1: AND2

Description
AND2 combines two one bit signals 'in0' and 'in1'. The output is '1' if both signals are '1'.

Modeling Type
I'm using dataflow as the modeling type. The reason I choose it is that the 'AND2' gate is a very simple circuit, and dataflow is usually used for small and simple circuits.

Waveform

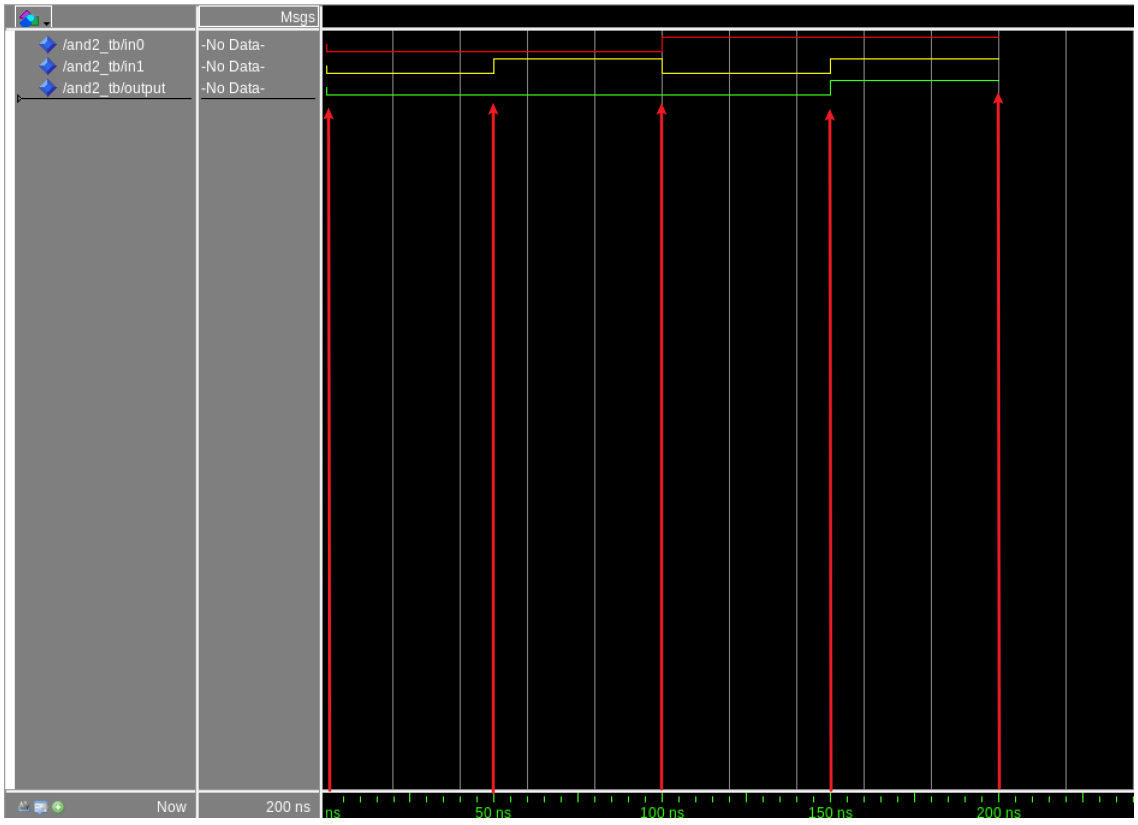

Figure 1: Simulation Waveform of AND2

According to 'AND2' gate's logic, $output = in0$ AND $in1$.
In the figure above, we can see the $output$ signal has four values corresponding to 4 different combination of values of $in0$ and $in1$ signals.
At 0 ns, $in0$ and $in1$ both changes from U to 0, $output$ changes from U to 0. ($output = 0$ AND $0 = 0$)
At 50 ns, $in1$ changes from 0 to 1, $output$ is still 0.($output = 0$ AND $1 = 0$)
At 100 ns, $in1$ changes from 1 to 0, $in0$ changes from 0 to 1, $output$ is still 0.($output = 1$ AND $0 = 0$)
At 150 ns, $in1$ changes from 0 to 1, $output$ changes from 0 to 1.($output = 1$ AND $1 = 1$)
At 200 ns, my testbench ends.

2: MUX5

Description
MUX5 is used to select a 5 bits output signal from two 5 bits input signals according to the select signal 'sel'.

Modeling Type
I'm using behavioral as the modeling type. The reason I choose it is that the MUX5's output is depend on the 'sel' signal, and behavioral is usually used to model how circuit outputs will react to circuit inputs.

Waveforms

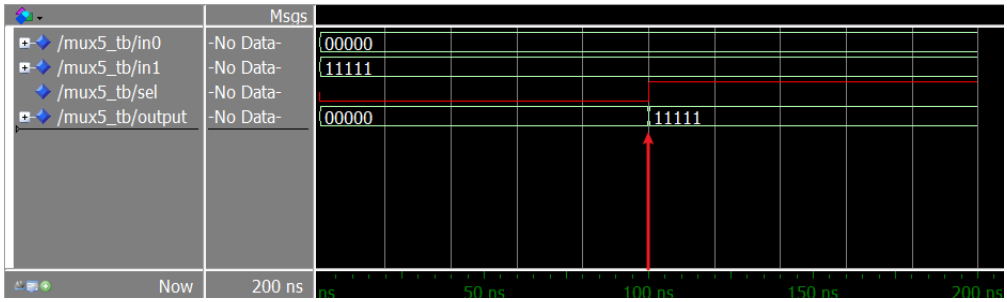

Figure 2: Simulation Waveform of MUX5

In my MUX5, 'in0' is set to "00000" and 'in1' is set to "11111". I use 'sel' to decide which signal in 'in0' and 'in1' would be the output signal 'output'. During 0-100 ns, 'sel' is '0'. 'in0' is selected to be the 'output', so the 'output' is the same as 'in0', and equals to "00000". During 100-200 ns, 'sel' is '1'. 'in1' is selected to be the 'output', so the 'output' is the same as 'in1', and equals to "11111".

3: MUX64

Description
MUX64 is used to select a 64 bits output signal from two 64 bits input signals according to the select signal 'sel'.

Modeling Type
I'm using behavioral as the modeling type. The reason I choose it is that the MUX64's output is depend on the 'sel' signal, and behavioral is usually used to model how circuit outputs will react to circuit inputs.

Waveforms

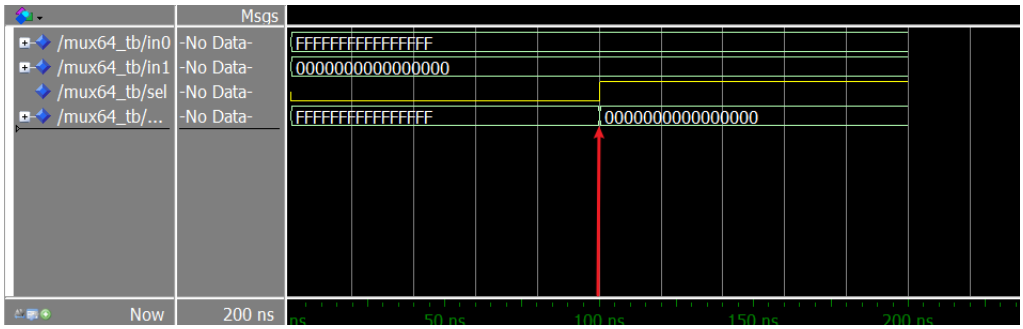

Figure 3: Simulation Waveform of MUX64

In my MUX64, 'in0' is set to x"FFFFFFFFFFFFFFFF" and 'in1' is set to x"0000000000000000". I use 'sel' to decide which signal in 'in0' and 'in1' would be the output signal 'output'.
During 0-100 ns, 'sel' is '0'. 'in0' is selected to be the 'output', so the 'output' is the same as 'in0', and equals to x"FFFFFFFFFFFFFFFF".
During 100-200 ns, 'sel' is '1'. 'in1' is selected to be the 'output', so the 'output' is the same as 'in1', and equals to x"0000000000000000".

4: ShiftLeft

Description
ShiftLeft discard two bits on the far left, and append two '0' bits on the far right. All bits are the same as shifted left by 2 bits.

Modeling Type
I'm using dataflow as the modeling type. The reason I choose it is that the logic relationship between the output and input of 'ShiftLeft' is very simple and straight forward, and dataflow is usually used for small and simple circuits.

Waveforms

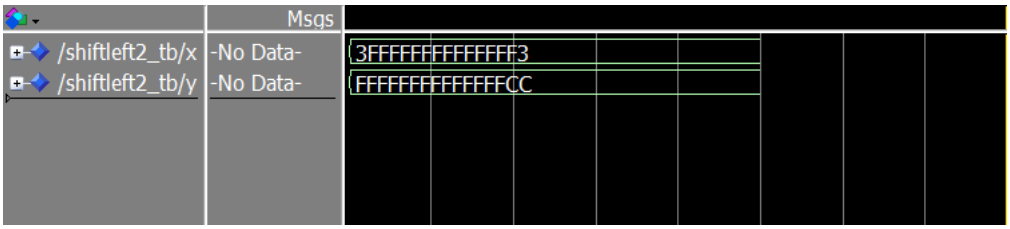

Figure 4: Simulation Waveform of ShiftLeft

In my 'ShiftLeft', 'x' is the origin signal, which equals to x"3FFFFFFFFFFFFF3". "y" is the shifted signal. The eight bits on the far left of 'x' is "00111111" in binary, the eight bits on the far right of 'x' is "11110011" in binary, and all the bits between are '1's.
After discarding two bits on the far left, the eight bits on the far left is "11111111", the eight bits on the far right is "11110011", and this signal has 62 bits. And then "00" is appended on the far right, the eight bits on the far right is "11001100", which is "CC" in hexadecimal, and the signal has 64 bits again. The final output signal 'y' is x"FFFFFFFFFFFFFFCC" in hexadecimal.

5: SignedExtend

Description
SignExtend is used to increase a 32 bits binary number to 64 bits binary number, while preserving the number's sign and value.

Modeling Type
I'm using behavioral as the modeling type. The reason I choose it is that the SignedExtend's output depends on the left most bit of the input signal, and behavioral is usually used to model how circuit outputs will react to circuit inputs.

Waveforms

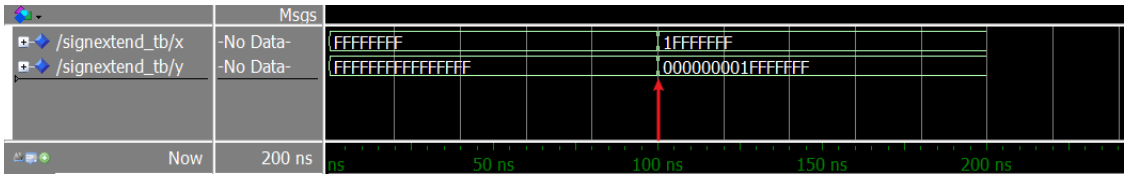

Figure 5: Simulation Waveform of SignedExtend

In my 'SignedExtend', 'x' is the original 32 bits binary number, 'y' is the extended 64 bits binary number.
During 0 to 100 ns, x is set to -1 in decimal, which is 32 bits '1' in binary, and x"FFFF FFFF" in hexadecimal. According to two's complement, this 'x' is negative, so I append 32 bits '1' on the left most of 'x' in binary. And make 'y' 64 bits '1', which is x"FFFF FFFF FFFF FFFF" in hexadecimal radix.
During 100 to 200 ns, x is set to 536870911 in decimal, which is "0001 1111 1111 1111 1111 1111 1111 1111" in binary, and x"1FFF FFFF" in hexadecimal. Now x is positive, so I append 32 bits '0' on the left most of 'x' in binary. And 'y' is x"0000 0000 1FFF FFFF" in hexadecimal.

6: PC

Description
PC is a 64 bits rising-edge triggered register with write-enable and synchronous reset. At the time of clk's rising edge when write_enable = '1', PC will pass AddressIn to AddressOut. And when rst = '1', PC will reset AddressOut to 0x0.

Modeling Type
I'm using behavioral as the modeling type. The reason I choose it is that the PC's output depends on three input signals 'clk', 'write_enable', and 'rst. And behavioral is usually used to model how circuit outputs will react to circuit inputs.

Waveforms

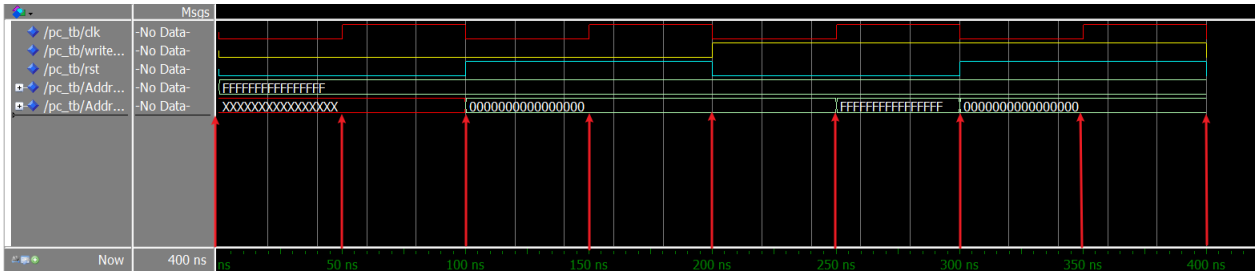

Figure 6: Simulation Waveform of PC

In my PC, 'clk' is clock, 'rst' is reset control, 'write_enable' is write control, 'AddressIn' is the adress waiting to be passed, and 'AddressOout' is output address. In other cases, the PC does not do any work. 'AddressIn' is set to x"FFFF FFFF FFFF FFFF" all the time.

clk	rst	write_enable	time(ns)
0	0	0	0-50
0	0	1	200-250
0	1	0	100-150
0	1	1	300-350
1	0	0	50-100
1	0	1	250-300
1	1	0	150-200
1	1	1	350-400

Table 1: Inputs Combinations in PC

From the above table, we can see eight different combinations of three input signals. When rst = '1', no matter what other inputs' values are and no matter what AddressIn's value is, the output address is set to x"0000 0000 0000 0000". This happens at 100-200ns, 300-400ns, AddressOut = x"0000 0000 0000 0000" in these two period.

When clk is a rising edge (changing from '0' to '1') and write_enable = '1', PC writes AddressIn into AddressOut. This happens at 250ns and 350ns. AddressOut = AddressIn = x"FFFF FFFF FFFF FFFF" at 250ns. But rst = '1' at 350ns, so AddressOut doesn't change into x"FFFF FFFF FFFF FFFF", it's still reset to x"0000 0000 0000 0000".
For the rest of the time, the value of AddressOut remains the same.