Description

The one bit adder was implemented using four OR gates, four AND gates, and two NOT gates. In this lab we must design the full adder using structural modeling. Therefore, I first defined a 2-input AND gate, 2-input OR gate, 3-input AND gate, 3-input OR gate, 2-input NOR gate, and NOT gate (each in their own entity).

In the full-adder entity the three input ports A, B, and cin and the two output ports sum and cout were declared. The previously defined gates were also all declared as components. Eight internal signals (D-K) are declared to perform all of the intermediate steps required in the structural diagram. All of the gates shown in the structural diagram were instantiated (A1 as and gate #1, A2 as and gate #2, etc).

VDHL Code

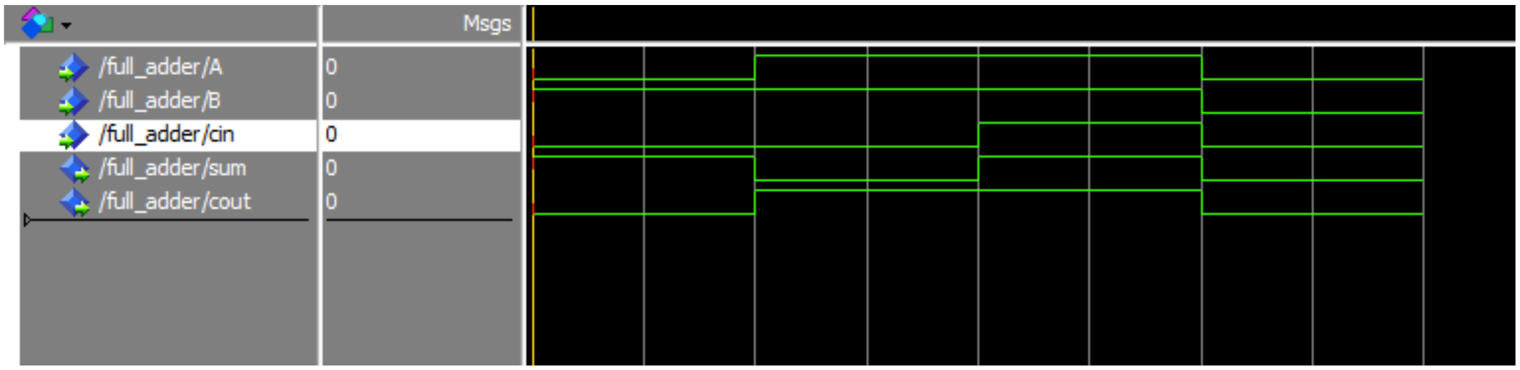
```
--2 input or gate
library ieee;
use ieee.std_logic_1164.all;
entity OR2 is
	port(in1, in2: in std_logic; out1: out std_logic);
end entity;
architecture behav of OR2 is
begin
	out1 <= in1 or in2;
end architecture;

 --2 input and gate
library ieee;
use ieee.std_logic_1164.all;
entity AND2 is
	port(in1, in2: in std_logic; out1: out std_logic);
end entity;
architecture behav of AND2 is
begin
	out1 <= in1 and in2;
end architecture;
```

```
--3 input or gate
library ieee;
use ieee.std_logic_1164.all;
entity OR3 is
        port(in1, in2, in3: in std_logic; out1: out std_logic);
end entity;
architecture behav of OR3 is
begin
        out1 <= in1 or in2 or in3;
end architecture;

--3 input and gate
library ieee;
use ieee.std_logic_1164.all;
entity AND3 is
        port(in1, in2, in3: in std_logic; out1: out std_logic);
end entity;
architecture behav of AND3 is
begin
        out1 <= in1 and in2 and in3;
end architecture;

--2 input nor gate
library ieee;
use ieee.std_logic_1164.all;
entity NOR1 is
        port(in1, in2: in std_logic; out1: out std_logic);
end entity;
architecture behav of NOR1 is
begin
        out1 <= in1 nor in2;
end architecture;

--1 input not gate
library ieee;
use ieee.std_logic_1164.all;
entity NOT1 is
        port(in1: in std_logic; out1: out std_logic);
end entity;
```

```
architecture behav of NOT1 is
begin
	out1 <= not in1;
end architecture;

--1 bit full adder
library ieee;
use ieee.std_logic_1164.all;
entity FULL_ADDER is
	port(A, B, cin: in std_logic; sum, cout: out std_logic);
end FULL_ADDER;
architecture FA_struct of FULL_ADDER is
	component OR2 is
		port(in1, in2: in std_logic; out1: out std_logic);
	end component OR2;
	component AND2 is
		port(in1, in2: in std_logic; out1: out std_logic);
	end component AND2;
	component OR3 is
		port(in1, in2, in3: in std_logic; out1: out std_logic);
	end component OR3;
	component AND3 is
		port(in1, in2, in3: in std_logic; out1: out std_logic);
	end component AND3;
	component NOR1 is
		port(in1, in2: in std_logic; out1: out std_logic);
	end component NOR1;
	component NOT1 is
		port(in1: in std_logic; out1: out std_logic);
	end component NOT1;
	signal D, E, F, G, H, I, J, K : std_logic;
begin
	O1 : OR2 port map(A,B,D);
	A1 : AND2 port map(A,B,E);
	A2 : AND2 port map(D,cin,F);
	N1 : NOR1 port map(E,F,G);
	O2 : OR3 port map(A,B,cin,H);
	A3 : AND3 port map(A,B,cin,I);
	A4 : AND2 port map(G,H,J);
```

```
        N2 : NOR1 port map(J,I,K);
        NT1 : NOT1 port map(G,cout);
        NT2 : NOT1 port map(K,sum);
end architecture FA_struct;
```

Simulation



Analysis

The full adder in the first clock cycle received A=1, B=0, cin=0. The result is sum=1 and cout=0. The next clock pulse took inputs 1, 1, and 0 and produced sum = 0 and cout = 1. The next clock pulse took inputs 1, 1, and 1 and produced sum = 1 and cout = 1. Finally, the last clock pulse took inputs of all 0 and produced sum = 0 and cout = 0.

Structural modeling for this simple 1-bit full adder is much more laborious than the equivalent behavioral model. For example, the data flow model of this circuit would have only required two lines of logic expressions (one for each output of the full adder). The power of structural modeling lies in the more complex circuits that have a very complicated logic expression and function.