



# Verilog HDL – I

黃元豪

Yuan-Hao Huang

國立清華大學電機工程學系

Department of Electrical Engineering

National Tsing-Hua University

# Outline

---



- Introduction
- Sample Design
- Structural Modeling
- RTL Modeling
- Logic Modeling and Simulation Using Xilinx ISE
- An Example of Combinational Circuits



# Introduction



# Verilog HDL

- Verilog 硬體描述語言 (Verilog Hardware Description Language)
  - 在積體電路設計（特別是超大型積體電路的計算機輔助設計）的電子設計自動化領域中，**Verilog HDL**是一種用於描述、設計電子系統（特別是數位電路的硬體描述語言）。
  - Verilog是電力電子工程師學會（IEEE）的1364號標準。
- Verilog 硬體描述語言在邏輯設計上的用途
  - 用途一: 邏輯電路設計的模擬與驗證
  - 用途二: FPGA邏輯電路(\*\*IC電路)的設計與實作
    - \*\* IC電路的設計與實作在【EE4292積體電路設計實驗】教授。



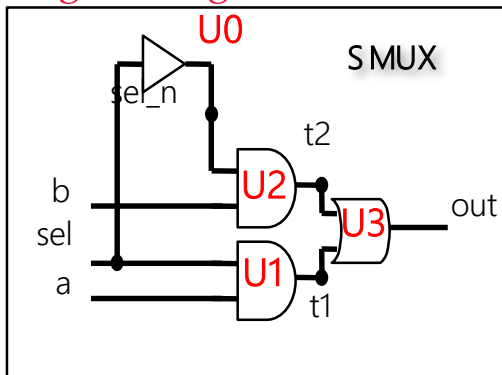
# Verilog HDL Utilization Scenario I

- Simulation and verification of logic circuits on PC.

## Specification



## Logic Design



## Verilog HDL Coding

```

module SMUX(out, a, b, sel);
    output out;
    input a,b,sel;
    wire sel_n,t1,t2;

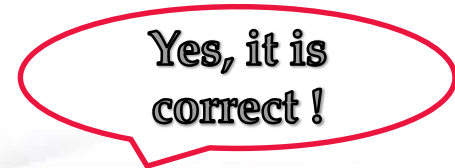
    not U0(sel_n,sel);
    and U1(t1,a,sel);
    and U2(t2,b,sel_n);
    or U3(out,t1,t2);

endmodule

```

## Test Pattern

a, b, sel
000
001
010
...
111



Verilog HDL Simulator

- Imaging what happens when the circuit is as complex as a CPU or MP3 player processor.



# Verilog HDL Utilization Scenario II

- Design and Implementation of logic circuits in FPGA (IC)

## Specification

2-to-1 Multiplexer

Verilog HDL Coding

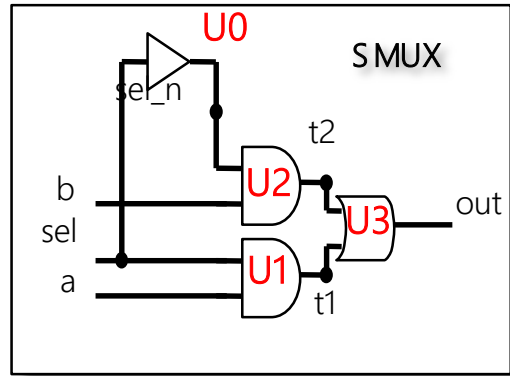
```

module SMUX(out, a, b, sel);
output out;
input a,b,sel;
wire sel_n,t1,t2;

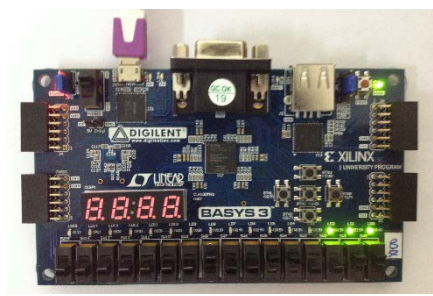
assign out = sel ? a:b;
endmodule

```

## Synthesized Logic Circuits



FPGA Implement Design and Programming



Now, it can work !



Verilog HDL Logic Synthesizer

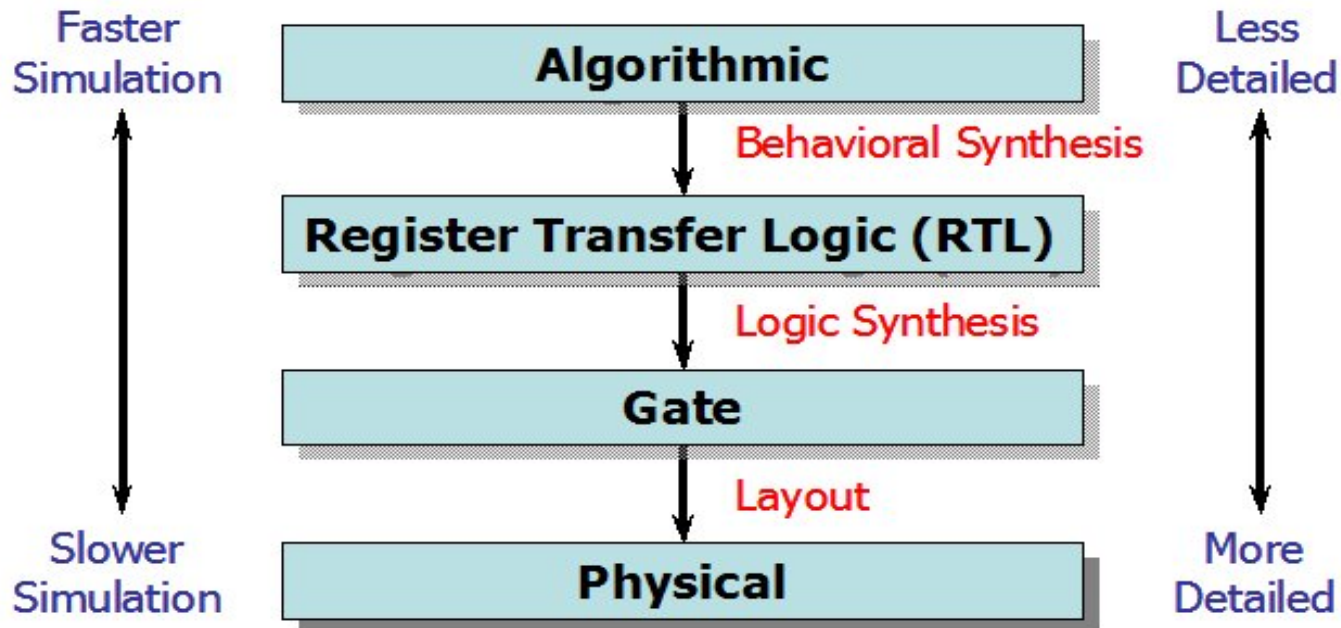


# Verilog HDL – Levels of Abstraction (2/2)



- **Behavioral Level (Architectural/Algorithmic Level)**
  - Describes a system by the flow of data between its functional blocks
  - Defines signal values when they change
- **Register Transfer Level (Dataflow)**
  - Describe a system by the flow of data and control signals between and within its functional blocks
  - Defines signal values with respect to a clock
  - RTL (Register Transfer Level) is frequently used for the Verilog description with the combination of behavioral and dataflow constructs which is acceptable to logic synthesis tools.
- **Gate Level (Structural)**
  - A model that describes the logic gates and the interconnections between them
- **Transistor/Switch/Physical Level**
  - A model that describes the transistors and the interconnections between them

# Verilog HDL – Levels of Abstraction (1/2)





# Level of Abstraction Example



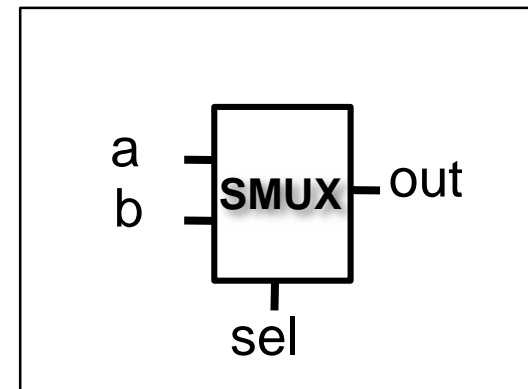
- Describe the operation of a circuit at various levels of abstraction (MP3 Player Decoder as example)
  - Behavior (MP3 Decoding, C/C++, HDL)
  - Function (Filtering, Fourier Transformation, C/C++, HDL)
  - Structure (Adder/Subtractor, Multiplier, Divider, HDL)
- Compared with C/C++ Program, verilog HDL
  - describes the timing (delay) of a circuit
  - expresses the concurrency (parallelism) of circuit operation

# Behavior Level Abstraction



- Describe the design without implying any specific internal architecture
  - Use high level constructs (@, case, if, repeat, wait, while)
  - Usually use behavioral construct in test-bench
  - Synthesis tools accept only a limited subset of behavior level description
    - Case 1 : assign Z=(S) ? A: B;

```
module SMUX(out, a, b, sel);  
  
output out;  
input a,b,sel;  
wire out;  
  
assign out = (sel) ? a : b ;  
  
endmodule
```

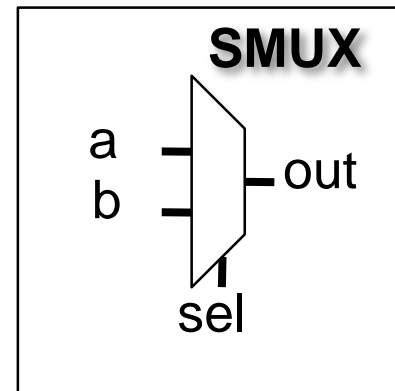


# Behavior Level Abstraction



- Case 2: always @ (input1 or input2 or ...)  
begin  
    out1=  
end

```
module SMUX(out, a, b, sel);  
  
output out;  
input a,b,sel;  
reg out;  
  
always @(a or b or sel)  
  if (sel)  
    out=a;  
  else  
    out=b;  
endmodule
```

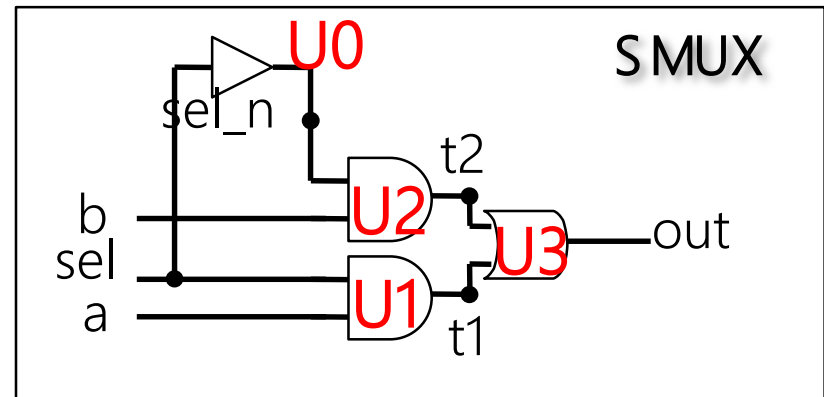




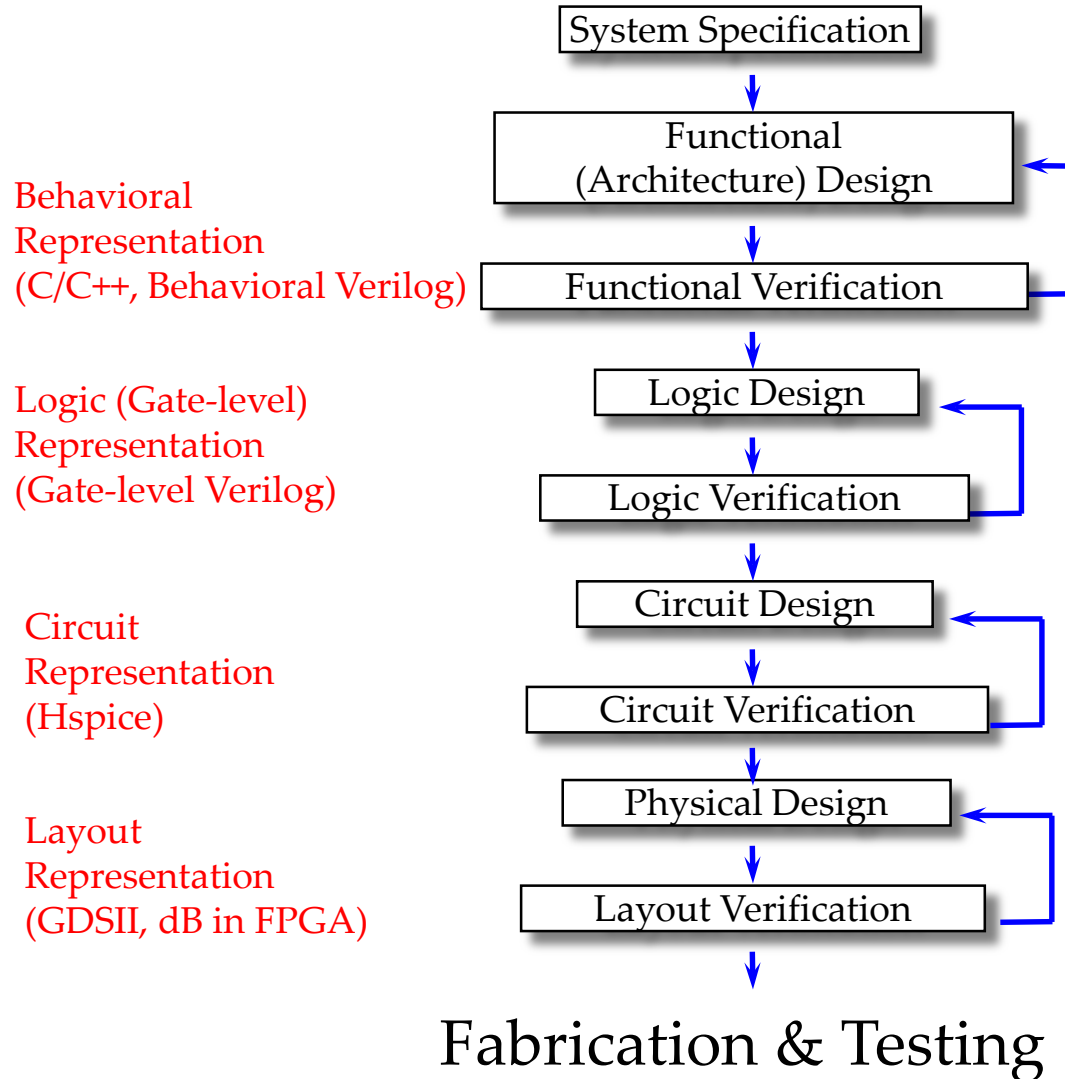
# Gate Level Abstraction

- Gate-level abstraction describes a pure structural design of circuits.
  - You must derive and draw the circuit schematics first before writing Verilog code.

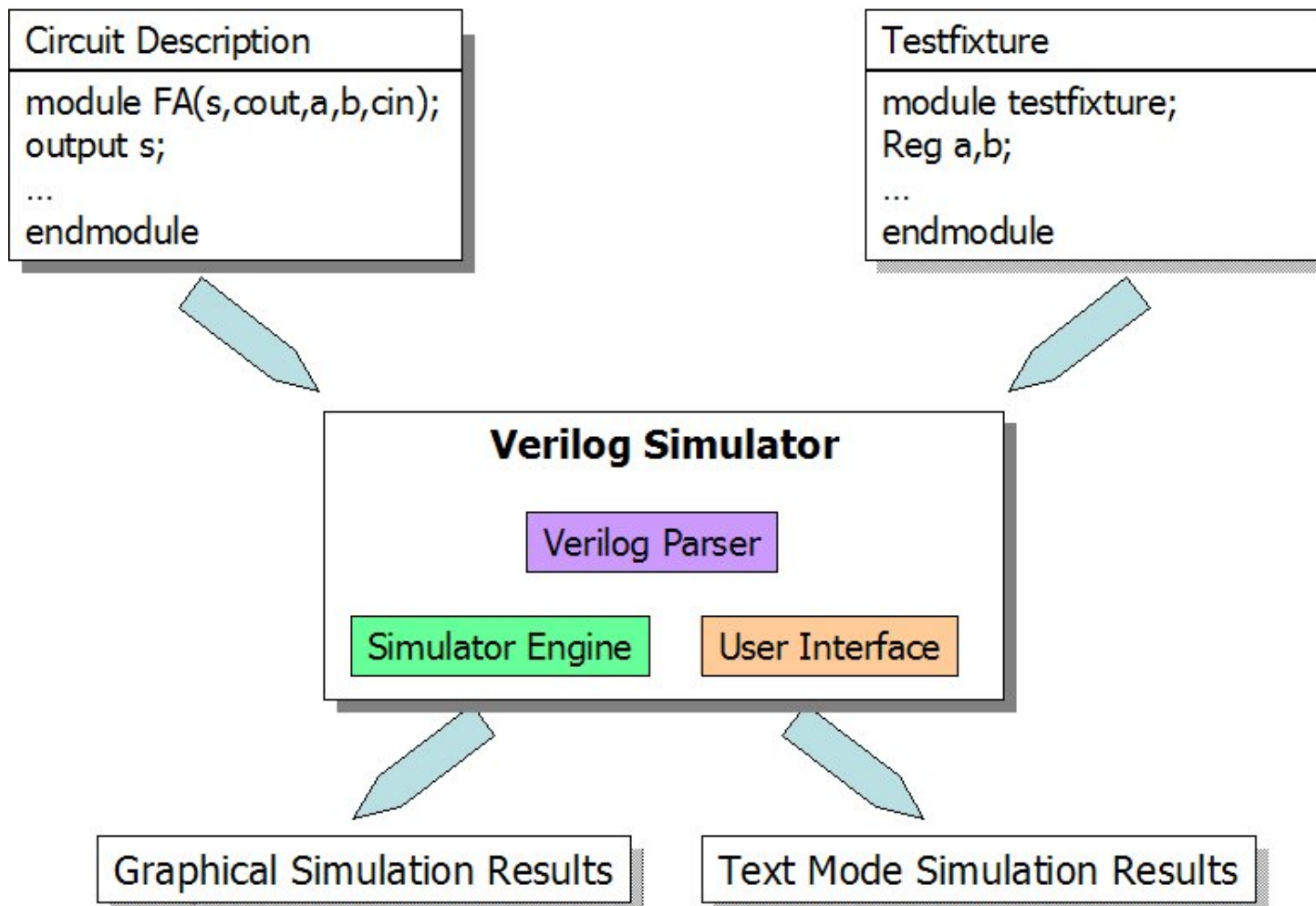
```
module SMUX(out, a, b, sel);  
  
output out;  
input a,b,sel;  
wire sel_n,t1,t2;  
  
not U0(sel_n,sel);  
and U1(t1,a,sel);  
and U2(t2,b,sel_n);  
or U3(out,t1,t2);  
  
endmodule
```



# VLSI Design Flow



# Verilog Simulation

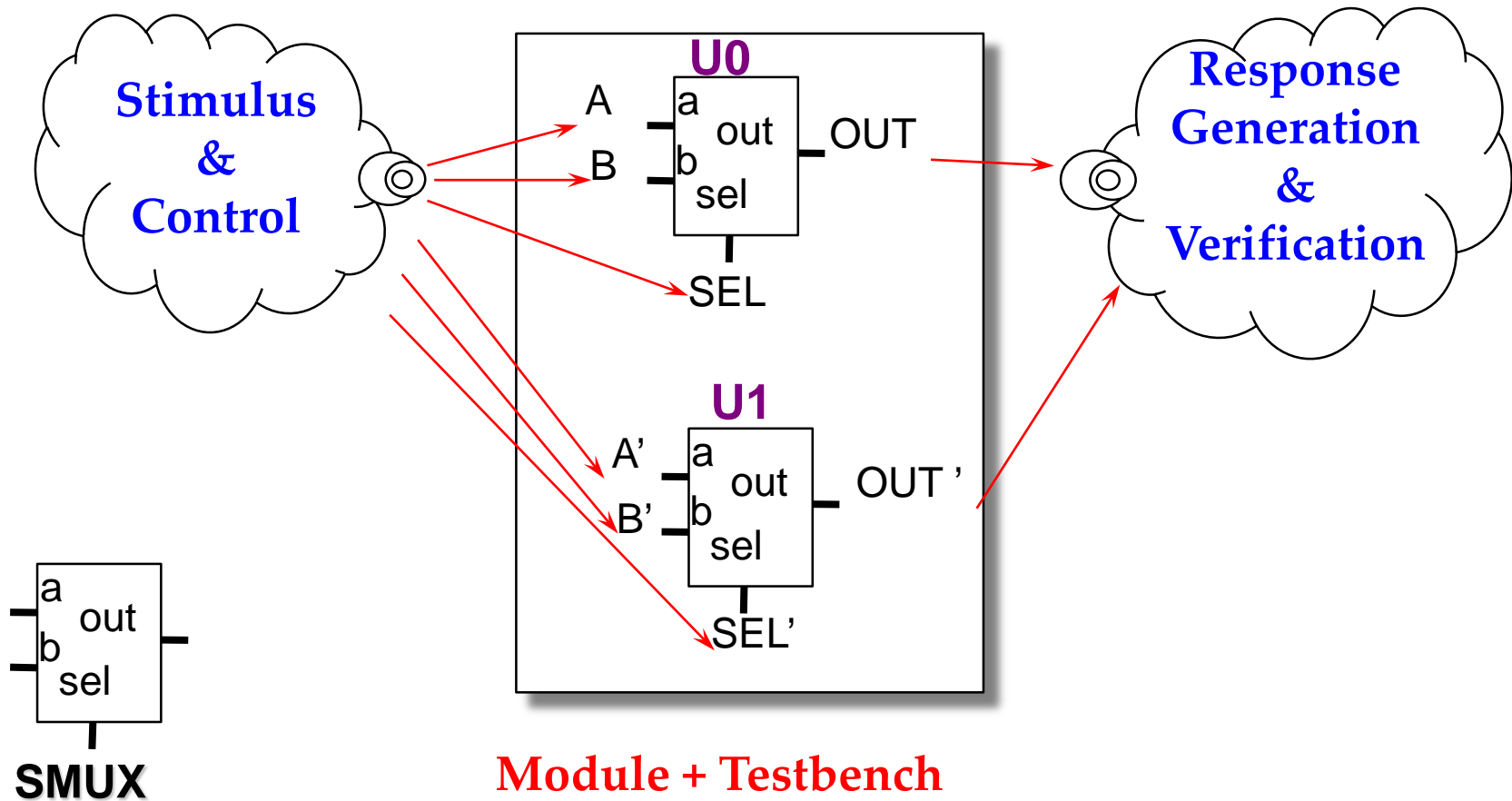




# Sample Design

# Scenario

Device under Test (DUT)







# Verilog Module

- `module module_name(port_names);`

- Port declaration

- Data type declaration

- Task & function declaration

- Module functionality or structure

- Timing Specification

- `endmodule`

```
module SMUX(out, a, b, sel);
```

```
output out;  
input a,b,sel;
```

```
wire sel_n,t1,t2;
```

```
not U0(sel_n,sel);  
and U1(t1,a,sel);  
and U2(t2,b,sel_n);  
or U3(out,t1,t2);
```

```
endmodule
```



# Testbench (1/4)

- module testfixture;
  - Declare signals
  - Instantiate modules
  - Applying stimulus
  - Monitor signals
- endmodule

# Testbench (2/4)

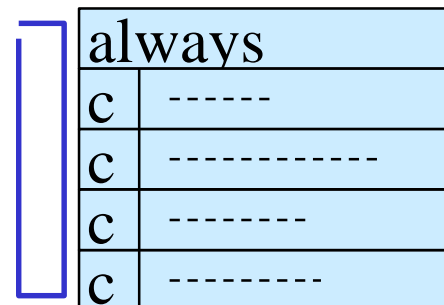
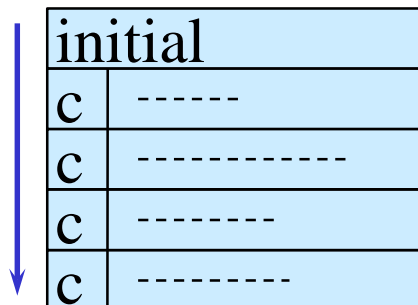


- Declare signals
  - Test pattern must be stored in storage elements first and then apply to DUT (Device under Test)
    - Use “reg” to declare the storage element
- Instantiate modules
  - Both behavioral level or gate level model can be used.



# Testbench (3/4)

- Describing Stimulus
  - The testbench always be described behaviorally.
  - Procedural blocks are bases of behavioral modeling.
  - The simulator starts executing all procedure blocks at time 0 and executes them concurrently.
  - Two types of procedural blocks
    - initial
    - always





# Testbench (4/4)

```
• module test_SMUX;  
• reg      A,B,SEL;  
• wire    OUT;  
• SMUX U0(.out(OUT),.a(A),.b(B),.sel(SEL));  
• initial  
• begin  
•     A=0;B=0;SEL=0;  
•     #10     A=0;B=1;SEL=1;  
•     #10     A=1;B=0;  
•     #10     SEL=0;  
•     .....  
•     #10     SEL=1;  
• end  
• endmodule
```

**Declare signals**

**Make an instance**

**Assign values to storage elements**

**#10 to specify 10 time unit delay**



# Structural Modeling

# Verilog Primitives

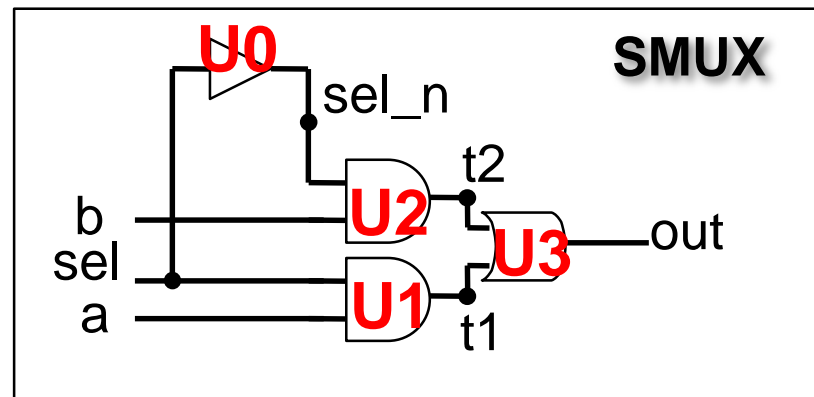


- and : Logical AND
- or : Logical OR
- not : Inverter
- buf : Buffer
- xor : Logical exclusive OR
- nand : Logical AND inverted
- nor : Logical OR inverted
- xnor : Logical exclusive OR inverted

# Structural Modeling



```
module SMUX(out, a, b, sel);  
  
output out;  
input a,b,sel;  
wire sel_n,t1,t2;  
  
not U0(sel_n,sel);  
and U1(t1,a,sel);  
and U2(t2,b,sel_n);  
or U3(out,t1,t2);  
  
endmodule
```







# RTL Modeling

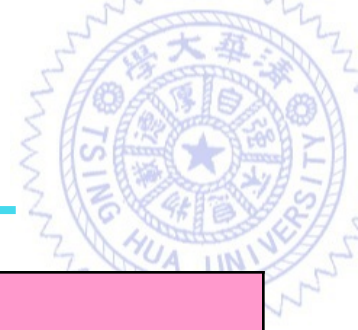


# Operators (1/3)

Bitwise Operators		
OP	Usage	Description
~	~m	Invert each bit of m
&	m & n	AND each bit of m with each bit of n
	m   n	OR each bit of m with each bit of n
^	m ^ n	Exclusive OR each bit of m with n
~^ or ^~	m ~^ n or m ^~ n	Exclusive NOR each bit of m with n

Unary Reduction Operators		
OP	Usage	Description
&	&m	AND all bits in m together (1-bit result)
~&	~&m	NAND all bits in m together (1-bit result)
	m	OR all bits in m together (1-bit result)
~	~ m	NOR all bits in m together (1-bit result)
^	^m	Exclusive OR all bits in m (1-bit result)
~^ or ^~	~^m or ^~m	Exclusive NOR all bits in m (1-bit result)

# Operators (2/3)



Arithmetic Operators		
OP	Usage	Description
+	$m + n$	Add $n$ to $m$
-	$m - n$	Subtract $n$ from $m$
-	$-m$	Negate $m$ (2's complement)
*	$m * n$	Multiply $m$ by $n$
/	$m / n$	* Divide $m$ by $n$
%	$m \% n$	* Modulus of $m / n$

\* **Synthesis not supported** : The divisor for divide operator may be restricted to constants and a power of 2

Logical Operators		
OP	Usage	Description
!	$!m$	Is $m$ not true? (1-bit True/False result)
&&	$m \&\& n$	Are both $m$ and $n$ true? (1-bit True/False result)
	$m \ \  n$	Are either $m$ or $n$ true? (1-bit True/False result)

Equality Operators (compares logic values of 0 and 1)		
OP	Usage	Description
==	$m == n$	Is $m$ equal to $n$ ? (1-bit True/False result)
!=	$m != n$	Is $m$ not equal to $n$ ? (1-bit True/False result)

Identity Operators (compares logic values of 0, 1, x, and z)		
OP	Usage	Description
===	$m === n$	* Is $m$ identical to $n$ ? (1-bit True/False result)
!==	$m !== n$	* Is $m$ not identical to $n$ ? (1-bit True/False result)

**Synthesis not supported**  
**Synthesis not supported**

# Operators (3/3)



Relational Operators		
OP	Usage	Description
<	$m < n$	Is m less than n? (1-bit True/False result)
>	$m > n$	Is m greater than n? (1-bit True/False result)
<=	$m <= n$	Is m less than or equal to n? (True/False result)
>=	$m >= n$	Is m greater than or equal to n? (True/False result)

Logical Shift Operators		
OP	Usage	Description
<<	$m << n$	Shift m left n-times
>>	$m >> n$	Shift m right n-times

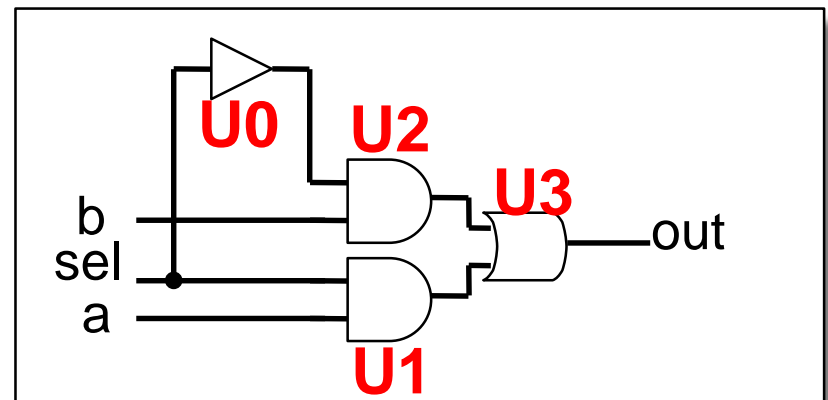
Misc Operators		
OP	Usage	Description
?:	$sel?m:n$	If sel is true, select m: else select n
{}	$\{m,n\}$	Concatenate m to n, creating larger vector
{}	$\{n\{m\}\}$	Replicate m n-times

# assign



- **assign** continuous construct
  - combinational logics

```
module SMUX (out,a,b,sel);  
output out;  
input a,b,sel;  
  
    assign out = (a&sel) | (b&(~sel));  
  
endmodule
```



This **out** has to be declared as “wire” or “output” data type.  
This expression can not be inside **always @()**.

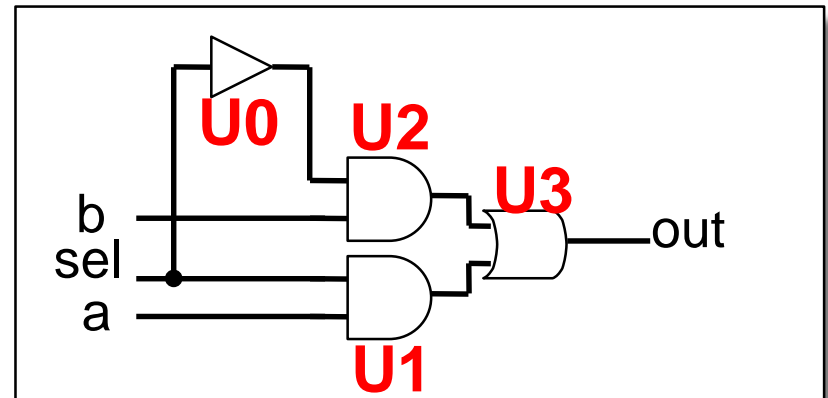
# always



- **always** statements

```
module SMUX (out,s,b,sel);  
output out;  
input a,b,sel;  
reg out;  
  
always @(a or b or sel)  
    out = (a&sel) | (b&(~sel));  
  
endmodule
```

sensitivity list



This **out** has to be declared as “reg” data type.

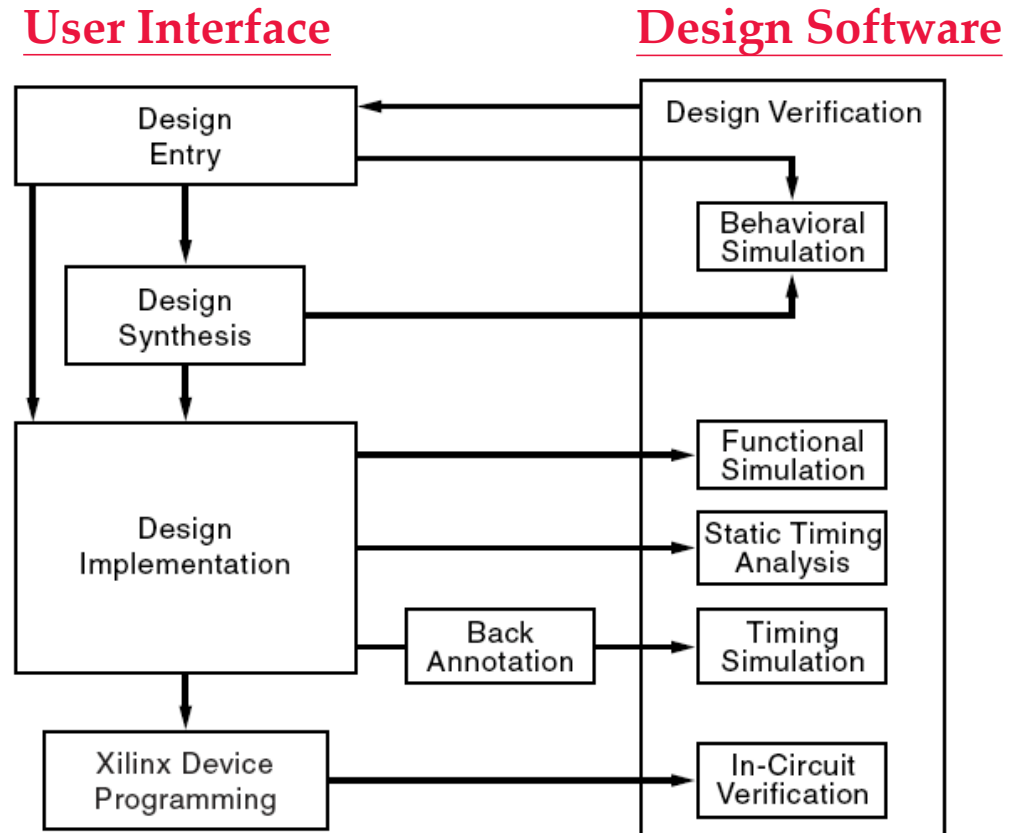


# Logic Modeling and Simulation Using Xilinx Vivado



# Design Flow

- General design flow
  - Design construction
  - Behavioral simulation
  - Design implementation
  - Timing simulation
- HDL-based design Flow



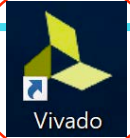


# Important Notes



- Draw schematic first and then construct Verilog codes.
- Verilog RTL coding philosophy is not the same as C programming
  - Every Verilog RTL construct has its own logic mapping (for synthesis)
  - You should have the logics (draw schematic) first and then the RTL codes
  - You have to write **synthesizable** RTL codes

# Open Vivado



A screenshot of the Vivado 2019.2 software interface. The window title is 'Vivado 2019.2'. The menu bar includes 'File', 'Flow', 'Tools', 'Window', and 'Help'. The main area is divided into three sections: 'Quick Start' (with options: Create Project, Open Project, Open Example Project), 'Tasks' (with options: Manage IP, Open Hardware Manager, Xilinx Tcl Store), and 'Learning Center' (with options: Documentation and Tutorials, Quick Take Videos, Release Notes Guide). On the right, there is a 'Recent Projects' panel listing: 'binary\_counter' (C:/Users/hp/LD/binary\_counter), 'nsw' (C:/Users/hp/LD/nsw), 'stopwatch' (C:/Users/hp/LD/stopwatch), 'shift\_reg' (C:/Users/hp/LD/shift\_reg), and 'ssd' (C:/Users/hp/LD/ssd). The bottom left corner shows a 'Tcl Console' tab.

# Open New Project (1/3)



Vivado 2019.2

File Flow Tools Window Help Q: Quick Access

**VIVADO**  
HLx Editions

**Double Click**

Quick Start

- Create Project >
- Open Project >
- Open Example Project >

Tasks

- Manage IP >
- Open Hardware Manager >
- Xilinx Tcl Store >

Learning Center

- Documentation and Tutorials >
- Quick Take Videos >
- Release Notes Guide >

Tcl Console

New Project

**Project Name**

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: smux

Project location: C:/Users/hp/LD

Create project subdirectory

Project will be created at: G:/Users/hp/LD/smux

< Back Next > Finish Cancel

? < Back Next > Finish Cancel

# Open New Project (2/3)



**New Project**

**Project Type**  
Specify the type of project to create.

- RTL Project**  
You will be able to add sources, create block designs in IP Integrator, and analysis.  
 Do not specify sources at this time
- Post-synthesis Project:** You will be able to add sources, view de  
 Do not specify sources at this time
- I/O Planning Project**  
Do not specify design sources. You will be able to view part/pac
- Imported Project**  
Create a Vivado project from a Synplify, XST or ISE Project File.
- Example Project**  
Create a new Vivado project from a predefined template.

**New Project**

**Default Part**  
Choose a default Xilinx part or board for your project. This can be changed later.

Select:  Parts  Boards

Filter

Product category: All      Speed grade: -1

Family: Artix-7      Temp grade: All Remaining

Package: cpq236

Reset All Filters

Search: xc7a35tcp (1 match)

Part	I/O Pin Count	Block RAMs	DSPs	FlipFlops	GTPE2 Transceivers	Gb Transceivers	Available IOBs	LUT Elements
xc7a35tcpq236-1	136	50	90	41600	2	2	106	20800

**New Project Summary**

① A new RTL project named 'lab1' will be created.

② The default part and product family for the new project:  
Default Part: xc7a35tcpq236-1  
Product: Artix-7  
Family: Artix-7  
Package: cpq236  
Speed Grade: -1

**Finish**

< Back **Next >** Finish Cancel

< Back **Finish** Cancel

# Open New Project (3/3)



The screenshot shows the Vivado 2019.2 Project Manager interface. The main window is titled "PROJECT MANAGER - smux". The left sidebar contains a "Flow Navigator" with sections for PROJECT MANAGER, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, SYNTHESIS, and IMPLEMENTATION. The "Sources" window is open, showing a tree view of Design Sources, Constraints, Simulation Sources, and Utility Sources. The "Project Summary" window is also open, displaying project details such as name, location, and target language. The "Design Runs" window is open at the bottom, showing a table of design runs.

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy	Report Str
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2019)	Vivado Syn
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2019)	Vivado Imp

# New Source (1/5)

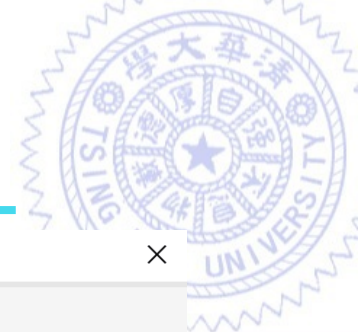


The screenshot shows the Vivado 2019.2 interface. The 'PROJECT MANAGER' window is open, displaying a tree view of sources. A right-click context menu is open over the 'Design Sources' folder, with the 'Add Sources...' option highlighted. A red arrow points from this menu to the 'Add Sources' dialog box. In the dialog box, the 'Add or create design sources' radio button is selected. Another red arrow points from this button to the 'Next >' button at the bottom of the dialog. The background shows the 'Properties' window and the 'Design Runs' table.

**Press and Right Click**

Name	Constraints	Status	WNS	TNS	WHS
synth_1	constrs_1	Not started			
impl_1	constrs_1	Not started			

# New Source (2/5)



**Add Sources**

### Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

**Create Source File**

Create a new source file and add it to your project.

File type: **Verilog**

File name: **smux.v**

File location: **<Local to Project>**

**OK** **Cancel**

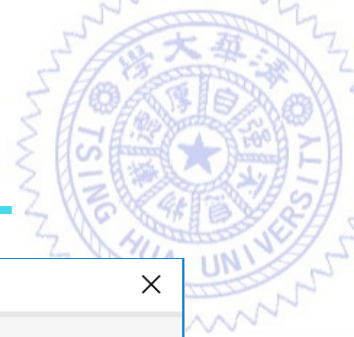
**Add Files** **Add Directories** **Create File**

Scan and add RTL include files into project  
 Copy sources into project  
 Add sources from subdirectories

**< Back** **Next >** **Finish** **Cancel**



# New Source (3/5)



The screenshot shows the 'Add Sources' dialog box with the following content:

**Add or Create Design Sources**  
Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

In...	Name	Library	Location
1	smux.v	xil_defaultlib	<Local to Project>

Buttons: Add Files, Add Directories, Create File (highlighted with a blue box)

Options:  
 Scan and add RTL include files into project  
 Copy sources into project  
 Add sources from subdirectories

Footer: ? (help icon), < Back, Next >, Finish (highlighted with a red arrow), Cancel



# New Source (4/5)



Define Module

Define a module and specify I/O Ports to add to your source file.  
For each port specified:  
MSB and LSB values will be ignored unless its Bus column is checked.  
Ports with blank names will not be written.

**Module Definition**

Module name: smux

**I/O Port Definitions**

Port Name	Direction	Bus	MSB	LSB
a	input	<input type="checkbox"/>	0	0
b	input	<input type="checkbox"/>	0	0
sel	input	<input type="checkbox"/>	0	0
out	output	<input type="checkbox"/>	0	0

OK Cancel

Design Runs

The screenshot shows the 'Define Module' dialog box. The 'Module name' field contains 'smux'. The 'I/O Port Definitions' table has four rows: 'a', 'b', 'sel', and 'out'. The 'out' row is selected, and its 'Direction' dropdown menu is open, showing 'output' as the selected option. A red circle highlights the 'out' row and the dropdown menu. A red arrow points from the 'output' option in the dropdown to the 'OK' button.

# New Source (5/5)



smux - [C:/Users/hp/LD/smux/smux.xpr] - Vivado 2019.2

File Edit Flow Tools Reports Window Layout View Help

Flow Navigator PROJECT MANAGER - smux

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation
- Open Implemented Design

PROGRAM AND DEBUG

- Generate Bitstream
- Open Hardware Manager

Sources

- Design Sources (1)
  - smux (smux.v)
- Constraints
- Simulation Sources (1)
- Utility Sources

Source File Properties

smux.v

Location: C:/Users/hp/LD/smux/smux.srscs/sources\_1/new

Type: Verilog

Library: xil\_defaultlib

Size: 0.5 KB

General Properties

Tcl Console Messages Log Reports Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	St
synth_1	constrs_1	Not started													
impl_1	constrs_1	Not started													

Project Summary x smux.v

```
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module smux(
24     input a,
25     input b,
26     input sel,
27     output out
28 );
29     andmodule
30
```

Project Summary x smux.v \*

```
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module smux(
24     input a,
25     input b,
26     input sel,
27     output out
28 );
29
30     assign out = (a&sel) | (b&(~sel));
31
32 endmodule
33
```

Remember to save files

press and double click

assign out = (a&sel) | (b&(~sel));

# Add Testbench (1/5)



The screenshot shows the Vivado 2019.2 interface. The 'Project Manager' window is open, displaying the 'Sources' tab. A red circle highlights the 'Simulation Sources (1)' folder, and a red arrow points to the 'Add Sources...' option in the context menu. The 'Add Sources' dialog box is open, showing the 'Add or create simulation sources' option selected. A red arrow points from this option to the 'Next >' button at the bottom of the dialog.

**Press and Right Click**

**Add Sources**

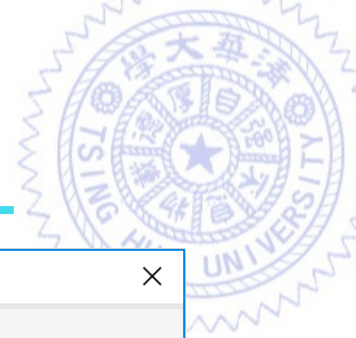
This guides you through the process of adding and creating sources for your project

- Add or create constraints
- Add or create design sources
- Add or create simulation sources

**Next >**

Name	Constraints	Status	WNS	TNS	WHS
synth_1	constrs_1	Not started			
impl_1	constrs_1	Not started			

# Add Testbench (2/5)



The screenshot shows the 'Add Sources' dialog box in a CAD tool. The main dialog is titled 'Add Sources' and contains the following elements:

- Add or Create Simulation Sources**: A section with the instruction: 'Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.'
- Specify simulation set:** A dropdown menu showing 'sim\_1'.
- Buttons:** '+', '-', '↑', and '↓' icons for file management.
- Buttons:** 'Add Files', 'Add Directories', and 'Create File' (circled in red).
- Checkboxes:** Four checked options: 'Scan and add RTL include files into project', 'Copy sources into project', 'Add sources from subdirectories', and 'Include all design sources for simulation'.
- Footer:** '?', '< Back', 'Next >', 'Finish', and 'Cancel' buttons.

The 'Create Source File' sub-dialog is open, showing:

- Create a new source file and add it to your project.**
- File type:** 'Verilog' (circled in red).
- File name:** 'test\_mux.v' (circled in red).
- File location:** '<Local to Project>' (circled in red).
- Buttons:** 'OK' and 'Cancel'.

Red arrows point from the 'OK' button in the sub-dialog to the 'Create File' button in the main dialog, and from the 'File type' and 'File name' fields to the 'Create File' button.

# Add Testbench (3/5)



### Add Sources

#### Add or Create Simulation Sources

Specify simulation specific HDL files, or directories containing HDL files, to your project.

Specify simulation set:

In...	Name	Library	Location
1	test_smux.v	xil_defaultlib	<Local to Project>

Add Files    Add Directories    Create File

- Scan and add RTL include files into project
- Copy sources into project
- Add sources from subdirectories
- Include all design sources for simulation

### Define Module

Define a module and specify I/O Ports to add to your source file.  
For each port specified:  
MSB and LSB values will be ignored unless its Bus column is checked.  
Ports with blank names will not be written.

**Module Definition**

Module name:

**I/O Port Definitions**

Port Name	Direction	Bus	MSB	LSB
	input	<input type="checkbox"/>	0	0

# Add Testbench (4/5)



The screenshot displays the Vivado 2019.2 interface. The **PROJECT MANAGER** window on the left shows the project hierarchy. Under **Simulation Sources (2)**, the **test\_smux (test\_smux.v)** source is highlighted with a red circle. A red arrow points from this circle to the **Source File Properties** window, which shows the file's location and type (Verilog). The **Code Editor** window on the right shows the Verilog code for **test\_smux.v**, with the `module test_smux(` line highlighted in yellow. A red arrow points from the highlighted text in the **Source File Properties** window to the highlighted code in the **Code Editor**. The **Design Runs** table at the bottom shows the status of the synthesis and implementation runs.

**Double Click and Edit**

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy	Report Str
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2019)	Vivado Syn
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2019)	Vivado Imp

# Add Testbench (5/5)

The screenshot displays the Vivado 2019.2 software interface. On the left, the 'PROJECT MANAGER' pane shows the project hierarchy with 'test\_smux (test\_smux.v)' selected under 'Simulation Sources'. Below it, the 'Source File Properties' pane shows the file is enabled and located at 'C:/Users/hp/LD/smux/smux.srcs/sim\_1/new'. The main editor window shows the Verilog code for 'test\_smux.v', which includes module declarations, register declarations, an instance of 'smux', and a series of testbench stimuli. The code is as follows:

```
21
22
23 module test_smux();
24 wire OUT;
25 reg A, B, SEL;
26
27 smux U0(.a(A), .b(B), .sel(SEL), .out(OUT));
28
29 initial
30 begin
31     A=0;B=0;SEL=0;
32     #10 A=0;B=0;SEL=1;
33     #10 A=0;B=1;SEL=0;
34     #10 A=0;B=1;SEL=1;
35     #10 A=1;B=0;SEL=1;
36     #10 A=1;B=0;SEL=0;
37     #10 A=1;B=1;SEL=1;
38     #10 A=1;B=1;SEL=0;
39     #10 A=0;B=0;SEL=1;
40     #10 A=0;B=0;SEL=0;
41 end
42
```

The bottom of the interface shows the 'Design Runs' table with columns for Name, Constraints, Status, WNS, TNS, WHS, THS, TPWS, Total Power, Failed Routes, LUT, FF, BRAM, URAM, DSP, Start, Elapsed, Run Strategy, and Report Str. The table shows two runs: 'synth\_1' and 'impl\_1', both with a status of 'Not started'.

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy	Report Str
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2019)	Vivado Syn
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2019)	Vivado Imp

# Simulation (1/4)



smux - [C:/Users/hp/LD/smux/smux.xpr] - Vivado 2019.2

File Edit Flow Tools Reports Window Layout View Help

Flow Navigator PROJECT MANAGER - smux

Sources

- Design Sources (1)
  - smux (smux.v)
- Constraints
- Simulation Sources (1)
  - sim\_1 (1)
    - test\_smux (test\_smux.v) (1)
- Utility Sources

Hierarchy Libraries Compile Order

Source File Properties

Run Simulation

Executing simulate step...

Background Cancel

```
21
22
23 module test_smux();
24 wire OUT;
25 reg A, B, SEL;
```

37 #10 A=1;B=0;SEL=1;
38 #10 A=1;B=1;SEL=0;
39 #10 A=1;B=1;SEL=1;
40 #10 A=0;B=0;SEL=0;
41 end
42

Run Behavioral Simulation

Press and Select

Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy	Report Stra
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2019)	Vivado Syn
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2019)	Vivado Imp



# Simulation (2/4)



smux - [C:/Users/hp/LD/smux/smux.xpr] - Vivado 2019.2

File Edit Flow Tools Reports Window Layout View Run Help Q: Quick Access

100 us

Default Layout

Ready

Press to maximize

Flow Navigator

**SIMULATION - Behavioral Simulation - Functional - sim\_1 - test\_smux**

**PROJECT MANAGER**

- Settings
- Add Sources
- Language Templates
- IP Catalog

**IP INTEGRATOR**

- Create Block Design
- Open Block Design
- Generate Block Design

**SIMULATION**

- Run Simulation

**RTL ANALYSIS**

- Open Elaborated Design

**SYNTHESIS**

- Run Synthesis
- Open Synthesized Design

**IMPLEMENTATION**

- Run Implementation
- Open Implemented Design

**PROGRAM AND DEBUG**

- Generate Bitstream
- Open Hardware Manager

**Scope**

Na...	De...	Bl...
test_s	Verilo	
smux	Verilo	
glbl	Verilo	

**Objects**

Na...	Va...	Da...
0	Logic	
0	Logic	
0	Logic	
0	Logic	

smux.v x test\_smux.v x Untitled 1 x

Name	Value	999,990 ps	999,992 ps	999,994 ps	999,996 ps	999,998 ps	1,000,000 ps
OUT	0						
A	0						
B	0						
SEL	0						

**Tcl Console**

```
# }
# run 1000ns
xsim: Time (s): cpu = 00:00:05 ; elapsed = 00:00:06 . Memory (MB): peak = 713.539 ; gain = 11.770
INFO: [USF-XSim-96] XSim completed. Design snapshot 'test_smux_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:07 ; elapsed = 00:00:16 . Memory (MB): peak = 713.539 ; gain = 13.695
```

Type a Tcl command here

Sim Time: 1 us

# Simulation (3/4)



Use to zoom in or zoom out

The screenshot shows a behavioral simulation window titled "SIMULATION - Behavioral Simulation - Functional - sim\_1 - test\_smux". The window contains a signal table with the following data:

Name	Value	999,990 ps	999,992 ps	999,994 ps	999,996 ps	999,998 ps	1,000,000 ps	1,000,002 ps	1,000,004 ps	1,000,006 ps	1,000,008 ps	1,000,010 ps	1,000,012 ps
OUT	0												
A	0												
B	0												
SEL	0												

A yellow vertical line is positioned at the 1,000,000 ps mark. A yellow arrow points to the scrollbar at the bottom right of the table, which is circled in red. The text "Scroll this to the beginning" is written in red below the arrow. The text "Use to zoom in or zoom out" is written in red above the zoom-in and zoom-out icons in the toolbar, which are also circled in red.

Scroll this to the beginning

Sim Time: 1 us

# Simulation (4/4)



Press to close simulation

smux - [C:/Users/hp/LD/smux/smux.xpr] - Vivado 2019.2

File Edit Flow Tools Reports Window Layout View Run Help Q: Quick Access Ready

Flow Navigator SIMULATION - Behavioral Simulation - Functional - sim\_1 - test\_smux

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation
- Open Implemented Design

PROGRAM AND DEBUG

- Generate Bitstream
- Open Hardware Manager

Scope Sources

Name	Value	0.000 ns	10.000 ns	20.000 ns	30.000 ns	40.000 ns	50.000 ns	60.000 ns	70.000 ns	80.000 ns	90.000 ns	100.000 ns	110.000 ns
OUT	0			█									
A	0					█							
B	0												
SEL	0		█										

Protocol Instances | Objects

Tcl Console Messages Log

Sim Time: 1 us