CSC3050 Tutorial

Introduction to Verilog

Teaching Assistants

- Wei Dai (戴维)
- Email: weidai@link.cuhk.edu.cn
- Office Hour:
  - We. 21:00 – 22: 00 or by appointment

- Mickey MA (马毓琦)
- Email: mickeyma@cuhk.edu.cn
- Office Hour:
  - Tu. 21:00 – 22: 00 or by appointment
-

- Tutorial Room
  - Time: Mo. Tu. We. 19:00 - 21:00 (Consulting sessions)
  - ZOOM ID: 962 2765 8133
  - Password: CSC3050

Overview of Verilog

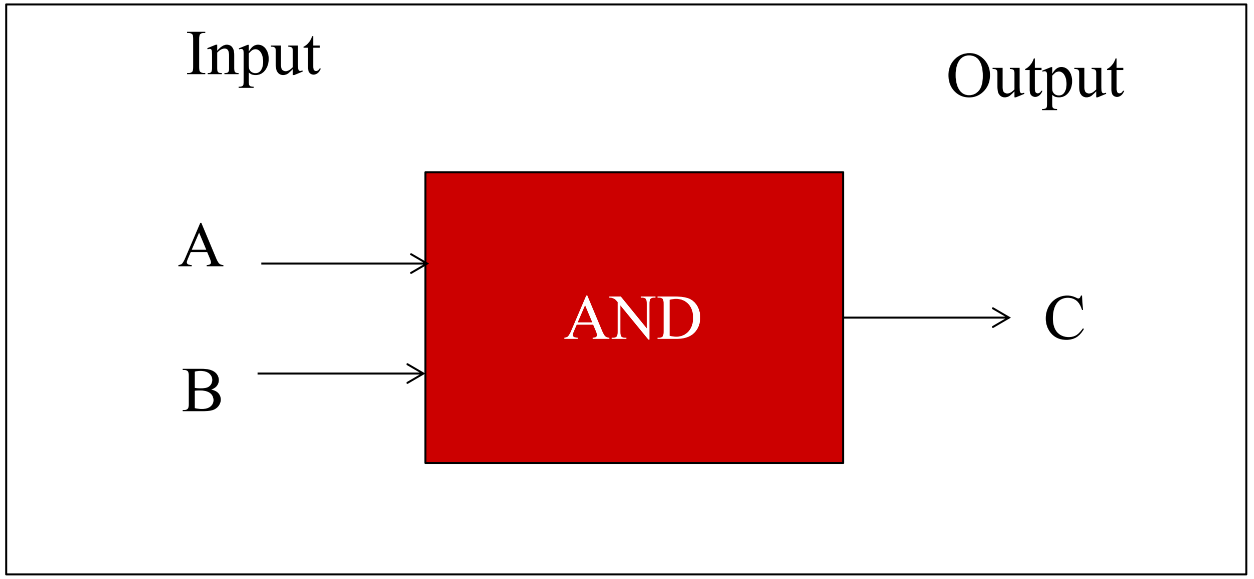
- Objectives
- Verilog Basics
  - Notation
  - Keywords & Constructs
  - Operators
- Types of Descriptions
  - Structural
  - Dataflow
  - Behavioral & Hierarchical

Objectives

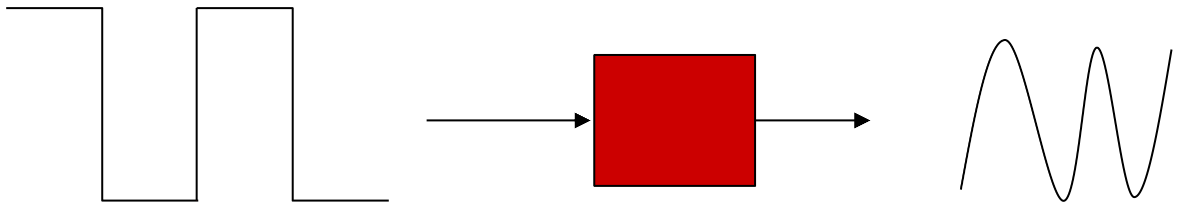
- To become familiar with the hardware description language (HDL) approach to specifying designs
  - Be able to read a simple Verilog HDL description
  - Be able to write a simple Verilog HDL description using a limited set of syntax and semantics
  - Understanding the need for a “hardware view” when reading and writing an HDL

What is HDL?

Describe a **Module**



Test a Module



Verilog Notation - 1

- Verilog is:
  - Case sensitive
  - Based on the programming language C
- Comments
  - Single Line

    `//                    [end of line]`
  - Multiple Line

    ```
    /*
                                                  */
    ```
- List element separator: ,
- Statement terminator: ;

Verilog Notation - 2

- Binary Values for Constants and Variables
  - 0
  - 1
  - X,x - Unknown
  - Z,z – High impedance state (open circuit)
- Constants
  - n'b[integer]: 1'b1 = 1, 8'b1 = 000000001, 4'b0101= 0101, 8'bxxxxxxxx, 8'bxxxx = 0000xxxx
  - n'h[integer]: 8'hA9 = 10101001, 16'hf1= 0000000011110001
- Identifier Examples
  - Scalar: A,C,RUN,stop,m,n
  - Vector: sel[0:2], f[0:5], ACC[31:0], SUM[15:0], sum[15:0]

Verilog Keywords & Constructs - 1

- Keywords are lower case
- `module` – fundamental building block for Verilog designs
  - Used to construct design hierarchy
  - Cannot be nested
- `endmodule` – ends a module – not a statement => no “;”
- Module Declaration
  - `module` *`module_name`* `(`*`module_port`*`,` *`module_port`*`, …);`
  - Example: `module full_adder (A, B, c_in, c_out, S);`

Verilog Keywords & Constructs - 2

- **Input Declaration**
  - Scalar
    - **input** *list of input identifiers*;
    - Example: **input A, B, c_in;**
  - Vector
    - **input**[*range*] *list of input identifiers*;
    - Example: **input[15:0] A, B, data;**
- **Output Declaration**
  - Scalar Example: **output c_out, OV, MINUS;**
  - Vector Example: **output[7:0] ACC, REG_IN, data_out;**

Verilog Keywords & Constructs - 3

- Two Major Data Type
  - wire: wire in a chip.
  - reg: register for storing data.
- Primitive Gates
  - not, and, or, nand, nor, xor, xnor
  - Syntax: *gate_operator instance_identifier (output, input_1, input_2, …)*
  - Examples:

```
and A1 (F, A, B); //F = A B
or O1 (w, a, b, c)
   O2 (x, b, c, d, e);    //w=a+b+c,x=b+c+d+e
```

Verilog Operators - 1

- **Bitwise Operators**
  - ~ NOT
  - & AND
  - | OR
  - ^ XOR
  - ^~ or ~^ XNOR
- **Example:**

```
input[3:0] A, B;
output[3:0] Z ;
assign Z = A | ~B;
```

Verilog Operators - 2

- **Arithmetic Operators**

  +, -, (plus others)
- **Logical & Relational Operators**

  !, &&, | |, = =, !=, >=, <=, >, < (plus others)
- **Concatenation & Replication Operators**

  {identifier_1, identifier_2, …}

  {n{identifier}}

  - **Examples: {REG_IN[6:0],Serial_in},**

    {8 {1’b0}}

Dataflow Verilog

- Circuit function can be described by assign statements using Boolean equations. (Concurrency)
- assign [delay] LHS_net = RHS_expression:

```
module fig519d (A0, B0, C0, C1, S0);
  input A0, B0, C0;
  output C1, S0;
  wire[1:0] N;
  assign N[1] = ~(A0 & B0);
  assign N[0] = ~(A0 | B0);
  assign C1 = ~((N[1] & ~C0) | N[0]);
  assign S0 = (~N[0] & N[1])^(~(~C0));
endmodule
```

Behavioral Verilog - 1

- There is often a sequential process associated with event control in the always statement.

```
module FA_Seq (A, B, Cin, Sum, Cout) ;
  input  A, B, Cin ;
  output  Sum, Cout;
  reg Sum, Cout;
  always @ (A or B or Cin)
     begin
     Sum = (A ^ B) ^ Cin
     Cout = A & Cin;
     end
endmodule
```

Behavioral Verilog - 2

initial statement includes a sequential process, which will be executed at 0ns, and after completion, it will be suspended forever.

```
module Test;
  initial
    begin
        Pop = 0
        Pip = 0
        #5 Pop = 1
        #3 Pid = 1
    end
endmodule
```

Structural Verilog

- Circuits can be described by a netlist as a text alternative to a diagram - Example :

```
module decoder(A0, B0, C0, C1, S0);
  input A0, B0, C0;
  output C1, S0;
  wire[1:7] N;
  not  G1 (N[3],C0), G2 (N[5],N[2]), G3 (N[6],N[3]);
  nand G4 (N[1],A0,B0);
  nor  G5 (N[2],A0,B0), G6 (C1,N[2],N[4]);
  and  G7 (N[4],N[1],N[3]), G8 (N[7],N[1],N[5]);
  xor  G9 (S0,N[6],N[7]);
endmodule
```

Hierarchical Verilog

```
module add (X, Y, C_in, S);
  input [3:0] X, Y;
  input C_in;
  output [3:0] S;
  assign S = X + Y + {3'b0, C_in};
endmodule
module M1comp (data_in, data_out, comp);
  input[3:0] data_in;
  input comp;
  output [3:0] data_out;
  assign data_out = {4{comp}} ^ data_in;
endmodule
```

Key points about Module Structure

- If you use Verilog to implement some functions, you must first find out which ones need to be executed concurrently and which ones need to be executed sequentially.

- “Assign” statement, “always” block is executed concurrently. Their sequence will not affect the logics of the function.

- Inside “always” block, logic is executed sequentially.

Test Bench

- Test Bench is a test module used to verify whether the working state of the designed circuit is correct:
- Add activation to the input signal of the circut, and verify whether the curcuit works normally by observing the output signal.

Common Form:

```
module test;
   reg ...;
   wire ...;
   initial begin ...; ...; ...; end    //generate testing signal
   always #delay begin ...; end  //generate testing signal
   // Instance reference of the module under test
   initial begin ...; ...; ...; end  //record the output signal
 endmodule
```