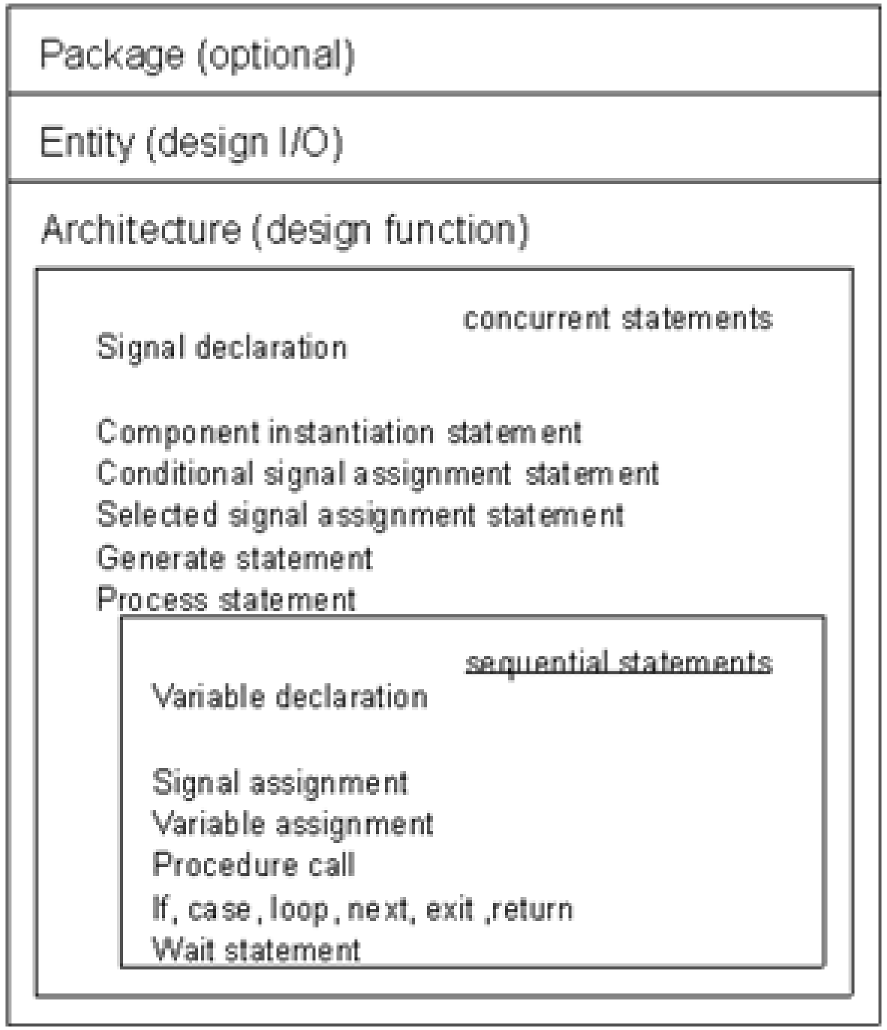
DIGITAL LOGIC in VHDL

Index

VHDL explanation



Package

```
library ieee;
use ieee. std_logic_1164.all;
```

Entity

```
entity mux21 is
port ( a, b    : in std_logic_vector(3 downto 0);
          s     : in std_logic;
          y     : out std_logic_vector(3 downto 0));
end mux21;
```

Architecture

```
architecture rtl of mux21 is
begin
   process(a, b, s)
   begin
      if ( s = '0') then
         y <= a;
      else
         y <= b;
      end if;
   end process;
end rtl;
```

VHDL explanation

In Architecture, there are three description

Dataflow

- LOGIC

```
Y <= X AND B;
```

Structural

- Interconnection between components

```
F1 : xxx port map (…);
F2 : xxx port map (…);
F3 : xxx port map (…);
F4 : xxx port map (…);
```

Sequential

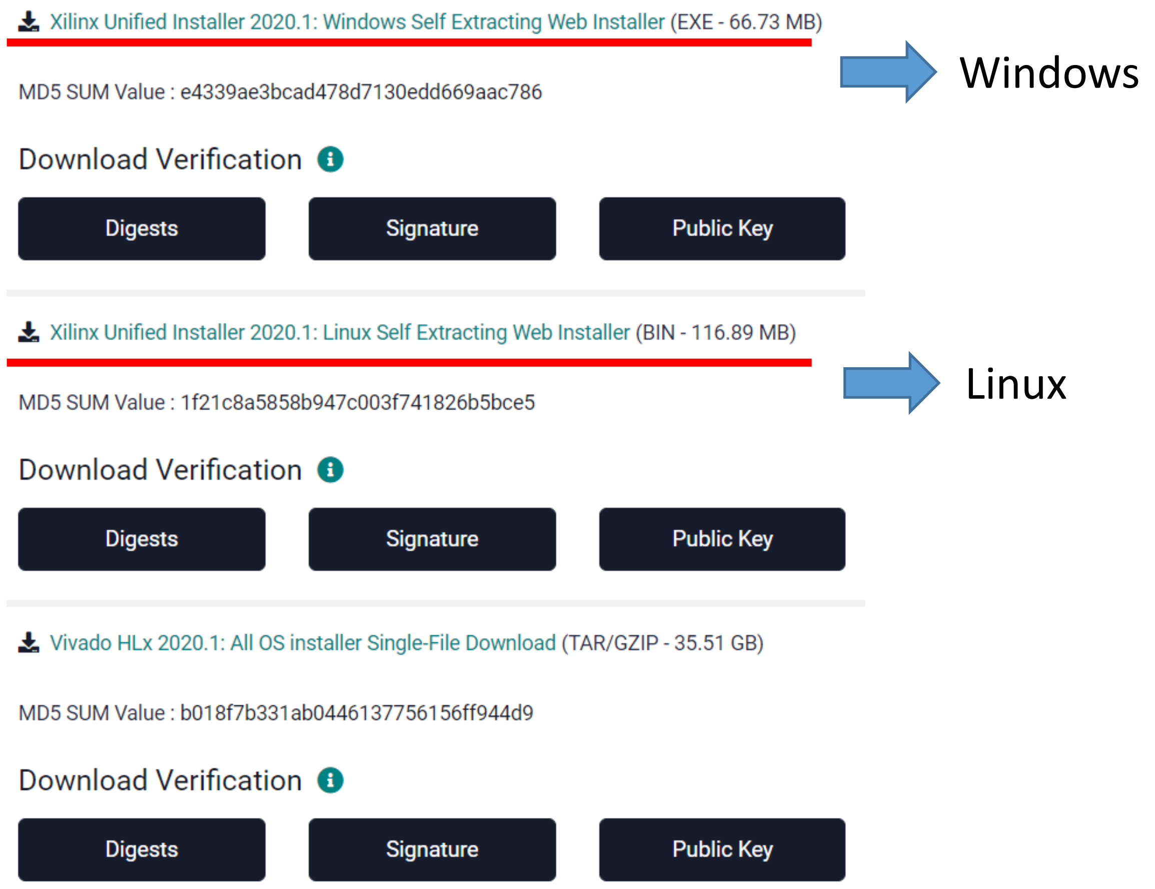
- When there is an order

```
Process (…)
…
End process;
```

Vivado Installation

Download Link : https://www.xilinx.com/support/download.html

1) Create Account ***with school email***
2) Choose an installer that suits your environment


Xilinx Unified Installer 2020.1: Windows Self Extracting Web Installer (EXE - 66.73 MB)

MD5 SUM Value : e4339ae3bcad478d7130edd669aac786

Download Verification

Xilinx Unified Installer 2020.1: Linux Self Extracting Web Installer (BIN - 116.89 MB)

MD5 SUM Value : 1f21c8a5858b947c003f741826b5bce5

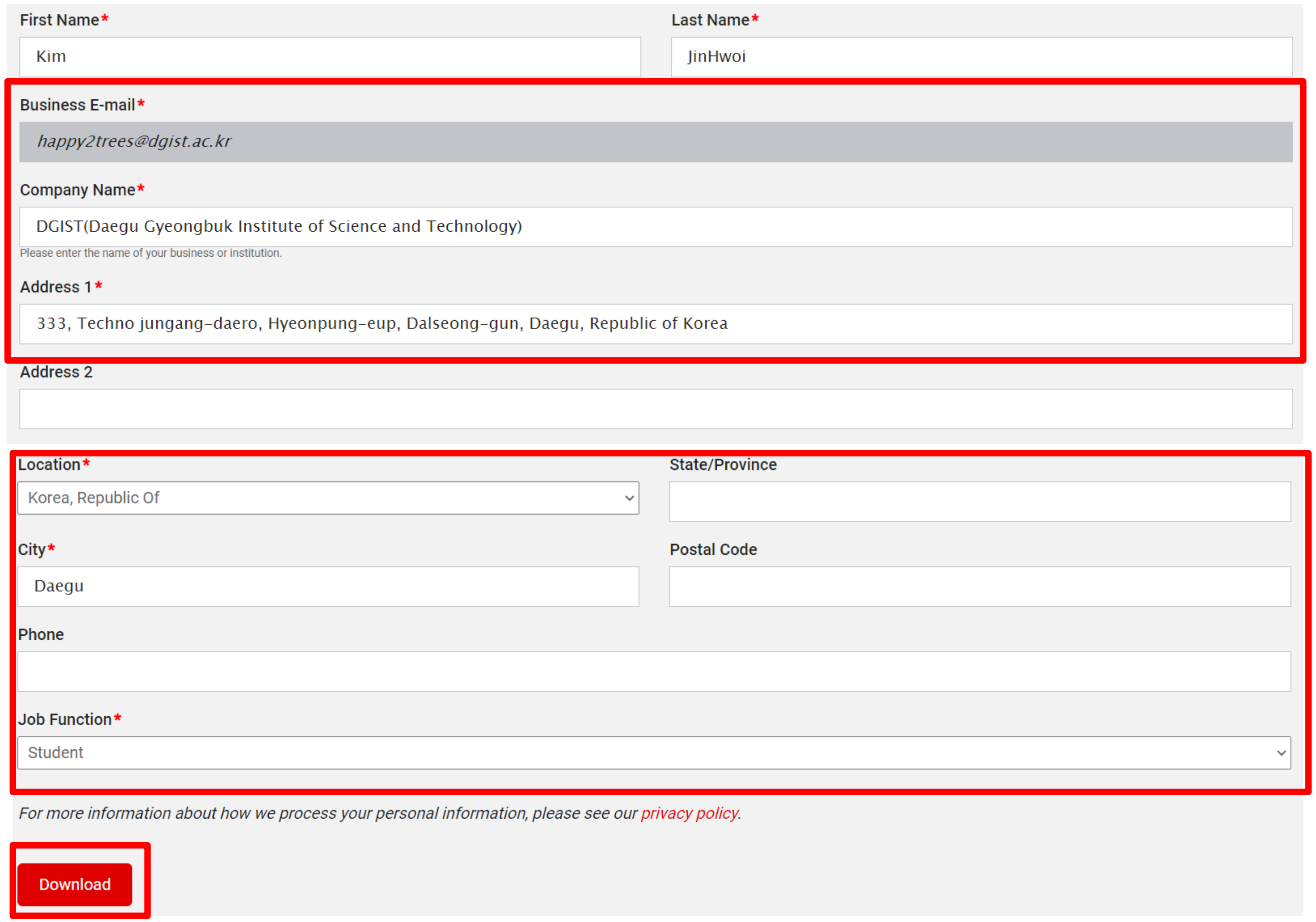
Download Verification

Vivado HLx 2020.1: All OS installer Single-File Download (TAR/GZIP - 35.51 GB)

MD5 SUM Value : b018f7b331ab0446137756156ff944d9

Download Verification

Vivado Installation



First Name*	Last Name*
Kim	JinHwoi

Business E-mail*
happy2trees@dgist.ac.kr

Company Name*
DGIST(Daegu Gyeongbuk Institute of Science and Technology)
Please enter the name of your business or institution.

Address 1*
333, Techno jungang-daero, Hyeonpung-eup, Dalseong-gun, Daegu, Republic of Korea

Address 2

Location*	State/Province
Korea, Republic Of	
City*	Postal Code
Daegu	

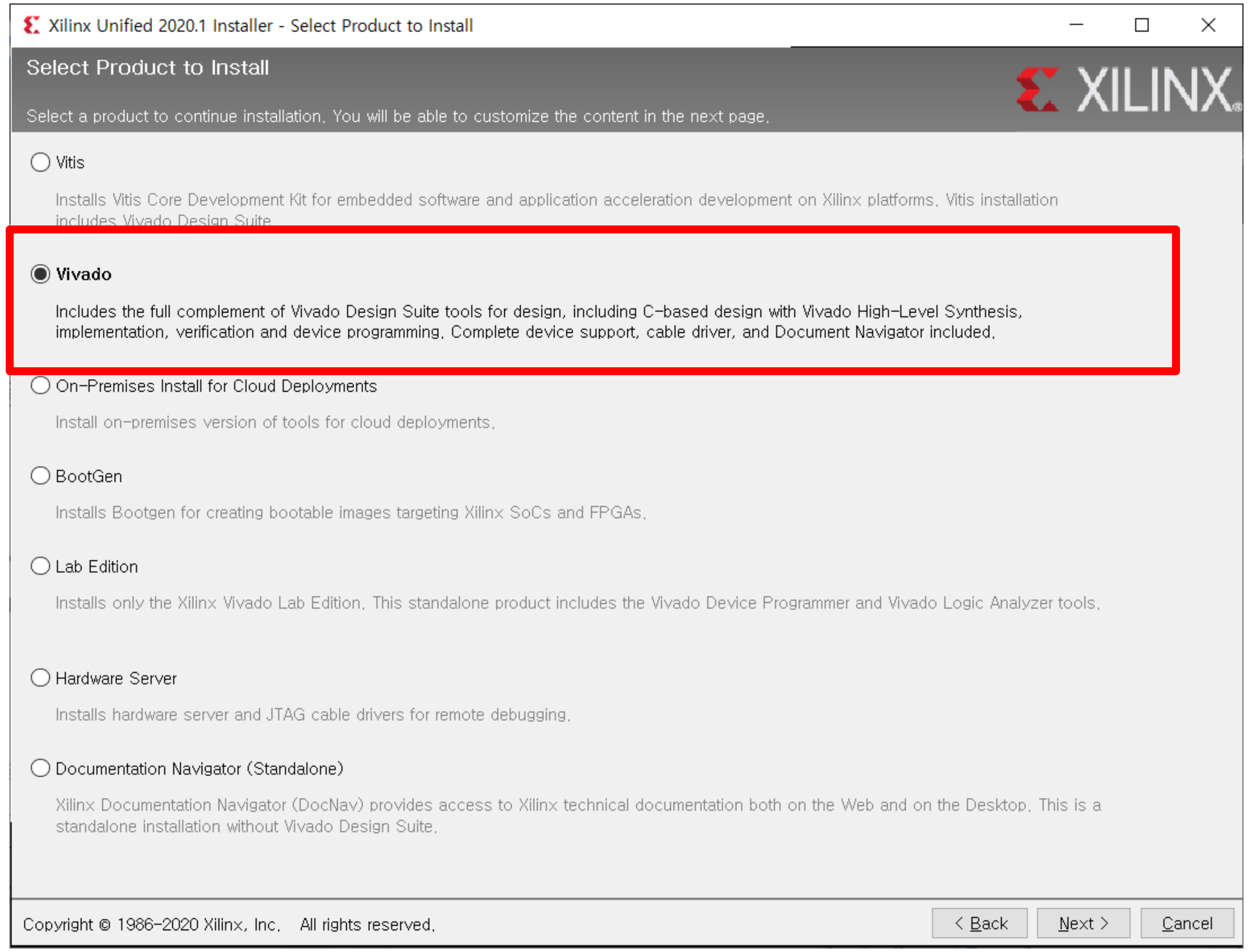
Phone

Job Function*
Student

For more information about how we process your personal information, please see our privacy policy.

Fill with this information

Vivado Installation



Xilinx Unified 2020.1 Installer - Select Product to Install

Select Product to Install

Select a product to continue installation. You will be able to customize the content in the next page.

( ) Vitis

Installs Vitis Core Development Kit for embedded software and application acceleration development on Xilinx platforms. Vitis installation includes Vivado Design Suite.

[✓ Vivado]

Includes the full complement of Vivado Design Suite tools for design, including C-based design with Vivado High-Level Synthesis, implementation, verification and device programming. Complete device support, cable driver, and Document Navigator included.

( ) On-Premises Install for Cloud Deployments

Install on-premises version of tools for cloud deployments.

( ) BootGen

Installs Bootgen for creating bootable images targeting Xilinx SoCs and FPGAs.

( ) Lab Edition

Installs only the Xilinx Vivado Lab Edition. This standalone product includes the Vivado Device Programmer and Vivado Logic Analyzer tools.

( ) Hardware Server

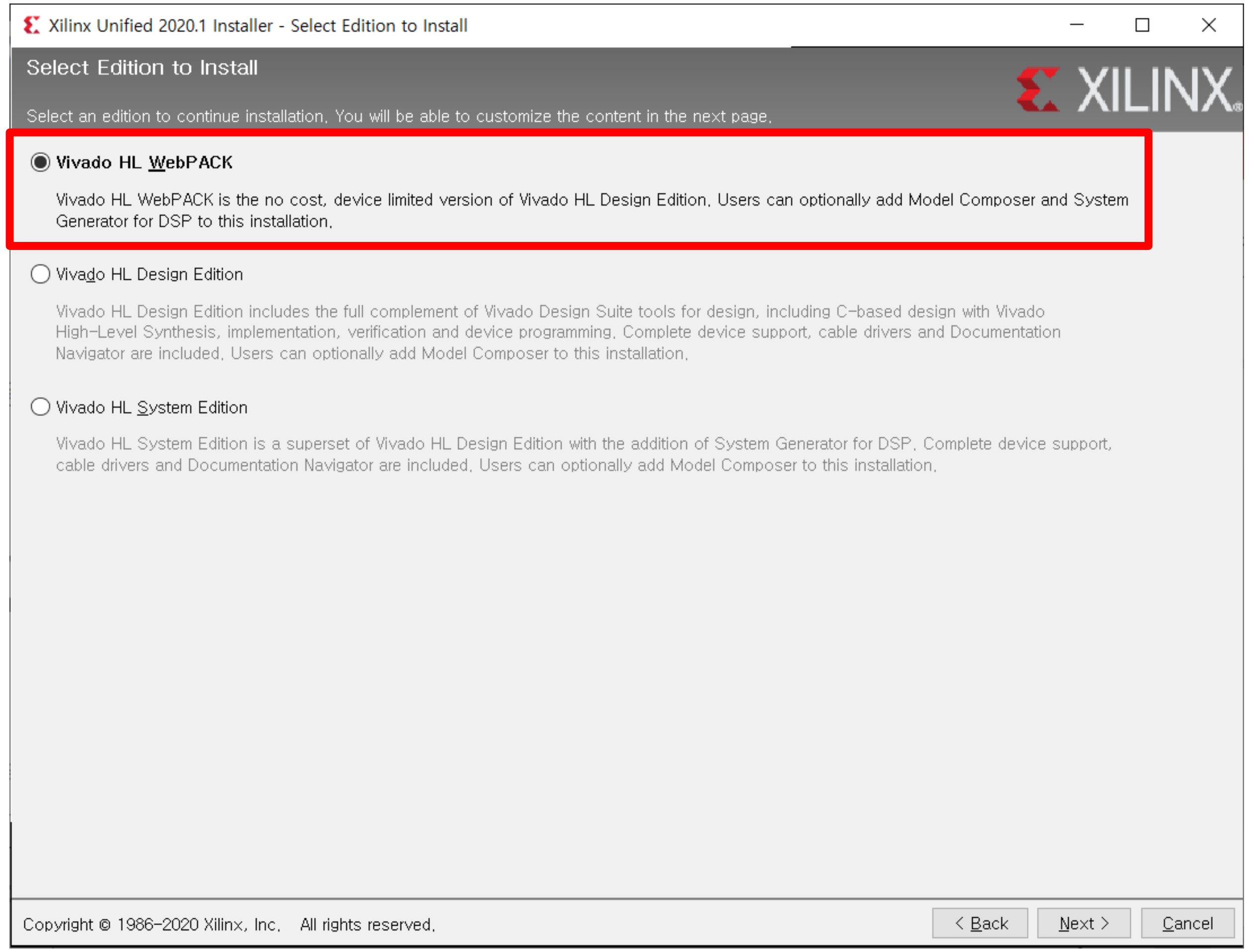
Installs hardware server and JTAG cable drivers for remote debugging.

( ) Documentation Navigator (Standalone)

Xilinx Documentation Navigator (DocNav) provides access to Xilinx technical documentation both on the Web and on the Desktop. This is a standalone installation without Vivado Design Suite.

Copyright © 1986-2020 Xilinx, Inc. All rights reserved.

Vivado Installation



Xilinx Unified 2020.1 Installer - Select Edition to Install

Select Edition to Install

Select an edition to continue installation. You will be able to customize the content in the next page.

[● Vivado HL WebPACK]

Vivado HL WebPACK is the no cost, device limited version of Vivado HL Design Edition. Users can optionally add Model Composer and System Generator for DSP to this installation.

○ Vivado HL Design Edition

Vivado HL Design Edition includes the full complement of Vivado Design Suite tools for design, including C-based design with Vivado High-Level Synthesis, implementation, verification and device programming. Complete device support, cable drivers and Documentation Navigator are included. Users can optionally add Model Composer to this installation.

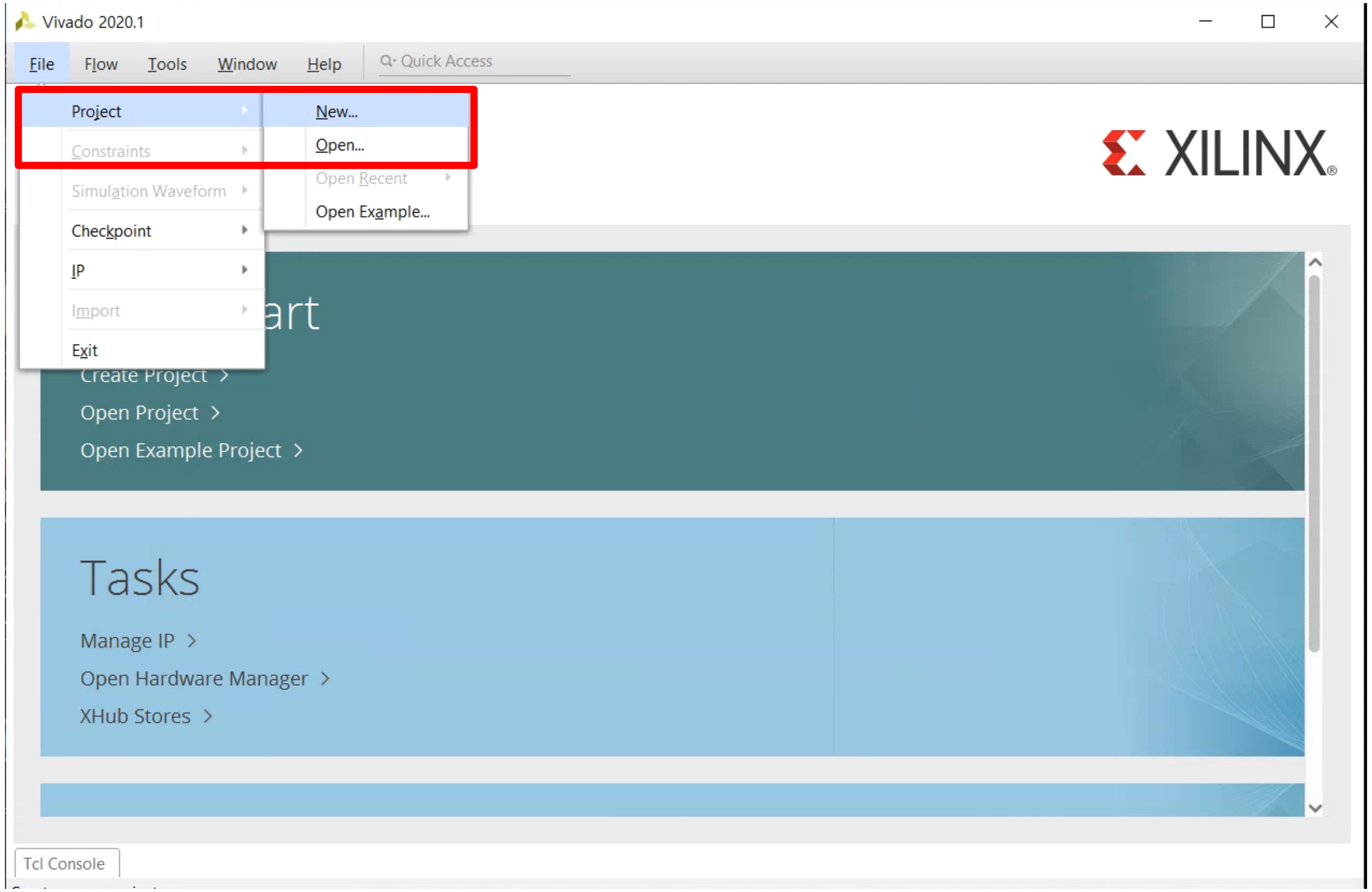
○ Vivado HL System Edition

Vivado HL System Edition is a superset of Vivado HL Design Edition with the addition of System Generator for DSP. Complete device support, cable drivers and Documentation Navigator are included. Users can optionally add Model Composer to this installation.

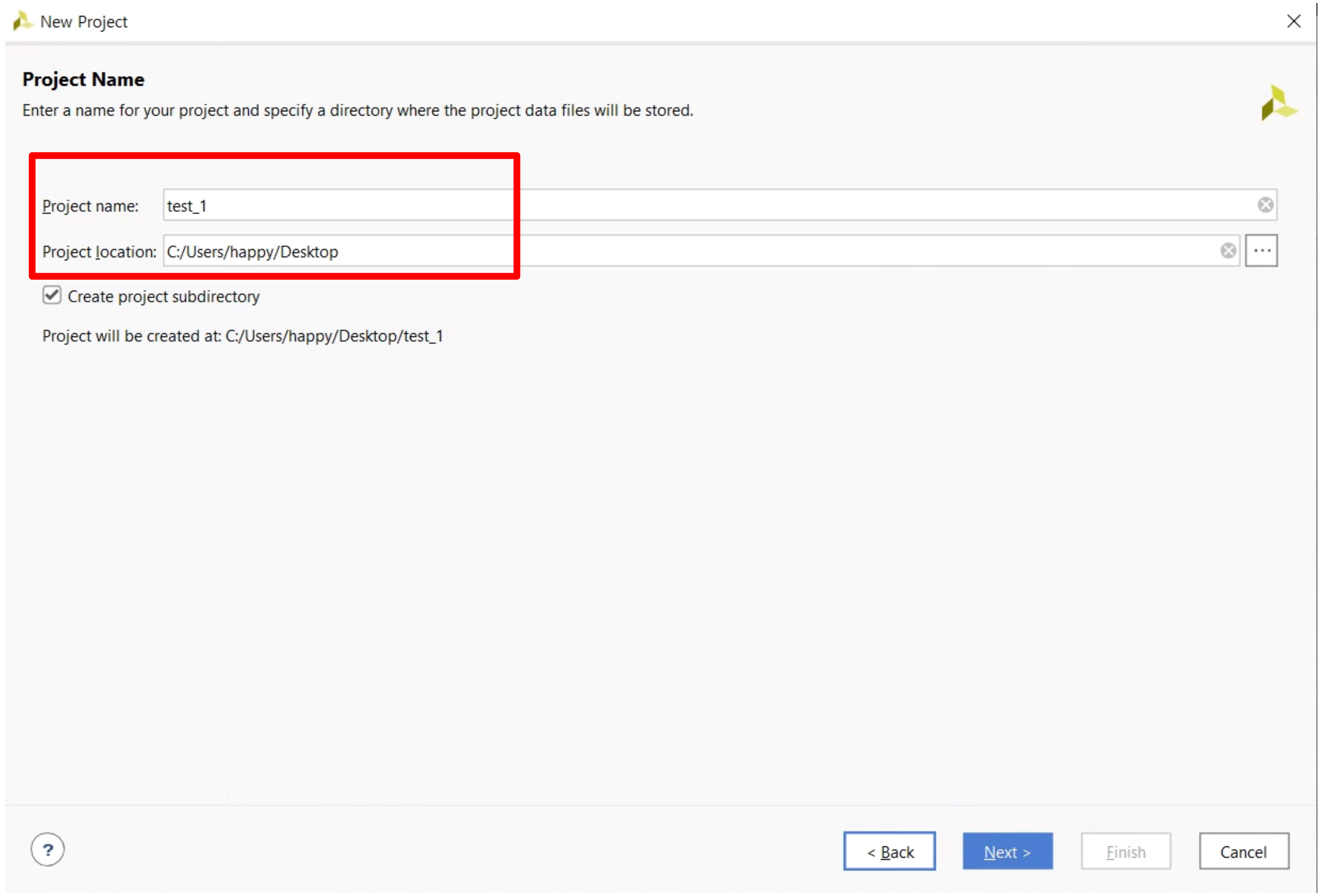
Copyright © 1986-2020 Xilinx, Inc. All rights reserved.

Free License

Create Project



Create Project



Project Name

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

Project name: test_1

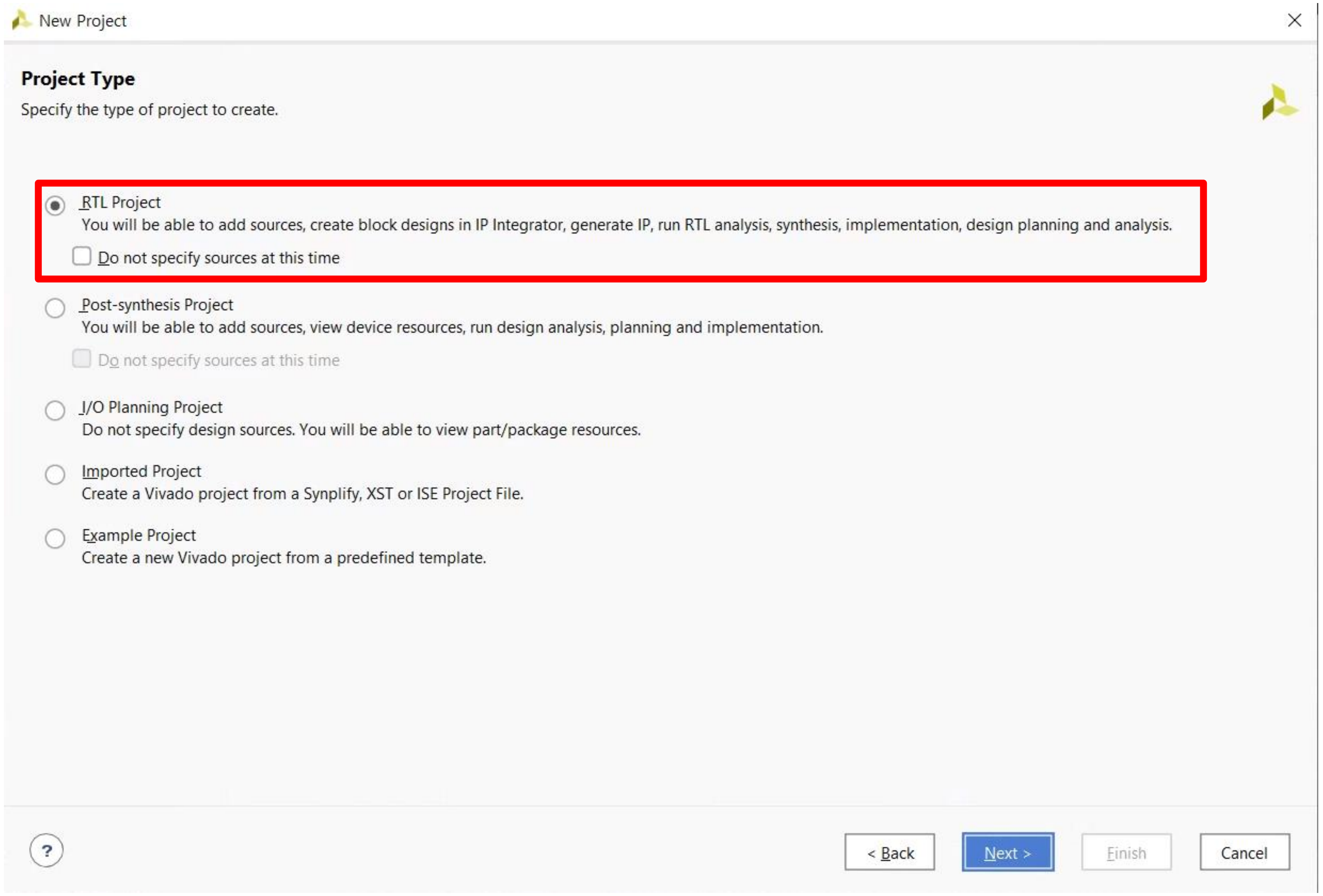
Project location: C:/Users/happy/Desktop

[✓] Create project subdirectory

Project will be created at: C:/Users/happy/Desktop/test_1

Name
Location

Create Project



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, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☐ Do not specify sources at this time

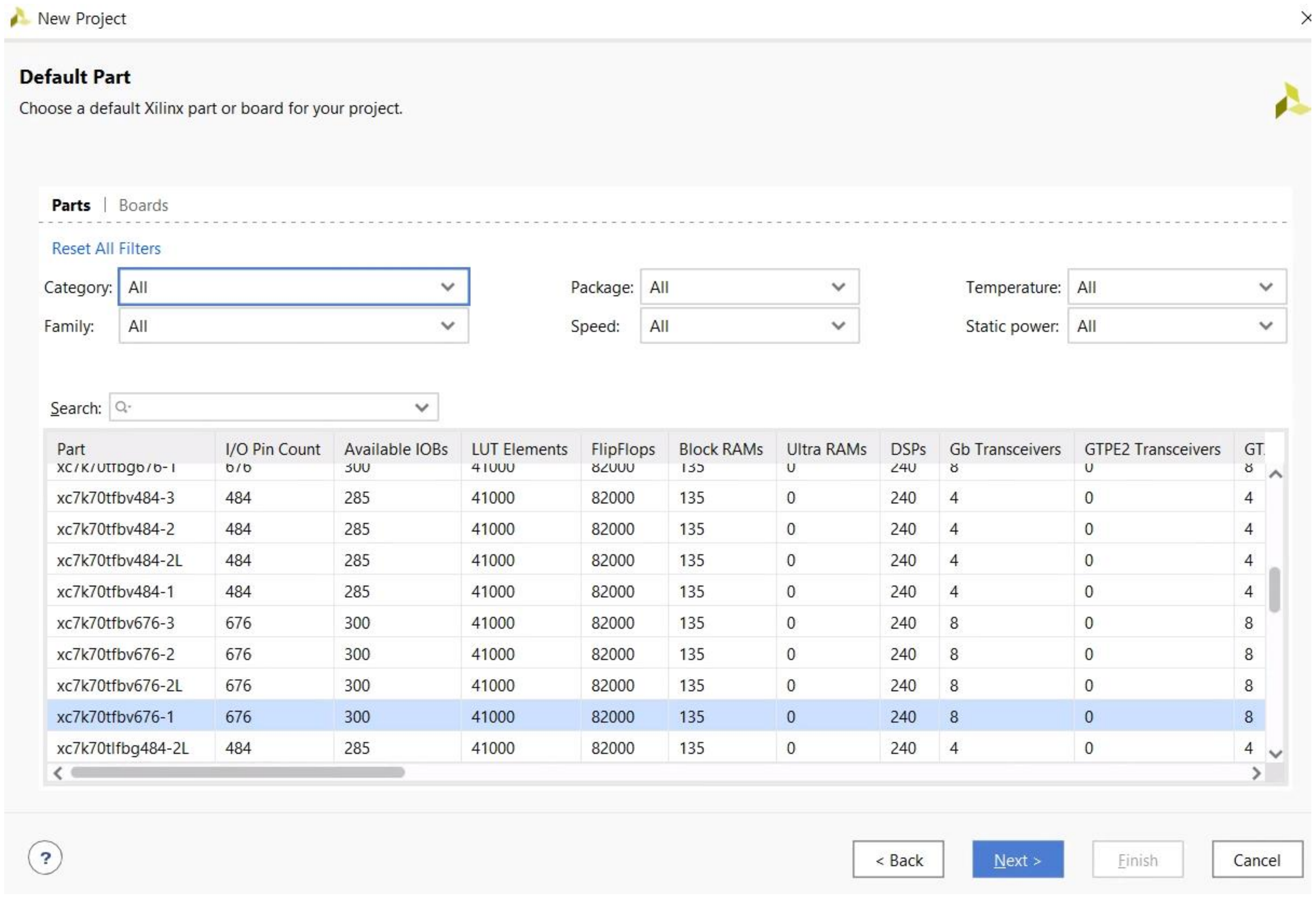
Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

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.

Create Project



New Project

Default Part

Choose a default Xilinx part or board for your project.

Parts | Boards

Reset All Filters

Category: All
Package: All
Temperature: All
Family: All
Speed: All
Static power: All

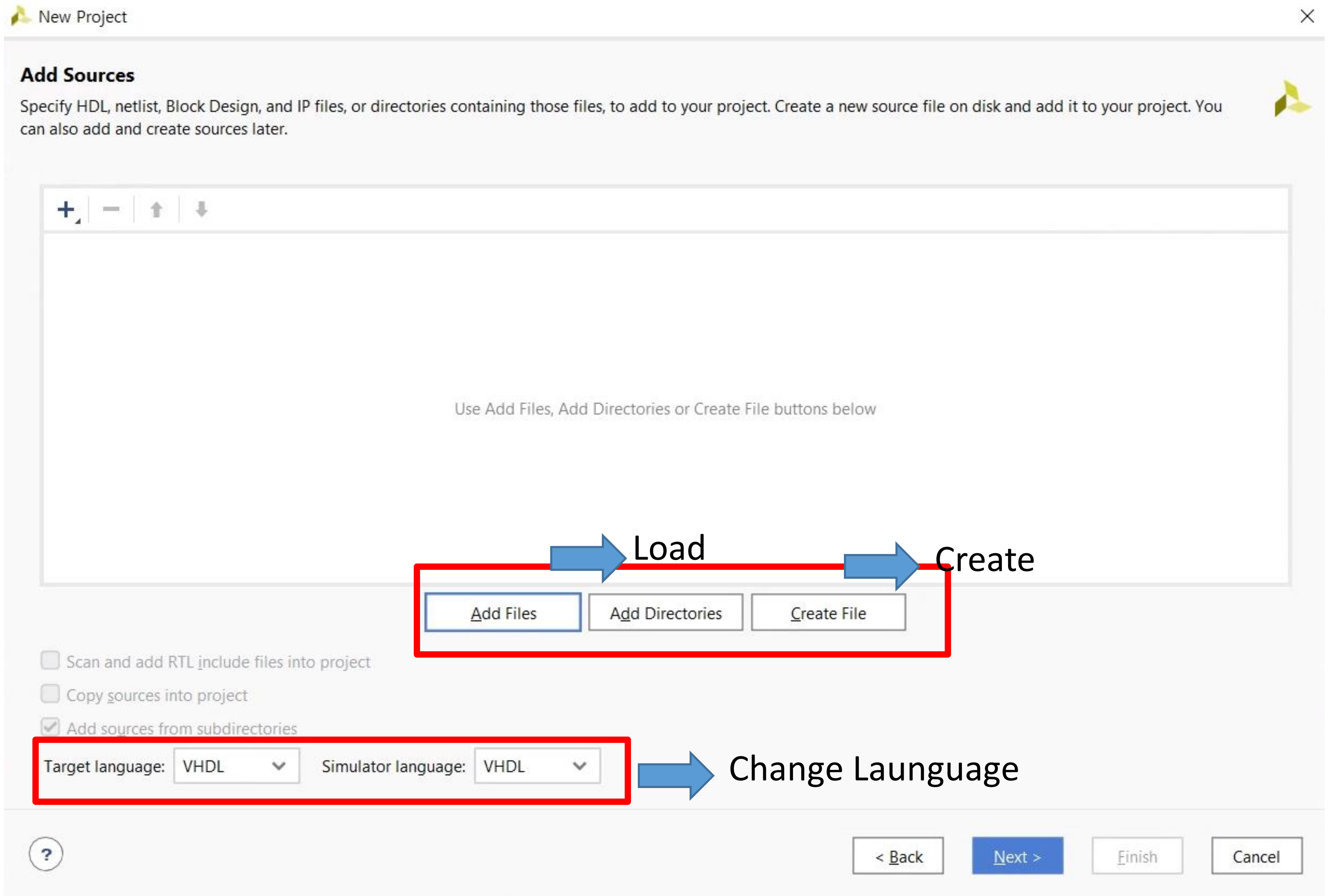
Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2 Transceivers	GT
xc7k70tfbg676-1	676	300	41000	82000	135	0	240	8	0	8
xc7k70tfbv484-3	484	285	41000	82000	135	0	240	4	0	4
xc7k70tfbv484-2	484	285	41000	82000	135	0	240	4	0	4
xc7k70tfbv484-2L	484	285	41000	82000	135	0	240	4	0	4
xc7k70tfbv484-1	484	285	41000	82000	135	0	240	4	0	4
xc7k70tfbv676-3	676	300	41000	82000	135	0	240	8	0	8
xc7k70tfbv676-2	676	300	41000	82000	135	0	240	8	0	8
xc7k70tfbv676-2L	676	300	41000	82000	135	0	240	8	0	8
xc7k70tfbv676-1	676	300	41000	82000	135	0	240	8	0	8
xc7k70tlfbg484-2L	484	285	41000	82000	135	0	240	4	0	4

➡ Default Value

Because, we only use simulation..

Create Project



New Project

Add Sources

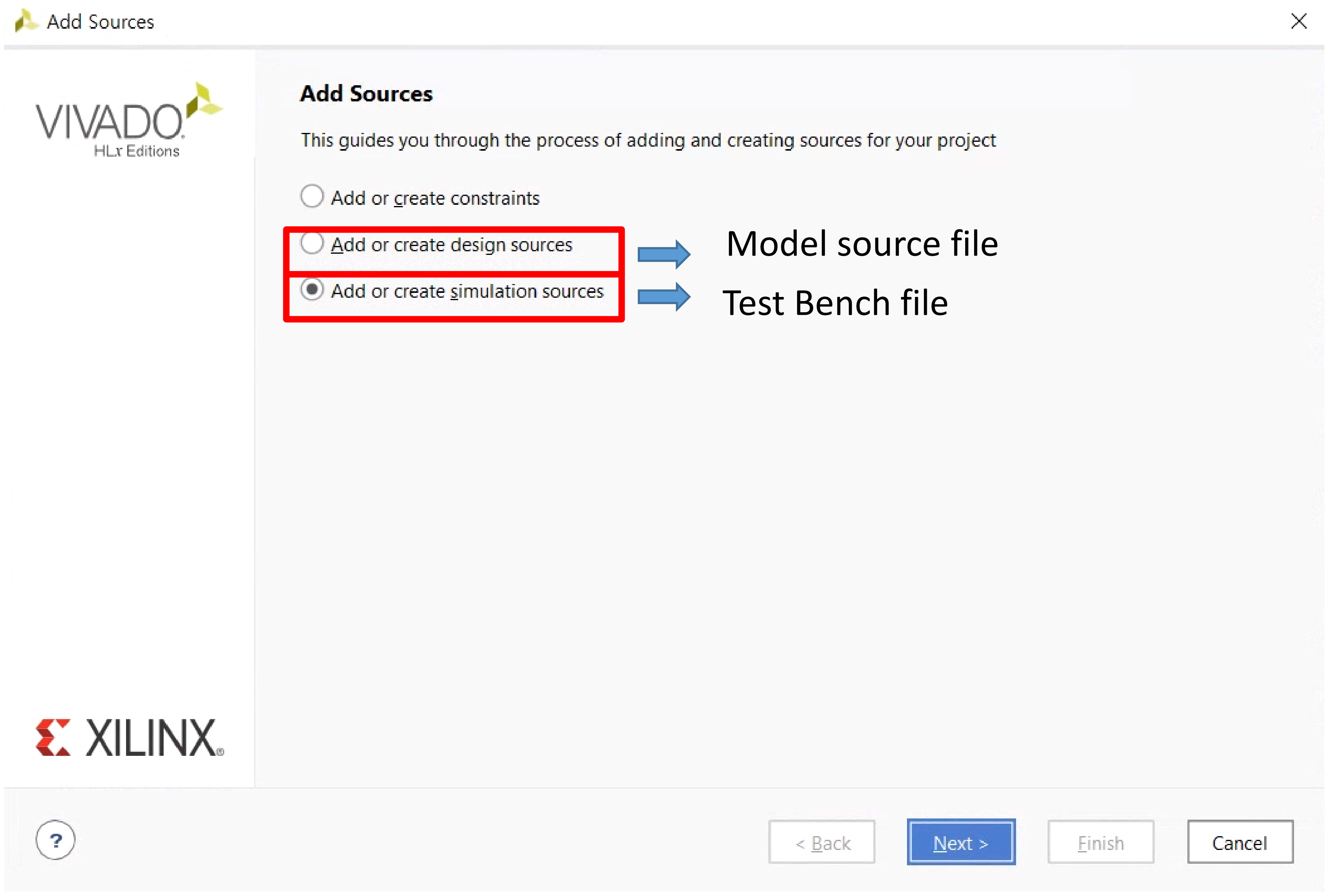
Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Use Add Files, Add Directories or Create File buttons below

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

Target language: VHDL
Simulator language: VHDL

Create Project



Add Sources

Add Sources

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

Add or create constraints

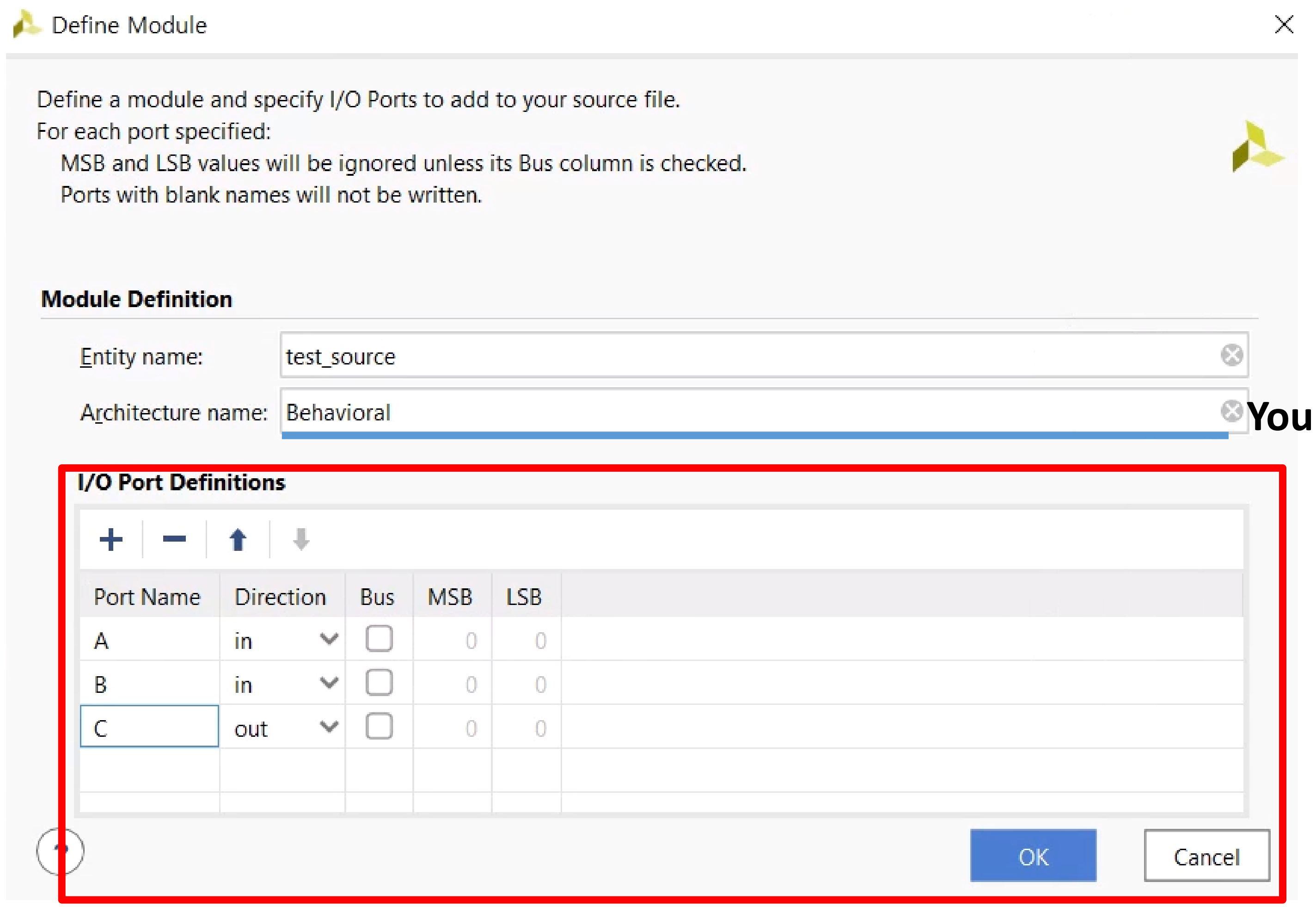
Add or create design sources

[✓ Add or create simulation sources]

Model source file

Test Bench file

Create Project



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

Entity name: test_source

Architecture name: Behavioral

I/O Port Definitions

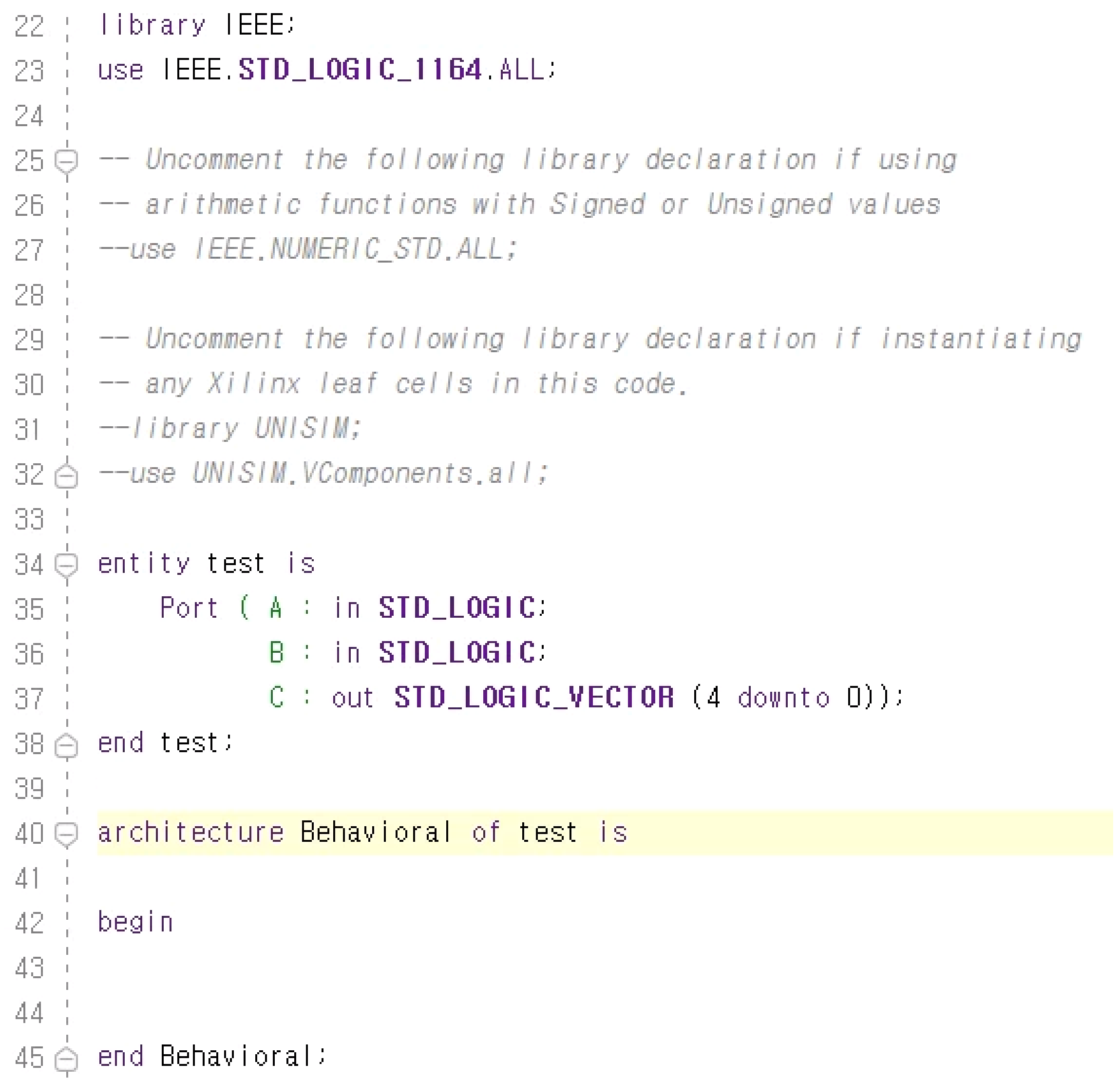
Port Name	Direction	Bus	MSB	LSB
A	in	☐	0	0
B	in	☐	0	0
C	out	☐	0	0

- STD_LOGIC→ BUS X
- STD_LOGIC_VECTOR(3 downto 0) → BUS, MSB=3

Your mind

It creates Port and overall architecture

Create Project

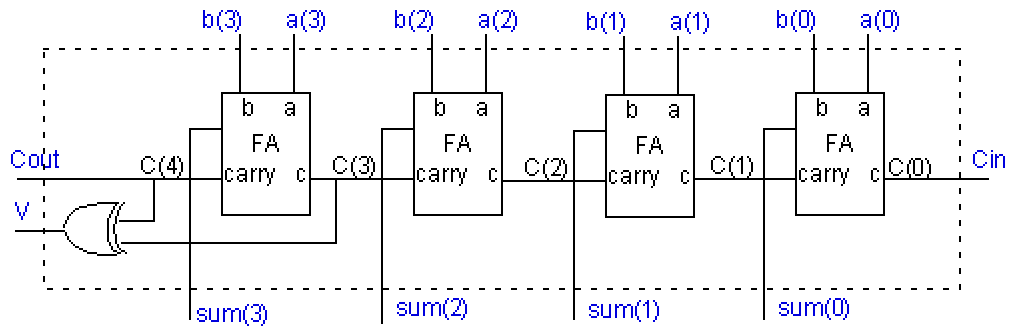


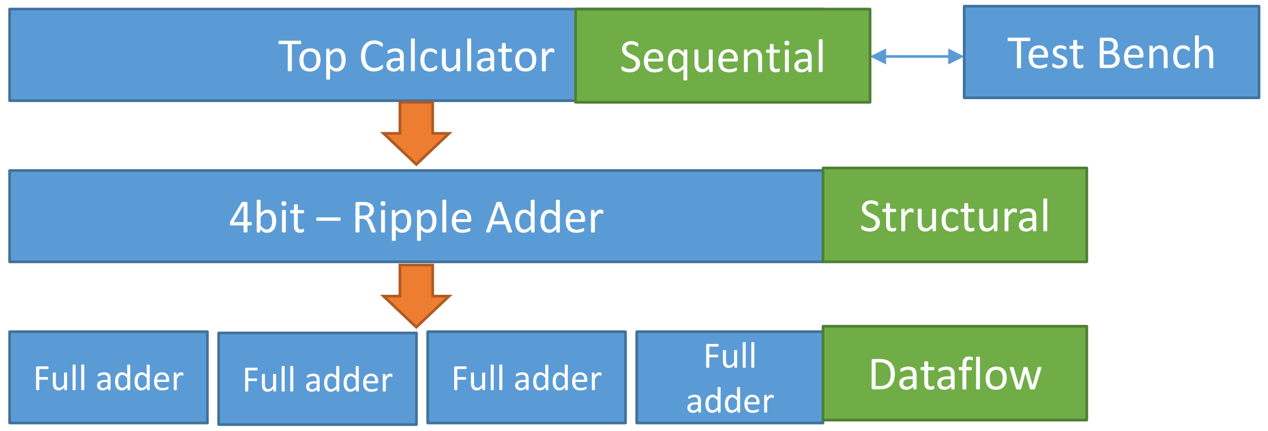
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity test is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C : out STD_LOGIC_VECTOR (4 downto 0));
end test;

architecture Behavioral of test is

begin


end Behavioral;
```

Port

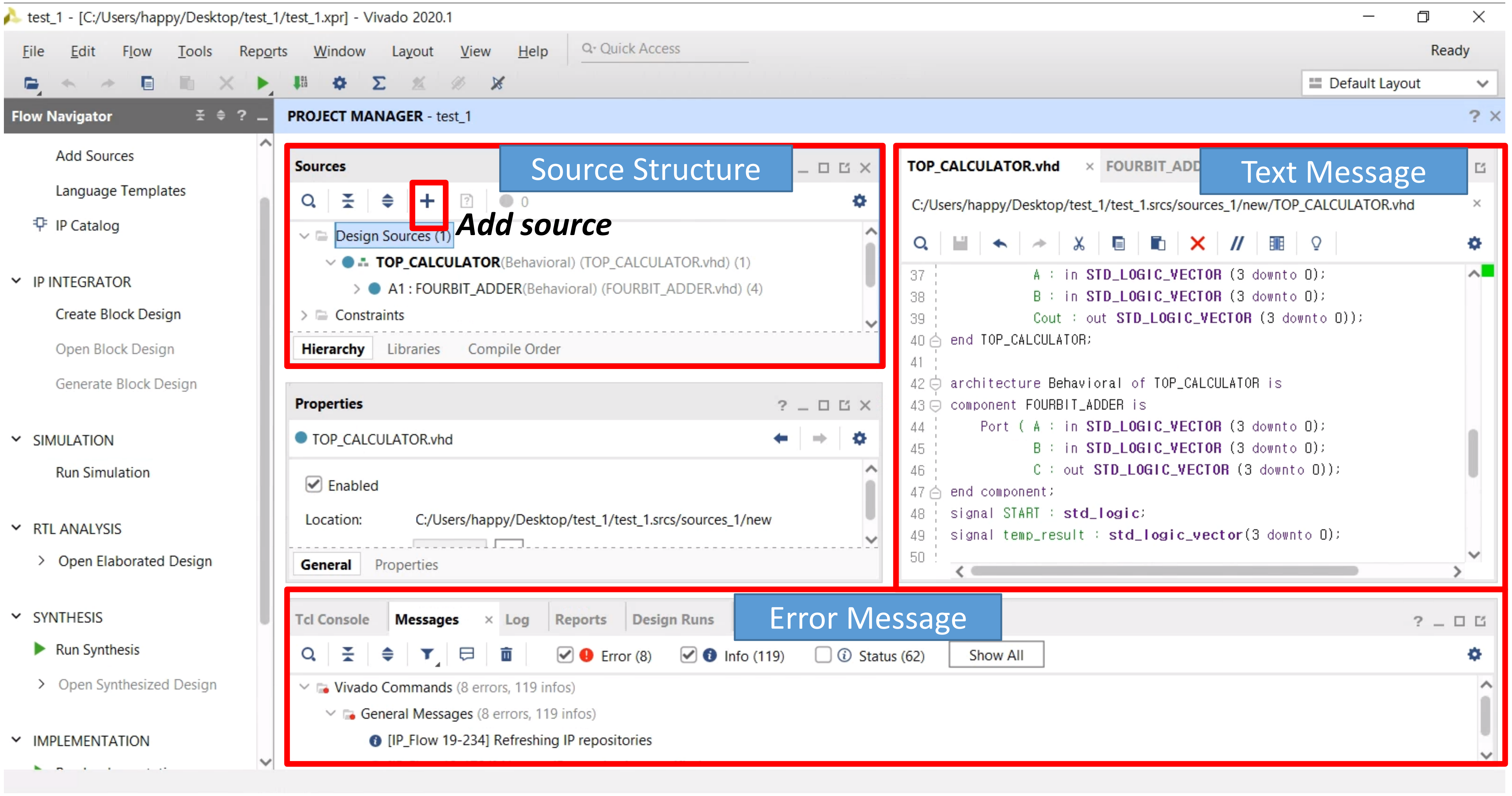
Contents

4-Bit Ripple Adder example





4-Bit Ripple Adder example



test_1 - [C:/Users/happy/Desktop/test_1/test_1.xpr] - Vivado 2020.1

Design Sources (1)
TOP_CALCULATOR(Behavioral) (TOP_CALCULATOR.vhd) (1)
A1 : FOURBIT_ADDER(Behavioral) (FOURBIT_ADDER.vhd) (4)
Constraints

TOP_CALCULATOR.vhd
Enabled ✓
Location: C:/Users/happy/Desktop/test_1/test_1.srcs/sources_1/new

TOP_CALCULATOR.vhd | FOURBIT_ADD

C:/Users/happy/Desktop/test_1/test_1.srcs/sources_1/new/TOP_CALCULATOR.vhd

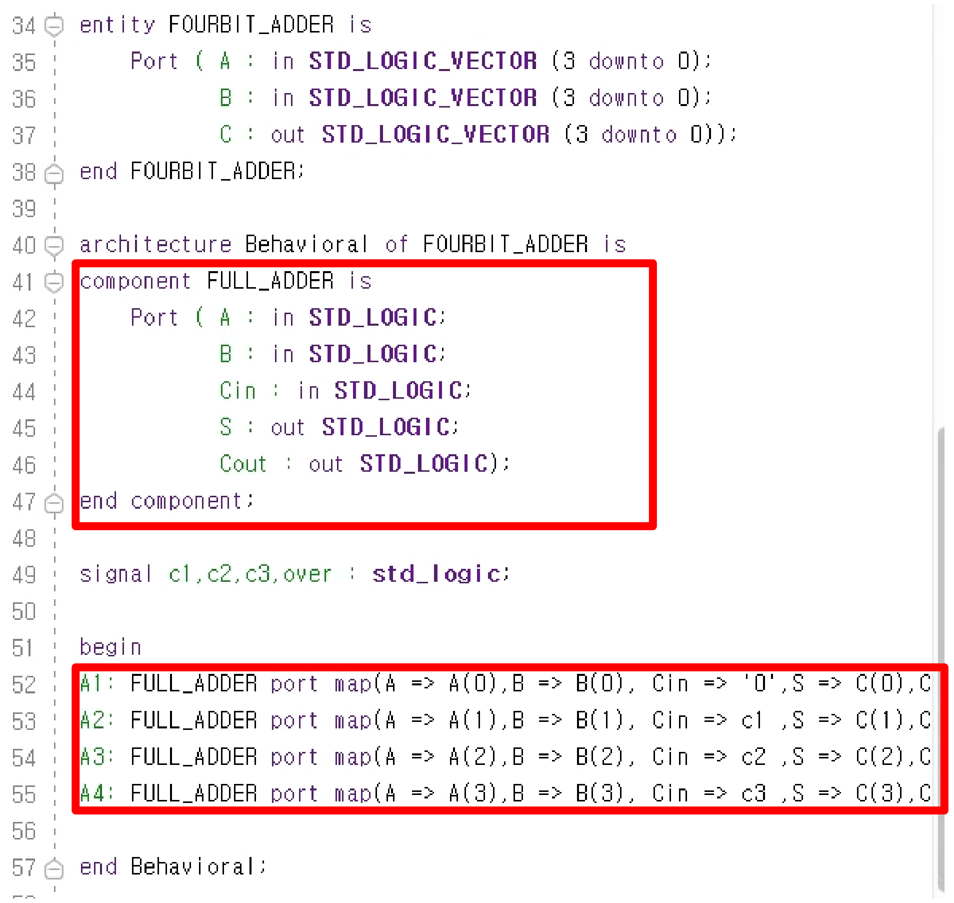
```
        A : in STD_LOGIC_VECTOR (3 downto 0);
        B : in STD_LOGIC_VECTOR (3 downto 0);
        Cout : out STD_LOGIC_VECTOR (3 downto 0));
end TOP_CALCULATOR;

architecture Behavioral of TOP_CALCULATOR is
component FOURBIT_ADDER is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           C : out STD_LOGIC_VECTOR (3 downto 0));
end component;
signal START : std_logic;
signal temp_result : std_logic_vector(3 downto 0);
```

✓ Error (8) ✓ Info (119) ☐ Status (62)

Vivado Commands (8 errors, 119 infos)
General Messages (8 errors, 119 infos)
[IP_Flow 19-234] Refreshing IP repositories

Port Mapping

How to import and use already designed components



```
entity FOURBIT_ADDER is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           C : out STD_LOGIC_VECTOR (3 downto 0));
end FOURBIT_ADDER;

architecture Behavioral of FOURBIT_ADDER is
component FULL_ADDER is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Cin : in STD_LOGIC;
           S : out STD_LOGIC;
           Cout : out STD_LOGIC);
end component;

signal c1,c2,c3,over : std_logic;

begin
A1: FULL_ADDER port map(A => A(0),B => B(0), Cin => '0',S => C(0),C
A2: FULL_ADDER port map(A => A(1),B => B(1), Cin => c1 ,S => C(1),C
A3: FULL_ADDER port map(A => A(2),B => B(2), Cin => c2 ,S => C(2),C
A4: FULL_ADDER port map(A => A(3),B => B(3), Cin => c3 ,S => C(3),C

end Behavioral;
```

Component Declartion

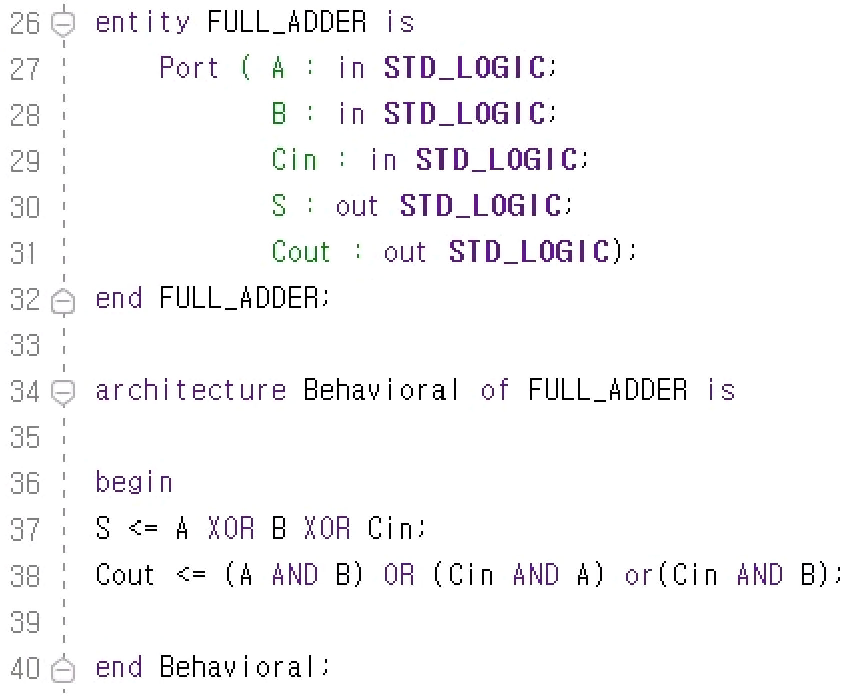
Component [name] is
    port(…);
End component;

Usage

[module name] : [Component_name] port map
    (component_port => [port or signal]);

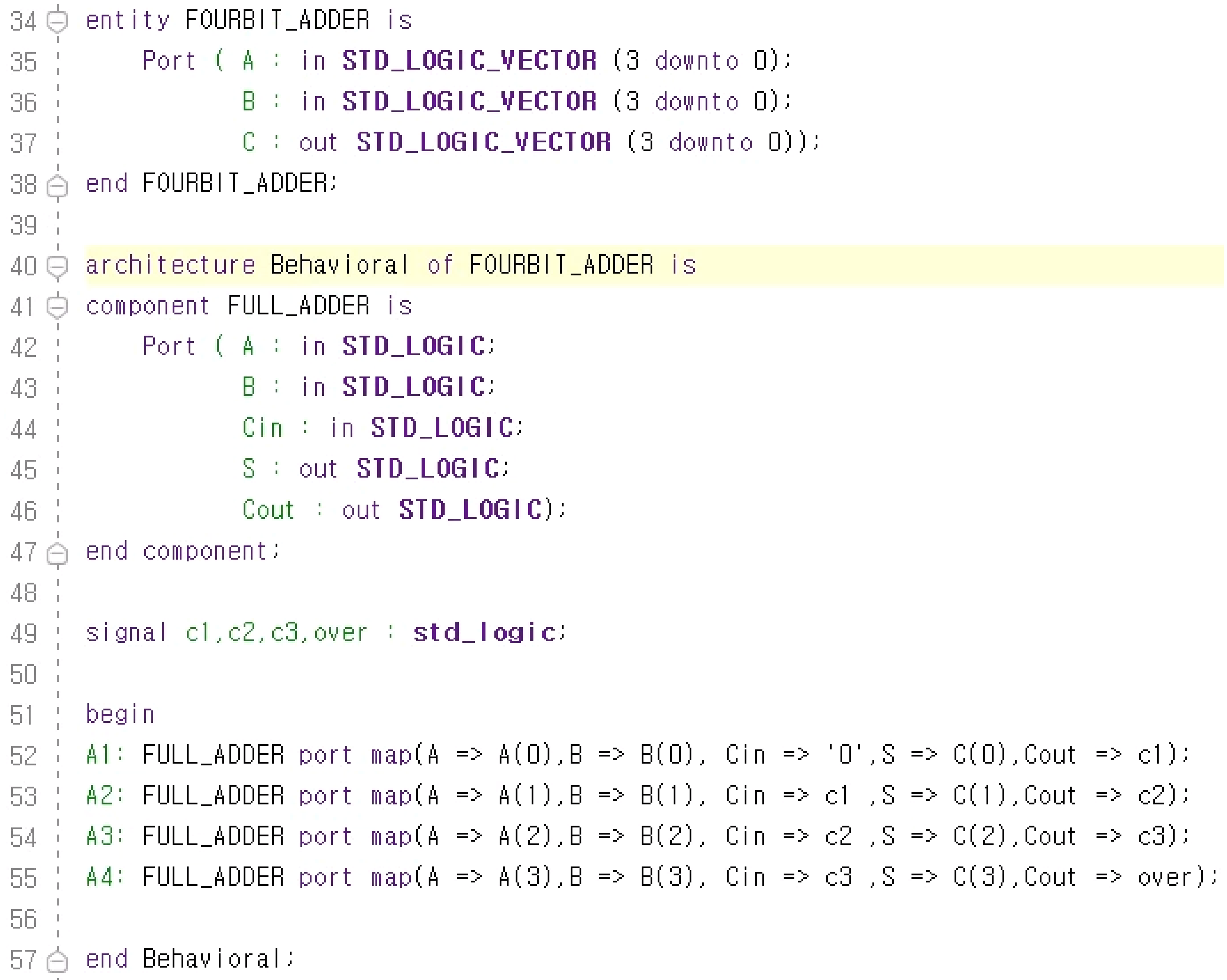
4-Bit Ripple Adder example

Full Adder (**Data flow**)



```
entity FULL_ADDER is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Cin : in STD_LOGIC;
           S : out STD_LOGIC;
           Cout : out STD_LOGIC);
end FULL_ADDER;

architecture Behavioral of FULL_ADDER is

begin
S <= A XOR B XOR Cin;
Cout <= (A AND B) OR (Cin AND A) or(Cin AND B);

end Behavioral;
```

4bit Adder (**Structural**)

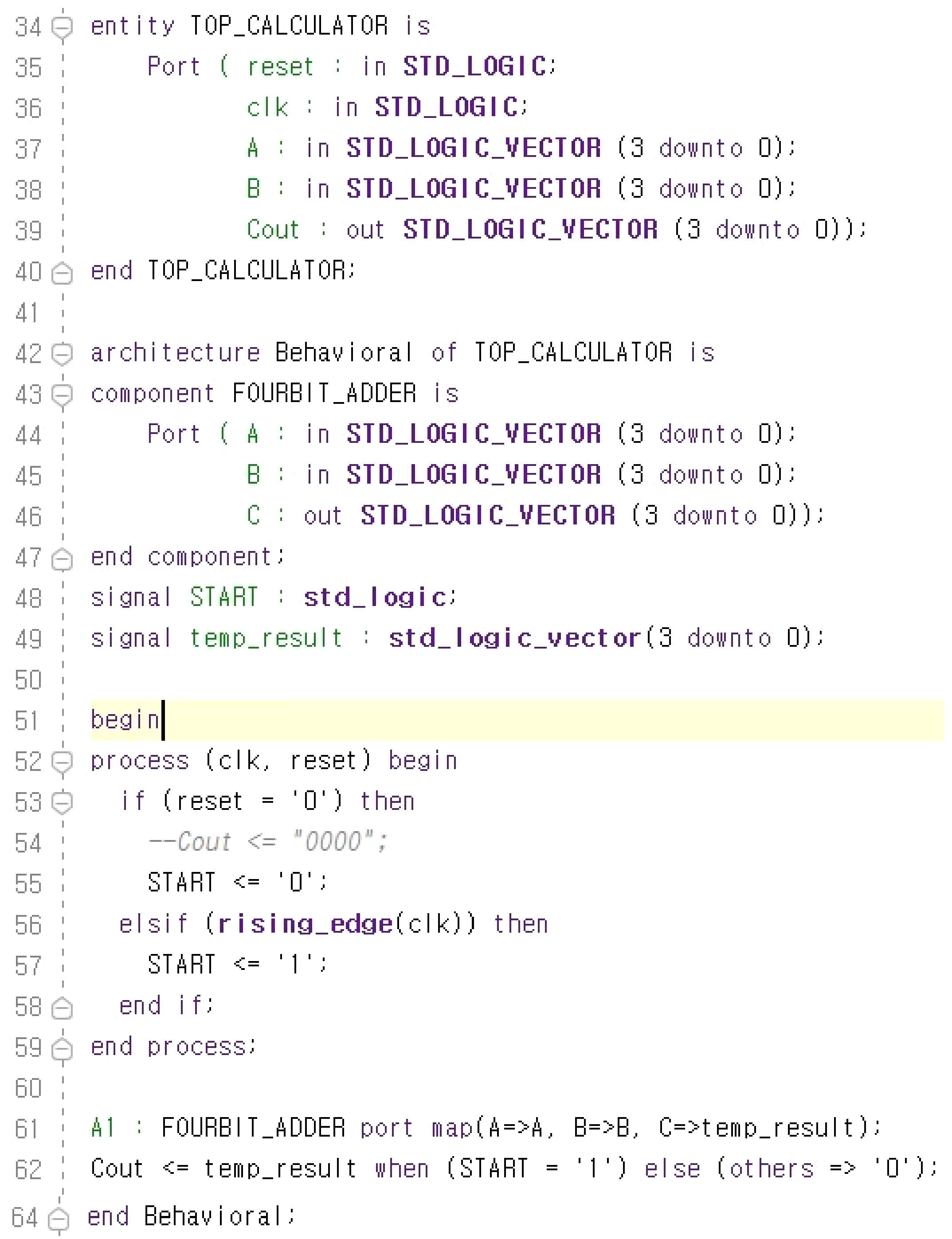


```
entity FOURBIT_ADDER is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           C : out STD_LOGIC_VECTOR (3 downto 0));
end FOURBIT_ADDER;

architecture Behavioral of FOURBIT_ADDER is
component FULL_ADDER is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Cin : in STD_LOGIC;
           S : out STD_LOGIC;
           Cout : out STD_LOGIC);
end component;

signal c1,c2,c3,over : std_logic;

begin
A1: FULL_ADDER port map(A => A(0),B => B(0), Cin => '0',S => C(0),Cout => c1);
A2: FULL_ADDER port map(A => A(1),B => B(1), Cin => c1 ,S => C(1),Cout => c2);
A3: FULL_ADDER port map(A => A(2),B => B(2), Cin => c2 ,S => C(2),Cout => c3);
A4: FULL_ADDER port map(A => A(3),B => B(3), Cin => c3 ,S => C(3),Cout => over);

end Behavioral;
```

Top Calculator(**Sequential**) - controller

➡ Reset, Clk

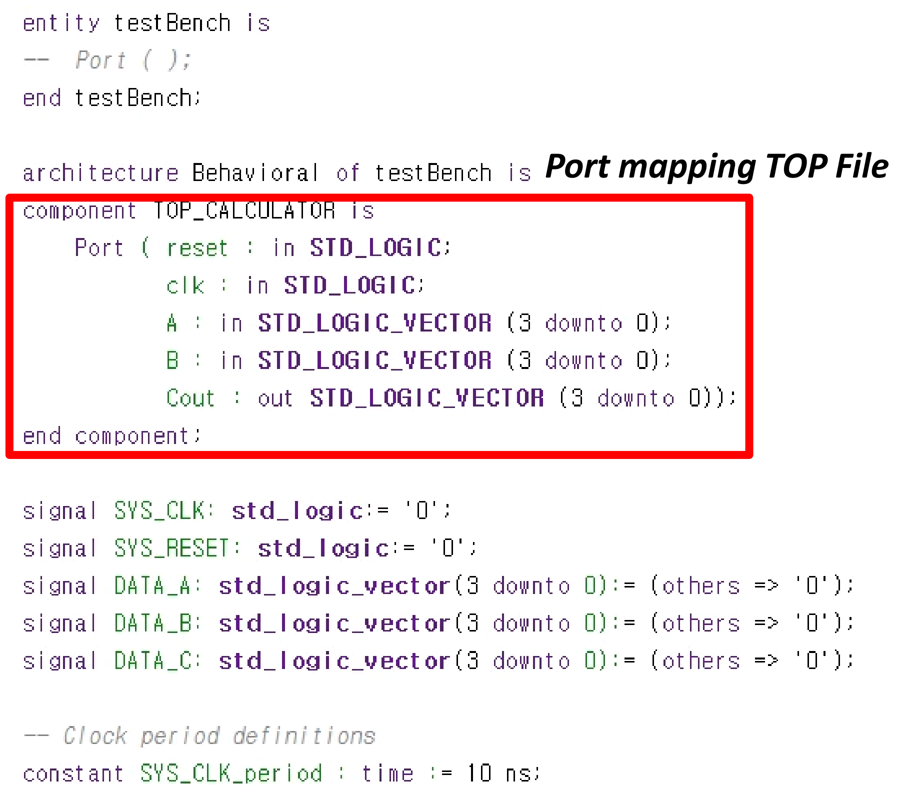
4-Bit Ripple Adder example



```
entity TOP_CALCULATOR is
    Port ( reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Cout : out STD_LOGIC_VECTOR (3 downto 0));
end TOP_CALCULATOR;

architecture Behavioral of TOP_CALCULATOR is
component FOURBIT_ADDER is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           C : out STD_LOGIC_VECTOR (3 downto 0));
end component;
signal START : std_logic;
signal temp_result : std_logic_vector(3 downto 0);

begin
process (clk, reset) begin
  if (reset = '0') then
    --Cout <= "0000";
    START <= '0';
  elsif (rising_edge(clk)) then
    START <= '1';
  end if;
end process;

A1 : FOURBIT_ADDER port map(A=>A, B=>B, C=>temp_result);
Cout <= temp_result when (START = '1') else (others => '0');
end Behavioral;
```

Simulation – Test Bench

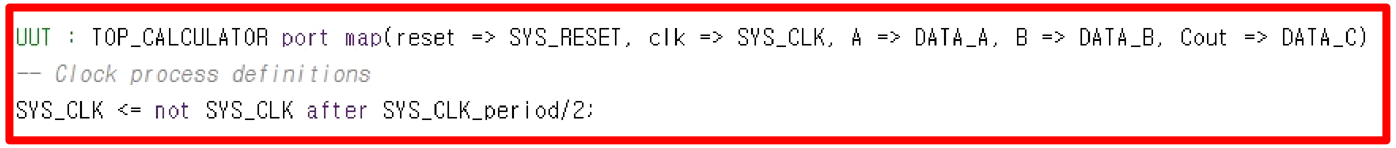
Create Test Bench file → Port map the TOP File



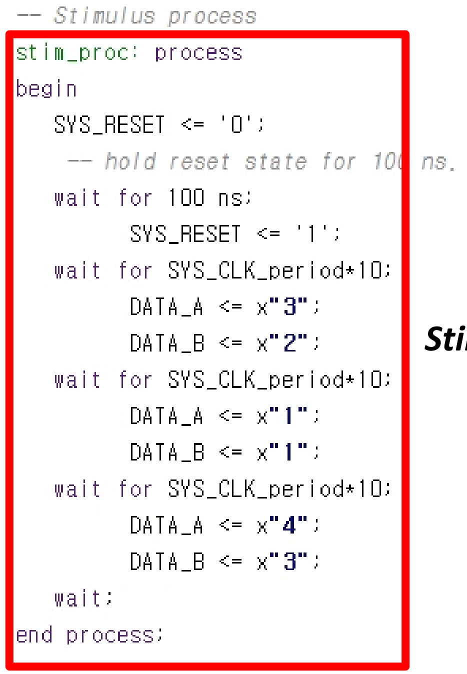
```
entity testBench is
--  Port ( );
end testBench;

architecture Behavioral of testBench is
component TOP_CALCULATOR is
    Port ( reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Cout : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal SYS_CLK: std_logic:= '0';
signal SYS_RESET: std_logic:= '0';
signal DATA_A: std_logic_vector(3 downto 0):= (others => '0');
signal DATA_B: std_logic_vector(3 downto 0):= (others => '0');
signal DATA_C: std_logic_vector(3 downto 0):= (others => '0');

-- Clock period definitions
constant SYS_CLK_period : time := 10 ns;
```

Port mapping TOP File

Clock Generation



```
UUT : TOP_CALCULATOR port map(reset => SYS_RESET, clk => SYS_CLK, A => DATA_A, B => DATA_B, Cout => DATA_C)
-- Clock process definitions
SYS_CLK <= not SYS_CLK after SYS_CLK_period/2;
```

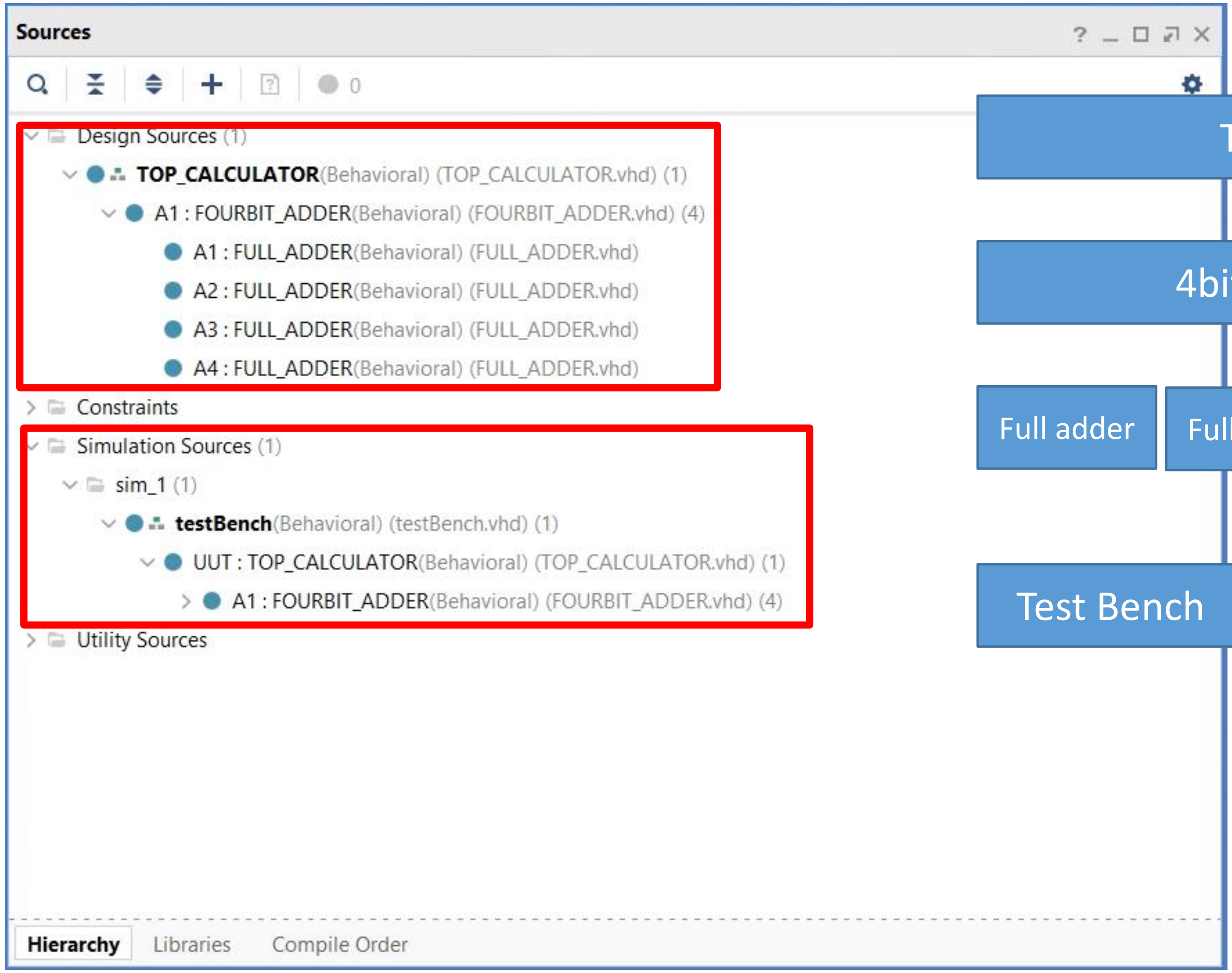


```
-- Stimulus process
stim_proc: process
begin
   SYS_RESET <= '0';
    -- hold reset state for 100 ns.
   wait for 100 ns;
         SYS_RESET <= '1';
   wait for SYS_CLK_period*10;
         DATA_A <= x"3";
         DATA_B <= x"2";
   wait for SYS_CLK_period*10;
         DATA_A <= x"1";
         DATA_B <= x"1";
   wait for SYS_CLK_period*10;
         DATA_A <= x"4";
         DATA_B <= x"3";
   wait;
end process;
```

Stimulus Part

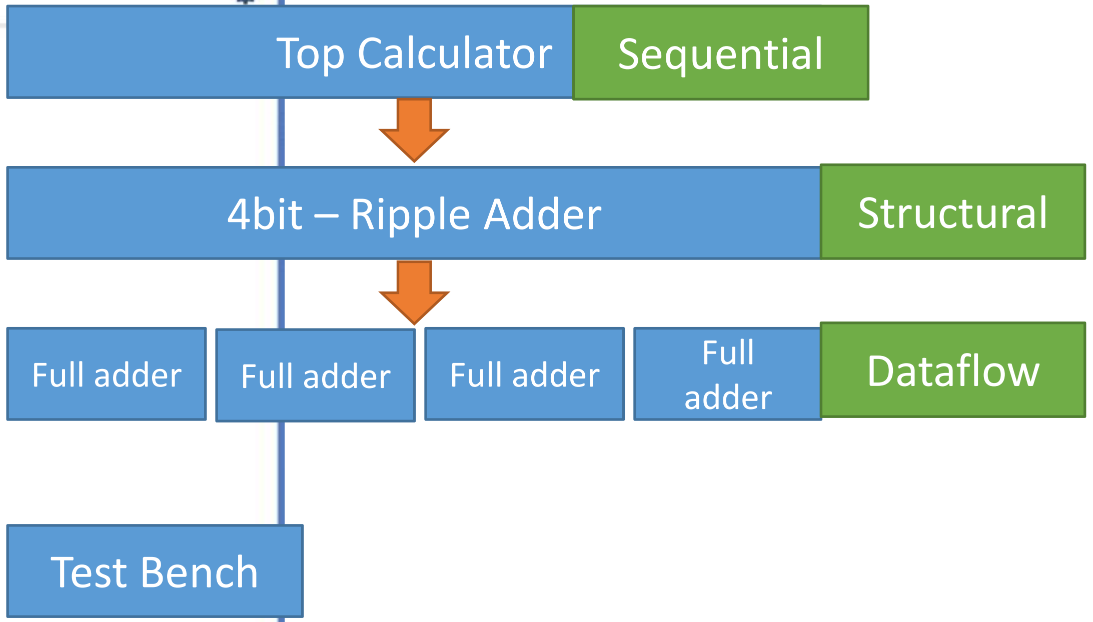
Contents will be here
Case 1) 3 + 2
Case 2) 1 + 1
Case 3) 4 + 3

Simulation – Test Bench



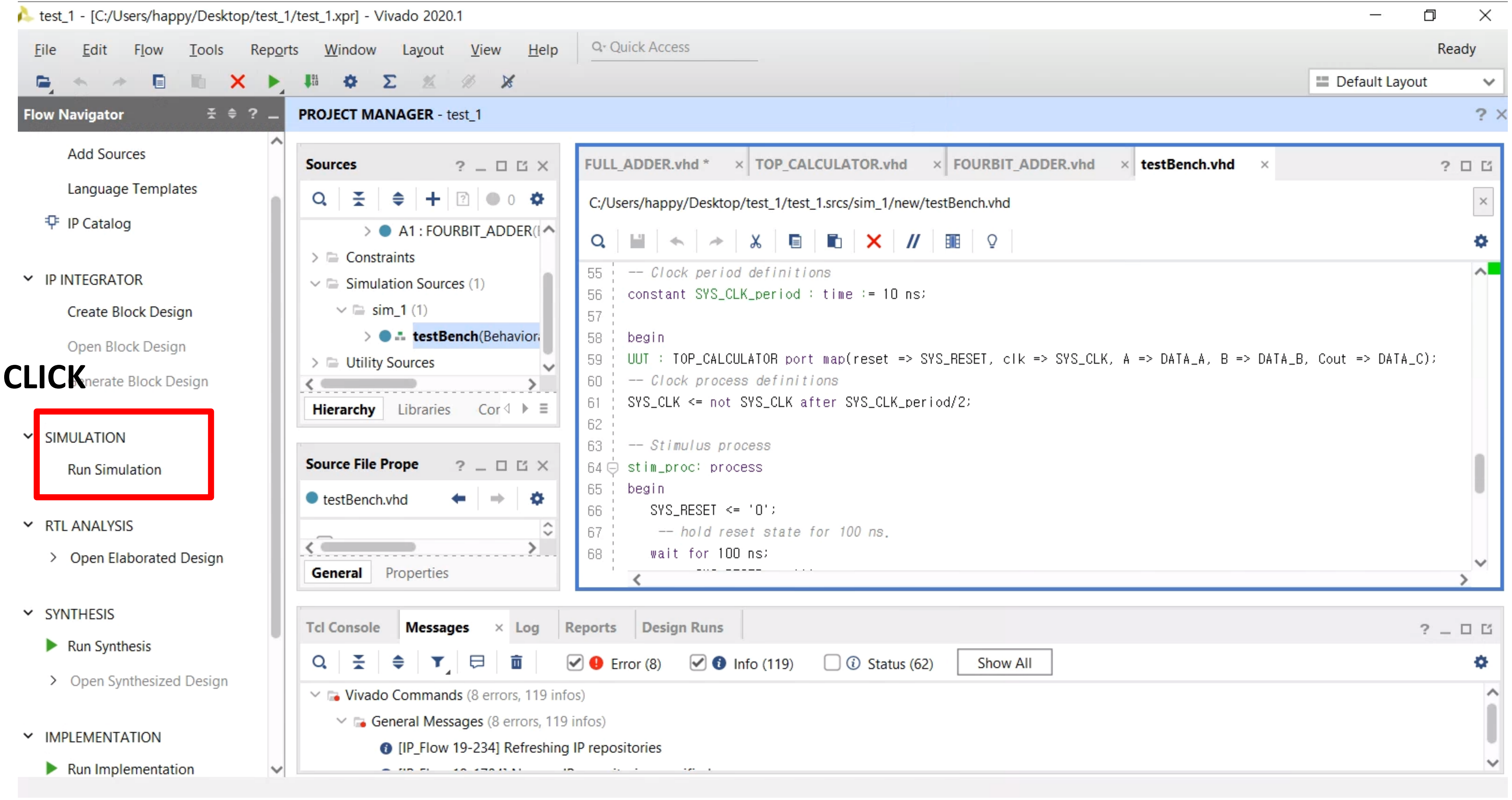
Sources

- Design Sources (1)
  - TOP_CALCULATOR(Behavioral) (TOP_CALCULATOR.vhd) (1)
    - A1 : FOURBIT_ADDER(Behavioral) (FOURBIT_ADDER.vhd) (4)
      - A1 : FULL_ADDER(Behavioral) (FULL_ADDER.vhd)
      - A2 : FULL_ADDER(Behavioral) (FULL_ADDER.vhd)
      - A3 : FULL_ADDER(Behavioral) (FULL_ADDER.vhd)
      - A4 : FULL_ADDER(Behavioral) (FULL_ADDER.vhd)
- Constraints
- Simulation Sources (1)
  - sim_1 (1)
    - testBench(Behavioral) (testBench.vhd) (1)
      - UUT : TOP_CALCULATOR(Behavioral) (TOP_CALCULATOR.vhd) (1)
        - A1 : FOURBIT_ADDER(Behavioral) (FOURBIT_ADDER.vhd) (4)
- Utility Sources



Simulation – Test Bench

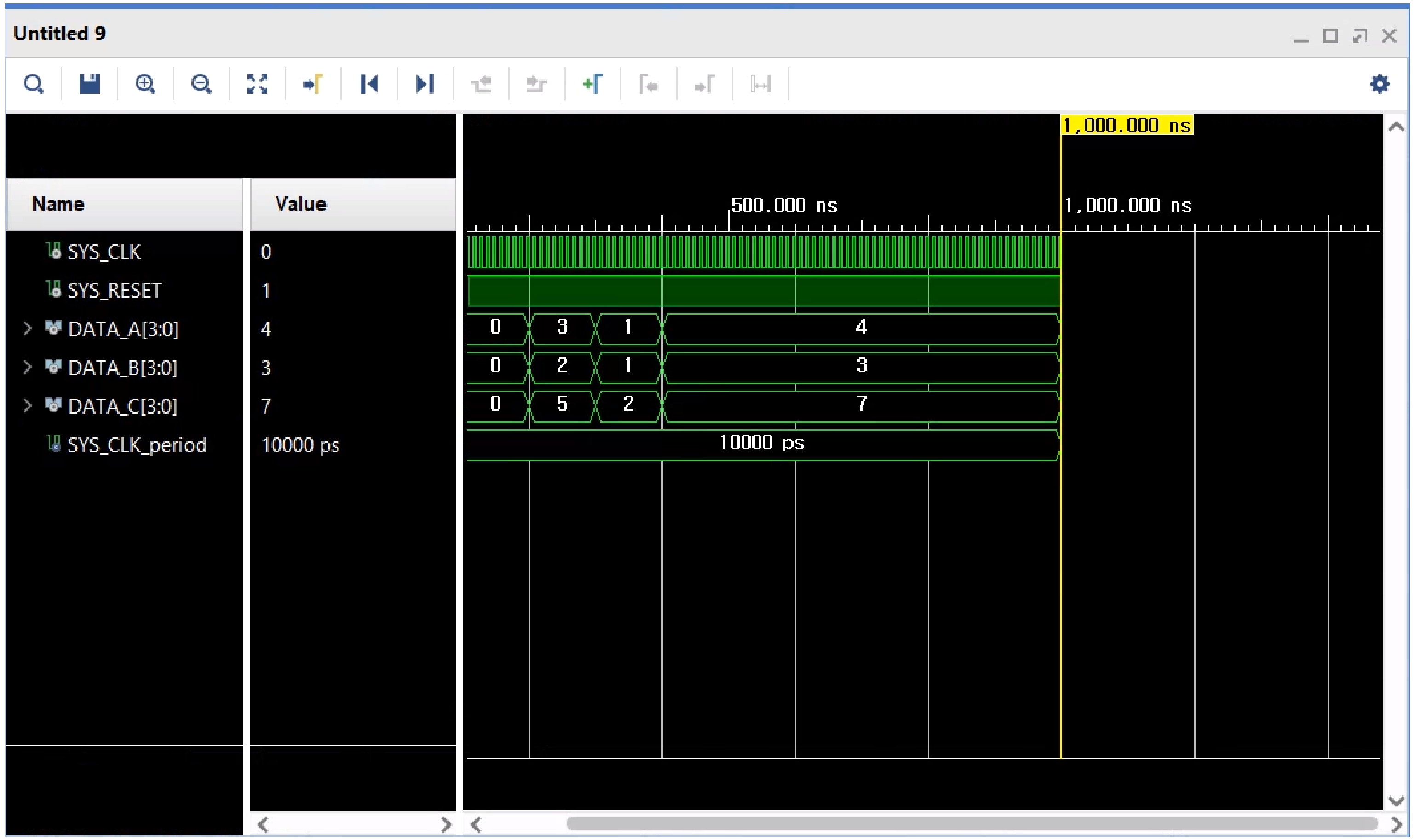
JUST CLICK



```
-- Clock period definitions
constant SYS_CLK_period : time := 10 ns;

begin
UUT : TOP_CALCULATOR port map(reset => SYS_RESET, clk => SYS_CLK, A => DATA_A, B => DATA_B, Cout => DATA_C);
-- Clock process definitions
SYS_CLK <= not SYS_CLK after SYS_CLK_period/2;

-- Stimulus process
stim_proc: process
begin
   SYS_RESET <= '0';
    -- hold reset state for 100 ns.
   wait for 100 ns;
```

```
Vivado Commands (8 errors, 119 infos)
  General Messages (8 errors, 119 infos)
    [IP_Flow 19-234] Refreshing IP repositories
```

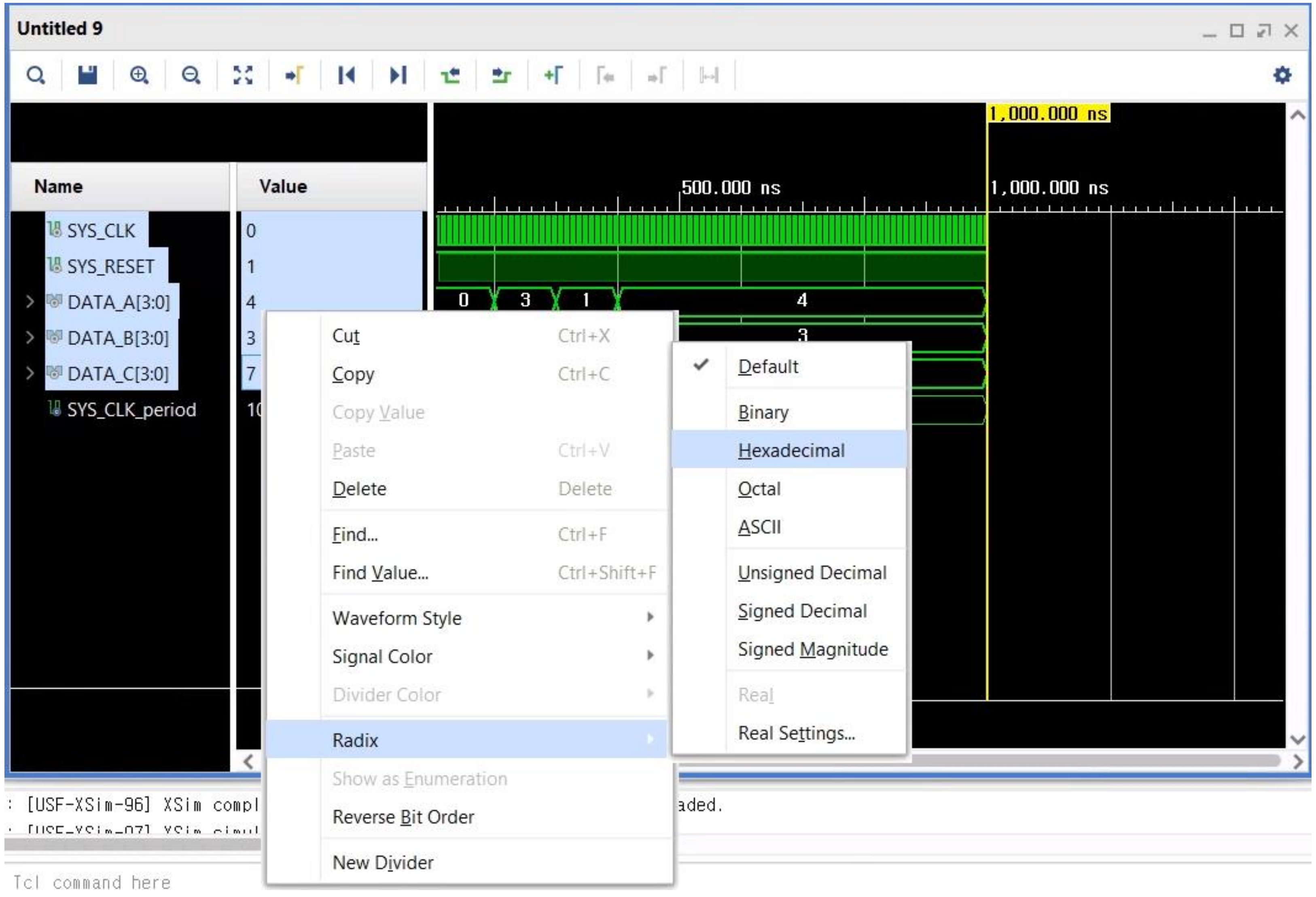
Simulation – Test Bench



Name	Value
SYS_CLK	0
SYS_RESET	1
DATA_A[3:0]	4
DATA_B[3:0]	3
DATA_C[3:0]	7
SYS_CLK_period	10000 ps

```
DATA_A[3:0]     0 | 3 | 1 | 4
DATA_B[3:0]     0 | 2 | 1 | 3
DATA_C[3:0]     0 | 5 | 2 | 7
SYS_CLK_period  10000 ps
```

Simulation – Test Bench



```
: [USF-XSim-96] XSim compl                          aded.
```

YOU CAN SEE
more comfortable

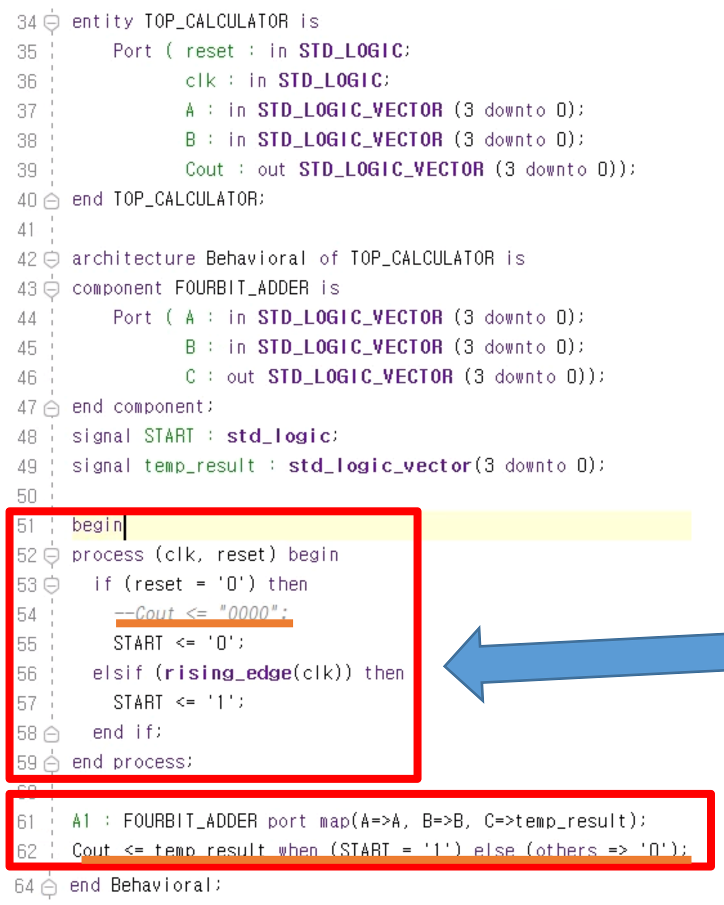
Radix,
Divider,
Waveform Style,
Color

Caution

The primary concurrent statement in VHDL is a process statement.

A number of processes may run at the same simulated time.

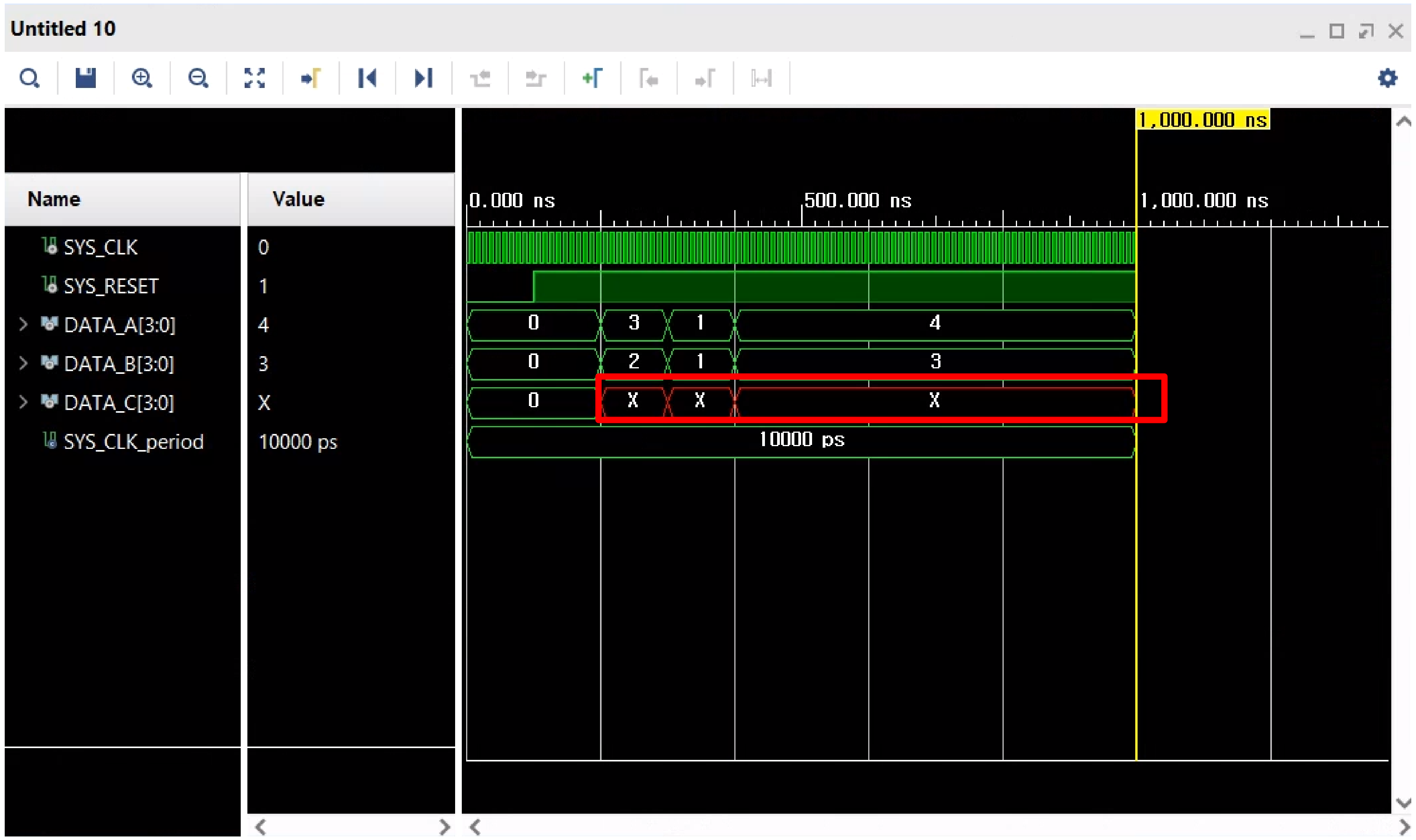
It is not C, C++



```
entity TOP_CALCULATOR is
    Port ( reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Cout : out STD_LOGIC_VECTOR (3 downto 0));
end TOP_CALCULATOR;

architecture Behavioral of TOP_CALCULATOR is
component FOURBIT_ADDER is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           C : out STD_LOGIC_VECTOR (3 downto 0));
end component;
signal START : std_logic;
signal temp_result : std_logic_vector(3 downto 0);

begin
process (clk, reset) begin
  if (reset = '0') then
    --Cout <= "0000";
    START <= '0';
  elsif (rising_edge(clk)) then
    START <= '1';
  end if;
end process;

A1 : FOURBIT_ADDER port map(A=>A, B=>B, C=>temp_result);
Cout <= temp_result when (START = '1') else (others => '0');

end Behavioral;
```

Run at the same time …

Caution



Untitled 10

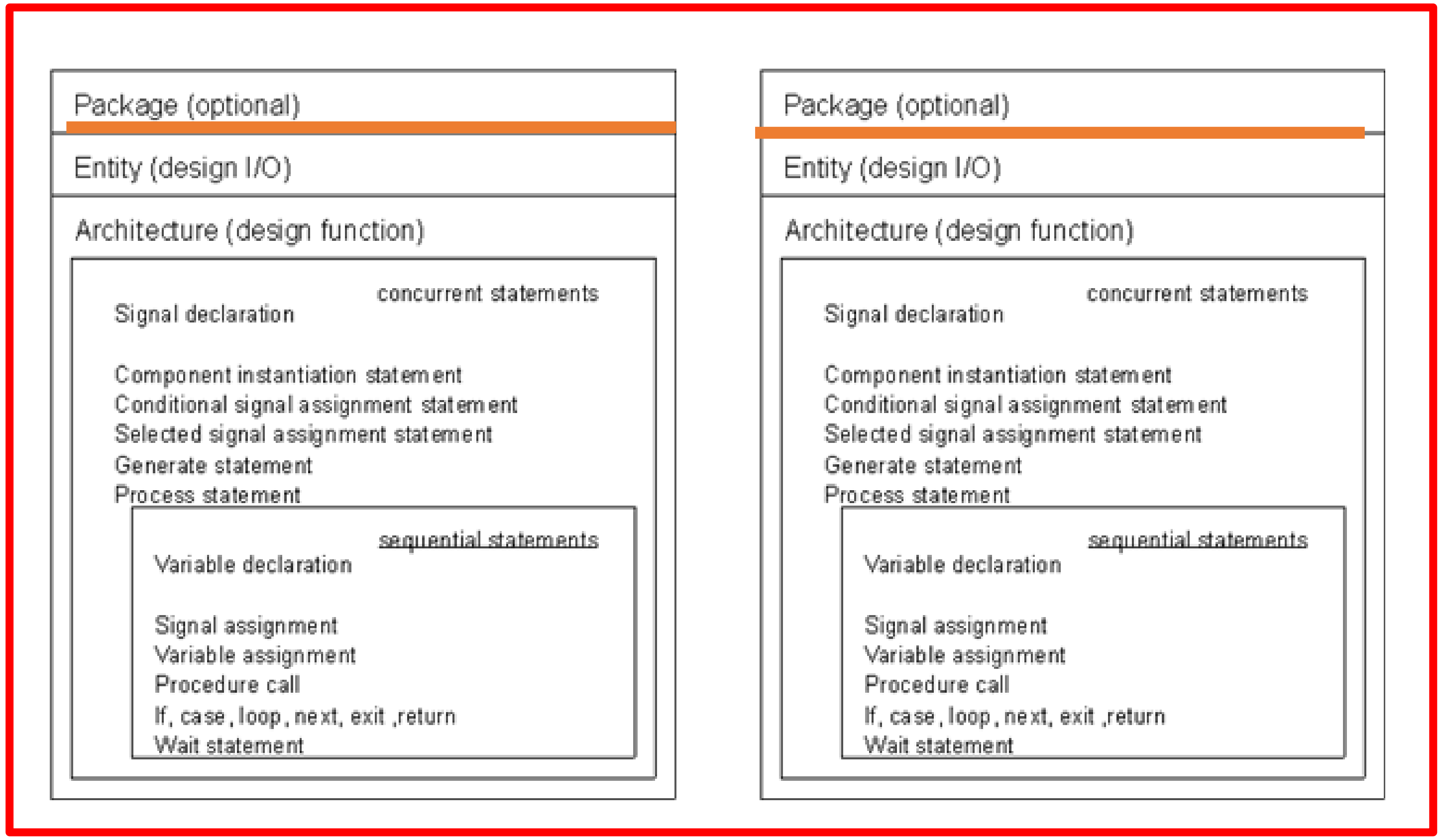
Name	Value
SYS_CLK	0
SYS_RESET	1
DATA_A[3:0]	4
DATA_B[3:0]	3
DATA_C[3:0]	X
SYS_CLK_period	10000 ps

Caution

In Same VHD File, Each architecture must be newly defined, **including libraries.**

It is not C, C++

XXX.vhd



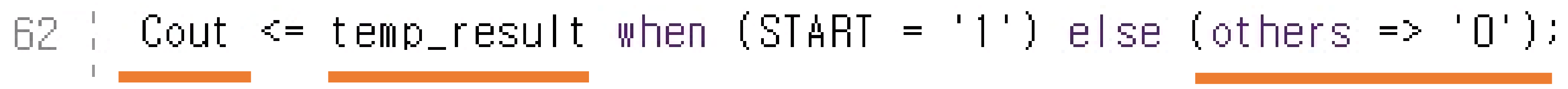
Caution

Operator Size must be same

XXX.vhd

It is not C, C++

Others → automatically replicate bits
ex) if Cout → 4bit , “0000” is assigned



```
Cout <= temp_result when (START = '1') else (others => '0');
```

→ **SAME SIZE**

END …