beginning of simulations, state nodes such as flip-flop outputs are initialized to an unknown state (x in Verilog and u in VHDL).
- if a gate receives an illegal or uninitialized input, it may produce an x output, when it can't determine the correct output value
- VHDL STD_LOGIC signals are '0', '1', 'z', 'x', and 'u'
- This is very useful in finding key defects in the design

VHDL – C. Logic: Z's and X's

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity tristate is
  port (a: in STD_LOGIC_VECTOR (3 downto 0);
  en: in STD_LOGIC;
  y: out STD_LOGIC_VECTOR (3 downto 0));
end;
architecture synth of tristate is
begin
  y <= "ZZZZ" when en = '0' else a;
end;
```

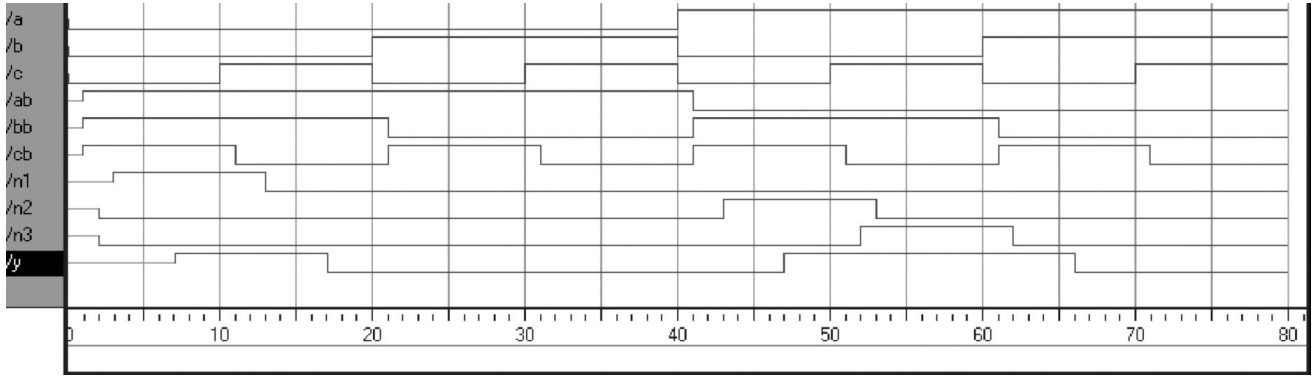
Example: VHDL AND Gate Output for all possible A and B signal values

AND		A				
		0	1	z	x	u
B	0	0	0	0	0	0
	1	0	1	x	x	u
	z	0	x	x	x	u
	x	0	x	x	x	u
	u	0	u	u	u	u

VHDL – C. Logic: Delays

- Sometimes adding delays to gates is helpful during simulation. For example, finding the source of a bad output could be hard if all signals change simultaneously.
- These delays are **ignored** during synthesis; HDL depends on the tpd and tcd specifications

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity example is
  port (a, b, c: in STD_LOGIC;
        y: out STD_LOGIC);
end;
architecture synth of example is
  signal ab, bb, cb, n1, n2, n3: STD_LOGIC;
begin
  ab <= not a after 1 ns;
  bb <= not b after 1 ns;
  cb <= not c after 1 ns;
  n1 <= ab and bb and cb after 2 ns;
  n2 <= a and bb and cb after 2 ns;
  n3 <= a and bb and c after 2 ns;
  y <= n1 or n2 or n3 after 4 ns;
end;
```



VHDL – C. Logic: Input/Output Signal

- VHDL does not allow an output signal to be used as input as well.
- For example, the below code will not compile.
- VHDL solves this problem by introducing buffer port type. A signal connected to a buffer port behaves as an output but may also be used within the module as an input

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity and23 is
  port(a, b, c: in STD_LOGIC;
       v, w: out STD_LOGIC);
end;
architecture synth of and23 is
begin
  v <= a and b;
  w <= v and c;
end;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity and23 is
  port(a, b, c: in STD_LOGIC;
       v: buffer STD_LOGIC;
       w: out STD_LOGIC);
end;
architecture synth of and23 is
begin
  v <= a and b;
  w <= v and c;
end;
```

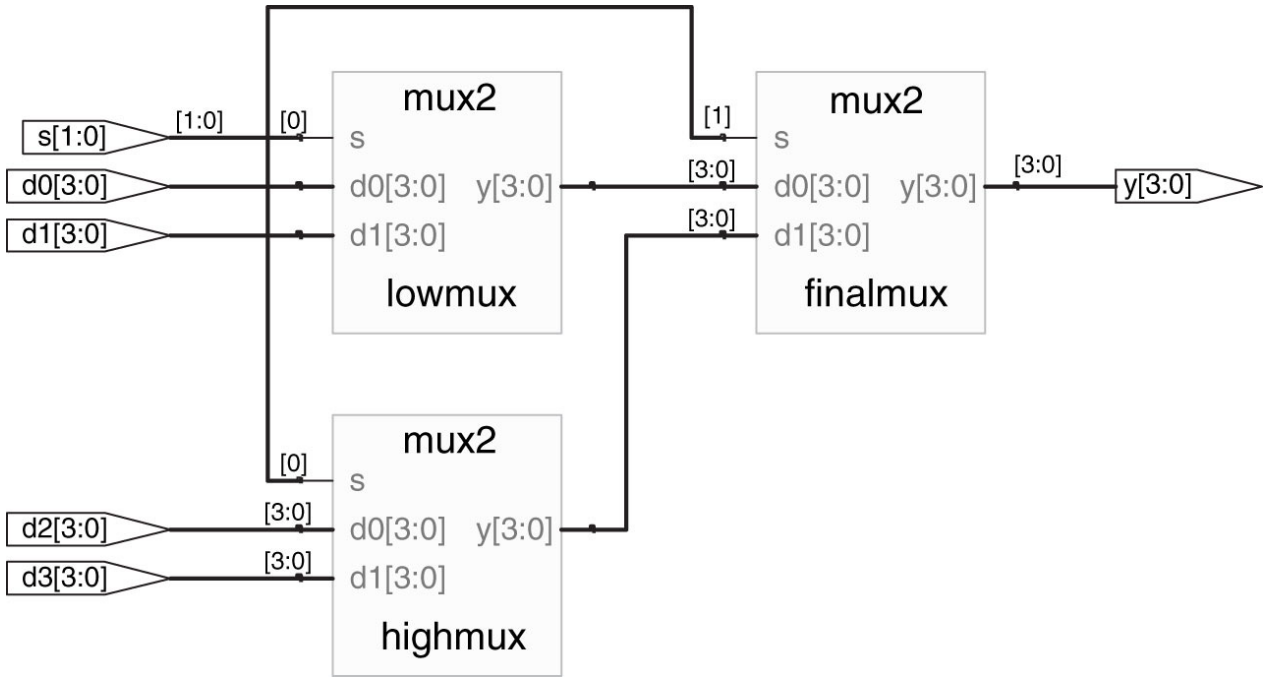
VHDL – C. Logic: enumeration types

- VHDL supports enumeration types as an abstracted way to represent information.

```
type statetype is (S0, S1, S2);
signal state, nextstate: statetype;
```

VHDL – Structural Modeling

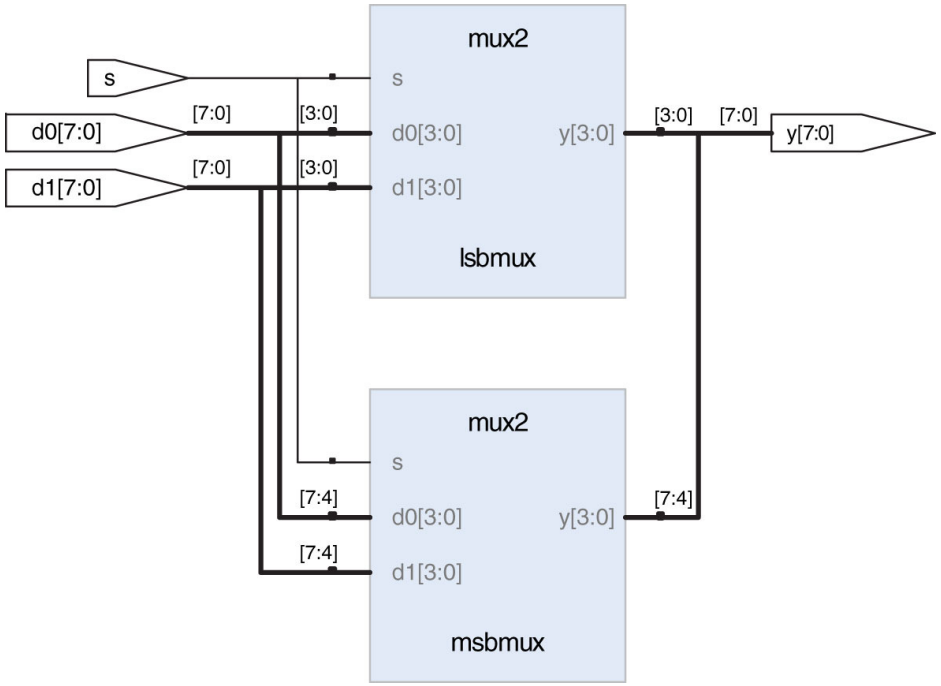
- Structural modeling describes a module in terms of how it is composed of simpler, rather than the relationships between inputs and outputs.
- Example: building 4x1 MUX using 3 2X1 MUX.




```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity mux4 is
  port (D0, D1,
        D2, D3: in STD_LOGIC_VECTOR (3 downto 0);
        S: in STD_LOGIC_VECTOR (1 downto 0);
        Y: out STD_LOGIC_VECTOR (3 downto 0));
end;
architecture struct of mux4 is
  component mux2
    port (d0,
          d1: in STD_LOGIC_VECTOR (3 downto 0);
          s: in STD_LOGIC;
          y: out STD_LOGIC_VECTOR (3 downto 0));
  end component;
  signal low, high: STD_LOGIC_VECTOR (3 downto 0);
begin
  lowmux: mux2 port map (D0, D1, S(0), low);
  highmux: mux2 port map (D2, D3, S(0), high);
  finalmux: mux2 port map (low, high, S(1), Y);
End;
```

VHDL – Structural Modeling

- Structural modeling can also build modules by accessing part of a bus.
- Example: building 8-bit wide 2x1 multiplexer using two of the 4-bit 2x1 multiplexers already defined.



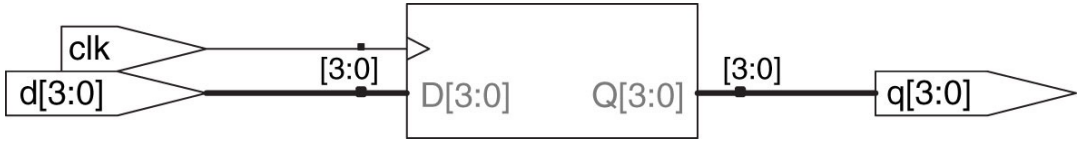

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity mux2_8 is
  port (d0, d1: in STD_LOGIC_VECTOR (7 downto 0);
        s: in STD_LOGIC;
        y: out STD_LOGIC_VECTOR (7 downto 0));
end;
architecture struct of mux2_8 is
  component mux2
    port (d0, d1: in STD_LOGIC_VECTOR(3 downto 0);
          s: in STD_LOGIC;
          y: out STD_LOGIC_VECTOR (3 downto 0));
    end component;
begin
  lsbmux: mux2
    port map (d0 (3 downto 0), d1 (3 downto 0),
              s, y (3 downto 0));
  msbhmux: mux2
    port map (d0 (7 downto 4), d1 (7 downto 4),
              s, y (7 downto 4));
end;
```

VHDL – S. Logic: Registers

- Signals keep their old values until an “event” in the “sensitivity” list takes place that explicitly cause them to change.

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity flop is
  port (clk: in STD_LOGIC;
  d: in STD_LOGIC_VECTOR (3 downto 0) ;
  q: out STD_LOGIC_VECTOR (3 downto 0)) ;
end;
architecture synth of flop is
begin
  process (clk) begin
    if clk’event and clk = ‘1’ then
      q <= d;
    end if;
  end process;
end;
```

```
process (sensitivity list) begin
  statement;
end process;
```




```
if RISING_EDGE (clk) then
  q <= d;
end if;
```

VHDL – S. Logic: Resettable Registers

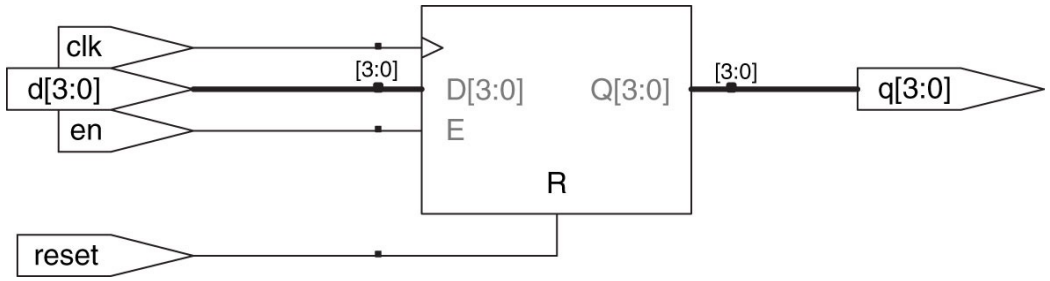
- Reset is a good practice at the start of the system.
- Reset can be synchronous or asynchronous (not easy to identify from the schematic diagram)

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity flopr is
  port (clk. reset: in STD_LOGIC;
  d: in STD_LOGIC_VECTOR (3 downto 0) ;
  q: out STD_LOGIC_VECTOR (3 downto 0)) ;
end;
architecture asynchronous of flopr is
begin
  process (clk, reset) begin
    if reset = '1' then
      q <= "0000";
    elsif clk' event and clk  '1' then
      q <= d;
    end if;
end process;
end;
```

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity flopr is
  port (clk. reset: in STD_LOGIC;
  d: in STD_LOGIC_VECTOR (3 downto 0) ;
  q: out STD_LOGIC_VECTOR (3 downto 0)) ;
end;
architecture synchronous of flopr is
begin
  process (clk) begin
    if clk'event and clk = '1' then
      if reset = '1' then
        q <= "0000";
      else q <= d;
      end if;
    end if;
  end process;
end;
```

VHDL – S. Logic: Enabled Registers

- Recall, enabled registers respond to the clk only when the enable is asserted.
- The following example is for an asynchronously resettable enabled register




```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity flopenr is
  port (clk, reset, en: in STD_LOGIC;
        d: in STD_LOGIC_VECTOR (3 downto 0);
        q: out STD_LOGIC_VECTOR (3 downto 0));
  end;
architecture asynchronous of flopenr is
— — asynchronous reset
begin
  process (clk, reset) begin
    if reset = '1' then
      q <= "0000";
    elsif clk'event and clk = '1' then
      if en = '1' then
        q <= d;
      end if;
    end if;
  end process;
end;
```

To Do List

- Review lecture notes