Introduction to HDLs, FPGAs, and Xilinx tools
HDLs

- A hardware description language (HDL) is a class of languages used for modeling electronics
  - Verilog
  - VHDL
    - to be used in this class.
- Features of HDLs
  - Primarily Concurrent
  - Notion of time (delays)
  - Modular
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FullAdder is
  Port( A: in STD_LOGIC;
        B: in STD_LOGIC;
        Cin: in STD_LOGIC;
        S: out STD_LOGIC;
        Cout: out STD_LOGIC);
end FullAdder;

architecture Behavioral of FullAdder is
  begin
    Cout <= s_and1 or s_and2;
    S    <= s_xor1 xor Cin;
    s_xor1 <= A xor B;
    s_and1 <= A and B;
    s_and2 <= Cin and s_xor1;
  end Behavioral;
VHDL: Standard Logic Type

- VHDL offers numerous data types, but the ones that are of most concern to us are `std_logic` and `std_logic_vector`.
- A `std_logic` can be
  - Usually:
    - '1' to represent logic high
    - '0' to represent logic low
  - Also:
    - 'X' to represent high-impedance
    - 'U' to represent uninitialized
    - 'W' to represent weak (metastatic)
    - 'L' to represent weak low
    - 'H' to represent weak high
VHDL: Standard Logic Type

• A std_logic_vector is a bus of std_logic values
• Specified with a range
  o signal A std_logic_vector(3 downto 0);
  o signal B std_logic_vector(0 to 3);
  o signal C std_logic_vector(7 downto 4);
• can be accessed/assigned
  o as a group
    ▪ A <= "1010";
    ▪ B <= X"A";
  o or individually
    ▪ C(4) <= '0';
  o or with ranges of vectors
    ▪ C <= "01" & B(1 downto 0)
• Order matters!
  o A(3) is '1'
  o B(3) is '0'
  o A <= B(3 downto 0); makes A "0101"
entity RA is
  Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
       B : in  STD_LOGIC_VECTOR (3 downto 0);
       Cin : in  STD_LOGIC;
       S : out  STD_LOGIC_VECTOR (3 downto 0);
       Cout : out  STD_LOGIC);
end RA;

architecture Behavioral of RA is

  signal carry: std_logic (3 downto 1);

  COMPONENT FullAdder
    PORT(
      A : IN std_logic;
      B : IN std_logic;
      Cin : IN std_logic;
      S : OUT std_logic;
      Cout : OUT std_logic
    );
  END COMPONENT;

begin

  --explicit association
  Adder0: FullAdder PORT MAP(
    A => A(0), B => B(0), Cin => Cin, S => S(0), Cout => carry(1));

  --positional assotiation
  Adder1: FullAdder PORT MAP(A(1), B(1), carry(1), S(1), carry(2));
  Adder2: FullAdder PORT MAP(A(2), B(2), carry(2), S(2), carry(3));
  Adder3: FullAdder PORT MAP(A(3), B(3), carry(3), S(3), Cout);

end Behavioral;
LIBRARY ieee;                    -- line 1
USE ieee.std_logic_1164.all;     -- line 2
USE ieee.std_logic_arith.all;   -- line 3
USE ieee.std_logic_signed.all;  -- line 4

ENTITY adder IS                -- line 5
  PORT (Cin     : IN  STD_LOGIC;
        a, b    : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
        s      : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        Cout   : OUT STD_LOGIC);
END adder;                      -- line 10

ARCHITECTURE example OF adder IS -- line 11
  SIGNAL Sum : STD_LOGIC_VECTOR(4 DOWNTO 0);    -- line 12
BEGIN                             -- line 13
  Sum <= a + b + Cin;             -- line 14
  s <= Sum(3 DOWNTO 0);          -- line 15
  Cout <= Sum(4);                -- line 16
END example;                     -- line 17
Example 2 – 16-bit full adder

Entity adder is
port( a, b : in bit_vector (15 downto 0);
        cin : in bit;
        s : out bit_vector (15 downto 0);
        cout : out bit );
end entity adder;

architecture behavioral of adder is
    signal c : bit_vector (15 downto 0);
begin
    array : for i in 0 to 15 generate
    begin
        first : if i=0 generate
        begin
            cell : component full_adder
            port map(a(i), b(i), cin, s(i), c(i));
        end generate first;
        other : if i/=0 generate
        begin
            cell : component full_adder
            port_map(a(i), b(i), c(i-1), s(i), c(i));
        end generate other;
    end generate array;
    cout <= c(15);
end architecture behavioral;
VHDL: Processes

• Processes are used for sequential programming
• Processes are triggered by sensitivity lists.
  o entered on changes to these signals
• clk' event is true when the change was in clk

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counter4 is
  Port ( clk : in  STD_LOGIC;
         rst : in  STD_LOGIC;
         count : out STD_LOGIC_VECTOR (3 downto 0));
end counter4;

architecture Behavioral of counter4 is

  signal count_s : std_logic_vector(3 downto 0);

begin

  countproc: process(rst,clk)
  begin
    if(rst='1') then
      count_s <= "0";
    elsif (clk'event and clk='1') then
      count_s <= count_s + 1;
    end if;
  end process;

  count <= count_s;

end Behavioral;
```
Library ieee;
USE ieee.std_logic_1164.all;
ENTITY simple_register IS
  GENERIC (N : INTEGER := 4);
  PORT (I : IN STD_LOGIC_VECTOR (N-1 DOWNTO 0);
         Clock, Clear, Preset : IN STD_LOGIC;
         Q : OUT STD_LOGIC_VECTOR (N-1 DOWNTO 0));
END simple_register;

ARCHITECTURE simple_memory OF simple_register IS
BEGIN
  PROCESS (Preset, Clear, Clock)
  BEGIN
    IF Preset = '0' THEN
      Q <= (OTHERS => '1');
    ELSIF Clear = '0' THEN
      Q <= (OTHERS => '0');
    ELSIF (Clock'EVENT AND Clock = '1') THEN
      Q <= I;
    END IF;
  END PROCESS;
END simple_memory;
VHDL: Variables and Time

- Variables may be used exclusively in processes.
- They are completely independent of the time effects of VHDL signals.
- Variables update immediately
- Signals update at the end of the process

```vhdl
architecture Behavioral of Example is
  signal c,d : std_logic;
begin
  exampleProcess: Process(clk)
    variable a,b : integer;
  begin
    --assume the values of a,b,c, and d are all 0 initially
    a := 5;
    b := b + a;
    c <= '1';
    d <= d or c;
  end process;
end architecture;
```
**Example: 4-to-2 Priority Encoder**

- The output of a priority encoder indicates the active input that has the highest priority. When an input with the higher priority is asserted, the other inputs with the lower priority are ignored.

- Assume that d0 has the lowest priority and d3 the highest priority.

<table>
<thead>
<tr>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
<th>F1</th>
<th>F2</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The outputs F1 and F2 represent the binary number that identifies the highest priority input set to 1. Output Z is set to 1 if at least one of the inputs is one. It is set to zero when all inputs is zero.
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY priority IS
    PORT (d :IN STD_LOGIC VECTOR (3 DOWNTO 0);
f :OUT STD_LOGIC VECTOR (1 DOWNTO 0);
z :OUT STD_LOGIC);
END priority;

ARCHITECTURE arch_cond OF priority IS
BEGIN
    f <= "11" WHEN d(3) = ‘1’ ELSE
        "10" WHEN d(2) = ‘1’ ELSE
        "01" WHEN d(1) = ‘1’ ELSE
        "00";
    z <= ‘0’ WHEN d = "0000" ELSE ‘1’;
END arch_cond;
FPGAs

- Field programmable gate arrays (FPGAs) are used to model HDL circuits
- Used to test circuit designs
  - Can implement any logic in an ASIC
  - Can change logic without re-fabricating
  - Less area efficient than ASICs
  - Slower than ASICs
FPGAs: Configurable Logic Blocks

logic cell
FPGAs: Lookup Tables

- LUTs behave like a programmable memory
- Program data at each address to behave like the intended logic
FPGAs: Configurable Logic Blocks

logic cell

3-LUT

3-LUT

FA

DFF

mux

in

out

carry in

clk

carry out

clk

out
FPGAs: Slices

Spartan3E Documentation: may be found here.
entity ro is
  Port (en : in STD_LOGIC;
        rout : out STD_LOGIC);
end ro;

architecture Behavioral of ro is
  -- component counter32
  -- Port (a : in STD_LOGIC;
  --       b : out STD_LOGIC);
  -- end component;

  component norGate
    PORT (a : in STD_LOGIC;
           b : out STD_LOGIC);
  end component;

  component andGate
    Port (a : in STD_LOGIC;
         b : in STD_LOGIC;
         c : out STD_LOGIC);
  end component;

  signal tmp : std_logic_vector(15 downto 0);
  attribute keep : string;
  attribute keep of tmp : signal is "TRUE";
begin
  not_1 : norGate port map(tmp(0), tmp(1));
  and_2 : andGate port map(tmp(1), tmp(2), tmp(3));
  not_3 : norGate port map(tmp(2), tmp(3));
Xilinx ISE: Design Flow

- Four main design flow steps:
  1. Synthesize
  2. Translate
  3. Map
  4. Place & Route
Xilinx ISE: PlanAhead
Xilinx ISE: UCFs

• LOC specifies a particular slice or I/O
• Required to connect to other on-board devices
  o LEDs, buttons, etc.
• Other constraints exist
  o RLOC - relative location
  o BEL - specify LUT
  o Many more
• More information may be found in the Xilinx Constraints Guide
-- Clock process definitions
clk_process : process
begin
  clk <= '0';
  wait for clk_period/2;
  clk <= '1';
  wait for clk_period/2;
end process;

-- Stimulus process
stim_proc: process
begin
  rst <= '1';
  wait for 100 ns;
  rst <= '0';
  -- insert stimulus here
  wait;
end process;

END;
Concluding Remarks

• In lab you will:
  o Practice the ISE design flow.
  o Simulate and verify a given VHDL model.
  o Program an FPGA board.

• Next lecture:
  o We will introduce the Hardware Trojan problem.