

# NEAR EAST UNIVERSITY

# FACULTY OF ENGINEERING

# DEPARTMENT OF COMPUTER ENGINEERING

COM-473 HARDWARE DESIGN PROJECT

# /w Load, CE And Async Active High Reset

ÖMÜR KUZUCU 20031116



# LIBRARY LIBRARY LIBRARY

#### 1) Create a Countupd Project:

-Select File -> New Project

-Type countupd in the project name field.

-Enter to location and a countupd subdirectory is created automatically.

-Verify that HDL is selected from top-level source type list.

-Click Next to move to the device properties page.

- Fill in the properties in the table as shown below:

- Product Category: All
- Family: Spartan3
- Device: XC3S200
- Package: FT256
- Speed Grade: -4
- Top-Level Source Type: HDL
- Synthesis Tool: XST (VHDL/Verilog)
- Simulator: ISE Simulator (VHDL/Verilog)
- Preferred Language: Verilog (or VHDL)
- Verify that Enable Enhanced Design Summary is selected.

-Click **Next** to proceed to the Create New Source window in the New Project Wizard. At the end of the next section, your new project will be complete.

#### 2) Create the HDL Source of the Countupd:

-Click the new source button in the new project wizard.

-Select VHDL module as the source type.

-Type in the file name countupd.

-Verify that the add to project checkbox is selected.

-Declare the ports for the countupd design by filling in the port information as shown below:

Port Name	Direction	Bus	MSB	LSB	
clock	in				
reset	in				
clocken	in				
loaden	in				
direction	in				
inp	in			3	0
count	out			3	0

#### 3) HDL Language Template:

-Edit -> Language Templates

-VHDL -> Synthesis Constructs -> Coding Examples -> Binary ->

Up/Down Counters -> /w Load, CE and Async Active High Reset

```
process (<clock>, <reset>)
begin
      if <reset>='1' then
        <count> <= (others => '0');
      elsif <clock>='1' and <clock>'event then
              if <clock enable>='1' then
                     if <load enable>='1' then
                            <count> <= <input>;
                     else
                            if <count direction>='1' then
                             <count> <= <count> + 1;
                            else
                             <count> <= <count> - 1;
                            end if:
                      end if;
              end if;
```

end if; end process;

-With -> /w Load, CE and Async Active High Reset selected, select Edit > Use in File, or select the Use Template in File toolbar button. This step copies the template into the countupd source file.

#### 4) Edit the HDL Code:

library IEEE; use IEEE.STD\_LOGIC\_1164.ALL; use IEEE.STD\_LOGIC\_ARITH.ALL; use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

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

entity countupd is Port ( clock : in STD\_LOGIC; reset : in STD\_LOGIC; clocken : in STD\_LOGIC; loaden : in STD\_LOGIC; direction : in STD\_LOGIC; inp : in STD\_LOGIC\_VECTOR (3 downto 0); count : out STD\_LOGIC\_VECTOR (3 downto 0)); end countupd;

architecture Behavioral of countupd is signal input\_int : std\_logic\_vector(3 downto 0) := "0000"; signal count\_int : std\_logic\_vector(3 downto 0) := "0000"; begin

```
-- Usage of Asynchronous resets may negatively impact FPGA resources
-- and timing. In general faster and smaller FPGA designs will
-- result from not using Asynchronous Resets. Please refer to
-- the Synthesis and Simulation Design Guide for more information.
process (clock, reset)
begin
if reset='1' then
   count int \leq  (others \geq 0');
  elsif clock='1' and clock'event then
   if clocken='1' then
     if loaden='1' then
       count int <= input int;
     else
       if direction='1' then
         count int \leq count int +1;
       else
        count int \leq count int - 1;
      end if;
     end if;
   end if;
  end if;
end process;
count <= count int;
```

end Behavioral;

#### 5) Syntax Check:

-Verify that Synthesis/Implementation is selected from the drop-drown list in the sources window.

-Click the '+' next to the Synthesize-XST process to expand the process group.

-Double-click the Check Syntax process.

#### 6) create test banch

Project → add new source → test banch Edit → language temples → VHDL → Clock simulations And arrange like below. LIBRARY ieee; USE ieee.std\_logic\_1164.ALL; USE ieee.std\_logic\_unsigned.all; USE ieee.numeric\_std.ALL;

ENTITY countupd\_tb\_vhd IS END countupd\_tb\_vhd;

3

#### ARCHITECTURE behavior OF countupd\_tb\_vhd IS

-- Component Declaration for the Unit Under Test (UUT) COMPONENT countupd PORT( clock : IN std\_logic; reset : IN std\_logic; clocken : IN std\_logic; loaden : IN std\_logic; direction : IN std\_logic; inp : IN std\_logic\_vector(3 downto 0); count : OUT std\_logic\_vector(3 downto 0) ); END COMPONENT;

--Inputs

SIGNAL clock : std\_logic := '0'; SIGNAL reset : std\_logic := '1'; SIGNAL clocken : std\_logic := '0'; SIGNAL loaden : std\_logic := '0'; SIGNAL direction : std\_logic := '0'; SIGNAL inp : std\_logic vector(3 downto 0) := (others=>'0');

--Outputs SIGNAL count : std logic vector(3 downto 0);

#### BEGIN

```
-- Instantiate the Unit Under Test (UUT)
uut: countupd PORT MAP(
clock => clock,
reset => reset,
clocken => clocken,
loaden => loaden,
direction => direction,
inp => inp,
count => count
```

);

-- Please ensure that the constant PERIOD is defined prior to the

-- begin statement in the architecture. Refer to the PERIOD Constant Template -- for more info.

process begin clock <= '0'; wait for 5 ns; clock <= '1'; wait for 5 ns; end process;

#### tb: PROCESS BEGIN

-- Wait 100 ns for global reset to finish wait for 100 ns;

-- Place stimulus here inp<="0101"; reset  $\leq ='0';$ clocken<='1'; loaden<='1'; wait for 100 ns; loaden<='0'; direction<='1'; wait for 100 ns; loaden<='0'; direction<='0'; wait; -- will wait forever S;

#### END;

1

1

#### 6) Creating a Verilog Source

Create the top-level Verilog source file for the project as follows:

- Click New Source in the New Project dialog box.

- Select Verilog Module as the source type in the New Source dialog box.

- Type in the file name countupd.

- Verify that the Add to Project checkbox is selected.

- Click Next.

- Declare the ports for the countupd design by filling in the port information as shown below:

Port Name	Direction	Bus	MSB		LSB	
clock	in					
reset	in					
clocken	in					
loaden	in					
direction	in					
inp	in			3		0
count	out			3		0

-Click Next, then Finish in the New Source Information dialog box to complete the new

source file template.

#### 7) Using Language Templates (Verilog)

- Place the cursor on the line below the output [3:0] COUNT\_OUT; statement.
- Open the Language Templates by selecting Edit 
  Language Templates...
- Using the "+" symbol, browse to the following code example:
  - Verilog -> Synthesis Constructs -> Coding Examples -> Binary -> Up/Down Counters -> /w Load, CE and Async Active High Reset

reg [<upper>:0] <reg\_name>;

```
always @(posedge <clock> or posedge <reset>)
if (<reset>)
  <reg_name> <= 0;
else if (<clock_enable>)
    if (<load_enable>)
        <reg_name> <= <load_signal_or_value>;
    else if (<up_down>)
        <reg_name> <= <reg_name> + 1;
    else
        <reg_name> <= <reg_name> - 1;
```

-With /w Load, CE and Async Active High Reset, select Edit > Use in File, or select the Use Template in File toolbar button. This step copies the template into the countupd source file.

#### 8) Final Editing of the Verilog Source

- To declare and initialize the register that stores the countupd value, modify the declaration statement in the first line of the template as follows: replace: reg [<uper>:0] <reg\_name>; with: reg [3:0] count\_int = 0;

-Add the following line just above the endmodule statement to assign the register value to the output port:
assign COUNT\_OUT = count\_int;
Save the file by selecting File > Save.
When you are finished, the code for the countupd will look like the following:

module countupd(clock, reset, clocken, loaden, direction, inp, count);
input clock;
input reset;
input clocken;
input loaden;
input loaden;
input direction;
input [3:0] inp;
output [3:0] count;

6

// and timing. In general faster and smaller FPGA designs will// result from not using asynchronous resets. Please refer to// the Synthesis and Simulation Design Guide for more information.

```
reg [3:0] count_int = 0;
always @(posedge clock or posedge reset)
if (reset)
    count_int <= 0;
else if (clocken)
    if (loaden)
        count_int <= 0;
else if (direction)
        count_int <= count_int + 1;
else
        count_int <= count_int - 1;
assign COUNT_OUT = count_int;
endmodule
```

-Double-click the Check Syntax process.

#### 9) Create the Test Bench of the Countupd HDL Source:

-Select the countupd HDL file in the sources window.

-Create a new test bench source by selecting project -> new source

-In the new source wizard, select test bench waveform as the source type and type countupd tb in the file name field.

#### 10) Gave Inputs to the Test Bench:

-Clock high time : 20 ns -Clock low time : 20 ns -Input setup time : 10 ns -Offset : 0 ns -Global signals : GSR(FPGA) -Initial length of test bench : 1500 ns -Direction high : 300 ns -Direction low : 900 ns

#### 11) Control the Inputs:

-Verify that behavioral simulation and countupd\_tb are selected in the sources window. -In the process tab, click the "+" to expand the xilinx ISE simulator process and double-click the simulate behavioral model process.

#### 12) Entering Timing Constraint:

7

-Select synthesis/implementation from the drop-down list in the sources window.

-Select the countupd HDL source file.

-Click the "+" sign next to the user constraints processes group and double-click the create timing constraint process.

-Select clock in the clock net name field, then select the period toolbar button or double-click the empty period field to display the clock period dialog box.

-Enter 40 ns in the time field.

-Select the pad to setup toolbar button or double-click the empty pad to setup field to display the pad to setup dialog box.

-Enter 10 ns in the offset field to set the input offset constraint.

-Select the clock to pad toolbar button or double-click the empty clock to pad field to display the clock to pad dialog box.

-Enter 10 ns in the offset field to set the output delay constraint.

#### **13) Implemantation:**

-Select the countupd source file in the sources window.

-Open the design summary by double-clicking the view design summary process in the processes tab.

-Double-click the implement design process in the processes tab.

#### 14) Pin Location Constraint:

-Verify that countupd is selected in the sources window.

-Double-click the assign package pins process found in the user constraints process group.

-Select the package view tab.

-In the design object list window, enter a pin location for each pin in the loc column using the following information:

\*clock input port connects to pin t9

\*cont\_out<0> output port connects to pin k12

\*cont\_out<1> output port connects to pin p14

\*cont\_out<2> output port connects to pin 112

\*cont\_out<3> output port connects to pin n14

\*direction input port connects to pin k13

#### 15) Download Design to the Spartan<sup>™</sup>-3 Demo Board

- Select countupd in the Sources window.

- In the Processes window, click the "+" sign to expand the Generate Programming File processes.

- Double-click the Configure Device (iMPACT) process.

- The Xilinx WebTalk Dialog box may open during this process. Click Decline.

- Select Disable the collection of device usage statistics for this project only and click OK.

- In the Welcome dialog box, select Configure devices using Boundary-Scan (JTAG).

- Verify that Automatically connect to a cable and identify Boundary-Scan chain is selected.

- Click Finish.

1

.

.....

- 5

In the Welcome dialog box, select Configure devices using Boundary-Scan (JTAG).
Verify that Automatically connect to a cable and identify Boundary-Scan chain is

selected. - Click **Finish**.

-Right-click on the xc3s200 device image, and select **Program...** The **Programming Properties** dialog box opens.

- Click **OK** to program the device.

9

# NEAR EAST UNIVERSITY

# FACULTY OF ENGINEERING DEPARTMENT OF COMPUTER ENGINEERING

COM-473 HARDWARE DESIGN PROJECT

> ÖMÜR KUZUCU 20031116

#### 1) Create a Counter Project:

-Select File -> New Project

-Type counter in the project name field.

-Enter to location and a counter subdirectory is created automatically.

-Verify that HDL is selected from top-level source type list.

#### 2) Create the HDL Source of the Counter:

-Click the new source button in the new project wizard.

-Select VHDL module as the source type.

-Type in the file name counter.

-Verify that the add to project checkbox is selected.

#### 3) HDL Language Template:

-Edit -> Language Templates

```
-VHDL -> Synthesis Constructs -> Coding Examples -> Binary ->
Up/Down Counters -> Simple Code
```

```
*process (<clock>)
begin
```

if <clock>='1' and <clock>'event then if <count\_direction>='1' then <count> <= <count> + 1; else <count> <= <count> - 1; end if;

end if; end process;

#### 4) Edit the HDL Code:

```
-library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity counter is
    Port (Clock: in STD_LOGIC;
        Direction: in STD_LOGIC;
        Count_Out : out STD_LOGIC_VECTOR (3 down to 0));
end counter;
architecture Behavioral of counter is
signal Count_Int :STD_LOGIC_VECTOR (3 downto 0) := "0000";
begin
    process (Clock)
begin
```

#### 5) Syntax Check:

-Verify that Synthesis/Implementation is selected from the drop-drown list in the sources window.

-Click the '+' next to the Synthesize-XST process to expand the process group. -Double-click the Check Syntax process.

#### 6) Create the Test Bench of the Counter HDL Source:

-Select the counter HDL file in the sources window.

-Create a new test bench source by selecting project -> new source

-In the new source wizard, select test bench waveform as the source type and type counter tb in the file name field.

#### 7) Gave Inputs to the Test Bench:

-Clock high time : 20 ns
-Clock low time : 20 ns
-Input setup time : 10 ns
-Offset : 0 ns
-Global signals : GSR(FPGA)
-Initial length of test bench : 1500 ns
-Direction high : 300 ns
-Direction low : 900 ns

#### 8) Control the Inputs:

-Verify that behavioral simulation and counter\_tb are selected in the sources window. -In the process tab, click the "+" to expand the xilinx ISE simulator process and double-click the simulate behavioral model process.

#### 9) Entering Timing Constraint:

-Select synthesis/implementation from the drop-down list in the sources window. -Select the counter HDL source file.

-Click the "+" sign next to the user constraints processes group and double-click the

create timing constraint process.

-Select clock in the clock net name field, then select the period toolbar button or double-click the empty period field to display the clock period dialog box.

-Enter 40 ns in the time field.

-Select the pad to setup toolbar button or double-click the empty pad to setup field to display the pad to setup dialog box.

LIBRARY

N. TAN

-Enter 10 ns in the offset field to set the input offset constraint.

-Select the clock to pad toolbar button or double-click the empty clock to pad field to display the clock to pad dialog box.

-Enter 10 ns in the offset field to set the output delay constraint.

#### 10) Implemantation:

-Select the counter source file in the sources window.

-Open the design summary by double-clicking the view design summary process in the processes tab.

-Double-click the implement design process in the processes tab.

#### **11) Pin Location Constraint:**

-Verify that counter is selected in the sources window.

-Double-click the assign package pins process found in the user constraints process

group.

-Select the package view tab.

-In the design object list window, enter a pin location for each pin in the loc column using the following information:

\*clock input port connects to pin t9

\*cont\_out<0> output port connects to pin k12

\*cont\_out<1> output port connects to pin p14

\*cont\_out<2> output port connects to pin 112

\*cont\_out<3> output port connects to pin n14

\*direction input port connects to pin k13

# FULLADDER

# DESING STEPS

DESING DESCRIPTION TOOLS USED CREATE THE NEW PROJECT FULLADDER CREATE AN HDL HALFADD CREATE TEST BENCH HALFADD CREATE AN HDL ORGATE CREATE TEST BENCH ORGATE CREATE TEST BENCH ORGATE CREATE TEST BENCH FULLADD SIMILATION

### DESING DESCRIPTION

This project name FULLADDER. Entity FULLADD is Port (A: in STD\_LOGIC; B: in STD\_LOGIC; cin: in STD\_LOGIC; sum: out STD\_LOGIC; carry: out STD\_LOGIC);

Entity halfadd is Port{ A: in STD\_LOGIC B: in STD\_LOGIC Sum: out STD\_LOGIC carry: out STD\_LOGIC

Entity orgate is Port{ a: in STD\_LOGIC b: in STD\_LOGIC z: out STD\_LOGIC The project family name Spartan3

Dvice XC35200

Package FT256

Top-lwvwl source type HDL

# TOOLS USED

HARDWARE

Family name Spartan3

Dvice XC3S200

Package FT256

Speed -4

SOFTWARE

XILINX-ISE 9.1 i

Top Level source typeVHDLSynthesis toolXST (VHDL/verilog)SimilatorISE Similator (VHDL/verilog)LanguageVHDL

## 1. Design Steps Of FULLADDER

File	Edit View Project	t Source Process Window Help	. *
	New Project Open Project Open Example Close Project Save Project As	©x ¤⊘ ₽₽₽ tigoie = = = = =	×≣× ▼
B	New Ctrl Open Ctrl Close	'I+N 'I+O	
5	Save Ctri Save As Save All	1+5 Libraries	×
	Print Preview Print Ctrl	1+P	
••••••••••••••••••••••••••••••••••••••	Recent Projects		

Enter a Name and Location for the Proj	ject
Project Name:	Project Location
fulladd	D:\WebPACK_SFD_91i\xilinx\myprojects\fulladd
Select the Type of Top-Level Source for	or the Project
Top-Level Source Type: HDL	×

Property Name	Value	
Product Category	All	~
Family	Spartan3	~
Device	×C3S200	<b>v</b>
Package	FT 256	<ul><li>✓</li></ul>
Speed	-4	
Top-Level Source Type	HDL	
Synthesis Tool	XST (VHDL/Verilog)	×
Simulator	ISE Simulator (VHDL/Verilog)	<b>v</b>
Preferred Language	VHDL	<b>\</b>
Enable Enhanced Design Summary		
Enable Message Filtering		
Display Incremental Messages		

Before create a fulladder in VHDL file we create firsr two halfadder and orgate.First we create the halfadder for that we declare the inputs .

#### Decleration of the Halfadder

File Edit View P	roject Source Process Win	ndow Help
Sources for: Syr	<ul> <li>New Source</li> <li>Add Source</li> <li>Add Copy of Source</li> <li>Cleanup Project Files</li> <li>Toggle Paths</li> <li>Archive</li> <li>Take Snapshot</li> <li>Make Snapshot Current</li> </ul>	★ A Q Q I < O < O  × •
ाद्व Sources	Apply Project Properties Source Control	•
Processes		×
Processes for: xc3s	200-4ft256	
Add Exis Create N Design L	ting Source ew Source Itilities	

<ul> <li>New Source Wizard - Select Source Ty</li> <li>BMM File</li> <li>IP (Coregen &amp; Architecture Wizard)</li> <li>MEM File</li> <li>Schematic</li> <li>Implementation Constraints File</li> <li>State Diagram</li> <li>Test Bench WaveForm</li> <li>User Document</li> <li>Verilog Module</li> <li>Verilog Test Fixture</li> <li>VHDL Module</li> <li>VHDL Library</li> <li>VHDL Package</li> <li>VHDL Test Bench</li> </ul>	Pe         File name:         halfadd         Location:         D:\WebPACK_SFD_91i\xilinx\myprojects\fulladd
More Info	< Back Next > Cancel

Entity Name	halfadd					
Architecture Name	Behavioral					
Port Name	Direction		Bus	MSB	LSB	
а	in	~				
Ь	in	~				
sum	out	<ul> <li>✓</li> </ul>				
carry	out	~				
	in	4				
	in	~				
	in	<b>~</b>				
	in	<b>~</b>				
	in	<ul> <li>•</li> </ul>				
and a second second and the second	in	×				V

## Entity halfadd is

ľ

Port{ A: in STD\_LOGIC

B: in STD\_LOGIC

Sum: out STD\_LOGIC

carry: out STD\_LOGIC

end halfadd;

architecture behaviroal of halfadd is

sum<=A xor B

carry<=A and B

end behaviroal;

In the entity part of the halfadd we declera the inputs of the halfadd. In entity part as we see A,B are the inputs and sum, carry are the output for the first halfadder. All the inputs and the outputs types are STD\_LOGIC. They can take as a value just 1 or 0.

By the architecture part of the halfadder as we see from the code. The ouput value comes from' A xor B' to the **sum** also the output value comes' A and B' to the carry. After cretating halfadder file by double clicking check syntax we compile the page to able to create the test bench file.

#### Decleration of the Halfadder Test Bench File

×	Sources for: Synthesis/Implementation				
			ld	🔄 🗐 fullad	
			200-4ft256	E Sc3s2	
	New Source	f	halfadd	И	
	Add Source Add Copy of Source —				
	Open	ľ	s to	©C\$ Source	
	Set as Top Module			Processes	
ł	Use SmartGuide		or: halfadd	Processes fo	
	New Partition		dd Existing	- Ξ Αι	
	Delete Partition		reate New		
	Partition Properties		iew Desigr	₩ ∑ Vi	
	Partition Force		esign Utilit	🕀 🌮 D	
	Remove		ser Constr ynthesize	⊕ 🌮 U ⊕ 🎝 S!	
	Move to Library		nplement D	ni 🚺 🕀	
	Toggle Paths	*******	enerate Pr	⊕ <b>₹</b> } G	
	Properties	Z	ses	Proces:	

New Source Wizard - Select Source T BMM File File Schematic Implementation Constraints File State Diagram Test Bench WaveForm User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Library P VHDL Package VHDL Test Bench	ype         File name:         halfadd_tb          Location:         D:\WebPACK_SFD_91i\xilinx\myprojects\fulladd
More Info	Add to project  Back Next > Cancel

#### BEGIN

uut: halfadd PORT MAP(

A => A,

B => B,

sum => sum,

carry => carry

); tb : PROCESS

BEGIN

-- Wait 100 ns for global reset to finish

wait for 100 ns;

A <= '0';

B <= '0';

-- Place stimulus here

wait for 100 ns;

A <= '0';

B <= '1';

wait for 100 ns;

A <= '1';

B <= '0';

wait for 100 ns;

A <= '1';

B <= '1';

wait; -- will wait forever

END PROCESS;

END;

At the port map part we just declera again the inputs and the outputs. Where the alues will come from. For the inputs A,B we assign the values to able to see the results in the simulation part. Then by double clicking simulate behaviroal model we can see the results. As shown below:

Decleration of Orgate

Mew Source Wizard - Select Source Ty	pe 📃 🗆 🔀
<ul> <li>BMM File</li> <li>IP (Coregen &amp; Architecture Wizard)</li> <li>MEM File</li> <li>Schematic</li> <li>Implementation Constraints File</li> <li>State Diagram</li> <li>Test Bench WaveForm</li> <li>User Document</li> <li>Verilog Module</li> <li>Verilog Test Fixture</li> <li>VHDL Module</li> <li>VHDL Library</li> <li>VHDL Package</li> <li>VHDL Test Bench</li> </ul>	File name: orgate Location: D:\WebPACK_SFD_91i\xilinx\myprojects\fulladd
More Info	< Back Next > Cancel

Entity Name	orgate	orgate							
rchitecture Name	Behavioral								
Port Name	Direction	adda a daler, ar shale andones e sente atter, and the a fee a s	Bus	MSB	LSB	1			
3	in	~		-					
ס	in	~							
Z	out								
	in	~							
	in	~							
	in	¥			- an prior that the statement of some constitute of the title of the second				
	in	~							
	in	~							
	in	~							
	in	~							

## Entity orgate is

Port{ a: in STD\_LOGIC

b: in STD\_LOGIC

z: out STD\_LOGIC

end orgate;

architecture behaviroal of orgate is

z<= a or b;

.

.

end behaviroal;

In the entity part of orgate we declared the inputs and outputs for the fulladder.In here again A,B are the inputs and the output is z.

In the architecture part of orgate we can see that we get the value of z by 'a or b'. After cretating orgate file by double clicking check syntax we compile the page to able to create the test bench file.

Decleration of the Orgate Test Bench File

IP (Coregen & Architecture Wizard)         Implementation Constraints File         State Diagram         Implementation Constraints File         State Diagram         Implementation Constraints File         State Diagram         Implementation Constraints File         Verilog Module         Verilog Module         Verilog Test Fixture         VHDL Module         VHDL Library         VHDL Package         Implement	File name: orgate_tb Location: D:\WebPACK_SFD_91i\xilinx\myprojects\fulladd
	Add to project

New Source Wizard - Associ	ate Source			
elect a source with which to associa	te the new source.			
alfadd				
rgate				
				****
More Info		< Back	Next >	Cancel

#### BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: orgate PORT MAP(

a => a, b => b,

tb : PROCESS

BEGIN

-- Wait 100 ns for global reset to finish

wait for 100 ns;

α <= '0';

b <= '0';

-- Place stimulus here

wait for 100 ns;

a <= '0'; b <= '1'; wait for 100 ns; a <= '1'; b <= '0'; wait for 100 ns; a <= '1'; b <= '1'; wait; -- will wait forever

END PROCESS;

It is the same with halfadder test bench file again in port map part we defined the inputs and outputs. Then able to see in the simulation we give values for a and b, the output values assigned to the z by a or b.

#### Decleration of Fulladder

le name: ulladd ocation: u:\WebPACK_SFD_91i\xilinx\myprojects\fulladd

Entity Name	fulladd		*****		nana mana ana sa	
architecture Name	Behavioral					
Port Name	Direction		Bus	MSB	LSB	2
3	in	<b>v</b>		-		
6	in	<b>~</b>				
cin	in	<b>v</b>				
sum	out	× (				
carry	out	<b>×</b>				1111 <sup>10</sup>
	in	<b>~</b>			all is a suspension of the second secon	TANKA KANANA MUTUMANA MUTUMANA MUT
· · · · · · · · · · · · · · · · · · ·	in	<b>v</b>			1	
	in	~				
	in	~				
	in	~				

#### entity fulladder is

Port (A: in STD\_LOGIC;

B: in STD\_LOGIC;

cin : in STD\_LOGIC;

sum : out STD\_LOGIC;

carry : out STD\_LOGIC);

end fulladder;

architecture Behavioral of fulladder is

signal I1,I2,I3:STD\_LOGIC;

#### COMPONENT halfadd

PORT(

A : IN std\_logic;

B : IN std\_logic;

sum : OUT std\_logic;

carry : OUT std\_logic

);

#### END COMPONENT;

COMPONENT orgate

PORT( a: IN std\_logic;

b : IN std\_logic;

z : OUT std\_logic

);

END COMPONENT;

begin

u1: halfadd port map (a,b,I1,I2);

u2: halfadd port map (I1,cin,sum,I3);

u3: orgate port map (I3,I2,carry);

end Behavioral;

In fulladder differently we declare signals they are 1,12,13,14. Those signals type also STD\_LOGIC they can get as value just 1or 0. By this part of code

begin

u1: halfadd port map (a,b,I1,I2);

u2: halfadd port map (I1,cin,sum,I3);

u3: orgate port map (I3,I2,carry);

end Behavioral;

We assign the outputs of first halfadder I1,I2, assign for second halfadder first input I1, second output I3, assign the inputs of orgate I3,I4. After finishing those assigments we save the file and double clicking check syntax we compile the file and then we can create the test bench file.

Decleration of Fulladder Test Bench File

Mew Source Wizard - Select Source T	ype
BMM File         IP (Coregen & Architecture Wizard)         MEM File         Schematic         Implementation Constraints File         State Diagram	File name:
n Test Bench WaveForm ■ User Document	fulladd_tb
V Verilog Module	Location:
Werilog Test Fixture WW VHDL Module WHDL Library PVHDL Package WW VHDL Test Bench	D:\WebPACK_SFD_91i\xilinx\myprojects\fulladd
	Add to project
More Info	< Back Next > Cancel

New Source Wizard - Associ	ate Source			
Select a source with which to associa	te the new source.			
fulladder				
halfadd				
orgate				
			<u>ر</u>	r
More Info		< Back	Next >	Cancel

ARCHITECTURE behavior OF fulladder\_tb\_vhd IS

BEGIN

-- Wait 100 ns for global reset to finish

wait for 100 ns;

A <= '0';

B <= '0';

cin <= '0';

-- Place stimulus here

wait for 100 ns;

A <= '0';

B <= '0'

cin <= '1';

wait for 100 ns;

A <= '0';

B <= '1';

cin <= '0';

wait for 100 ns;

A <= '0';

B <= '1';

cin <= '1';

wait for 100 ns;

A <= '1';

B <= '0';

cin <= '0';

wait for 100 ns;

A <= '1';

B <= '0';

cin <= '1';

wait for 100 ns;

A <= '1'; B <= '1'; cin <= '0'; wait for 100 ns; A <= '1'; B <= '1'; cin <= '1'; wait; -- will wait forever END PROCESS;

END;

In the test bench file we assign values for the inputs A,B,Cin for the simulation design as we can see from the pictures in the different values of inputs it gives different outputs.In the first picture all the values are 0 and outputs are zero .In the second picture a=0,b=1 so cin =0,sum=1,carry=0.As we see from the project 1+0=0 with co carry.

Surves for Denarrow Sincheon	New 3000 ns a a b c cn a cum a cum a cum	р 8 9 9					520	۶۶ میریند مراجع مراحه مراجع مراحه مراحم مراحم مراحم مراحم مراحه مراحه مراحه مراحه مراحه مراحه مراحه مراحم مراحه ما مرامع مرامع مرامع مرامع مراما مرامه مرام مم مرام مرما مرما
Elitada Elitada Sources Sourc	90000 hs	5 5 0 0 0						de manuel anno anno an de manuel anno anno an de manuel anno anno anno de manuel anno de ma
Conserved  Subscript John Server Masser Janon  Subscript John Server Masser Janon  Subscript John Server Server Server John Server  Subscript John Server Server John Server  Subscript John Server John Serve	<ul> <li>a</li> <li>a</li> <li>b</li> <li>a</li> <li>cn</li> <li>a</li> <li>cnis</li> <li>a</li> <li>cnis</li> <li>conis</li> </ul>	6 6 0 0						
Control of Science Behaviors functions:     Control of Science Behaviors functions:     Control of Science Behaviors for Scienc	•         •           •         •           •         •           •         •	0 0 0						
<ul> <li>Sources</li> <li>Sources</li></ul>	<ul> <li>Coll</li> <li>turn</li> <li>Coll</li> <li>Coll</li> </ul>	0 0 0						4
Content (b. (rd. server byske (b. ref)      Source     Content     Conten	Carlo	ð ð				1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		ł ,
SSURVE Stapenos Distance ceans exch of Lassim Sty into Malantee Joy not Talaster Sty ind Enfance		0				, <b>, , , , , , , , , , , , , , , , , , </b>	 	4 2
S Source Statement D Laborere Rector of Nutacoly (5 yr no Mulance (5 yr no Nutacoly (5 yr no Nutaco)))								r de la composition de la comp
Sources By Encourton Discourse recent marchy of Klanday (St. (Md. Seferico Dislanday (St. (Md. Seferico								Pilling and a
SSucces Stapenous Cateneses sectory of Alabaser, Jo Units Statemer Jo Units Taladoer, Jo Units Estavior								
ີ້ຊາງສາງສະມີລັ້ນເຊິ່ງກ່າງສຽງສະມີຂຶ້ນເຊິ່ງດີນຊາມ ເຫັດມີ ລາຍການ							all in the	
રક્ષમાં સ્ટાઇલ ઈપ્ટિસ્ટાઇલ ટ્રાઇન્ડ્રે ∎સ્ટાહાલાક્રા ઇંડ પ્રેલી રાહિસ્ટાઇલ ટ્રાઇન્ડ્રેલ ટ્રાઇન્ડ્રિટ ટ્રાઇન્ડ્રેલ ટ્રાઇન્ડ્રિટ ટ્રાઇન્ડ્રિટ ટ્રાઇન્ડ્ર								
narda officialar found i falanara El antar found i faladar found i falaron								
Caracter Da Tale - Strander Da Tale - Cara								
			and the second second	and the second			 	
Effercessee III See Herwithy fuladay 30, 5hd	**************************************							
	II Design Summary	i San Arriantian	1					
This is a lite vecaido dE 25E Simulation.								
Simulator is doiny discust initiality for	887 <b>44</b> 8.							
. Finished curcuit initialization process.								
P								

grunne voje vojekovenske vojelo zvjedi de vojekoven) 1. nav zana ovu Propins Source Pracese TextBends Soude	age Zindan Miliji						
CONG & XOOX OG B I	SXXXX M.		AN 10 4	• 9 10	and state	00 111	
油台 计品件内 国际编制成 1000	• (m. +)						
iavse		100010		322 S			
Currently Demonstration	100%	Ð	2890	60 <b>8</b>	500	800	1360
Children Chi	10.30 123		Elizabeth Andrews	the second s	the second second		
· Malatte 12, 110 - benever filester 15 and		Q	anna an an ann an an an an an an an an a		den anno 1975 - 197		undanungan series
> Nus - futadow - Dehavora Bulado vico	<b>A</b> 0		in the state of the second				*********
and the second the second second second	<b>C</b> 1391						****
<ul> <li>Polase D and parama polase page</li> </ul>						and the second	
	de casi	U	ningen talden Schladeren etter en en en en en en etter etter				1 1 - Y
						18 M 1	
Charles an Snapabola Oliverates					NTU A		
10101							14 - X
Herandry of Fulletone Jo. ( n d	and the second se						1.00
The state of the second second second							
					1.1.1		
	1						5 6 2 2
	1000						A-11-14-5
	1000						
							C ( 1997)
	A COLORADOR	A transfer to a second	the second second	all and the second second		and the second second	A DECK
1 MM -	· · ·	**. **					
Churchenen I I Sun Lewis Con - President St Aug	ST Design Summar	y i 🎬 SeesJadiem					
Tois as a lite version of LBE Structed Simulator is doing siroust initials	tor: atlof EXoceas.						
Ficialed circuit initialization prov							
							-
							*
fall Consider Di Entes ( Warmerer Di Tel Stat	🙀 fend en Hinne 🛄 S	in Company Adaption Str. 1	hd				

# DESING STEPS

DESING DESCRIPTION

TOOLS USED

CREATE THE NEW PROJECT COUNTER

CREATE AN HDL SOURCE

DESING SIMILATION

SIMILATION DESING FUNCTIONALITY

CREATE VHDL TEST BENCH

# DESING DESCRIPTION

This project name STAT\_TWO.The project have tree ports.A: in STD\_LOGIC.CLOCK: in STD\_LOGIC.RESET: in STD\_LOGIC.X: out STD\_LOGIC.

The project family name Spartan3 Dvice XC3S200 Package FT256 Top-lwvwl source type HDL

# TOOLS USED

HARDWARE

Family name Spartan3 Dvice XC3S200 Package FT256 Speed -4

#### SOFTWARE

XILINX-ISE 9.1 i Top Level source type Synthesis tool Similator Language

HDL XST (VHDL/verilog) ISE Similator (VHDL/verilog) VHTL

# STAT\_TWO

#### Create a New Project Counter

Project Name:	Project Location	
stat_two	C:\%ilinx91i\stat_two	
elect the Type of Top-Level Sourc	e for the Project	***************************************
op-Level Source Type:		
HDL		
<del>Qurburg (1.0 univ us us d. 1.0 d. 1</del>		Mgg gag gag and an

Create a new ISE project which will target the FPGA device on the Spartan-3 Startup Kit

demo board.

To create a new project:

1. Select File ODNew Project... The New Project Wizard appears.

2. Type stat\_two in the Project Name field.

3. Enter or browse to a location (directory path) for the new project. A tutorial subdirectory is created automatically.

4. Verify that HDL is selected from the Top-Level Source Type list.

5. Click Next to move to the device properties page.

6. Fill in the properties in the table as shown below:

DDProduct Category: All

DEFamily: Spartan3

DDevice: XC3S200

IIIPackage: FT256
IIISpeed Grade: -4
IIITop-Level Source Type: HDL
IIISynthesis Tool: XST (VHDL/Verilog)
IIISimulator: ISE Simulator (VHDL/Verilog)
IIIPreferred Language: Verilog (or VHDL)
IIIVerify that Enable Enhanced Design Summary is selected.
Leave the default values in the remaining fields.
When the table is complete, your project properties will look like the following: Figure 2: Project Device Properties

#### Mew Project Wizard - Device Properties

\_ 🗆 ×

-Select the Device and Design Flow for the Project -----

Property Name	Value				
Product Category	All				
Family	Spartan3				
Device	×C3S200				
Package	FT 256				
Speed	-4				
Top-Level Source Type	HDL				
Synthesis Tool	XST (VHDL/Verilog)	<b>•</b>			
Simulator	ISE Simulator (VHDL/Verilog)				
Preferred Language	VHDL				
Enable Enhanced Design Summary					
Enable Message Filtering					
Display Incremental Messages					

More Info

< Back

Next >

Cancel

#### Create an HDL Source

Create a VHDL source file for the project as follows:

1. Click the New Source button in the New Project Wizard.

- 2. Select VHDL Module as the source type.
- 3. Type in the file name stat\_two.
- 4. Verify that the Add to project checkbox is selected.
- 5. Click Next.

6. Declare the ports for the counter design by filling in the port information as shown below:

BMM File         IP (Coregen & Architecture Wizard)         Image: Schematic         Implementation Constraints File         State Diagram	File name:	
Test Bench WaveForm	stat_two	]
Werilog Test Fixture       Warner       WHDL Module       VHDL Library       VHDL Package       Warner       WHDL Test Bench	C:\Xilinx91i\stat_two	
	Add to project	
More Info	<pre></pre>	Cancel

#### 🚾 New Source Wizard - Define Module \_ 🗆 🗙 Entity Name stat\_two Architecture Name Behavioral LSB \* Port Name Direction Bus MSB • in а clock in in reset out х in in in in -in -| in •

More Info

< Back Next >

Cancel

Project Navigati	or will create	e a new skeleton source v	with the following specifications:
Add to Project: Source Director Source Type: V Source Name: st Architecture Na Port Definitions: a clock reset x	Yes Yy: C:'XilinxS HDL Modul stat_two.vhr st_two me: Behavi Men Pin Pin Pin Pin	11i\stat_two e j oral in in out	

7. Click Next, then Finish in the New Source Wizard - Summary dialog box to complete the new source file template.

8. Click Next, then Next, then Finish.

Stat\_two displays in the Source tab, as shown below:

Sources for: Behavioral Simulation  Sources for: Behavioral Simulation  Stat_two  Stat_two  Stat_two_tb_vhd - behavior (stat_tt  Stat_two-Behavioral (sta  Sources Snapshots Libraries  Processes for: uut - stat_two - Behavioral  Add Existing Source  Create New Source  Create New Source  Kilinx ISE Simulator  Simulate Behavioral Model	<pre>44 45 if (reset='1') then 46</pre>					
Parsing "stat_two_stx.prj":	63 64 65 66 67 68 ✔ ☑ Design	if a='0' then next_state <= elsif a='1' th next_state <= end if; Summary	s0; en s1;			
Process "Check Syntax" comp	leted su	uccessfully				
Engineer:  Create Date: 13:39:12 08/14/200 Design Name: Module Name: stat_two - Behavin Project Name: Target Devices: Tool versions: Description:	08 oral					

-- Dependencies:

- ---
- -- Revision:

-- Revision 0.01 - File Created

- -- Additional Comments:
- --

library IEEE; use IEEE.STD\_LOGIC\_1164.ALL; use IEEE.STD\_LOGIC\_ARITH.ALL; use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

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

entity stat\_two is Port ( a : in STD\_LOGIC; clock : in STD\_LOGIC; reset : in STD\_LOGIC; x : out STD\_LOGIC); end stat\_two;

architecture Behavioral of stat\_two is

type state\_type is (s0,s1); signal next\_state, current\_state: state\_type; begin state\_reg: process (clock, reset) begin

if (reset='1') then

current\_state <= s0; elsif (clock'event and clock='1') then current\_state <= next\_state;

end if; end process;

-- current process#2:combinational logic comb\_logic:process(current\_state, a) begin

-- use case statement to show the --state transistion

```
case current_state is
```

```
when s0 => x <= '0';
if a='0' then
next_state <= s0;
elsif a='1' then
next_state <= s1;
end if;
```

```
when s1=> x <= '1';
if a='0' then
next_state <= s1;
elsif a='1' then
next_state <= s0;
end if;
when others=>
x <= '0';
next_state <= s0;
end case;
end process;
```

end Behavioral; Synthesis Report:

**Design Simulation** 



Now: 1000 ns	0	200	400 		600 	800
<b>i</b> ∐a	U			u –		$\frac{1}{1 - \frac{1}{2}} \frac{1}{1 - \frac$
Ll clock	U			U		
ill reset	U			- U		
<b>a</b> ∐∮x	0					
o∬next_state	sO			sO		
ourrent_state	sO			sO		
•						
📡 Design Summary	stat_two.vhd	THe stat_two_tb.vhd	Simulation			

#### CREATE THE TEST BENCH

Sources for: Behavioral Simulation	58state transistion
	59 60 case current_state is
A KC3S2UU-4II206	61
New Source	62  when s0 => x <= '0';
Add Source	64 next state <= s0;
Sources Add Copy of Source	
[2] Open	56
Set as Top Module	68
Processes for: stat_t Use SmartGuide	69 when s1=> x <= '1';
Create Ne New Partition	
- Xilinx ISE Delete Partition	72 elsif a='1' then
	73
Partition Force	▶ 75 when others=>
Remove	76 x <= '0';
Move to Library	77
	79 end process;
l oggle Paths	во
Properties	B1 end Behavioral;
	82
Ett Processes	
L HOUSSES	📡 Design Summary 🛛 🖌 stat_two.vhd

BMM File IP (Coregen & Architecture Wizard) MEM File Schematic Implementation Constraints File State Diagram Test Bench WaveForm User Document V Verilog Module Verilog Test Fixture VHDL Module VHDL Library P VHDL Package VHDL Test Bench	File name: stat_two_tb Location: C:\Xilinx91i\stat_two 
More Info	Add to project < Back Next > Cancel

-- Company:

-- Engineer:

- -- Create Date: 01:17:18 08/10/2008
- -- Design Name: stat\_two
- -- Module Name: C:/Xilinx91i/stat\_two/stat\_two\_tb.vhd
- -- Project Name: stat\_two
- -- Target Device:
- -- Tool versions:
- -- Description:
- -- VHDL Test Bench Created by ISE for module: stat two
- --
- -- Dependencies:
- ---

-----

- -- Revision:
- -- Revision 0.01 File Created
- -- Additional Comments:
- -- Notes:
- -- This testbench has been automatically generated using types std\_logic and
- -- std\_logic\_vector for the ports of the unit under test. Xilinx recommends
- -- that these types always be used for the top-level I/O of a design in order
- -- to guarantee that the testbench will bind correctly to the post-implementation

-- simulation model.

LIBRARY ieee; USE ieee.std\_logic\_1164.ALL; USE ieee.std\_logic\_unsigned.all; USE ieee.numeric\_std.ALL;

ENTITY stat\_two\_tb\_vhd IS END stat\_two\_tb\_vhd;

ARCHITECTURE behavior OF stat\_two\_tb\_vhd IS

);

END COMPONENT;

```
--Inputs
SIGNAL a : std_logic := '0';
SIGNAL clock : std_logic := '0';
SIGNAL reset : std_logic := '0';
```

```
--Outputs
SIGNAL x : std_logic;
```

#### BEGIN

end process;

#### tb : PROCESS BEGIN

```
-- Wait 100 ns for global reset to finish
wait for 100 ns;
reset <='0';
a<='0';
wait for 100 ns;
a<='1';
wait for 100 ns;
a<='0';
```

-- Place stimulus here

wait; -- will wait forever END PROCESS;

END;

ARCHITECTURE behavior OF stat two tb vh 37 Sources for: Behavioral Simulation 38 --- 🖻 stat\_two 39 -- Component Declaration for the Uni Ė- ₩ xc3s200-4ft256 40 COMPONENT stat two 🚊 🖫 stat\_two\_tb\_vhd - behavior (stat\_tv 41 PORT ( " 🐂 uut - stat\_two - Behavioral (sta 🔽 42 a : IN std\_logic; 4 . 43 clock : IN std logic; 44 reset : IN std logic; C Sources Libraries Snapshots x : OUT std logic 45 46 ); × 47 END COMPONENT; Processes for: uut - stat\_two - Behavioral 48 --- Add Existing Source 49 --Inputs 50 SIGNAL a : std\_logic := '0'; Create New Source ----51 SIGNAL clock : std\_logic := '0'; 🗄 🎲 🛛 Xilinx ISE Simulator 52 SIGNAL reset : std logic := '0'; Check Syntax 53 Simulate Behavioral Model 54 --Outputs 55 SIGNAL x : std logic; 56 BEGIN 57 58 59 -- Instantiate the Unit Under Test ( uut: stat\_two PORT MAP( 60 61 a => a, --• Processes E Design Summary stat\_two\_tb.vhd ×

Parsing "stat two stx.prj": 0.06

Process "Check Syntax" completed successfully



ļ

l

Now: 1000 ns		0	20	)0 	4	00	6	00	
<b>o</b> j]a	0								
olock 🗧	0	IIIII			IIIIII				IIIII
<b>i</b> ∫ reset	0			I. O.		Paule			
<b>o</b> , [] x	1		e brude h						
🔀 Design Summary	ин <sub>о</sub> sta	t_two.vhd	"Ha stat_tw	o_tb.vhd	Simul	ation			

1

1

l

b