Lab 9.1

Lab 9.1

- In this lab you will:
  - Complete the description of a simple combinational function
  - Become familiarized with the synthesis tool
- In directory lab9.1 you will find the following files:
  - comb.v — Arbitrary combinational function (incomplete)
  - tb.v — Self-checking testbench

Lab 9.1

- Edit the file "comb.v" and locate the *always* block. Notice the empty sensitivity list.
- Add the name of all input signals to the *always* block.
- Simulate the design using the provided testbench:

```
% verilog tb.v comb.v
```

- If the testbench does not declare your circuit as correct, make sure your sensitivity list includes all input signals (and only input signals).
- Compile the design using the synthesis tool of your choice. Detailed instructions are available in Appendix C. Use only the tool you are most likely to use at work.

Lab 9.1: Synopsys

- Invoke dc_shell using the command:

```
% dc_shell
```

- Read the design using the command:

```
dc_shell> read -format verilog comb.v
```

- Pay attention to any warning or error issued by the synthesis compiler.
  - It should indicate if some signals are missing or extraneous from the sensitivity list of the *always* block.
- Add any missing signals or remove any extraneous signals. Resimulate and recompile until no warnings are produced.

Lab 9.1: Synopsys

- Map your design to gates using the command:

```
dc_shell> compile
```

- Report the area and timing performance of the final design using the commands:

```
dc_shell> report_area
dc_shell> report_timing
```

  Exit the synthesis tool using the command:

```
dc_shell> quit
```

Lab 9.2

Lab 9.2

- In this lab you will:
  - Complete the description of a simple combinational function
- In directory lab9.2 you will find the following files:
  - decode.v 2-to-4 decoder (incomplete)
  - tb.v Self-checking testbench

Lab 9.2

- The circuit to describe is a 2-to-4 decoder. The following truth table describes the function:

CODE	DECODE
00	0001
01	0010
10	0100
11	1000

- Add the necessary statements in the module to complete a synthesizable functional description of the decoder.

Lab 9.2

- Verify the correctness of your description by simulating the decoder model using the provided testbench.

```
% ... tb.v decode.v
```

- Once the functionality is declared correct, synthesize your design.
- Compare the performance of your design with that of others and with the proposed solution.
  - Compare the description of your design with another and try to understand why one is faster and/or smaller.

Lab 9.2 : Optional

- Describe the decoder function using only continuous assignments
  - Verify correctness, then synthesize.
- Which solution yielded:
  - A working design in less time?
  - A smaller and/or faster design?
  - A description easier to understand and maintain?

Lab 9.3

Lab 9.3

- In this lab you will:
  - Describe a complex combinational logic circuit using abstract statements
- In directory lab9.3 you will find the following files:
  - bit_count.v Bit counter (incomplete)
  - tb.v Self-checking testbench

Lab 9.3

- The function of the circuit is to combinationally count the number of bits that are set to '1' in a 32-bit input vector.
- A second 4-bit output is set to '1', if there is a '1' in the corresponding byte in the input vector.
- Examples:

INPUT_VECTOR	COUNT	BYTE
00000000_00000000_00000000_00000000	0	0000
00000001_00000000_10000000_00000000	2	1010
00000011_00000000_00000001_00011000	5	1011
00000000_11111111_11111111_00000000	16	0110
11111111_11111111_11111111_11111111	32	1111

Lab 9.3

- Verify the correctness of your description by simulating the decoder model using the provided testbench.

```
% ... tb.v bit_count.v
```

- Once the functionality is declared correct, synthesize your design.
- Compare the performance of your design with that of others and with the proposed solution.
  - Compare the description of your design with another and try to understand why one is faster and/or smaller.

Lab 9.4

Lab 9.4

- In this lab you will:
  - Learn how to avoid inferring latches
  - Deal with incomplete specifications
- In directory lab9.4 you will find the following files:
  - logic_unit.v Logical unit (incomplete)
  - tb.v Self-checking testbench

Lab 9.4

- The logical unit takes as input three arguments:
  - Operands A & B: 3-bit 2's complement numbers
  - Opcode
- Depending on the selected opcode, the output "C" is set to:

OPCODE	C
0000	A&B
0001	A|B

- Depending on the selected opcode, the output "T" is set to:

OPCODE	T
1000	'1' if A < 0, '0' otherwise
1001	'1' if B < 0, '0' otherwise

Lab 9.4

- Add the necessary statements in the module to complete a synthesizable functional description of the logical unit.
- Verify the correctness of your description by simulating the decoder model using the provided testbench.

```
% ... tb.v logic_unit.v
```

- Once the functionality is declared correct, synthesize your design.
- Compare the performance of your design with that of others and with the proposed solution.
  - Compare the description of your design with another and try to understand why one is faster and/or smaller.

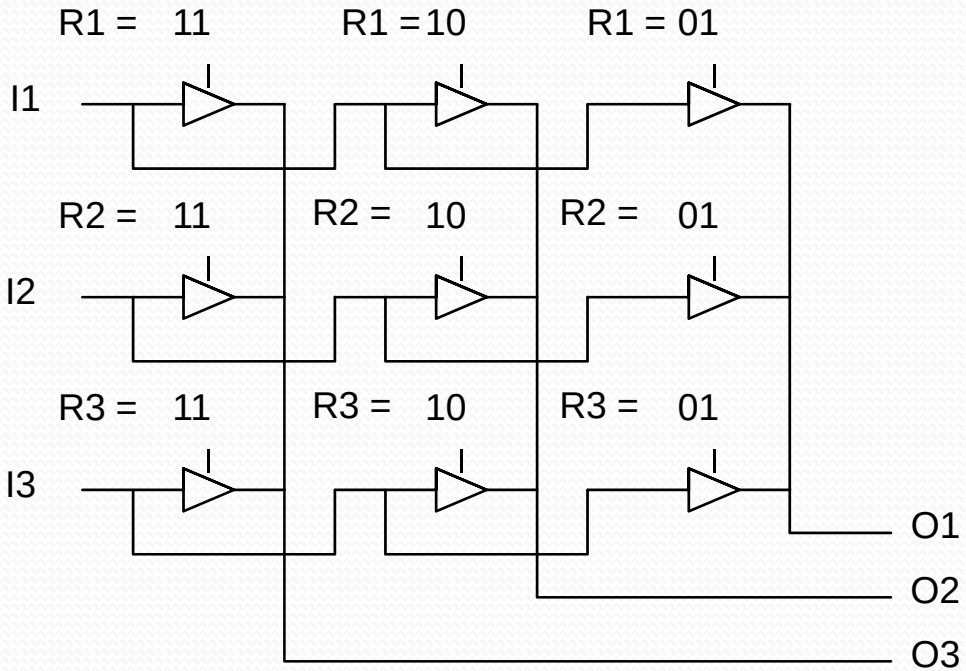
Lab 9.5

Lab 9.5

- In this lab you will:
  - Learn how to infer tristate devices
  - Differentiate between multiplexers and internal busses
- In directory lab9.5 you will find the following files:
  - xbar.v Cross-bar switch (incomplete)
  - tb.v Self-checking testbench

Lab 9.5

- The design is a 3x3 crossbar switch is described in the schematic below.



Lab 9.5

- For example, if R1 is set to "11, then input I1 is routed to output "O3". If R1 is set to "00", then input I1 is not routed to any output.
- It is possible for the crossbar switch to leave an output floating (e.g. "O1" is left floating if none of R1, R2, nor R3 is set to "01").
- It is possible for the crossbar switch to create contention on an output (e.g. there will be contention on output "O2", if more than one of R1, R2, or R3 is set to "10").
- Hint: Using a single *always* block produces a multiplexer; the *always* block doesn't infer tristate devices.
  - Why?

Lab 9.5

- Add the necessary statements in the module to complete a synthesizable functional description of the crossbar switch.
- Verify the correctness of your description by simulating the decoder model using the provided testbench

```
% ... tb.v xbar.v
```

- Once the functionality is declared correct, synthesize your design.
- Compare the performance of your design with that of others and with the proposed solution.
  - Compare the description of your design with another and try to understand why one is faster and/or smaller.

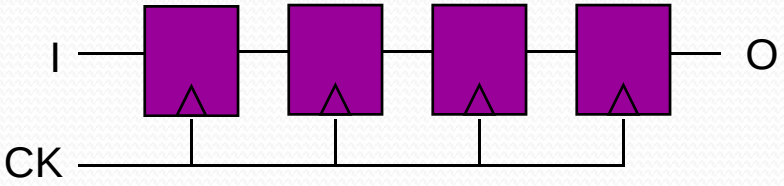
Lab 9.6

Lab 9.6

- In this lab you will:
  - Learn how to infer flip-flops
  - Control the number of flip-flops inferred
- In directory lab9.6 you will find the following files:
  - shifter.v Linear shifter (incomplete)
  - tb.v Self-checking testbench

Lab 9.6

- Add the necessary *always* block in the SHIFTER module to infer the following circuit:



- The clock is active on the rising edge.
- Be careful not to infer 5 flip-flops instead of 4.

Lab 9.6

- Add the necessary statements in the module to complete a synthesizable functional description of the linear shifter.
- Verify the correctness of your description by simulating the decoder model using the provided testbench:

```
% ... tb.v shifter.v
```

- Once the functionality is declared correct, synthesize your design.
- Compare the performance of your design with that of others and with the proposed solution.
  - Compare the description of your design with another and try to understand why one is faster and/or smaller.