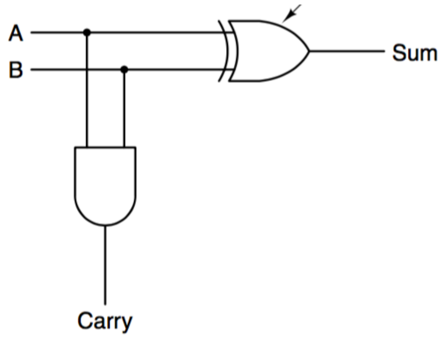
VHDL

A Full Adder

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

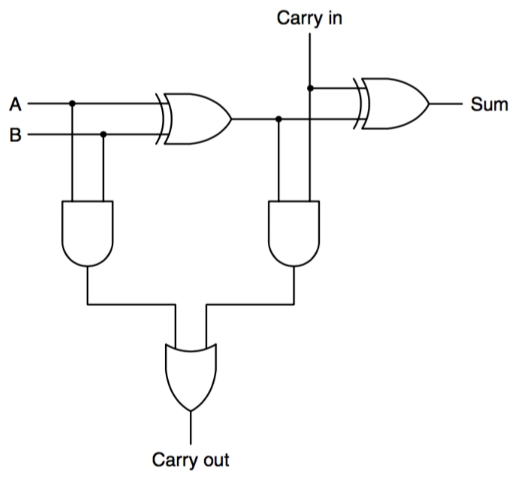
A	B	SUM	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Compare the truth table with the ones of those basic gates
  - Carry is AND
  - SUN 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	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



- Now, we know how to design a full adder using a pencil and paper and validate it by setting different values at inputs and checking the output values manually.
- The validate is do-able if the circuit is simple. But, if we have a more complex circuit, we need to leverage some tools to help with the validation.
- In this course, we use VHDL to assist our circuit design.

- 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.

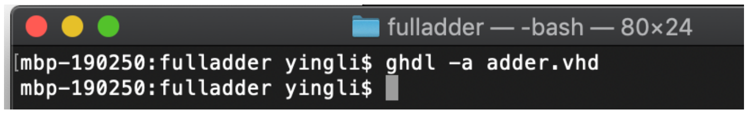
- What is GHDL?
  - Like Java and C, VHDL programs need to be compiled before running.
  - GHDL is a compiler for VHDL.
  - GHDL is short for G Hardware Design Language. Currently, G has no meaning.

- How do you get VHDL and GHDL?
  - The dwarves in the robotics lab have VHDL and GHDL ready for you to use.
  - If you want to have them installed on your computer, here is a blog I recommend you to read https://medium.com/@maxime.bonn/vhdl-on-your-mac-264ff6cc0600. I followed the instructions in this blog and installed VHDL and GHDL on my laptop successfully.

- Now, let's see how to use VHDL to describe the full adder.

```
-- 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



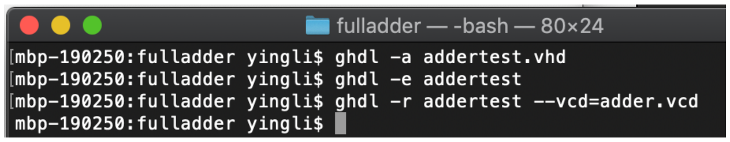
```
mbp-190250:fulladder yingli$ ghdl -a adder.vhd
mbp-190250:fulladder yingli$ 
```

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

```
-- Ying Li
-- A testbench for the full adder

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;
```

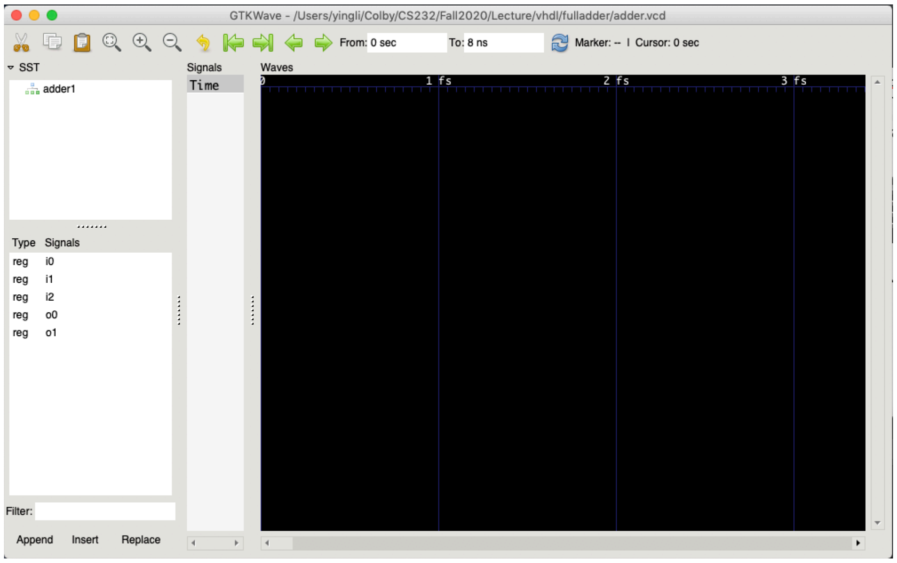
- The above code is in
- The above code is in a file named `addertest.vhd`
- The testbed gives three inputs signals to the three inputs of the full adder. In the first 8 ns, the first input has value 0 during the first 4 ns and 1 afterward. The second input starts with 0 and varies the value every 2 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 valid the correctness of the full adder behavior.

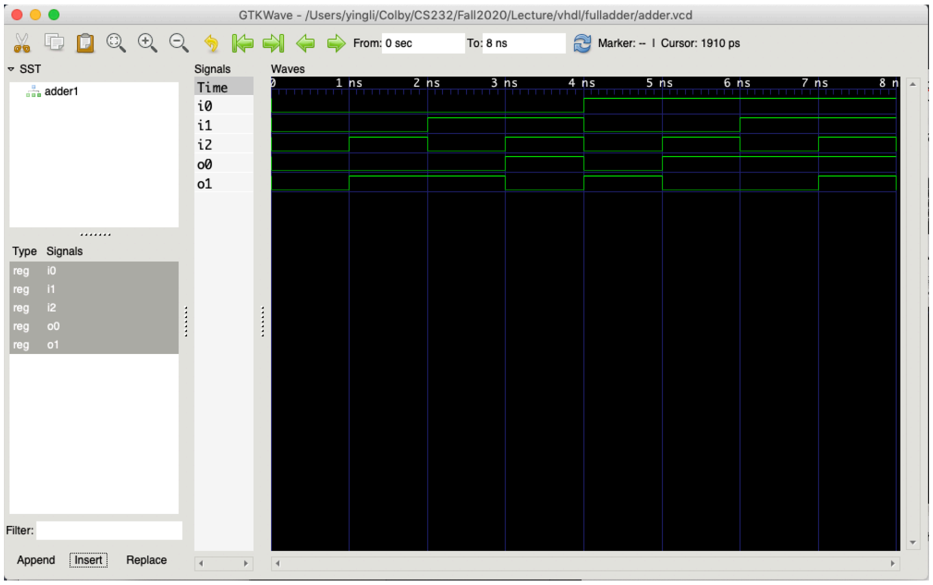


```
fulladder — -bash — 80×24
mbp-190250:fulladder yingli$ ghdl -a addertest.vhd
mbp-190250:fulladder yingli$ ghdl -e addertest
mbp-190250:fulladder yingli$ ghdl -r addertest --vcd=adder.vcd
mbp-190250:fulladder yingli$
```

- Open the adder.vcd file using gktwave, 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.


GTKWave - /Users/yingli/Colby/CS232/Fall2020/Lecture/vhdl/fulladder/adder.vcd
From: 0 sec
To: 8 ns
Marker: -- | Cursor: 1910 ps
SST
adder1
Signals
Time
i0
i1
i2
o0
o1
Waves
Type Signals
reg i0
reg i1
reg i2
reg o0
reg o1
Filter:
Append
Insert
Replace