Verilog

Coding in Xilinx ISE Design Suite v14.6

Prof. Dr. Mehmet Önder Efe
onderefe@cs.hacettepe.edu.tr
Welcome to the ISE® Design Suite

**Project commands**

- Open Project...
- Project Browser...
- New Project...
- Open Example...

**Recent projects**

Double click on a project in the list below to open

<table>
<thead>
<tr>
<th>Project Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbm231</td>
</tr>
<tr>
<td>onderede02</td>
</tr>
<tr>
<td>sireldevre</td>
</tr>
<tr>
<td>onderede03</td>
</tr>
</tbody>
</table>

**Additional resources**

- Tutorials on the Web
- Design Resources
- Application Notes
Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project

- **Name:** myproject
- **Location:** C:\AAA_ONDER\C_EFE\verlog_onederefe\myproject
- **Working Directory:** C:\AAA_ONDER\C_EFE\verlog_onederefe\myproject
- **Description:**

Select the type of top-level source for the project

- **Top-level source type:** HDL
### Project Settings

Specify device and project properties. Select the device and design flow for the project.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation Development Board</td>
<td>None Specified</td>
</tr>
<tr>
<td>Product Category</td>
<td>All</td>
</tr>
<tr>
<td>Family</td>
<td>Spartan3E</td>
</tr>
<tr>
<td>Device</td>
<td>XC3S500E</td>
</tr>
<tr>
<td>Package</td>
<td>FG320</td>
</tr>
<tr>
<td>Speed</td>
<td>-4</td>
</tr>
<tr>
<td>Top-Level Source Type</td>
<td>HDL</td>
</tr>
<tr>
<td>Synthesis Tool</td>
<td>XST (VHDL/Verilog)</td>
</tr>
<tr>
<td>Simulator</td>
<td>ISim (VHDL/Verilog)</td>
</tr>
<tr>
<td>Preferred Language</td>
<td>Verilog</td>
</tr>
<tr>
<td>Property Specification in Project File</td>
<td>Store all values</td>
</tr>
<tr>
<td>Manual Compile Order</td>
<td></td>
</tr>
<tr>
<td>VHDL Source Analysis Standard</td>
<td>VHDL.03</td>
</tr>
<tr>
<td>Enable Message Filtering</td>
<td></td>
</tr>
</tbody>
</table>

[Next]
New Project Wizard

Project Summary

Project Navigator will create a new project with the following specifications.

Project:
   Project Name: myproject
   Project Path: C:\AAA_ONDER\C_EFE\verilog_onderefe\myproject
   Working Directory: C:\AAA_ONDER\C_EFE\verilog_onderefe\myproject
   Description:
   Top Level Source Type: HDL

Device:
   Device Family: Spartan3E
   Device: xc3s500e
   Package: fg320
   Speed: -4
   Top-Level Source Type: HDL
   Synthesis Tool: XST (VHDL/Verilog)
   Simulator: ISim (VHDL/Verilog)
   Preferred Language: Verilog
   Property Specification in Project File: Store all values
   Manual Compile Order: false
   VHDL Source Analysis Standard: VHDL-93
   Message Filtering: disabled

Finish Cancel
Right Click and choose New Source
Choose this

Give a name

File name: mymodule01
Location: C:\AAA_ONDER\C_EFE\verilog_onderefe\myproject
You can define inputs and outputs of your module here. Press Next and you will do it later.
Summary

Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: C:\AAA\ORDER\C_EFE\verilog_under_efe\myproject
Source Type: Verilog Module
Source Name: mymodule01.v
Module name: mymodule01
Port Definitions:
module mymodule01;

endmodule
module mymodule01();

endmodule
mymodule01 : 2 to 4 decoder

module mymodule01(D, A, B, e);

    output [0:3] D;
    input  A, B, e;
    wire A_n, B_n;
    not G1(A_n, A);
    not G2 (B_n, B);
    and G3(D[0], e, A_n, B_n);
    and G4(D[1], e, A_n, B);
    and G5(D[2], e, A, B_n);
    and G6(D[3], e, A, B);

endmodule
module mymodule01(D, A, B, e);

  output [0:3] D;
  input A, B, e;
  wire A_n, B_n;
  not G1(A_n, A);
  not G2(B_n, B);
  and G3(D[0], e, A_n, B_n);
  and G4(D[1], e, A_n, B);
  and G5(D[2], e, A, B_n);
  and G6(D[3], e, A, B);

endmodule
Our decoder code is ready. Save it.

\[ D_0 = eA'B' \]
\[ D_1 = eA'B \]
\[ D_2 = eAB' \]
\[ D_3 = eAB \]
Now we will write a testbench

Apply A, B, e, obtain $D_0, D_1, D_2, D_3$

$D_0 = eA'B'$
$D_1 = eA'B$
$D_2 = eAB'$
$D_3 = eAB$
Right Click and choose New Source
Choose Verilog Test Fixture

File name: mymodule01_tb
Location: C:\AAA_ONDER\C_EFE\verilog_onderefe\myproject

mymodule01_tb
This testbench is related only with mymodule01.v and it is chosen. If the project had more than one file, we would choose the relevant ones here.
Summary

Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: C:\AAA_QNDER\C_EFE\verilog_onderefe\myproject
Source Type: Verilog Test Fixture
Source Name: mymodule01_tb.v

Association: mymodule01
module mymodule01_tb;

// Inputs
reg A;
reg B;
reg e;

// Outputs
wire [0:3] D;

// Instantiate the Unit Under Test (UUT)
mymodule01 uut (  
  .D(D),  
  .A(A),  
  .B(B),  
  .C(e)  
);

initial begin
  // Initialize Inputs
  A = 0;
  B = 0;
  e = 0;

  // Wait 100 ns for global reset to finish
  #100;

  // Add stimulus here
end

endmodule
module mymodule01_tb;

    // Inputs
    reg A;
    reg B;
    reg e;

    // Outputs
    wire [0:3] D;

    // Instantiate the Unit Under Test (UUT)
    mymodule01 uut (  
        .D(D),
        .A(A),
        .B(B),
        .e(e)
    );

    initial begin
        // Initialize Inputs
        A = 0;
        B = 0;
        e = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
    end

endmodule
module mymodule01_tb;

    // Inputs
    reg A;
    reg B;
    reg e;

    // Outputs
    wire [0:3] D;

    // Instantiate the Unit Under Test (UUT)
    mymodule01 uut (    
          .D(D),
          .A(A),
          .B(B),
          .e(e)       
    );

    initial begin
        // Initialize Inputs
        A = 0;
        B = 0;
        e = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
    end

endmodule
initial begin
  // Initialize Inputs
  A = 0;
  B = 0;
  e = 0;
  // Wait 100 ns for global reset to finish
  #100;
  // Add stimulus here
  // ENTER YOUR TESTING CONDITIONS HERE
end
endmodule
// Add stimulus here
A = 0;
B = 0;
e = 1;
#20;

A = 0;
B = 1;
e = 1;
#40;

A = 1;
B = 0;
e = 1;
#60;

A = 1;
B = 1;
e = 1;
#80;
Single click to choose

See ISim simulator here
```verilog
// Initialize Inputs
A = 0;
B = 0;
e = 0;

// Wait 100 ns for global reset to finish
#100;

// Add stimulus here
A = 0;
B = 0;
e = 1;
#20;

A = 0;
e = 1;
#40;

A = 1;
B = 0;
e = 1;
#60;

A = 1;
B = 1;
e = 1;
#80;

end

endmodule
```
Double click “Simulate Behavioral Model”
```Verilog
// Initialize Inputs
A = 0;
B = 0;
C = 0;

// Wait 100 ns for global reset to finish
#100;

// Add stimulus here
A = 0;
B = 0;
C = 1;
#20;

A = 1;
B = 1;
C = 1;
#40;
A = 1;
B = 1;
C = 1;
#60;
A = 1;
B = 1;
C = 0;
#80;

end

endmodule
```
What is next?

• After the processing is finished, Isim window will appear automatically
Pull this to the very left. Use Ctrl+mouse scroll to adjust the time axis zoom to see something meaningful.
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full IISim license.
WARNING: IISim will run in Lite mode. Please refer to the IISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of IISim.
Time resolution is 1 ps.
Simulator is doing circuit initialization process.
Finished circuit initialization process.
issim>
Creating the Schematic - Synthesis

- We defined the logic system via a bunch of code lines. What is the circuit corresponding to our code?
- Can it be optimized according to the physical hardware in hand?
1. Click to choose mymodule01.v

2. Press to compile, wait...
module mymodule01(D, A, B, C, output [0:3] D);

// Create Date: 19:
// Tool versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File
// Additional Comments

module mymodule01 (mymodule01_tb.v)

mymodule01 (mymodule01)

01_tb (mymodule01_tb.v)
Select how the RTL/Tech Viewer behaves when it is initially invoked

Startup mode

- **Start with the Explorer Wizard**
  
  In this mode, the Explorer Wizard is the initial screen, and allows you to select the elements that you want to see on the initial schematic

- **Start with a schematic of the top-level block**
  
  In this mode, the Explorer Wizard is bypassed and an initial schematic is created with only the top-level block displayed. You can then use the logic expansion capabilities of the Viewer to start expanding from the top-level block

You can also change the startup mode by selecting Edit->Preferences under the RTL/Tech Viewer page

- Show this dialog on startup

[OK]
Create RTL Schematic

1) Select items you want on the schematic from the "Available Elements" list and move them to the "Selected Elements" list.
   - Use the Filter control to filter the "Available Elements" list by name.
2) Press the "Create Schematic" button to generate a schematic view using the...
Create RTL Schematic

1) Select items you want on the schematic from the "Available Elements" list and move them to the "Selected Elements" list.
   - Use the Filter control to filter the "Available Elements" list by name
2) Press the "Create Schematic" button to generate a schematic view using the

Available Elements
- Primitives
- Signals
- Top Level Ports

Selected Elements
**RTL View**
Viewing an RTL schematic opens an NGR file that can be viewed as a gate-level schematic.
This schematic is generated after the HDL synthesis phase of the synthesis process. It shows a representation of the pre-optimized design in terms of generic symbols, such as adders, multipliers, counters, AND gates, and OR gates, that are independent of the targeted Xilinx device.

**Technology View**
Viewing a Technology schematic opens an NGC file that can be viewed as an architecture-specific schematic.
This schematic is generated after the optimization and technology targeting phase of the synthesis process. It shows a representation of the design in terms of logic elements optimized to the target Xilinx device or "technology"; for example, in terms of LUTs, carry logic, I/O buffers, and other technology-specific components. Viewing this schematic allows you to see a technology-level representation of your HDL optimized for a specific Xilinx architecture, which might help you discover design issues early in the design process.
You should always refer to technology schematic for synthesized result.
Select how the RTL/Tech Viewer behaves when it is initially invoked

Startup mode

- Start with the Explorer Wizard
  In this mode, the Explorer Wizard is the initial screen, and allows you to select the elements that you want to see on the initial schematic

- Start with a schematic of the top-level block
  In this mode, the Explorer Wizard is bypassed and an initial schematic is created with only the top-level block displayed. You can then use the logic expansion capabilities of the Viewer to start expanding from the top-level block

You can also change the startup mode by selecting Edit->Preferences under the RTL/Tech Viewer page

- Show this dialog on startup

OK
Create Technology Schematic

1) Select items you want on the schematic from the "Available Elements" list and move them to the "Selected Elements" list.
   - Use the Filter control to filter the "Available Elements" list by name.
2) Press the "Create Schematic" button to generate a schematic view using the items in the "Selected Elements" list.
0 = (10 * !11 * !12);
<table>
<thead>
<tr>
<th>D2</th>
<th>D1</th>
<th>I0</th>
<th>I1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
LUT3_02

Truth Table

<table>
<thead>
<tr>
<th>I0</th>
<th>I1</th>
<th>I2</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
module mux1( select, d, q );
    input [1:0] select;
    input [3:0] d;
    output q;
    wire q;
    wire [1:0] select;
    wire [3:0] d;
    assign q = d[select];
endmodule
module mux1_tb;
  // Inputs
  reg [1:0] select;
  reg [3:0] d;
  // Outputs
  wire q;
  integer i;
  // Instantiate the Unit Under Test (UUT)
  mux1 uut (  
    .select(select),
    .d(d),
    .q(q)
  );
MUX Testbench 2/4

```verilog
initial
begin
#1 $monitor("d = %b", d, " | select = ", select, " | q = ", q );
for( i = 0; i <= 15; i = i + 1)
begin
    d = i;
    select = 0; #1;
    select = 1; #1;
    select = 2; #1;
    select = 3; #1;
    $display("------------------------------------");
end
endmodule
```
<table>
<thead>
<tr>
<th>d</th>
<th>select</th>
<th>q</th>
<th>d</th>
<th>select</th>
<th>q</th>
<th>d</th>
<th>select</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>0011</td>
<td>0</td>
<td>0</td>
<td>1001</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0000</td>
<td>1</td>
<td>0</td>
<td>0011</td>
<td>1</td>
<td>1</td>
<td>1001</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0000</td>
<td>2</td>
<td>0</td>
<td>0011</td>
<td>1</td>
<td>1</td>
<td>1001</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0000</td>
<td>3</td>
<td>0</td>
<td>0011</td>
<td>1</td>
<td>1</td>
<td>1001</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>0</td>
<td>1</td>
<td>0011</td>
<td>1</td>
<td>1</td>
<td>1001</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>0</td>
<td>0011</td>
<td>1</td>
<td>1</td>
<td>1001</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>2</td>
<td>0</td>
<td>0011</td>
<td>1</td>
<td>1</td>
<td>1001</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>3</td>
<td>0</td>
<td>0011</td>
<td>1</td>
<td>1</td>
<td>1001</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0010</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0010</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>0</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0010</td>
<td>3</td>
<td>0</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0011</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0011</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0011</td>
<td>2</td>
<td>0</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>0</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0100</td>
<td>0</td>
<td>0</td>
<td>1001</td>
<td>0</td>
<td>1</td>
<td>1001</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0100</td>
<td>1</td>
<td>0</td>
<td>1001</td>
<td>0</td>
<td>1</td>
<td>1001</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0100</td>
<td>2</td>
<td>1</td>
<td>1001</td>
<td>0</td>
<td>1</td>
<td>1001</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0100</td>
<td>3</td>
<td>0</td>
<td>1001</td>
<td>0</td>
<td>1</td>
<td>1001</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0101</td>
<td>0</td>
<td>1</td>
<td>1001</td>
<td>0</td>
<td>1</td>
<td>1001</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0101</td>
<td>1</td>
<td>0</td>
<td>1001</td>
<td>0</td>
<td>1</td>
<td>1001</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

MUX Testbench 3/4
MUX Testbench 4/4
MUX using *always* block

```verilog
module mux2( select, d, q );
    input[1:0] select;
    input[3:0] d;
    output q;
    reg q;
    wire[1:0] select;
    wire[3:0] d;
    always @(d or select)
        q = d[select];
endmodule
```
MUX using if statements

```verilog
module mux3 (select, d, q);
    input[1:0] select;
    input[3:0] d;
    output q;
    reg q;
    wire[1:0] select;
    wire[3:0] d;
    always @(select or d)
    begin
        if (select == 0)
            q = d[0];
        if (select == 1)
            q = d[1];
        if (select == 2)
            q = d[2];
        if (select == 3)
            q = d[3];
    end
endmodule
```
MUX using `case` statements

```verilog
module mux4( select, d, q );
    input[1:0] select;
    input[3:0] d;
    output q;
    reg q;
    wire[1:0] select;
    wire[3:0] d;
    always @( select or d )
    begin
        case( select )
            0 : q = d[0];
            1 : q = d[1];
            2 : q = d[2];
            3 : q = d[3];
        endcase
    end
endmodule
```
module mux5( select, d, q );
    input [1:0] select;
    input [3:0] d;
    output q;
    wire q;
    wire [1:0] select;
    wire [3:0] d;
    assign q = ( select == 0 )? d[0] : ( select == 1 )? d[1] : ( select == 2 )? d[2] : d[3];
endmodule
module mux6( select, d, q );
  input[1:0] select;
  input[3:0] d;
  output q;
  reg q;
  wire[1:0] select;
  wire[3:0] d;
  always @( select or d )
  begin
    q = ( ~select[0] & ~select[1] & d[0] )
      | ( select[0] & select[1] & d[3] );
  end
endmodule
MUX using gates

module mux7( select, d, q );
    input[1:0] select;
    input[3:0] d;
    output q;
    wire q, q1, q2, q3, q4, NOTselect0, NOTselect1;
    wire[1:0] select;
    wire[3:0] d;
    not n1( NOTselect0, select[0] );
    not n2( NOTselect1, select[1] );
    and a1( q1, NOTselect0, NOTselect1, d[0] );
    and a2( q2, select[0], NOTselect1, d[1] );
    and a3( q3, NOTselect0, select[1], d[2] );
    and a4( q4, select[0], select[1], d[3] );
    or o1( q, q1, q2, q3, q4 );
endmodule
DEMUX

module demux1(select,d,q);

input d;
input [2:0] select;
output [7:0] q;
assign q[0] = d & ~select[2] & ~select[1] & ~select[0];
endmodule
module demux1_tb;
    // Inputs
    reg [2:0] select;
    reg d;
    // Outputs
    wire [7:0] q;
    integer i;
    // Instantiate the Unit Under Test (UUT)
    demux1 uut ( 
        .select(select),
        .d(d),
        .q(q)
    );

initial begin
    // Initialize Inputs
    select = 3'b000;
    d = 0;

    // Wait 100 ns for global reset to finish
    #10;
end
DEMUX Testbench 2/3
Choose left code or right code
Are they the same?

```
d = 1; select = 3'b000; #10;
d = 1; select = 3'b001; #10;
d = 1; select = 3'b010; #10;
d = 1; select = 3'b011; #10;
d = 1; select = 3'b100; #10;
d = 1; select = 3'b101; #10;
d = 1; select = 3'b110; #10;
d = 1; select = 3'b111; #10;
d = 0; select = 3'b000; #10;
end
endmodule
```

```
for(i = 0; i <= 7; i = i + 1)
begin
    select = i;
    d = 1;
    #10;
    d = 0;
    #10;
end
end
endmodule
```
Few Notes

• Initial block declares a single-pass behavior
  – Executes once when simulator is activated
• Delay control operator (#) and delay value - #10
• Timescale compiler directive
  – timescale <reference_time_unit>/<time_precision>
  – `timescale 1ns/1ps
• Inputs are declared as reg values – retains value until updated
• Outputs are just monitored as wires
Sequential Logic Implementations

• Generating clock signal
• Single D Flip Flop
• Three D Flip flops in series
• Parallel load shift register
Generating Clock Signal

always
    begin
        clk = 0;
        #5;
        clk = 1;
        #5;
    end

initial begin
    clk = 0;
end

always begin
    #5; clk = 0;
    #5; clk = 1;
end
module single_d_ff(clk,D,Q);
  input clk;
  input D;
  output reg Q;
  always @(posedge clk)
  begin
    Q <= D;
  end
endmodule
module single_d_ff_tb;
    // Inputs
    reg clk;
    reg D;
    // Outputs
    wire Q;
    // Instantiate the Unit Under Test (UUT)
    single_d_ff uut (  
        .clk(clk),  
        .D(D),  
        .Q(Q)  
    );
initial begin
  // Initialize Inputs
  clk = 0;
  D = 0;
  // Wait 10 ns for global reset to finish
  #10;
  // Add stimulus here
  D = 1; # 25;
  D = 0; # 25;
  D = 1; # 25;
  D = 0; # 25;
  D = 1; # 33;
  D = 0; # 33;
  D = 1; # 33;
  D = 0; # 33;
end
Single D Flip-flop Testbench 3/3

```verilog
always begin
    clk = 0 ; #13;
    clk = 1 ; #19;
end
endmodule
```
We have mux1.v, demux1.v, single_d_ff.v and so on. Whichever you would like to synthesize the circuitry, you need to designate it as the “top module” by clicking this.