

DAC USING SPARTAN 3E

DSD Lab Project

Submitted to Prof. Hardik Joshi



21BEC095 Vedanti Patel
21BEC121 Mantra Solanki

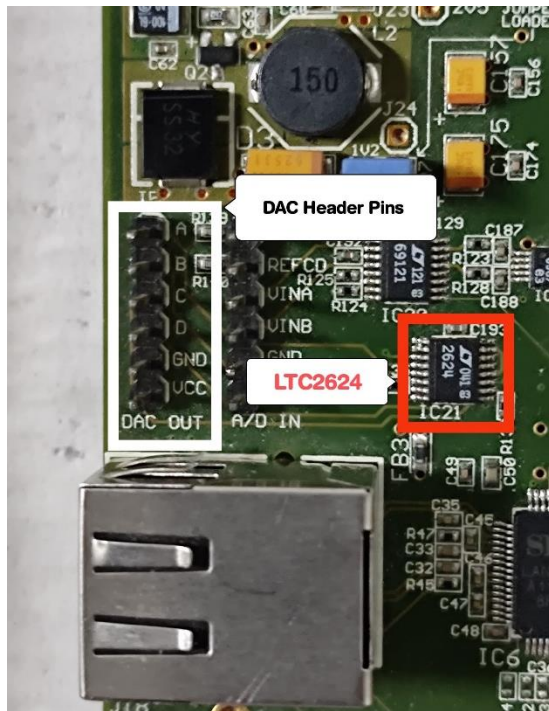


Figure 1

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.

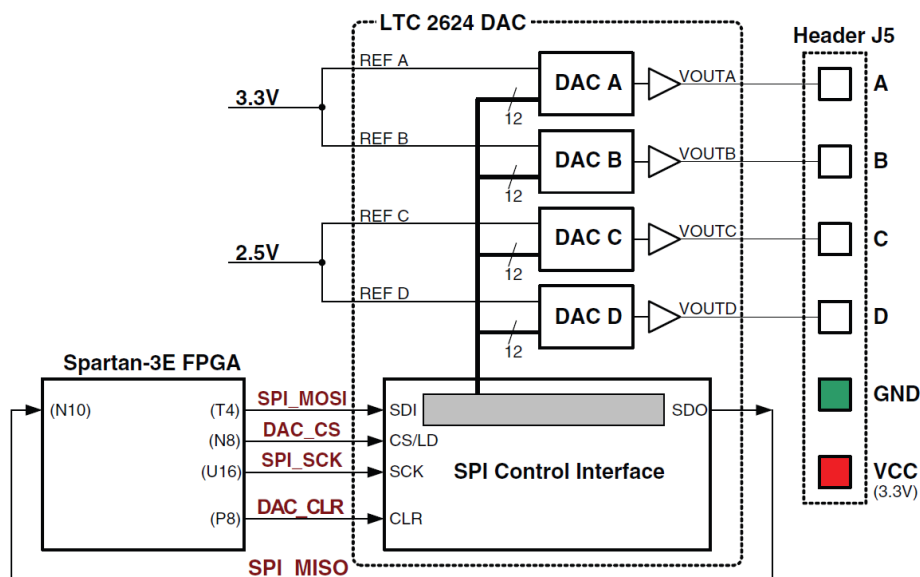


Figure 2

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_CEO	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.

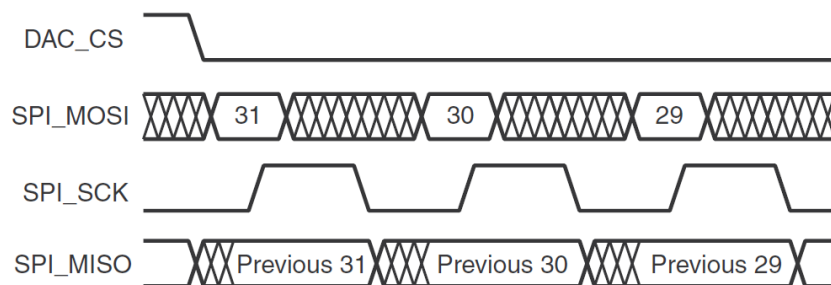


Figure 3

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.

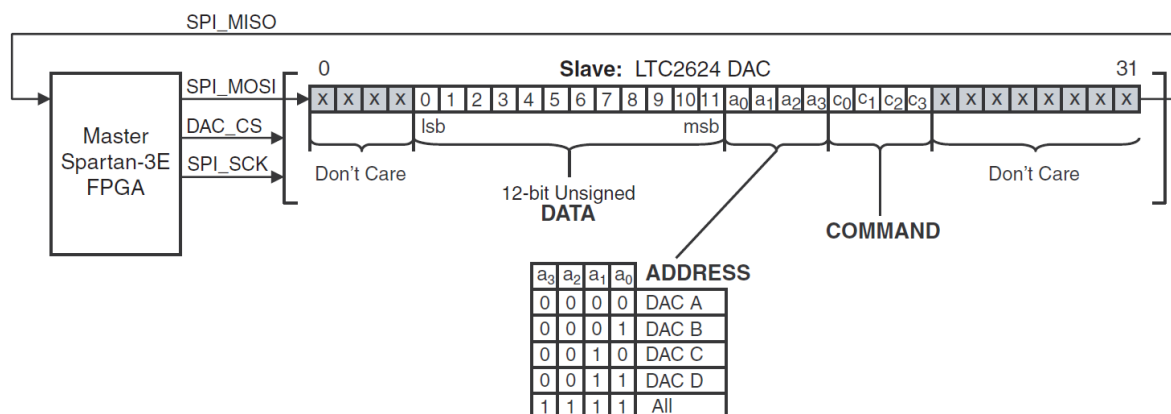


Figure 4

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 $\pm 5\%$ 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 $3.3V \pm 5\%$.

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 $2.5V \pm 5\%$.

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.

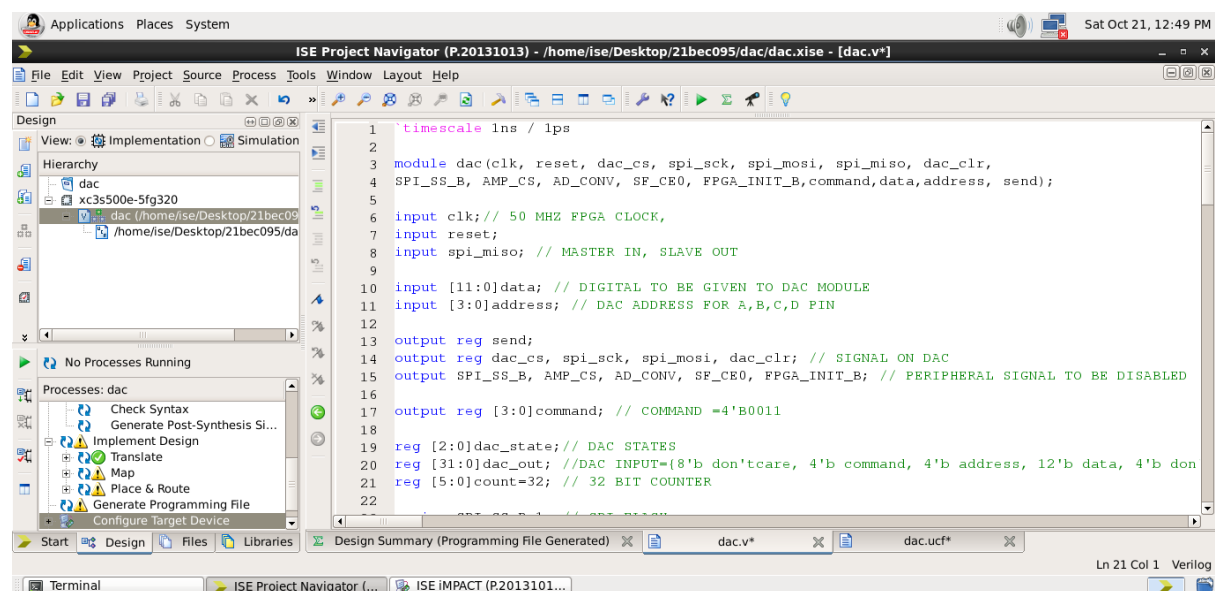


Figure 5

User Constraint File:

This file is made for pin assignments.

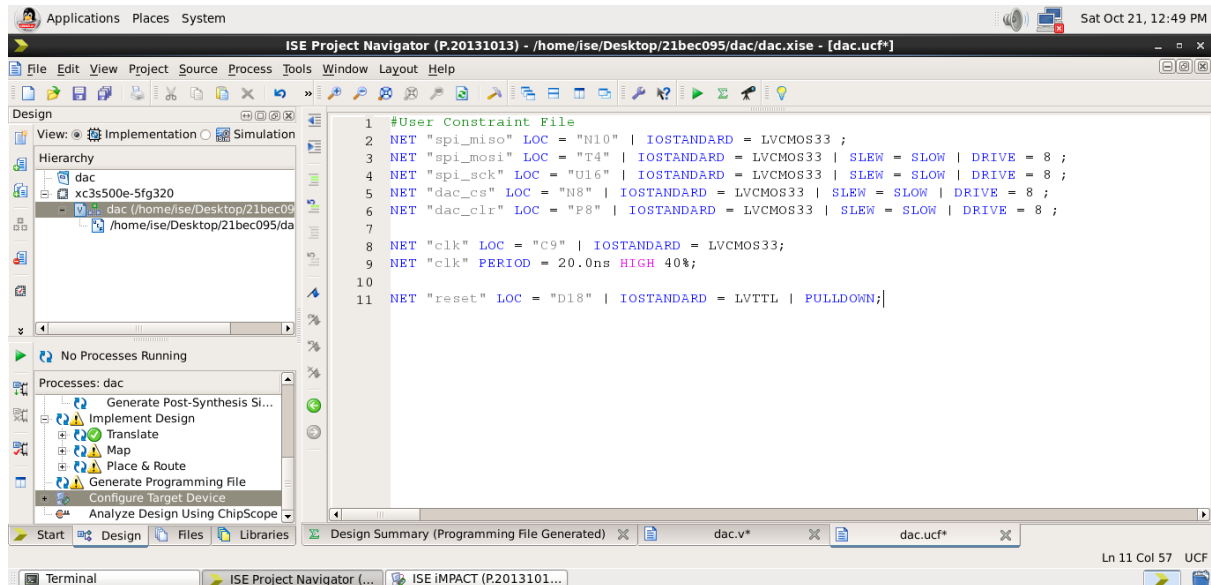


Figure 6

Dumping the code through ISE iMPACT:

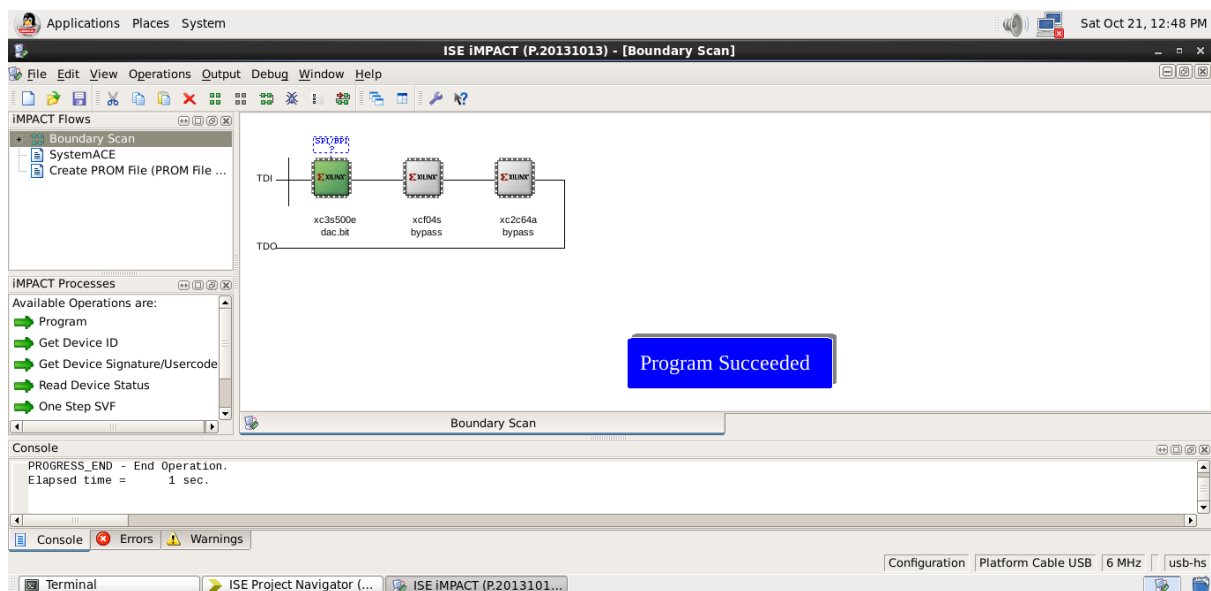


Figure 7

Output:

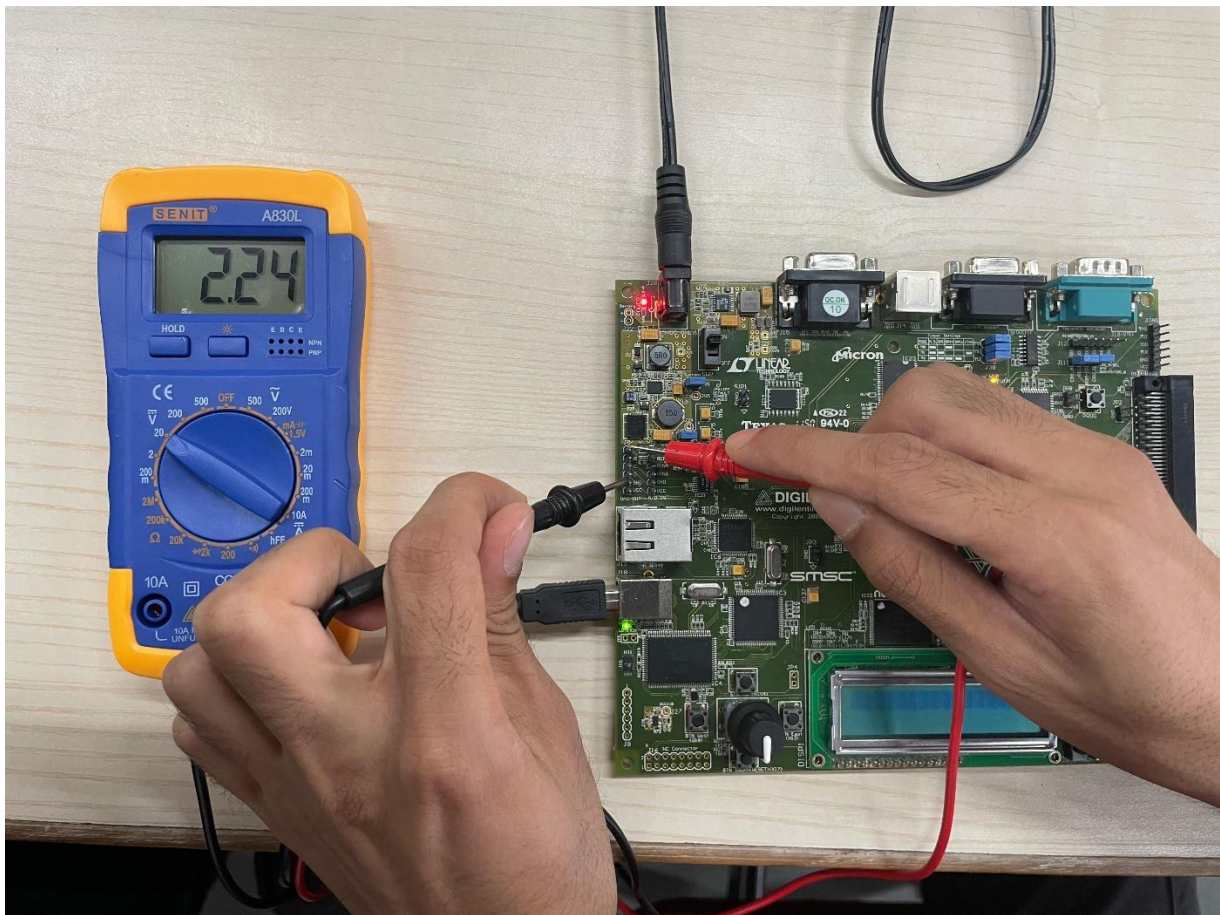
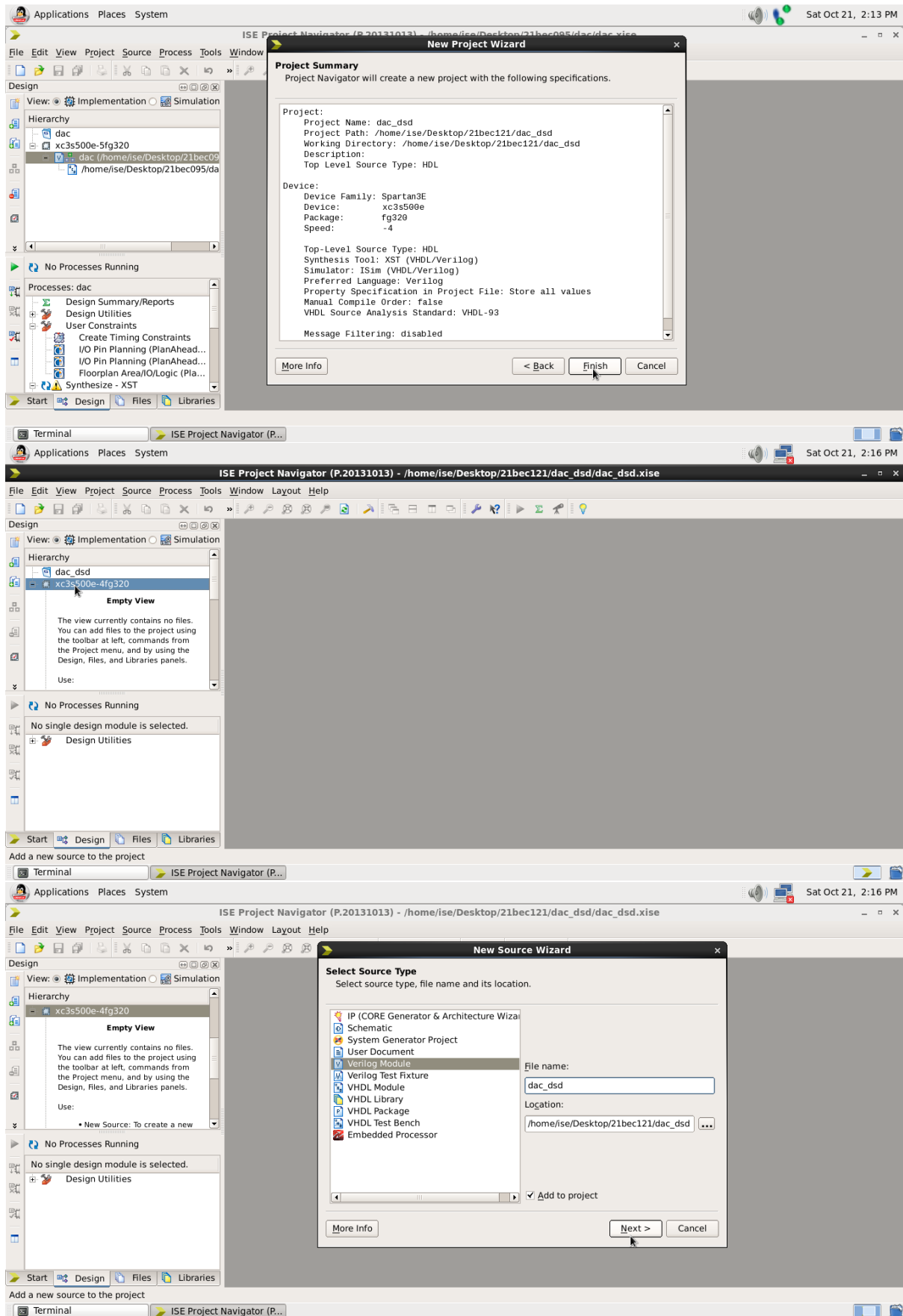


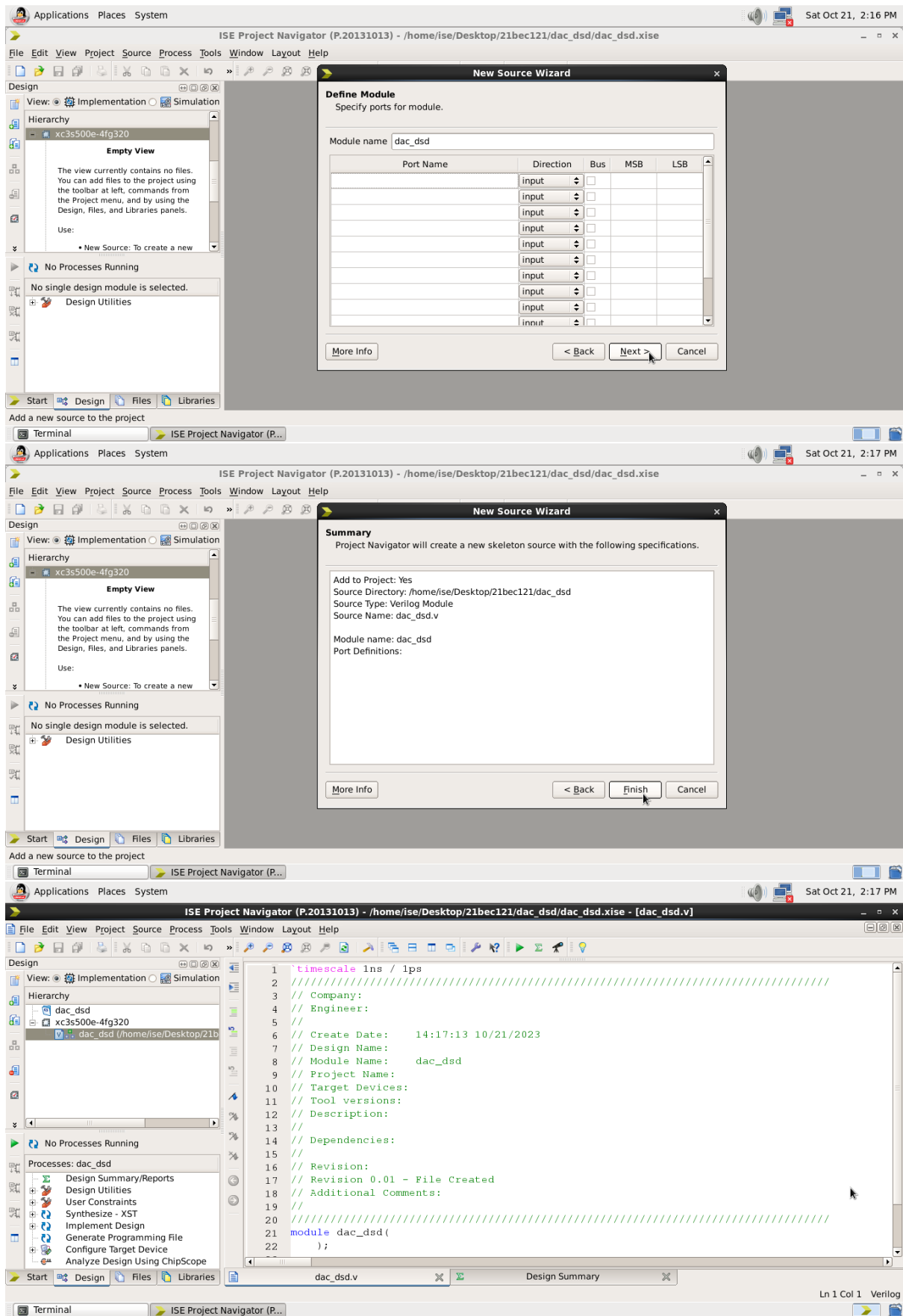
Figure 8

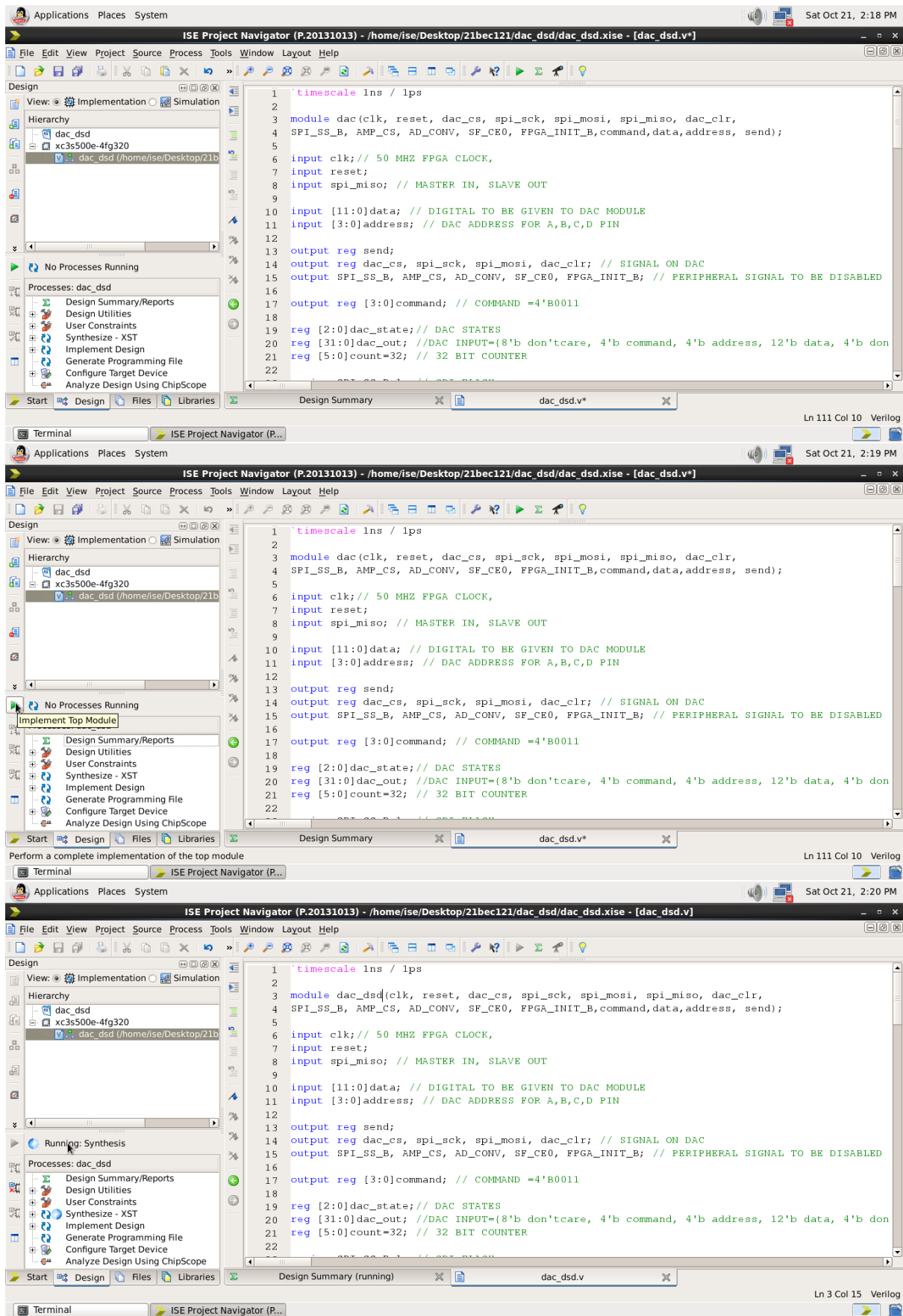
Reference:

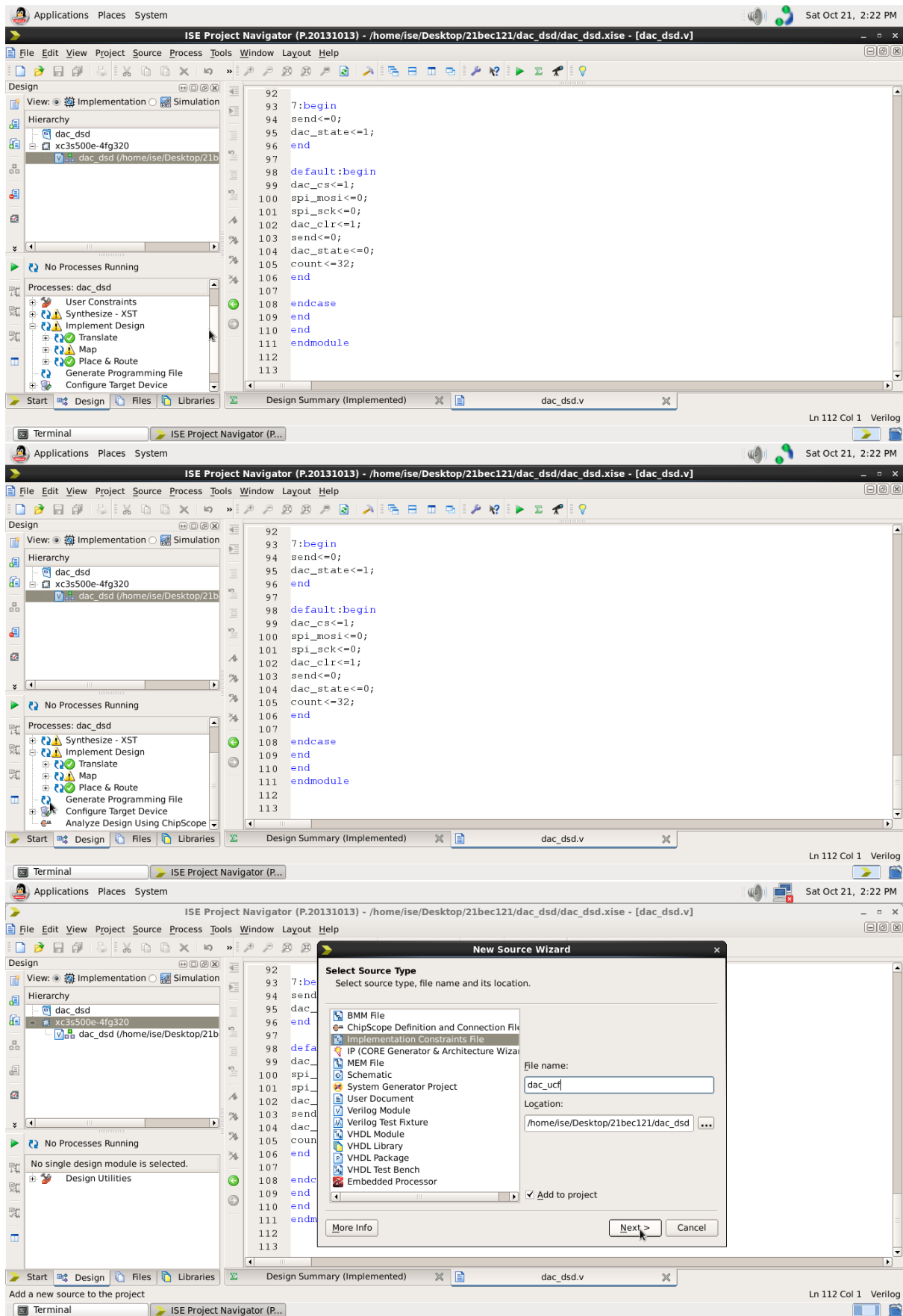
1. Spartan-3E FPGA Starter Kit Board User Guide by Xilinx
2. DAC Interfacing with FPGA using SPI by Component Byte (YouTube Video)
3. FPGA Dumping by Knowledge Unlimited (YouTube Video)

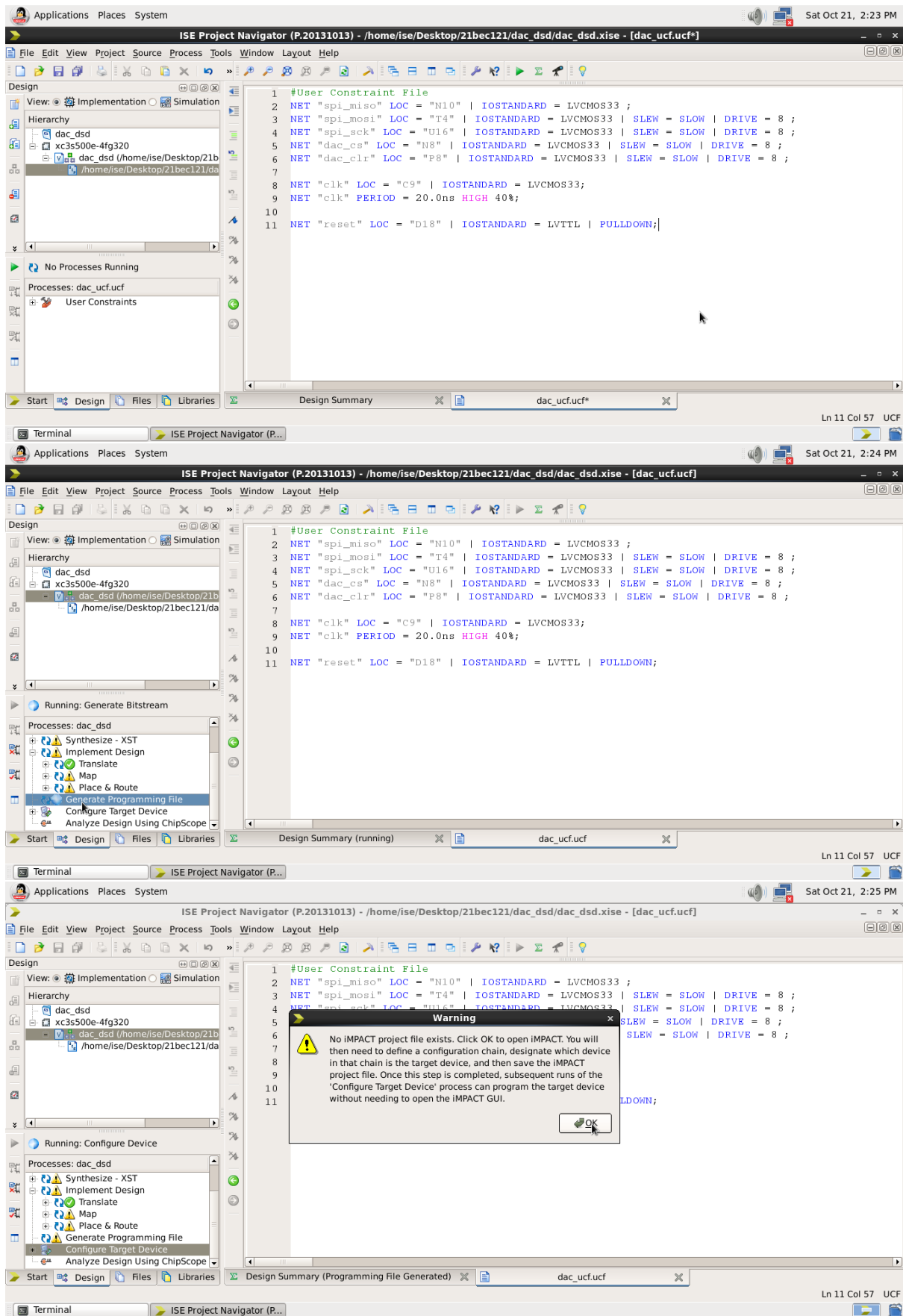
Steps:

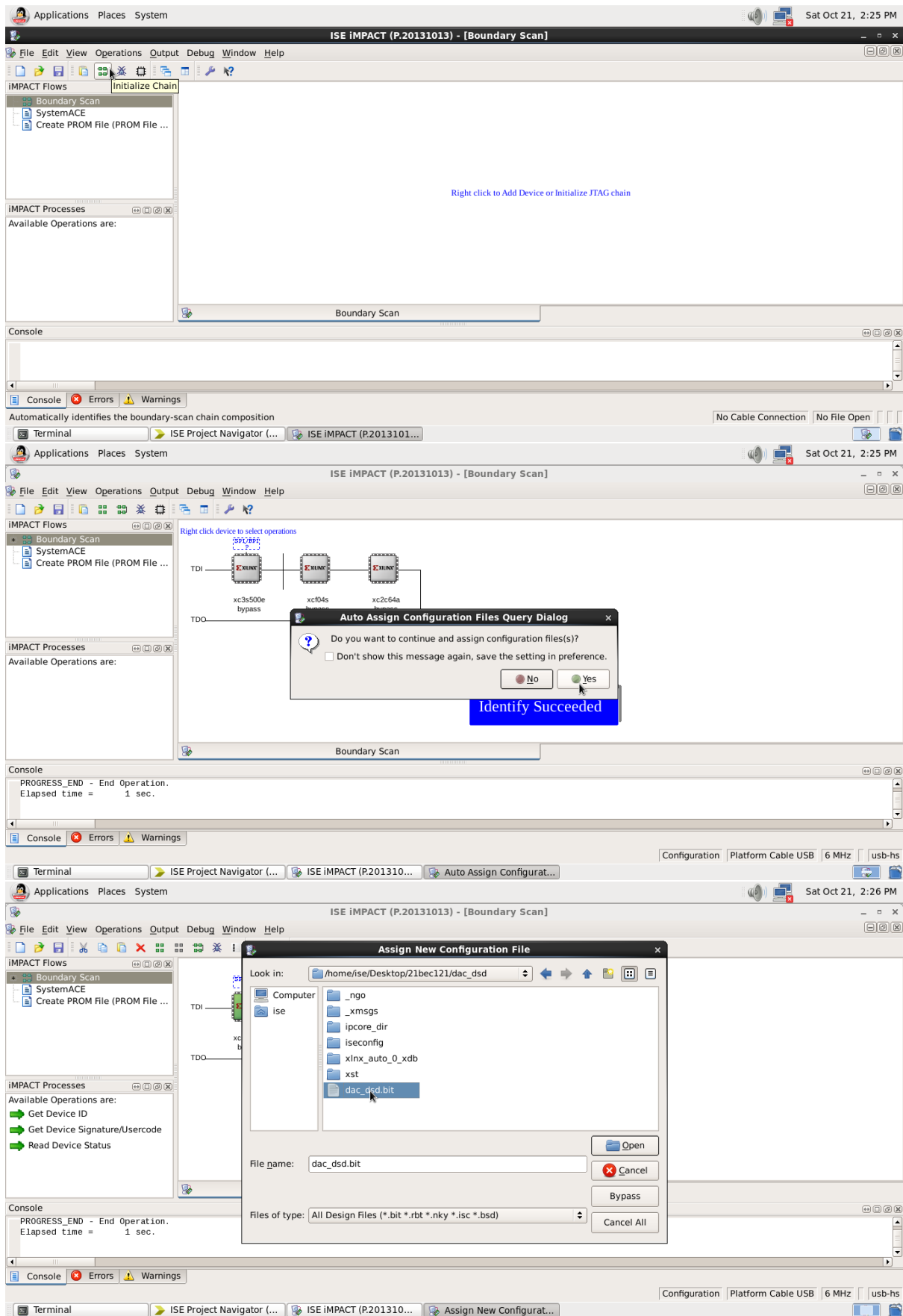


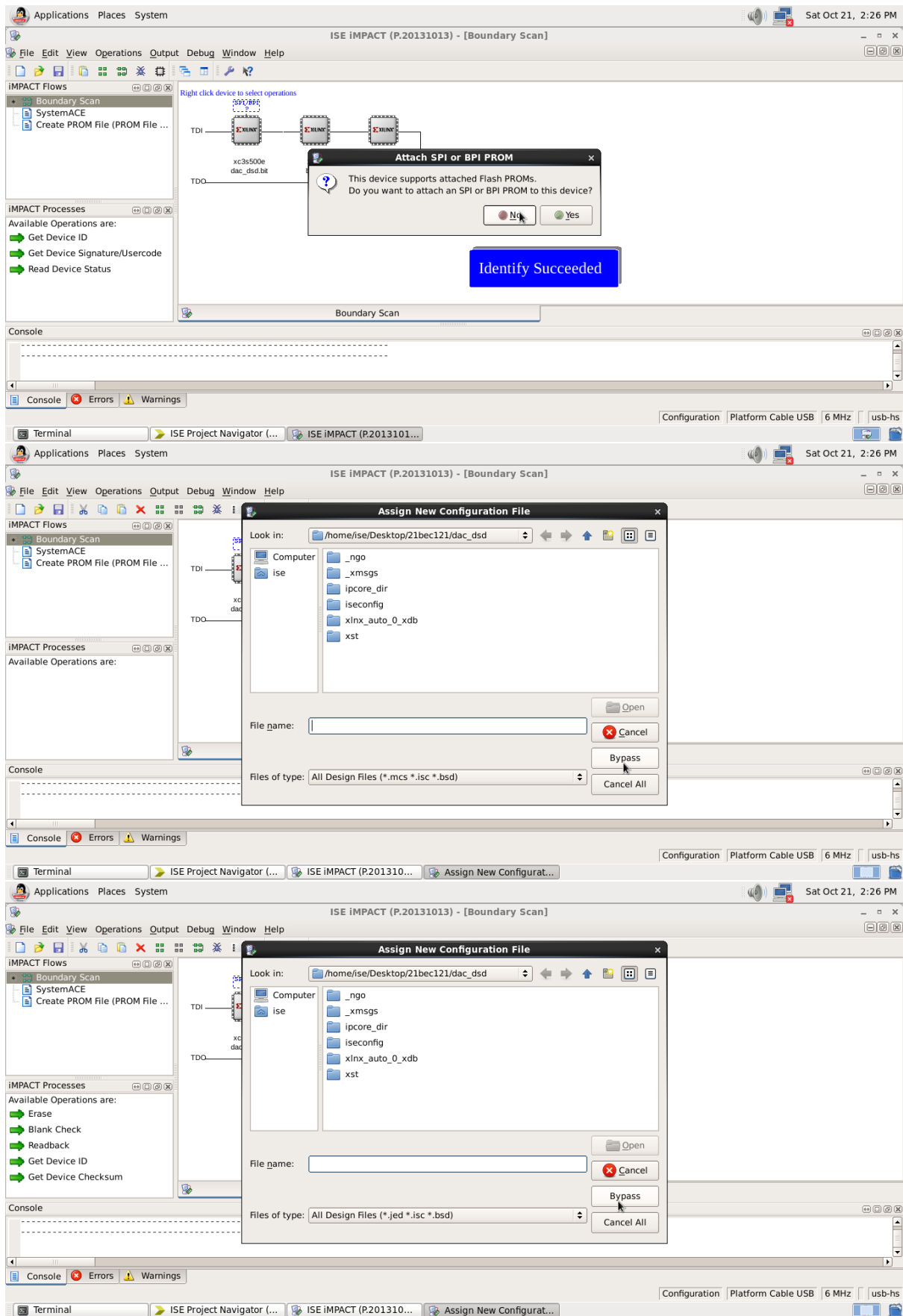












The image shows two screenshots of the ISE iMPACT (P.20131013) - [Boundary Scan] application window. The top screenshot shows the 'Device Programming Properties - Device 1 Programming Properties' dialog box. The bottom screenshot shows the 'Boundary Scan' diagram and a 'Identify Succeeded' message.

Top Screenshot: Device Programming Properties - Device 1 Programming Properties

The dialog box displays the following information:

- Category:** Boundary-Scan
 - Device 1 (FPGA xc3s500e)
 - Device 2 (PROM xc404s)
 - Device 3 (CPLD2 xc2c64a)
- Property Name:**
 - Verify
 - FPGA Device Specific Programming Properties
 - Pulse PROG
- Value:**
 - Verify: ☐
 - FPGA Device Specific Programming Properties: ☐
 - Pulse PROG: ☐

Buttons: OK, Cancel, Apply, Help

Bottom Screenshot: Boundary Scan Diagram

The diagram shows the following components and connections:

- TDI:** Connected to xc3s500e dac_dsd.bit
- xc3s500e dac_dsd.bit:** Connected to xc404s bypass
- xc404s bypass:** Connected to xc2c64a bypass
- TD0:** Connected to xc2c64a bypass

A blue box with the text 'Identify Succeeded' is overlaid on the diagram.

The bottom screenshot also shows the 'IMPACT Processes' panel with the following available operations:

- Program
- Get Device ID
- Get Device Signature/Usercode
- Read Device Status
- One Step SVF

