VHDL (I)

A Full Adder

- Before introducing you to VHDL, let's build a simple circuit, so that we can use it to calculate binary adder.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>SUM</th>
<th>Carry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- Compare the truth table with the ones of those basic gates
  - Carry is AND
  - SUM is XOR

- This is a half adder. Why “half”? [A full adder would have a carry in as well]
  - A full adder can be implemented as two half-adders and one OR gate.
  - The SUM output of the full add is XOR on the SUM output of the half add and the carry in.
  - The Carry out is 1 if either A and B are both 1 (left input to the OR gate) or exactly one of them is 1 and the Carry In bit is also 1.
A Full Adder in VHDL

- In this course, we use VHDL to assist our circuit design.
- So, if we have a more complex circuit, we can leverage some VHDL to help with the validation.

- What is VHDL?
  - An acronym for Very High Speed Integrated Circuit Hardware Description Language.
  - A program language used to describe a logic circuit by function, data flow behavior, or structure.

- What is GHDL?
  - Like Java and C, VHDL programs need to be compiled before running.
  - GHDL is a compiler for VHDL.
  - GHDL is short for G Hardware Design Language. Currently, G has no meaning.

- How do you get VHDL and GHDL?
  - The dwarves in the robotics lab have VHDL and GHDL ready for you to use.

- Now, let’s see how to use VHDL to describe the full adder we talked in last class.

```vhdl
-- Comments in VHDL start with --
-- Ying Li
-- A full adder

-- import useful packages
library ieee;
use ieee.std_logic_1164.all; -- provide enhanced signal types

-- filename must be the same as entity name
-- entity defines the inputs and outputs of the circuit
entity adder is
  -- A, B, Ci are inputs
  -- S and Co are the outputs
  port (A, B, Ci : in std_logic; -- std_logic represent one bit signal
        S, Co : out std_logic); -- that can take on the value 0 or 1
end adder;

-- architecture define the circuit
architecture behavior of adder is
begin
  -- statements in the architecture enclosed by begin and end are executed concurrently
  -- compute the sum
  s <= A xor B xor Ci; -- <= assign the result of A xor B xor ci to s
  -- compute the carry
  co <= (A and B) or ((a xor b) and ci); -- VHDL is case-insensitive
end behavior;

-- to analyze this design, using ghdl -a adder.vhd
```
- The above code is in a file named adder.vhd.
- It describes the full adder circuit with 3 inputs, 2 outputs, two XOR gates, two AND gates, and one OR gate.
- To compile the program, use the command `ghdl -a adder.vhd`
  - `–a` means analysis

- Now, we need a test program for the circuit so that we can tell whether the circuit works correctly.

```vhdl
library ieee;
use ieee.std_logic_1164.all;

-- A testbench has no ports
entity addertest is
end addertest;

architecture behavior of addertest is
  -- Declaration of the component that will be instantiated
  component adder
    port (A, B, Ci: in std_logic;
          S, Co: out std_logic);
  end component;

  -- the adder signals
  signal I0, I1, I2, O0, O1 : std_logic;

begin
  -- component instantiation
  -- connect the inputs and outputs of the entity to the local signals
  adder1 : adder port map (A=>I0, B=>I1, Ci=>I2, Co=>O0, S=>O1);

  I0 <= '0', '1' after 4 ns; -- send signals to port a, the signal is 0 in the first 4 ns and 1 afterward
  I1 <= '0', '1' after 2 ns, '0' after 4 ns, '1' after 6 ns;
  I2 <= '0', '1' after 1 ns, '0' after 2 ns, '1' after 3 ns, '0' after 4 ns, '1' after 5 ns, '0' after 6 ns, '1' after 7 ns, '0' after 8 ns;

end behavior;
```

- The above code is in a file named addertest.vhd
- The testbed gives three inputs signals to the three inputs of the full adder. In the first 8 ns, the first input has value 0 during the first 4 ns and 1 afterward. The second input starts with 0 and varies the value every 2 ns. The third input starts with 0 and varies the value every ns.
- We need to compile the program using `ghdl -a addertest.vhd`
- Then, elaborate the testbench: `ghdl -e addertest`
  - This step may not generate output files on some platforms.
- The last step is to run the test program using `ghdl -r addertest --vcd=adder.vcd`
  - This step will generate an `adder.vcd` file. You can use `gtkwave` to open the vcd file and validate the correctness of the full adder behavior.

- Open the `adder.vcd` file using `gtkwave`, you will get a window like this

- Select the all the signals and click “Insert”, then click the “Zoom Fit” button. You will get a window like this. The green signals can help us to validate the full adder.