DAC USING SPARTAN 3E

DSD Lab Project

Submitted to Prof. Hardik Joshi

21BEC095 Vedanti Patel
21BEC121 Mantra Solanki

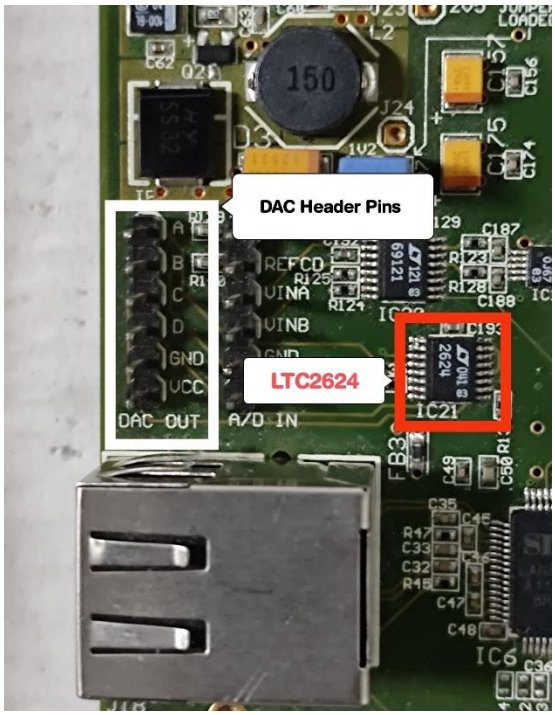


Figure 1

The Spartan®-3E FPGA Starter Kit board includes an SPI-compatible, four-channel, serial Digital-to-Analog Converter (DAC). The DAC device is a *Linear Technology LTC2624* quad DAC with 12-bit unsigned resolution. The four outputs from the DAC appear on the J5 header, which uses the Digilent 6-pin Peripheral Module format. The DAC and the header are located immediately above the Ethernet RJ-45 connector, as shown in Figure 1.

SPI_MOSI	: (T4)
SPI_MISO	: (N10)
SPI_SCK	: (U16)
DAC_CS	: (N8)
DAC_CLR	: (P8)

SPI Communication

As shown in Figure 2, the FPGA uses a Serial Peripheral Interface (SPI) to communicate digital values to each of the four DAC channels. The SPI bus is a full-duplex, synchronous, character-oriented channel employing a simple four-wire interface. A bus master—the FPGA in this example—drives the bus clock signal (SPI_SCK) and transmits serial data (SPI_MOSI) to the selected bus slave—the DAC in this example. At the same time, the bus slave provides serial data (SPI_MISO) back to the bus master.

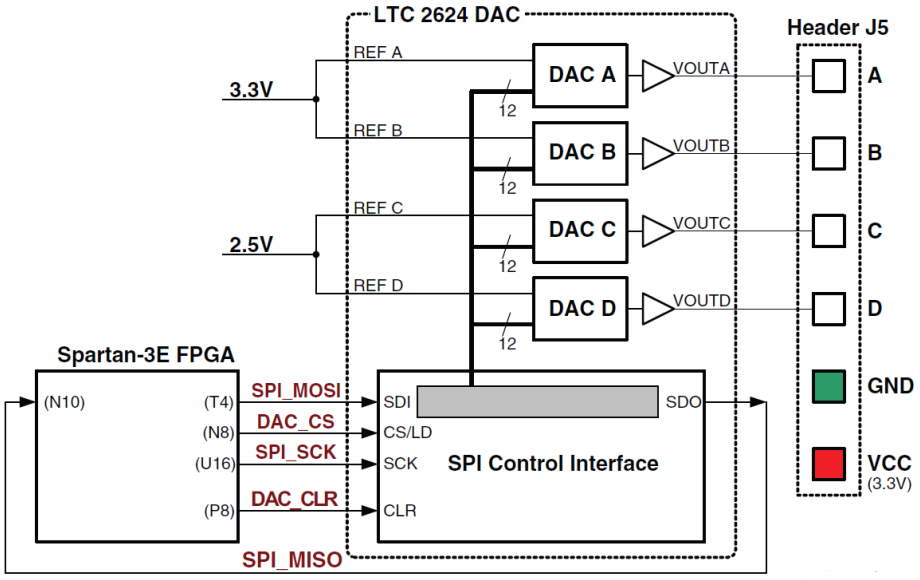


Figure 2

Interface Signals

Table 1 lists the interface signals between the FPGA and the DAC. The SPI_MOSI, SPI_MISO, and SPI_SCK signals are shared with other devices on the SPI bus. The DAC_CS signal is the active-Low slave select input to the DAC. The DAC_CLR signal is the active-Low, asynchronous reset input to the DAC.

Signal	***FPGA Pin***	***Direction***	***Description***
SPI_MOSI	T4	FPGA → DAC	Serial data: Master Output, Slave Input
DAC_CS	N8	FPGA → DAC	Active-Low chip-select. Digital-to-analog conversion starts when signal returns High.
SPI_SCK	U16	FPGA → DAC	Clock
DAC_CLR	P8	FPGA → DAC	Asynchronous, active-Low reset input
SPI_MISO	N10	FPGA ← DAC	Serial data: Master Input, Slave Output

Table 1

The serial data output from the DAC is primarily used to cascade multiple DACs. This signal can be ignored in most applications although it does demonstrate full-duplex communication over the SPI bus.

Disable Other Devices on the SPI Bus to Avoid Contention

The SPI bus signals are shared by other devices on the board. It is vital that other devices are disabled when the FPGA communicates with the DAC to avoid bus contention. Table 9-2 provides the signals and logic values required to disable the other devices. Although the StrataFlash PROM is a parallel device, its least-significant data bit is shared with the SPI_MISO signal.

Signal	***Disabled Device***	***Disable Value***
SPI_SS_B	SPI serial Flash	1
AMP_CS	Programmable pre-amplifier	1
AD_CONV	Analog-to-Digital Converter (ADC)	0
SF_CE0	StrataFlash Parallel Flash PROM	1
FPGA_INIT_B	Platform Flash PROM	0

Table 2

SPI Communication Details

Figure 3 shows a detailed example of the SPI bus timing. Each bit is transmitted or received relative to the SPI_SCK clock signal. The bus is fully static and supports clocks rate up to the maximum of 50 MHz. However, check all timing parameters using the LTC2624 data sheet if operating at or close to the maximum speed.

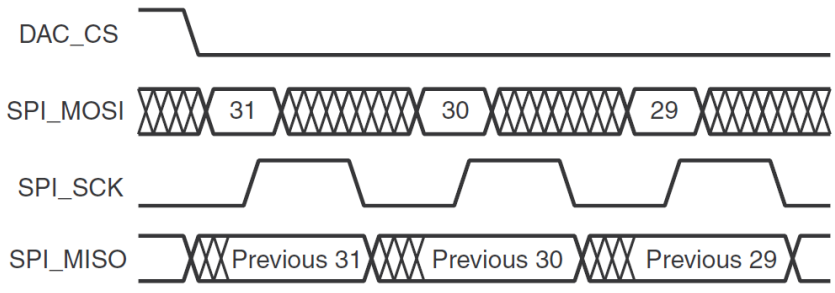


Figure 3

After driving the DAC_CS slave select signal Low, the FPGA transmits data on the SPI_MOSI signal, MSB first. The LTC2624 captures input data (SPI_MOSI) on the rising edge of SPI_SCK; the data must be valid for at least 4 ns relative to the rising clock edge. The LTC2624 DAC transmits its data on the SPI_MISO signal on the falling edge of SPI_SCK. The FPGA captures this data on the next rising SPI_SCK edge. The FPGA must read the first SPI_MISO value on the first rising SPI_SCK edge after DAC_CS goes Low. Otherwise, bit 31 is missed.

After transmitting all 32 data bits, the FPGA completes the SPI bus transaction by returning the DAC_CS slave select signal High. The High-going edge starts the actual digital-to-analog conversion process within the DAC.

Communication Protocol

Figure 4 shows the communications protocol required to interface with the LTC2624 DAC. The DAC supports both a 24-bit and 32-bit protocol. The 32-bit protocol is shown. Inside the D/A converter, the SPI interface is formed by a 32-bit shift register. Each 32-bit command word consists of a command, an address, followed by data value. As a new command enters the DAC, the previous 32-bit command word is echoed back to the master. The response from the DAC can be ignored although it is a useful to confirm correct communication.

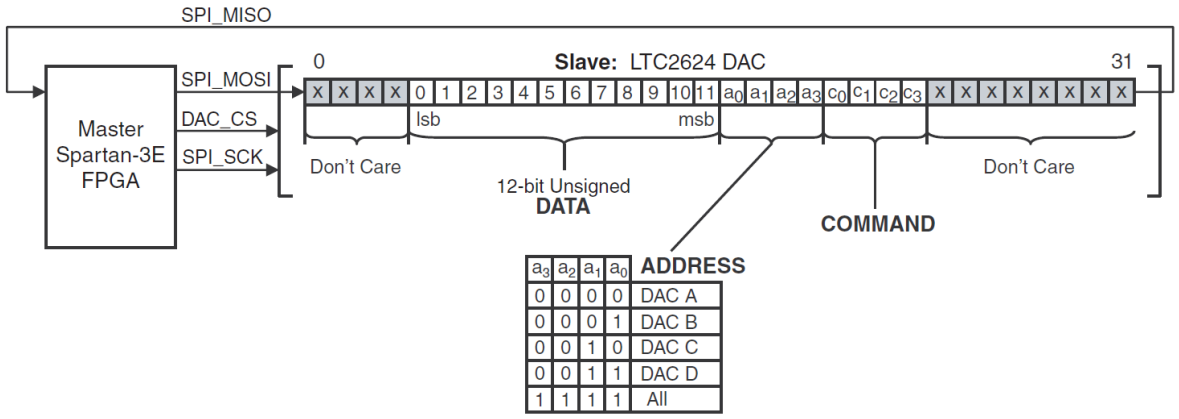


Figure 4

The FPGA first sends eight dummy or "don't care" bits, followed by a 4-bit command. The most commonly used command with the board is COMMAND[3:0] = "0011", which immediately updates the selected DAC output with the specified data value. Following the command, the FPGA selects one or all the DAC output channels via a 4-bit address field. Following the address field, the FPGA sends a 12-bit unsigned data value that the DAC converts to an analog value on the selected output(s). Finally, four additional dummy or don't care bits pad the 32-bit command word.

Specifying the DAC Output Voltage

As shown in Figure 2, each DAC output level is the analog equivalent of a 12-bit unsigned digital value, D[11:0], written by the FPGA to the DAC via the SPI interface. The voltage on a specific output is generally described in Equation 1. The reference voltage, VREFERENCE, is different between the four DAC outputs. Channels A and B use a 3.3V reference voltage and Channels C and D use a 2.5V

reference. The reference voltages themselves have a ±5% tolerance, so there will be slight corresponding variances in the output voltage.

Equation 1:

$$V_{OUT} = \frac{D[11:0]}{4{,}096} \times V_{REFERENCE}$$

DAC Outputs A and B

Equation 2 provides the output voltage equation for DAC outputs A and B. The reference voltage associated with DAC outputs A and B is 3.3V ±5%.

Equation 2:

$$V_{OUTA} = \frac{D[11:0]}{4{,}096} \times (3.3V \pm 5\%)$$

DAC Outputs C and D

Equation 3 provides the output voltage equation for DAC outputs A and B. The reference voltage associated with DAC outputs A and B is 2.5V ±5%.

Equation 3:

$$V_{OUTC} = \frac{D[11:0]}{4{,}096} \times (2.5V \pm 5\%)$$

Code explanation:

Module Declaration:

The code starts by declaring the module named "dac." This module defines input and output ports for connecting the FPGA to the DAC. Key ports include the clock signal (`clk`), reset signal (`reset`), data to be sent to the DAC (`data`), control signals (`dac_cs`, `spi_sck`, `spi_mosi`, `dac_clr`), and various other peripheral control signals.

Internal Registers and Wires:

The module uses internal registers and wires to manage its operation. Key registers include `dac_state` to track the state of the DAC interface, `dac_out` to format the data to be sent to the DAC, and `count` to keep track of the number of bits being sent.

Assignments:

Several control signals (`SPI_SS_B`, `AMP_CS`, `AD_CONV`, `SF_CEO`, `FPGA_INIT_B`) are assigned values to disable other peripherals connected to the SPI bus, ensuring that the DAC communication occurs independently.

State Machine:

The core of the module is a state machine implemented within an `always` block. This state machine controls the sequence of actions for transmitting data to the DAC. The states are numbered from 0 to 7, representing different stages of DAC data transmission.

1. State 0 (IDLE): In this state, the control signals are set to default values. The interface is idle, and no data is transmitted.
2. State 1 (Assigning 32 Bits to DAC): The data to be sent to the DAC is formatted in this state. It includes command bits, address bits, and the data itself. The data is placed in the `dac_out` register.
3. State 2 (Assigning Bits to SPI_MOSI): In this state, the DAC's chip select (`dac_cs`) is lowered to initiate data transmission. Data is then shifted out bit by bit on the `spi_mosi` line.
4. State 3 (Waiting for Complete 32-Bit Input): This state checks if there are more bits to be transmitted. If there are remaining bits, the process returns to State 2. Otherwise, it proceeds to State 4.
5. State 4 (SPI_SCK Low): The clock signal (`spi_sck`) is set low to end the data transmission.
6. State 5 (DAC_CS High): The chip select (`dac_cs`) signal is set high, marking the end of data transmission to the DAC.
7. State 6 (Send Data): The `send` signal is asserted, indicating that data has been sent to the DAC.
8. State 7 (Send Data Complete): The `send` signal is de-asserted, and the state machine returns to the IDLE state (State 0).

The state machine continuously loops through these states, and the entire process is governed by the clock signal (`clk`). When a reset occurs (`reset` is high), the interface is reset, and other peripherals are disabled.

Snippet of the code in ISE Project Navigator:

ISE is a software by Xilinx for programing Spartan 3e FPGA board.

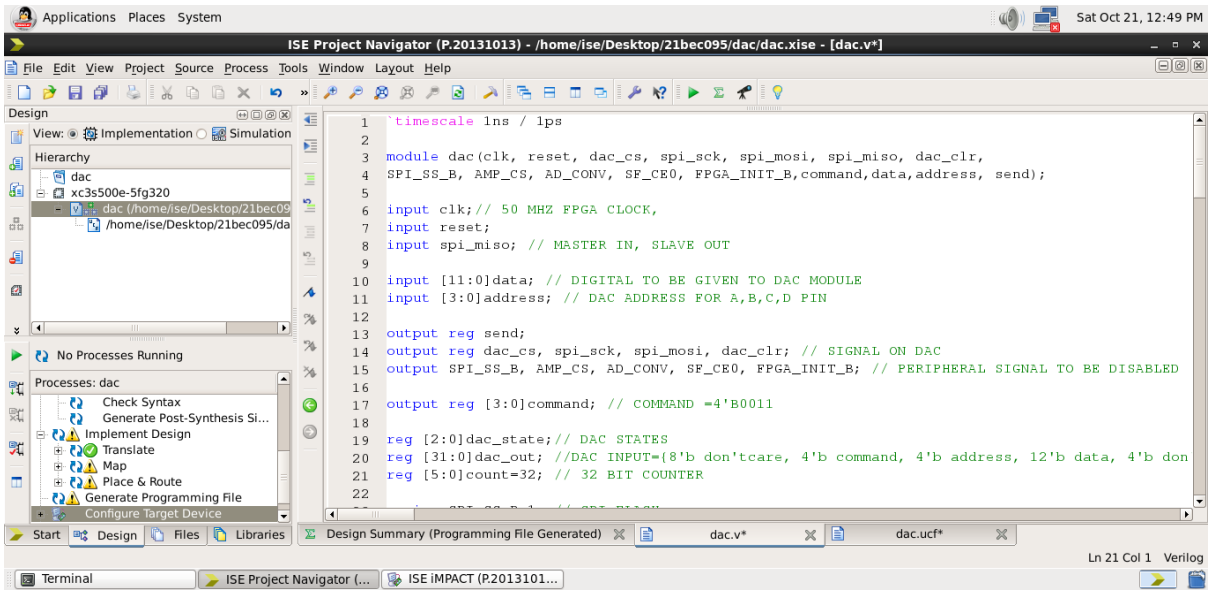


Figure 5

User Constraint File:

This file is made for pin assignments.

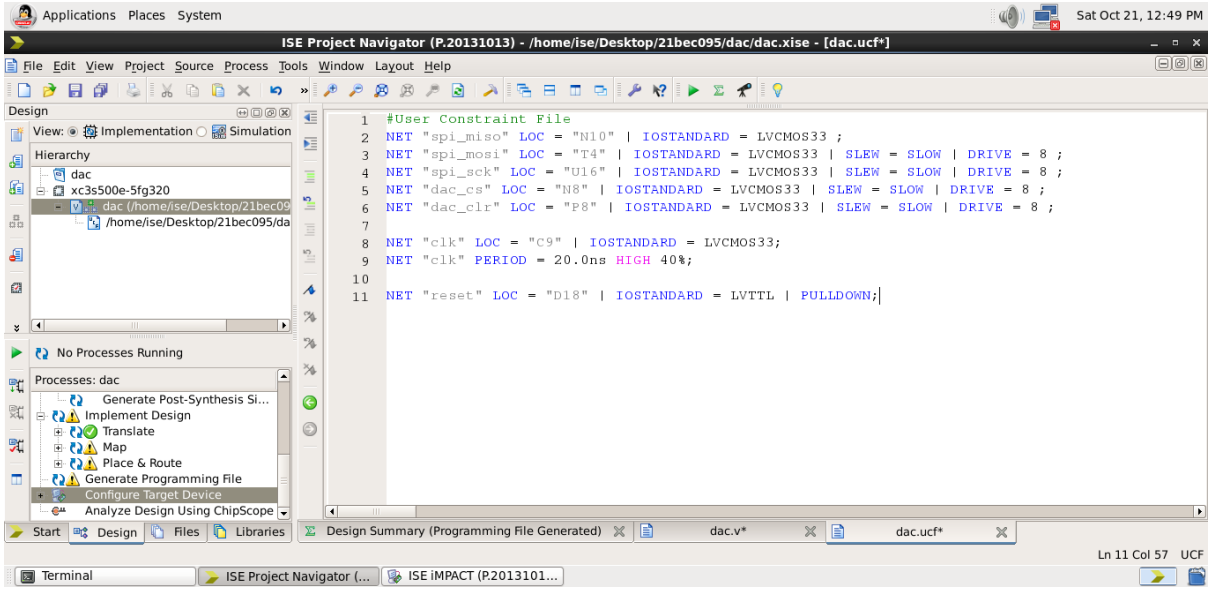


Figure 6

Dumping the code through ISE iMPACT:

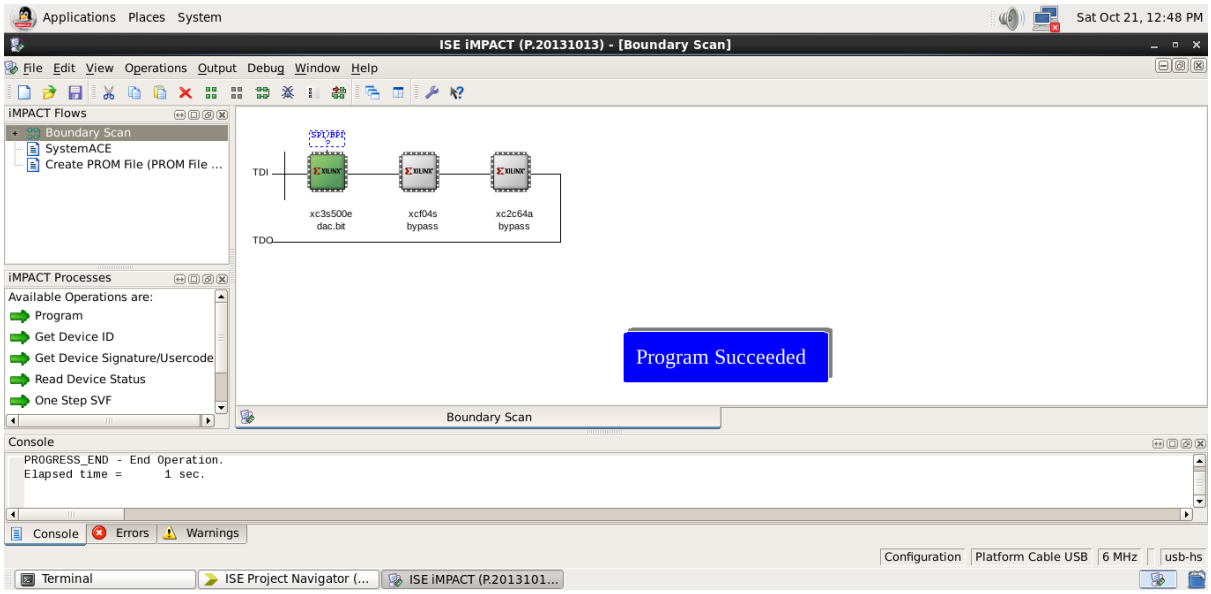


Figure 7

Output:

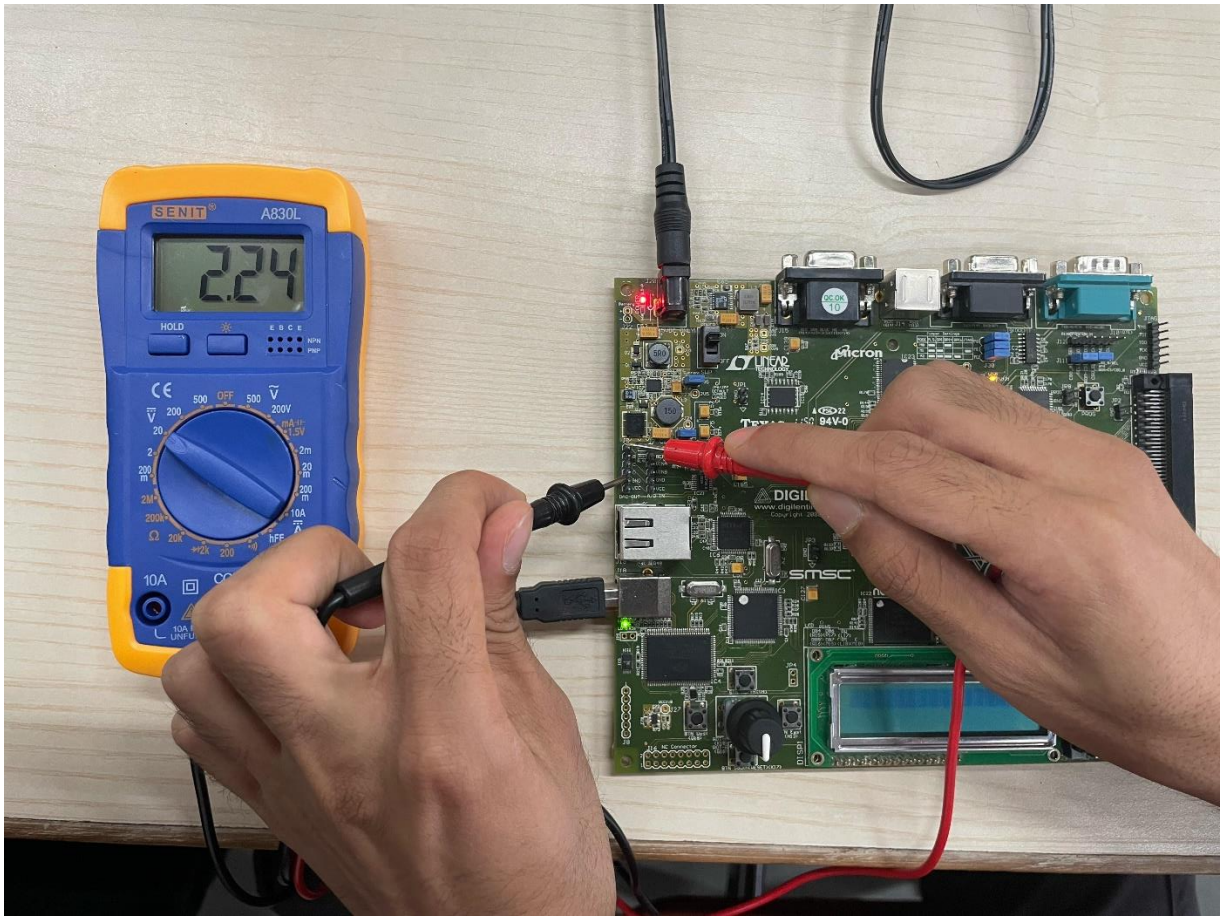
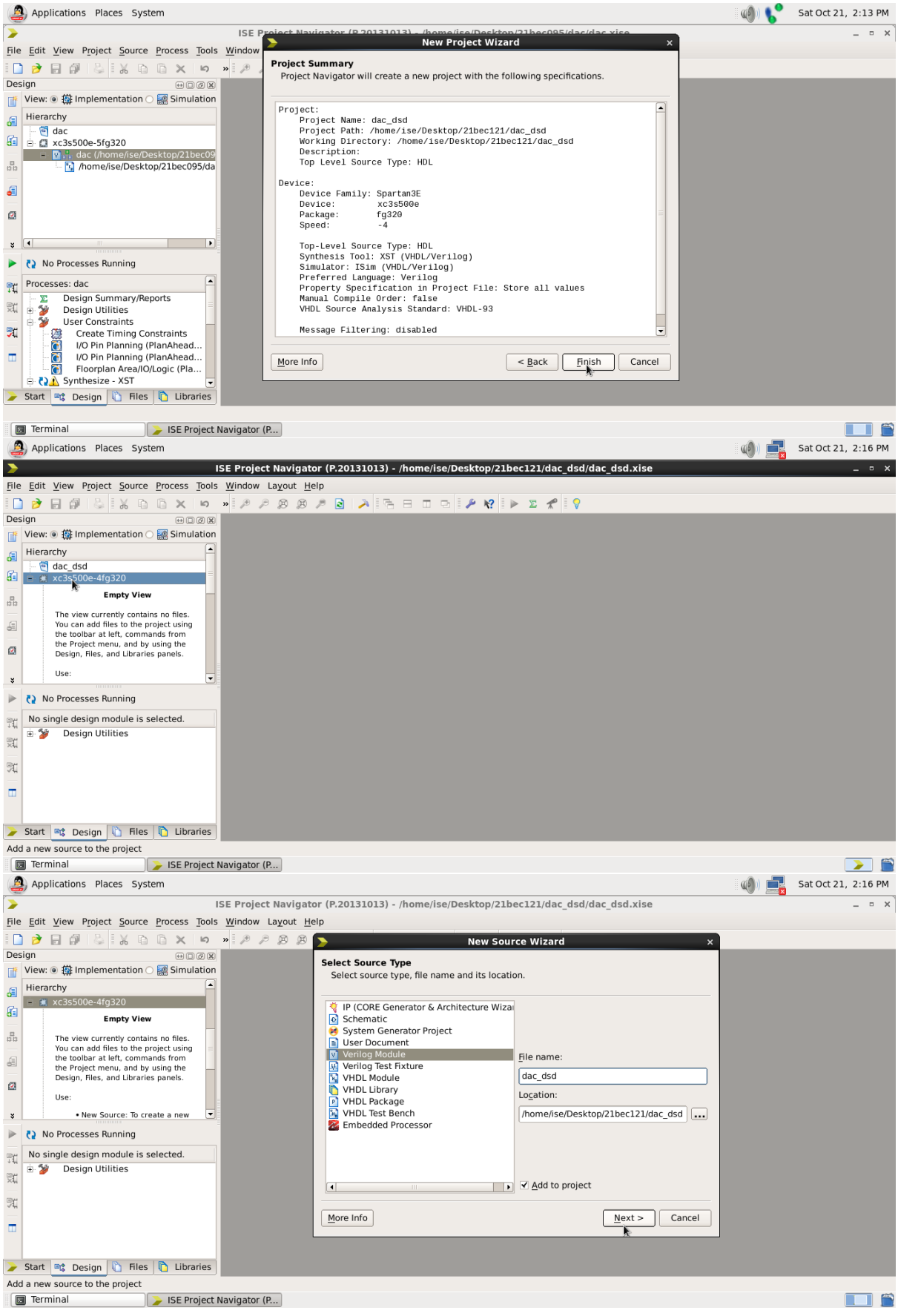
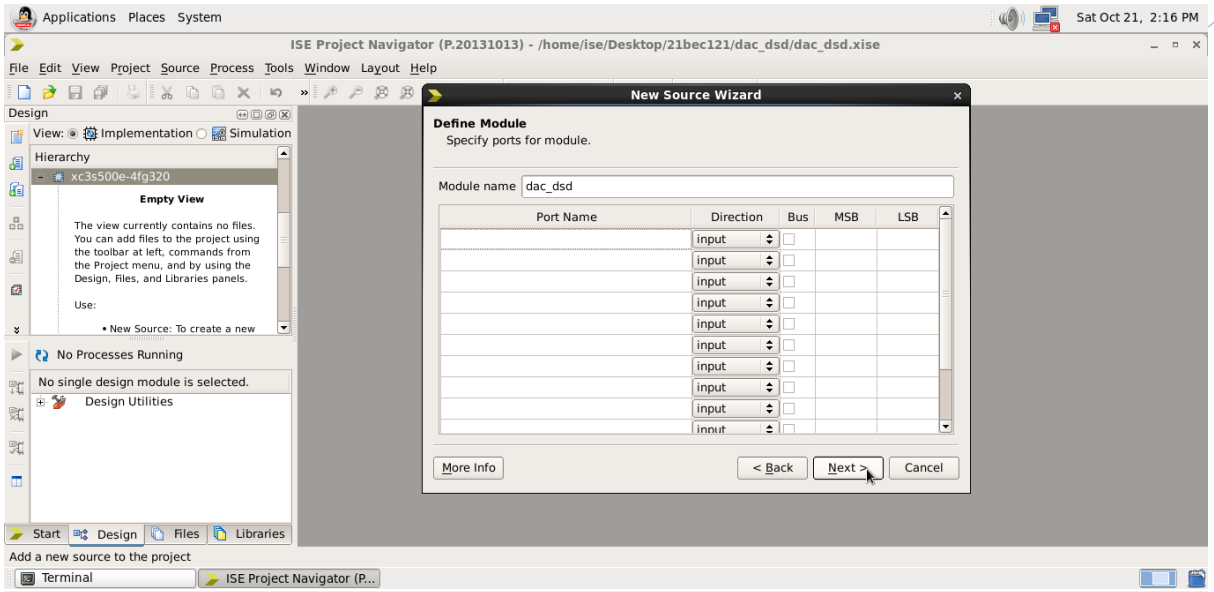
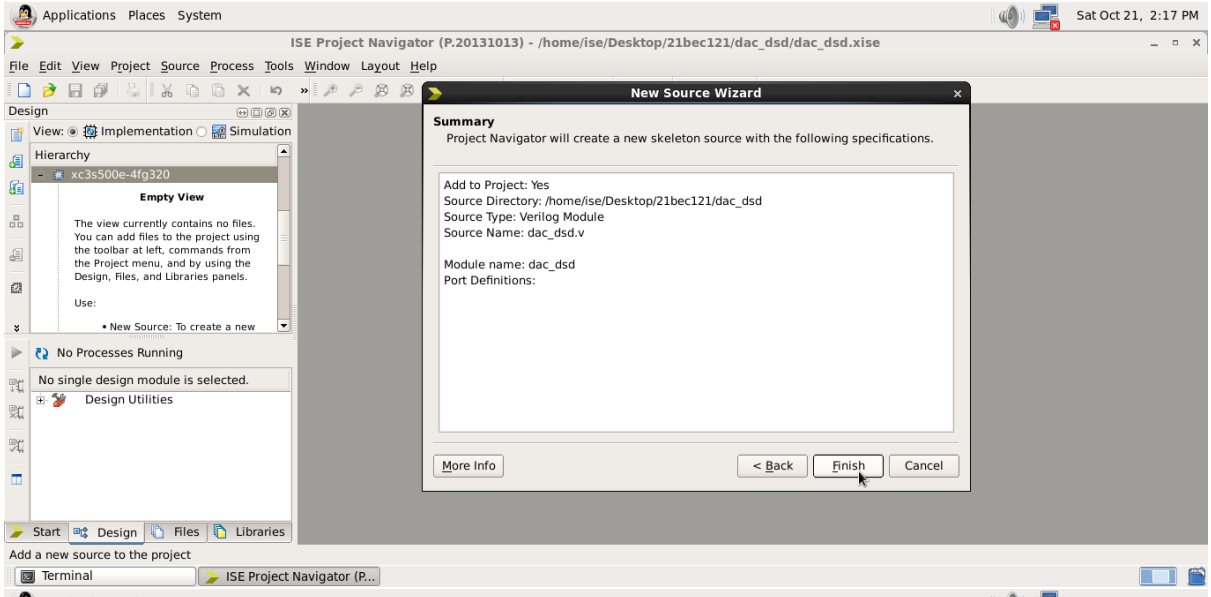


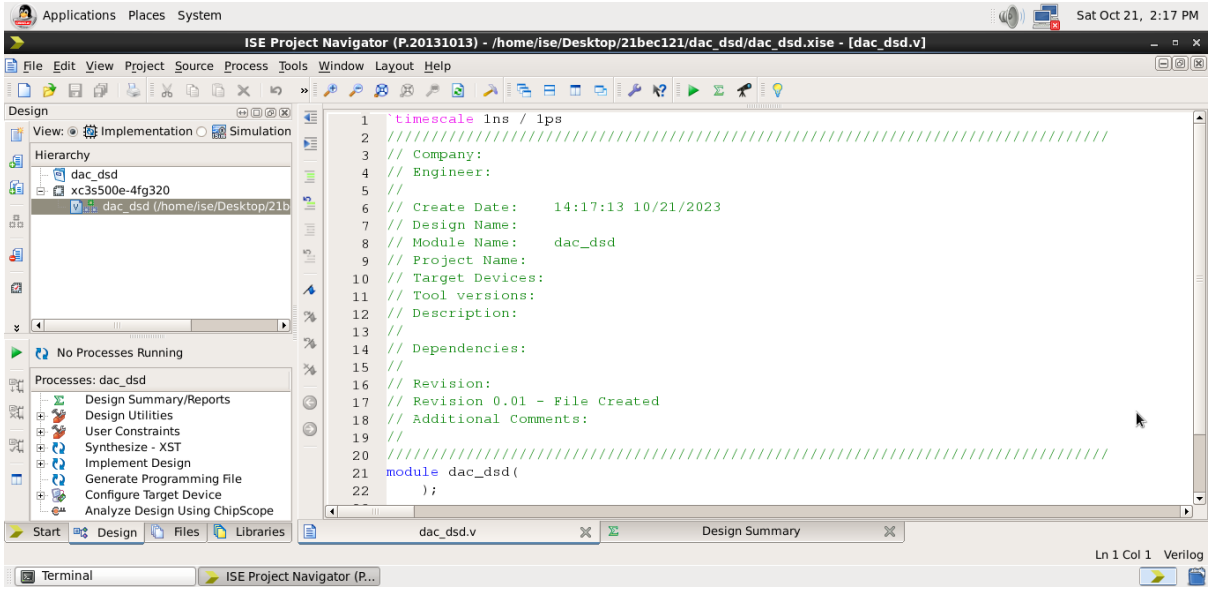
Figure 8

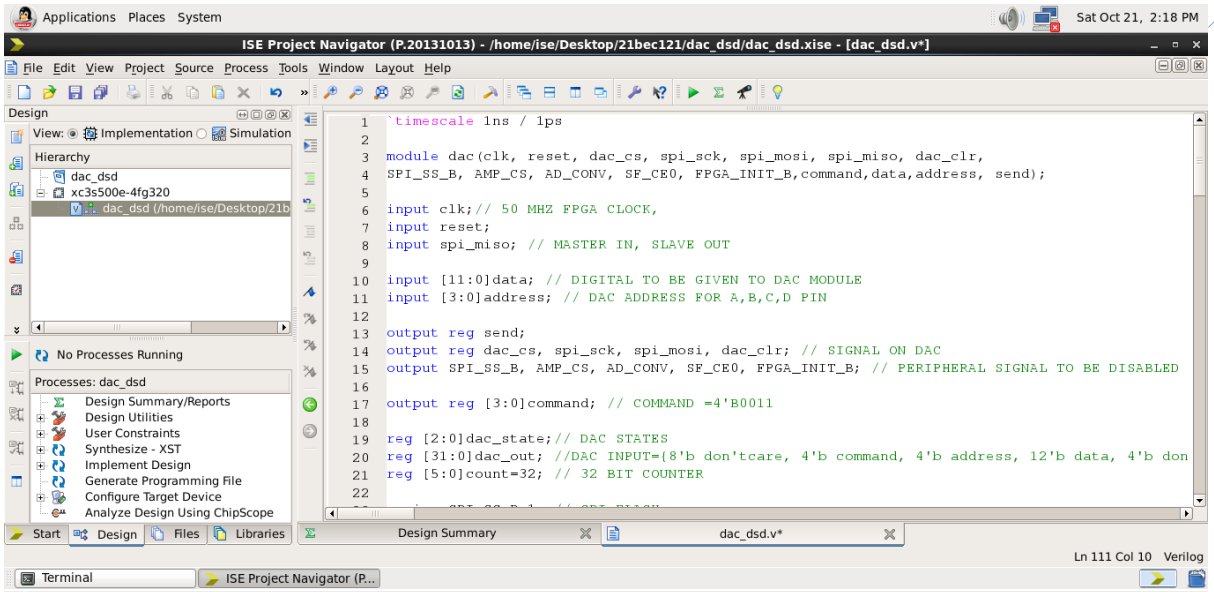
Steps:

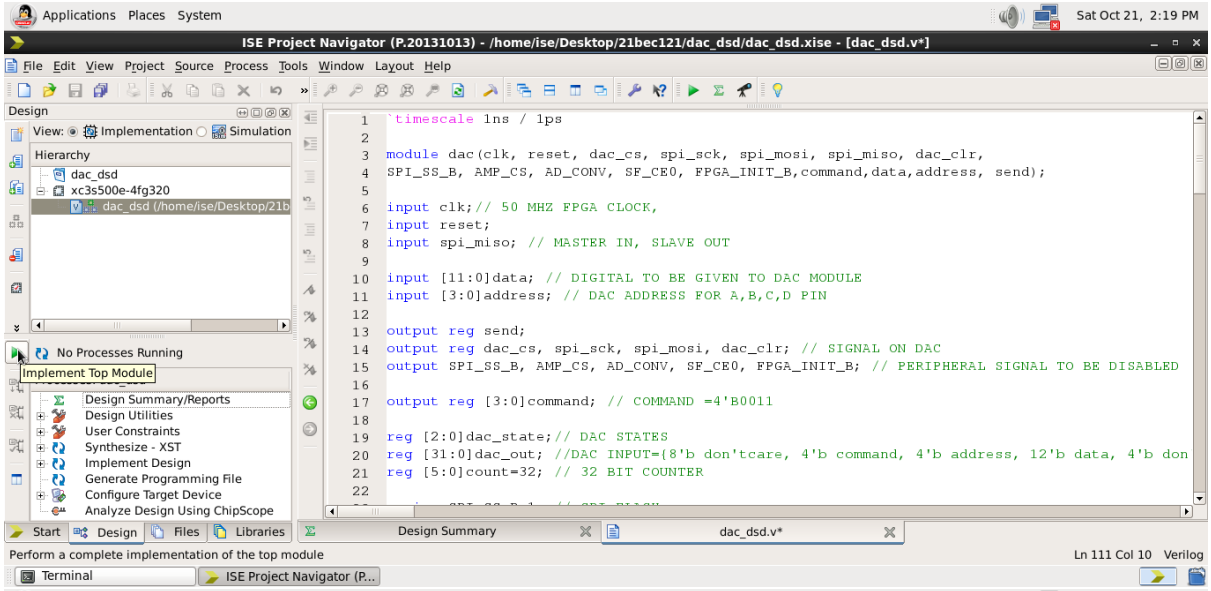

Applications Places System
Sat Oct 21, 2:13 PM
New Project Wizard
Project Summary
Project Navigator will create a new project with the following specifications.
Project:
Project Name: dac_dsd
Project Path: /home/ise/Desktop/21bec121/dac_dsd
Working Directory: /home/ise/Desktop/21bec121/dac_dsd
Description:
Top Level Source Type: HDL
Device:
Device Family: Spartan3E
Device: xc3s500e
Package: fg320
Speed: -4
Top-Level Source Type: HDL
Synthesis Tool: XST (VHDL/Verilog)
Simulator: ISim (VHDL/Verilog)
Preferred Language: Verilog
Property Specification in Project File: Store all values
Manual Compile Order: false
VHDL Source Analysis Standard: VHDL-93
Message Filtering: disabled
More Info
< Back
Finish
Cancel
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
dac_dsd
xc3s500e-4fg320
Empty View
The view currently contains no files. You can add files to the project using the toolbar at left, commands from the Project menu, and by using the Design, Files, and Libraries panels.
Use:
No Processes Running
No single design module is selected.
Design Utilities
Start Design Files Libraries
Add a new source to the project
Terminal
ISE Project Navigator (P...
Sat Oct 21, 2:16 PM
New Source Wizard
Select Source Type
Select source type, file name and its location.
IP (CORE Generator & Architecture Wizard)
Schematic
System Generator Project
User Document
Verilog Module
Verilog Test Fixture
VHDL Module
VHDL Library
VHDL Package
VHDL Test Bench
Embedded Processor
File name:
dac_dsd
Location:
/home/ise/Desktop/21bec121/dac_dsd
Add to project
More Info
Next >
Cancel

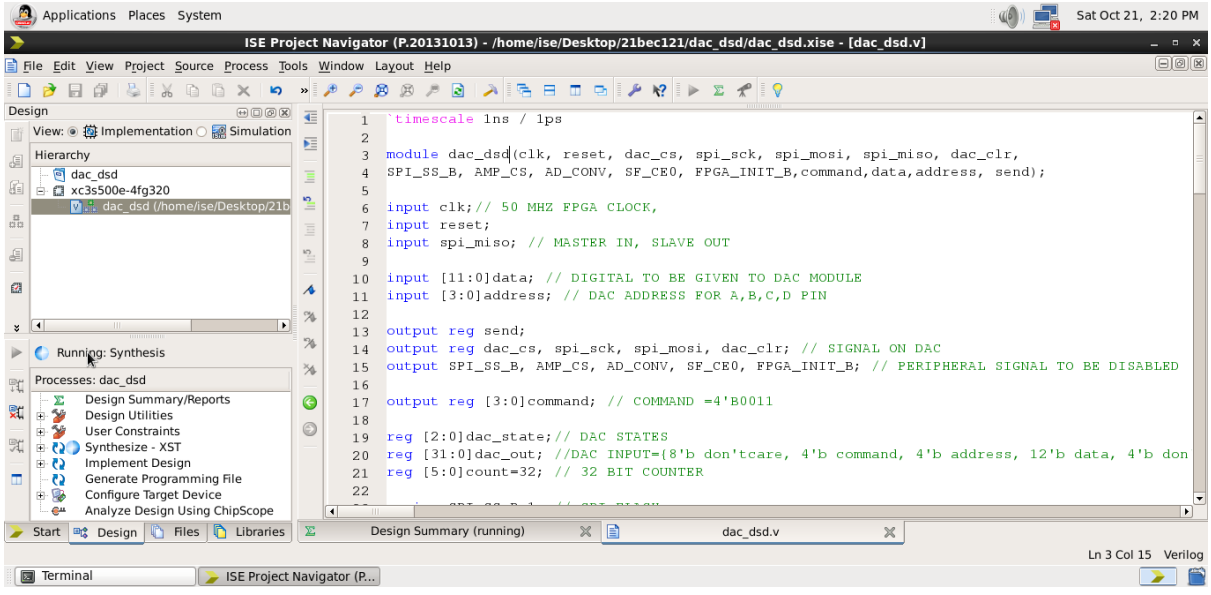

Applications Places System
Sat Oct 21, 2:16 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise
File Edit View Project Source Process Tools Window Layout Help
New Source Wizard
Define Module
Specify ports for module.
Module name dac_dsd
Port Name
Direction
Bus
MSB
LSB
input
More Info
< Back
Next >
Cancel
Design
View: Implementation Simulation
Hierarchy
xc3s500e-4fg320
Empty View
The view currently contains no files. You can add files to the project using the toolbar at left, commands from the Project menu, and by using the Design, Files, and Libraries panels.
Use:
• New Source: To create a new
No Processes Running
No single design module is selected.
Design Utilities
Start
Design
Files
Libraries
Add a new source to the project
Terminal
ISE Project Navigator (P...

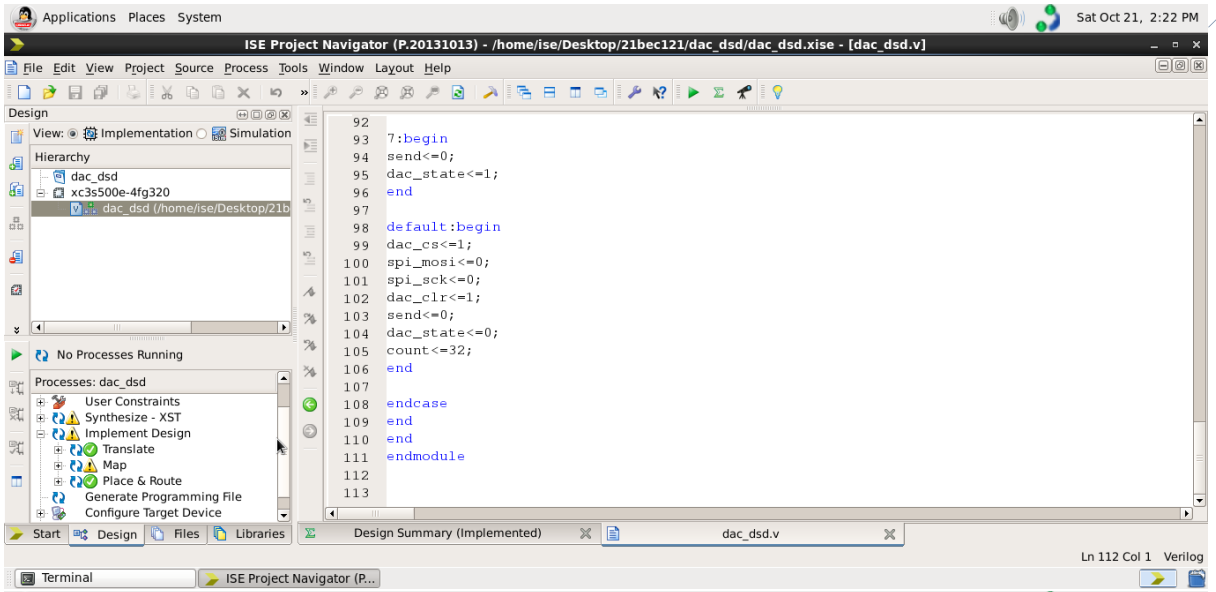

Applications Places System
Sat Oct 21, 2:17 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise
File Edit View Project Source Process Tools Window Layout Help
New Source Wizard
Summary
Project Navigator will create a new skeleton source with the following specifications.
Add to Project: Yes
Source Directory: /home/ise/Desktop/21bec121/dac_dsd
Source Type: Verilog Module
Source Name: dac_dsd.v
Module name: dac_dsd
Port Definitions:
More Info
< Back
Finish
Cancel
Design
View: Implementation Simulation
Hierarchy
xc3s500e-4fg320
Empty View
The view currently contains no files. You can add files to the project using the toolbar at left, commands from the Project menu, and by using the Design, Files, and Libraries panels.
Use:
• New Source: To create a new
No Processes Running
No single design module is selected.
Design Utilities
Start
Design
Files
Libraries
Add a new source to the project
Terminal
ISE Project Navigator (P...

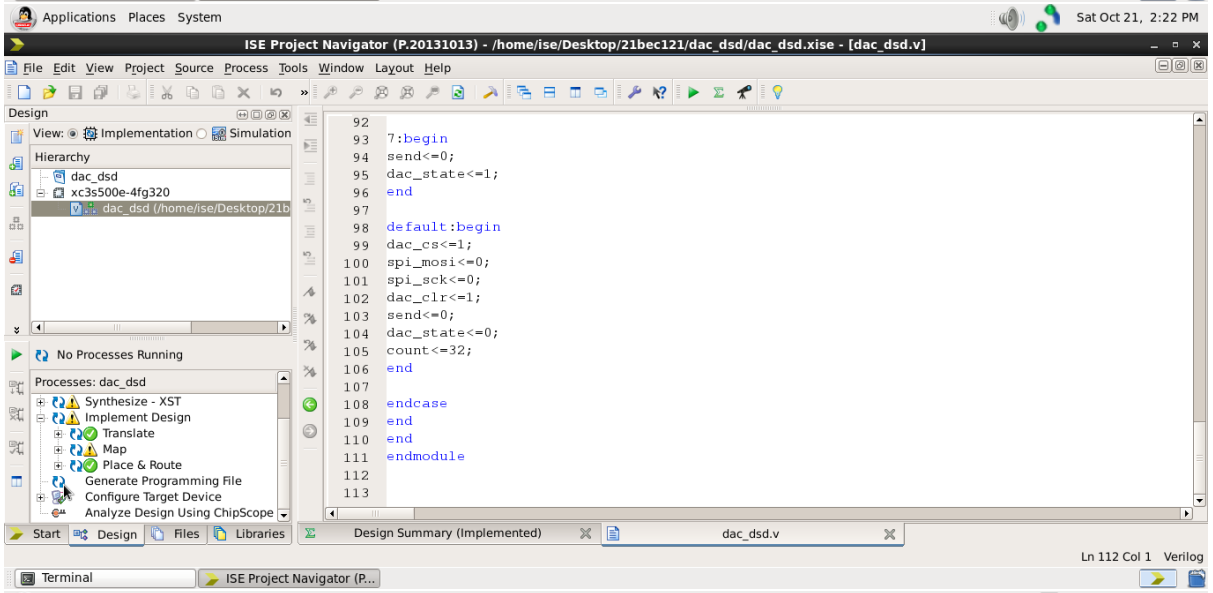

Applications Places System
Sat Oct 21, 2:17 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise - [dac_dsd.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
dac_dsd
xc3s500e-4fg320
dac_dsd (/home/ise/Desktop/21b
No Processes Running
Processes: dac_dsd
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries
1 `timescale 1ns / 1ps
2 //
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 14:17:13 10/21/2023
7 // Design Name:
8 // Module Name: dac_dsd
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 module dac_dsd(
22);
dac_dsd.v
Design Summary
Ln 1 Col 1 Verilog
Terminal
ISE Project Navigator (P...

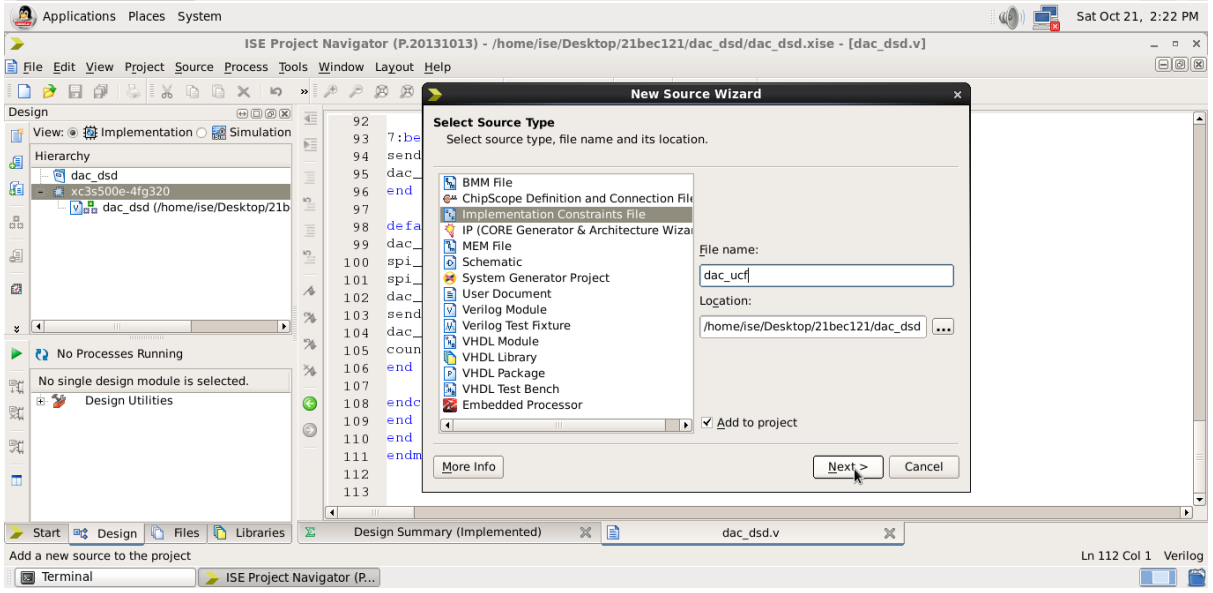

Applications Places System
Sat Oct 21, 2:18 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise - [dac_dsd.v*]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
dac_dsd
xc3s500e-4fg320
dac_dsd (/home/ise/Desktop/21b
No Processes Running
Processes: dac_dsd
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
Design Summary
dac_dsd.v*
Ln 111 Col 10 Verilog
Terminal
ISE Project Navigator (P...
`timescale 1ns / 1ps
module dac(clk, reset, dac_cs, spi_sck, spi_mosi, spi_miso, dac_clr,
SPI_SS_B, AMP_CS, AD_CONV, SF_CE0, FPGA_INIT_B,command,data,address, send);
input clk;// 50 MHZ FPGA CLOCK,
input reset;
input spi_miso; // MASTER IN, SLAVE OUT
input [11:0]data; // DIGITAL TO BE GIVEN TO DAC MODULE
input [3:0]address; // DAC ADDRESS FOR A,B,C,D PIN
output reg send;
output reg dac_cs, spi_sck, spi_mosi, dac_clr; // SIGNAL ON DAC
output SPI_SS_B, AMP_CS, AD_CONV, SF_CE0, FPGA_INIT_B; // PERIPHERAL SIGNAL TO BE DISABLED
output reg [3:0]command; // COMMAND =4'B0011
reg [2:0]dac_state;// DAC STATES
reg [31:0]dac_out; //DAC INPUT={8'b don'tcare, 4'b command, 4'b address, 12'b data, 4'b don
reg [5:0]count=32; // 32 BIT COUNTER

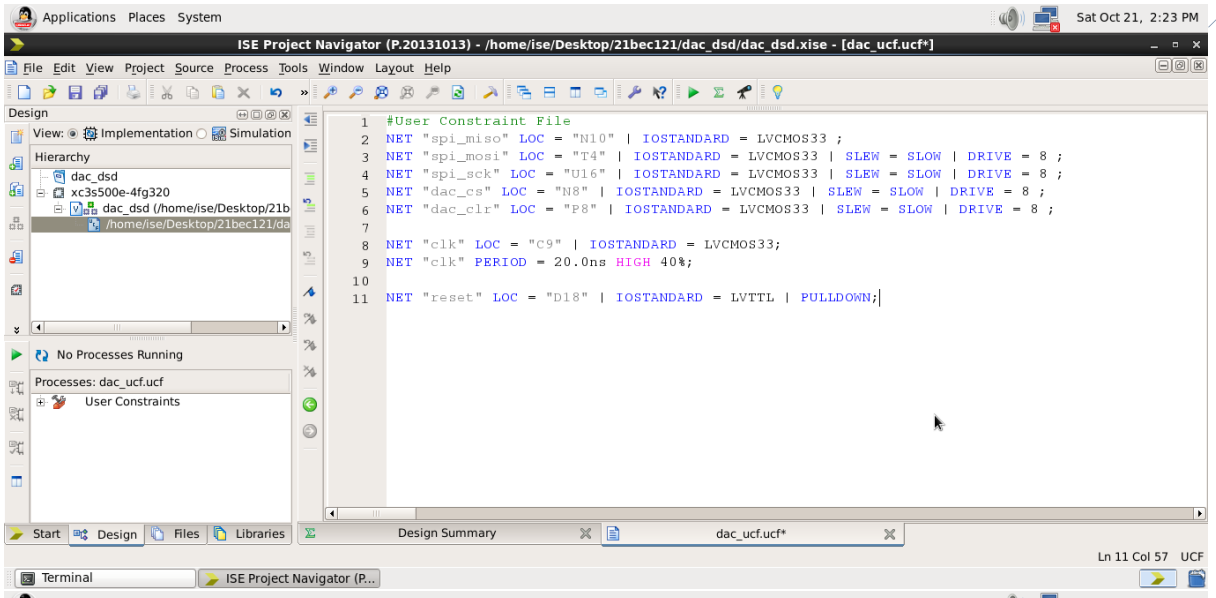

Applications Places System
Sat Oct 21, 2:19 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise - [dac_dsd.v*]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
dac_dsd
xc3s500e-4fg320
dac_dsd (/home/ise/Desktop/21b
No Processes Running
Implement Top Module
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
Design Summary
dac_dsd.v*
Perform a complete implementation of the top module
Ln 111 Col 10 Verilog
Terminal
ISE Project Navigator (P...
`timescale 1ns / 1ps
module dac(clk, reset, dac_cs, spi_sck, spi_mosi, spi_miso, dac_clr,
SPI_SS_B, AMP_CS, AD_CONV, SF_CE0, FPGA_INIT_B,command,data,address, send);
input clk;// 50 MHZ FPGA CLOCK,
input reset;
input spi_miso; // MASTER IN, SLAVE OUT
input [11:0]data; // DIGITAL TO BE GIVEN TO DAC MODULE
input [3:0]address; // DAC ADDRESS FOR A,B,C,D PIN
output reg send;
output reg dac_cs, spi_sck, spi_mosi, dac_clr; // SIGNAL ON DAC
output SPI_SS_B, AMP_CS, AD_CONV, SF_CE0, FPGA_INIT_B; // PERIPHERAL SIGNAL TO BE DISABLED
output reg [3:0]command; // COMMAND =4'B0011
reg [2:0]dac_state;// DAC STATES
reg [31:0]dac_out; //DAC INPUT={8'b don'tcare, 4'b command, 4'b address, 12'b data, 4'b don
reg [5:0]count=32; // 32 BIT COUNTER

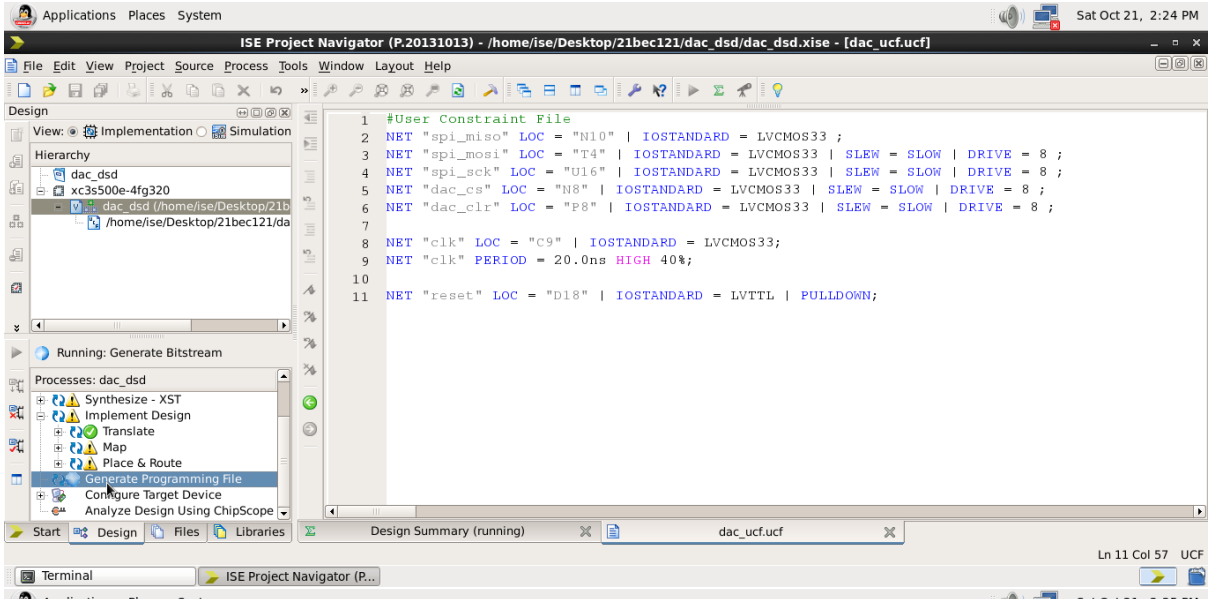

Applications Places System
Sat Oct 21, 2:20 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise - [dac_dsd.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
dac_dsd
xc3s500e-4fg320
dac_dsd (/home/ise/Desktop/21b
Running: Synthesis
Processes: dac_dsd
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
Design Summary (running)
dac_dsd.v
Ln 3 Col 15 Verilog
Terminal
ISE Project Navigator (P...
`timescale 1ns / 1ps
module dac_dsd(clk, reset, dac_cs, spi_sck, spi_mosi, spi_miso, dac_clr,
SPI_SS_B, AMP_CS, AD_CONV, SF_CE0, FPGA_INIT_B,command,data,address, send);
input clk;// 50 MHZ FPGA CLOCK,
input reset;
input spi_miso; // MASTER IN, SLAVE OUT
input [11:0]data; // DIGITAL TO BE GIVEN TO DAC MODULE
input [3:0]address; // DAC ADDRESS FOR A,B,C,D PIN
output reg send;
output reg dac_cs, spi_sck, spi_mosi, dac_clr; // SIGNAL ON DAC
output SPI_SS_B, AMP_CS, AD_CONV, SF_CE0, FPGA_INIT_B; // PERIPHERAL SIGNAL TO BE DISABLED
output reg [3:0]command; // COMMAND =4'B0011
reg [2:0]dac_state;// DAC STATES
reg [31:0]dac_out; //DAC INPUT={8'b don'tcare, 4'b command, 4'b address, 12'b data, 4'b don
reg [5:0]count=32; // 32 BIT COUNTER

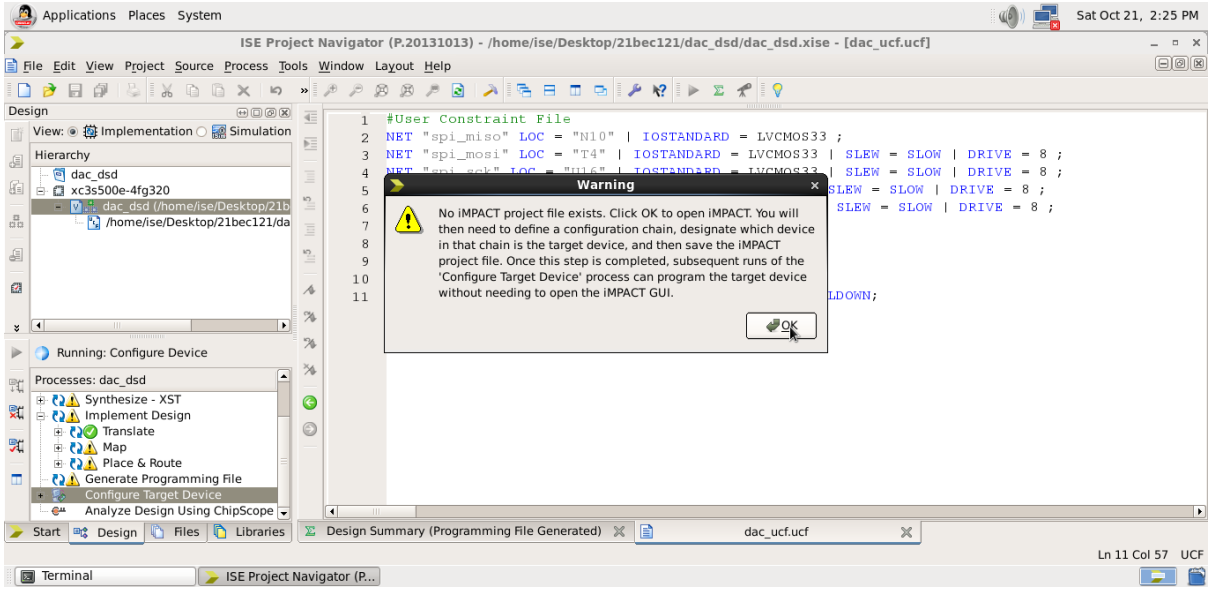

Applications Places System
Sat Oct 21, 2:22 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise - [dac_dsd.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
dac_dsd
xc3s500e-4fg320
dac_dsd (/home/ise/Desktop/21b
No Processes Running
Processes: dac_dsd
User Constraints
Synthesize - XST
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Start Design Files Libraries
Design Summary (Implemented)
dac_dsd.v
Ln 112 Col 1 Verilog
Terminal
ISE Project Navigator (P...

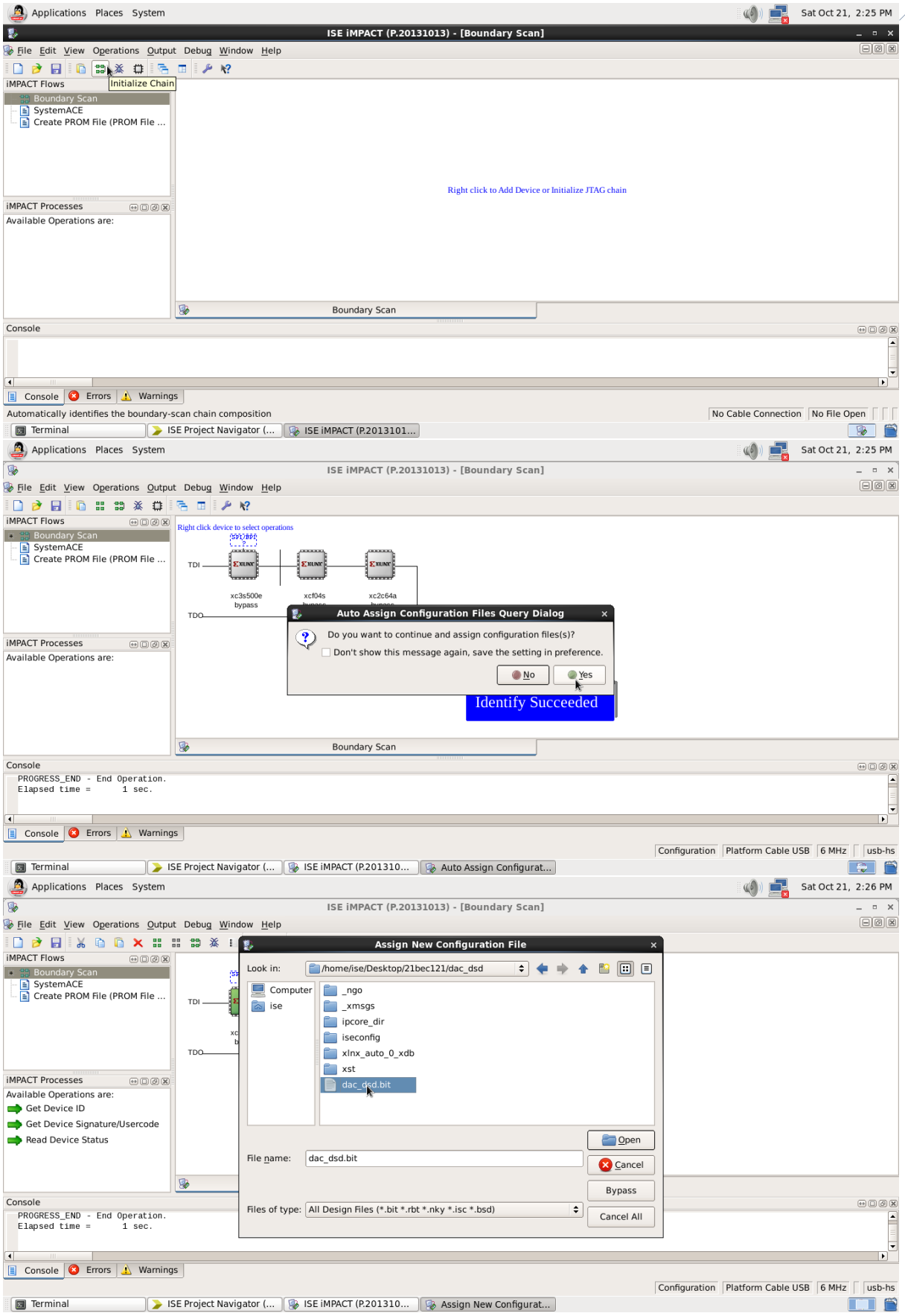

Applications Places System
Sat Oct 21, 2:22 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise - [dac_dsd.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
dac_dsd
xc3s500e-4fg320
dac_dsd (/home/ise/Desktop/21b
No Processes Running
Processes: dac_dsd
Synthesize - XST
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
Design Summary (Implemented)
dac_dsd.v
Ln 112 Col 1 Verilog
Terminal
ISE Project Navigator (P...

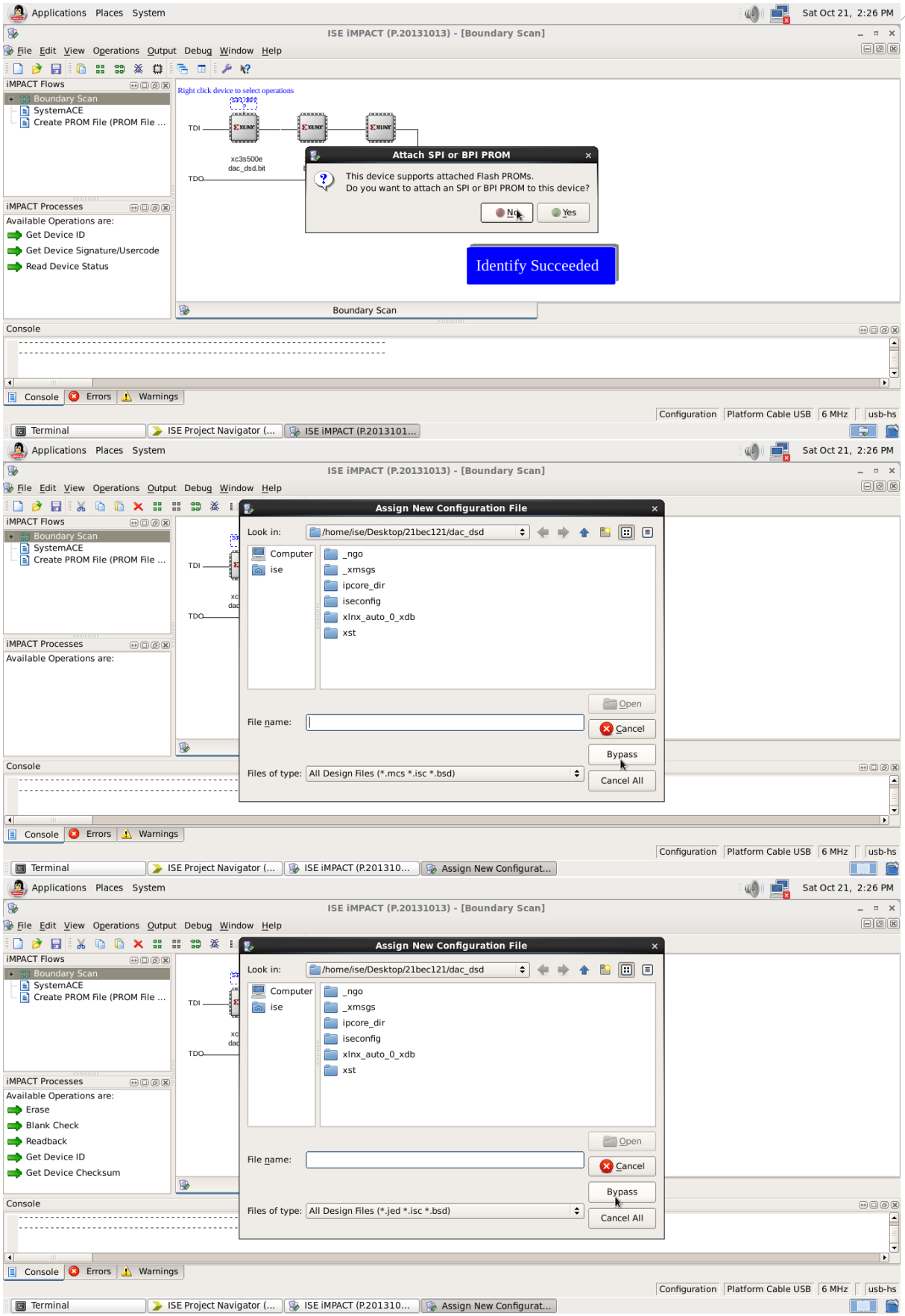

Applications Places System
Sat Oct 21, 2:22 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise - [dac_dsd.v]
File Edit View Project Source Process Tools Window Layout Help
New Source Wizard
Select Source Type
Select source type, file name and its location.
BMM File
ChipScope Definition and Connection File
Implementation Constraints File
IP (CORE Generator & Architecture Wizard)
MEM File
Schematic
System Generator Project
User Document
Verilog Module
Verilog Test Fixture
VHDL Module
VHDL Library
VHDL Package
VHDL Test Bench
Embedded Processor
File name:
dac_ucf
Location:
/home/ise/Desktop/21bec121/dac_dsd
Add to project
More Info
Next >
Cancel
Design
View: Implementation Simulation
Hierarchy
dac_dsd
xc3s500e-4fg320
dac_dsd (/home/ise/Desktop/21b
No Processes Running
No single design module is selected.
Design Utilities
Start Design Files Libraries
Design Summary (Implemented)
dac_dsd.v
Add a new source to the project
Ln 112 Col 1 Verilog
Terminal
ISE Project Navigator (P...

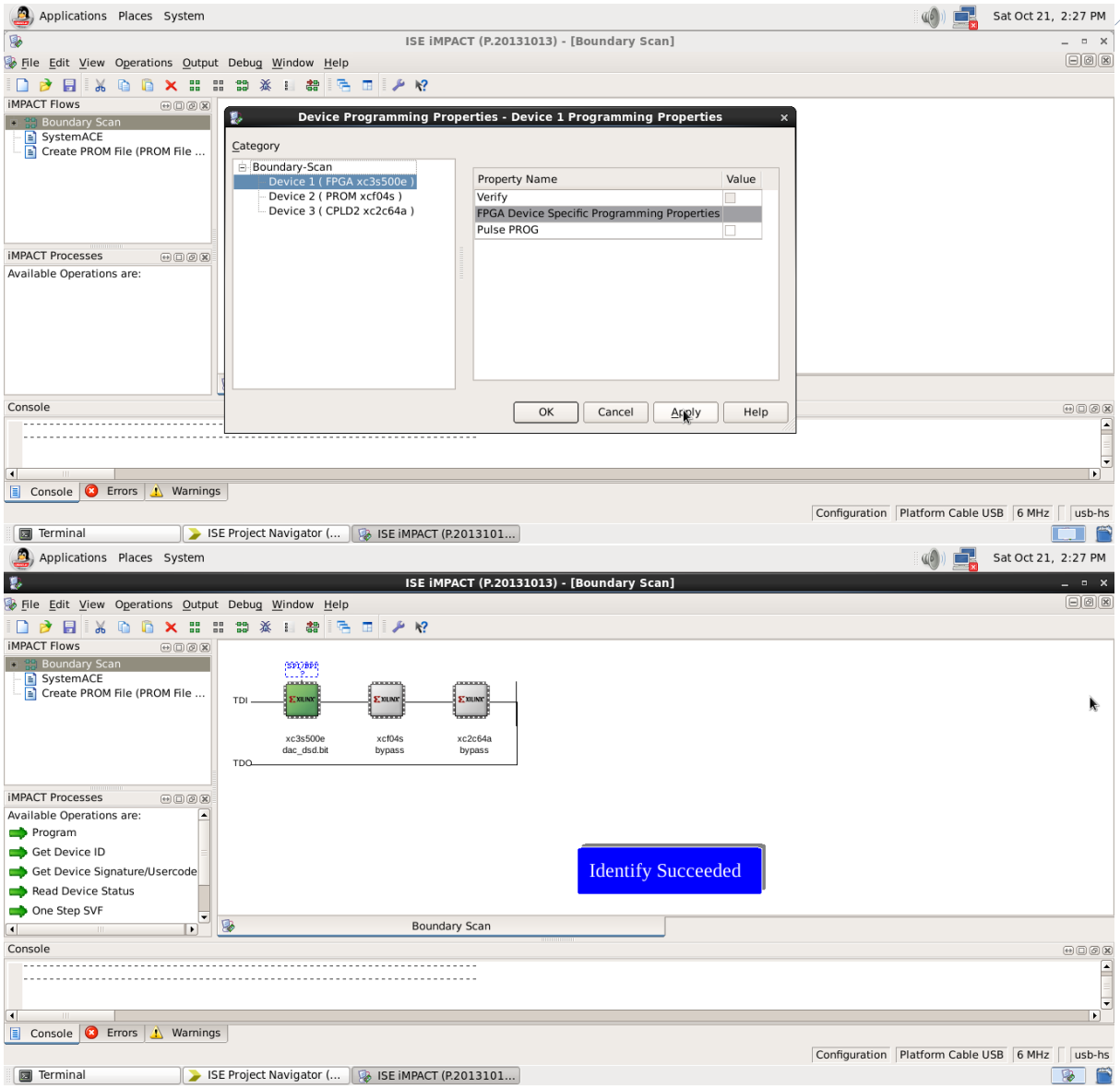

Applications Places System
Sat Oct 21, 2:23 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise - [dac_ucf.ucf*]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
dac_dsd
xc3s500e-4fg320
dac_dsd (/home/ise/Desktop/21b
/home/ise/Desktop/21bec121/da
No Processes Running
Processes: dac_ucf.ucf
User Constraints
Start Design Files Libraries
Design Summary
dac_ucf.ucf*
#User Constraint File
NET "spi_miso" LOC = "N10" | IOSTANDARD = LVCMOS33 ;
NET "spi_mosi" LOC = "T4" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "spi_sck" LOC = "U16" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "dac_cs" LOC = "N8" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "dac_clr" LOC = "P8" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "clk" LOC = "C9" | IOSTANDARD = LVCMOS33;
NET "clk" PERIOD = 20.0ns HIGH 40%;
NET "reset" LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN;
Ln 11 Col 57 UCF
Terminal
ISE Project Navigator (P...

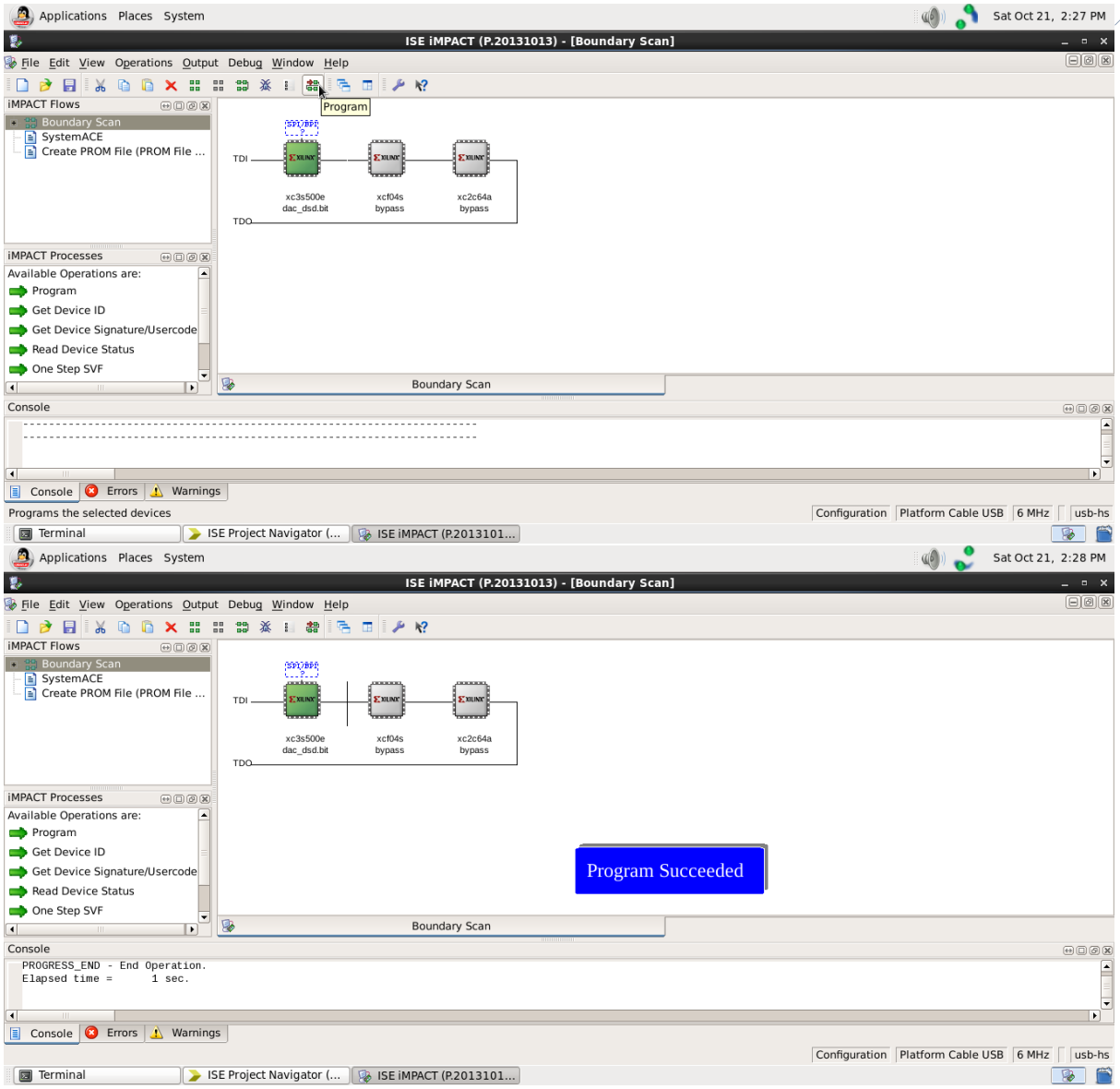

Applications Places System
Sat Oct 21, 2:24 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise - [dac_ucf.ucf]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
dac_dsd
xc3s500e-4fg320
dac_dsd (/home/ise/Desktop/21b
/home/ise/Desktop/21bec121/da
Running: Generate Bitstream
Processes: dac_dsd
Synthesize - XST
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
Design Summary (running)
dac_ucf.ucf
#User Constraint File
NET "spi_miso" LOC = "N10" | IOSTANDARD = LVCMOS33 ;
NET "spi_mosi" LOC = "T4" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "spi_sck" LOC = "U16" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "dac_cs" LOC = "N8" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "dac_clr" LOC = "P8" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "clk" LOC = "C9" | IOSTANDARD = LVCMOS33;
NET "clk" PERIOD = 20.0ns HIGH 40%;
NET "reset" LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN;
Ln 11 Col 57 UCF
Terminal
ISE Project Navigator (P...


Applications Places System
Sat Oct 21, 2:25 PM
ISE Project Navigator (P.20131013) - /home/ise/Desktop/21bec121/dac_dsd/dac_dsd.xise - [dac_ucf.ucf]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
dac_dsd
xc3s500e-4fg320
dac_dsd (/home/ise/Desktop/21b
/home/ise/Desktop/21bec121/da
Running: Configure Device
Processes: dac_dsd
Synthesize - XST
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
Design Summary (Programming File Generated)
dac_ucf.ucf
Warning
No iMPACT project file exists. Click OK to open iMPACT. You will then need to define a configuration chain, designate which device in that chain is the target device, and then save the iMPACT project file. Once this step is completed, subsequent runs of the 'Configure Target Device' process can program the target device without needing to open the iMPACT GUI.
OK
#User Constraint File
Ln 11 Col 57 UCF
Terminal
ISE Project Navigator (P...


Applications Places System
Sat Oct 21, 2:25 PM
ISE iMPACT (P.20131013) - [Boundary Scan]
File Edit View Operations Output Debug Window Help
Initialize Chain
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File ...
Right click to Add Device or Initialize JTAG chain
iMPACT Processes
Available Operations are:
Boundary Scan
Console
Console Errors Warnings
Automatically identifies the boundary-scan chain composition
No Cable Connection
No File Open
Terminal
ISE Project Navigator (...
ISE iMPACT (P.2013101...
Right click device to select operations
TDI
TDO
xc3s500e
bypass
xcf04s
xc2c64a
Auto Assign Configuration Files Query Dialog
Do you want to continue and assign configuration files(s)?
Don't show this message again, save the setting in preference.
No
Yes
Identify Succeeded
PROGRESS_END - End Operation.
Elapsed time = 1 sec.
Configuration
Platform Cable USB
6 MHz
usb-hs
Auto Assign Configurat...
Sat Oct 21, 2:26 PM
Assign New Configuration File
Look in:
/home/ise/Desktop/21bec121/dac_dsd
Computer
ise
_ngo
_xmsgs
ipcore_dir
iseconfig
xlnx_auto_0_xdb
xst
dac_dsd.bit
File name:
dac_dsd.bit
Open
Cancel
Bypass
Cancel All
Files of type:
All Design Files (*.bit *.rbt *.nky *.isc *.bsd)
Get Device ID
Get Device Signature/Usercode
Read Device Status
Assign New Configurat...


Applications Places System
Sat Oct 21, 2:26 PM
ISE iMPACT (P.20131013) - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File ...
Right click device to select operations
TDI
TDO
xc3s500e
dac_dsd.bit
Attach SPI or BPI PROM
This device supports attached Flash PROMs.
Do you want to attach an SPI or BPI PROM to this device?
No
Yes
Identify Succeeded
iMPACT Processes
Available Operations are:
Get Device ID
Get Device Signature/Usercode
Read Device Status
Boundary Scan
Console
Console Errors Warnings
Configuration Platform Cable USB 6 MHz usb-hs
Terminal ISE Project Navigator (... ISE iMPACT (P.2013101...
Assign New Configuration File
Look in: /home/ise/Desktop/21bec121/dac_dsd
Computer
ise
_ngo
_xmsgs
ipcore_dir
iseconfig
xlnx_auto_0_xdb
xst
File name:
Open
Cancel
Bypass
Cancel All
Files of type: All Design Files (*.mcs *.isc *.bsd)
ISE iMPACT (P.201310...
Assign New Configurat...
Erase
Blank Check
Readback
Get Device ID
Get Device Checksum
Files of type: All Design Files (*.jed *.isc *.bsd)


Applications Places System
Sat Oct 21, 2:27 PM
ISE iMPACT (P.20131013) - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File ...
iMPACT Processes
Available Operations are:
Device Programming Properties - Device 1 Programming Properties
Category
Boundary-Scan
Device 1 (FPGA xc3s500e)
Device 2 (PROM xcf04s)
Device 3 (CPLD2 xc2c64a)
Property Name
Value
Verify
FPGA Device Specific Programming Properties
Pulse PROG
OK
Cancel
Apply
Help
Console
Console
Errors
Warnings
Configuration
Platform Cable USB
6 MHz
usb-hs
Terminal
ISE Project Navigator (...
ISE iMPACT (P.2013101...
Applications Places System
Sat Oct 21, 2:27 PM
ISE iMPACT (P.20131013) - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File ...
iMPACT Processes
Available Operations are:
Program
Get Device ID
Get Device Signature/Usercode
Read Device Status
One Step SVF
TDI
TDO
xc3s500e
dac_dsd.bit
xcf04s
bypass
xc2c64a
bypass
Identify Succeeded
Boundary Scan
Console
Console
Errors
Warnings
Configuration
Platform Cable USB
6 MHz
usb-hs
Terminal
ISE Project Navigator (...
ISE iMPACT (P.2013101...



Sine wave generation:

Step 1: Define Your Requirements

Determine the resolution, frequency, and amplitude of the sine wave you want to generate. This information will help you set up your lookup table.

Step 2: Create a New Verilog Module

Start by creating a new Verilog module. This module will be responsible for generating the sine wave based on the phase information.

Step 3: Declare the Lookup Table

Inside your Verilog module, declare an array to hold the precomputed sine values. The size of this array depends on the resolution of your sine wave (e.g., 8-bit, 12-bit, etc.). In this example, we'll use a 12-bit resolution with 4096 entries.

Step 4: Fill the Lookup Table

In the `initial` block of your Verilog module, fill the lookup table with precomputed sine values. You can either calculate these values using a mathematical function or obtain them from a tool or reference.

Step 5: Implement the Sine Wave Generation Logic

Write the logic to select the appropriate sine value from the lookup table based on the phase information. This logic should output the sine value.

Step 6: Use the Lookup Table in Your Design

Integrate the Verilog module containing the lookup table and sine wave generation logic into your FPGA project. Connect the phase information to the module.

Step 7: Synthesize and Generate a Bitstream

Use Xilinx ISE or Vivado to synthesize your design and generate a bitstream file that can be loaded onto your Spartan-3E board.

Step 8: Load the Bitstream onto the FPGA

Load the generated bitstream file onto your Spartan-3E board using a JTAG programmer or other suitable methods.

Step 9: Test and Debug

Verify the functionality of your sine wave generator by observing the output. Use an oscilloscope or other measurement tools to ensure the sine wave meets your requirements.

Step 10: Fine-Tune if Necessary

If the generated sine wave doesn't meet your requirements, you may need to adjust the lookup table values, resolution, or other parameters in your design.

- Creating a lookup table is a practical and resource-efficient way to generate a sine wave on an FPGA. Ensure that your lookup table's size and accuracy match your application's needs, and remember to thoroughly test and debug your design.

Reference:

1. Spartan-3E FPGA Starter Kit Board User Guide by Xilinx
2. DAC Interfacing with FPGA using SPI
3. FPGA Dumping
4. Interfacing of DAC IC to generate Sine wave
5. FPGA Programming Tutorial: Sine Wave Generation