Digital System Design ELEC373/473

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

Structural Verilog

Module Styles

- Modules can be specified in different ways
  1) **Structural**: connect primitives and modules
  2) **RTL** (Register Transfer Level) :
     - use continuous assignments
  3) **Behavioral**: use **initial** and **always** blocks
     - Note that “initial” is primarily for simulation rather than for synthesis.
- A single module can use more than one method.

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

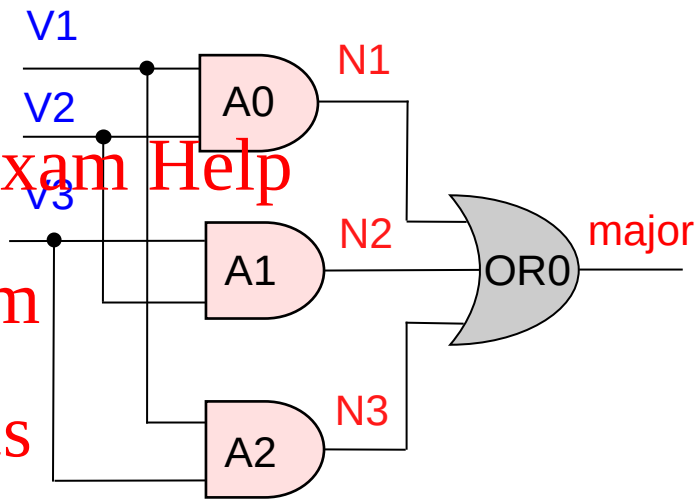
Verilog Structural Model

- A schematic in text form
- Build up a circuit from gates/flip-flops
  - Gates are primitives (part of the language)
  - Flip-flops are User Defined Primitives (UDP)
- Structural design
  1) Create module interface (inputs and outputs)
  2) Instantiate the gates in the circuit
  3) Declare the internal wires needed to connect the gates
  4) Put the names of the wires in the correct port locations of the gates (For primitives, outputs always come first)

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Structural Example (Majority Circuit)

```
module Majority (major, V1, V2, V3);
/*  start your comments with a forward
    slash and star and finish it with
    a star and forward slash.   */
    output   major;
    input    V1, V2, V3;
    wire     N1, N2, N3;
    and      A0 (N1, V1, V2);
    and      A1 (N2, V2, V3);
    and      A2 (N3, V3, V1);
    or       Or0(major, N1, N2, N3);
  // Use two forward slashes for one line comments
endmodule      /* Each statement in Verilog must terminate with
                  a semicolon except endmodule.*/
```



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

All identifiers (names) in Verilog have a scope (i.e., domain of definition) that is local to the module, function, task or named block in which they are declared.

Identifiers

- Identifiers (names) are composed of a sequence of
  - case sensitive alphabetic characters
  - digits (0 to 9)
  - underscore (_) and ($) symbol
- Identifiers cannot start with digits or the ($) symbol
- Spaces are not allowed in an identifier
- An identifier can be up to 1024 characters long.

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Legal Identifier	Illegal Identifier
Full_Adder	Full-Adder
Decoder2	2Decoder
_$Multiplier1	$Multiplier1
Half_Adder	Half Adder

- Use the same rules for module names in Quartus II or you may get errors during the simulation

Representing Numbers

- Representation: <size>’ <base> <number>
  - size => number of **BITS** (regardless of base used)
  - base => base the given number is specified in
  - number => the actual value in the given base
- Can use different bases
  - Decimal (d or D) – default if no base specified!
  - Hex (h or H)
  - Octal (o or O)
  - Binary (b or B)
- Size defaults to at least 32…
  - You should specify the size explicitly!
  - Why create 32-bit register if you only need 5 bits?
  - May cause compilation errors on some compilers

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

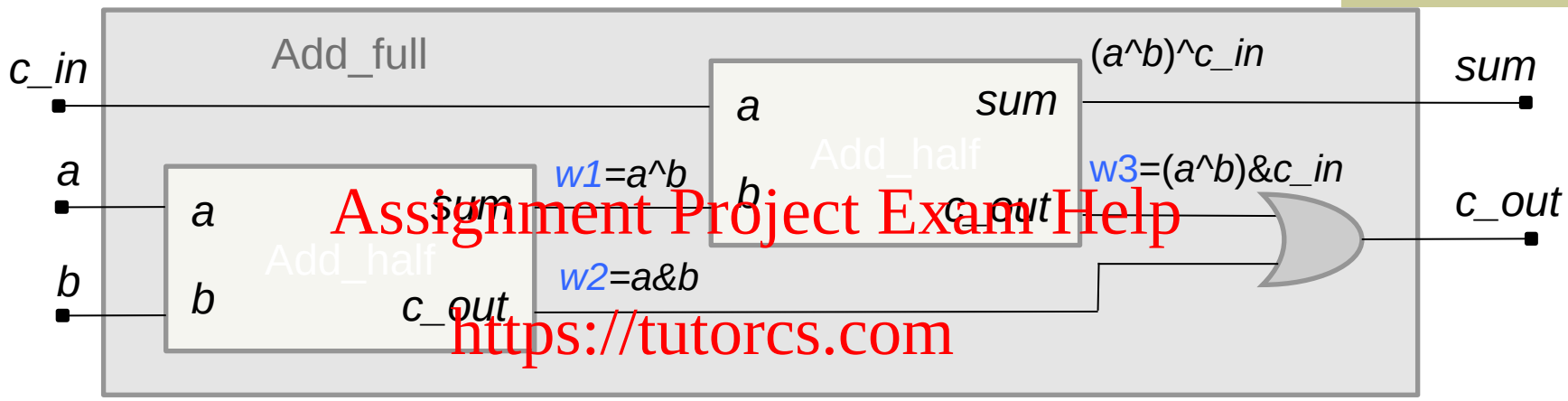
Number Examples

Number	Decimal Equivalent	Actual Binary
4'd3	3	0011
8'ha	10	00001010
8'o26	22	00010110
5'b111	7	00111
8'b0101_1101	93	01011101
8'bx1101	-	xxxx1101
-8'd6	-6	11111010

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

*Numbers with MSB of **x** or **z** are extended with that value*

Full Adder Example



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

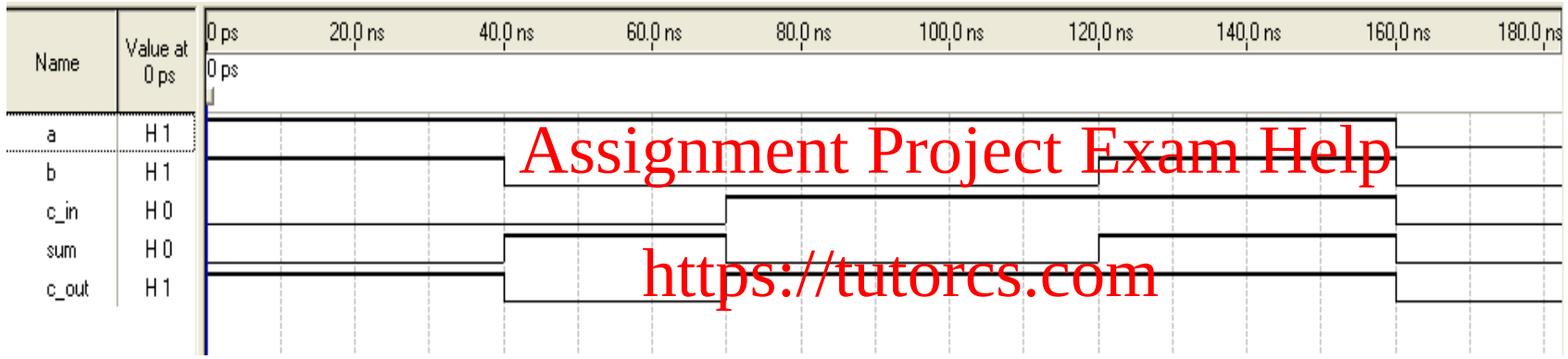
```
module Add_full (sum, c_out, a, b, c_in);
    input       a, b, c_in;
    output      sum, c_out;
    wire        w1, w2, w3;
    Add_half    M1 (w1, w2, a ,b);
    Add_half    M2 (sum, w3, c_in , w1);
    or          (c_out, w2, w3);
endmodule
```

Module instance name

```
module Add_half ( sum, c_out, a ,b);
    input       a, b;
    output      c_out, sum;
    xor         (sum, a, b);
    and         (c_out, a, b);
endmodule
```

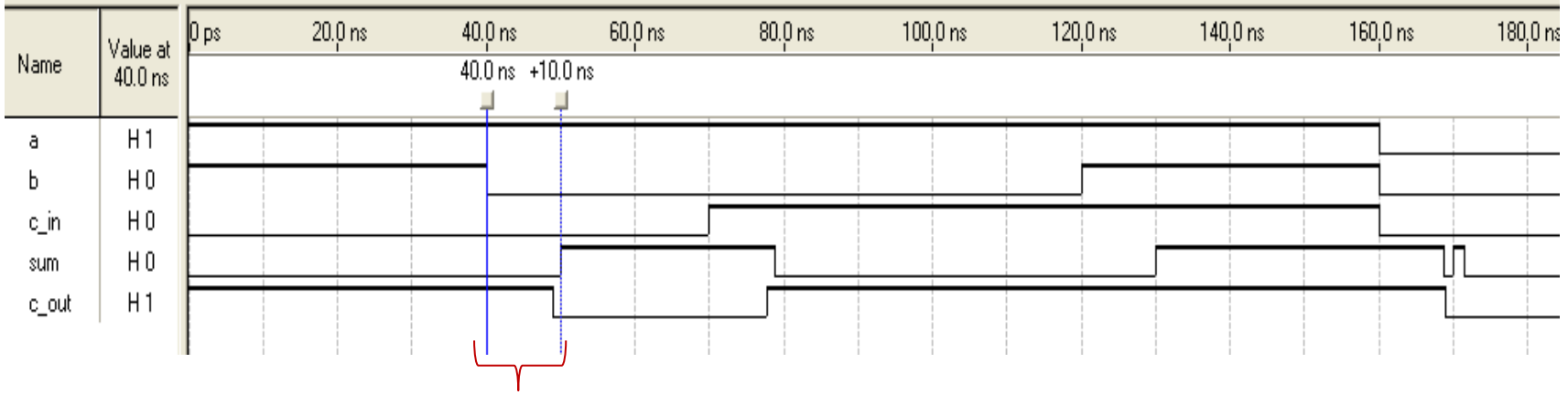
Functional and Timing Simulations with Quartus II

Functional Simulation



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

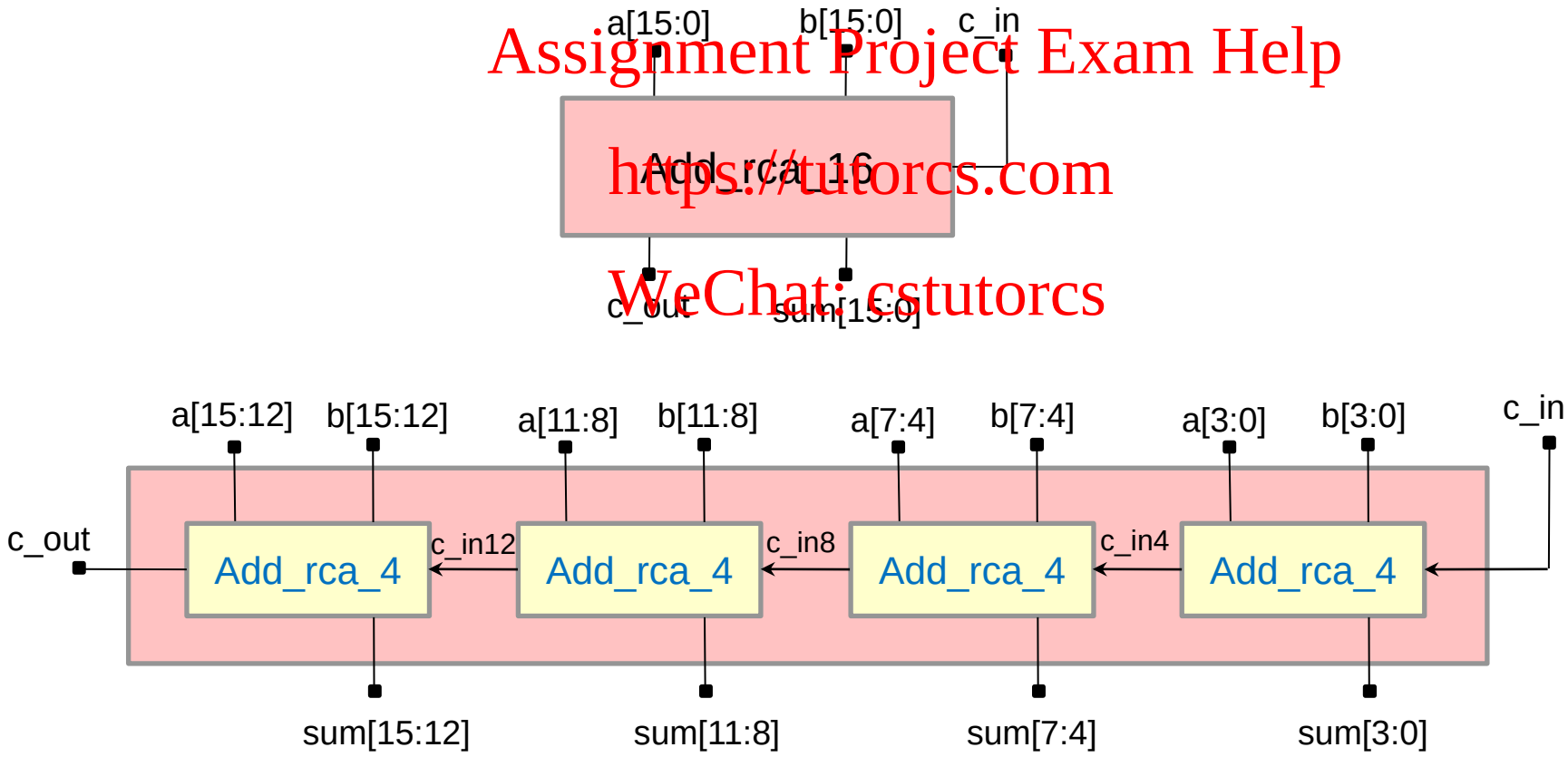
Timing Simulation



Propagation delay

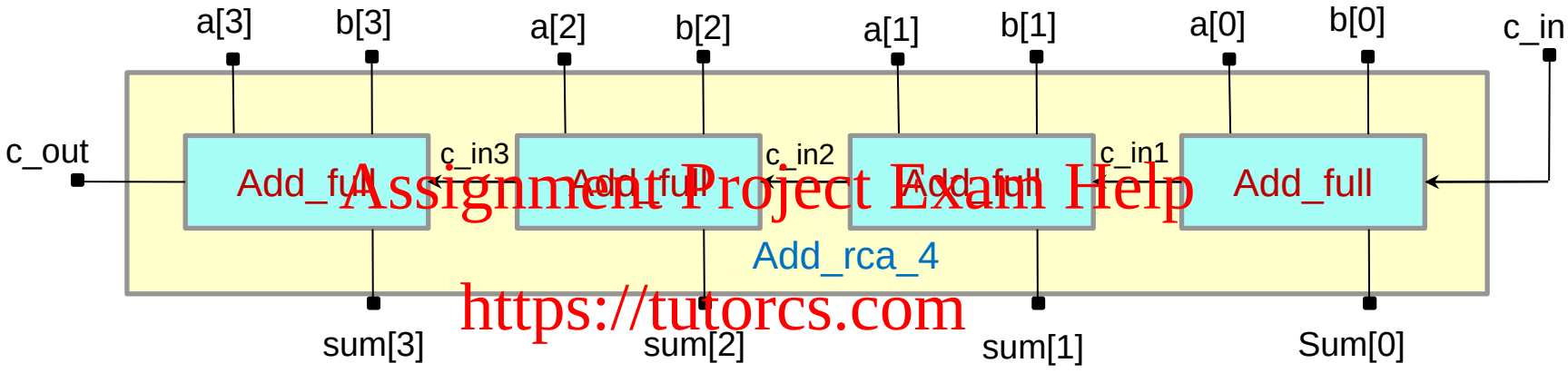
Design Hierarchy
Example: Ripple Carry Adder

A 16-bit ripple-carry adder can be formed by cascading four 4-bit ripple-carry adders.

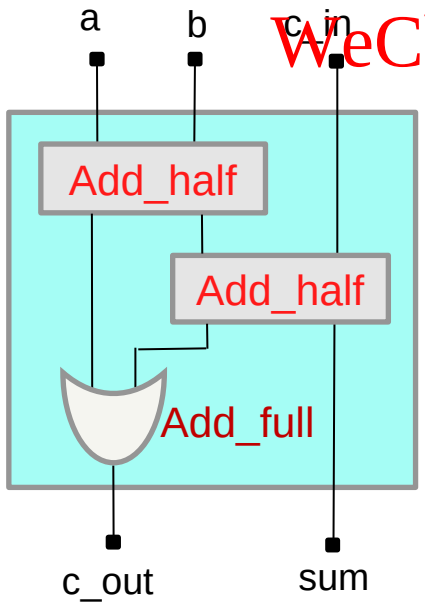


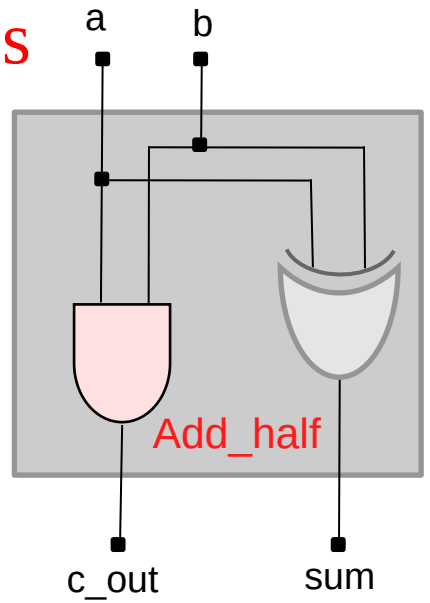
Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Design Hierarchy



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs





Structural Verilog Code (1)

```
 1  module Add_rca_16 (sum, c_out, a, b, c_in);
 2      output[15:0]    sum;
 3      output          c_out;
 4      input[15:0]     a, b;
 5      input           c_in;
 6      wire            c_in4, c_in8, c_in12;
 7
 8      Add_rca_4   M1  (sum[3:0],   c_in4,  a[3:0],   b[3:0],   c_in);
 9      Add_rca_4   M2  (sum[7:4],   c_in8,  a[7:4],   b[7:4],   c_in4);
10      Add_rca_4   M3  (sum[11:8],  c_in12, a[11:8],  b[11:8],  c_in8);
11      Add_rca_4   M4  (sum[15:12], c_out,  a[15:12], b[15:12], c_in12);
12  endmodule
13
14  module Add_rca_4 (sum, c_out, a, b, c_in);
15      output[3:0]     sum;
16      output          c_out;
17      input[3:0]      a, b;
18      input           c_in;
19      wire            c_in1, c_in2, c_in3;
20
21      Add_full    M1  (sum[0],  c_in1,  a[0],  b[0],  c_in);
22      Add_full    M2  (sum[1],  c_in2,  a[1],  b[1],  c_in1);
23      Add_full    M3  (sum[2],  c_in3,  a[2],  b[2],  c_in2);
24      Add_full    M4  (sum[3],  c_out,  a[3],  b[3],  c_in3);
25  endmodule
```

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

- Any identifier that is referenced without having a type declaration is by default of type wire (The pointed declarations can be removed).

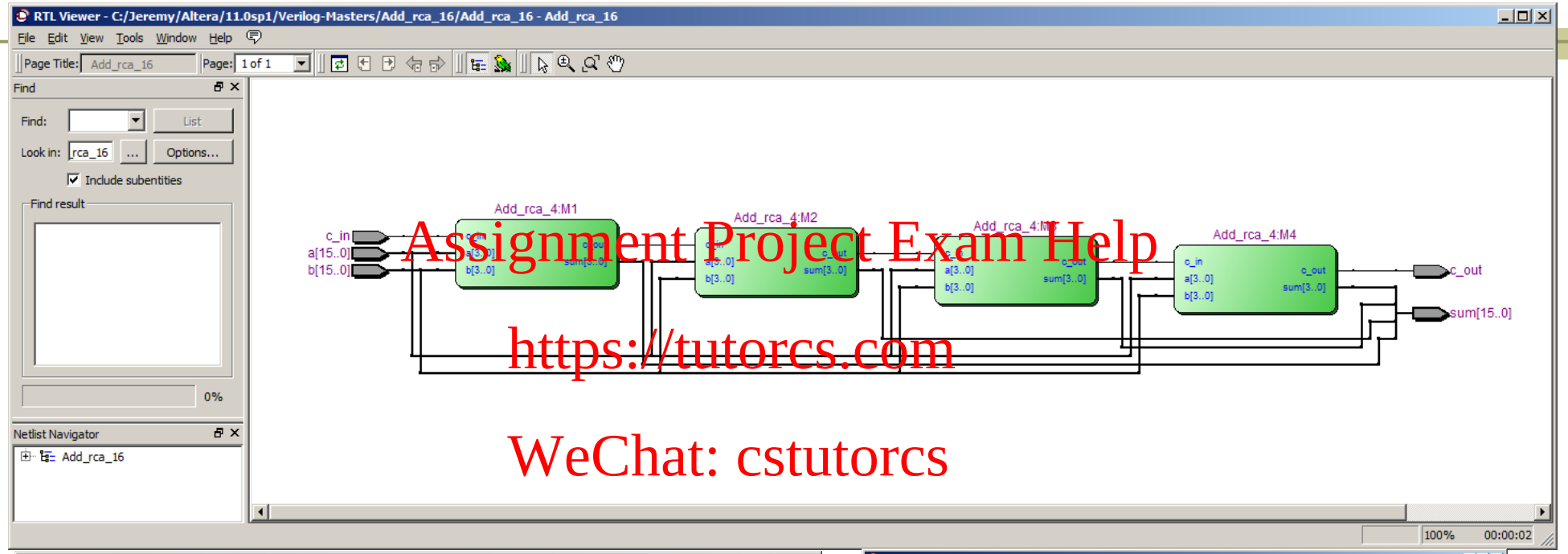
Structural Verilog Code (2)

```
26
27  module Add_full (sum, c_out, a, b, c_in);
28        output       sum, c_out;
29        input        a, b, c_in;
30        wire         w1, w2, w3;
31        Add_half     M1 (w1, w2, a , b);
32        Add_half     M2 (sum, w3, c_in , w1);
33        or           (c_out, w2, w3);
34  endmodule
35
36  module Add_half ( sum, c_out, a ,b);
37        output   c_out, sum;
38        input    a, b;
39        xor      (sum, a, b);
40        and      (c_out, a, b);
41  endmodule
42
```

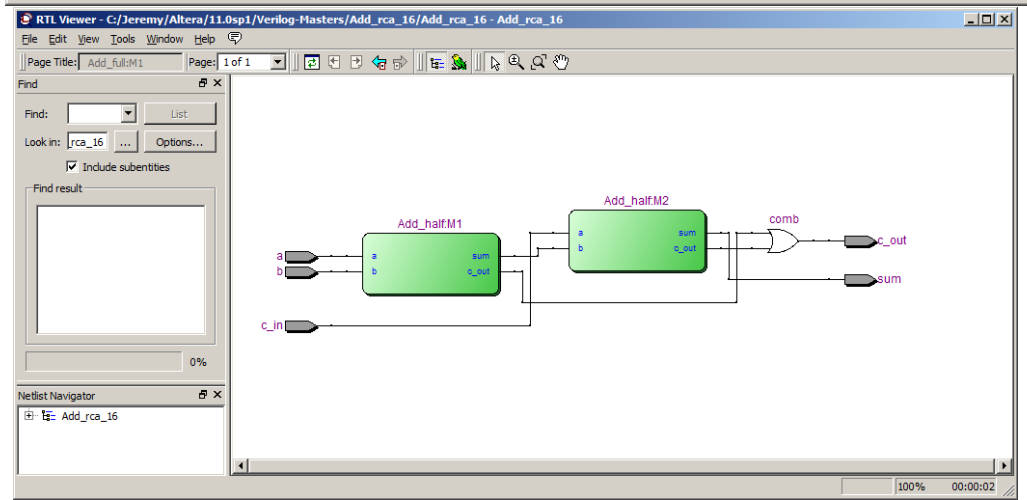
Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

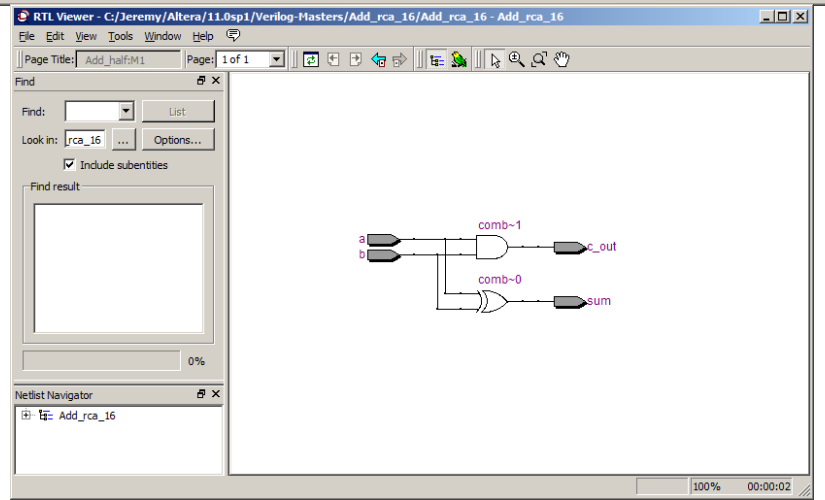
- A **vector** in Verilog is denoted by square brackets, e.g. Sum[7:0]
- The leftmost index in the bit range is the most significant bit, and the rightmost is the least significant bit.
- If Sum has a value of 4 then: Sum[3:0]=4 , Sum[2]=1, and Sum[5:1]=2
- **Wires** in Verilog establish connectivity between design objects.

Synthesised Design (RTL Viewer)

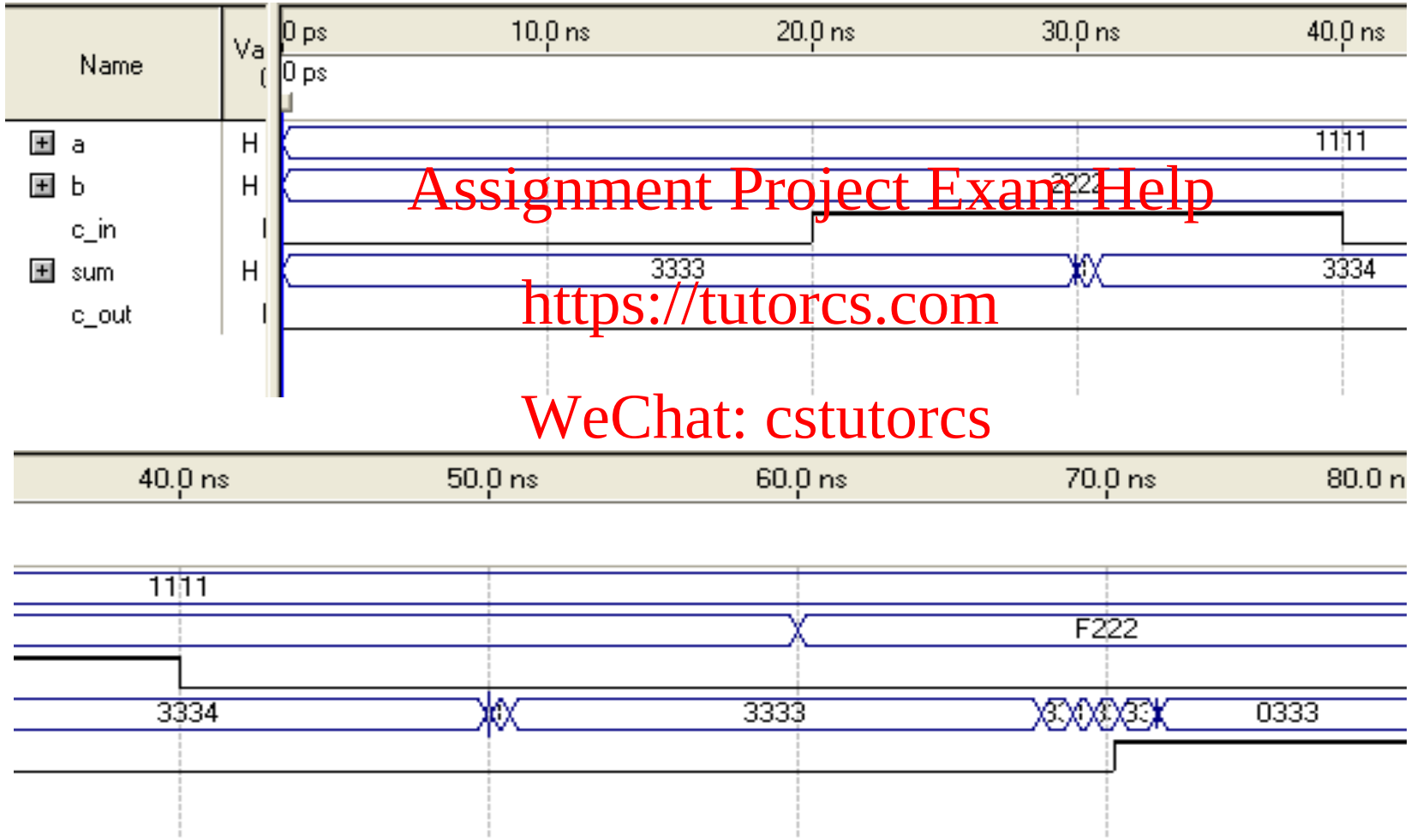


Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs





Simulation



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Associating Ports of Instantiated Modules

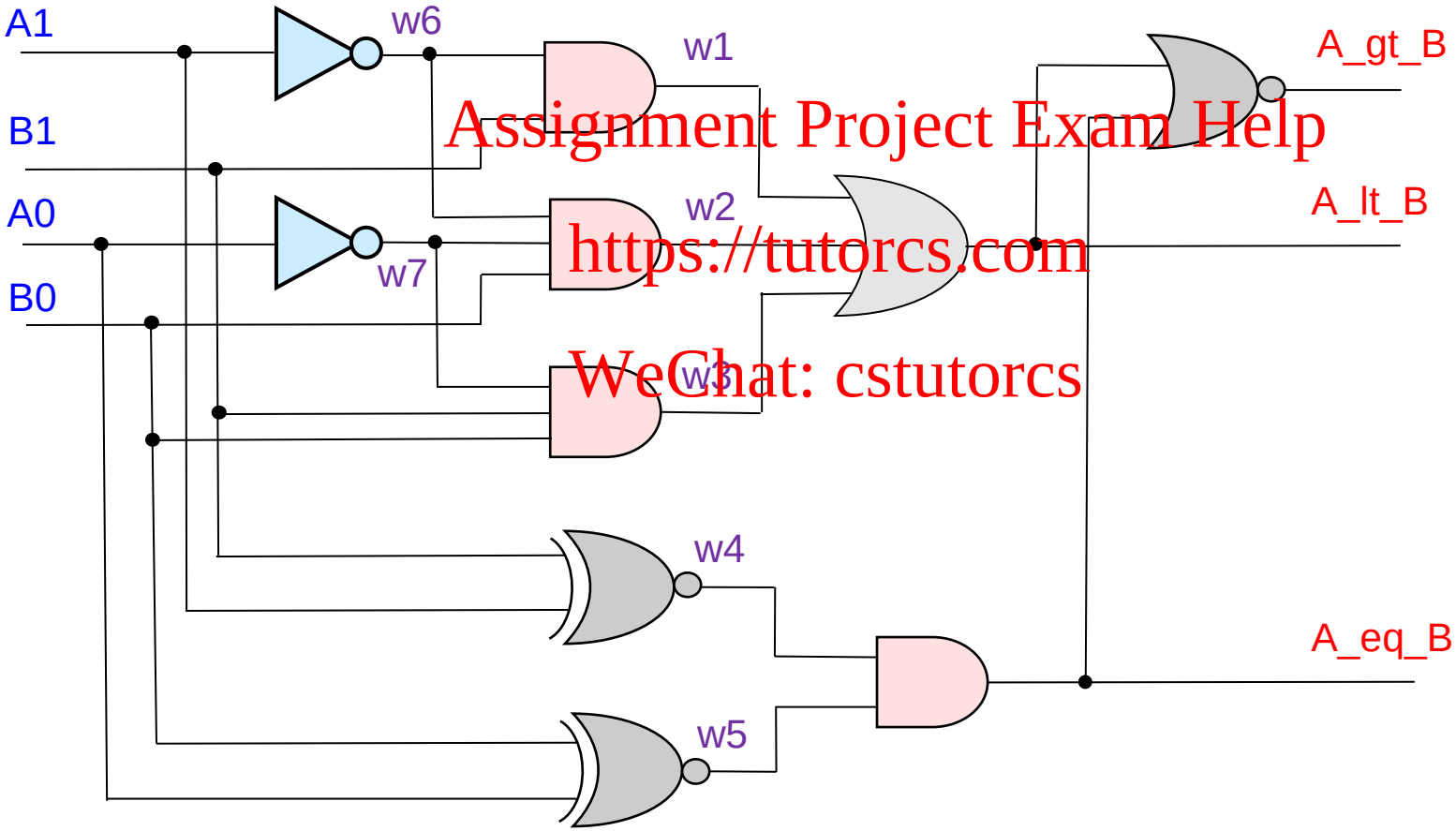
- By Position
  - The names of the connecting signals can be different but they must be consistent with the module declaration.
  - **module** Add_half (sum, c_out, a , b);
  - Add_half M1 (w1, w2, a , b);
- By Name
  - This method is useful for modules with many ports.
  - Syntax: *.formal_name (actual_name)*
  - Add_half M1 (.c_out (w2), .b(b), .a(a), .sum(w1));

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Module declaration

Module instantiation

Example: 2-bit Comparator



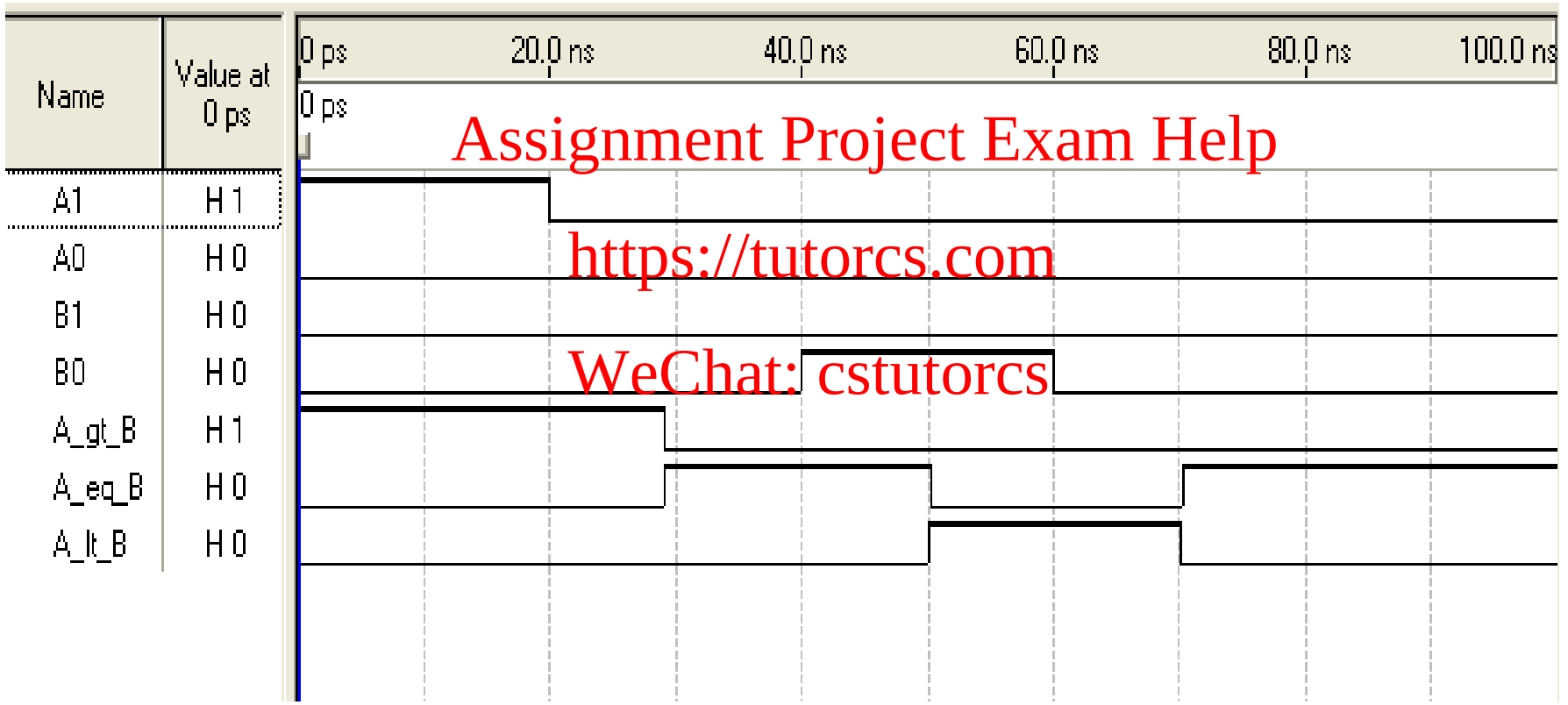
Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Structural Verilog Code

```
1  module Compare_2_str(A_gt_B, A_lt_B, A_eq_B, A0, A1,B0, B1);
2      output  A_gt_B, A_lt_B, A_eq_B ;
3      input   A0, A1,B0, B1 ;
4
5      nor     (A_gt_B, A_lt_B, A_eq_B);
6      or      (A_lt_B, w1, w2, w3);
7      and     (w1,w6,B1);
8      and     (w2,w6,w7,B0);
9      and     (w3,w7,B1,B0);
10     and     (A_eq_B,w4,w5);
11     not     (w6,A1);
12     not     (w7,A0);
13     xnor    (w4,A1,B1);
14     xnor    (w5,A0,B0);
15 endmodule
```

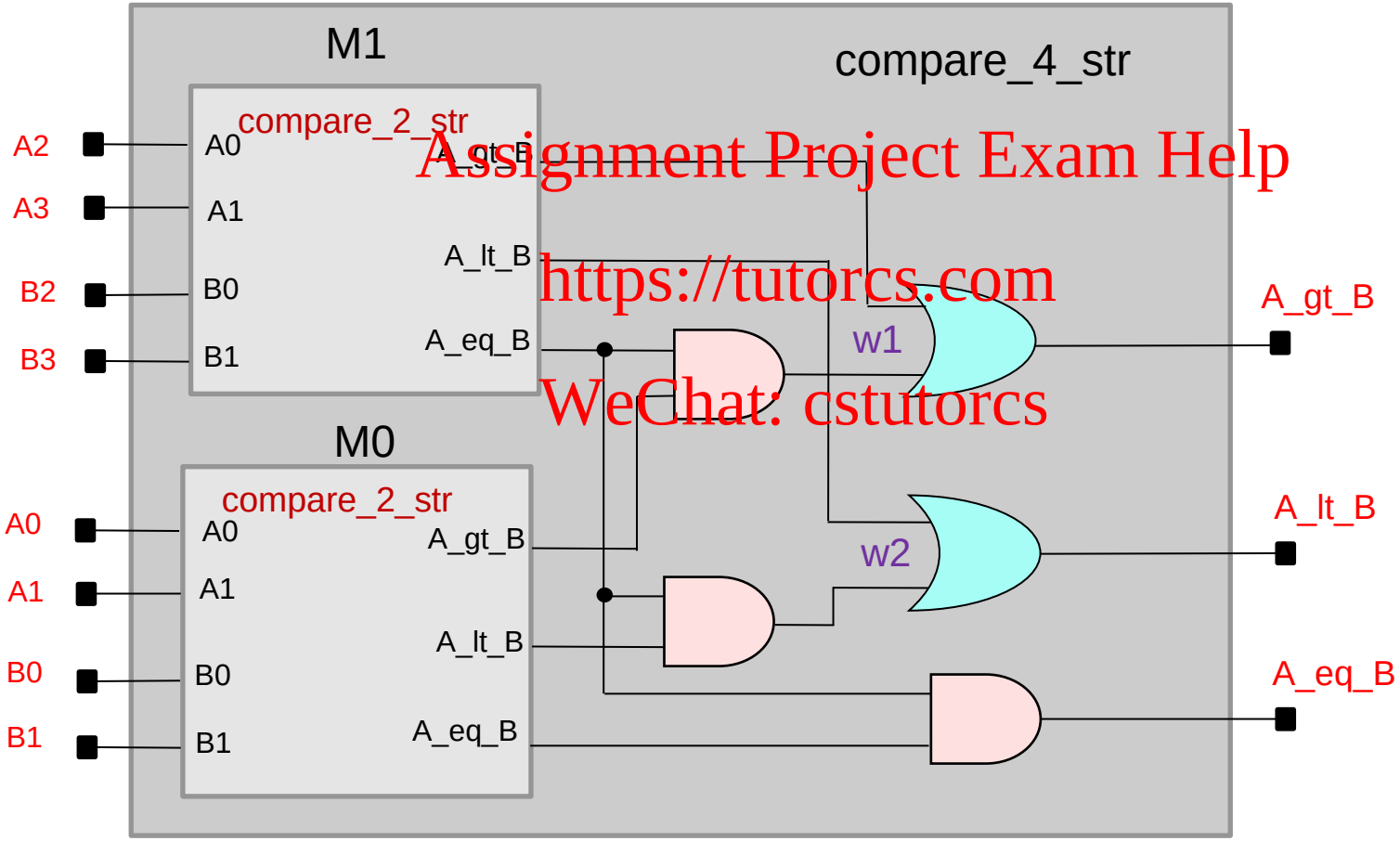
Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Timing Simulation



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

4-bit Comparator



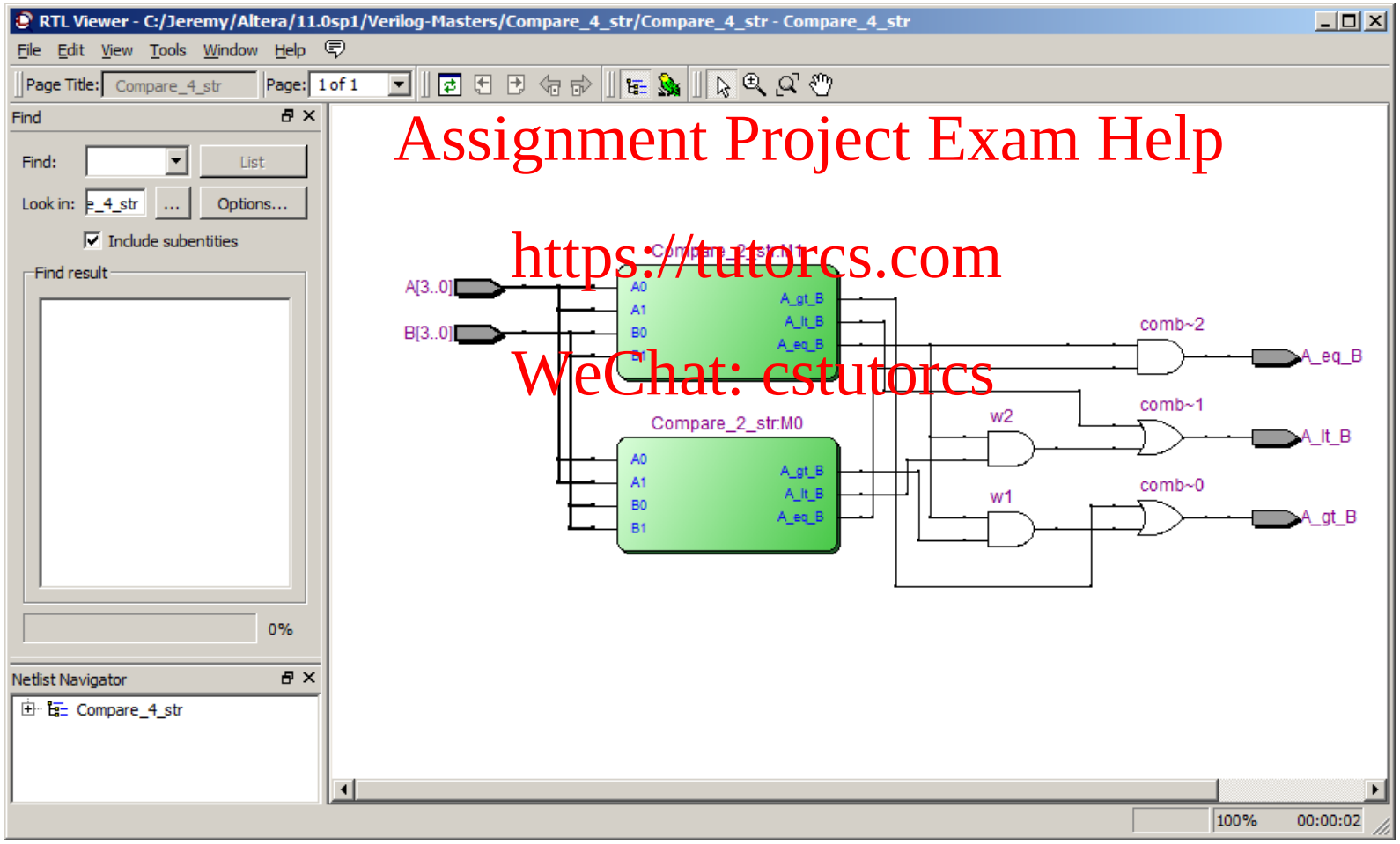
Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Verilog Code

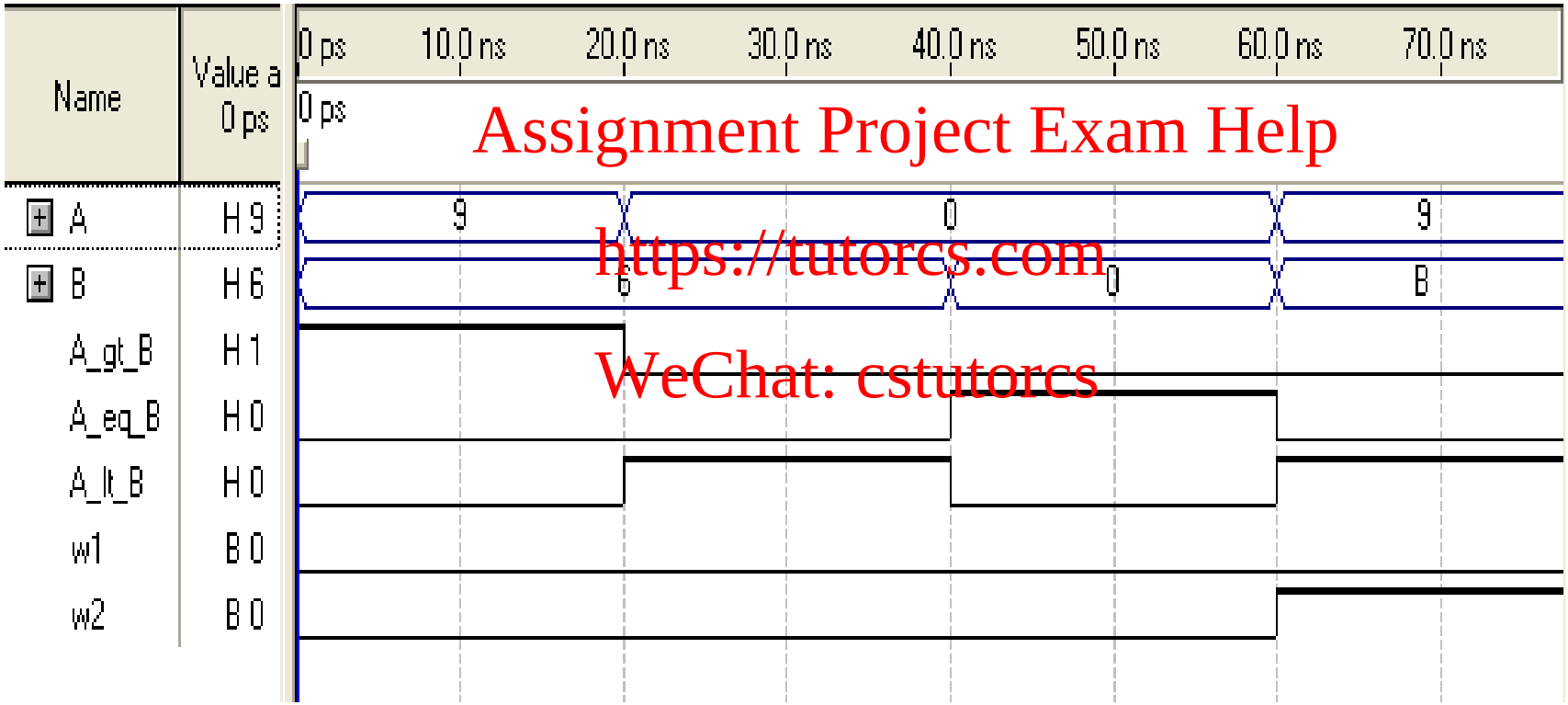
```
1  module Compare_4_str(A_gt_B, A_lt_B, A_eq_B, A,  B);
2      output  A_gt_B, A_lt_B, A_eq_B ;
3      input   [3:0]   A, B;
4
5      Compare_2_str M1 (A_gt_B_M1, A_lt_B_M1, A_eq_B_M1, A[2], A[3],B[2], B[3]);
6      Compare_2_str M0 (A_gt_B_M0, A_lt_B_M0, A_eq_B_M0, A[0], A[1],B[0], B[1]);
7
8      or      (A_gt_B, A_gt_B_M1, w1);
9      or      (A_lt_B, A_lt_B_M1, w2);
10     and     (A_eq_B, A_eq_B_M1, A_eq_B_M0);
11     and     (w1, A_eq_B_M1, A_gt_B_M0);
12     and     (w2, A_eq_B_M1, A_lt_B_M0);
13
14 endmodule
15
```

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Synthesised design



Functional Simulation



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs