SPARTAN 3E TRAINER KIT

Model No : (VPTB - 10)

(Specially designed for Polytechnic Syllabus - Tamil Nadu)

User Manual

Version 1.0

Technical Clarification /Suggestion :

✍ / ☎

Technical Support Division,

Vi Microsystems Pvt. Ltd.,

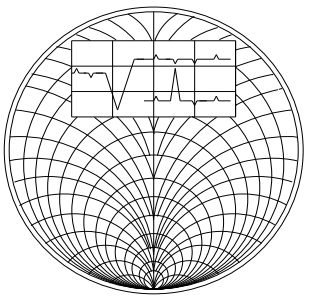
Plot No :75, Electronics Estate,

Perungudi, Chennai - 600 096, INDIA.

Ph: 91- 44-2496 1842, 91-44-2496 1852

Mail : sales@vimicrosystems.com,

Web : www.vimicrosystem.com



CONTENTS

CHAPTER - 1	Introduction	1
CHAPTER - 2	Clock Source	3
CHAPTER - 3	Switches & LEDs	5
CHAPTER - 4	Push Button Keys, Seven Segment Display	7
CHAPTER - 5	Relay & Buzzer	9
CHAPTER - 6	Stepper Motor & DC Motor Control	10
CHAPTER - 7	Character LCD Screen	12
CHAPTER - 8	Procedure to Work in Xilinx Software	13
CHAPTER - 9	Syllabus Programs in VHDL using VPTB-10	88
EXPERIMENT - 1a	VHDL Code for 4 Bit Addition	88
EXPERIMENT - 1b	VHDL Code for 4 Bit Subtraction	89
EXPERIMENT - 1c	VHDL Code for 4 Bit Multiplication	90
EXPERIMENT -1d	VHDL Code for 4 Bit Division	91
EXPERIMENT - 2	VHDL Code for Switch Led Interface	93
EXPERIMENT - 3	VHDL Code for 4x4 Matrix Keypad Interface	94
EXPERIMENT - 4a	VHDL Code for Buzzer Interface	97
EXPERIMENT - 4b	VHDL Code for Relay Interface	98

EXPERIMENT - 5	VHDL Code for Seven Segment Led Display Interface	99
EXPERIMENT - 6	VHDL Code for Clockwise Rotation of Stepper Motor	101
	VHDL Code for Anti-clockwise Rotation of Stepper Motor	103
EXPERIMENT - 7	VHDL Code for Traffic Light Controller	104
EXPERIMENT - 8	VHDL Code for 4 Bit up down Counter	107
	VHDL Code for Simulating 4 Bit up down Counter	109
EXPERIMENT - 9	VHDL Code for Displaying "Vi Microsystems" in the LCD	110
EXPERIMENT - 10	VHDL Code to Generate PWM signal for DC Motor Control	111
EXPERIMENT - 11	VHDL Code for Multiplexer 4:1	114
	VHDL Code for Demultiplexer 1:4	115
EXPERIMENT - 12	VHDL Code for Decoder 3:8	117
	VHDL Code for Encoder 8:3	118
	VHDL Code for 4 Bit Serial in Parallel out Shift Register	120
CHAPTER-10	Syllabus Programs in Verilog using VPTB-10	122
EXPERIMENT - 1a	Verilog Code for 4 Bit Addition	122
EXPERIMENT - 1b	Verilog Code for 4 Bit Subtraction	123
EXPERIMENT - 1c	Verilog Code for 4 Bit Multiplication	124
EXPERIMENT -1d	Verilog Code for 4 Bit Division	125
EXPERIMENT - 2	Verilog Code for Switch Led Interface	127
EXPERIMENT - 3	Verilog Code for 4x4 Matrix Keypad Interface	128
EXPERIMENT - 4a	Verilog Code for Buzzer Interface	130
EXPERIMENT - 4b	Verilog Code for Relay Interface	131

EXPERIMENT - 5	Verilog Code for Seven Segment Led Display Interface	132
EXPERIMENT - 6a)	Verilog Code for Clockwise Rotation of Stepper Motor	133
EXPERIMENT - 6b)	Verilog Code for Anti-clockwise Rotation of Stepper Motor	135
EXPERIMENT - 7	Verilog Code for Traffic Light Controller	136
EXPERIMENT - 8	Verilog Code for 4 Bit up down Counter	139
	Verilog Code for Simulating 4 Bit up down Counter	141
EXPERIMENT - 9	Verilog Code for Displaying "Vi Microsystems" in the LCD	142
EXPERIMENT - 10	Verilog Code to Generate PWM signal for DC Motor Control	144
EXPERIMENT - 11	Verilog Code for Multiplexer 4:1	147
	Verilog Code for Demultiplexer 1:4	148
EXPERIMENT - 12	Verilog Code for Decoder 3:8	150
	Verilog Code for Encoder 8:3	151
	Verilog Code for 4 Bit Serial in Parallel out Shift Register	153

CHAPTER - 1

INTRODUCTION

The Vi Microsystems Xilinx Spartan-3E FPGA Trainer Kit is a demonstration platform intended for you to become familiar with the new features and availability of the Spartan-3E FPGA family. This Kit provides a low-cost, easy-to-use development and evaluation platform for Spartan-3E FPGA designs.

KEY COMPONENTS AND FEATURES

Figure-1 shows the Spartan-3E Low Cost board block diagram, which includes the following components and features:

* 250,000-gates Xilinx Spartan-3E **XC3S250E** FPGA in a 208-Plastic Quad Flat Pack package (**XC3S250E-PQ208**)
  - # 5,508 logic cell equivalents
  - # Twelve 18K-bit block RAMs (216K bits)
  - # Twelve 18x18 pipelined hardware multipliers
  - # Four Digital Clock Managers (DCMs)
* One 20 pin header to interface VLSI based experiment modules
* One 26 pin VLIM header to interface VLSI based experiment modules
* 16 input Slide switches with LED indication
* 16 output Light Emitting Diodes(LEDs)
* One 5V Buzzer and Relay
* 4 Digit seven segment LED display
* 4 x 4 matrix keypad interface
* One reset switch
* On Board programmable oscillator (3 to 100 MHz)
* 16x2 Alphanumeric LCD
* On Board 2 Mbit configuration Flash PROM XCF02S

BLOCK DIAGRAM

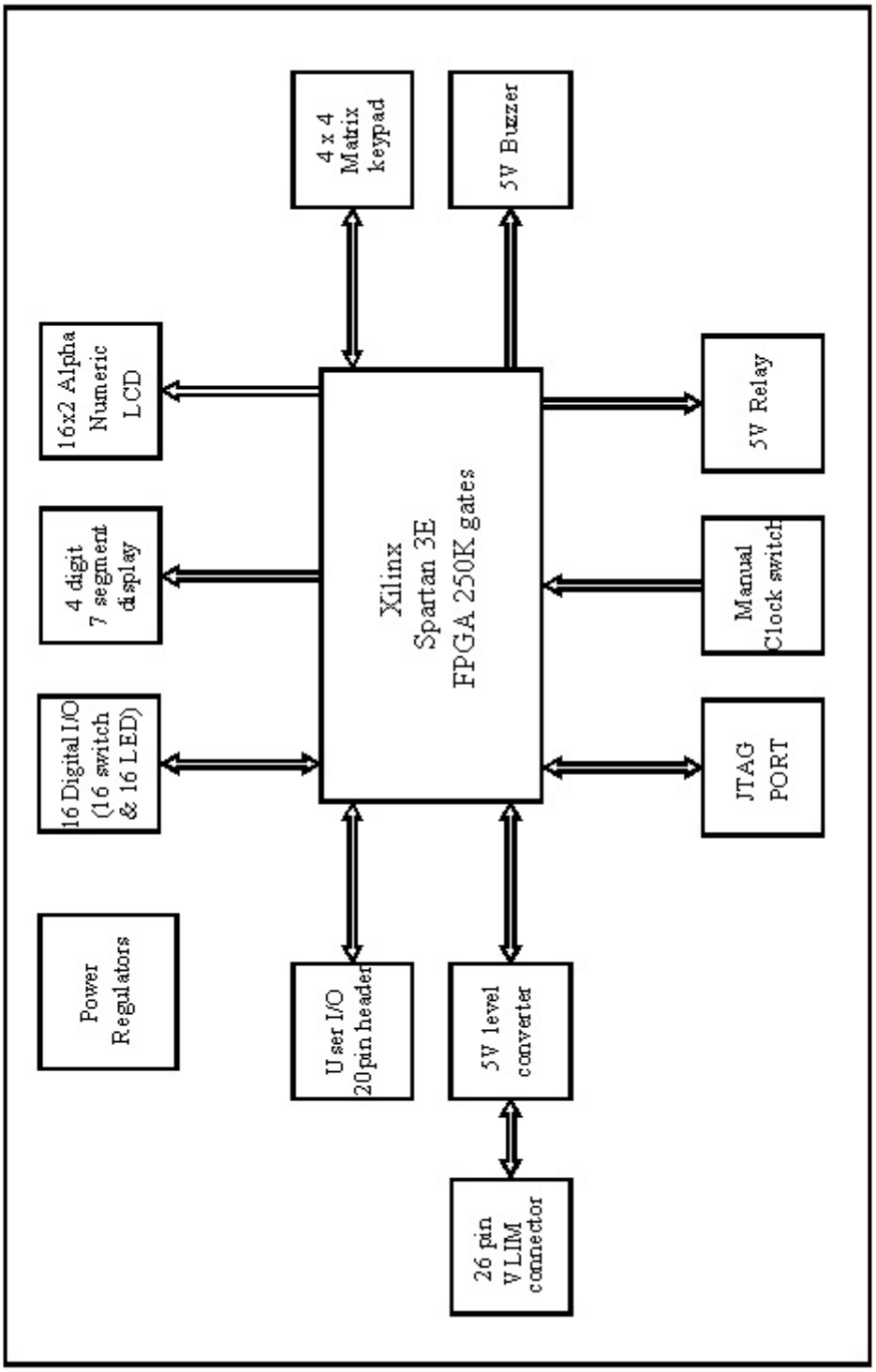


Figure -1

CHAPTER - 2

CLOCK SOURCE

Spartan3E FPGA works in different Clock frequencies. User can use any frequencies for their application as given below,

PLL Oscillator Settings

Default Factory settings = 20MHz

For PLL, ICS525-01 . Select the clock settings as per the PLL in the onboard

0 = Shorted
1 = Open

ICS525-01 Clock Table Between 1 to 100 MHZ
Table -1

	1MHz	3.6864 MHZ	4MHz	**20MHz**	24MHz	25.175 MHz	48MHz	66MHz	80MHz	100MHz
S2	0	0	0	**0**	1	1	0	0	0	0
S1	0	0	0	**1**	0	1	0	0	0	0
S0	0	0	0	**0**	0	1	1	1	1	1
R6	0	0	0	**0**	0	1	0	0	0	0
R5	1	1	0	**0**	0	0	0	0	0	0
R4	0	1	0	**0**	0	0	0	0	0	0
R3	1	0	1	**0**	0	1	0	1	0	0
R2	1	0	0	**0**	0	0	0	0	0	0
R1	1	0	1	**1**	1	1	1	0	0	0
R0	0	1	0	**0**	0	0	1	0	1	1
V8	0	0	0	**0**	0	1	0	0	0	0
V7	0	0	0	**0**	0	0	0	0	0	0
V6	0	0	0	**0**	0	0	0	0	0	0
V5	0	1	0	**0**	0	0	0	0	0	0
V4	0	0	0	**0**	0	1	0	1	0	0
V3	0	0	0	**0**	0	0	0	1	0	0
V2	1	1	1	**1**	1	1	1	0	1	1
V1	0	1	0	**0**	0	1	0	0	0	1
V0	0	1	0	**0**	0	1	0	1	0	1

For any other frequency Use the online ICS525 calculator at **http://www.idt.com/?app=calculators&device=525_01** or alternatively, the output of the ICS525-output frequency in between 1MHz to 100MHz use the following formula.

$$\text{CLK frequency} = \text{Input frequency} * 2 \frac{(\text{VDW} + 8)}{(\text{RDW} + 2)(\text{OD})}$$

Where,

Reference Divider Word (RDW)	=	1 to 127 (0 is not permitted)
VCO Divider Word (VDW)	=	4 to 511 (0,1,2,3 are not permitted)
Output Divider (OD)	=	Values below

EXAMPLE

To generate 12 MHz, assume Crystal frequency or Input frequency is 20MHz.

In general,

$$\text{Clock Frequency} = \text{Input Frequency} \times 2 \times \frac{\text{VDW} + 8}{(\text{RDW} + 2)\ (\text{OD})}$$

$$\text{Clock Frequency} = 20\ \text{MHz} \times 2 \times \frac{4 + 8}{(18 + 2)(2)} = \mathbf{12\ MHz}$$

ICS525-01 Output Divider and Maximum Output Frequency Table

Table:2

S2	S1	S0	CLK	Max. Output Frequency (MHz)			
Pin5	Pin4	Pin3	Output Divider	VDD = 5V		VDD = 3.3V	
				0 - 70^{0}C	-40 to 85 ^{0}C	0 - 70^{0}C	-40 to 85 ^{0}C
0	0	0	10	26	23	18	16
0	0	1	2	160	140	100	90
0	1	0	8	40	36	25	22
0	1	1	4	80	72	50	45
1	0	0	5	50	45	34	30
1	0	1	7	40	36	26	23
1	1	0	9	33.3	30	20	18
1	1	1	6	53	47	27	24

* VCO Divider Word (VDW) = V8 to V0 = 000000100 = 4 (Decimal);
* Reference Divider Word (RDW) = R6 to R0 = 0010010 = 18 (Decimal);
* Output Divider (OP) = 2 (from the table given above)

CHAPTER - 3

SWITCHES & LEDS

3.1. Power Switch (Slide Switch(SW1))

The Spartan-3E Trainer Kit has a slide power switch. Moving the power switch Up for Power ON and down for power OFF.

3.2. Configuration Switch (SW3)

The Spartan-3E Trainer Kit has a push button Switch (named as CONFIG) to Configure the FPGA from Xilinx Serial Flash PROM.

3.3. Input Switches

The Spartan-3E Trainer Kit has 16 nos of slide switches with led indication for giving inputs to the FPGA i/o lines. The slide switches are located in the bottom corner of the board and are labeled as I/P1 through I/P16. Switch I/P1 is the left-most switch, and I/P16 is the right-most switch**.** When in the UP or ON position, a switch connects the FPGA pin to 3.3V, a logic High.When DOWN or in the OFF position, the switch connects the FPGA pin to ground, a logicLow. The switches typically exhibit about 2 ms of mechanical bounce. There is no active debouncing circuitry, although such circuitry could easily be added to the FPGA design programmed on the board.

Slide Switch connections with FPGA

SWITCHES	I/P1	I/P2	I/P3	I/P4	I/P5	I/P6	I/P7	I/P8
FPGA Pin	P71	P72	P91	P101	P110	P118	P124	P130

SWITCHES	I/P9	I/P10	I/P11	I/P12	I/P13	I/P14	I/P15	I/P16
FPGA Pin	P136	P142	P148	P154	P159	P169	P194	P174

Output LEDs

The Spartan-3E Trainer Kit board has sixteen individual surface-mount LEDs located immediately above the slide switches. The LEDs are labeled O/P1 through O/P16. O/P1 is the left-most LED, O/P16 the right-most LED.

Each LED has one side connected to ground and the other side connected to a pin on the Spartan-3E device via a 270Ω current limiting resistor. To light an individual LED, drive the associated FPGA control signal High.

LED connections with FPGA

LED	O/P1	O/P2	O/P3	O/P4	O/P5	O/P6	O/P7	O/P8
FPGA Pin	P102	P106	P107	P108	P109	P112	P113	P115

LED	O/P9	O/P10	O/P11	O/P12	O/P13	O/P14	O/P15	O/P16
FPGA Pin	P116	P119	P120	P122	P123	P126	P127	P128

CHAPTER - 4
PUSH BUTTON KEYS, SEVEN SEGMENT DISPLAY

PUSH BUTTON SWITCHES

★ The VPTB-10 board has 16 momentary-contact push button switches.

★ The 16 push button keys are arranged in 4 x 4 matrix format, and multiplexed by 8 IO pins to FPGA.

PUSH BUTTON Connections with FPGA

There are 4 output lines named a3 to a0 and 4 input lines are available named as b3 to b0. These lines are must be pulled up in the ucf file for proper working.

PUSH BUTTON	a<3>	a<2>	a<1>	a<0>	b<3>	b<2>	b<1>	b<0>
FPGA Pin	P138	P135	P133	P129	P204	P137	P134	P132

SEVEN SEGMENT DISPLAY

The VPTB-10 board has a four-character, seven segment LED display controlled by FPGA user-I/O pins. Each individual character has a separate common cathode control input. The LED control signals are connected to the individual line of the FPGA I/O.

Table 4.1

Character	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	0	0	1	1

A	1	1	1	0	1	1	1
B	0	0	1	1	1	1	1
C	1	0	0	1	1	1	0
D	0	1	1	1	1	0	1
E	1	0	0	1	1	1	1
F	1	0	0	0	1	1	1

SEVEN SEGMENT Connections with FPGA

DISPLAY 1	seg-g	seg-f	seg-e	seg-d	seg-c	seg-b	seg-a	seg-com	seg-dp
FPGA Pin	P205	P203	P177	P172	P168	P200	P202	P171	P167

DISPLAY 2	seg-g	seg-f	seg-e	seg-d	seg-c	seg-b	seg-a	seg-com	seg-dp
FPGA Pin	P199	P197	P165	P164	P162	P192	P196	P163	P161

DISPLAY 3	seg-g	seg-f	seg-e	seg-d	seg-c	seg-b	seg-a	seg-com	seg-dp
FPGA Pin	P189	P187	P160	P153	P151	P185	P186	P152	P150

DISPLAY 4	seg-g	seg-f	seg-e	seg-d	seg-c	seg-b	seg-a	seg-com	seg-dp
FPGA Pin	P181	P180	P147	P146	P144	P178	P179	P145	P140

CHAPTER-5
RELAY & BUZZER

BUZZER & RELAY

The VPTB-10 contains Buzzer & Relay circuit, and both works in 5V DC Voltage .

A buzzer or beeper is a signalling device, usually electronic, typically used in automobiles, household appliances such as a microwave oven, or game shows.

Buzzer generates different tone Generator. Relay is used as On/Off switch depending upon user needs.

BUZZER Connections with FPGA

SIGNAL	buzzer
FPGA Pin	P190

RELAY Connections with FPGA

SIGNAL	relay
FPGA Pin	P193

CHAPTER - 6
STEPPER MOTOR & DC MOTOR CONTROL

STEPPER MOTOR

The VPTB-10 Contains Stepper Motor Interface Connector to Control the Stepper Motor Speed and it also controls the Step Angle of the Motor.

The Connector has the option to connect both 5V & 12V Stepper Motor, for this we have to set the following settings,

Jumper Details

For ON-BOARD

* To control the 5V Motor, Connect **Jumper J13** in Upward Direction.

* To control the 12V Motor, Connect **Jumper J13** in Downward Direction and give External supply 12V in P9 connector.

Stepper motor connections with FPGA

SIGNAL	M3	M2	M1	M0
FPGA Pin	P82	P78	P77	P76

For VVSI-32

* Connect external io connector P6 (VPTB-10) to the external io connector P3 (VVSI-32).

* Connect the 5 pin RMC cable of the stepper motor to the 5 pin connector P4(VVSI-32) and the **Jumpers J5** of VPTB-10 must be kept in **+5V** position & **J2** of VVSI-32 must be kept in **MVCC** position.

Stepper motor connections with FPGA

SIGNAL	M3	M2	M1	M0
FPGA Pin	P3	P2	P4	P5

DC MOTOR

The VPTB-10 contains the DC Motor Interface Connector to control the speed of the DC Motor by varying the duty cycle of the PWM which is read using the Push buttons switches of the VPTB-10 board.

Jumper Details

For ON-BOARD

* To control the Motor by on-board VCC, Connect **Jumper J1** in Upward Direction.

* To control the Motor by external VCC, Connect **Jumper J1** in downward Direction and the external supply can be given to the connector P3.

* Connect the 2 pin RMC cable of the DC motor to the 2 pin RMC connectors P1(VPTB-10).

DC motor connections with FPGA

SIGNAL	DC1	DC2
FPGA Pin	P139	P206

For VVSI-32

* Connect external io connector P6 (VPTB-10) to the external io connector P3 (VVSI-32).

* The **Jumpers J5** of VPTB-10 must be kept in +5V position and **J1** of VVSI-32 must be kept in MVCC position.

* Connect the 2 pin RMC cable of the DC motor to the 2 pin RMC connectors P5(VVSI- 32).

* Press sw13 of the matrix keypad for increasing the duty cycle and sw14 of the keypad for decreasing the duty cycle of the dc motor.

DC motor connections with FPGA

SIGNAL	DC1	DC2
FPGA Pin	P9	P8

CHAPTER - 7
CHARACTER LCD SCREEN

Figure - 4

Once mastered, the LCD is a practical way to display a variety of information using standard ASCII and custom characters. However, these displays are not fast. Scrolling the display at half-second intervals tests the practical limit for clarity.

LCD Connections with FPGA

LCD	D0	D1	D2	D3	D4	D5	D6	D7	RS	CS
FPGA Pin	P90	P93	P94	P96	P97	P98	P99	P100	P83	P89

Voltage Compatibility

The character LCD is power by +5V. The FPGA I/O signals are powered by 3.3V.However, the FPGA's output levels are recognized as valid Low or High logic levels by the LCD. The LCD controller accepts 5V TTL signal levels and the 3.3V LVC MOS outputs provided by the FPGA meet the 5V TTL voltage level requirements.

The 390Ω series resistors on the data lines prevent over stressing on the FPGA and Strata Flash I/O pins when the character LCD drives a High logic value. The character LCD drives the data lines when LCD R/W is High. Most applications treat the LCD as a write only peripheral and never read from the display and hence in this trainer the R/W pin is grounded by default.

CHAPTER - 8
PROCEDURE TO WORK IN XILINX SOFTWARE

☞ After installing **Xilinx 11** software, go to Start menu ➢ Programs ➢ Xilinx ISE Design Suite 11 ➢ ISE ➢ Project Navigator (refer **Figure-1**). A Window shown in **Figure-2** will appear.

Figure-1

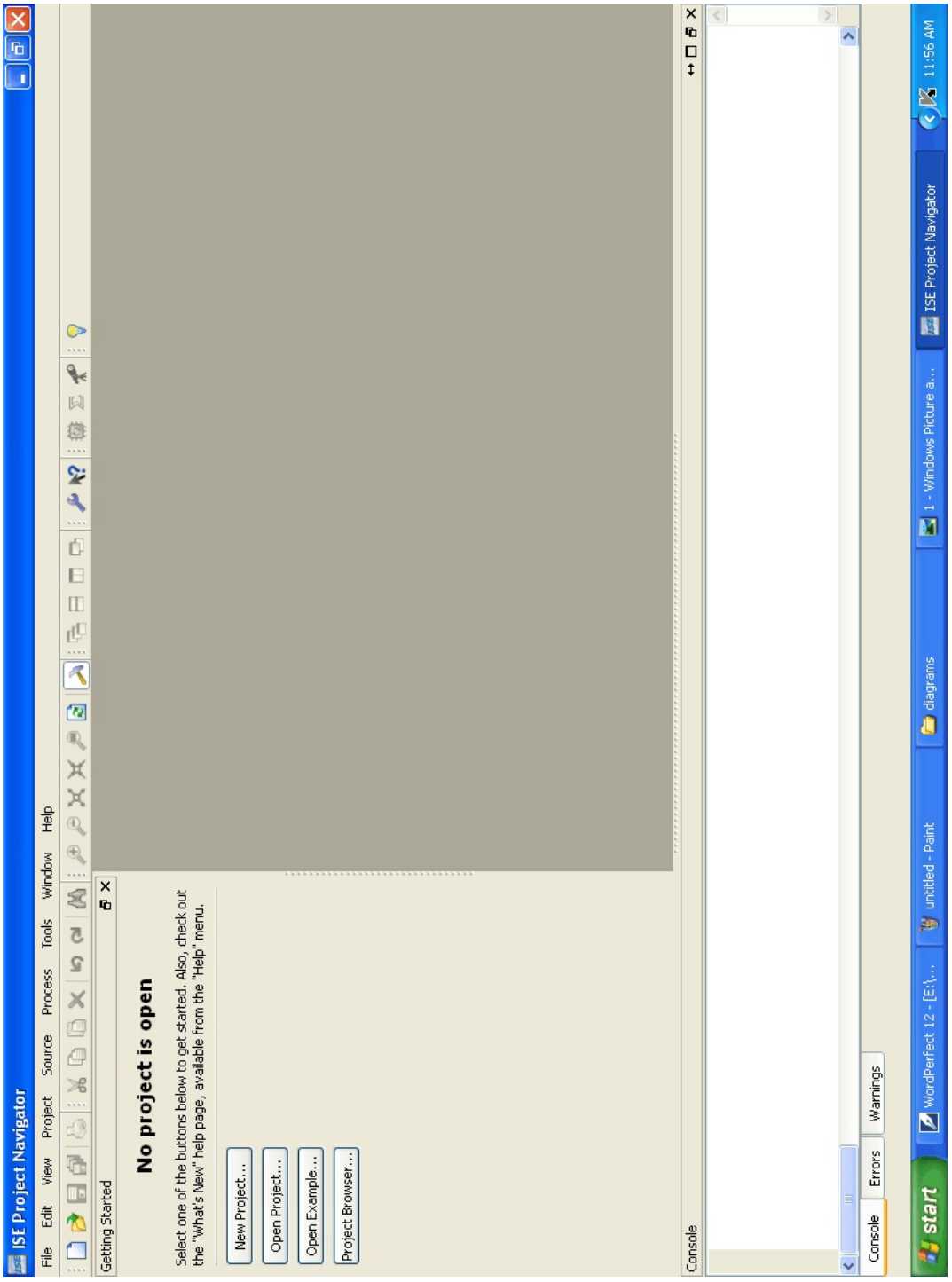


Figure-2

1. Select **File ➢ New Project** (refer **figure-3**).

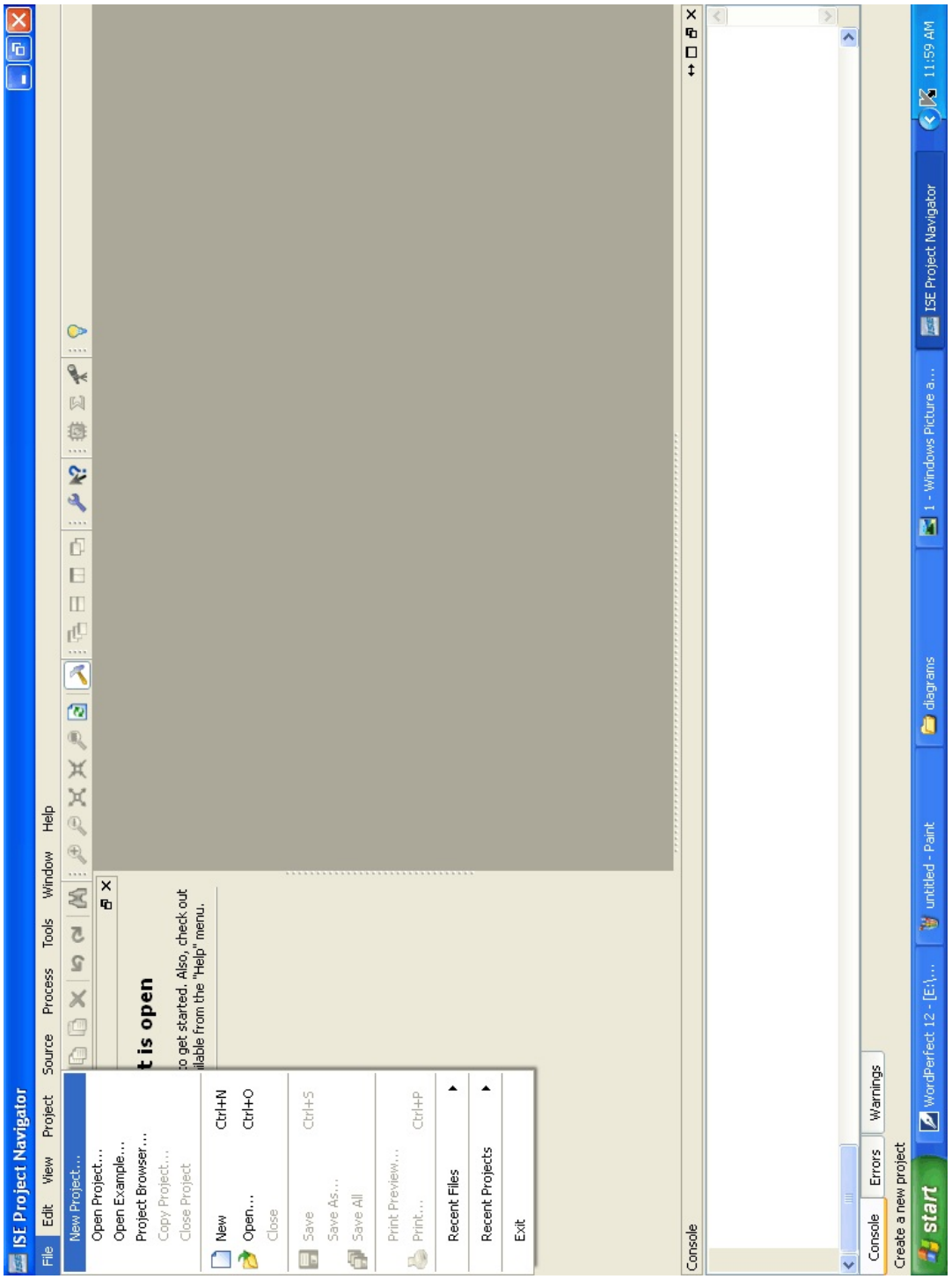

ISE Project Navigator
File Edit View Project Source Process Tools Window Help
New Project...
Open Project...
Open Example...
Project Browser...
Copy Project...
Close Project
New Ctrl+N
Open... Ctrl+O
Close
Save Ctrl+S
Save As...
Save All
Print Preview...
Print... Ctrl+P
Recent Files
Recent Projects
Exit
t is open
o get started. Also, check out
ilable from the "Help" menu.
Console
Console Errors Warnings
Create a new project
start
WordPerfect 12 - [E:\...
untitled - Paint
diagrams
1 - Windows Picture a...
ISE Project Navigator
11:59 AM

Figure-3

2. **New Project wizard** window shown in **Figure -4** will appear. In the **Name** field, enter your project name and enter the location where you want to create the project in the **Location** field. In the **Top-Level Module** select **HDL** and click **Next (**refer **Figure-5)**.

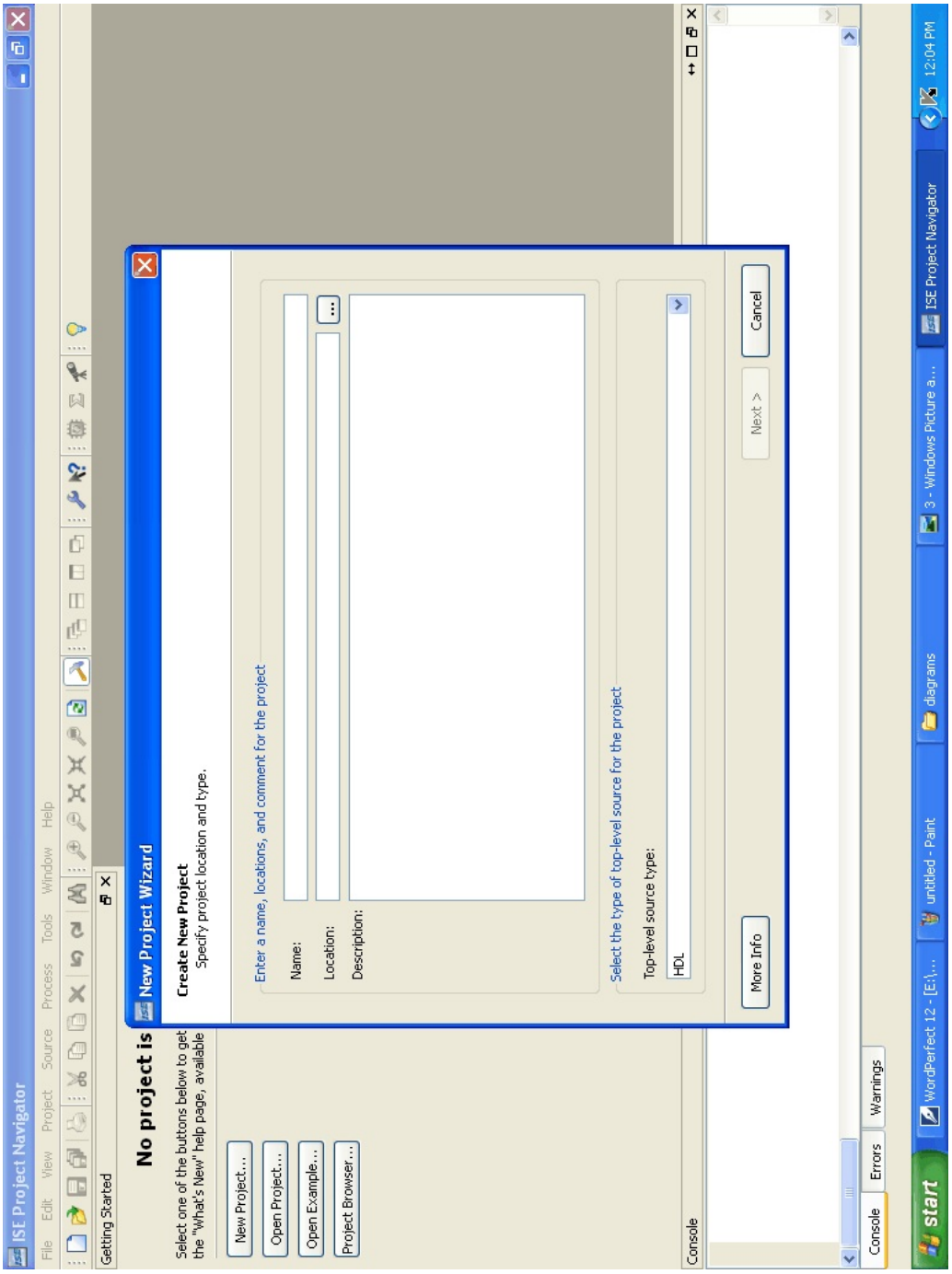


Figure-4

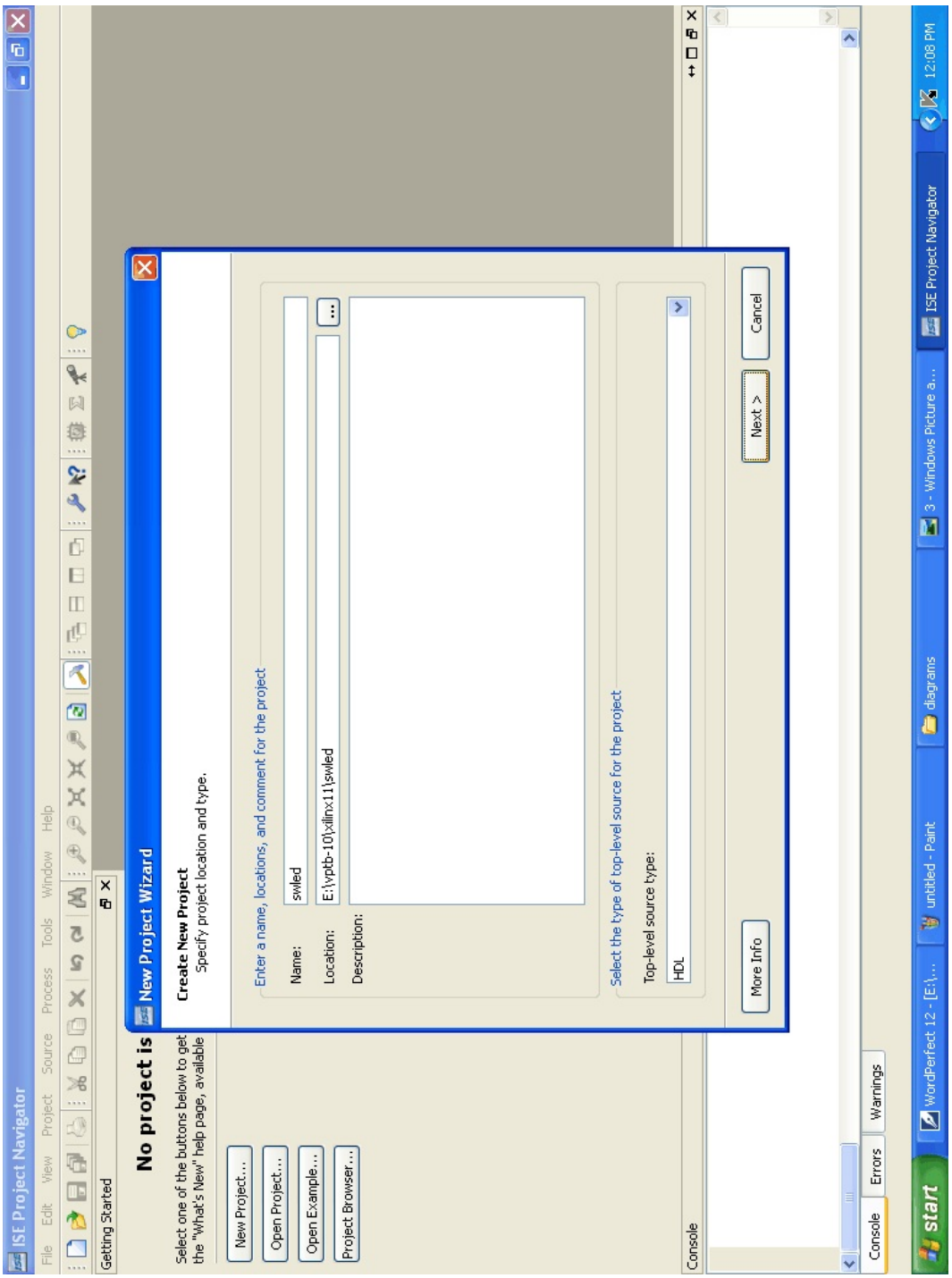

ISE Project Navigator
File Edit View Project Source Process Tools Window Help
Getting Started
No project is
Select one of the buttons below to get the "What's New" help page, available
New Project...
Open Project...
Open Example...
Project Browser...
New Project Wizard
Create New Project
Specify project location and type.
Enter a name, locations, and comment for the project
Name: swled
Location: E:\vptb-10\xilinx11\swled
Description:
Select the type of top-level source for the project
Top-level source type:
HDL
More Info
Next >
Cancel
Console
Errors
Warnings
Console
start
WordPerfect 12 - [E:\...
untitled - Paint
diagrams
3 - Windows Picture a...
ISE Project Navigator
12:08 PM

Figure-5

3. A window given in **Figure-6** will appear.

In the Device and Design flow for the project, select the options as shown below :

Product category	➜ All
Family	➜ Spartan3E
Device	➜ XC3S250E
Package	➜ PQ208
Speed grade	➜ -4

Top-Level Source Type	➜ HDL.
Synthesis Tool	➜ XST (VHDL / Verilog)
Simulator	➜ Modelsim-XE VHDL
Preferred language	➜ VHDL

Then click "**Next**" (refer **Figure- 6,7,8)** and then "**Finish**" (refer **Figure- 9**).

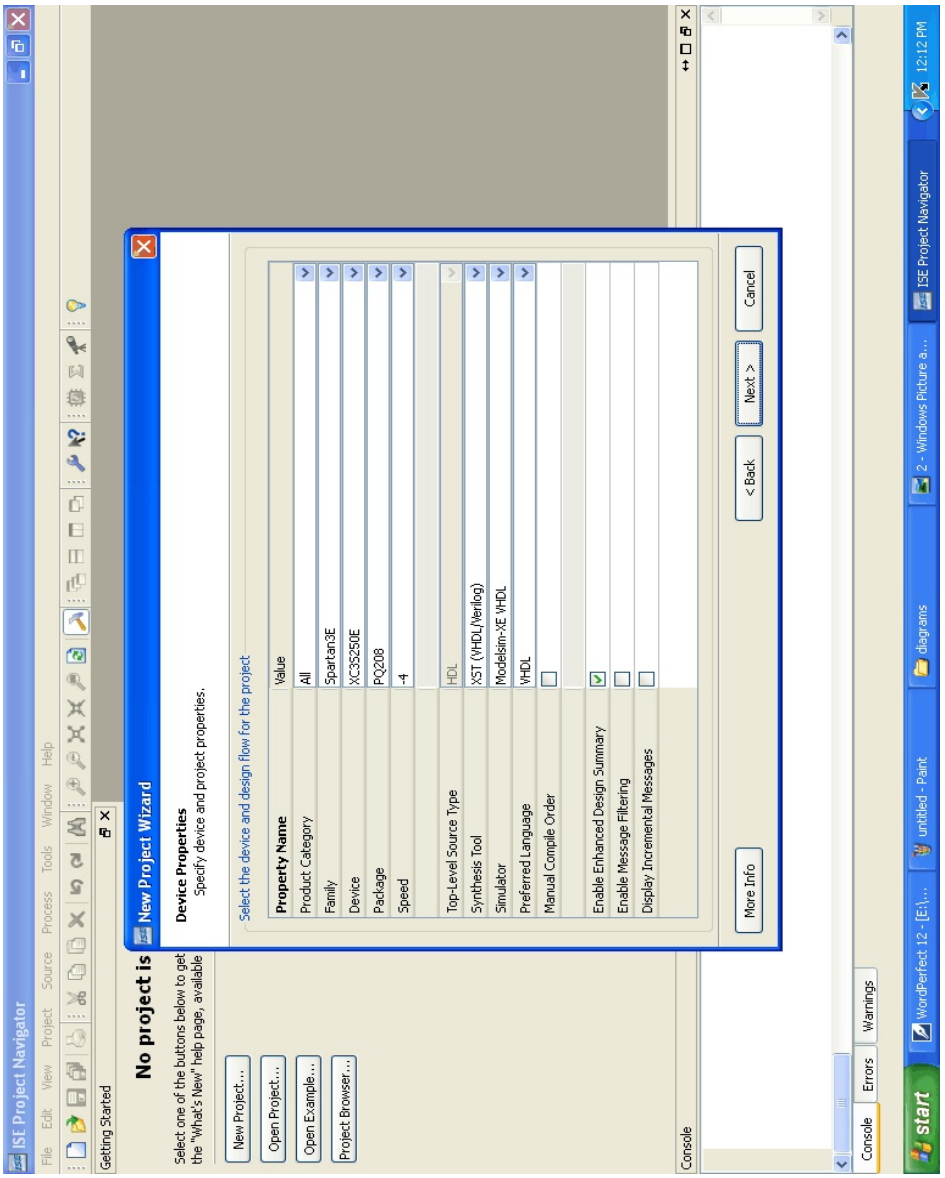


Figure-6

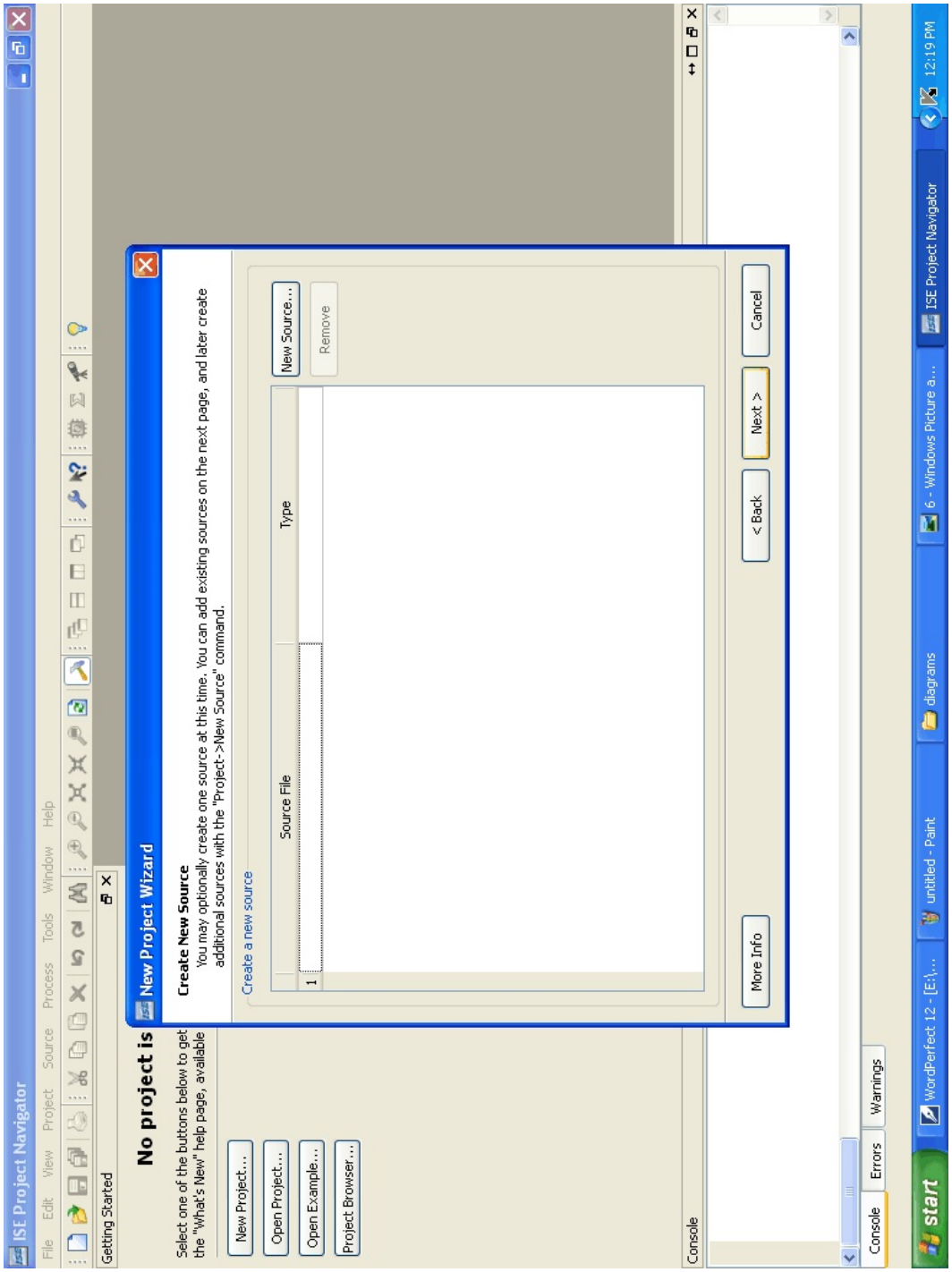

ISE Project Navigator
File Edit View Project Source Process Tools Window Help
Getting Started
No project is
Select one of the buttons below to get the "What's New" help page, available
New Project...
Open Project...
Open Example...
Project Browser...
New Project Wizard
Create New Source
You may optionally create one source at this time. You can add existing sources on the next page, and later create additional sources with the "Project->New Source" command.
Create a new source
Source File
Type
New Source...
Remove
More Info
< Back
Next >
Cancel
Console
Errors
Warnings
start
WordPerfect 12 - [E:\...
untitled - Paint
diagrams
6 - Windows Picture a...
ISE Project Navigator
12:19 PM

Figure-7

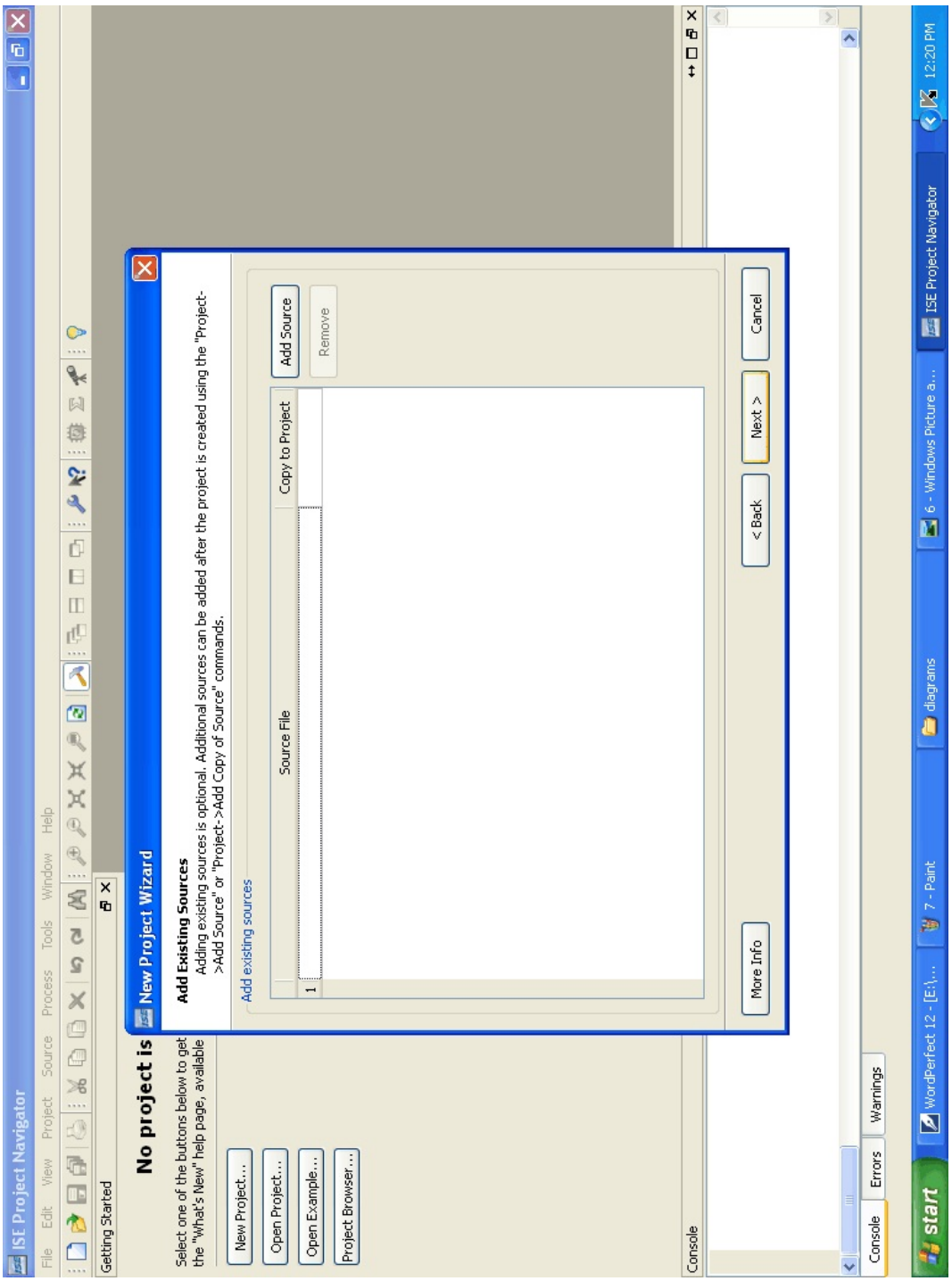

ISE Project Navigator
File Edit View Project Source Process Tools Window Help
Getting Started
No project is
Select one of the buttons below to get
the "What's New" help page, available
New Project...
Open Project...
Open Example...
Project Browser...
New Project Wizard
Add Existing Sources
Adding existing sources is optional. Additional sources can be added after the project is created using the "Project->Add Source" or "Project->Add Copy of Source" commands.
Add existing sources
Source File
Copy to Project
Add Source
Remove
1
More Info
< Back
Next >
Cancel
Console
Errors
Warnings
start
WordPerfect 12 - [E:\...
7 - Paint
diagrams
6 - Windows Picture a...
ISE Project Navigator
12:20 PM

Figure-8

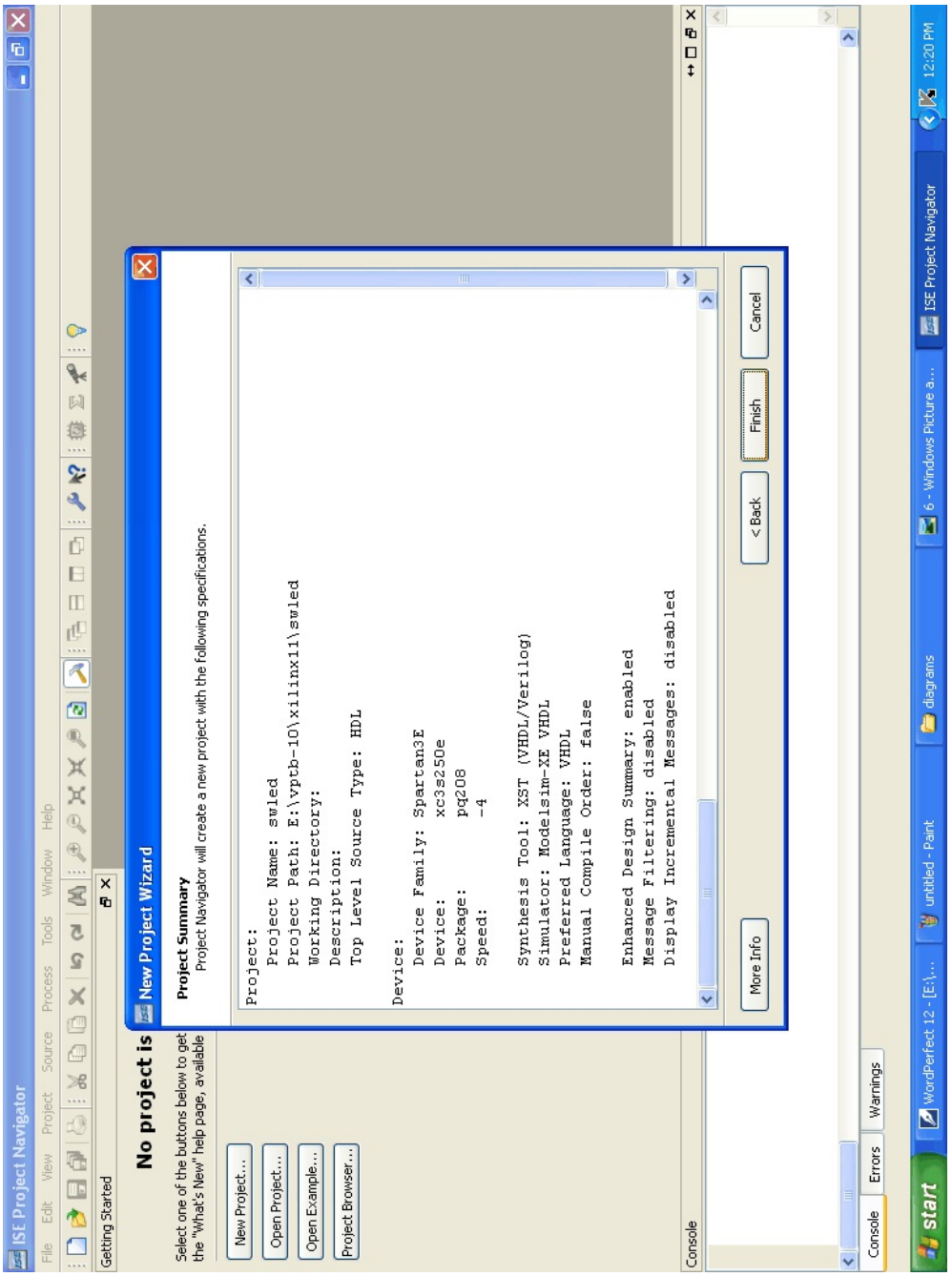

ISE Project Navigator
File Edit View Project Source Process Tools Window Help
Getting Started
No project is
Select one of the buttons below to get
the "What's New" help page, available
New Project...
Open Project...
Open Example...
Project Browser...
New Project Wizard
Project Summary
Project Navigator will create a new project with the following specifications.
Project:
Project Name: swled
Project Path: E:\vptb-10\xilinx11\swled
Working Directory:
Description:
Top Level Source Type: HDL
Device:
Device Family: Spartan3E
Device: xc3s250e
Package: pq208
Speed: -4
Synthesis Tool: XST (VHDL/Verilog)
Simulator: Modelsim-XE VHDL
Preferred Language: VHDL
Manual Compile Order: false
Enhanced Design Summary: enabled
Message Filtering: disabled
Display Incremental Messages: disabled
More Info
< Back
Finish
Cancel
Console
Errors
Warnings
Console
start
WordPerfect 12 - [E:\...
untitled - Paint
diagrams
6 - Windows Picture a...
ISE Project Navigator
12:20 PM

Figure-9

4. A window given in **Figure-10** will appear. Select **Project** menu ➢**New Source** (refer **Figure-11**). A window given in **Figure-12** will appear. Then select **VHDL module**, and specify the file name in appropriate field as shown in figure-13 and Click **Next**.If you want you can give inputs & outputs in the appropriate positions in the window shown in **Figure -14.**You can also skip these information by simply clicking N**ext** button and click **Finish** (refer **Figure - 15)**.

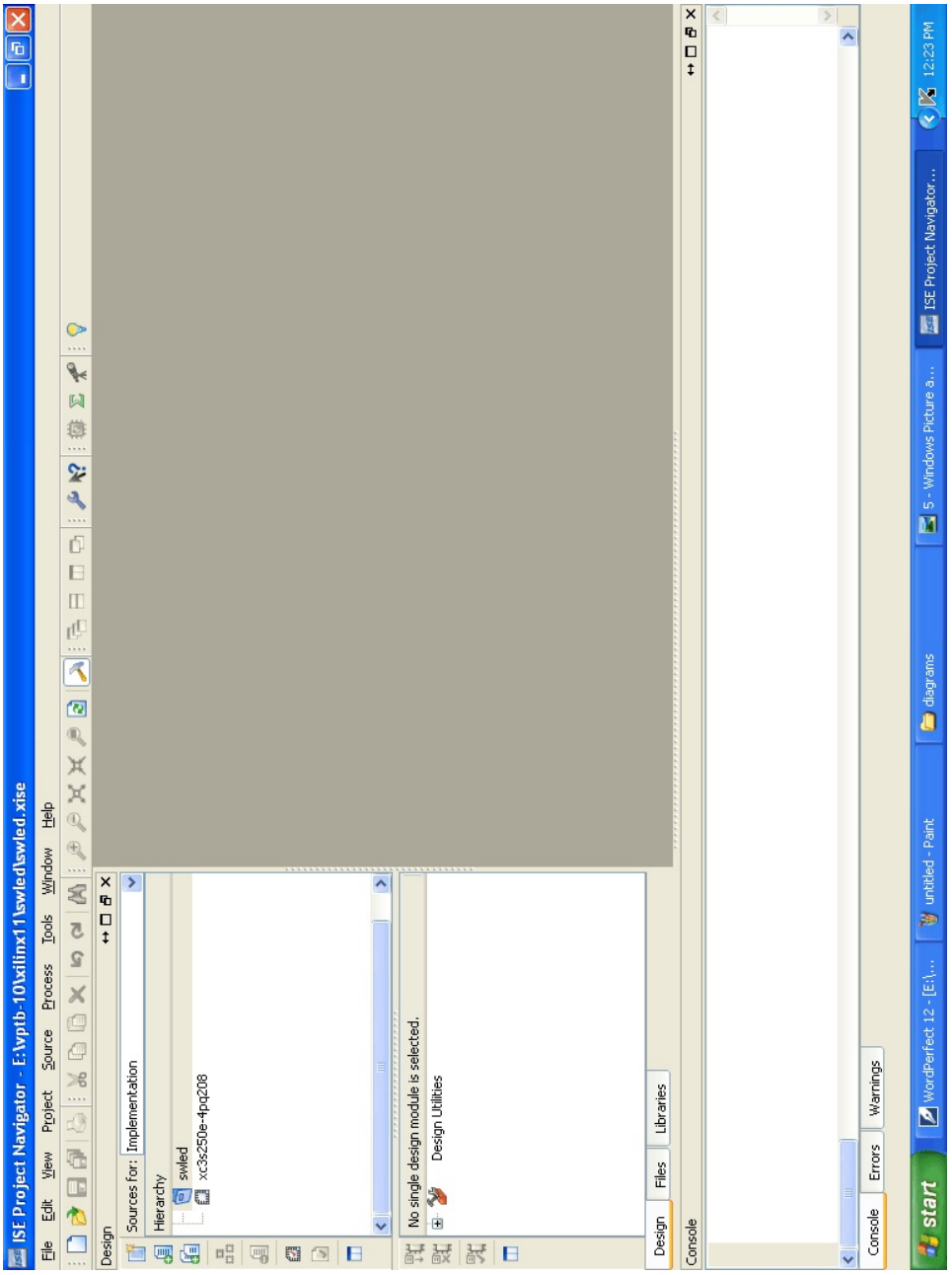


Figure-10

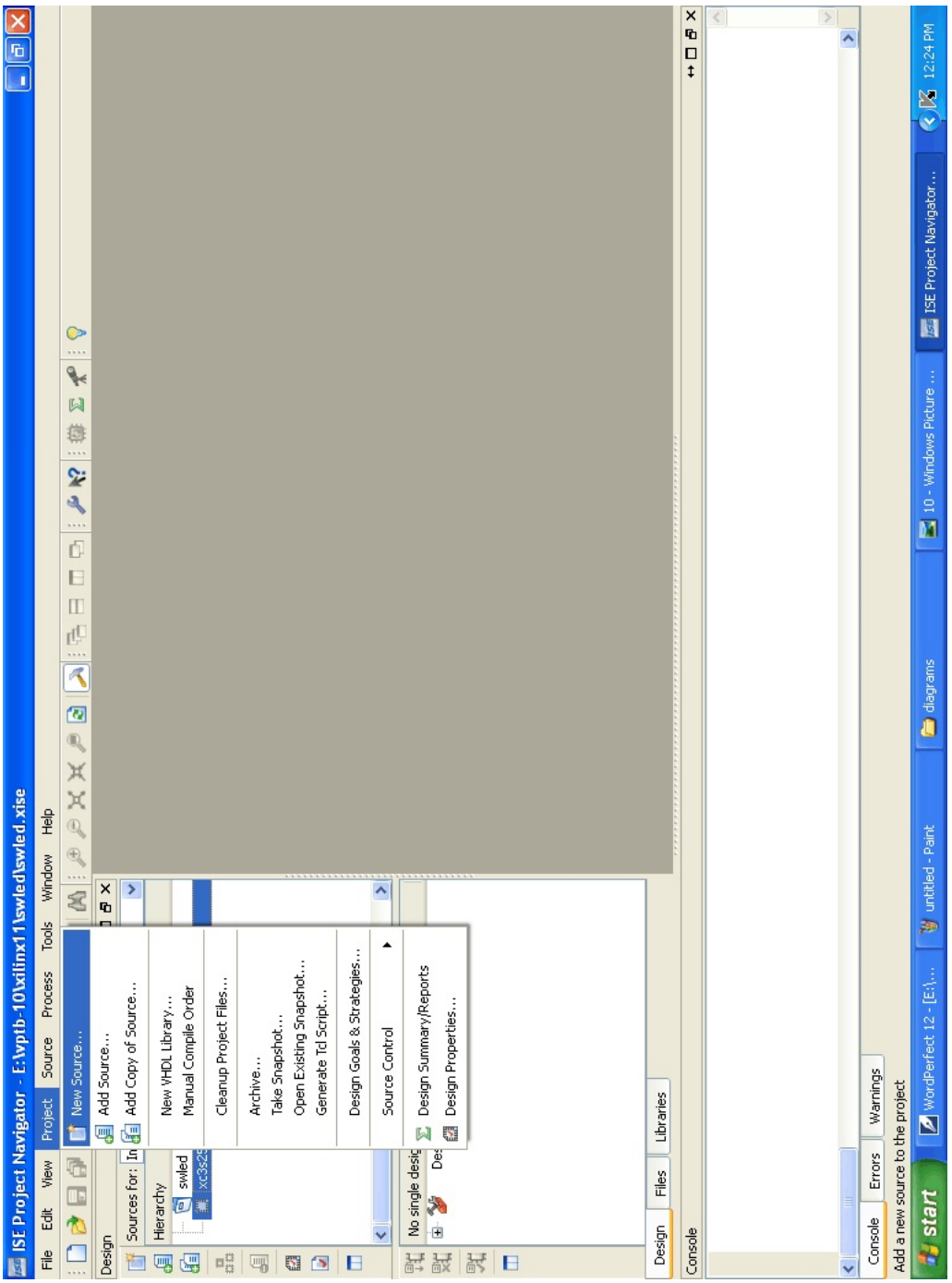

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise
File Edit View Project Source Process Tools Window Help
New Source...
Add Source...
Add Copy of Source...
New VHDL Library...
Manual Compile Order
Cleanup Project Files...
Archive...
Take Snapshot...
Open Existing Snapshot...
Generate Tcl Script...
Design Goals & Strategies...
Source Control
Design Summary/Reports
Design Properties...
Design
Hierarchy
swled
Sources for:
Design
Files
Libraries
Console
Errors
Warnings
Add a new source to the project
start
WordPerfect 12 - [E:\...
untitled - Paint
diagrams
10 - Windows Picture ...
ISE Project Navigator ...
12:24 PM

Figure-11

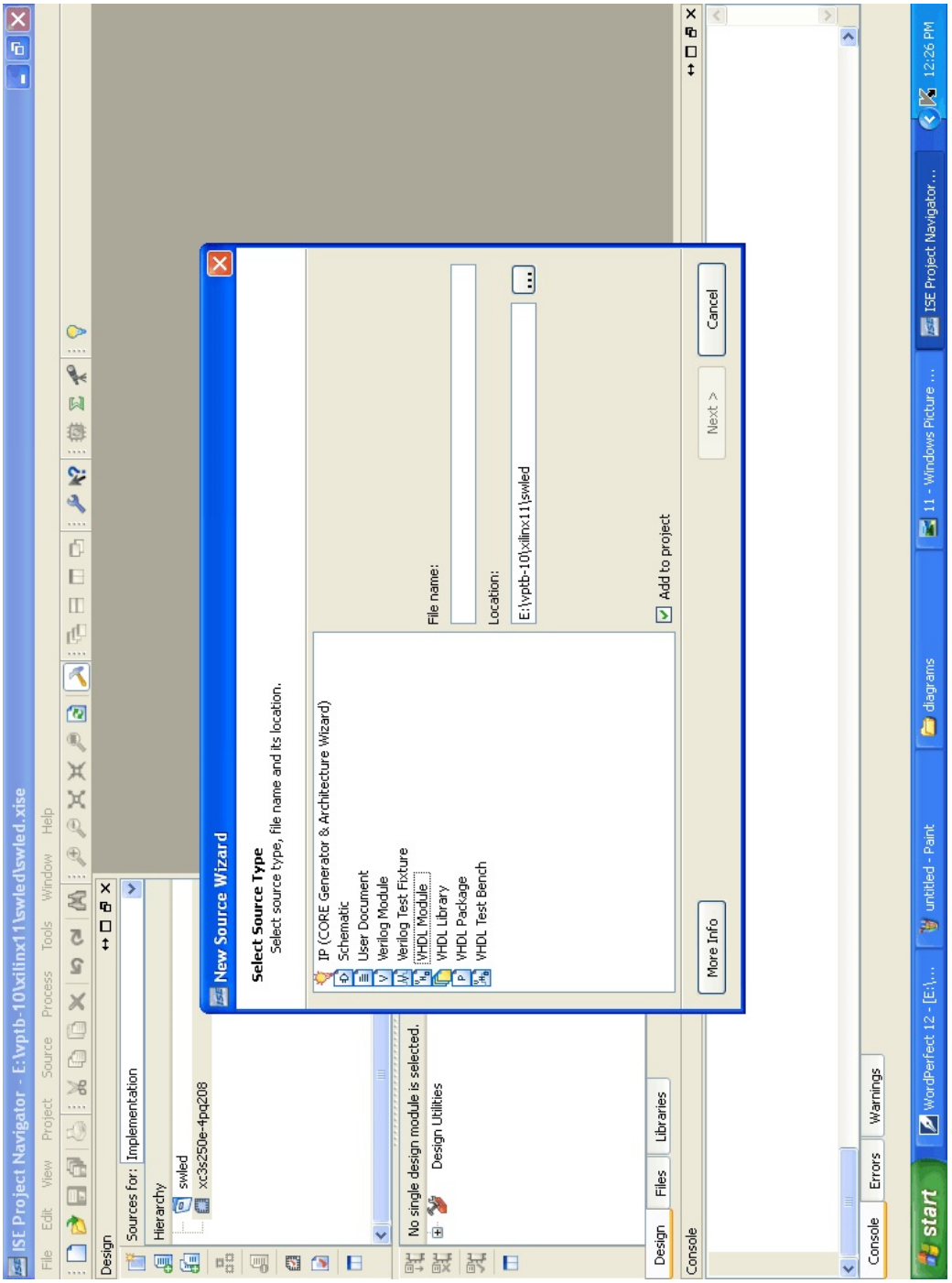

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise
New Source Wizard
Select Source Type
Select source type, file name and its location.
IP (CORE Generator & Architecture Wizard)
Schematic
User Document
Verilog Module
Verilog Test Fixture
VHDL Module
VHDL Library
VHDL Package
VHDL Test Bench
File name:
Location:
E:\vptb-10\xilinx11\swled
Add to project
More Info
Next >
Cancel
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
No single design module is selected.
Design Utilities
Design
Files
Libraries
Console
Errors
Warnings
start
WordPerfect 12 - [E:\...
untitled - Paint
diagrams
11 - Windows Picture ...
ISE Project Navigator...
12:26 PM

Figure-12

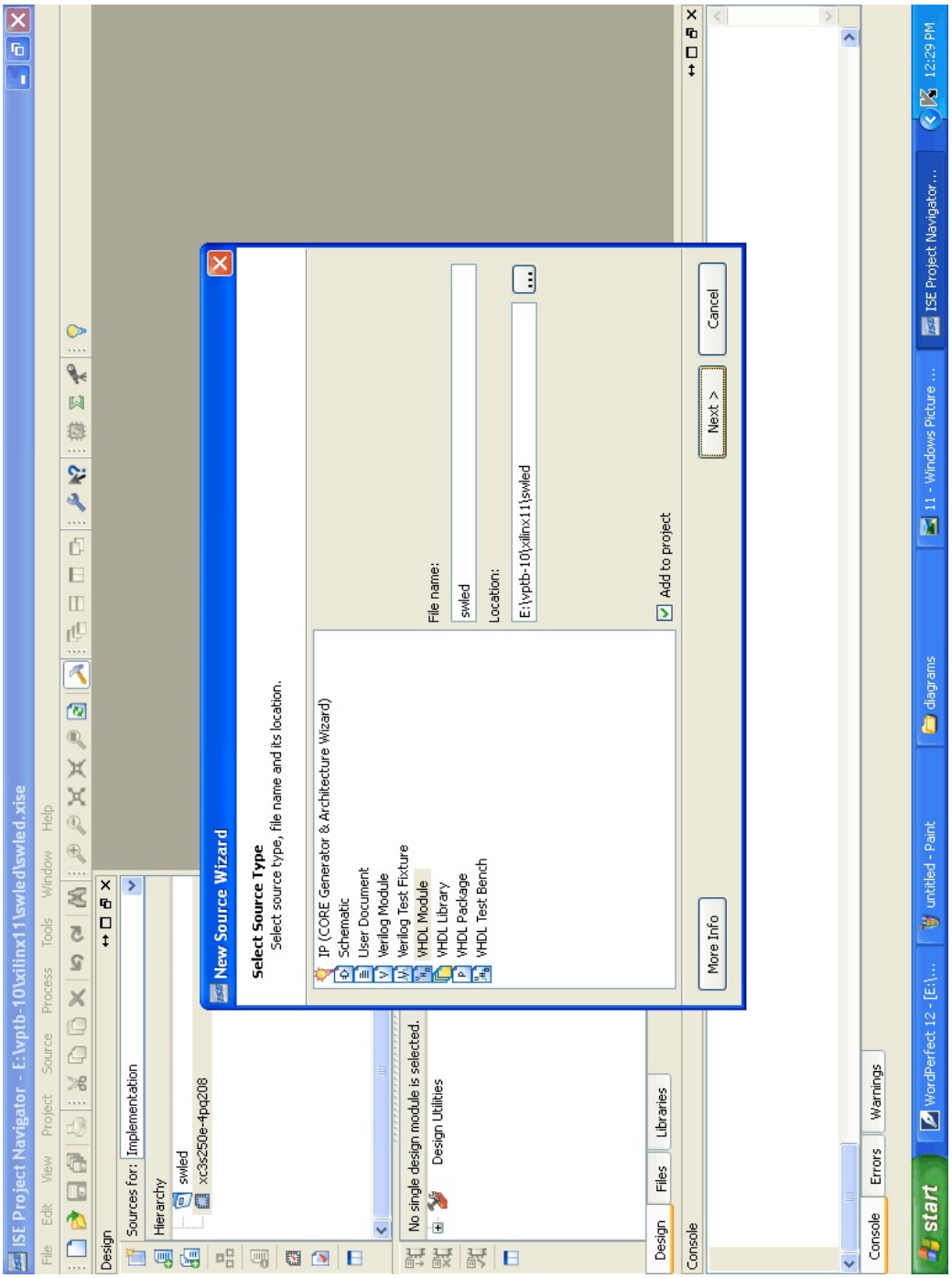

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise
New Source Wizard
Select Source Type
Select source type, file name and its location.
IP (CORE Generator & Architecture Wizard)
Schematic
User Document
Verilog Module
Verilog Test Fixture
VHDL Module
VHDL Library
VHDL Package
VHDL Test Bench
File name:
swled
Location:
E:\vptb-10\xilinx11\swled
Add to project
More Info
Next >
Cancel
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
No single design module is selected.
Design Utilities
Design
Files
Libraries
Console
Errors
Warnings
start
WordPerfect 12 - [E:\...
untitled - Paint
diagrams
11 - Windows Picture ...
ISE Project Navigator...
12:29 PM

Figure-13

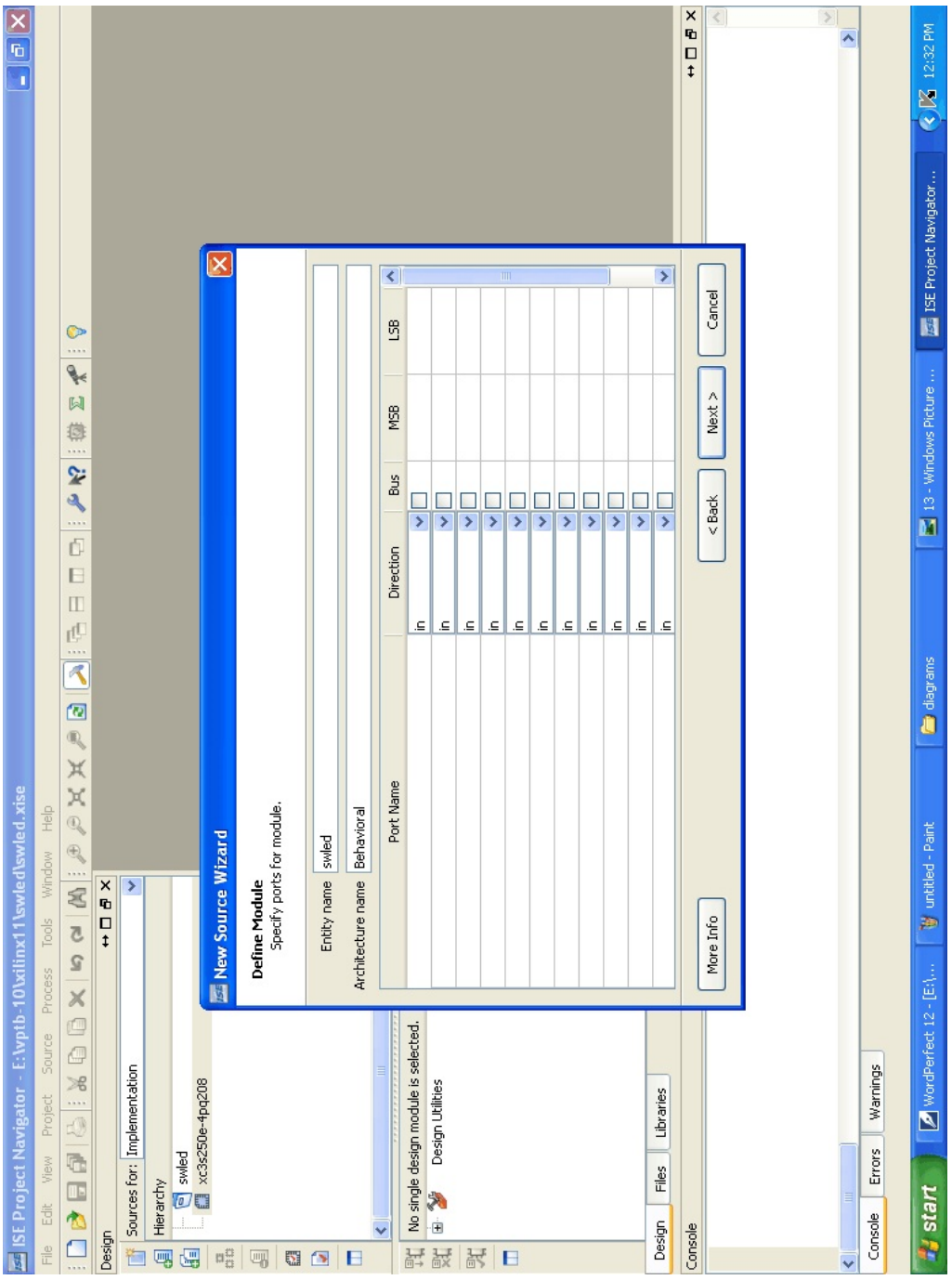

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise
File Edit View Project Source Process Tools Window Help
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
New Source Wizard
Define Module
Specify ports for module.
Entity name
swled
Architecture name
Behavioral
Port Name
Direction
Bus
MSB
LSB
in
More Info
< Back
Next >
Cancel
No single design module is selected.
Design Utilities
Design
Files
Libraries
Console
Errors
Warnings
start
WordPerfect 12 - [E:\...
untitled - Paint
diagrams
13 - Windows Picture ...
ISE Project Navigator...
12:32 PM

Figure-14

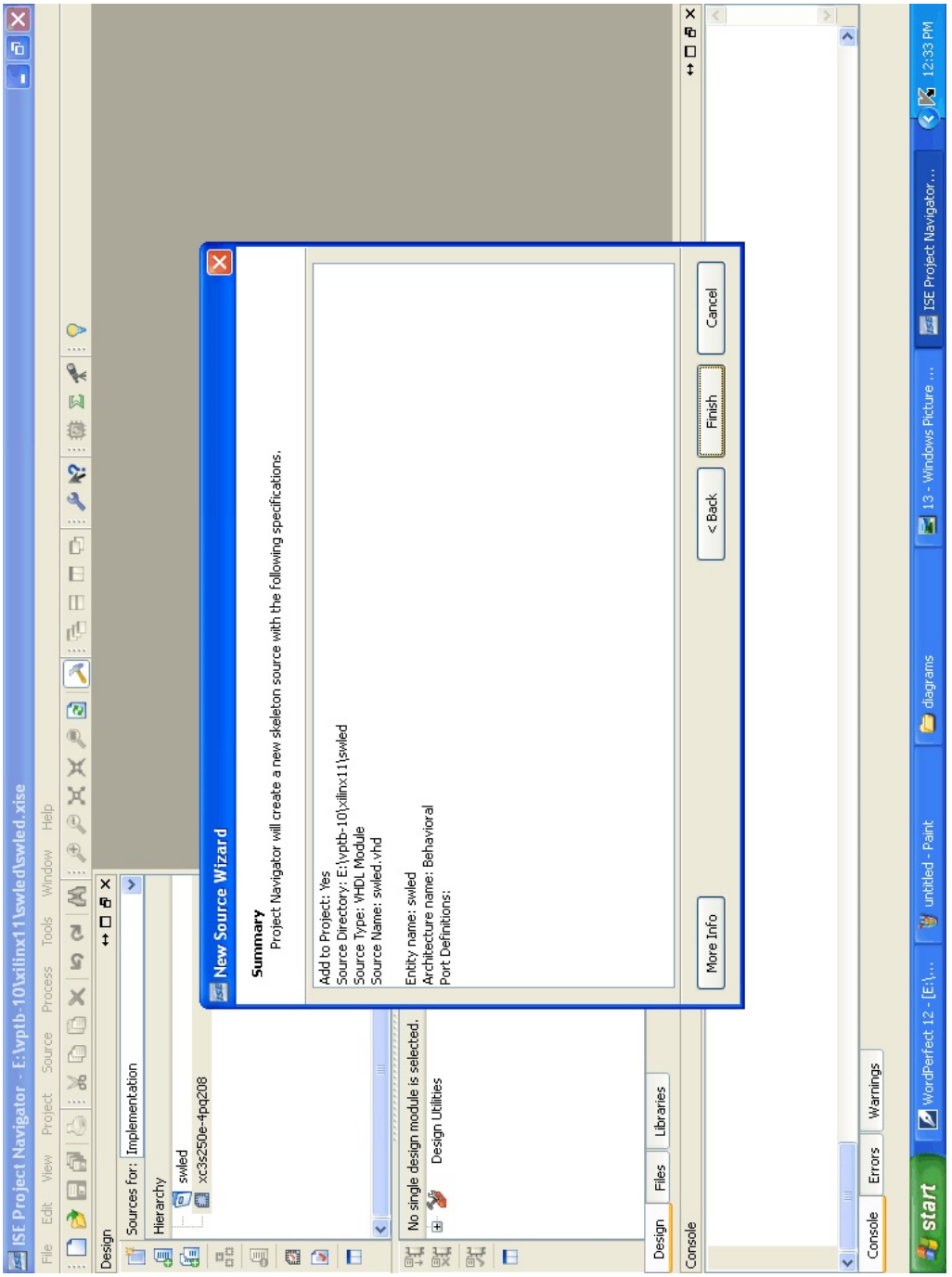

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise
File Edit View Project Source Process Tools Window Help
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
New Source Wizard
Summary
Project Navigator will create a new skeleton source with the following specifications.
Add to Project: Yes
Source Directory: E:\vptb-10\xilinx11\swled
Source Type: VHDL Module
Source Name: swled.vhd
Entity name: swled
Architecture name: Behavioral
Port Definitions:
More Info
< Back
Finish
Cancel
No single design module is selected.
Design Utilities
Design
Files
Libraries
Console
Errors
Warnings
start
WordPerfect 12 - [E:\...
untitled - Paint
diagrams
13 - Windows Picture ...
ISE Project Navigator...
12:33 PM

Figure-15

5. A window shown in **Figure-16** will appear.You can type your VHDL code in the right side of window and save it by clicking on the **Save** button (refer **Figure-17**). Now in the "**Processes:**" window double click **Synthesize-XST** as shown in **Figure-18**.

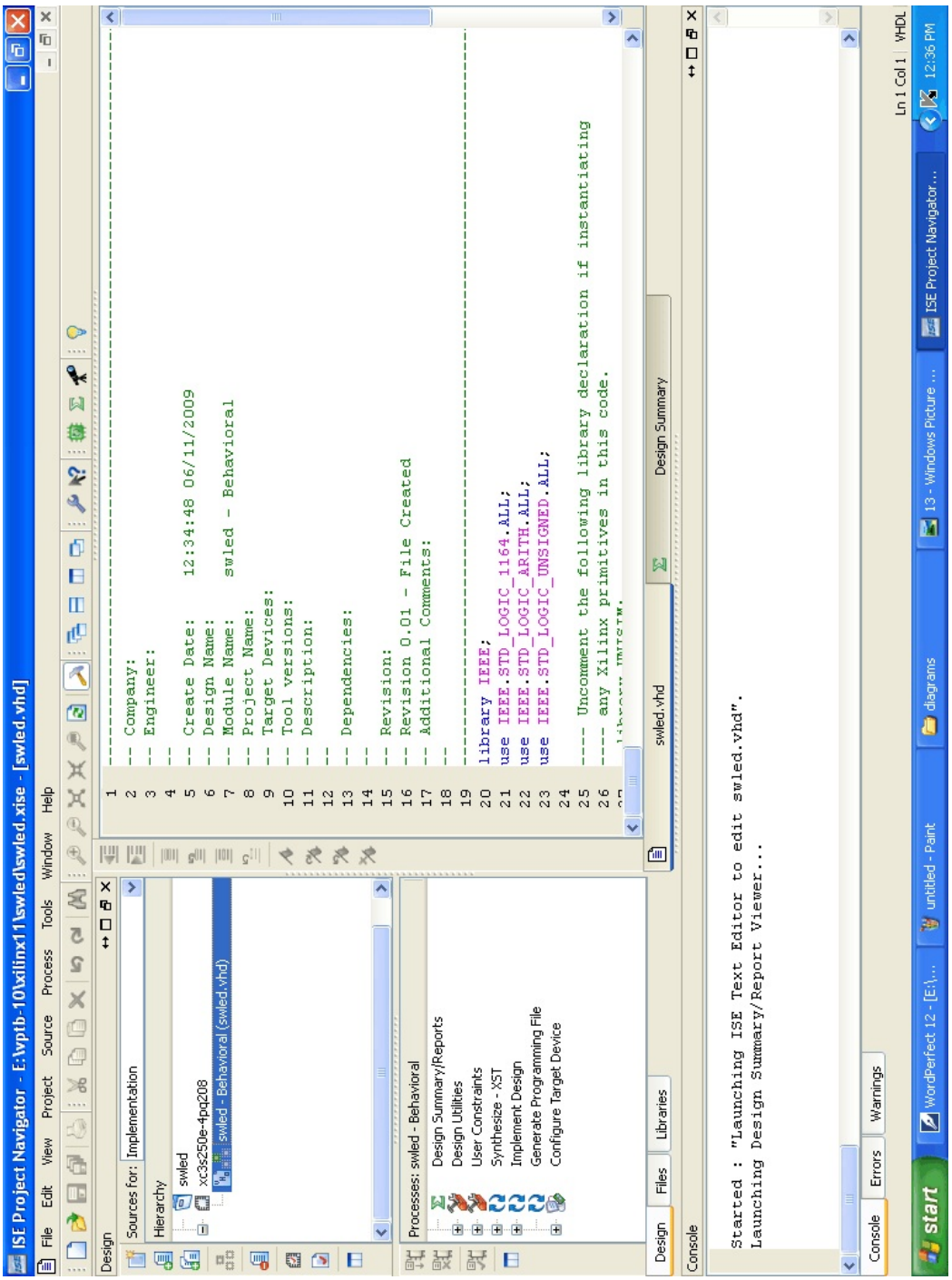


Figure-16

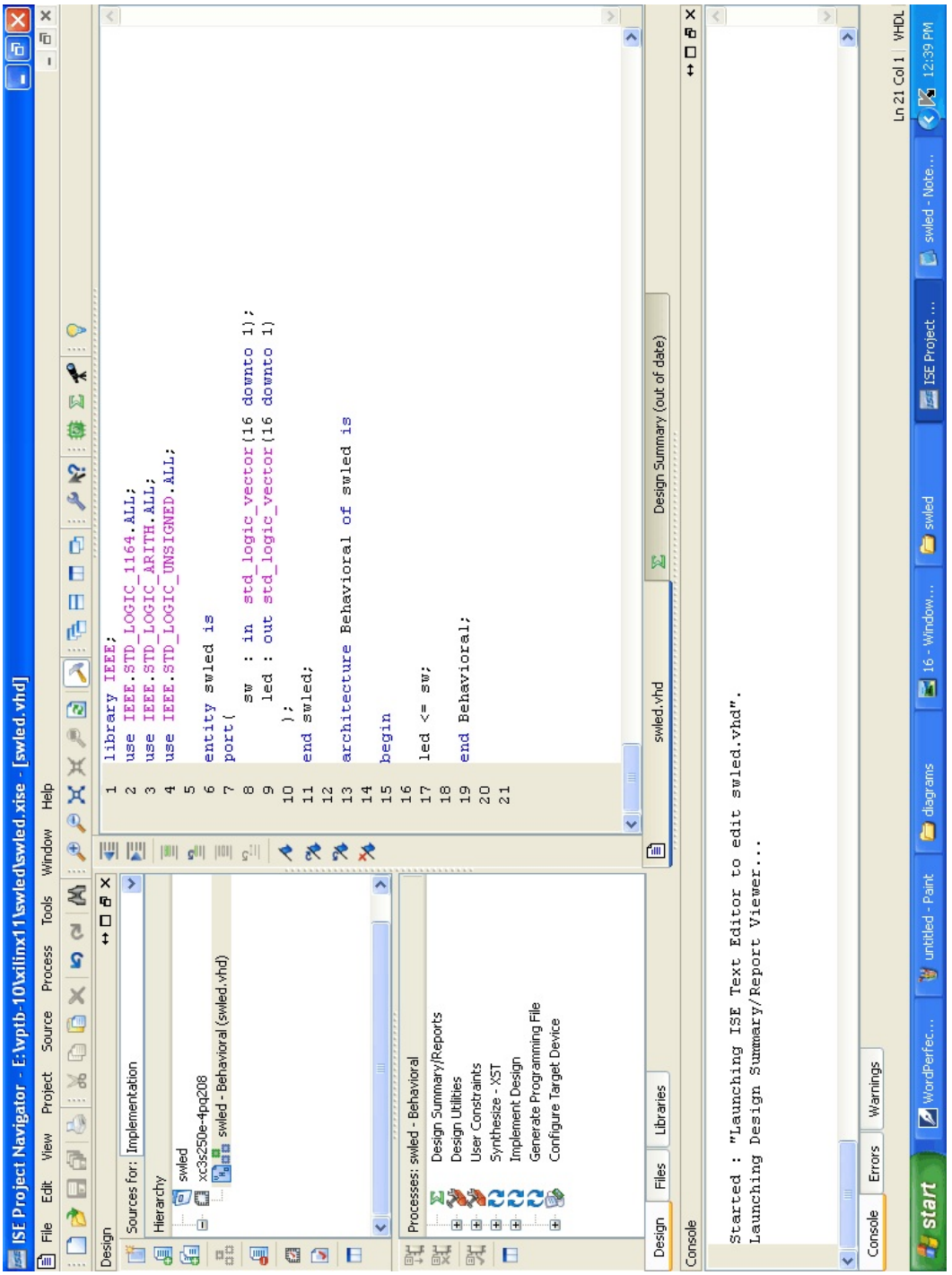

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise - [swled.vhd]
File Edit View Project Source Process Tools Window Help
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
swled - Behavioral (swled.vhd)
Processes: swled - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Design Files Libraries
swled.vhd
Design Summary (out of date)
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_ARITH.ALL;
4 use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6 entity swled is
7 port(
8 sw : in std_logic_vector(16 downto 1);
9 led : out std_logic_vector(16 downto 1)
10);
11 end swled;
12
13 architecture Behavioral of swled is
14
15 begin
16
17 led <= sw;
18
19 end Behavioral;
20
21
Console
Started : "Launching ISE Text Editor to edit swled.vhd".
Launching Design Summary/Report Viewer...
Console Errors Warnings
Ln 21 Col 1 VHDL
start
WordPerfec... untitled - Paint diagrams 16 - Window... swled ISE Project ... swled - Note...
12:39 PM

Figure-17

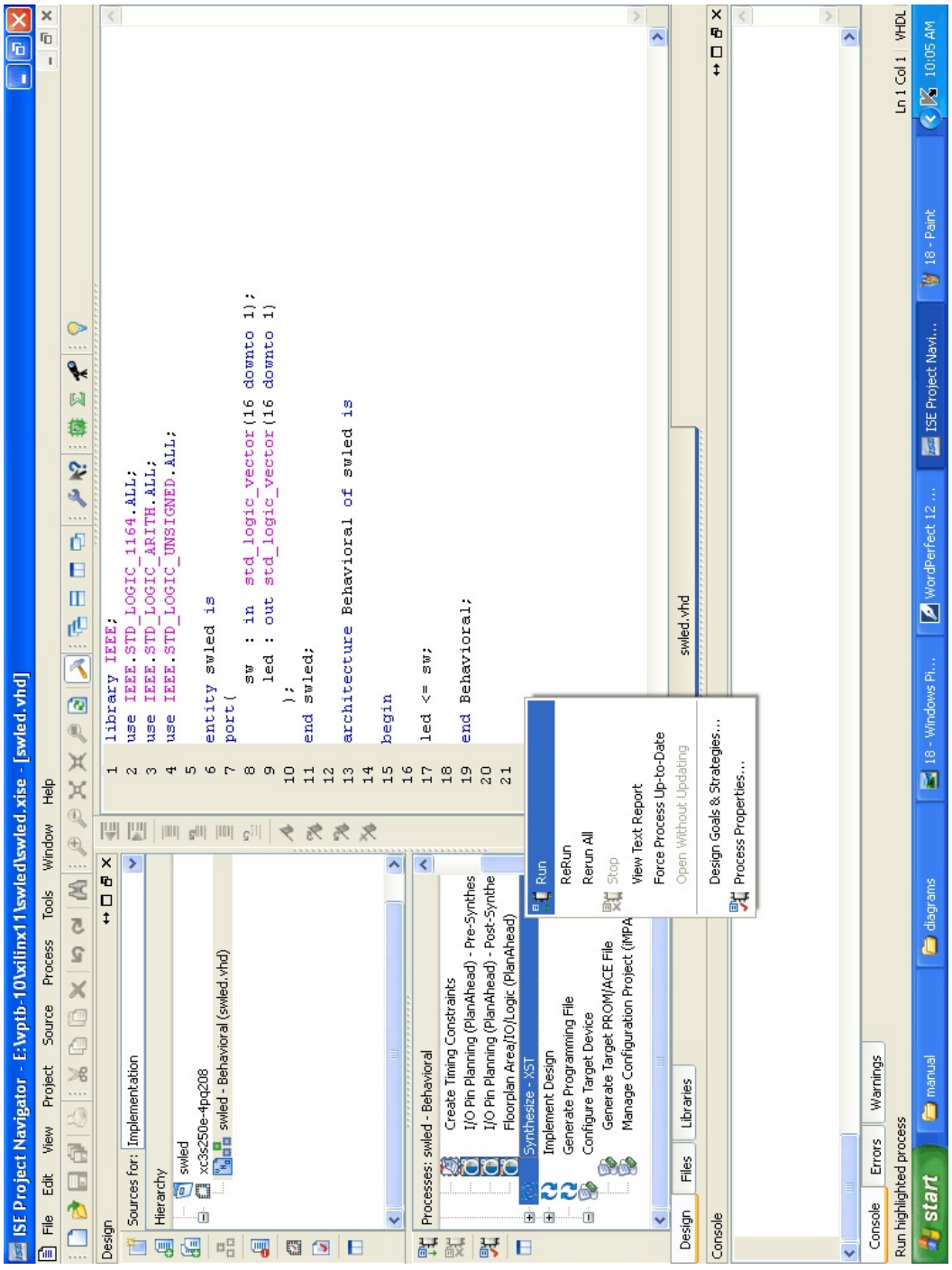

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise - [swled.vhd]
File Edit View Project Source Process Tools Window Help
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
swled - Behavioral (swled.vhd)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity swled is
port(
sw : in std_logic_vector(16 downto 1);
led : out std_logic_vector(16 downto 1)
);
end swled;
architecture Behavioral of swled is
begin
led <= sw;
end Behavioral;
swled.vhd
Processes: swled - Behavioral
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Synthes
I/O Pin Planning (PlanAhead) - Post-Synthe
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Generate Target PROM/ACE File
Manage Configuration Project (iMPA
Run
ReRun
Rerun All
Stop
View Text Report
Force Process Up-to-Date
Open Without Updating
Design Goals & Strategies...
Process Properties...
Design
Files
Libraries
Console
Errors
Warnings
Run highlighted process
Ln 1 Col 1
VHDL
start
manual
diagrams
18 - Windows Pi...
WordPerfect 12 ...
ISE Project Navi...
18 - Paint
10:05 AM

Figure-18

6. After **synthesis** is completed successfully, select **Project** menu ➢click **New Source** (refer **Figure-19**). Then select **Implementation Constraints File**, give the filename in the appropriate field click **Next** and **Finish** (refer **Figure-20,21**).

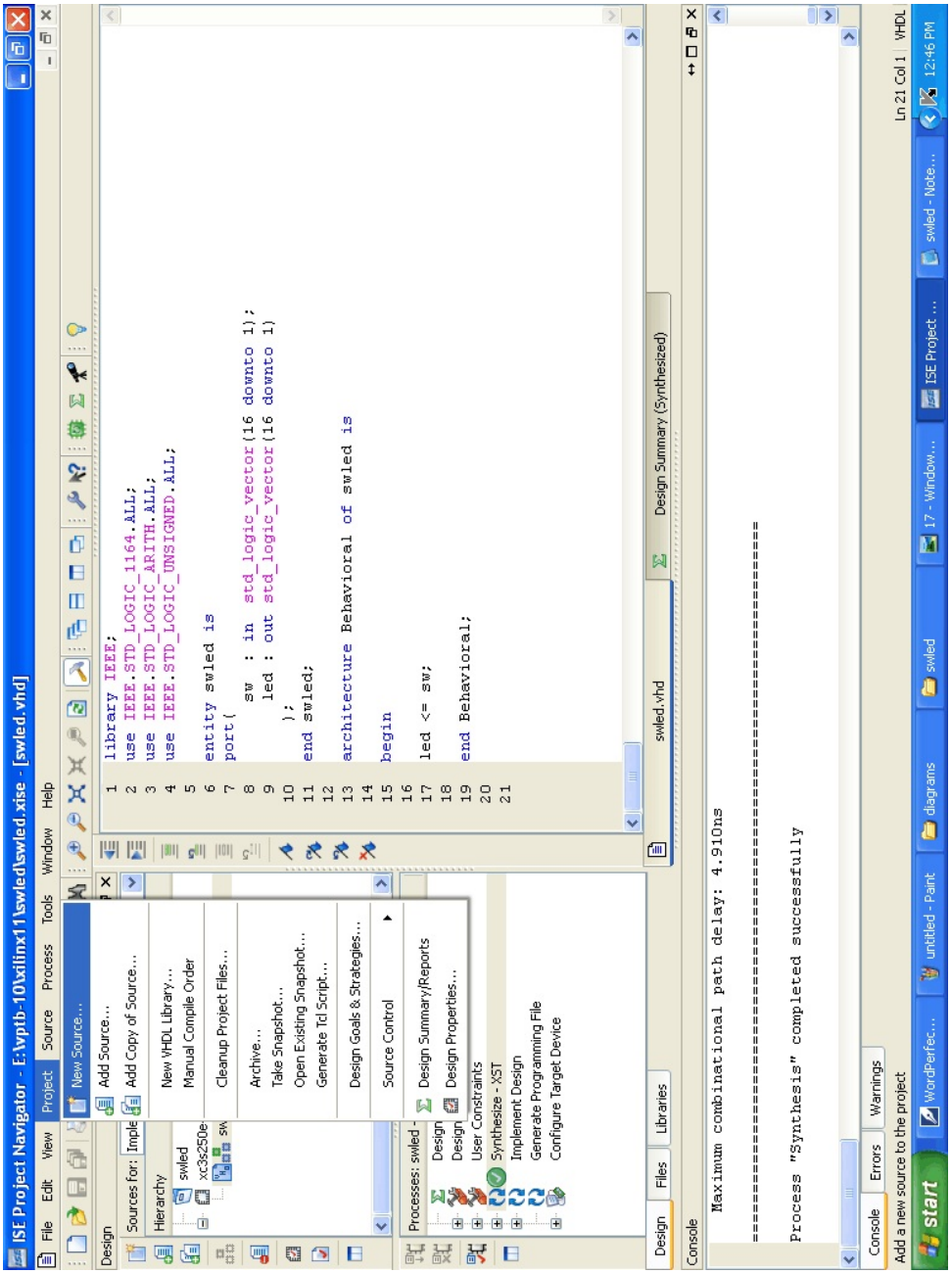


Figure-19

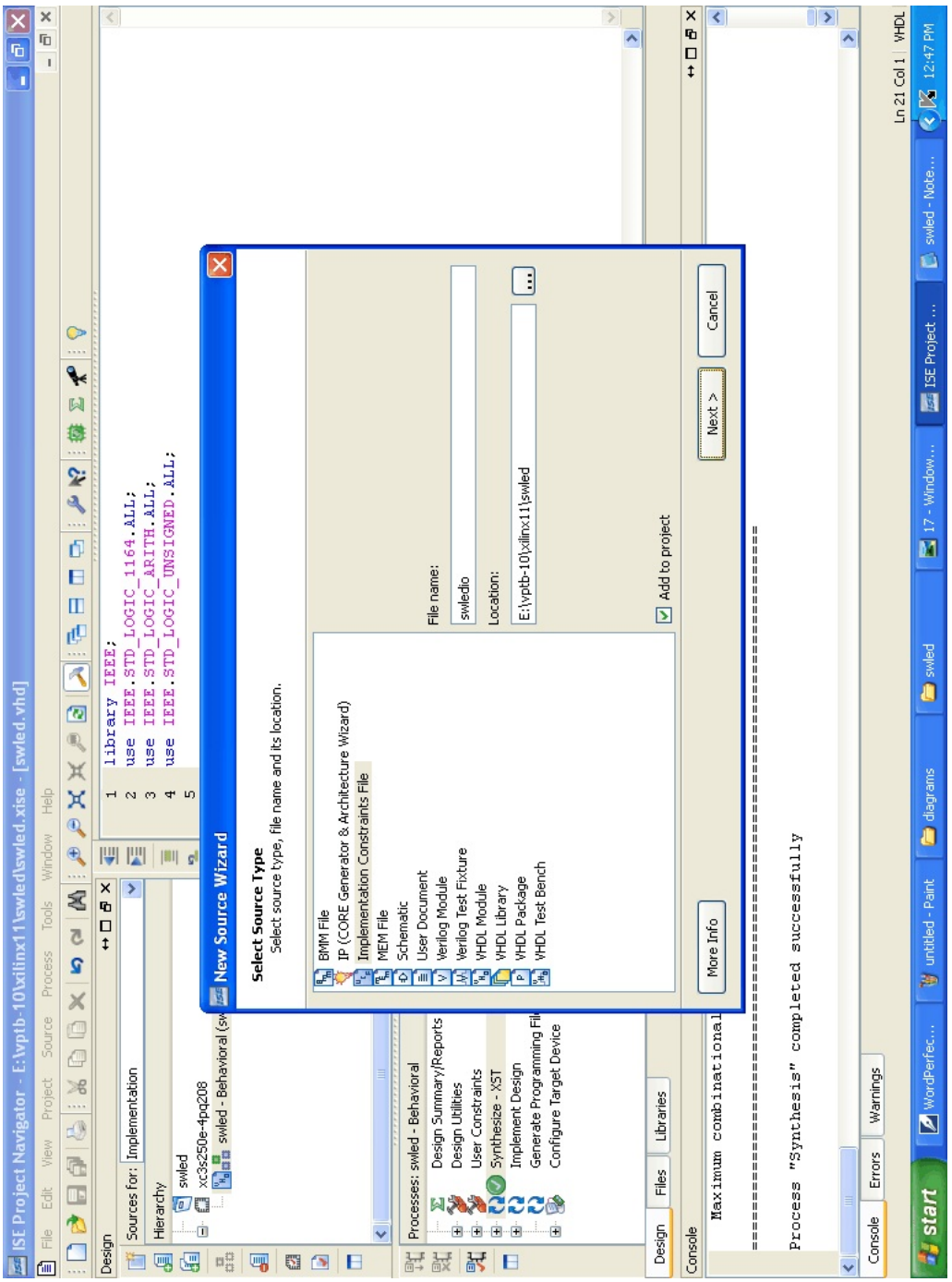

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise - [swled.vhd]
File Edit View Project Source Process Tools Window Help
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
swled - Behavioral (sw
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_ARITH.ALL;
4 use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
New Source Wizard
Select Source Type
Select source type, file name and its location.
BMM File
IP (CORE Generator & Architecture Wizard)
Implementation Constraints File
MEM File
Schematic
User Document
Verilog Module
Verilog Test Fixture
VHDL Module
VHDL Library
VHDL Package
VHDL Test Bench
File name:
swledio
Location:
E:\vptb-10\xilinx11\swled
Add to project
More Info
Next >
Cancel
Processes: swled - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming Fil
Configure Target Device
Design
Files
Libraries
Console
Maximum combinational
Process "Synthesis" completed successfully
Console
Errors
Warnings
start
WordPerfec...
untitled - Paint
diagrams
swled
17 - Window...
ISE Project ...
swled - Note...
12:47 PM
Ln 21 Col 1
VHDL

Figure-20

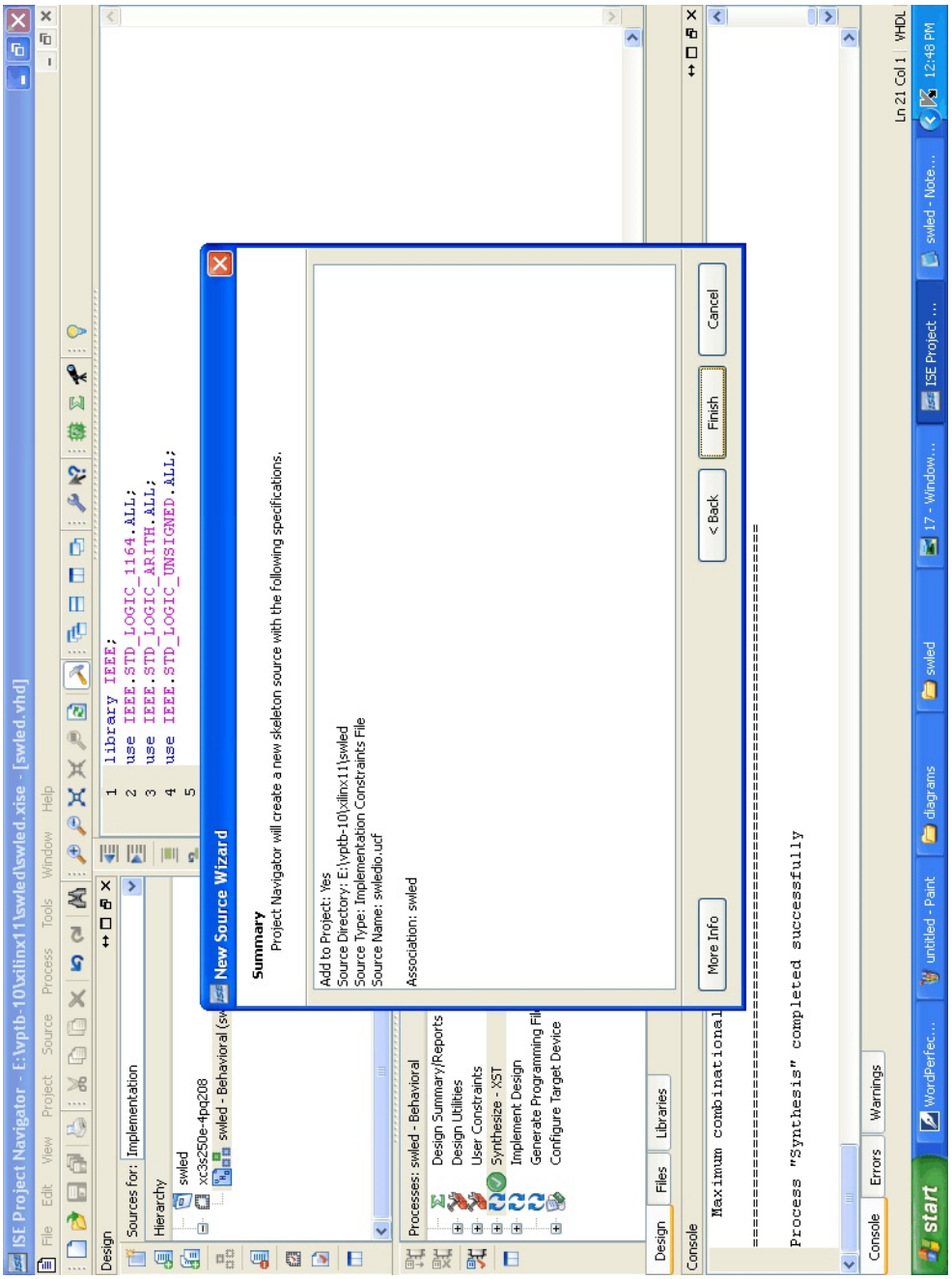


Figure-21

7. Now the **.UCF** file gets added to the project at the **Sources** window.
8. There are two ways for assigning pins of the design:

First one is by manually typing the Pin locations of the port using the **NET** and **LOC** keywords which is done by selecting the **.UCF** file in the **Sources:** window and then double clicking the **Edit Constraints (Text)** option under the **User Constraints** category.A new tab with the **filename.ucf** appears next to the **filename.vhd** tab (refer **Figure-21a,21b)**.

Second way is to enter the pin location in the text box of the **Plan Ahead window** which is done by clicking the .**VHD** file in the **Sources** window and in the **Processes** window double click **I/O Pin Planning(Plan-Ahead) - Post-Synthesis** under the **User constraints** option(refer **Figure-22a, 22b**).A new window named **Plan Ahead 11.1** as shown in **Figure-23** will appear, the pins of the design can be assigned in this window by double clicking the port name for eg. led[1] under the **I/O Ports** category (refer **Figure-24)**.A new tab named **I/O Port Properties** appears next to the **Netlist** tab, in that tab enter the pin location of the port in the **Site** text box as shown in **Figure-25**.Repeat the same procedure for assigning pins for other ports of the design and then click the **Save** button (refer **Figure-26**) .

NOTE : User can assign pins by anyone of the ways mentioned above.

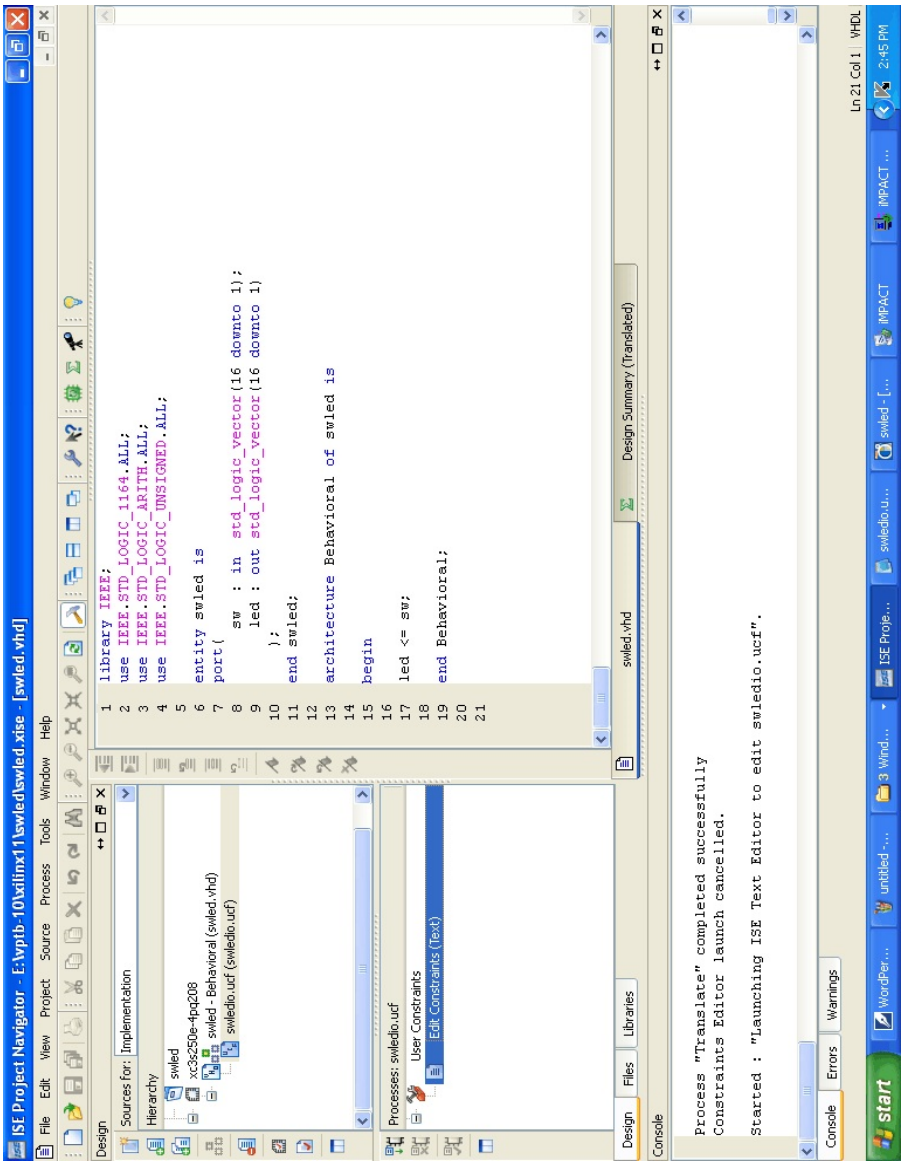


Figure-21a

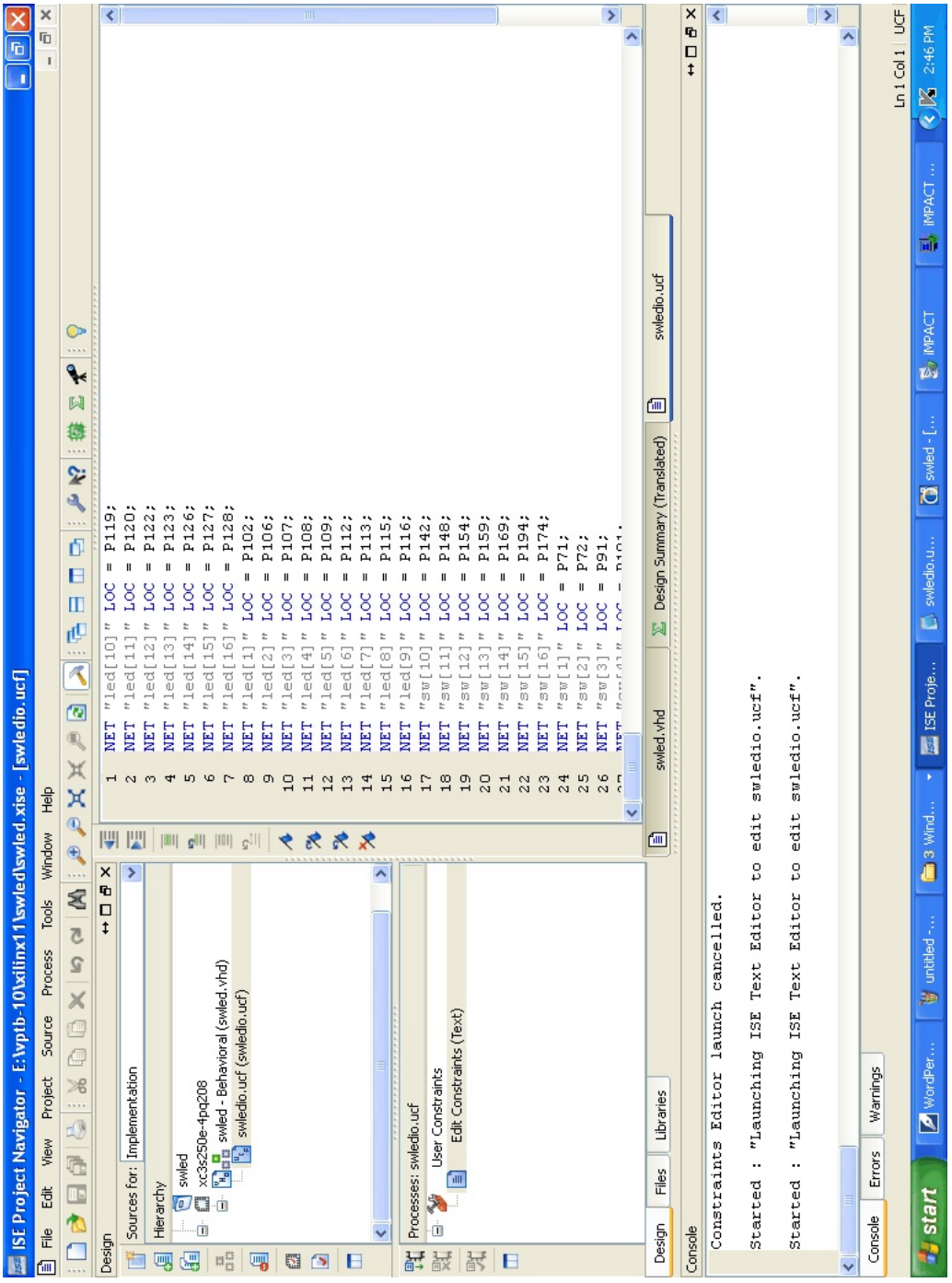

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise - [swledio.ucf]
File Edit View Project Source Process Tools Window Help
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
swled - Behavioral (swled.vhd)
swledio.ucf (swledio.ucf)
Processes: swledio.ucf
User Constraints
Edit Constraints (Text)
Design Files Libraries
swled.vhd
Design Summary (Translated)
swledio.ucf
Console
Errors
Warnings
Constraints Editor launch cancelled.
Started : "Launching ISE Text Editor to edit swledio.ucf".
Started : "Launching ISE Text Editor to edit swledio.ucf".
Ln 1 Col 1 UCF
start
2:46 PM

Figure-21b

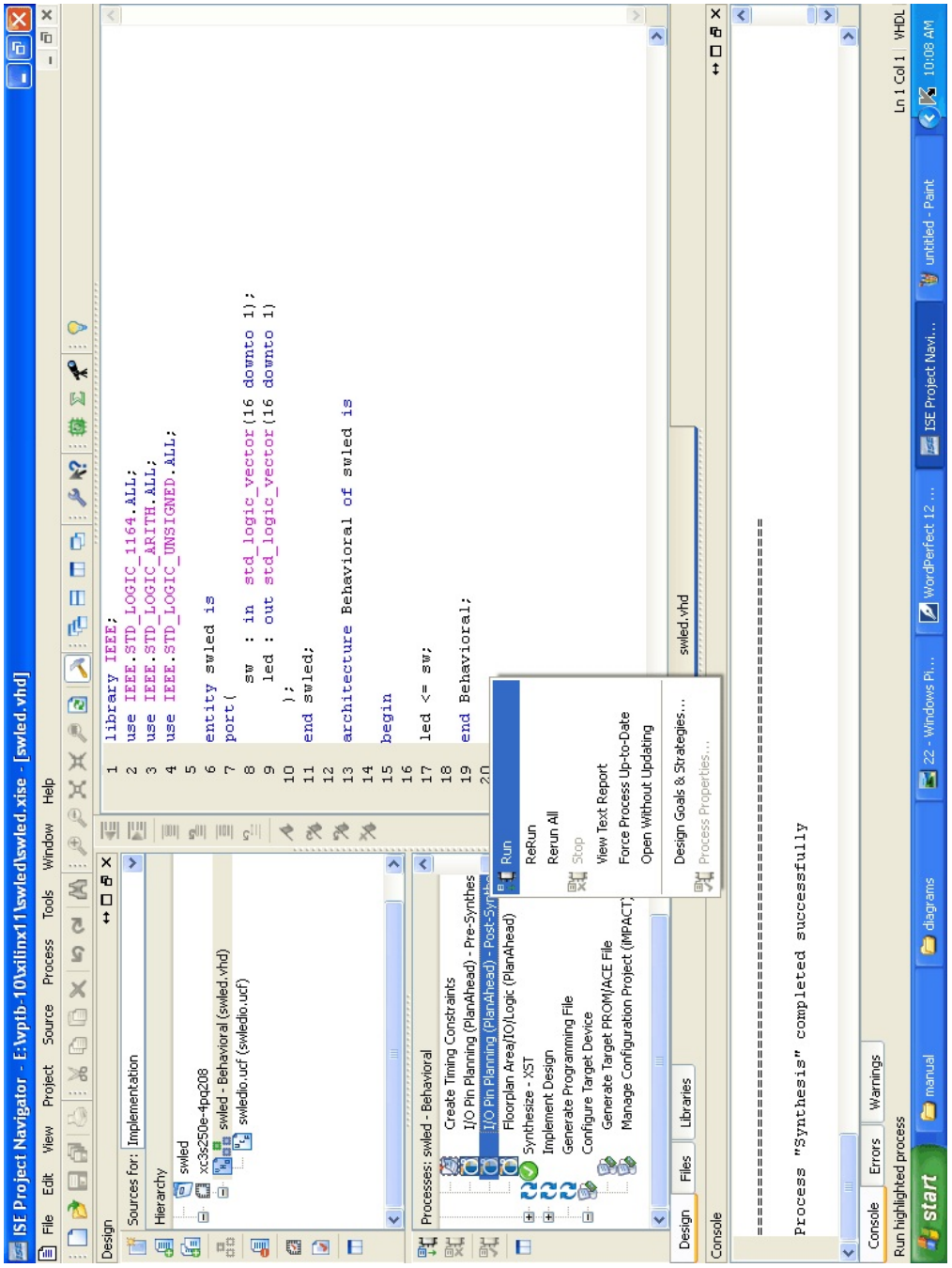

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise - [swled.vhd]
File Edit View Project Source Process Tools Window Help
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
swled - Behavioral (swled.vhd)
swledio.ucf (swledio.ucf)
Processes: swled - Behavioral
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Synthes
I/O Pin Planning (PlanAhead) - Post-Synth
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Generate Target PROM/ACE File
Manage Configuration Project (iMPACT)
Run
ReRun
Rerun All
Stop
View Text Report
Force Process Up-to-Date
Open Without Updating
Design Goals & Strategies...
Process Properties...
Design
Files
Libraries
swled.vhd
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_ARITH.ALL;
4 use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6 entity swled is
7 port(
8 sw : in std_logic_vector(16 downto 1);
9 led : out std_logic_vector(16 downto 1)
10);
11 end swled;
12
13 architecture Behavioral of swled is
14
15 begin
16
17 led <= sw;
18
19 end Behavioral;
20
Console
Process "Synthesis" completed successfully
Console
Errors
Warnings
Run highlighted process
Ln 1 Col 1 VHDL
start
manual
diagrams
22 - Windows Pi...
WordPerfect 12 ...
ISE Project Navi...
untitled - Paint
10:08 AM

Figure-22a

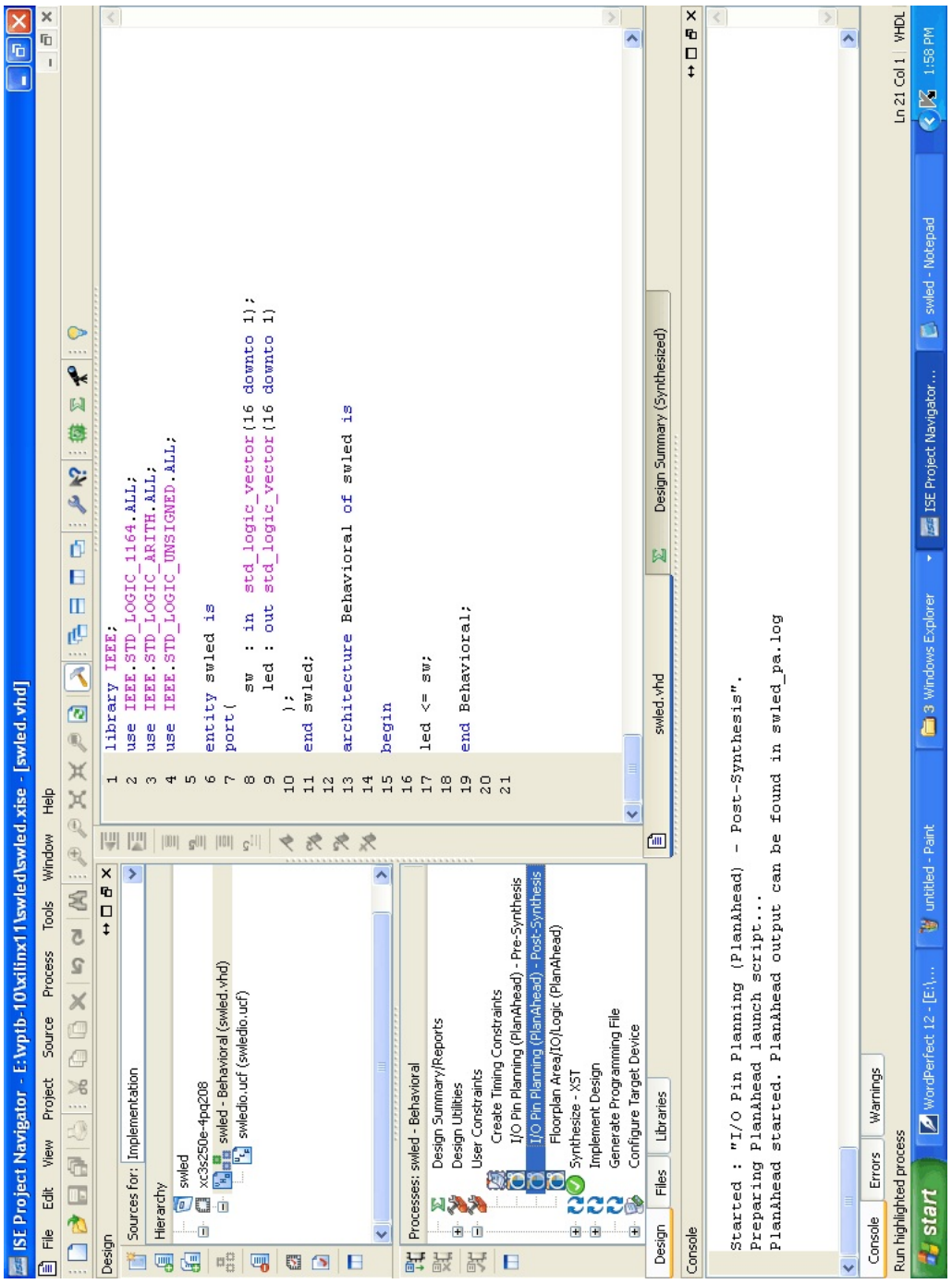

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise - [swled.vhd]
File Edit View Project Source Process Tools Window Help
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
swled - Behavioral (swled.vhd)
swledio.ucf (swledio.ucf)
Processes: swled - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Synthesis
I/O Pin Planning (PlanAhead) - Post-Synthesis
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Design
Files
Libraries
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_ARITH.ALL;
4 use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6 entity swled is
7 port (
8 sw : in std_logic_vector(16 downto 1);
9 led : out std_logic_vector(16 downto 1)
10);
11 end swled;
12
13 architecture Behavioral of swled is
14
15 begin
16
17 led <= sw;
18
19 end Behavioral;
20
21
swled.vhd
Design Summary (Synthesized)
Console
Started : "I/O Pin Planning (PlanAhead) - Post-Synthesis".
Preparing PlanAhead launch script...
PlanAhead started. PlanAhead output can be found in swled_pa.log
Console
Errors
Warnings
Run highlighted process
Ln 21 Col 1
VHDL
start
WordPerfect 12 - [E:\...
untitled - Paint
3 Windows Explorer
ISE Project Navigator...
swled - Notepad
1:58 PM

Figure 22b

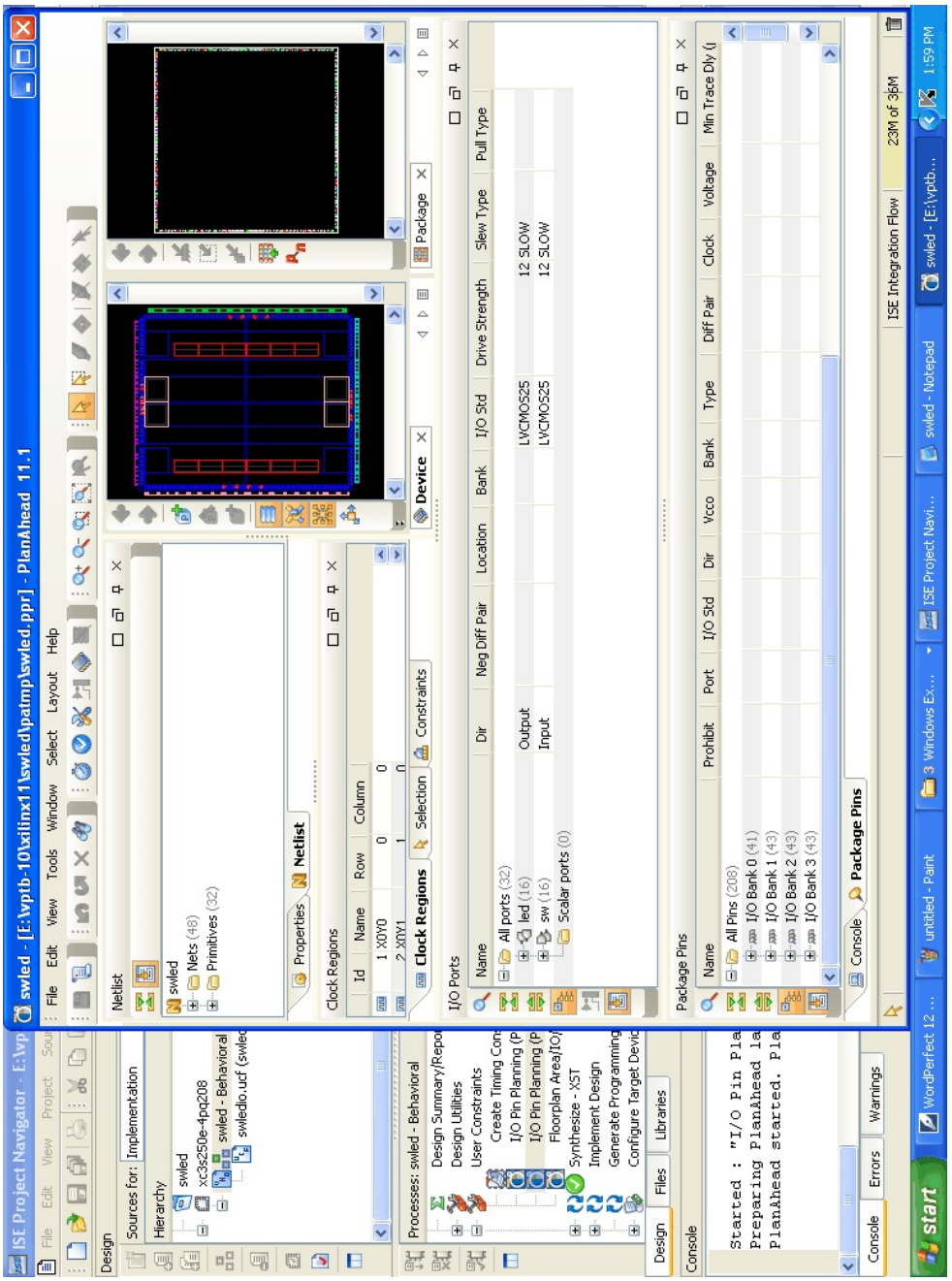


Figure-23

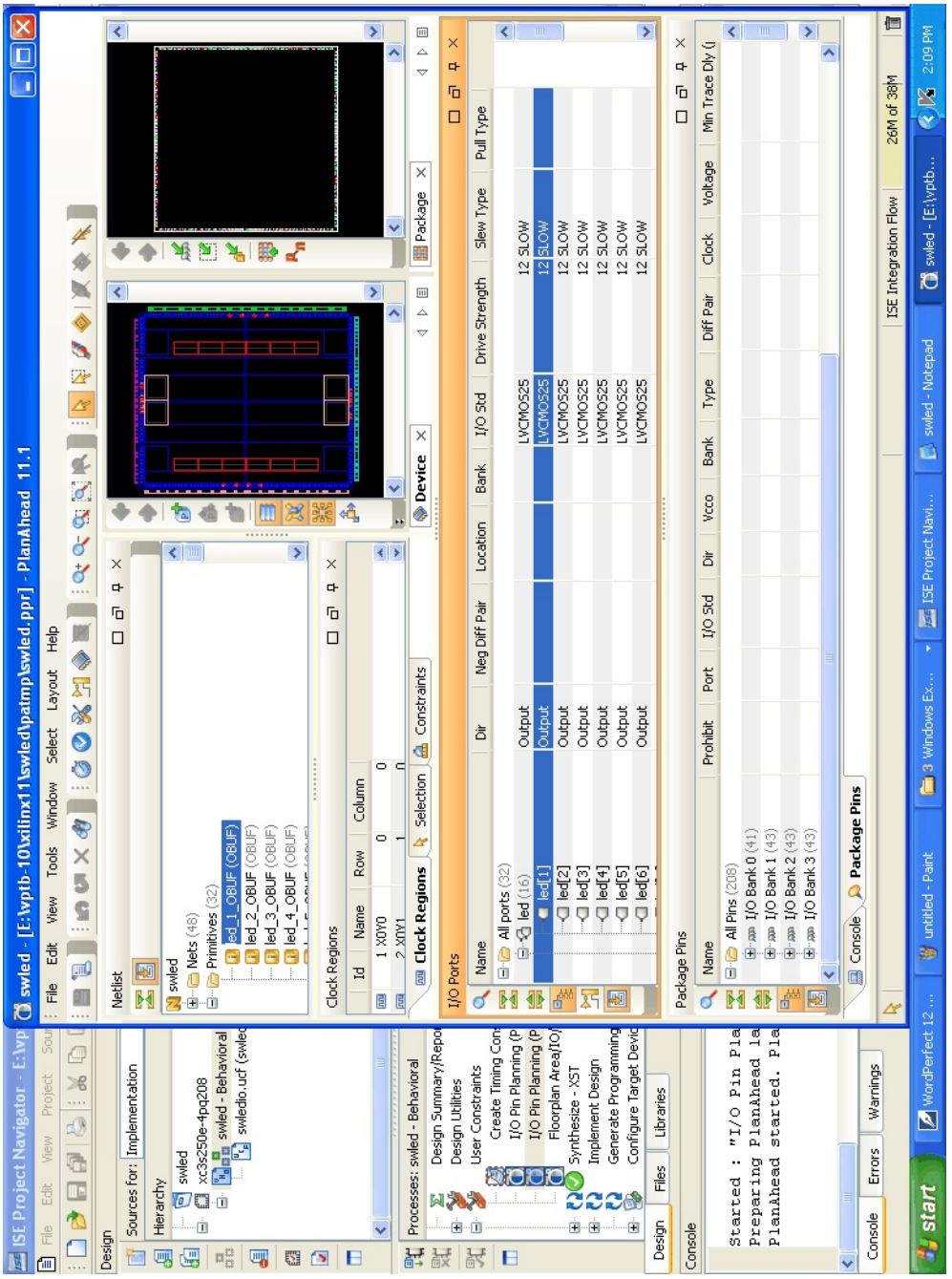


Figure-24

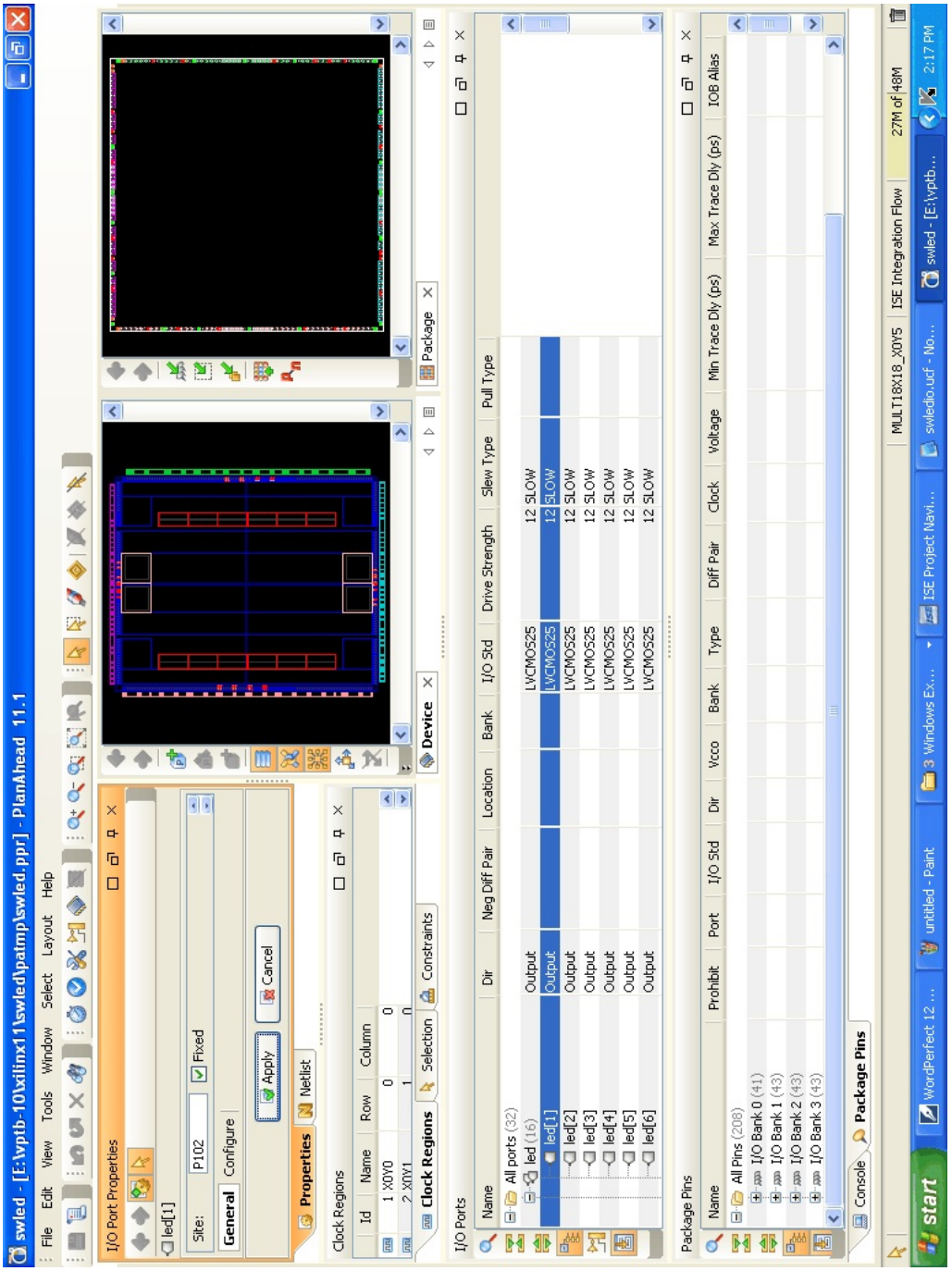


Figure-25

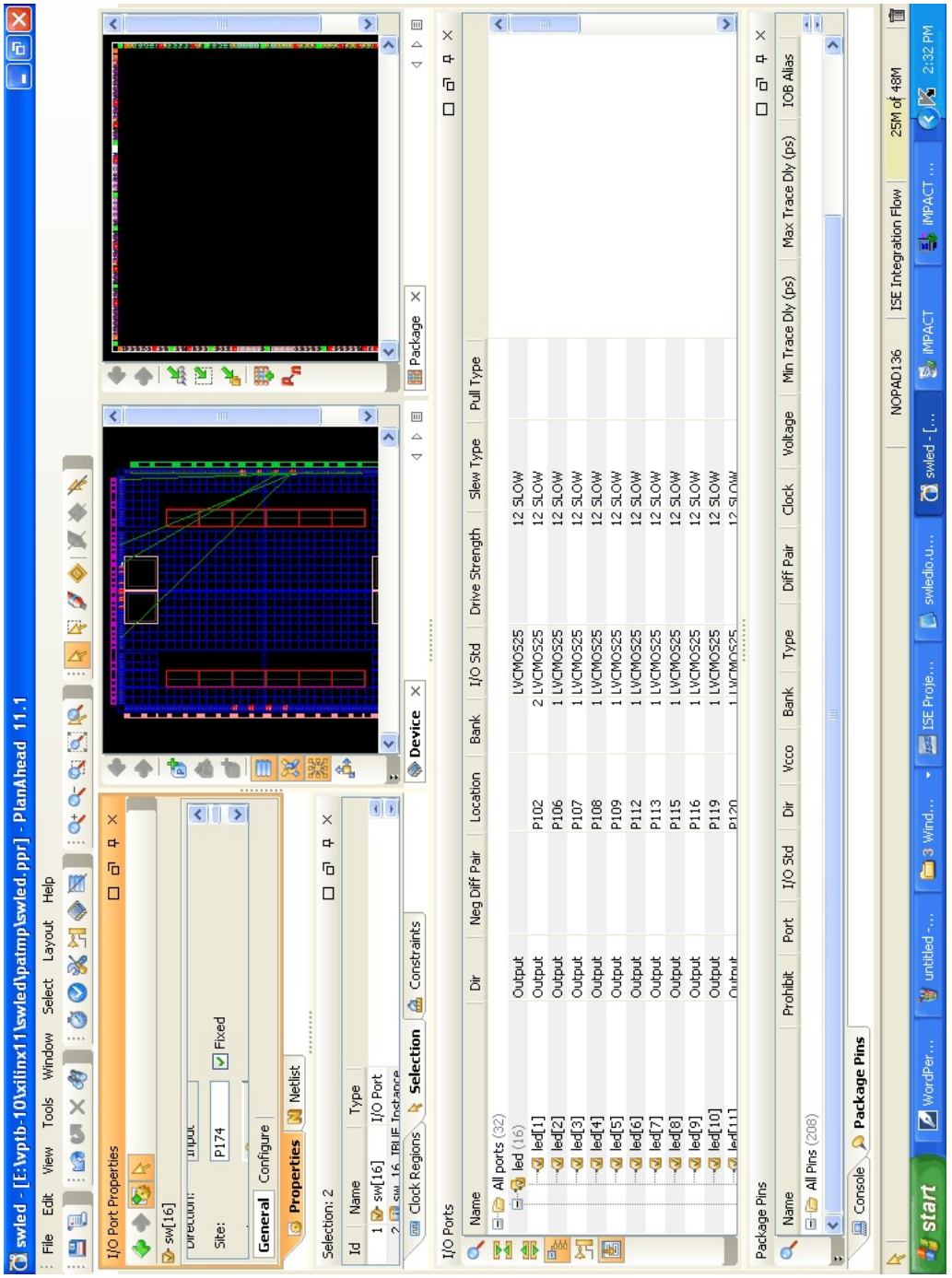


Figure-26

9. There are two ways of downloading the program into the target hardware, one is generating the**.bit file** for downloading into the **FPGA** device and the other is generating the.**mcs file** for downloading into the **PROM** device.

10. To download the program as **.bit** file in **Boundary Scan mode**, do the following steps:

    i) Shunt of Jumper **J2** of **VPTB-10** must be in **S3** position for **FPGA** selection.

    ii) In the **Project Navigator** window and click **Implement design** in **Processes:**category(refer **Figure-27**). After implement design has become successful(refer **Figure-28**), click **Generate Programming File** (refer **Figure-29**). After generate programming file has been completed successfully click **Manage Configuration Project (IMPACT)**(refer **Figure-30**).

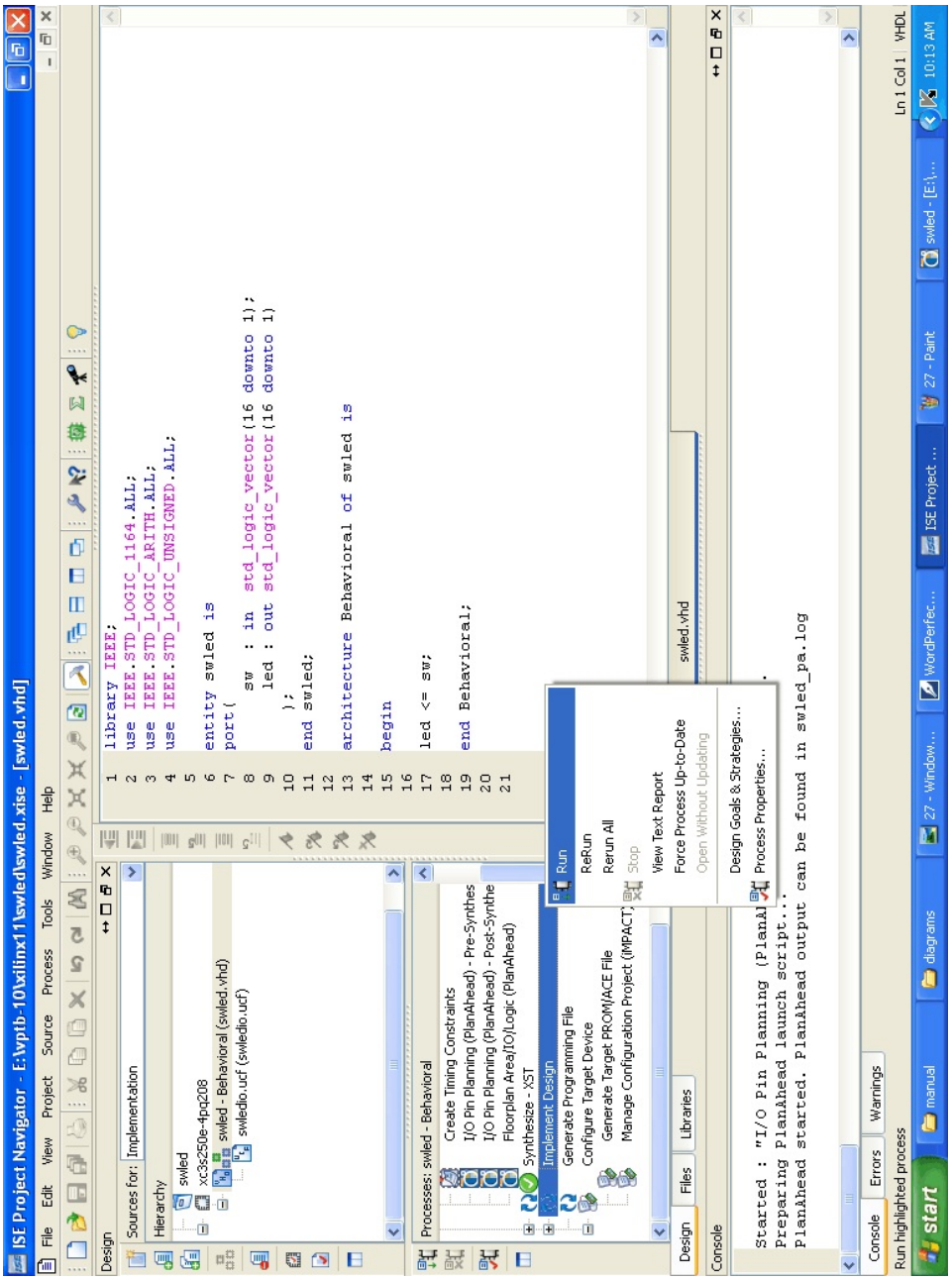


Figure-27

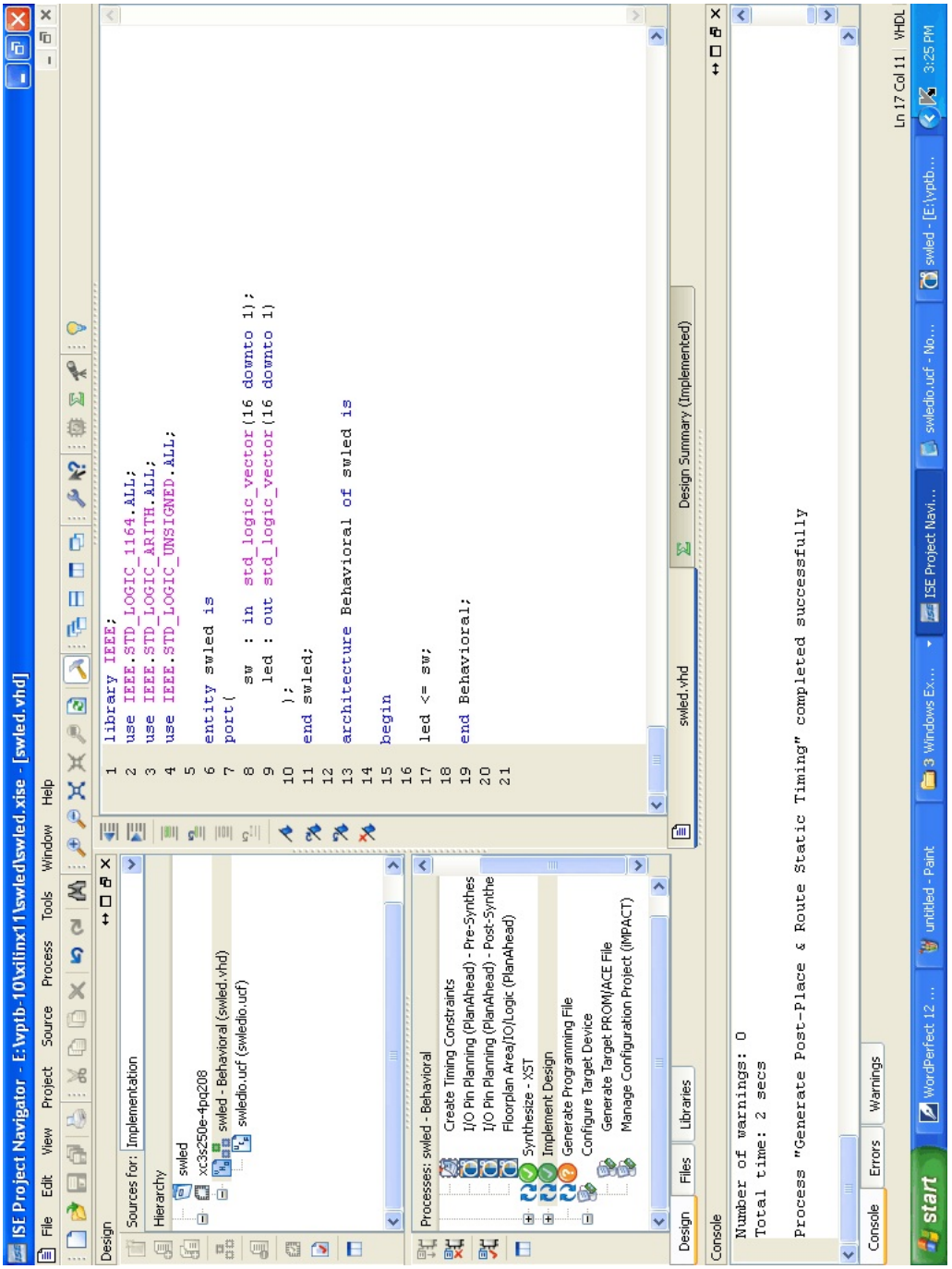

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise - [swled.vhd]
File Edit View Project Source Process Tools Window Help
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
swled - Behavioral (swled.vhd)
swledio.ucf (swledio.ucf)
Processes: swled - Behavioral
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Synthes
I/O Pin Planning (PlanAhead) - Post-Synthe
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Generate Target PROM/ACE File
Manage Configuration Project (iMPACT)
Design Files Libraries
swled.vhd
Design Summary (Implemented)
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_ARITH.ALL;
4 use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6 entity swled is
7 port(
8 sw : in std_logic_vector(16 downto 1);
9 led : out std_logic_vector(16 downto 1)
10);
11 end swled;
12
13 architecture Behavioral of swled is
14
15 begin
16
17 led <= sw;
18
19 end Behavioral;
20
21
Console
Number of warnings: 0
Total time: 2 secs
Process "Generate Post-Place & Route Static Timing" completed successfully
Console Errors Warnings
Ln 17 Col 11 VHDL
start
WordPerfect 12 ...
untitled - Paint
3 Windows Ex...
ISE Project Navi...
swledio.ucf - No...
swled - [E:\vptb...
3:25 PM

Figure-28

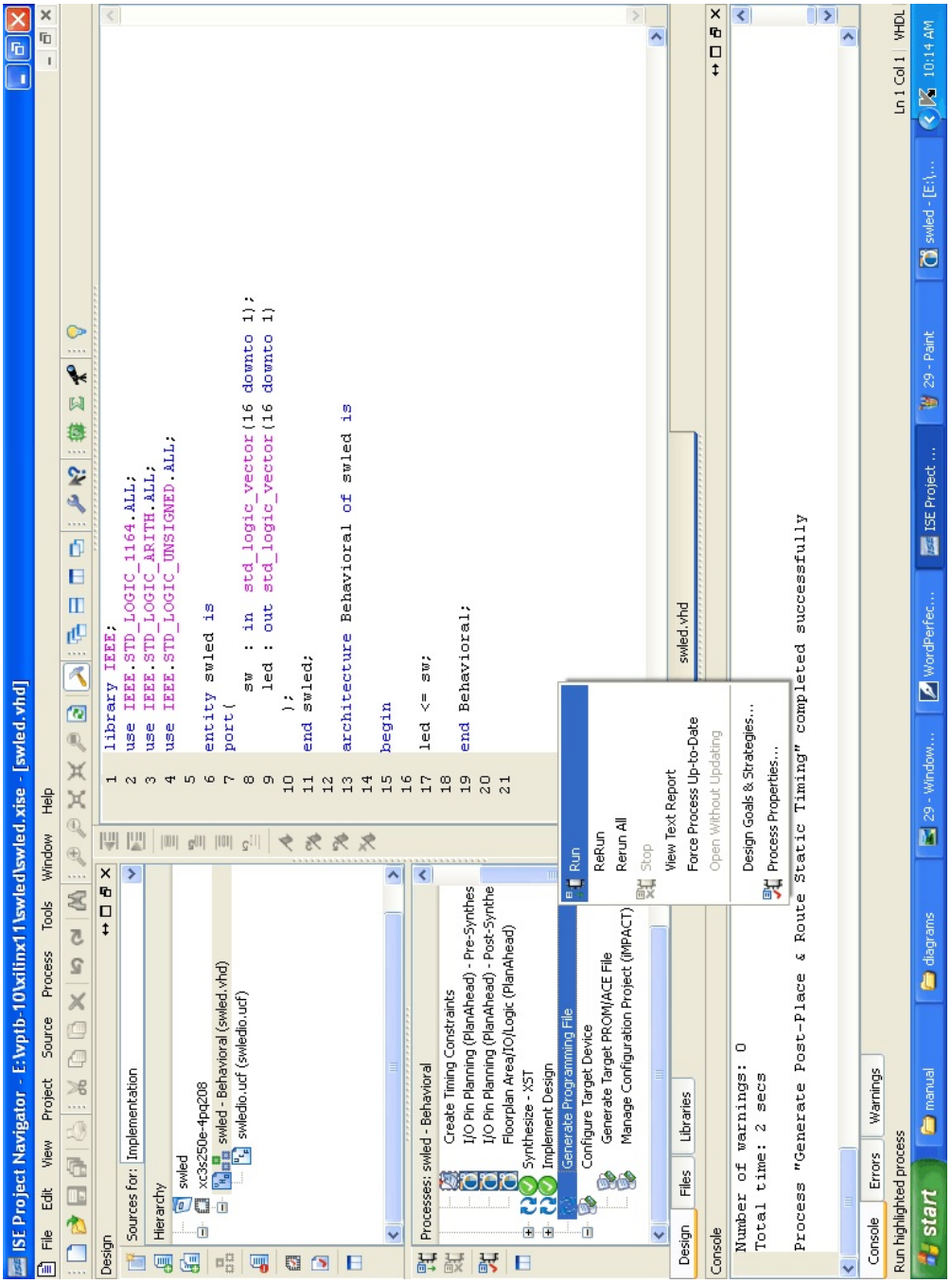

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise - [swled.vhd]
File Edit View Project Source Process Tools Window Help
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
swled - Behavioral (swled.vhd)
swledio.ucf (swledio.ucf)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity swled is
port(
sw : in std_logic_vector(16 downto 1);
led : out std_logic_vector(16 downto 1)
);
end swled;
architecture Behavioral of swled is
begin
led <= sw;
end Behavioral;
Processes: swled - Behavioral
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Synthes
I/O Pin Planning (PlanAhead) - Post-Synthe
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Generate Target PROM/ACE File
Manage Configuration Project (iMPACT)
Run
ReRun
Rerun All
Stop
View Text Report
Force Process Up-to-Date
Open Without Updating
Design Goals & Strategies...
Process Properties...
swled.vhd
Design
Files
Libraries
Console
Number of warnings: 0
Total time: 2 secs
Process "Generate Post-Place & Route Static Timing" completed successfully
Console
Errors
Warnings
Run highlighted process
Ln 1 Col 1
VHDL
start
manual
diagrams
29 - Window...
WordPerfec...
ISE Project ...
29 - Paint
swled - [E:\...
10:14 AM

Figure-29

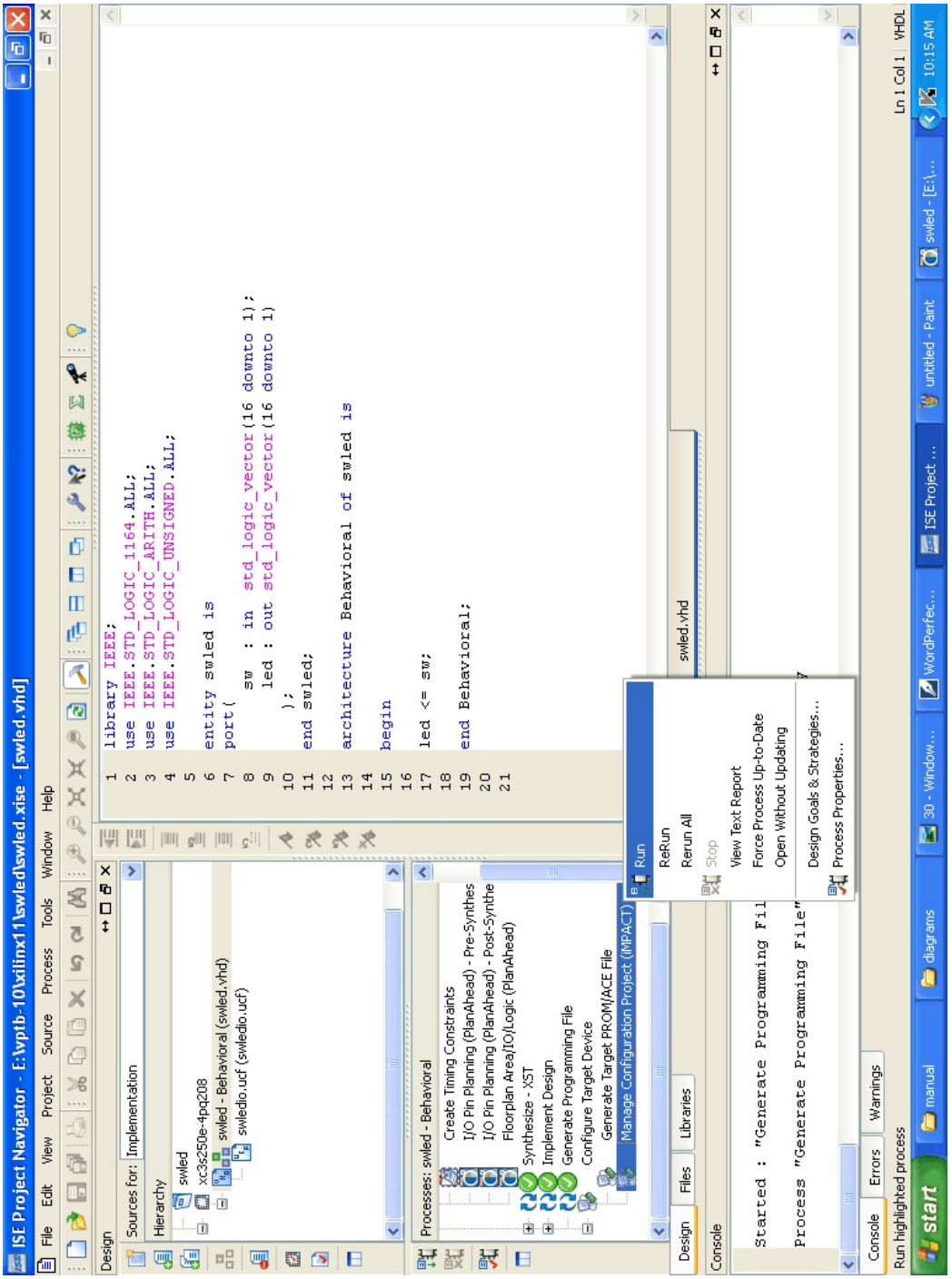

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise - [swled.vhd]
File Edit View Project Source Process Tools Window Help
Design
Sources for: Implementation
Hierarchy
swled
xc3s250e-4pq208
swled - Behavioral (swled.vhd)
swledio.ucf (swledio.ucf)
Processes: swled - Behavioral
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Synthes
I/O Pin Planning (PlanAhead) - Post-Synthe
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Generate Target PROM/ACE File
Manage Configuration Project (IMPACT)
Design Files Libraries
Console
Started : "Generate Programming Fil
Process "Generate Programming File"
Console Errors Warnings
Run highlighted process
Run
ReRun
Rerun All
Stop
View Text Report
Force Process Up-to-Date
Open Without Updating
Design Goals & Strategies...
Process Properties...
swled.vhd
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_ARITH.ALL;
4 use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6 entity swled is
7 port(
8 sw : in std_logic_vector(16 downto 1);
9 led : out std_logic_vector(16 downto 1)
10);
11 end swled;
12
13 architecture Behavioral of swled is
14
15 begin
16
17 led <= sw;
18
19 end Behavioral;
20
21
Ln 1 Col 1 VHDL
start
manual
diagrams
30 - Window...
WordPerfec...
ISE Project ...
untitled - Paint
swled - [E:\...
10:15 AM

Figure-30

11. A new window named **ISE iMPACT** appears as shown in **Figure-31**.In that window under the **iMPACT Flows** category click the Boundary Scan option (refer **Figure-32**).A new tab named **Boundary Scan** appears as shown in **Figure-32a**. Now right click in that tab to select the **Initialize Chain** option from the pop-up menu that appears (refer **Figure-32b**).

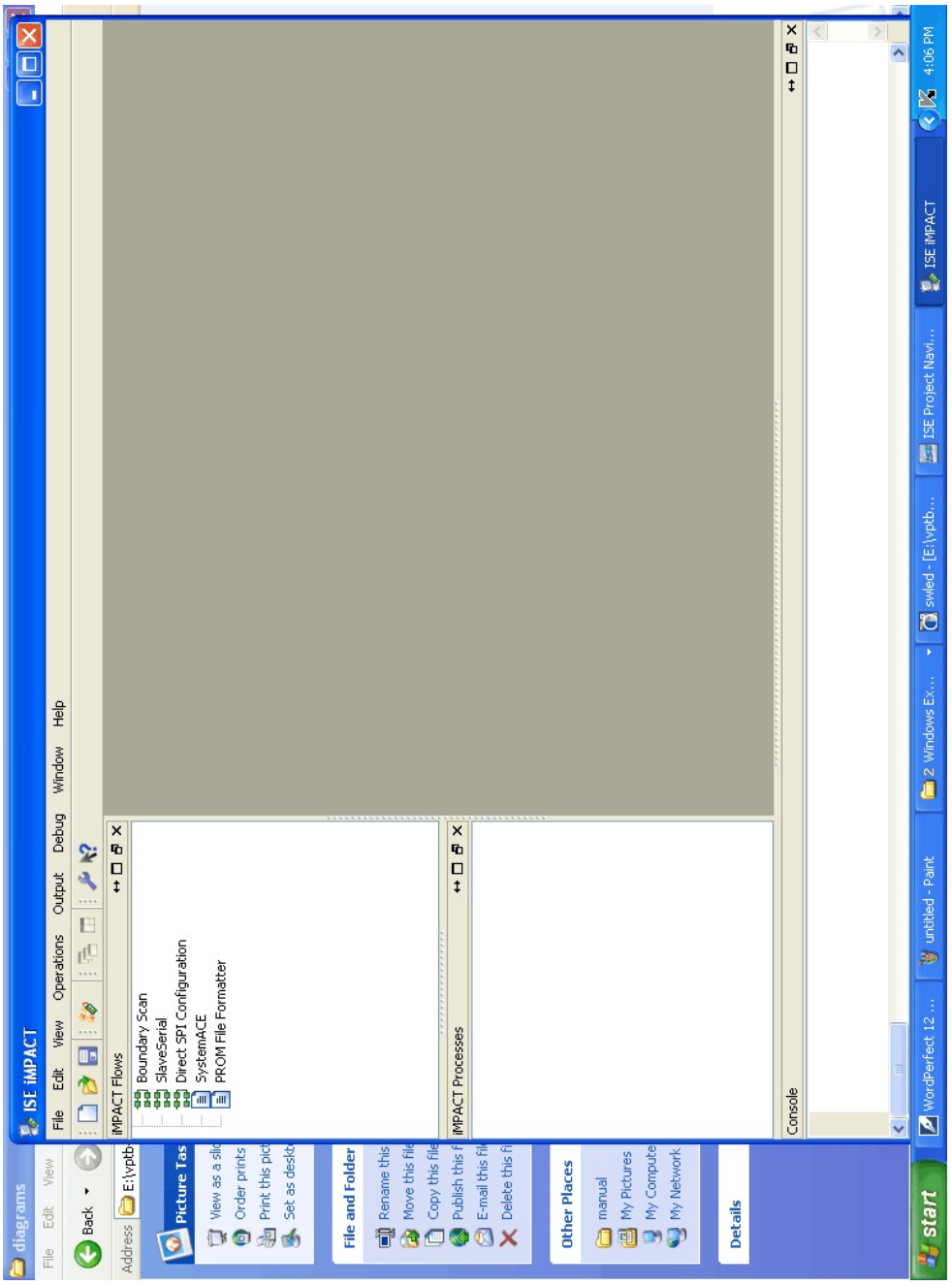


Figure-31

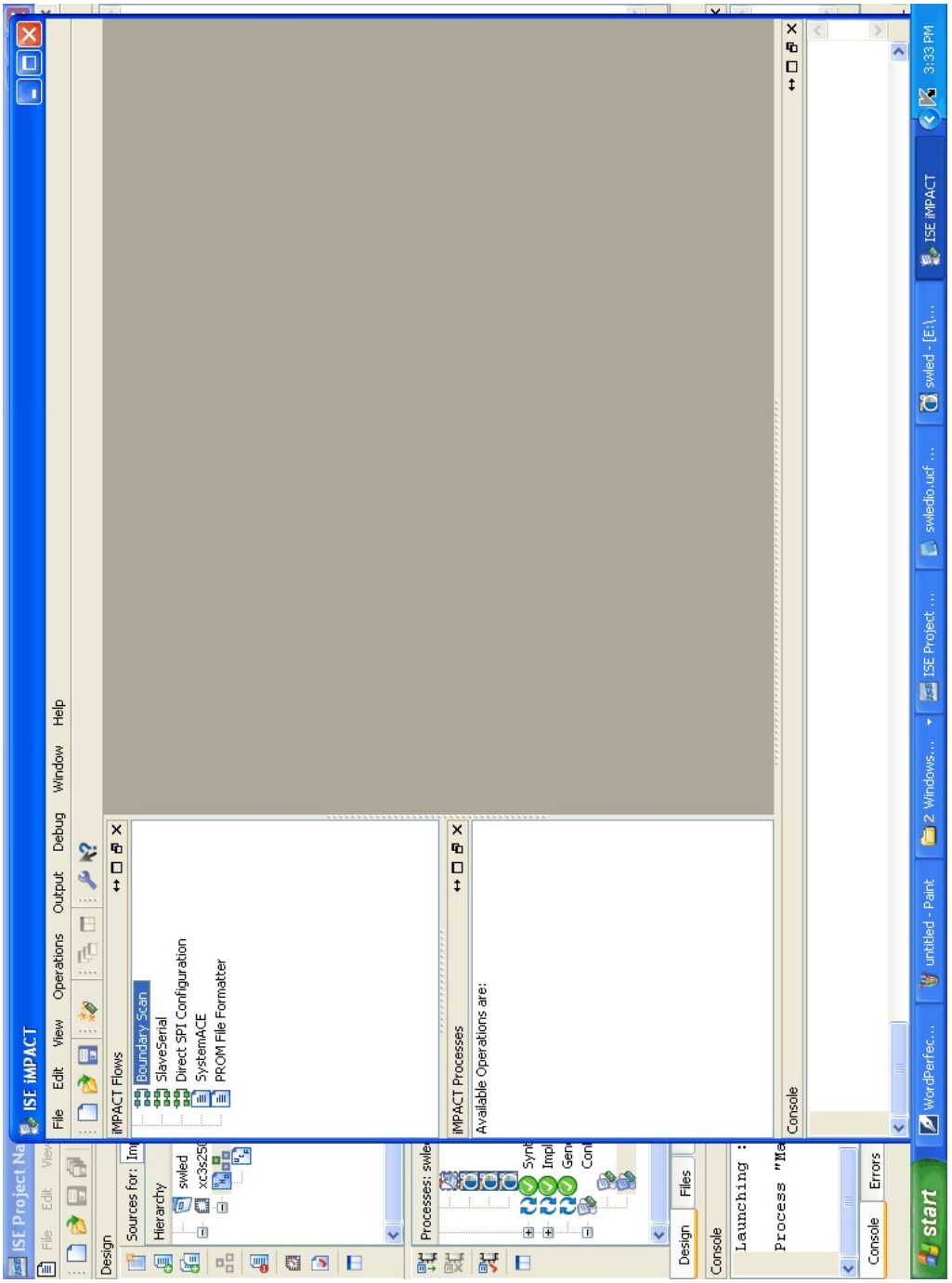

ISE iMPACT
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Console
Design
Sources for: Im
Hierarchy
swled
Processes: swled
Design Files
Console
Launching :
Process "Ma
Console Errors
start
WordPerfec...
untitled - Paint
2 Windows...
ISE Project ...
swledio.ucf ...
swled - [E:\...
ISE iMPACT
3:33 PM

Figure-32

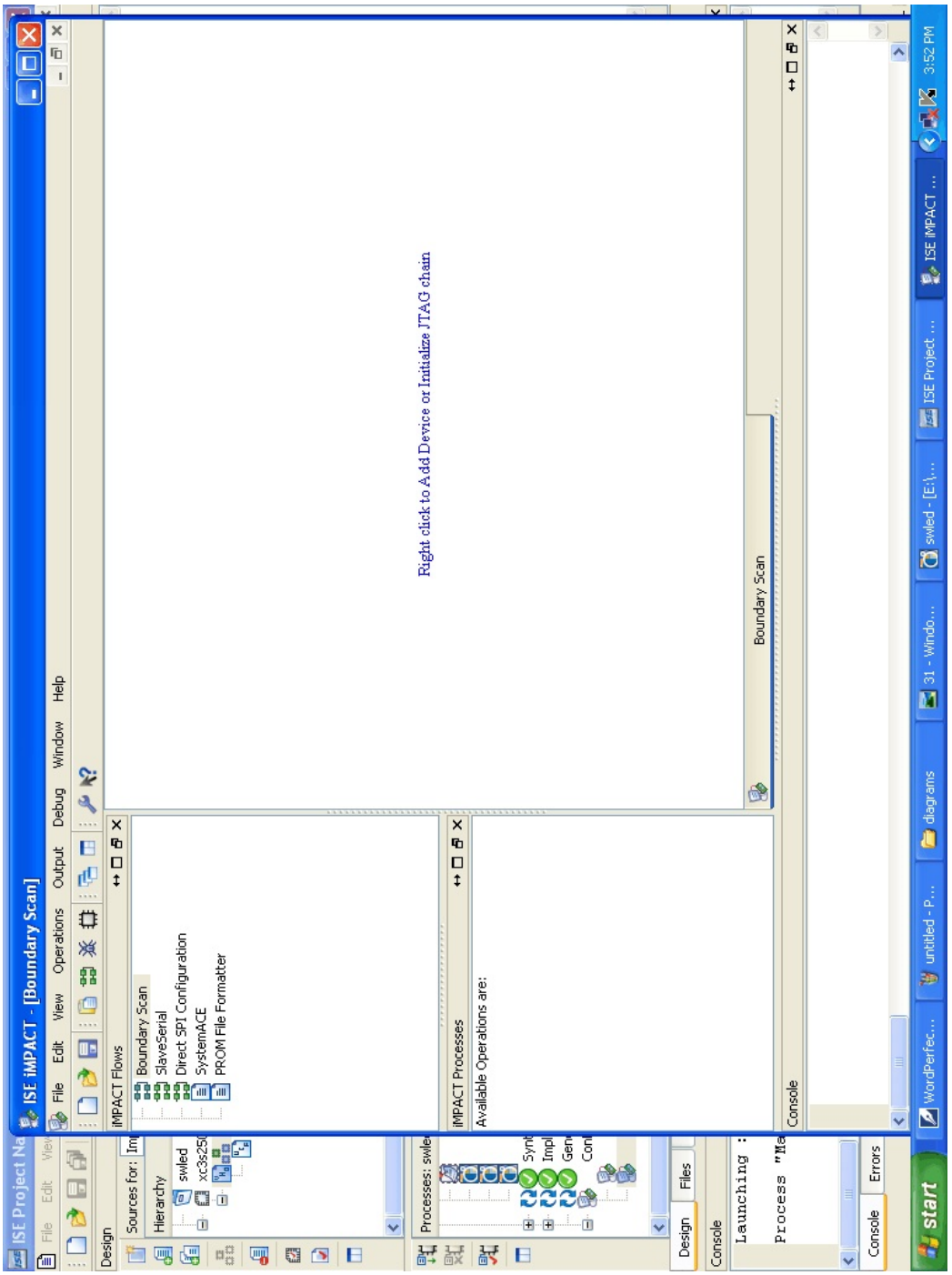

ISE iMPACT - [Boundary Scan]
File Edit View Operations Output Debug Window Help
Right click to Add Device or Initialize JTAG chain
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Boundary Scan
Console
Design
Hierarchy
Processes: swled
Design Files
Console
Launching :
Console Errors
start
3:52 PM

Figure-32a

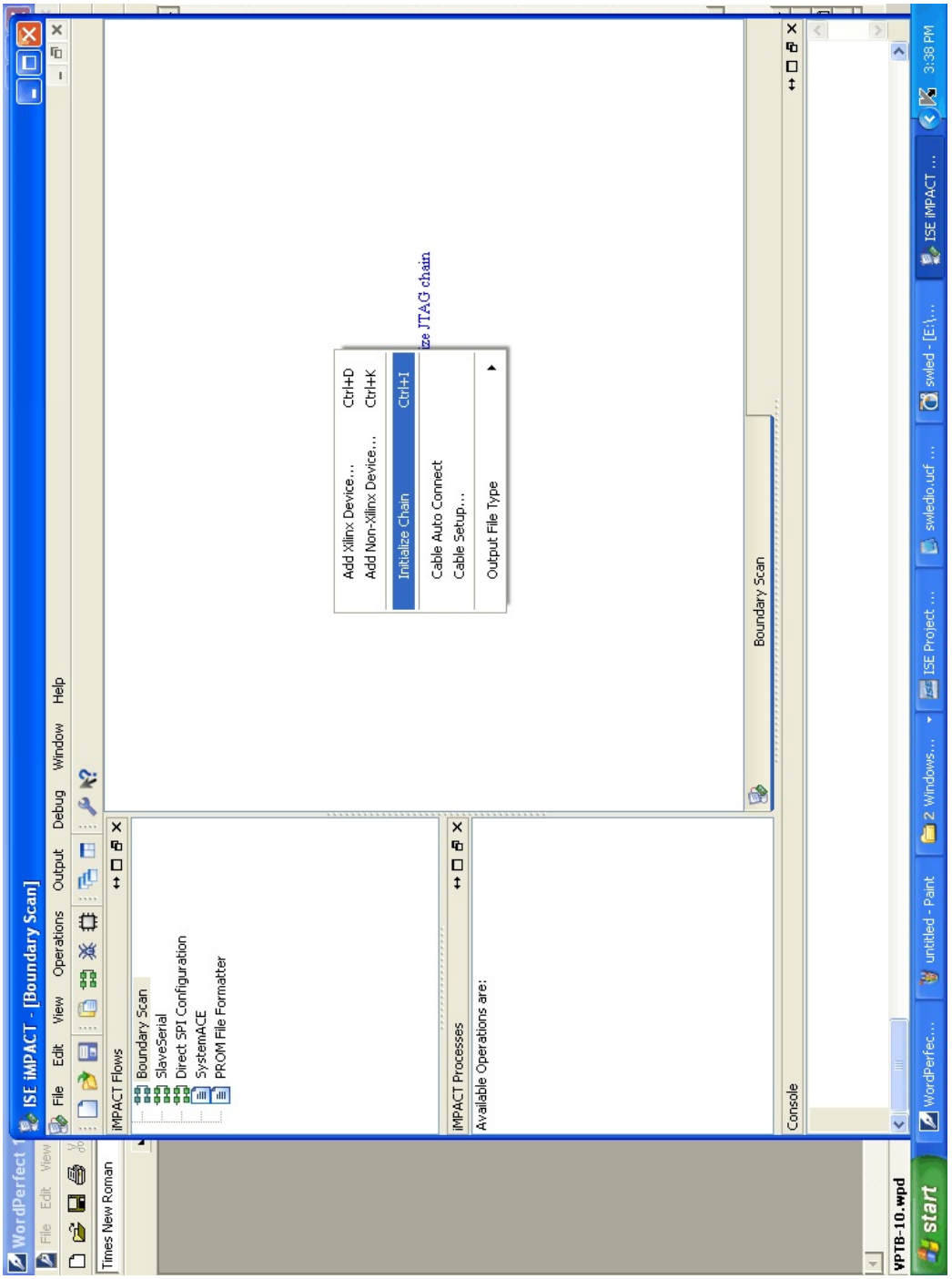

ISE iMPACT - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Boundary Scan
Console
Add Xilinx Device... Ctrl+D
Add Non-Xilinx Device... Ctrl+K
Initialize Chain Ctrl+I
Cable Auto Connect
Cable Setup...
Output File Type
ze JTAG chain
WordPerfect
Times New Roman
VPTB-10.wpd
start
WordPerfec...
untitled - Paint
2 Windows...
ISE Project ...
swledio.ucf ...
swled - [E:\...
ISE iMPACT ...
3:38 PM

Figure-32b

12. A new window named **Assign New Configuration File** as shown in **Figure-33** will appear. Now select the respective **.bit** file from the correct project path you want to download in that window and click **Open**(refer **Figure-34**).In the new window that appears click **No** (refer **Figure-35).** In the new window click **OK** (refer **Figure 36a, 36b).**

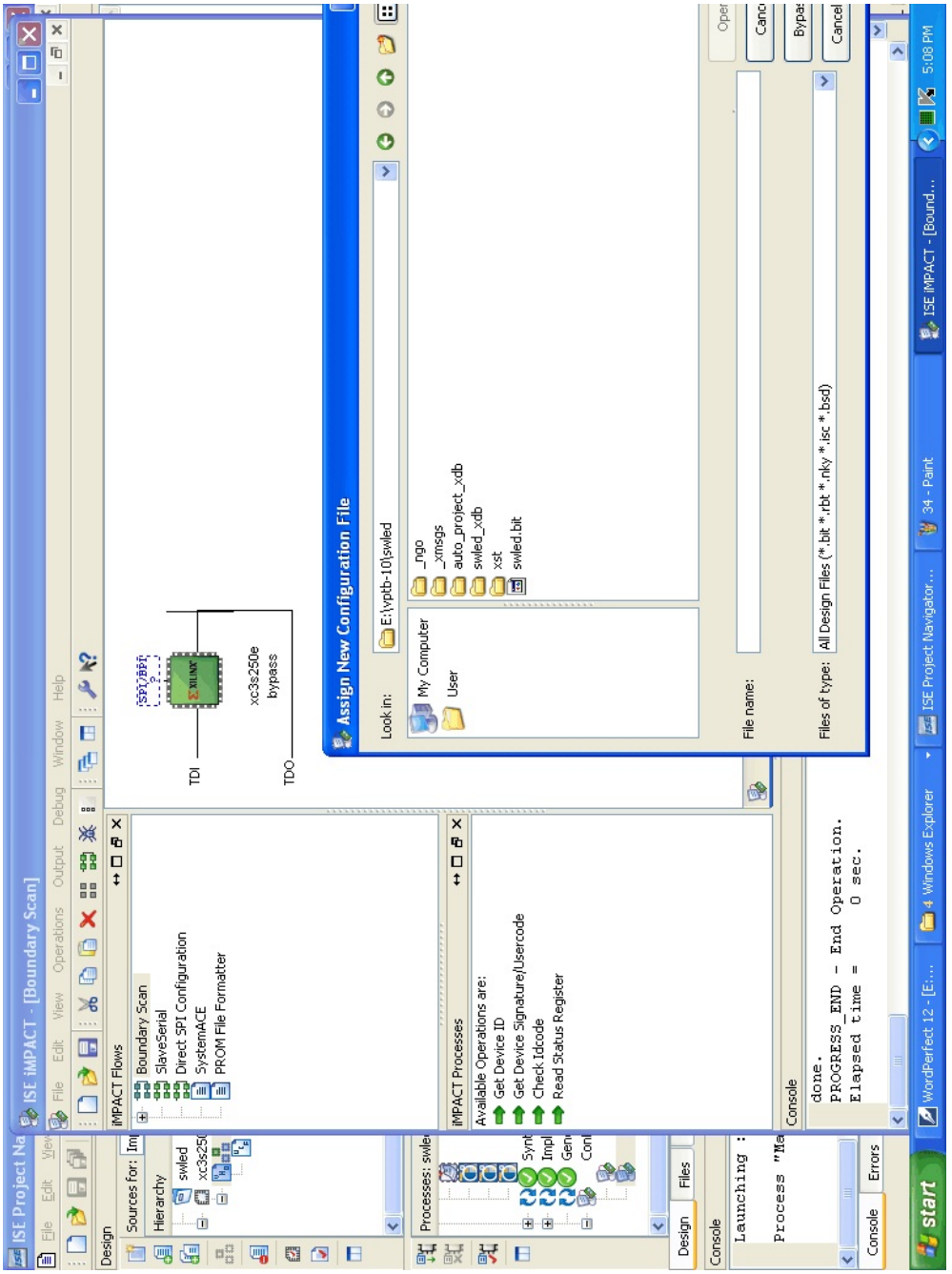


Figure-33

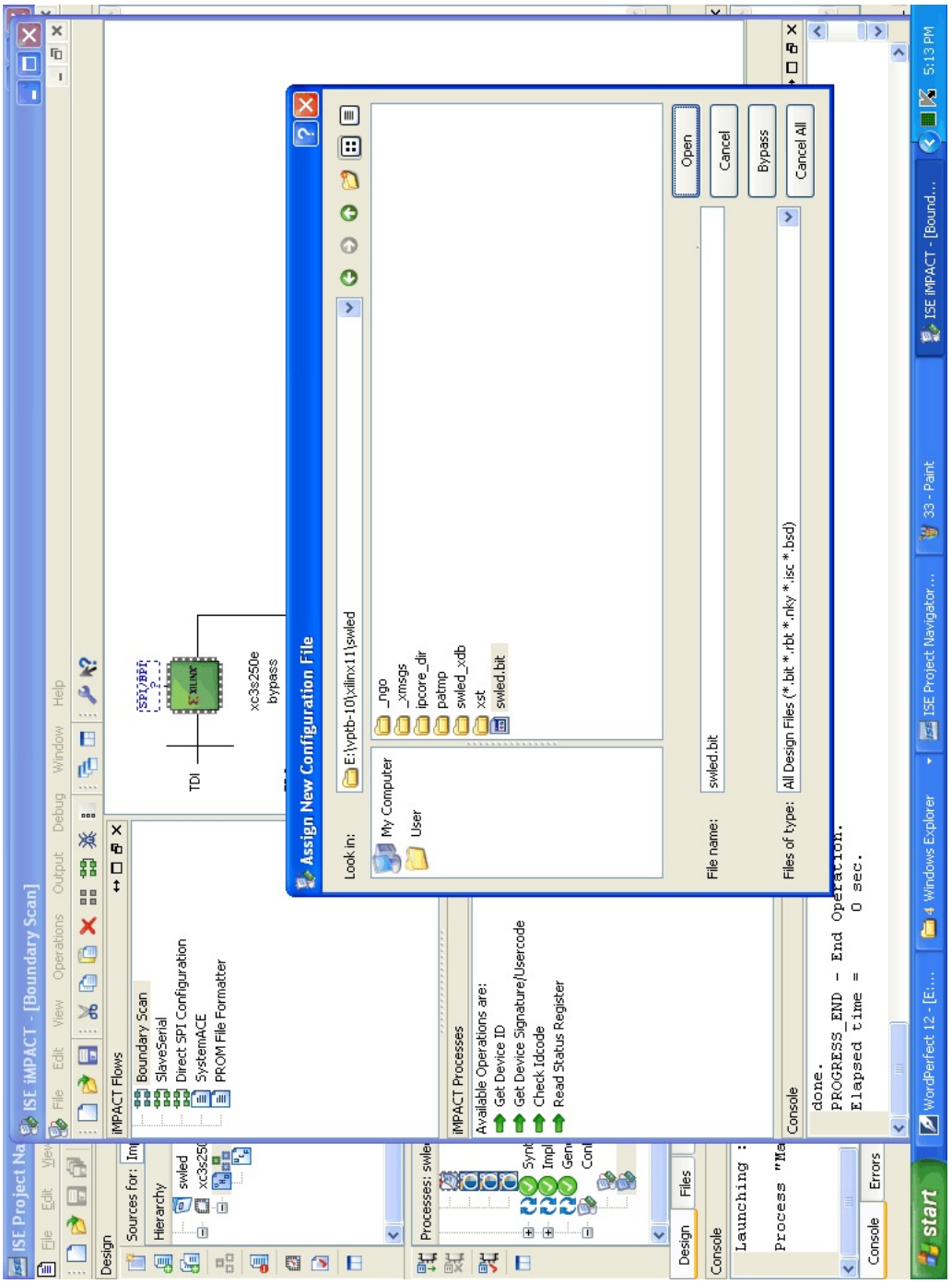

ISE iMPACT - [Boundary Scan]
Assign New Configuration File
Look in:
E:\vptb-10\xilinx11\swled
_ngo
_xmsgs
ipcore_dir
patmp
swled_xdb
xst
swled.bit
My Computer
User
File name:
swled.bit
Files of type:
All Design Files (*.bit *.rbt *.nky *.isc *.bsd)
Open
Cancel
Bypass
Cancel All
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Get Device ID
Get Device Signature/Usercode
Check Idcode
Read Status Register
TDI
xc3s250e
bypass
Console
done.
PROGRESS_END - End Operation.
Elapsed time = 0 sec.
Launching :
Design
Files
Errors
start

Figure-34

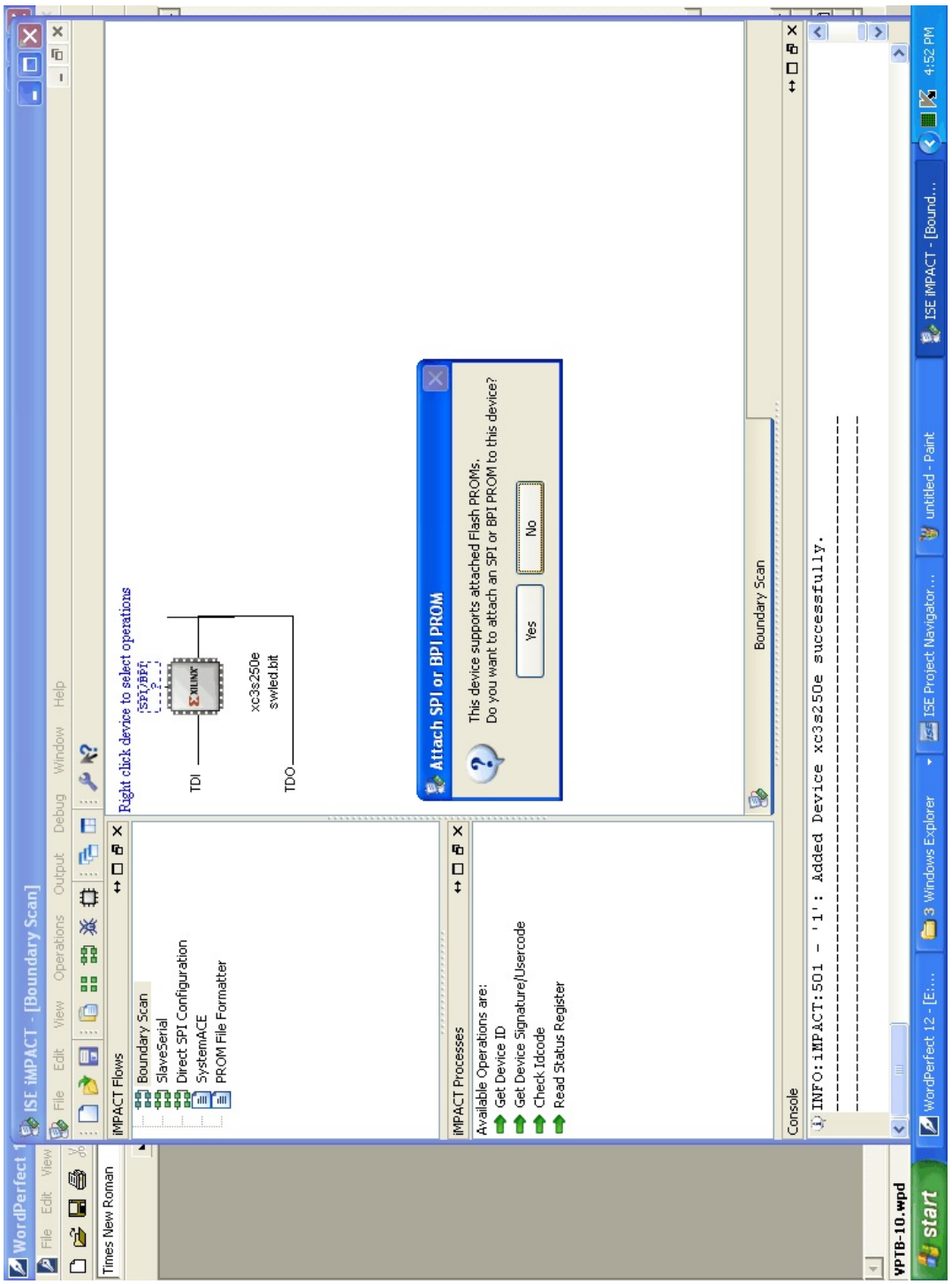

ISE iMPACT - [Boundary Scan]
Right click device to select operations
TDI
TDO
xc3s250e
swled.bit
Attach SPI or BPI PROM
This device supports attached Flash PROMs.
Do you want to attach an SPI or BPI PROM to this device?
Yes
No
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Get Device ID
Get Device Signature/Usercode
Check Idcode
Read Status Register
Console
INFO:iMPACT:501 - '1': Added Device xc3s250e successfully.
Boundary Scan

Figure-35

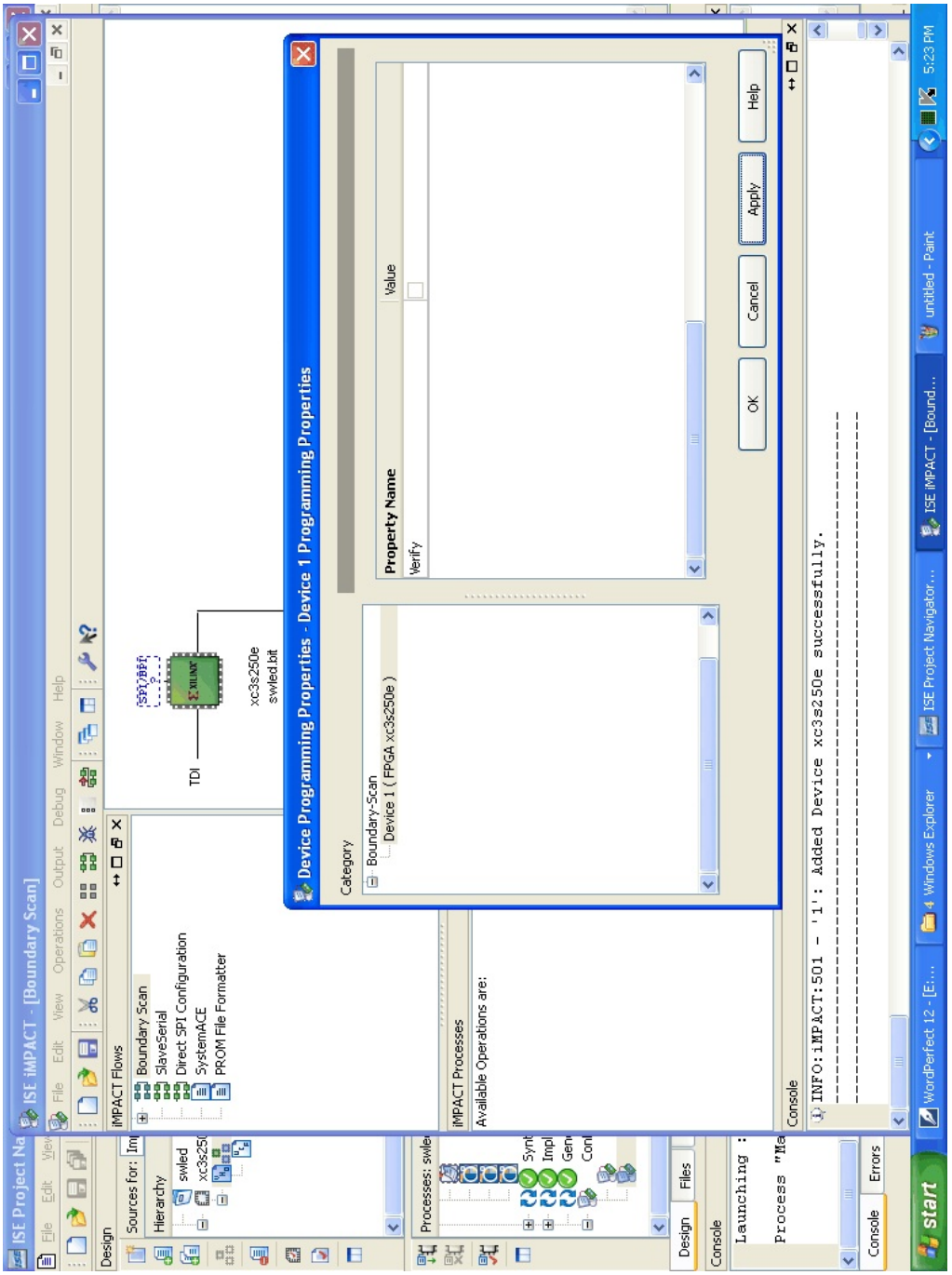

ISE iMPACT - [Boundary Scan]
Device Programming Properties - Device 1 Programming Properties
Category
Boundary-Scan
Device 1 (FPGA xc3s250e)
Property Name
Value
Verify
OK
Cancel
Apply
Help
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Console
INFO:iMPACT:501 - '1': Added Device xc3s250e successfully.
TDI
xc3s250e
swled.bit

Figure 36a

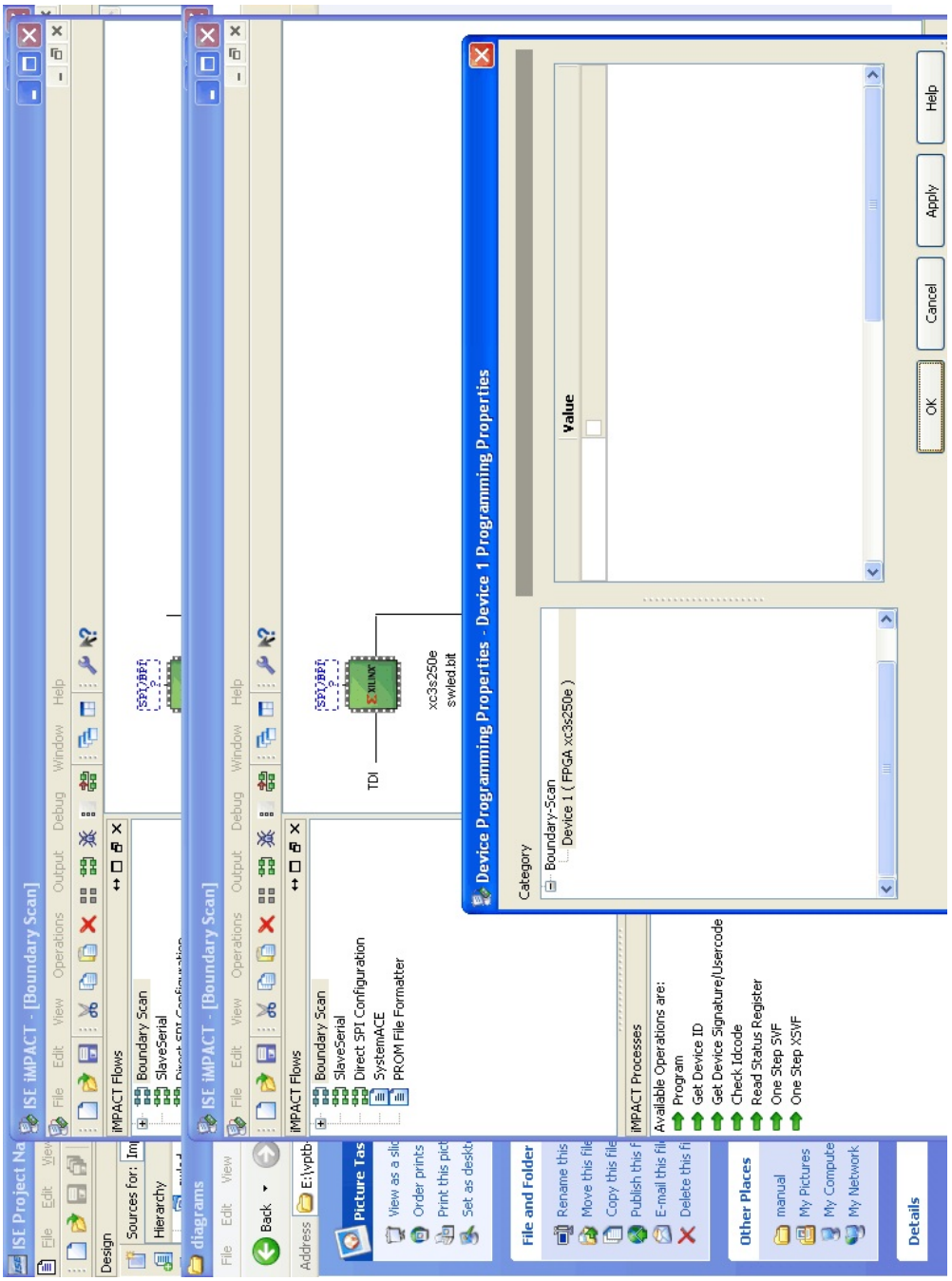


Figure-36b

13. Before downloading the program, connect the downloading cable into the **JTAG(P4)** connector of the trainer kit.

14. Power **ON** the board.

15. Right Click the chip **xc3s250e** and select **Program** (Refer **Figure-**37).

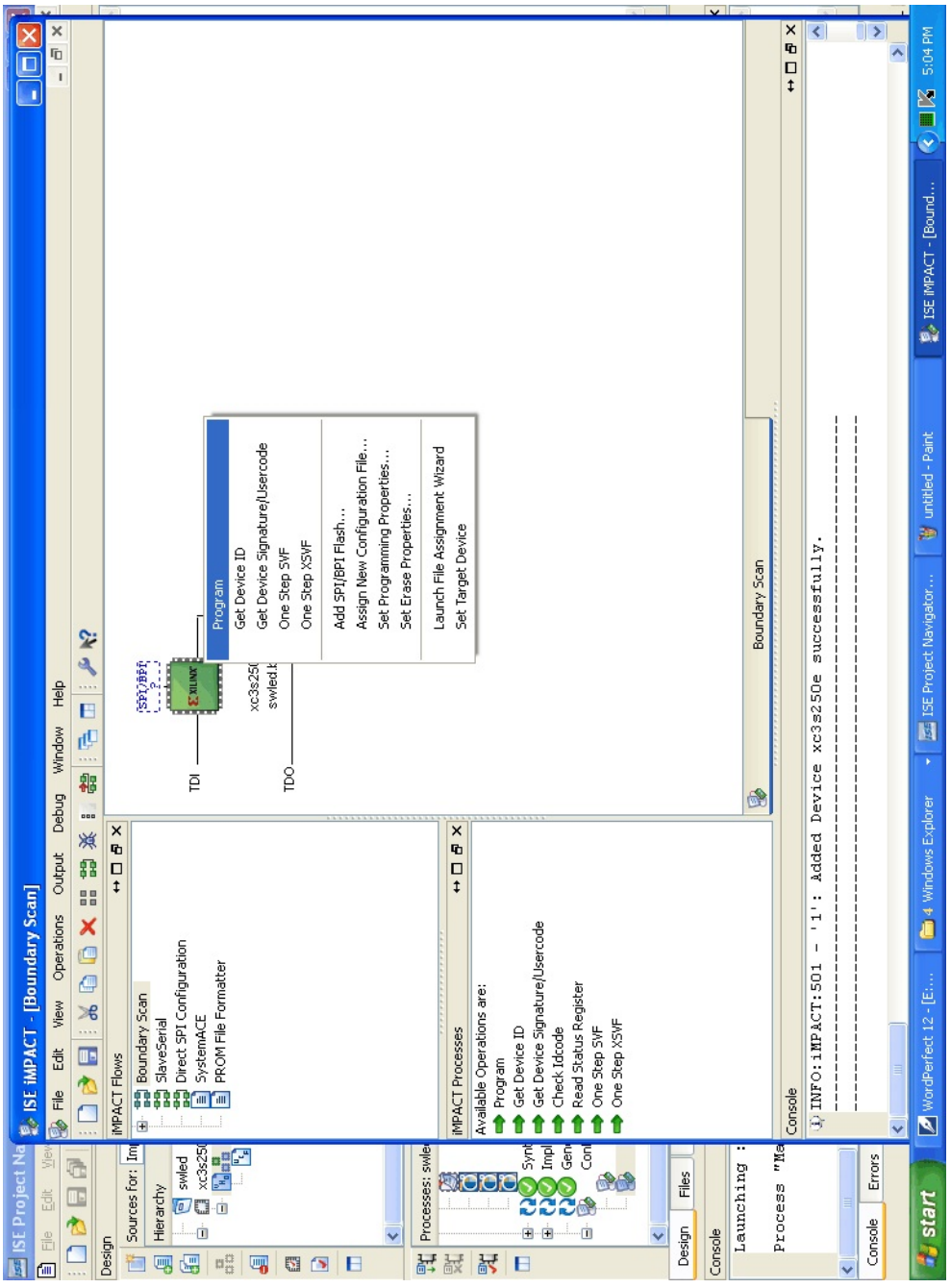


Figure-37

16. Then a window named **Configuration Operation Status** with *percentage level* shown in **Figure-38** will appear and it starts downloading your program into the hardware. After the *percentage level* reaches 100% program is downloaded successfully into the device and the message "**Program Succeeded**" will appear in bold. (refer **Figure-39**).

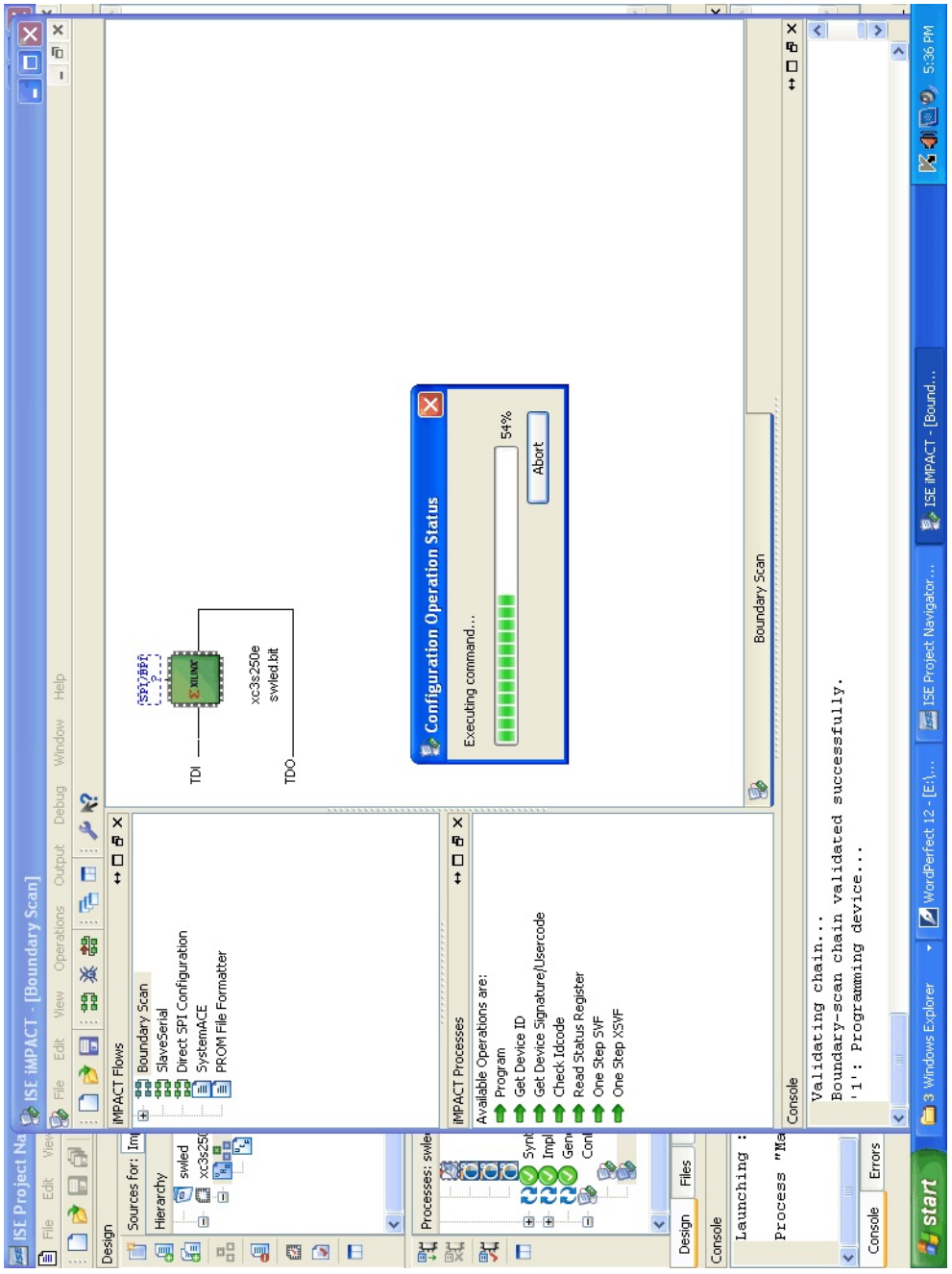


Figure-38

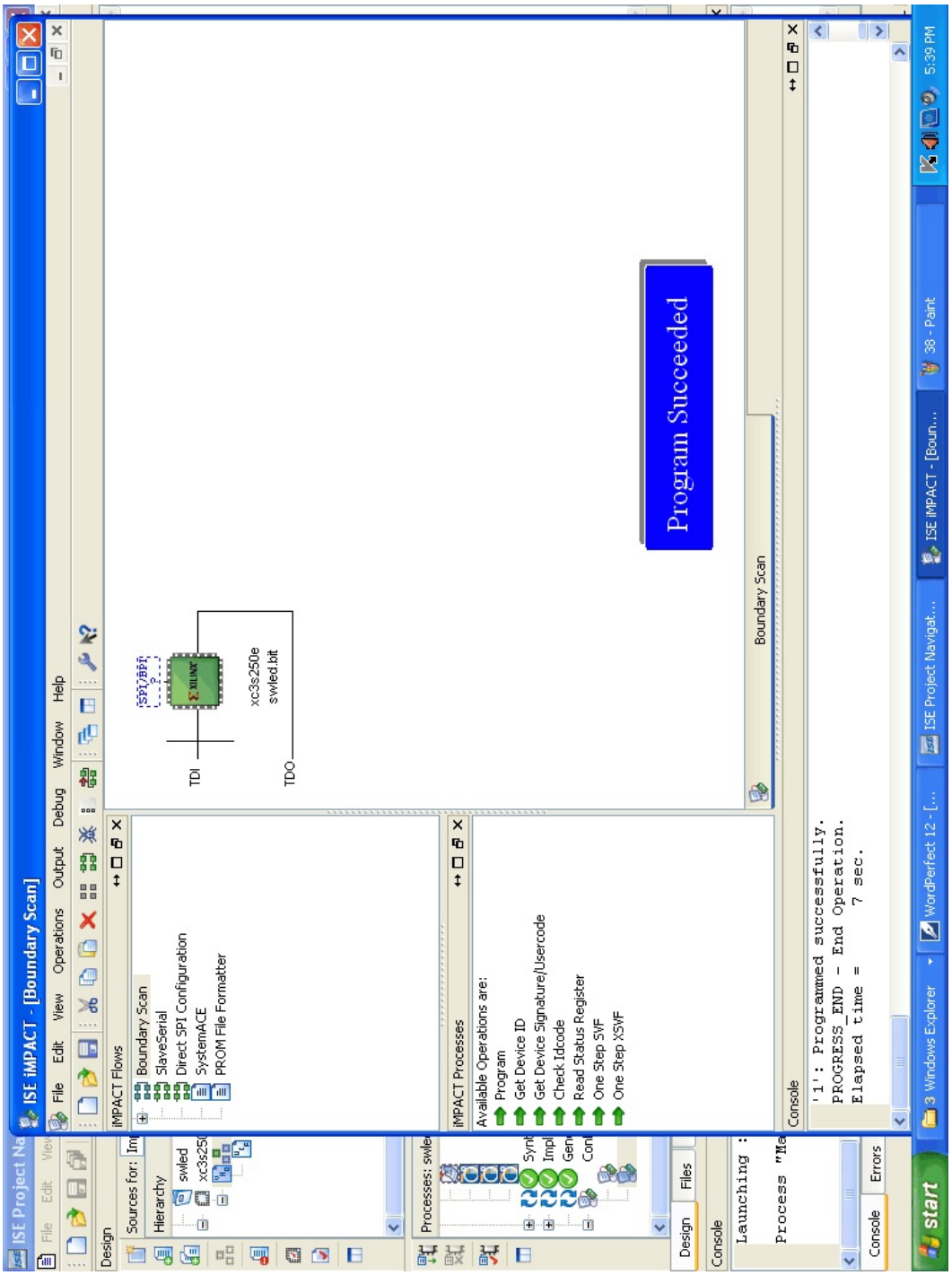


Figure-39

17. After successful configuration you can check the output on your target hardware.

18. To generate the prom file in **.mcs** format do the following steps:

Right click the **Generate Programming File** option and select **Process Properties** option from the menu as shown in **Figure-40a**.The **Process Properties** dialog box opens in that select **Startup Options** form **Category,** check whether for the ***Property Name*** **FPGA Start-up Clock**, the ***Value*** is set as **CCLK** if so click **Apply** and then **Ok** (refer **Figure-40b).**

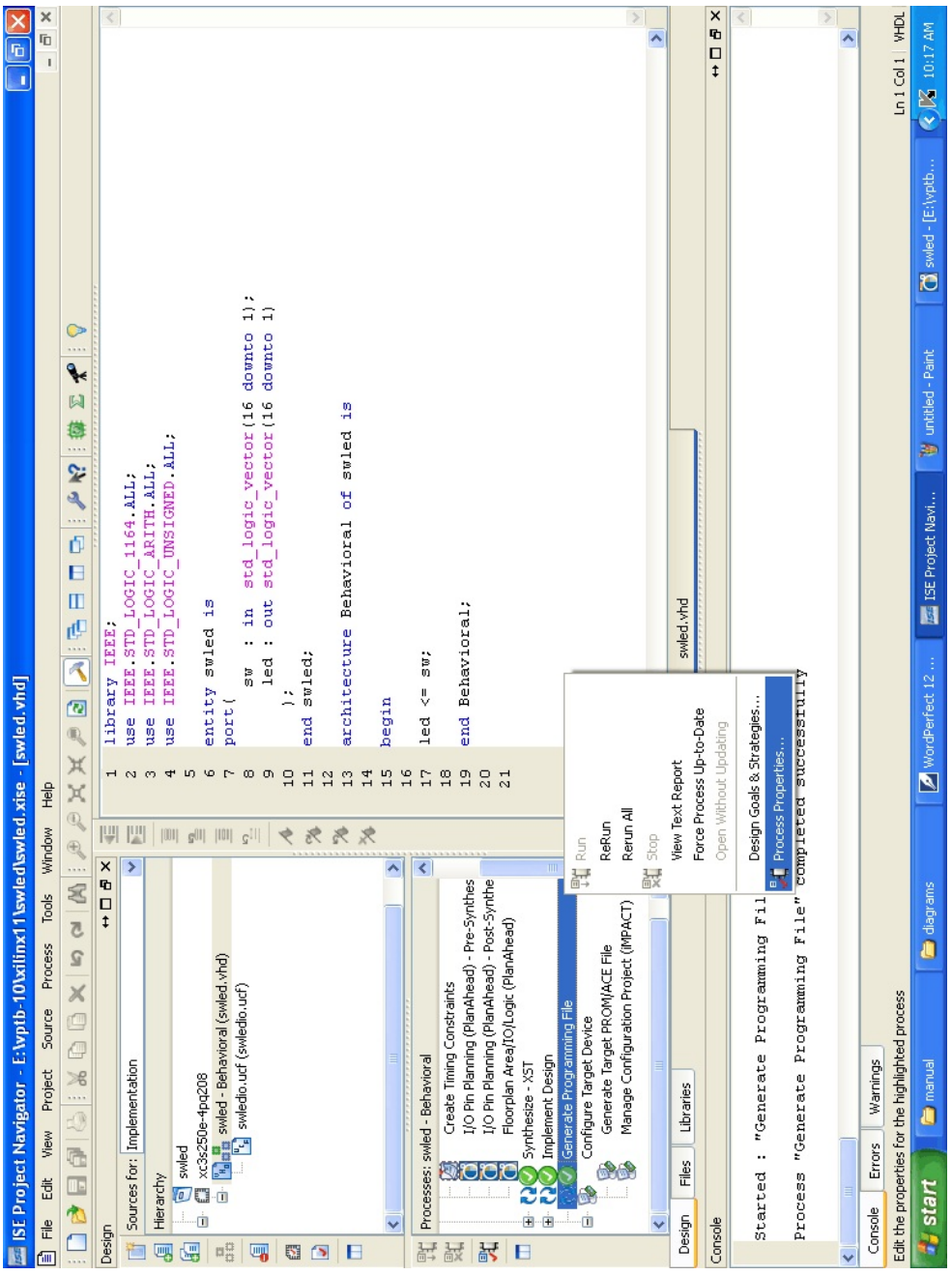


Figure- 40a

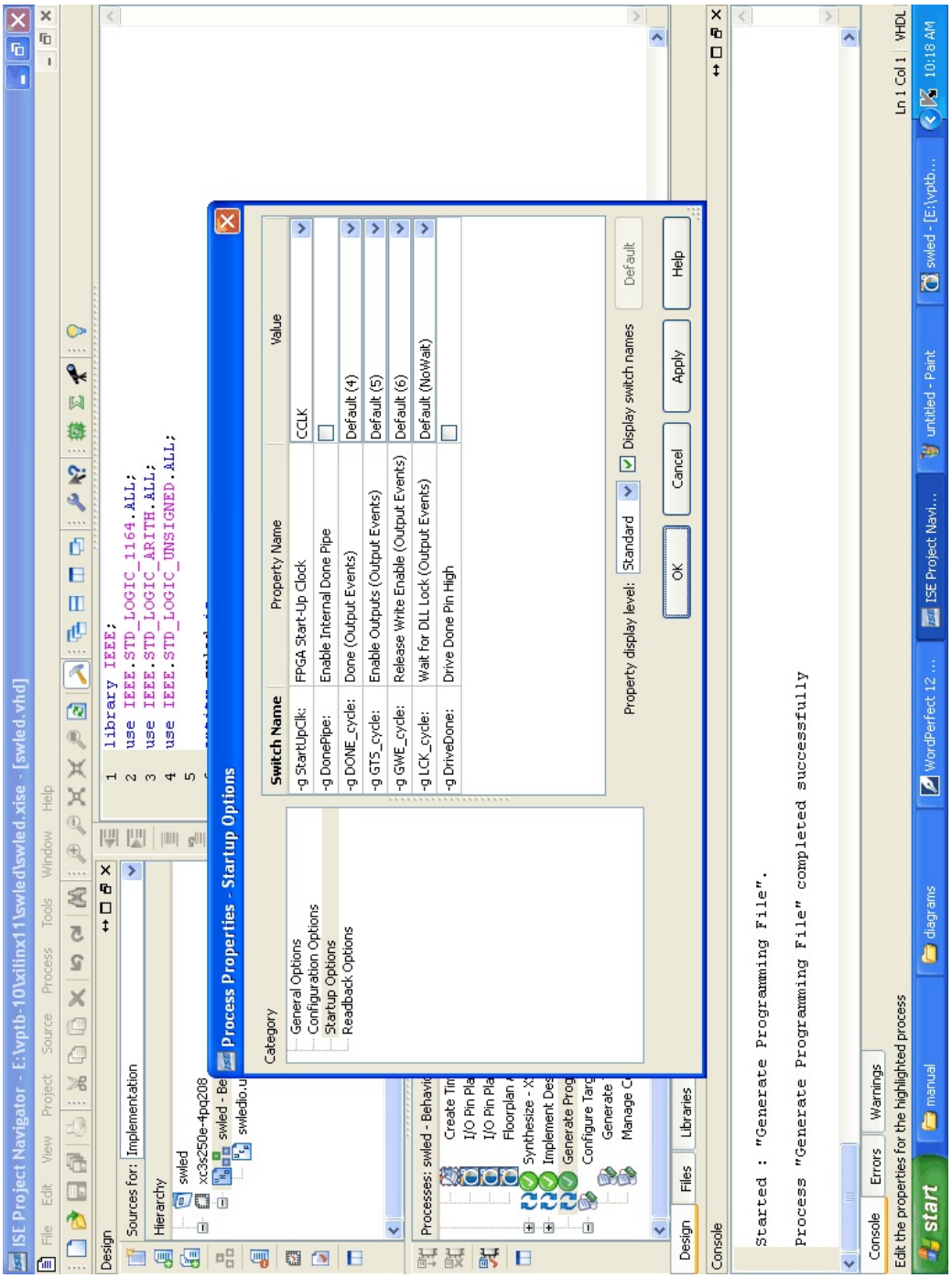

ISE Project Navigator - E:\vptb-10\xilinx11\swled\swled.xise - [swled.vhd]
Process Properties - Startup Options
Category
General Options
Configuration Options
Startup Options
Readback Options
Switch Name
Property Name
Value
-g StartUpClk:
FPGA Start-Up Clock
CCLK
-g DonePipe:
Enable Internal Done Pipe
-g DONE_cycle:
Done (Output Events)
Default (4)
-g GTS_cycle:
Enable Outputs (Output Events)
Default (5)
-g GWE_cycle:
Release Write Enable (Output Events)
Default (6)
-g LCK_cycle:
Wait for DLL Lock (Output Events)
Default (NoWait)
-g DriveDone:
Drive Done Pin High
Property display level:
Standard
Display switch names
Default
OK
Cancel
Apply
Help
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
Started : "Generate Programming File".
Process "Generate Programming File" completed successfully
Design
Files
Libraries
Console
Errors
Warnings
start
manual
diagrams
WordPerfect 12 ...
ISE Project Navi...
untitled - Paint
swled - [E:\vptb...
10:18 AM

Figure-40b

19. Now in the **Processes:** window double click the **Generate Target PROM/ACE File** as shown in **Figure -40c**. A new window named **ISE iMPACT** appears as shown in **Figure -41** will open in that double click **PROM File Formatter** option (refer **Figure-42**). Then a new window named **PROM File Formatter** appears as shown in **Figure-43**.In this new window under the **Step1. Select Storage Target** select **Xilinx Flash/PROM** and click the ➪ button as shown in **Figure-44**.For the **Step2. Add Storage Devices** select **Platform Flash,xcf02s [2M]** for **PROM Family** and **Device(bits)** and then click **Add Storage Device** (refer **Figure-45**).The PROM device **xcf02s** is added below,now click the ➪ button as shown in **Figure-46**.For the **Step 3. Enter data** enter the name of the .mcs file in the **File Name** text box and select the correct path location from the **File Location** and then click **Ok** (refer **Figure-47**).

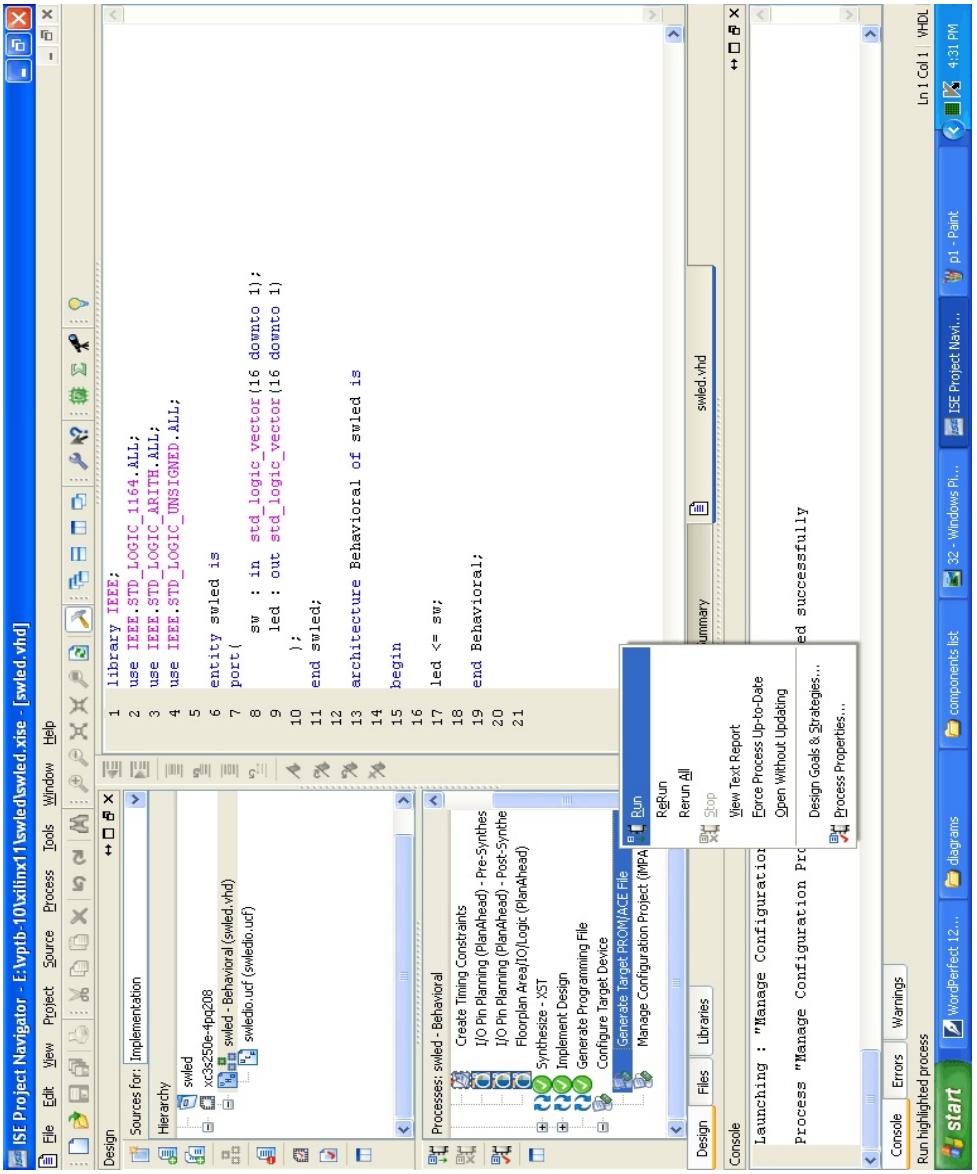


Figure-40c

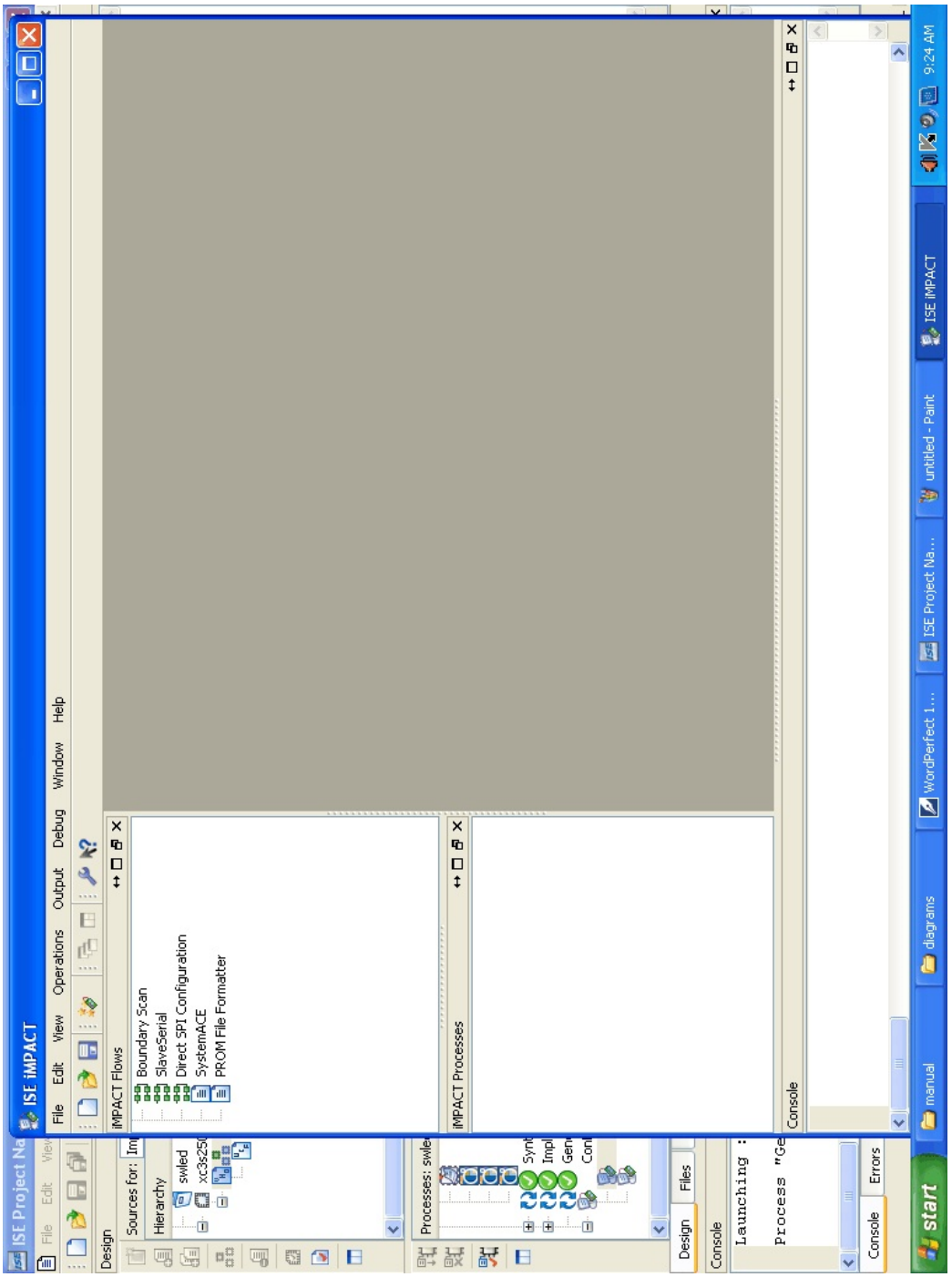

ISE iMPACT
File Edit View Operations Output Debug Window Help
IMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
IMPACT Processes
Console
Design
Sources for:
Hierarchy
Processes:
Files
Console
Errors
Launching :
Process "Ge
start
manual
diagrams
WordPerfect 1...
ISE Project Na...
untitled - Paint
ISE iMPACT
9:24 AM

Figure-41

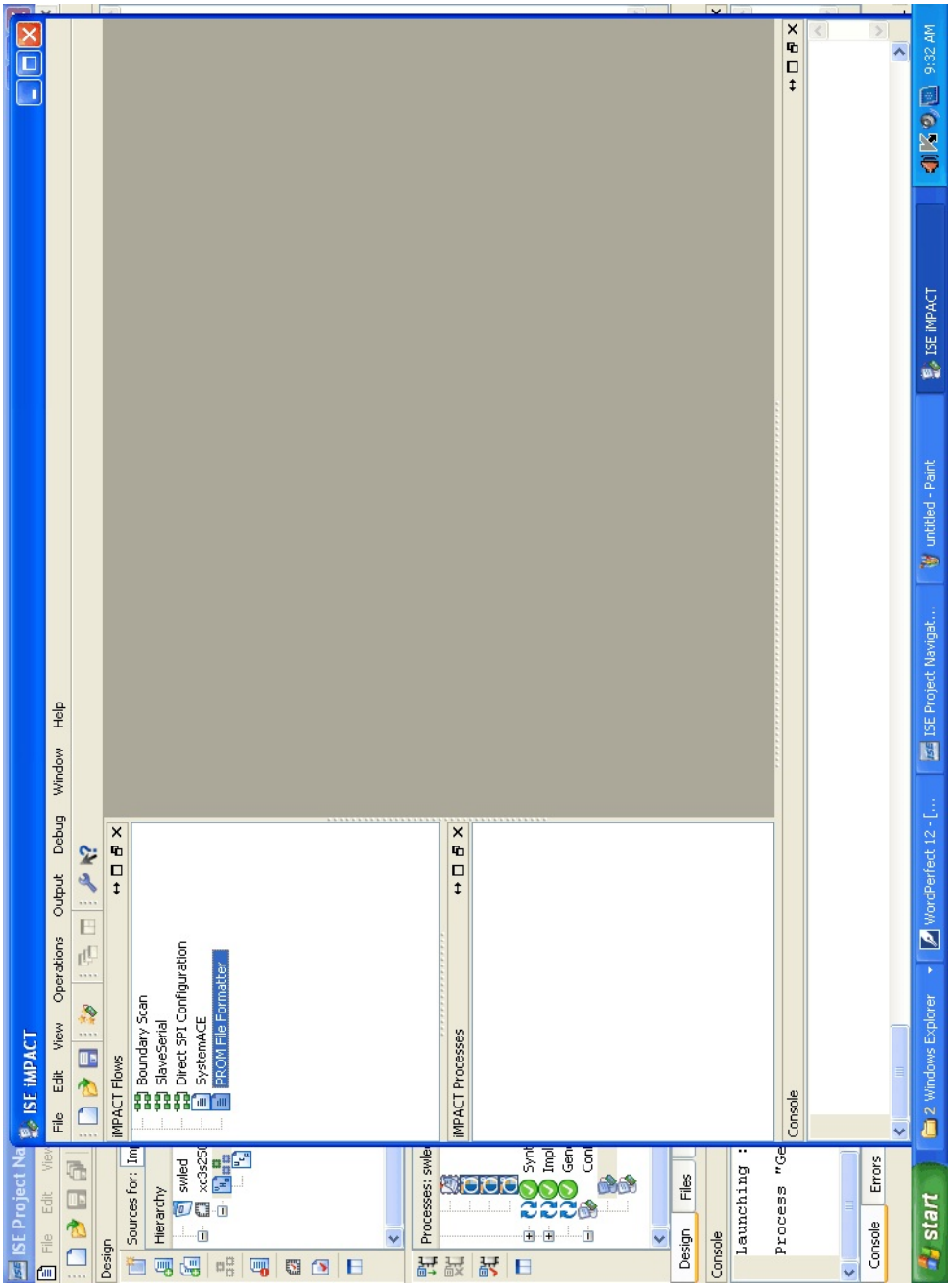

ISE iMPACT
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Console
Design
Hierarchy
Processes
Files
Errors
Launching :
Process "Ge
2 Windows Explorer
WordPerfect 12 - [...
ISE Project Navigat...
untitled - Paint
ISE iMPACT
start
9:32 AM

Figure-42

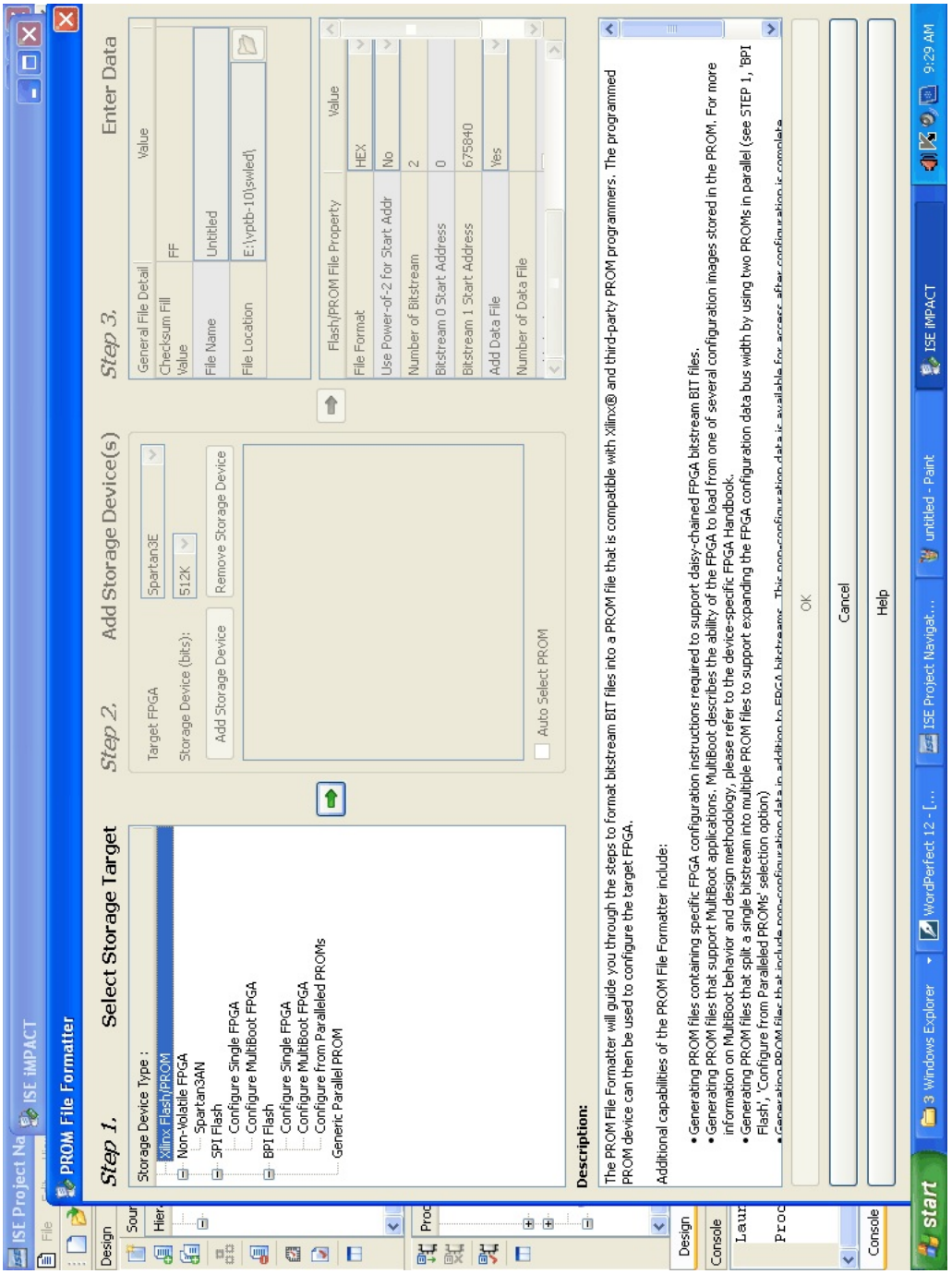

ISE iMPACT
PROM File Formatter
Step 1. Select Storage Target
Storage Device Type :
Xilinx Flash/PROM
Non-Volatile FPGA
Spartan3AN
SPI Flash
Configure Single FPGA
Configure MultiBoot FPGA
BPI Flash
Configure Single FPGA
Configure MultiBoot FPGA
Configure from Paralleled PROMs
Generic Parallel PROM
Step 2. Add Storage Device(s)
Target FPGA
Spartan3E
Storage Device (bits):
512K
Add Storage Device
Remove Storage Device
Auto Select PROM
Step 3. Enter Data
General File Detail
Value
Checksum Fill Value
FF
File Name
Untitled
File Location
E:\vptb-10\swled\
Flash/PROM File Property
Value
File Format
HEX
Use Power-of-2 for Start Addr
No
Number of Bitstream
2
Bitstream 0 Start Address
0
Bitstream 1 Start Address
675840
Add Data File
Yes
Number of Data File
Description:
The PROM File Formatter will guide you through the steps to format bitstream BIT files into a PROM file that is compatible with Xilinx® and third-party PROM programmers. The programmed PROM device can then be used to configure the target FPGA.
Additional capabilities of the PROM File Formatter include:
• Generating PROM files containing specific FPGA configuration instructions required to support daisy-chained FPGA bitstream BIT files.
• Generating PROM files that support MultiBoot applications. MultiBoot describes the ability of the FPGA to load from one of several configuration images stored in the PROM. For more information on MultiBoot behavior and design methodology, please refer to the device-specific FPGA Handbook.
• Generating PROM files that split a single bitstream into multiple PROM files to support expanding the FPGA configuration data bus width by using two PROMs in parallel (see STEP 1, 'BPI Flash', 'Configure from Paralleled PROMs' selection option)
OK
Cancel
Help
Design
Console
start
3 Windows Explorer
WordPerfect 12 - [...
ISE Project Navigat...
untitled - Paint
ISE iMPACT
9:29 AM

Figure-43

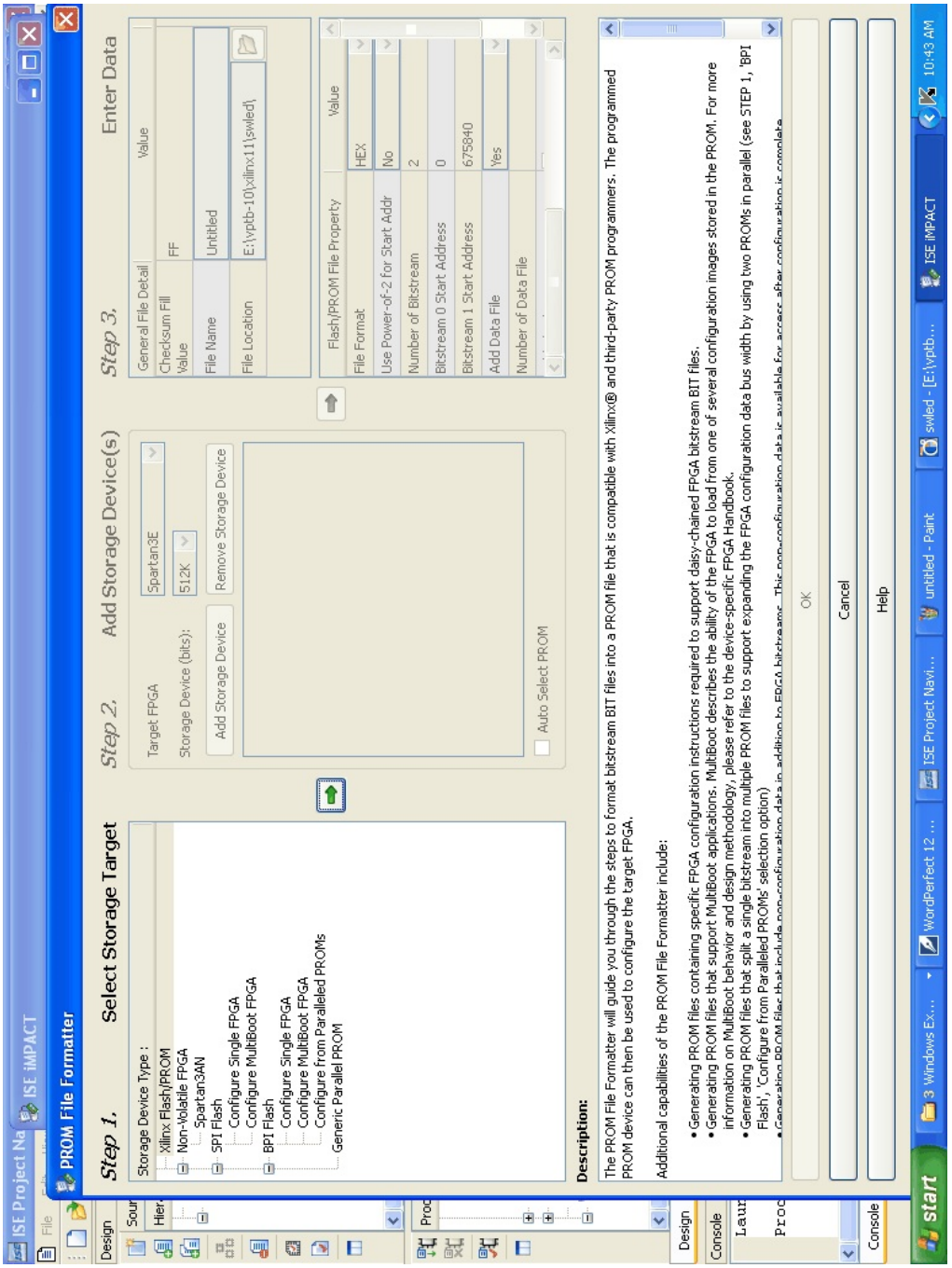

ISE iMPACT
PROM File Formatter
Step 1. Select Storage Target
Storage Device Type :
Xilinx Flash/PROM
Non-Volatile FPGA
Spartan3AN
SPI Flash
Configure Single FPGA
Configure MultiBoot FPGA
BPI Flash
Configure Single FPGA
Configure MultiBoot FPGA
Configure from Paralleled PROMs
Generic Parallel PROM
Step 2. Add Storage Device(s)
Target FPGA
Spartan3E
Storage Device (bits):
512K
Add Storage Device
Remove Storage Device
Auto Select PROM
Step 3. Enter Data
General File Detail
Value
Checksum Fill Value
FF
File Name
Untitled
File Location
E:\vptb-10\xilinx11\swled\
Flash/PROM File Property
Value
File Format
HEX
Use Power-of-2 for Start Addr
No
Number of Bitstream
2
Bitstream 0 Start Address
0
Bitstream 1 Start Address
675840
Add Data File
Yes
Number of Data File
Description:
The PROM File Formatter will guide you through the steps to format bitstream BIT files into a PROM file that is compatible with Xilinx® and third-party PROM programmers. The programmed PROM device can then be used to configure the target FPGA.
Additional capabilities of the PROM File Formatter include:
• Generating PROM files containing specific FPGA configuration instructions required to support daisy-chained FPGA bitstream BIT files.
• Generating PROM files that support MultiBoot applications. MultiBoot describes the ability of the FPGA to load from one of several configuration images stored in the PROM. For more information on MultiBoot behavior and design methodology, please refer to the device-specific FPGA Handbook.
• Generating PROM files that split a single bitstream into multiple PROM files to support expanding the FPGA configuration data bus width by using two PROMs in parallel (see STEP 1, 'BPI Flash', 'Configure from Paralleled PROMs' selection option)
OK
Cancel
Help
Design
Console
start
3 Windows Ex...
WordPerfect 12 ...
ISE Project Navi...
untitled - Paint
swled - [E:\vptb...
ISE iMPACT
10:43 AM

Figure-44

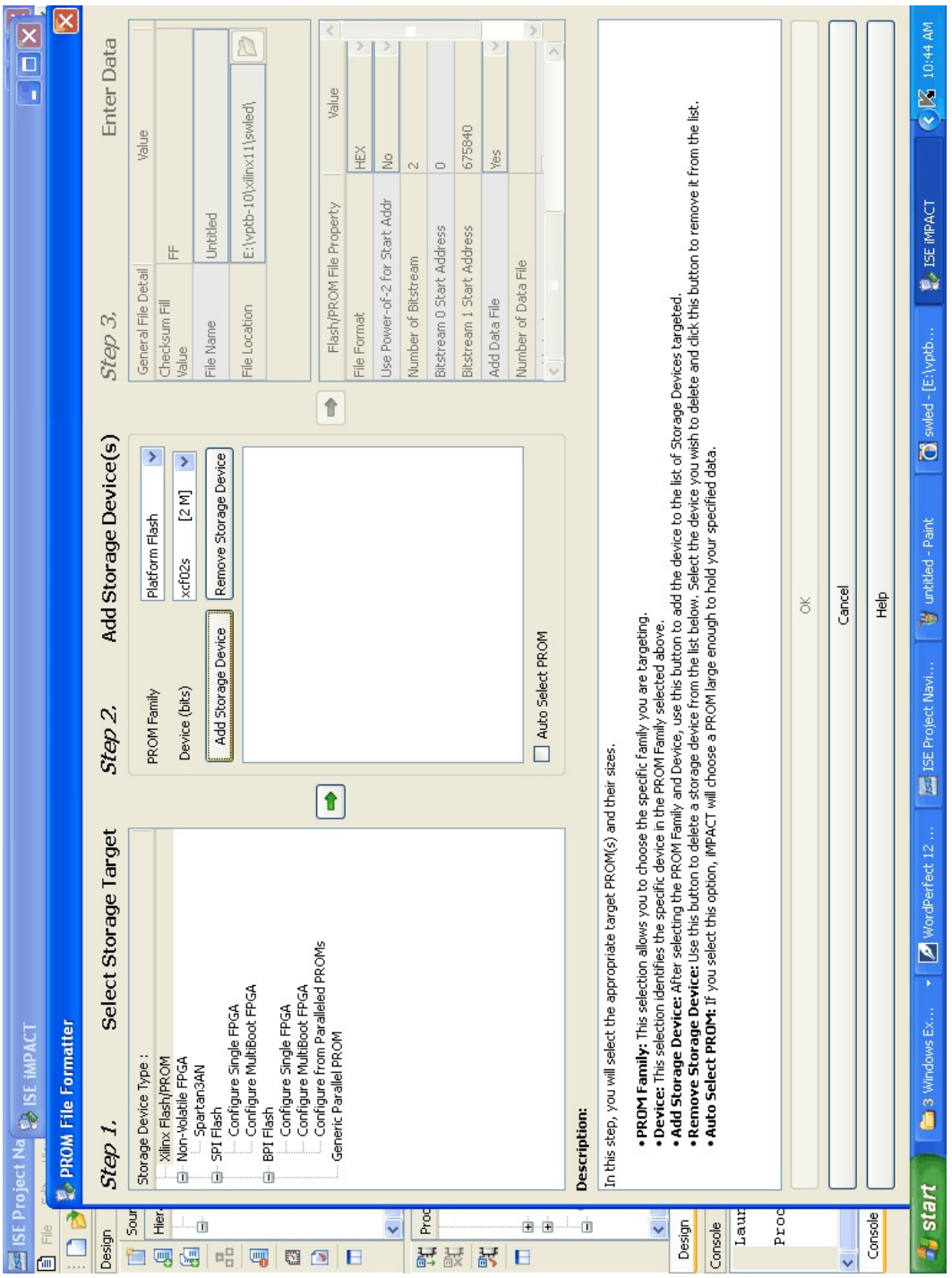

PROM File Formatter
Step 1. Select Storage Target
Storage Device Type :
Xilinx Flash/PROM
Non-Volatile FPGA
Spartan3AN
SPI Flash
Configure Single FPGA
Configure MultiBoot FPGA
BPI Flash
Configure Single FPGA
Configure MultiBoot FPGA
Configure from Paralleled PROMs
Generic Parallel PROM
Step 2. Add Storage Device(s)
PROM Family
Device (bits)
Platform Flash
xcf02s
[2 M]
Add Storage Device
Remove Storage Device
Auto Select PROM
Step 3. Enter Data
General File Detail
Value
Checksum Fill Value
FF
File Name
Untitled
File Location
E:\vptb-10\xilinx11\swled\
Flash/PROM File Property
Value
File Format
HEX
Use Power-of-2 for Start Addr
No
Number of Bitstream
2
Bitstream 0 Start Address
0
Bitstream 1 Start Address
675840
Add Data File
Yes
Number of Data File
Description:
In this step, you will select the appropriate target PROM(s) and their sizes.
• PROM Family: This selection allows you to choose the specific family you are targeting.
• Device: This selection identifies the specific device in the PROM Family selected above.
• Add Storage Device: After selecting the PROM Family and Device, use this button to add the device to the list of Storage Devices targeted.
• Remove Storage Device: Use this button to delete a storage device from the list below. Select the device you wish to delete and click this button to remove it from the list.
• Auto Select PROM: If you select this option, iMPACT will choose a PROM large enough to hold your specified data.
OK
Cancel
Help
ISE Project Na
ISE iMPACT
File
Design
Console
start
3 Windows Ex...
WordPerfect 12 ...
ISE Project Navi...
untitled - Paint
swled - [E:\vptb...
ISE iMPACT
10:44 AM

Figure-45

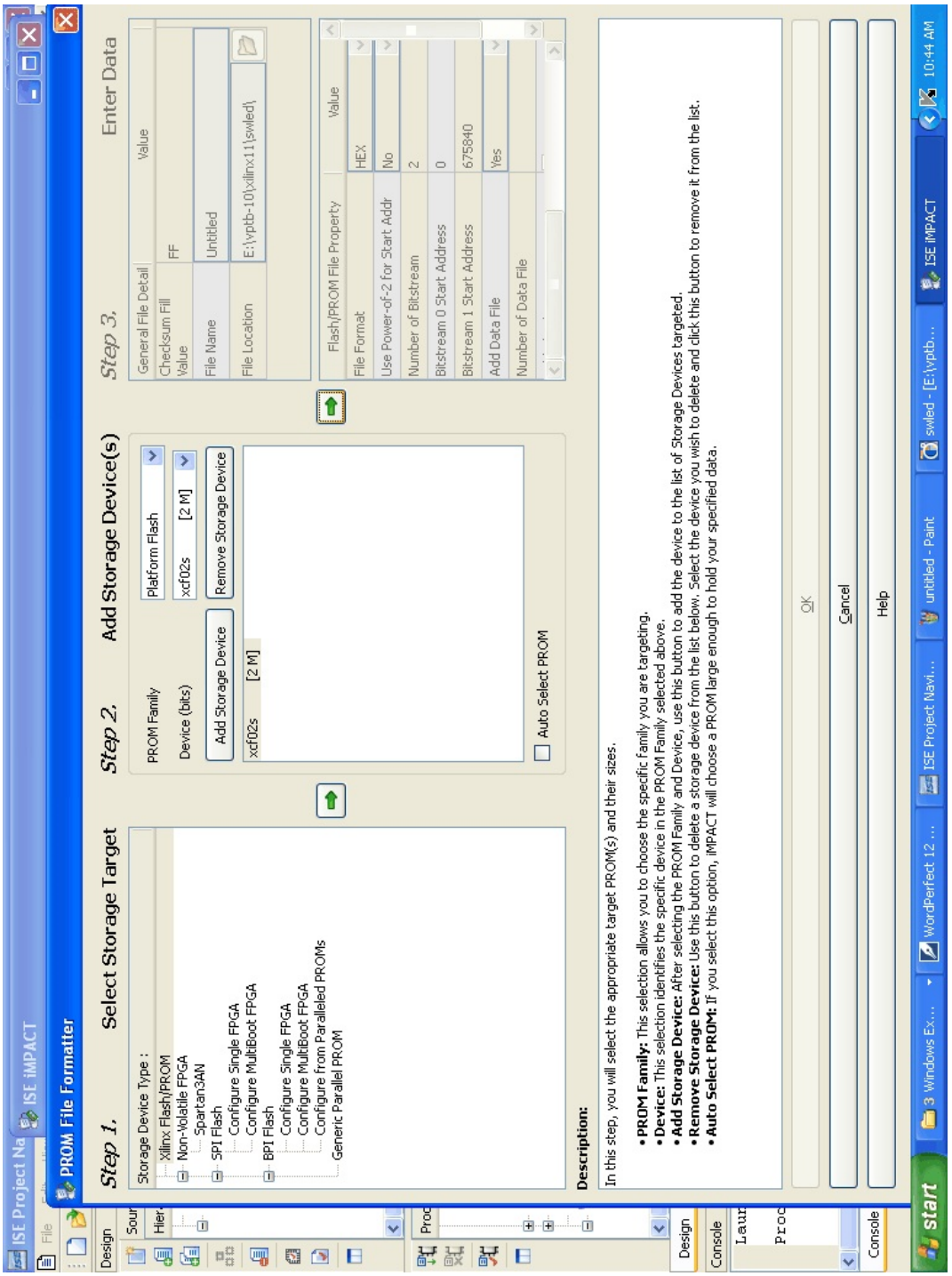

PROM File Formatter
Step 1. Select Storage Target
Storage Device Type :
Xilinx Flash/PROM
Non-Volatile FPGA
Spartan3AN
SPI Flash
Configure Single FPGA
Configure MultiBoot FPGA
BPI Flash
Configure Single FPGA
Configure MultiBoot FPGA
Configure from Paralleled PROMs
Generic Parallel PROM
Step 2. Add Storage Device(s)
PROM Family
Platform Flash
Device (bits)
xcf02s [2 M]
Add Storage Device
Remove Storage Device
xcf02s [2 M]
Auto Select PROM
Step 3. Enter Data
General File Detail
Value
Checksum Fill Value
FF
File Name
Untitled
File Location
E:\vptb-10\xilinx11\swled\
Flash/PROM File Property
Value
File Format
HEX
Use Power-of-2 for Start Addr
No
Number of Bitstream
2
Bitstream 0 Start Address
0
Bitstream 1 Start Address
675840
Add Data File
Yes
Number of Data File
Description:
In this step, you will select the appropriate target PROM(s) and their sizes.
• PROM Family: This selection allows you to choose the specific family you are targeting.
• Device: This selection identifies the specific device in the PROM Family selected above.
• Add Storage Device: After selecting the PROM Family and Device, use this button to add the device to the list of Storage Devices targeted.
• Remove Storage Device: Use this button to delete a storage device from the list below. Select the device you wish to delete and click this button to remove it from the list.
• Auto Select PROM: If you select this option, iMPACT will choose a PROM large enough to hold your specified data.
OK
Cancel
Help
ISE iMPACT
start
10:44 AM

Figure-46

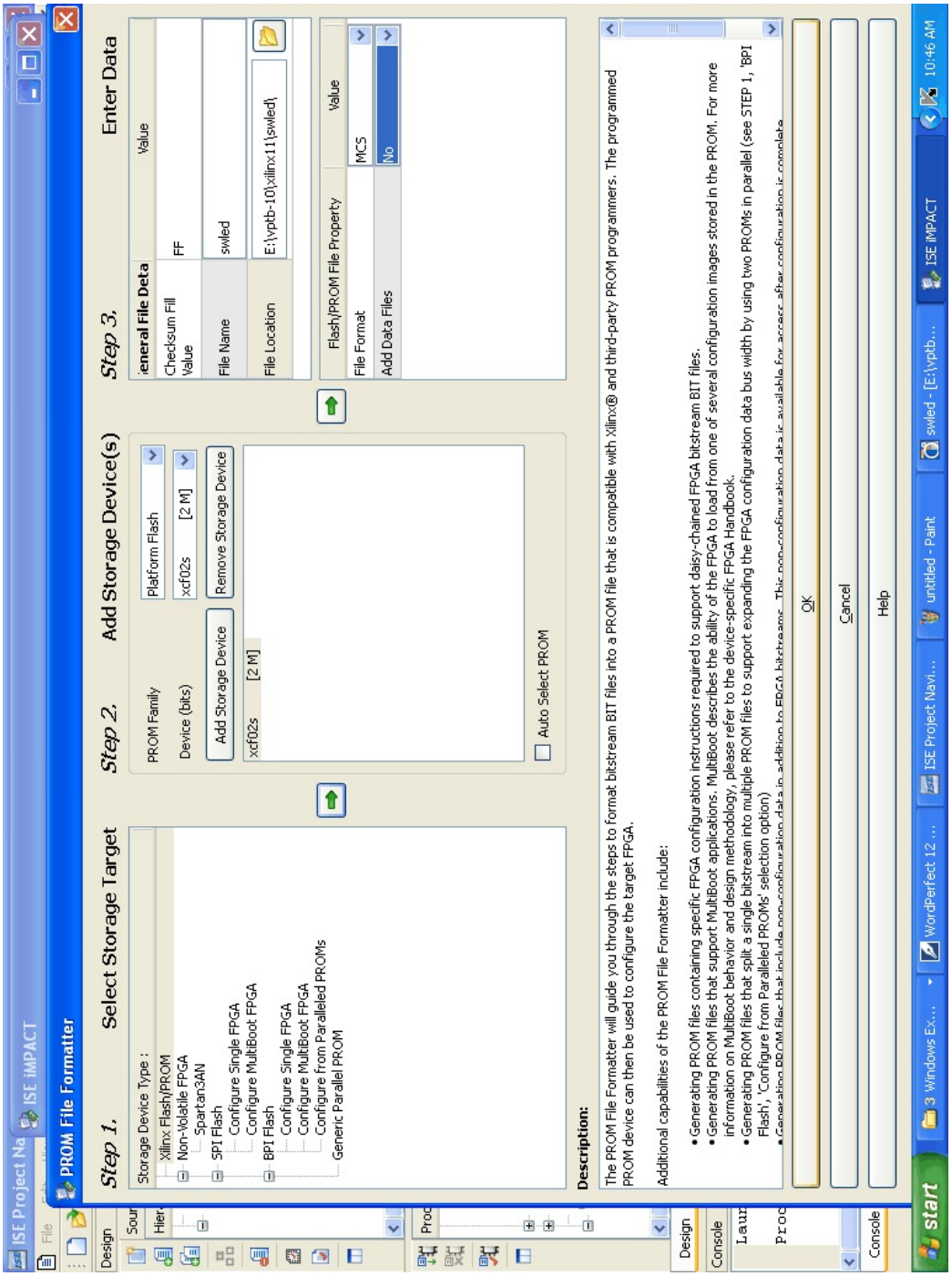

PROM File Formatter
Step 1.
Select Storage Target
Storage Device Type :
Xilinx Flash/PROM
Non-Volatile FPGA
Spartan3AN
SPI Flash
Configure Single FPGA
Configure MultiBoot FPGA
BPI Flash
Configure Single FPGA
Configure MultiBoot FPGA
Configure from Paralleled PROMs
Generic Parallel PROM
Step 2.
Add Storage Device(s)
PROM Family
Platform Flash
Device (bits)
xcf02s [2 M]
Add Storage Device
Remove Storage Device
xcf02s [2 M]
Auto Select PROM
Step 3.
Enter Data
Value
Checksum Fill Value
FF
File Name
swled
File Location
E:\vptb-10\xilinx11\swled\
Flash/PROM File Property
Value
File Format
MCS
Add Data Files
No
Description:
The PROM File Formatter will guide you through the steps to format bitstream BIT files into a PROM file that is compatible with Xilinx® and third-party PROM programmers. The programmed PROM device can then be used to configure the target FPGA.
Additional capabilities of the PROM File Formatter include:
• Generating PROM files containing specific FPGA configuration instructions required to support daisy-chained FPGA bitstream BIT files.
• Generating PROM files that support MultiBoot applications. MultiBoot describes the ability of the FPGA to load from one of several configuration images stored in the PROM. For more information on MultiBoot behavior and design methodology, please refer to the device-specific FPGA Handbook.
• Generating PROM files that split a single bitstream into multiple PROM files to support expanding the FPGA configuration data bus width by using two PROMs in parallel (see STEP 1, 'BPI Flash', 'Configure from Paralleled PROMs' selection option)
OK
Cancel
Help
ISE Project Na
ISE iMPACT
Design
Console
Start
3 Windows Ex...
WordPerfect 12 ...
ISE Project Navi...
untitled - Paint
swled - [E:\vptb...
ISE iMPACT
10:46 AM

Figure-47

20. A dialog box named **Add Device** opens in that click **Ok** as shown in **Figure-48**.Now a new window named **Add Device** which has the **.bit** file opens,in that window select the respective **swled.bit file** (for eg.) and then click **Open** (refer **Figure-49**).The **Add Device** dialog box appears in that click **No** and then **Ok** (refer **Figure-50,51**).

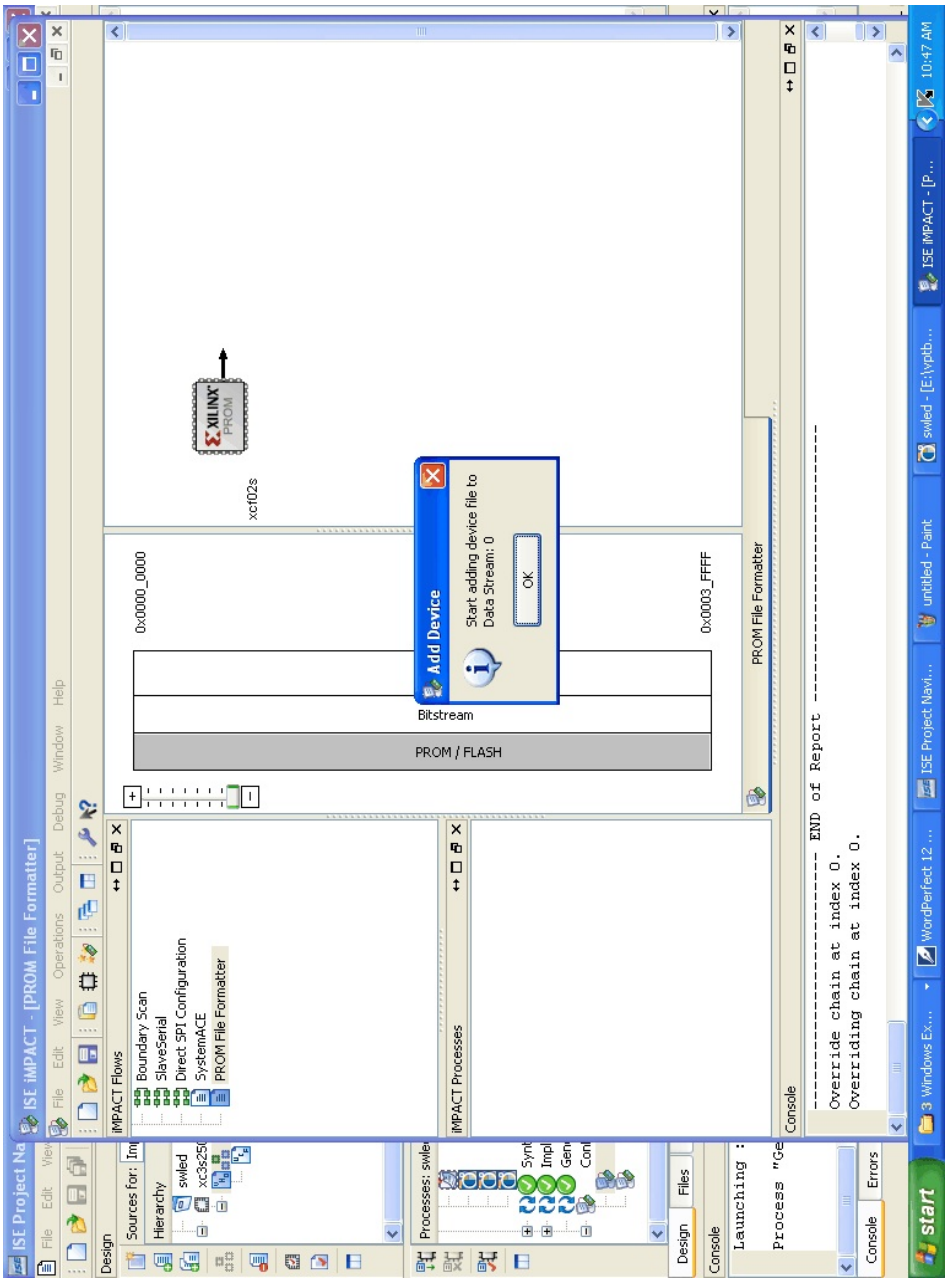


Figure-48

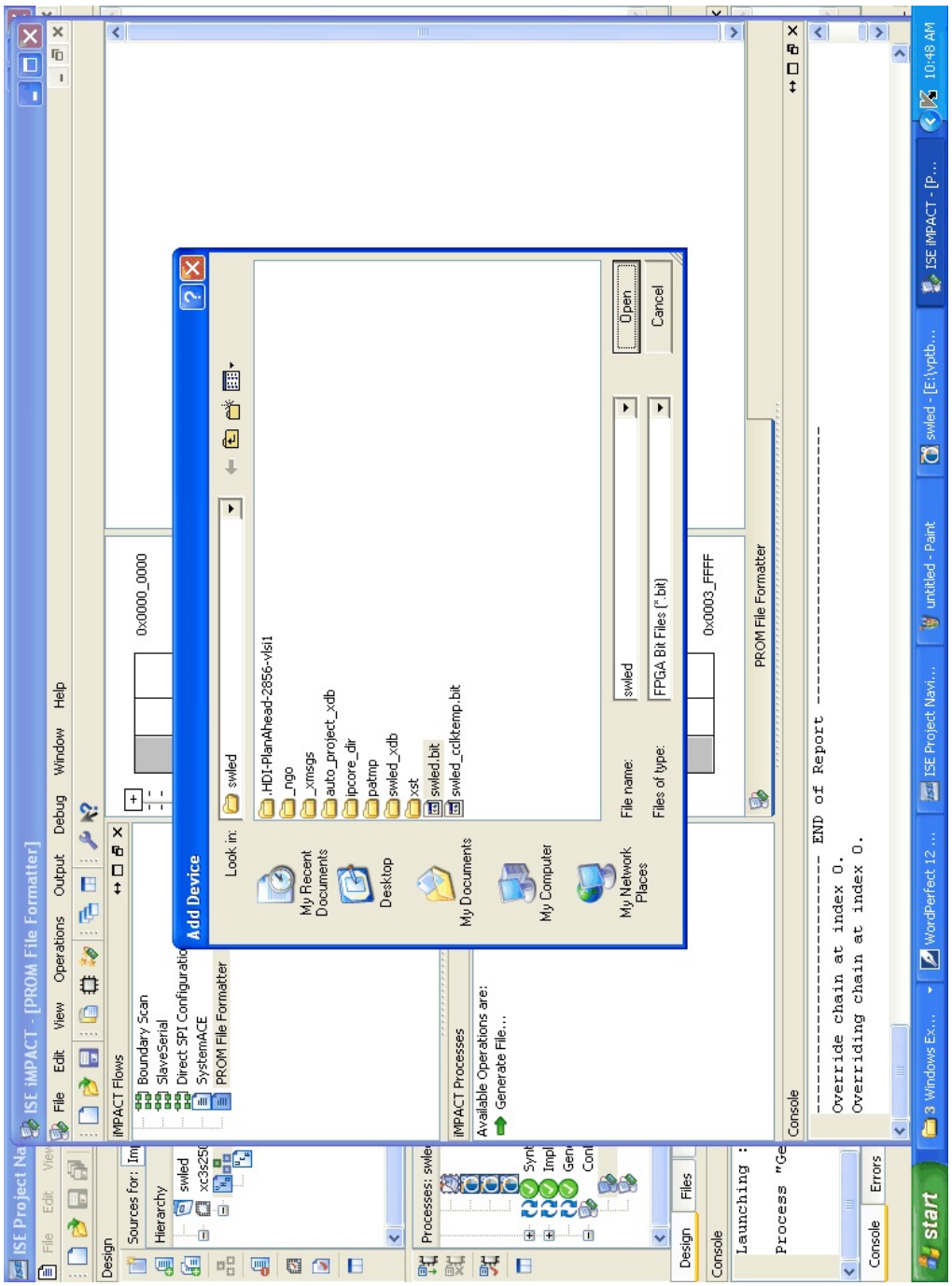

ISE iMPACT - [PROM File Formatter]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuratio
SystemACE
PROM File Formatter
Add Device
Look in:
swled
.HDI-PlanAhead-2856-vlsi1
_ngo
_xmsgs
auto_project_xdb
ipcore_dir
patmp
swled_xdb
xst
swled.bit
swled_cclktemp.bit
My Recent Documents
Desktop
My Documents
My Computer
My Network Places
File name:
swled
Files of type:
FPGA Bit Files (*.bit)
Open
Cancel
0x0000_0000
0x0003_FFFF
PROM File Formatter
iMPACT Processes
Available Operations are:
Generate File...
Console
Override chain at index 0.
Overriding chain at index 0.
END of Report
Design
Files
Hierarchy
Sources for:
Processes: swle
Launching :
Process "Ge
Console
Errors
start
3 Windows Ex...
WordPerfect 12 ...
ISE Project Navi...
untitled - Paint
swled - [E:\vptb...
ISE iMPACT - [P...
10:48 AM

Figure-49

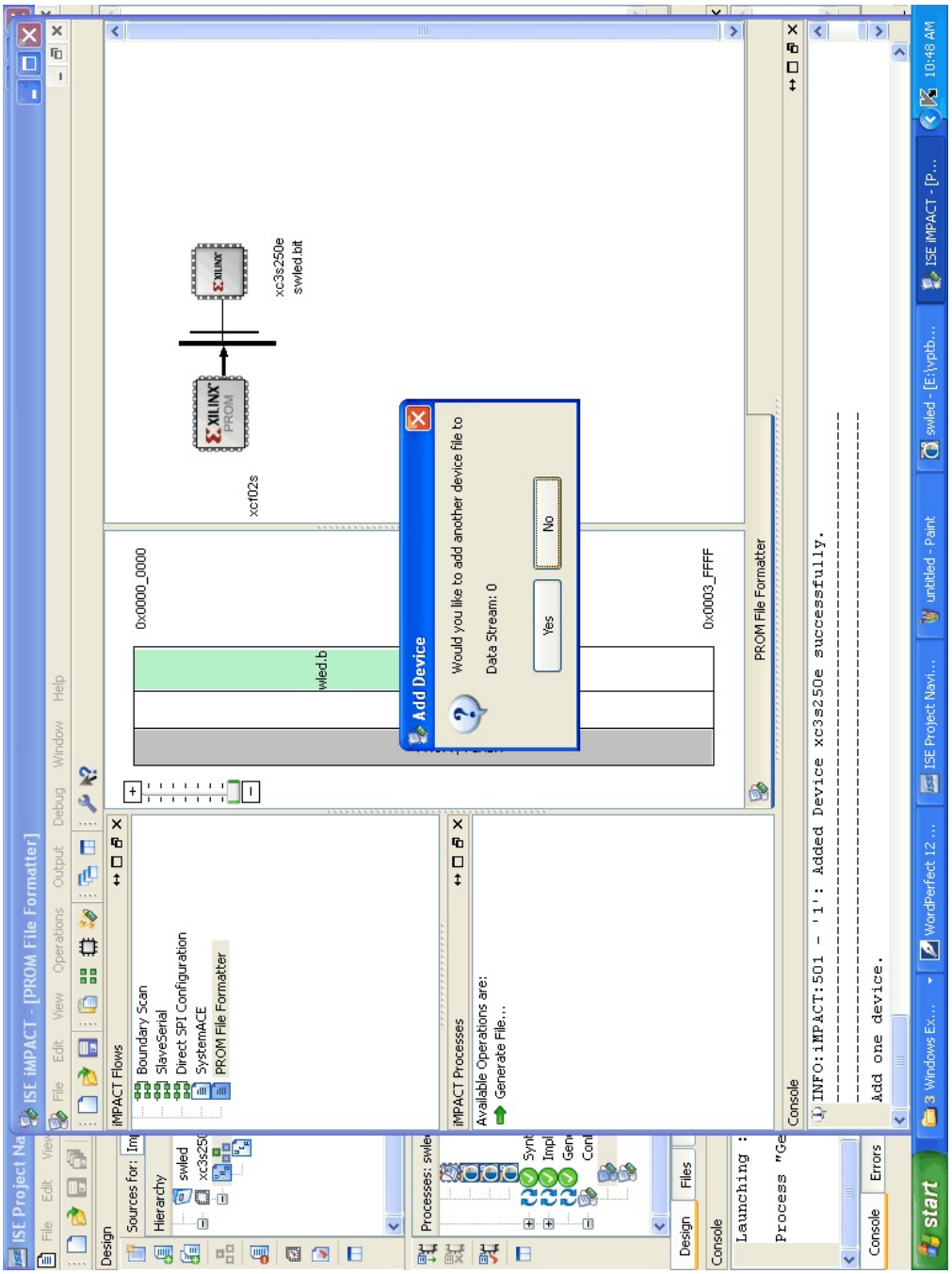

ISE iMPACT - [PROM File Formatter]
xc3s250e
swled.bit
xcf02s
0x0000_0000
0x0003_FFFF
swled.b
Add Device
Would you like to add another device file to
Data Stream: 0
Yes
No
PROM File Formatter
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Generate File...
Console
INFO:iMPACT:501 - '1': Added Device xc3s250e successfully.
Add one device.

Figure-50

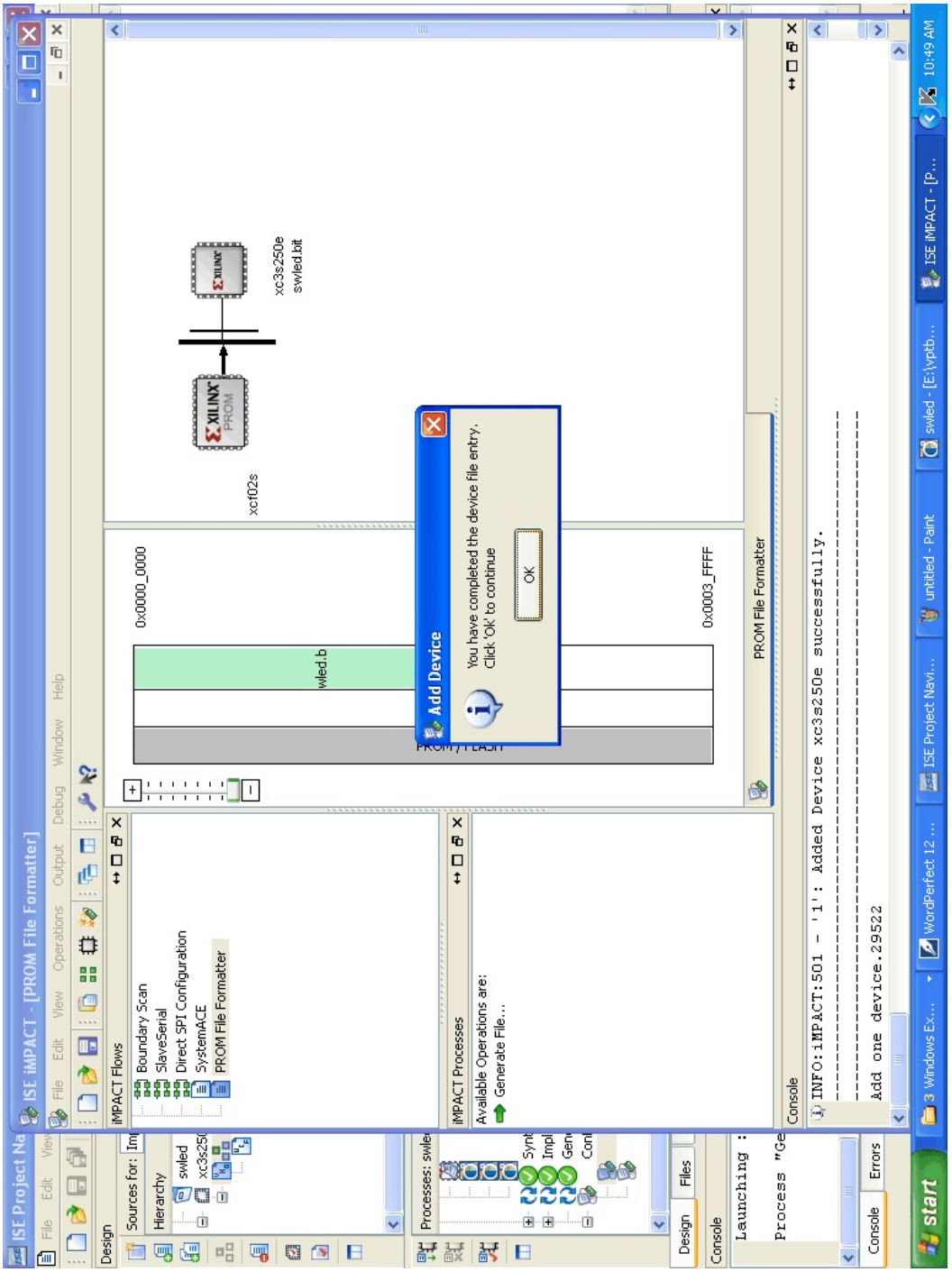

ISE iMPACT - [PROM File Formatter]
xc3s250e
swled.bit
xcf02s
Add Device
You have completed the device file entry.
Click 'Ok' to continue
OK
0x0000_0000
0x0003_FFFF
wled.b
PROM File Formatter
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Generate File...
Console
INFO:iMPACT:501 - '1': Added Device xc3s250e successfully.
Add one device.29522
Launching :
Process "Ge
Design
Files
Errors
Hierarchy
swled
Sources for:
Processes: swle
start
10:49 AM

Figure-51

21. In the **ISE iMPACT(PROM File Formatter)** window double click the **Generate File...** option under the **iMPACT Processes:** category as shown in **Figure-52**. Then **PROM file** is generated, after which the message "**Generate Succeeded**" will appear in bold that is shown in **Figure-53**.

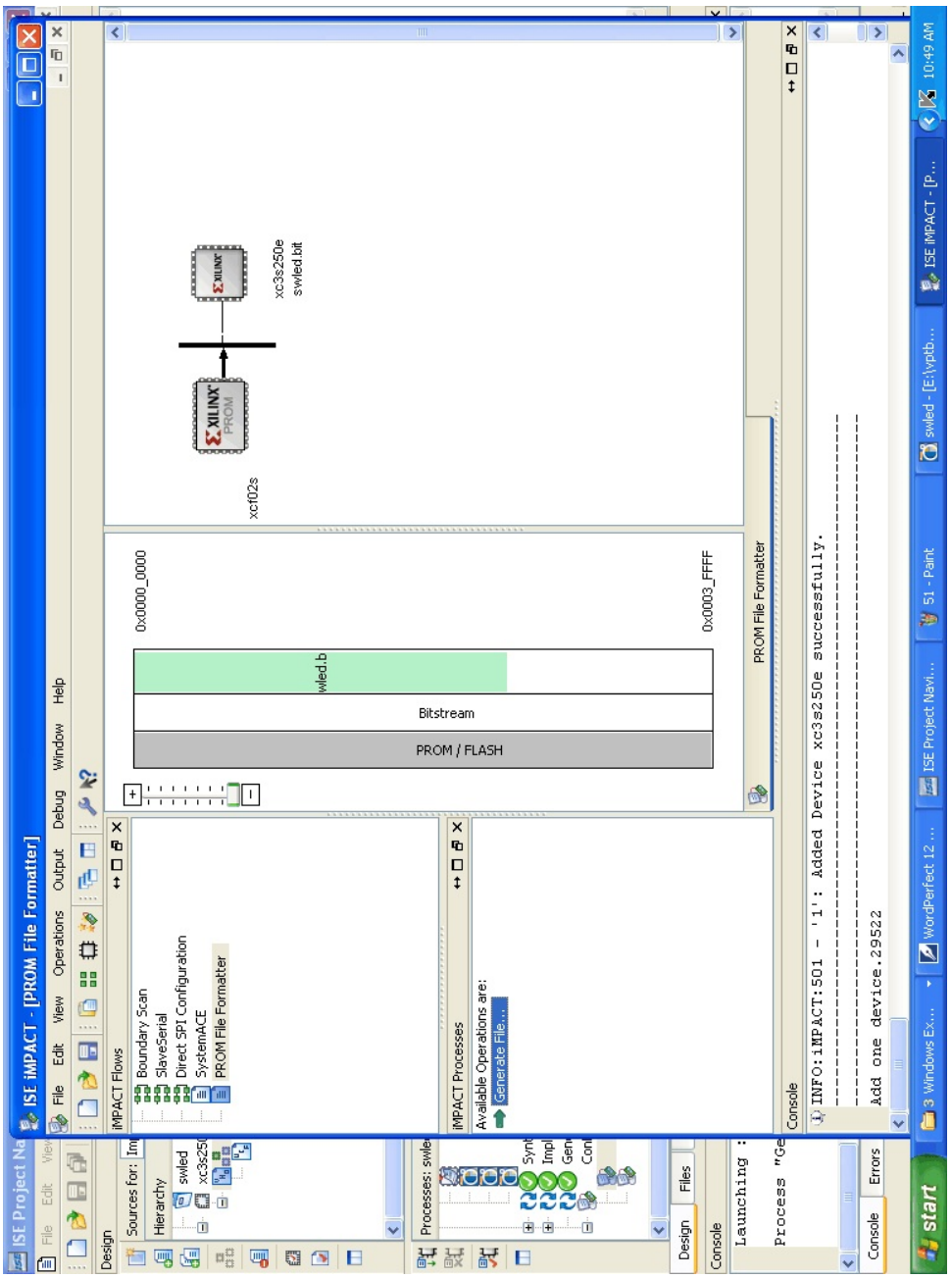


Figure-52

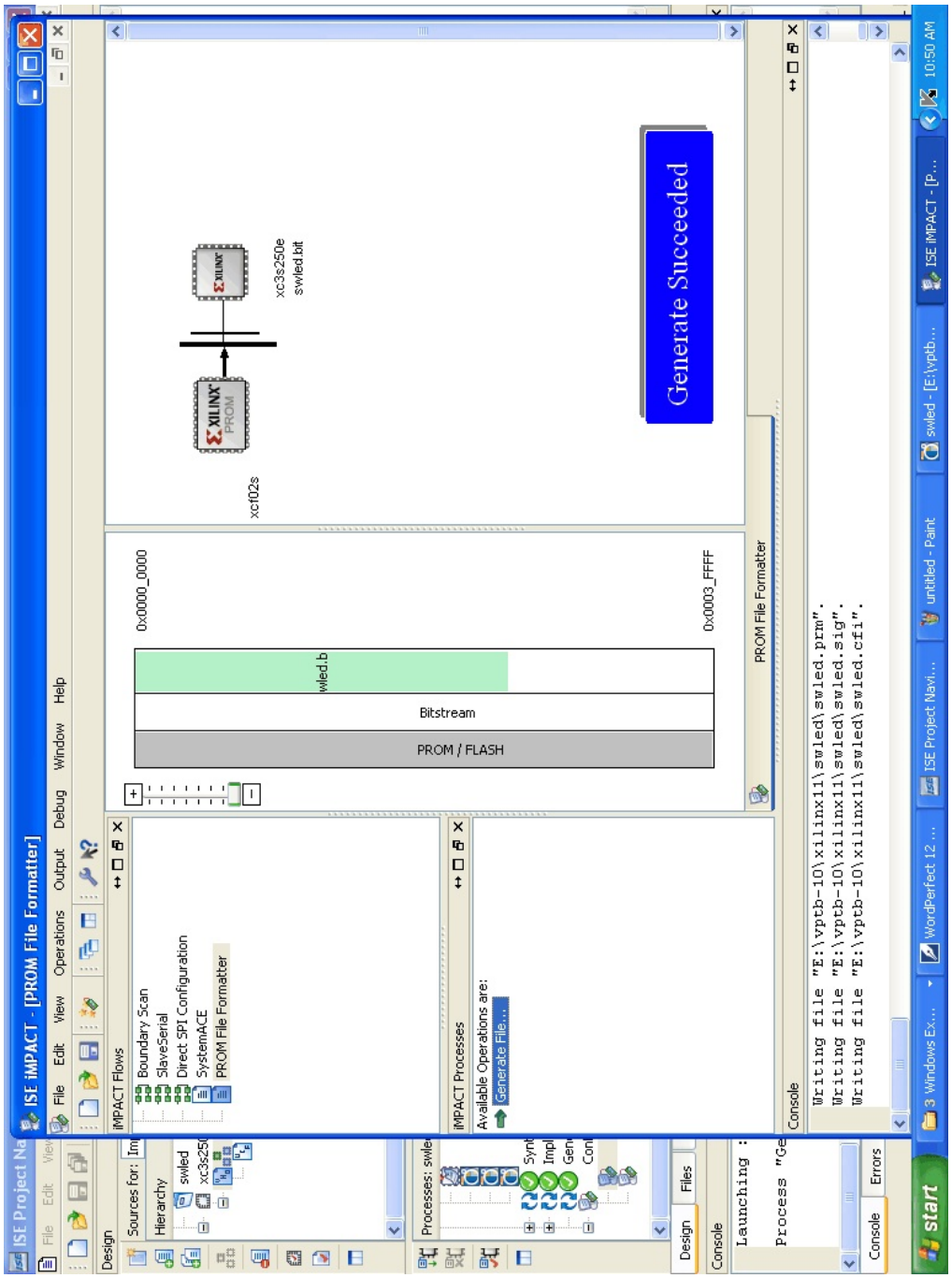


Figure-53

22. Before downloading the program,

i) Shunt of Jumper **J2** of **VPTB-10** must be in **PR** position for **PROM & FPGA** selection.

ii) Connect the downloading cable into the **JTAG(P4)** connector of the trainer kit.

23. Power **ON** the board.

24. Now click the **Boundary Scan** option in the **ISE iMPACT(PROM File Formatter)** window as shown in **Figure-54**.Then the **Boundary Scan** tab appears next to the **PROM File Formatter** tab in that tab right click and select the **Initialize Chain** option from the pop-up menu that appears (refer **Figure-55,56**).Now the **Assign New Configuration File** window appears in that select the corresponding **.bit** file (for eg.**swled.bit**) And then click **Open** (refer **Figure-57**).In the **Attach SPI or BPI PROM** dialog box click **No** as shown in **Figure-58**.

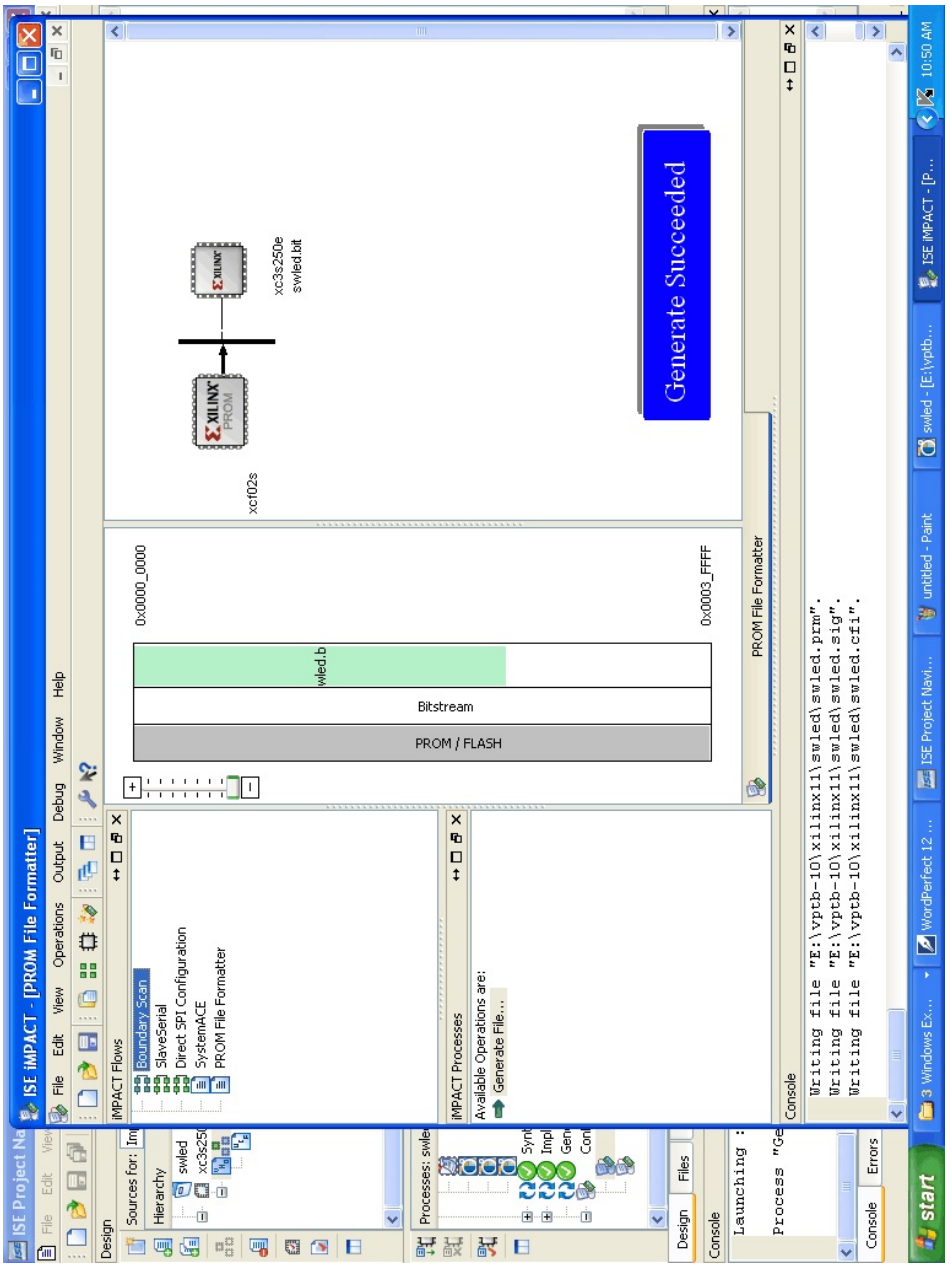


Figure-54

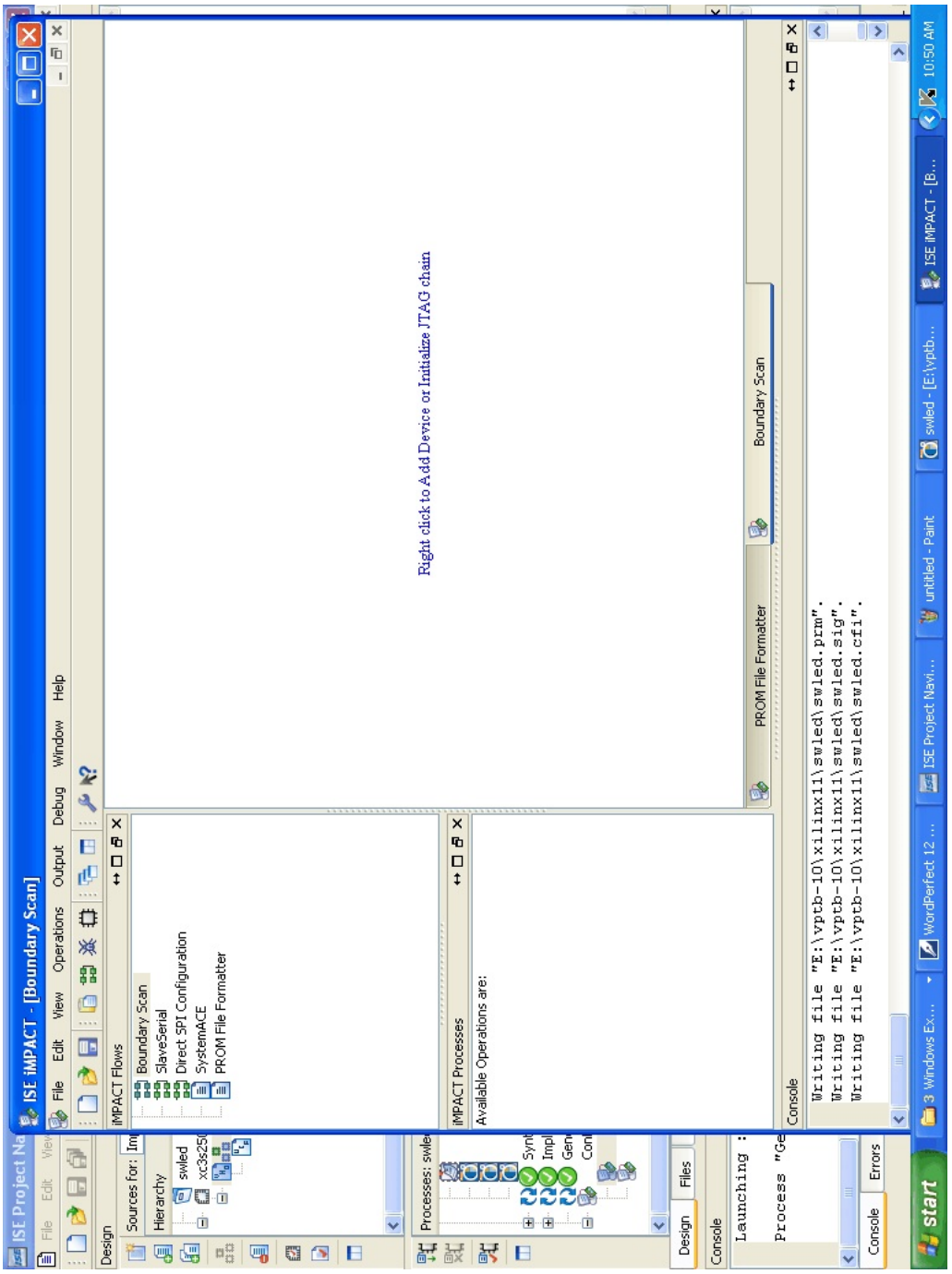

ISE iMPACT - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
Right click to Add Device or Initialize JTAG chain
Boundary Scan
PROM File Formatter
iMPACT Processes
Available Operations are:
Console
Writing file "E:\vptb-10\xilinx11\swled\swled.prm".
Writing file "E:\vptb-10\xilinx11\swled\swled.sig".
Writing file "E:\vptb-10\xilinx11\swled\swled.cfi".
Design
Hierarchy
swled
Processes: swled
Design
Files
Console
Launching :
Process "Ge
Console
Errors
start
3 Windows Ex...
WordPerfect 12 ...
ISE Project Navi...
untitled - Paint
swled - [E:\vptb...
ISE iMPACT - [B...
10:50 AM

Figure-55

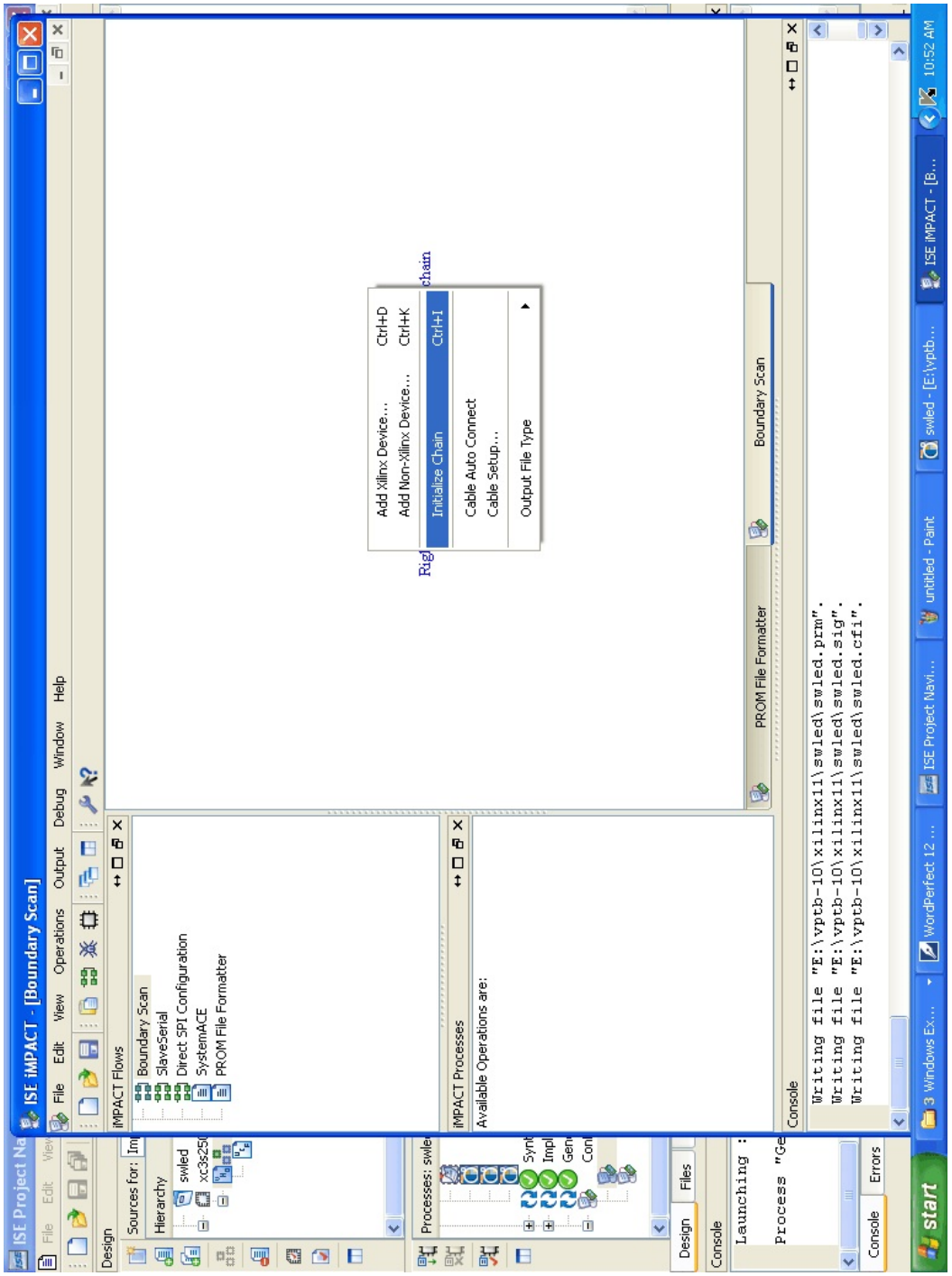

ISE iMPACT - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
Add Xilinx Device... Ctrl+D
Add Non-Xilinx Device... Ctrl+K
Initialize Chain Ctrl+I
Cable Auto Connect
Cable Setup...
Output File Type
PROM File Formatter
Boundary Scan
Writing file "E:\vptb-10\xilinx11\swled\swled.prm".
Writing file "E:\vptb-10\xilinx11\swled\swled.sig".
Writing file "E:\vptb-10\xilinx11\swled\swled.cfi".
Launching :
Process "Ge
Console
Errors
Design
Files
start

Figure-56

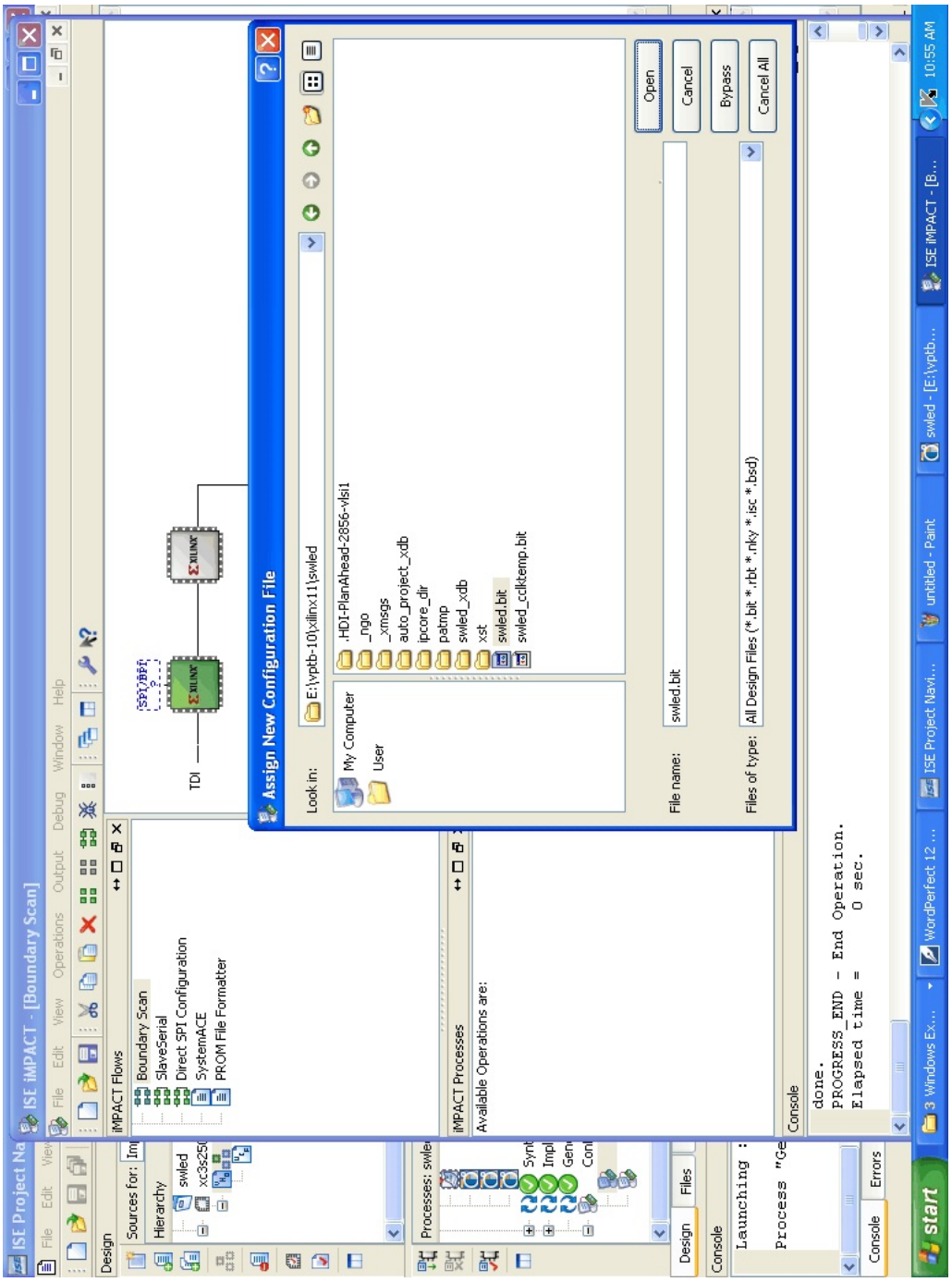

ISE iMPACT - [Boundary Scan]
Assign New Configuration File
Look in:
E:\vptb-10\xilinx11\swled
My Computer
User
.HDI-PlanAhead-2856-vlsi1
_ngo
_xmsgs
auto_project_xdb
ipcore_dir
patmp
swled_xdb
xst
swled.bit
swled_cclktemp.bit
File name:
swled.bit
Files of type:
All Design Files (*.bit *.rbt *.nky *.isc *.bsd)
Open
Cancel
Bypass
Cancel All
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Console
done.
PROGRESS END - End Operation.
Elapsed time = 0 sec.
TDI
start
10:55 AM

Figure-57

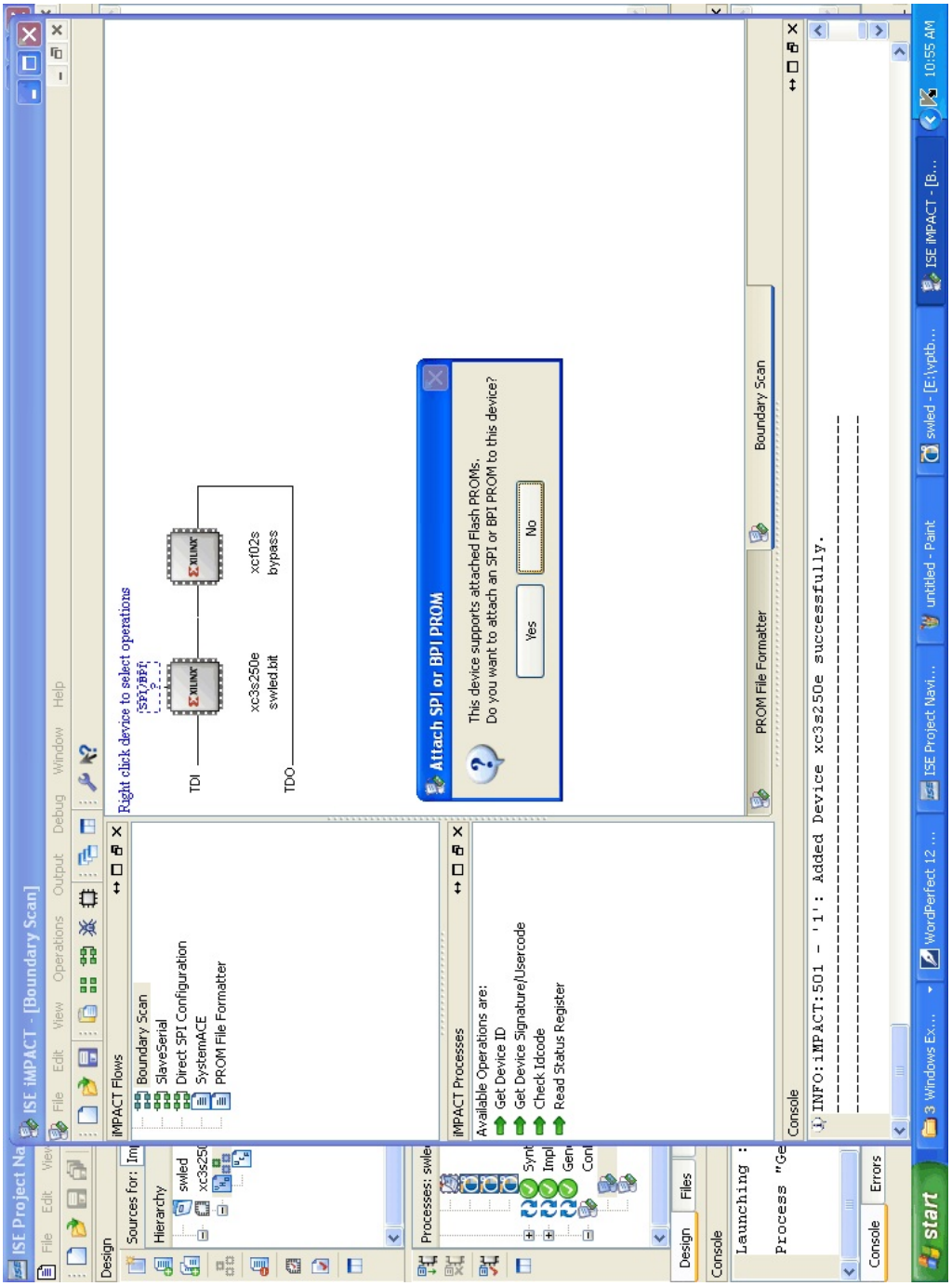

ISE iMPACT - [Boundary Scan]
Right click device to select operations
TDI
TDO
xc3s250e
swled.bit
xcf02s
bypass
Attach SPI or BPI PROM
This device supports attached Flash PROMs.
Do you want to attach an SPI or BPI PROM to this device?
Yes
No
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Get Device ID
Get Device Signature/Usercode
Check Idcode
Read Status Register
PROM File Formatter
Boundary Scan
Console
INFO:iMPACT:501 - '1': Added Device xc3s250e successfully.

Figure-58

25. The **Assign New Configuration File** window appears again,in that select the respective **.mcs** file (for eg.**swled.mcs**) and then click **Open** (refer **Figure-59**).

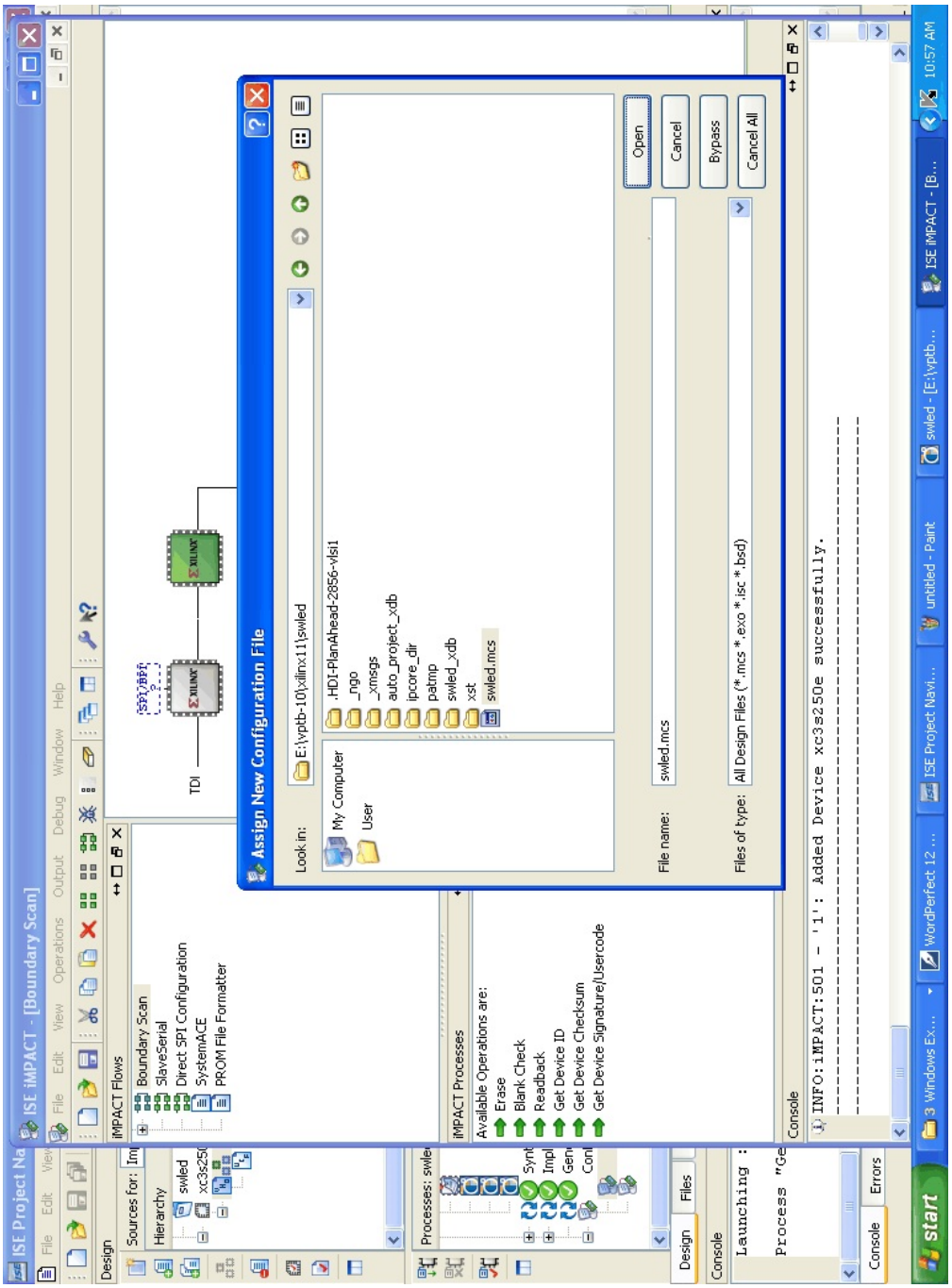


Figure-59

26. A new dialog box will appear as shown in **Figure-60**, in that dialog box for **Device1 (FPGA xc3s250e)** the **Verify** check box **must not be checked** and the **Device2 (PROM xcf2s)** and click the **Verify** and **Erase Before Programming** check boxes as shown in **Figure-60,61** and then click **Apply** and then click **Ok (**refer **Figure-62,63).**

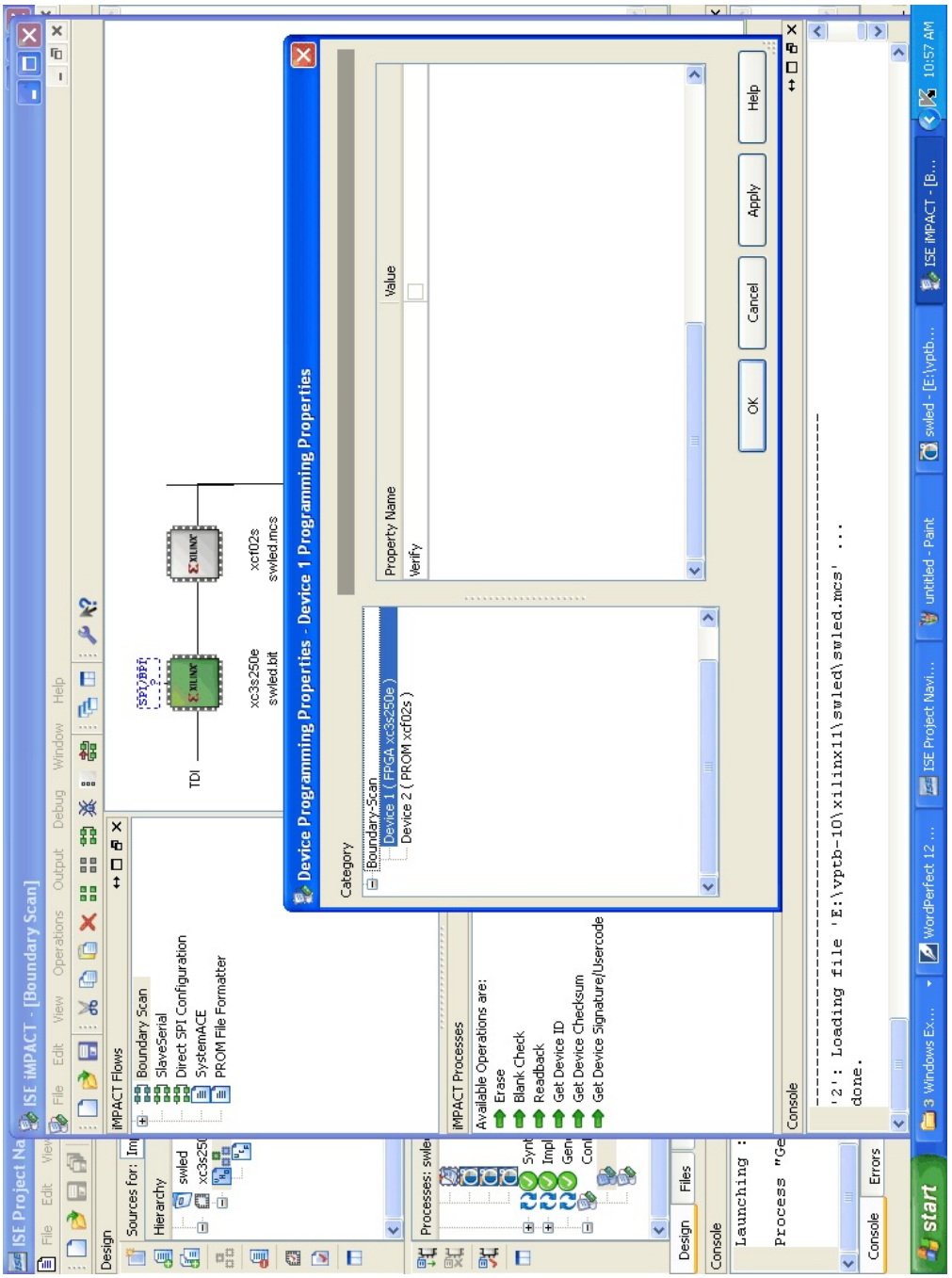


Figure-60

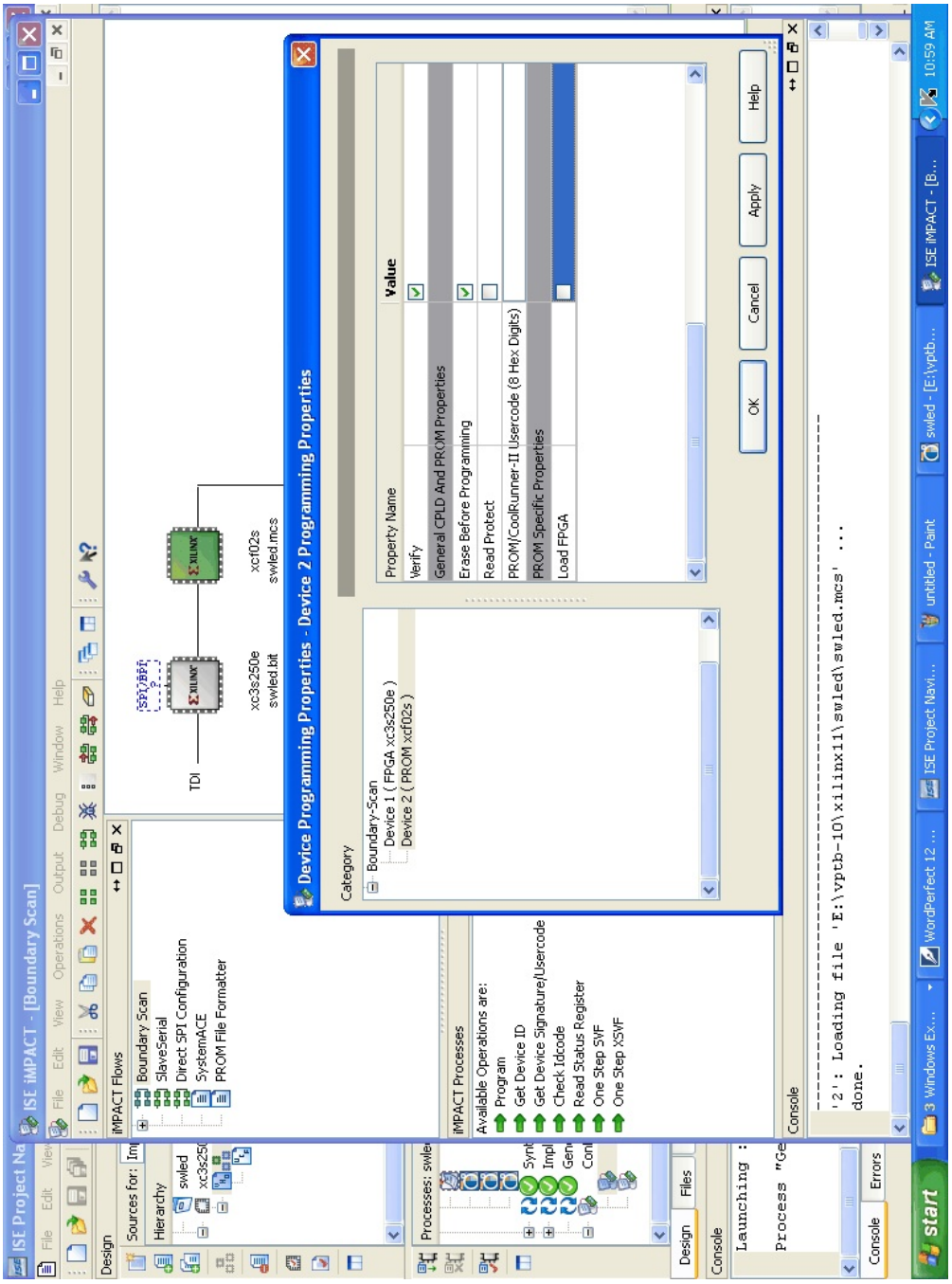

ISE iMPACT - [Boundary Scan]
Device Programming Properties - Device 2 Programming Properties
Property Name
Value
Verify
General CPLD And PROM Properties
Erase Before Programming
Read Protect
PROM/CoolRunner-II Usercode (8 Hex Digits)
PROM Specific Properties
Load FPGA
Category
Boundary-Scan
Device 1 (FPGA xc3s250e)
Device 2 (PROM xcf02s)
OK
Cancel
Apply
Help
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Program
Get Device ID
Get Device Signature/Usercode
Check Idcode
Read Status Register
One Step SVF
One Step XSVF
TDI
xc3s250e
swled.bit
xcf02s
swled.mcs
Console
'2': Loading file 'E:\vptb-10\xilinx11\swled\swled.mcs' ...
done.

Figure-61

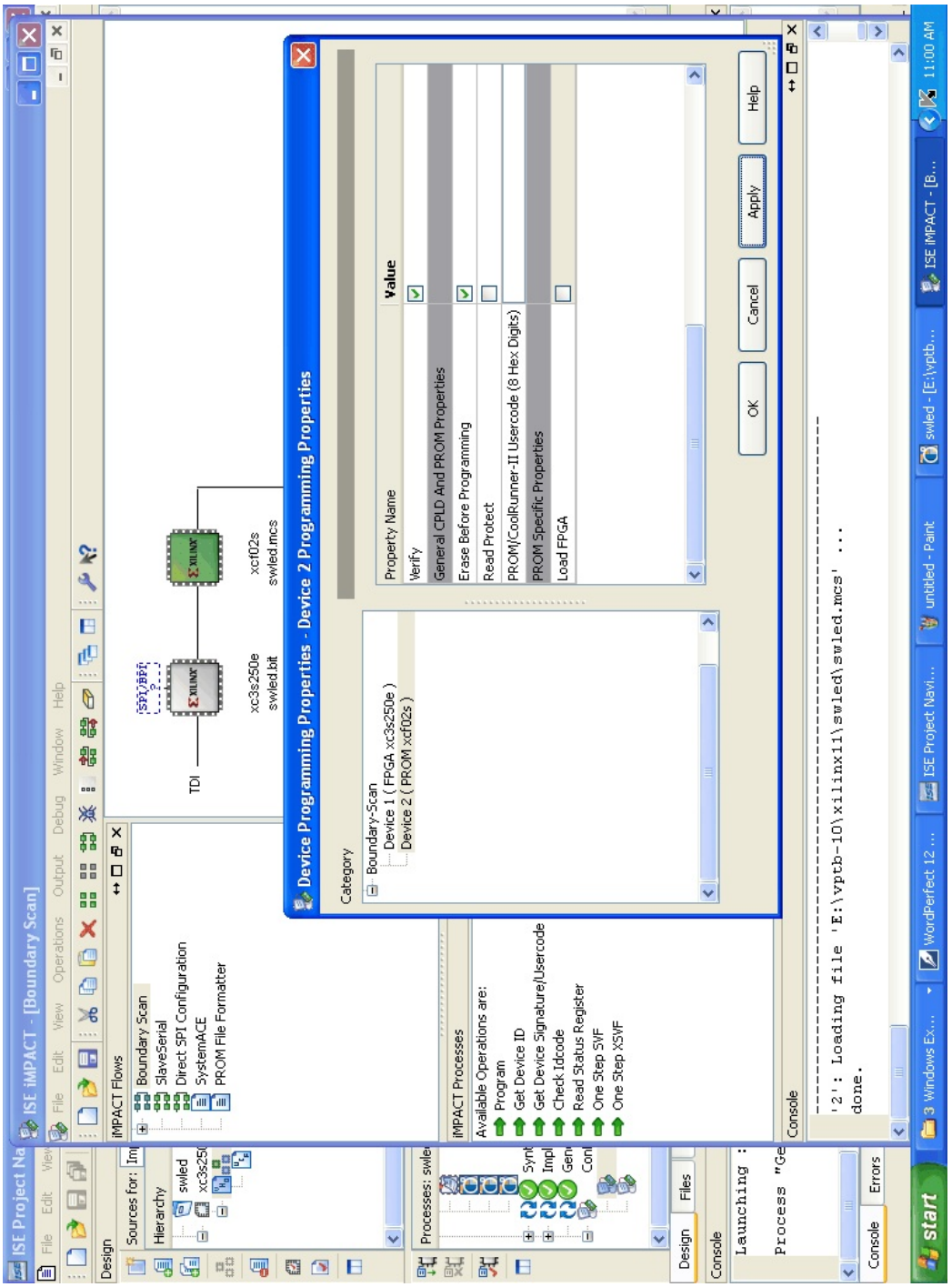

ISE iMPACT - [Boundary Scan]
Device Programming Properties - Device 2 Programming Properties
Category
Boundary-Scan
Device 1 (FPGA xc3s250e)
Device 2 (PROM xcf02s)
Property Name
Value
Verify
General CPLD And PROM Properties
Erase Before Programming
Read Protect
PROM/CoolRunner-II Usercode (8 Hex Digits)
PROM Specific Properties
Load FPGA
OK
Cancel
Apply
Help
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Program
Get Device ID
Get Device Signature/Usercode
Check Idcode
Read Status Register
One Step SVF
One Step XSVF
TDI
xc3s250e
swled.bit
xcf02s
swled.mcs
Console
'2': Loading file 'E:\vptb-10\xilinx11\swled\swled.mcs' ...
done.
Launching :
Process "Ge
Design
Files
Console
Errors
start

Figure-62

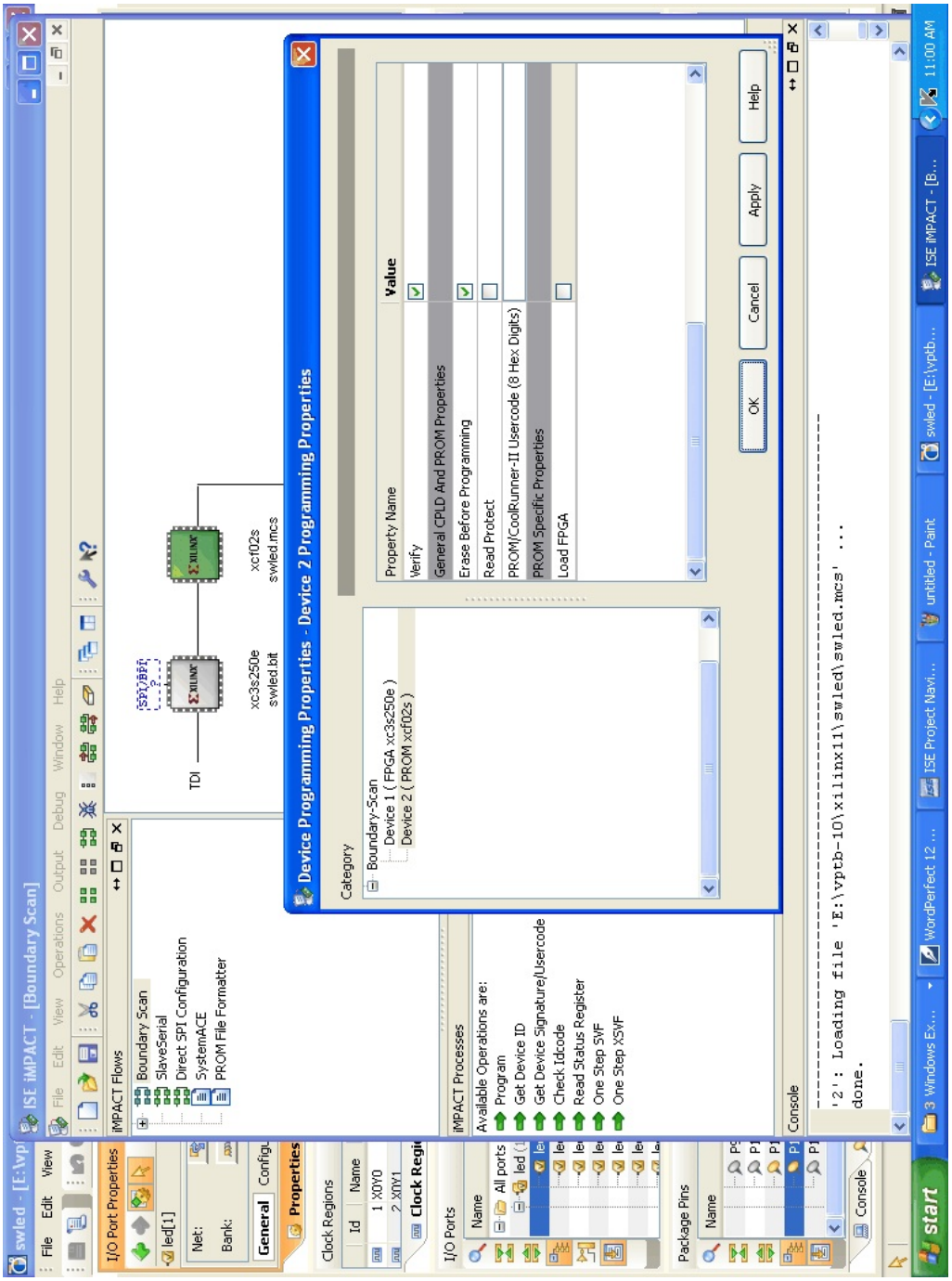

ISE iMPACT - [Boundary Scan]
Device Programming Properties - Device 2 Programming Properties
Category
Boundary-Scan
Device 1 (FPGA xc3s250e)
Device 2 (PROM xcf02s)
Property Name
Value
Verify
General CPLD And PROM Properties
Erase Before Programming
Read Protect
PROM/CoolRunner-II Usercode (8 Hex Digits)
PROM Specific Properties
Load FPGA
OK
Cancel
Apply
Help
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Program
Get Device ID
Get Device Signature/Usercode
Check Idcode
Read Status Register
One Step SVF
One Step XSVF
Console
'2': Loading file 'E:\vptb-10\xilinx11\swled\swled.mcs' ...
done.
TDI
xc3s250e
swled.bit
xcf02s
swled.mcs

Figure-63

27. Now right click the **PROM xcf02s** device in the **Boundary Scan** tab and then click **Program** as shown in **Figure-64.** The program starts downloading into your **PROM** device**.**

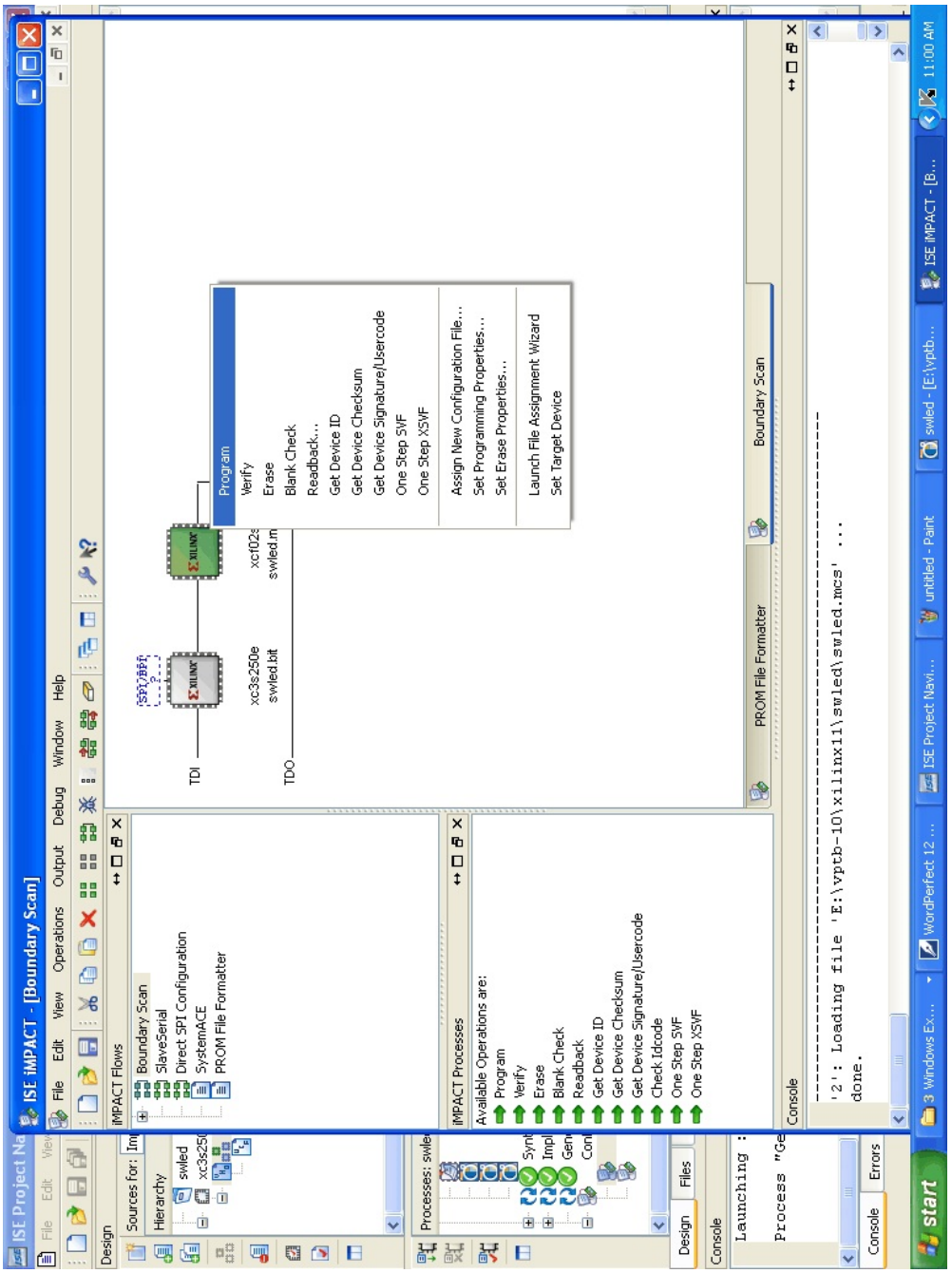


Figure-64

28. Then a window named **Configuration Operation Status** with *percentage level* shown in **Figure-65** will appear and it starts downloading your program into the hardware. After the *percentage level* reaches 100% program is downloaded successfully into the device and the message "**Program Succeeded**" will appear in bold. (refer **Figure-66,67**).

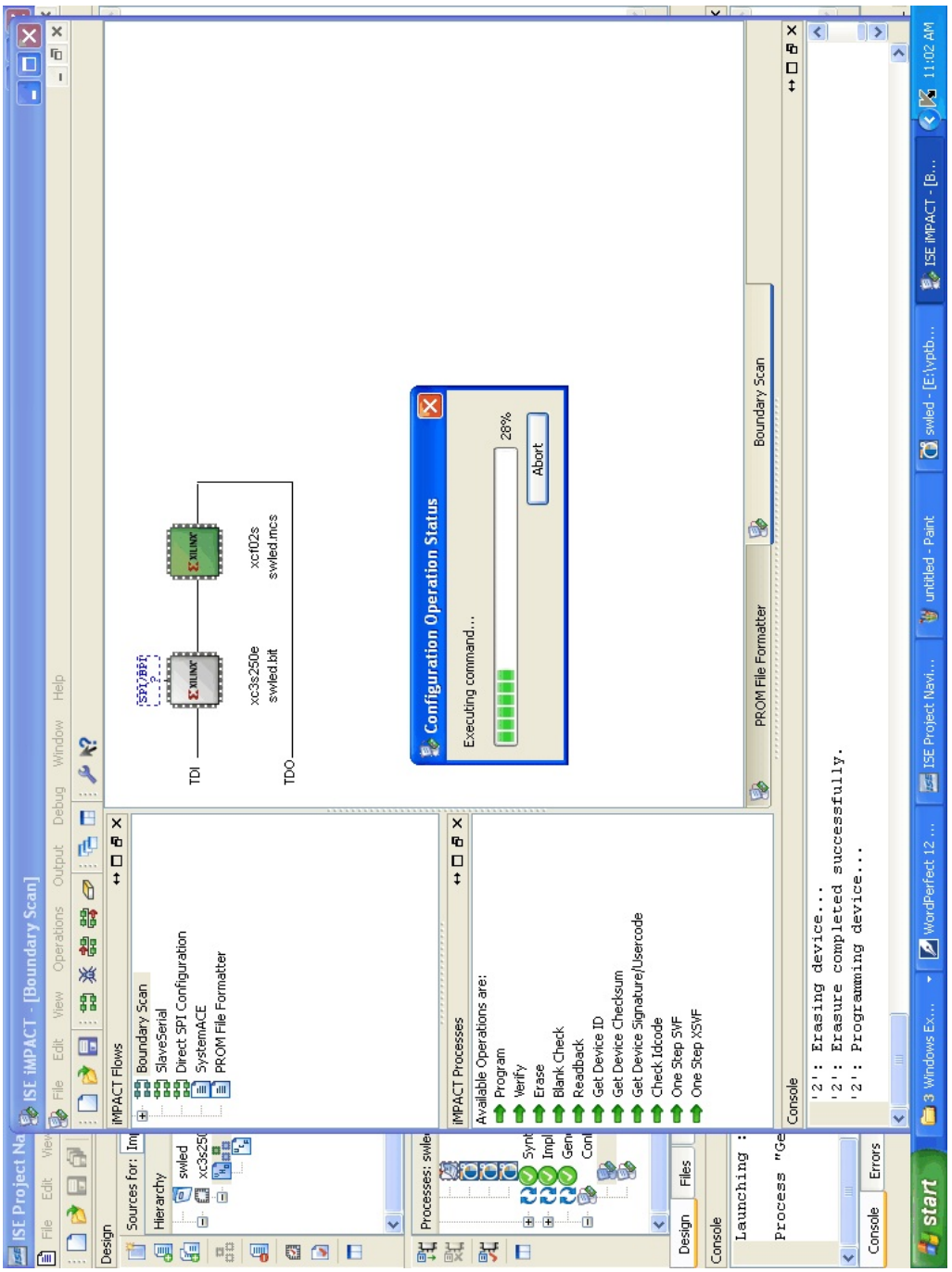


Figure-65

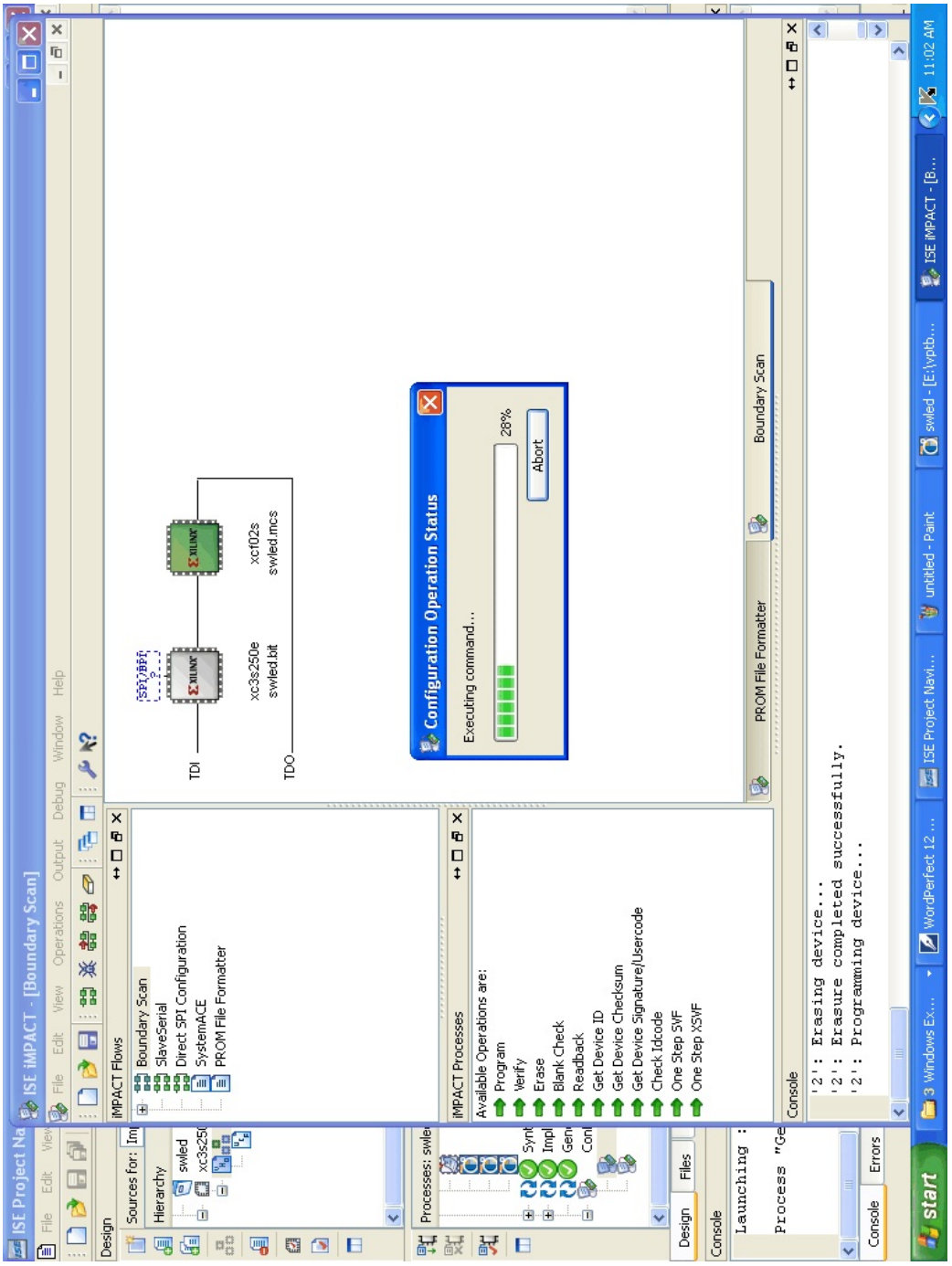

ISE iMPACT - [Boundary Scan]
Configuration Operation Status
Executing command...
28%
Abort
TDI
TDO
xc3s250e
swled.bit
xcf02s
swled.mcs
iMPACT Flows
Boundary Scan
SlaveSerial
Direct SPI Configuration
SystemACE
PROM File Formatter
iMPACT Processes
Available Operations are:
Program
Verify
Erase
Blank Check
Readback
Get Device ID
Get Device Checksum
Get Device Signature/Usercode
Check Idcode
One Step SVF
One Step XSVF
PROM File Formatter
Boundary Scan
Console
'2': Erasing device...
'2': Erasure completed successfully.
'2': Programming device...
Launching :
Process "Ge
Design
Files
Console
Errors
start
11:02 AM

Figure-66

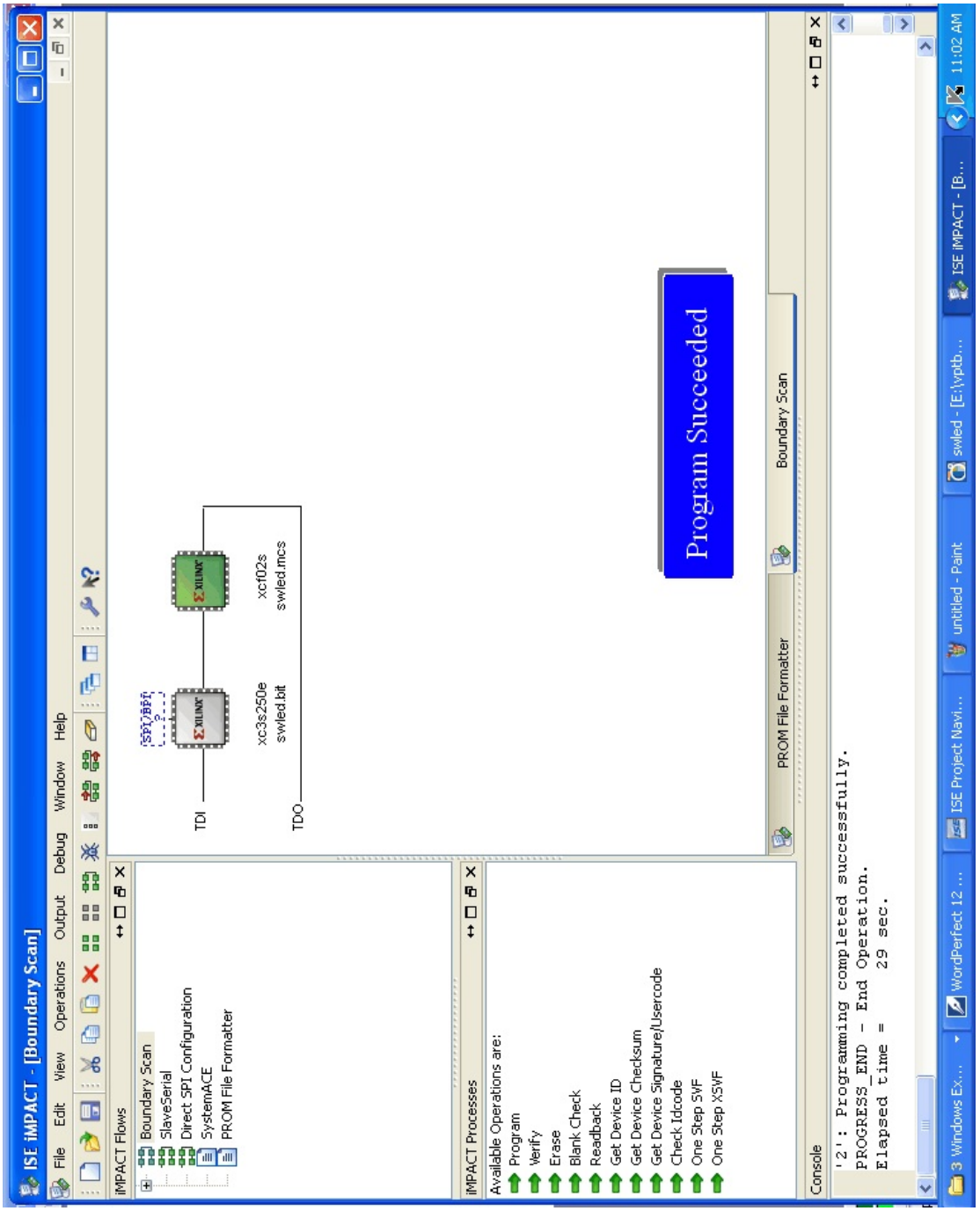


Figure-67

29. After successful configuration you can check the output on your target hardware.

CHAPTER-9
SYLLABUS PROGRAMS IN VHDL

Experiment1a) VHDL CODE FOR 4 BIT ADDITION IN BEHAVIORAL MODELLING

```
--------------------------------------------------------------------------------
--In VPTB-10 :
--9  sw : a<3> ; 10 sw : a<2> ; 11 sw : a<1> ; 12 sw : a<0> ;
--13 sw : b<3> ; 14 sw : b<2> ; 15 sw : b<1> ; 16 sw : b<0> ;
--12 led : result<4>(carry) ;
--13 led : result<3>(msb) ; 14 led : result<2> ;
--15 led : result<1>       ; 16 led : result<0>(lsb) ;
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity addition is
port(
                a       : in  std_logic_vector(3 downto 0); --1st input value--
                b       : in  std_logic_vector(3 downto 0); --2nd input value--
                result  : out std_logic_vector(4 downto 0) --output result--
   );
end addition;

architecture Behavioral of addition is
--signal declarations--
signal tmp      : std_logic_vector(4 downto 0);
begin
------------------------ 4 BIT ADDITION MODULE ------------------------------
process(a,b)
begin

tmp <= conv_std_logic_vector((conv_integer(a) + conv_integer(b)),5);      -- addition --
result(3 downto 0) <= tmp(3 downto 0);  -- sum --
result(4) <= tmp(4);                                                      -- carry --


end process;

end Behavioral;
```

User Constraints file :

```
NET "a<3>"          LOC = "P136";
NET "a<2>"          LOC = "P142";
NET "a<1>"          LOC = "P148";
NET "a<0>"          LOC = "P154";
NET "b<3>"          LOC = "P159";
NET "b<2>"          LOC = "P169";
NET "b<1>"          LOC = "P194";
NET "b<0>"          LOC = "P174";
NET "result<4>"     LOC = "P122";
NET "result<3>"     LOC = "P123";
NET "result<2>"     LOC = "P126";
NET "result<1>"     LOC = "P127";
NET "result<0>"     LOC = "P128";
```

Experiment1b) VHDL CODE FOR 4 BIT SUBTRACTION IN BEHAVIORAL MODELLING

```
--------------------------------------------------------------------------------
--In VPTB-10 :
--9  sw : a<3> ; 10 sw : a<2> ; 11 sw : a<1> ; 12 sw : a<0> ;
--13 sw : b<3> ; 14 sw : b<2> ; 15 sw : b<1> ; 16 sw : b<0> ;
--12 led : result<4>(borrow) ;
--13 led : result<3>(msb) ; 14 led : result<2> ;
--15 led : result<1>       ; 16 led : result<0>(lsb) ;
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity subtract is
port(
                a       : in  std_logic_vector(3 downto 0); --1st input value--
                b       : in  std_logic_vector(3 downto 0); --2nd input value--
                result  : out std_logic_vector(4 downto 0) --output result--
   );
end subtract;
architecture Behavioral of subtract is
--signal declarations--
signal tmp        : std_logic_vector(4 downto 0);
begin
-------------------------4 BIT SUBTRACTION MODULE-----------------------------
process(a,b)
```

```
begin

tmp <= conv_std_logic_vector((conv_integer(a) - conv_integer(b)),5);       -- subtraction --
result(3 downto 0) <= tmp(3 downto 0);  -- difference --
result(4) <= tmp(4);                    -- borrow --

end process;
end Behavioral;
```

User Constraints file :

```
NET "a<3>"          LOC = "P136";
NET "a<2>"          LOC = "P142";
NET "a<1>"          LOC = "P148";
NET "a<0>"          LOC = "P154";
NET "b<3>"          LOC = "P159";
NET "b<2>"          LOC = "P169";
NET "b<1>"          LOC = "P194";
NET "b<0>"          LOC = "P174";
NET "result<4>"     LOC = "P122";
NET "result<3>"     LOC = "P123";
NET "result<2>"     LOC = "P126";
NET "result<1>"     LOC = "P127";
NET "result<0>"     LOC = "P128";
```

Experiment1c) VHDL CODE FOR 4 BIT MULTIPLICATION IN BEHAVIORAL MODELING

```
--------------------------------------------------------------------------------
--In VPTB-10 :
--9  sw : a<3> ; 10 sw : a<2> ; 11 sw : a<1> ; 12 sw : a<0> ;
--13 sw : b<3> ; 14 sw : b<2> ; 15 sw : b<1> ; 16 sw : b<0> ;
--9  led : result<7>(msb) ; 10 led : result<6> ;
--11 led : result<5>      ; 12 led : result<4> ;
--13 led : result<3>      ; 14 led : result<2> ;
--15 led : result<1>      ; 16 led : result<0>(lsb) ;
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity multiply is
port(
            a       : in  std_logic_vector(3 downto 0); --1st input value--
            b       : in  std_logic_vector(3 downto 0); --2nd input value--
```

```
                result   : out std_logic_vector(7 downto 0) --output result--
    );
end multiply;

architecture Behavioral of multiply is
begin
-----------------------4 BIT MULTIPLICATION MODULE-----------------------------
process(a,b)
begin

result(7 downto 0) <= a * b;

end process;
end Behavioral;
```

User Constraints file :

```
NET "a<3>"          LOC = "P136";
NET "a<2>"          LOC = "P142";
NET "a<1>"          LOC = "P148";
NET "a<0>"          LOC = "P154";
NET "b<3>"          LOC = "P159";
NET "b<2>"          LOC = "P169";
NET "b<1>"          LOC = "P194";
NET "b<0>"          LOC = "P174";
NET "result<7>"     LOC = "P116";
NET "result<6>"     LOC = "P119";
NET "result<5>"     LOC = "P120";
NET "result<4>"     LOC = "P122";
NET "result<3>"     LOC = "P123";
NET "result<2>"     LOC = "P126";
NET "result<1>"     LOC = "P127";
NET "result<0>"     LOC = "P128";
```

Experiment1d) VHDL CODE FOR 4 BIT DIVISION IN BEHAVIORAL MODELLING

```
--------------------------------------------------------------------------------
--In VPTB-10 :
--9  sw : a<3>(msb) ; 10 sw : a<2> ; 11 sw : a<1> ; 12 sw : a<0>(lsb) ;
--13 sw : b<3>(msb) ; 14 sw : b<2> ; 15 sw : b<1> ; 16 sw : b<0>(lsb) ;
--13 led : result<3>(msb) ; 14 led : result<2> ;
--15 led : result<1>       ; 16 led : result<0>(lsb) ;
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity division is
port(
        clk     : in  std_logic;                        --system clock(default 20MHz)
         a      : in  std_logic_vector(3 downto 0); --1st input value--
         b      : in  std_logic_vector(3 downto 0); --2nd input value--
         result : out std_logic_vector(3 downto 0) --output result--
   );
end division;

architecture Behavioral of division is
--signal declarations--
signal reg1,reg2 : integer;
signal delay     : std_logic_vector(2 downto 0) := "000";
begin
-------------------------4 BIT DIVISION MODULE----------------------------------
process(a,b,clk)
variable cnt : integer range 0 to 16 := 0;
begin
 if rising_edge(clk) then
     case delay is
         when "000" =>
                  reg1 <= conv_integer(a); --convert 4 bit a i/p as integer & store in reg1
                  reg2 <= conv_integer(b);  --convert 4 bit b i/p as integer & store in reg2
                 delay <= "001";
         when "001" =>
                  if(reg2 = 0)then              --check reg2 is zero or not
                           delay <= "011";
                  else
                           delay <= "010";
                  end if;
         when "010" =>
                  if (reg1 >= reg2) then  --if reg1 >= reg2 ,do repeated subtraction and increment
                           reg1 <= reg1 - reg2; --cnt by 1 every time till if condition becomes false.
                           cnt  := cnt + 1;
                           delay <= "010";
                  else
                           result(3 downto 0) <= conv_std_logic_vector(cnt,4); --convert cnt into
                           delay <= "100";           --4 bit binary value and store in result.
                  end if;
         when "011" =>
                           result(3 downto 0) <= "0000"; --o/p result as zero if reg2 =  0.
                           delay <= "100";
```

```
        when "100" =>
                        cnt := 0;
                        delay <= "000";
        when others =>
                        delay <= "000";
   end case;
  end if;
end process;
end Behavioral;
```

User Constraints file :

```
NET "clk"           LOC = "P184";
NET "a<3>"          LOC = "P136";
NET "a<2>"          LOC = "P142";
NET "a<1>"          LOC = "P148";
NET "a<0>"          LOC = "P154";
NET "b<3>"          LOC = "P159";
NET "b<2>"          LOC = "P169";
NET "b<1>"          LOC = "P194";
NET "b<0>"          LOC = "P174";
NET "result<3>"     LOC = "P123";
NET "result<2>"     LOC = "P126";
NET "result<1>"     LOC = "P127";
NET "result<0>"     LOC = "P128";
```

Experiment 2) VHDL CODE FOR SWITCH LED INTERFACE IN DATA FLOW MODELLING

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

entity swled is
port(
        sw  : in   std_logic_vector (16 downto 1);
        led : out  std_logic_vector (16 downto 1)
         );
end swled;

architecture Behavioral of swled is
begin

led     <= sw;
```

```
end Behavioral;
```

User Constraints file :

```
NET "led<10>"       LOC = "p119"  ;
NET "led<11>"       LOC = "p120"  ;
NET "led<12>"       LOC = "p122"  ;
NET "led<13>"       LOC = "p123"  ;
NET "led<14>"       LOC = "p126"  ;
NET "led<15>"       LOC = "p127"  ;
NET "led<16>"       LOC = "p128"  ;
NET "led<1>"        LOC = "p102"  ;
NET "led<2>"        LOC = "p106"  ;
NET "led<3>"        LOC = "p107"  ;
NET "led<4>"        LOC = "p108"  ;
NET "led<5>"        LOC = "p109"  ;
NET "led<6>"        LOC = "p112"  ;
NET "led<7>"        LOC = "p113"  ;
NET "led<8>"        LOC = "p115"  ;
NET "led<9>"        LOC = "p116"  ;
NET "sw<10>"        LOC = "p142"  ;
NET "sw<11>"        LOC = "p148"  ;
NET "sw<12>"        LOC = "p154"  ;
NET "sw<13>"        LOC = "p159"  ;
NET "sw<14>"        LOC = "p169"  ;
NET "sw<15>"        LOC = "p194"  ;
NET "sw<16>"        LOC = "p174"  ;
NET "sw<1>"         LOC = "p71"  ;
NET "sw<2>"         LOC = "p72"  ;
NET "sw<3>"         LOC = "p91"  ;
NET "sw<4>"         LOC = "p101"  ;
NET "sw<5>"         LOC = "p110"  ;
NET "sw<6>"         LOC = "p118"  ;
NET "sw<7>"         LOC = "p124"  ;
NET "sw<8>"         LOC = "p130"  ;
NET "sw<9>"         LOC = "p136"  ;
```

Experiment 3) VHDL CODE FOR MATRIX KEYPAD INTERFACE IN BEHAVIORAL MODELLING

```
----------------------------------------------------------------------------------
-- Press the push buttons of the matrix keypad in the VPTB-10 board and observe
-- the corresponding value in the seven segemnt display.
----------------------------------------------------------------------------------
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity matrixkey is
port(
                clk     : in  std_logic;                                --System clock 20 MHz
                b               : in  std_logic_vector(3 downto 0); --column matrix
                a               : out std_logic_vector(3 downto 0); --row    matrix
                dispcom   : out std_logic;
                disp      : out std_logic_vector(6 downto 0):="0000000"--Seven segment display
      );                                                -- for displaying the value of the matrixkey.
end matrixkey;

architecture Behavioral of matrixkey is
--signal declarations--
signal sig1 : std_logic_vector(12 downto 0);
begin
dispcom         <= '0'; --for common cathode seven segment display
----------------------4X4 MATRIX KEYPAD MODULE-----------------------
test: process(clk) is
begin
        if rising_edge(clk) then
                sig1 <= sig1 + 1;
                case sig1(10 downto 8) is
                        when "000" =>
                                a <= "0111"; --first row
                        when "001" =>
                                if b = "0111" then
                                        disp <= "1001111"; -- 3
                                elsif b = "1011" then
                                        disp <= "1011011"; -- 2
                                elsif b = "1101" then
                                        disp <= "0000110"; -- 1
                                elsif b = "1110" then
                                        disp <= "0111111"; -- 0
                                end if;
                        when "010" =>
                                a <= "1011"; --second row
                        when "011" =>
                                if b = "1110" then
                                        disp <= "1100110"; -- 4
                                elsif b = "1101" then
                                        disp <= "1101101"; -- 5
                                elsif b = "1011" then
```

```
                                        disp <= "1111101"; -- 6
                                elsif b = "0111" then
                                        disp <= "0000111"; -- 7
                                end if;
                        when "100" =>
                                a <= "1101"; --third row
                        when "101" =>
                                if b = "1110" then
                                        disp <= "1111111"; -- 8
                                elsif b= "1101" then
                                        disp <= "1100111"; -- 9
                                elsif b = "1011" then
                                        disp <= "1110111"; -- a
                                elsif b = "0111" then
                                        disp <= "1111100"; -- b
                                end if;
                        when "110" =>
                                a <= "1110"; --fourth row
                        when "111" =>
                                if b = "1110" then
                                        disp <= "0111001"; -- C
                                elsif b = "1101" then
                                        disp <= "1011110"; -- d
                                elsif b = "1011" then
                                        disp <= "1111001"; -- e
                                elsif b= "0111" then
                                        disp <= "1110001"; -- f
                                end if;
                        when others =>
                                        disp <= "0000000";
                end case;
        end if;
end process;
end Behavioral;
```

User Constraints file :

```
NET "clk"           LOC = "p184"  ;
NET "a<0>"          LOC = "p129" | PULLUP ;
NET "a<1>"          LOC = "p133" | PULLUP ;
NET "a<2>"          LOC = "p135" | PULLUP ;
NET "a<3>"          LOC = "p138" | PULLUP ;
NET "b<0>"          LOC = "p132" | PULLUP ;
NET "b<1>"          LOC = "p134" | PULLUP ;
NET "b<2>"          LOC = "p137" | PULLUP ;
NET "b<3>"          LOC = "p204" | PULLUP ;
NET "disp<0>"       LOC = "p179";
NET "disp<1>"       LOC = "p178";
NET "disp<2>"       LOC = "p144";
NET "disp<3>"       LOC = "p146";
NET "disp<4>"       LOC = "p147";
NET "disp<5>"       LOC = "p180";
NET "disp<6>"       LOC = "p181";
NET "dispcom"       LOC = "p145";
```

Experiment 4a) VHDL CODE FOR BUZZER INTERFACE IN BEHAVIORAL MODELLING

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

entity buzzer is
port(
                clk       : in  std_logic; --system clock default 20MHz
                buzzer : out std_logic
   );
end buzzer;

architecture Behavioral of buzzer is
--signal declarations--
signal delay     : std_logic_vector(30 downto 0);
begin
------------------BUZZER MODULE---------------------
process(clk)
begin
if rising_edge(clk) then
        delay   <= delay + 1;
        case delay(25 downto 24) is
```

```
                when "00"       =>
                        buzzer <= '0';
                when "10"       =>
                        buzzer <= '1';
                when others =>
        end case;
end if;
end process;
end Behavioral;
```

User Constraints file :

```
NET "clk"       LOC    = "p184";
NET "buzzer" LOC    = "p190" ;
```

Experiment 4b) VHDL CODE FOR RELAY INTERFACE IN BEHAVIORAL MODELLING

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

entity relay is
port
        (
                clk     : in std_logic;
                relay   : out std_logic
        );
end relay;

architecture Behavioral of relay is
--signal declarations--
signal delay    : std_logic_vector(30 downto 0);
begin
---------------------- RELAY INTERFACE MODULE ------------------------
process(clk)
begin

if rising_edge(clk) then
        delay   <= delay + 1;
        case delay(25 downto 24) is
                when "00"       =>
                        relay   <= '0';
                when "10"       =>
```

```
                        relay    <= '1';
                when others =>
        end case;
end if;
end process;

end Behavioral;
```

User Constraints file :

```
NET "clk"       LOC    = "p184";
NET "relay"     LOC    = "p193" ;
```

Experiment 5) VHDL CODE FOR SEVEN SEGMENT LED INTERFACE IN BEHAVIORAL MODELLING

```
--------------------------------------------------------------------------------
-- VPTB-10 board :
-- 5 sw : bin<3>(msb) ; 6 sw : bin<2> ; 7 sw : bin<1> ; 8 sw : bin<0>(lsb) ;
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sevenseg is
port(
           bin     : in  std_logic_vector(3 downto 0); ---Input binary value from 0 to 15---
          dispcom : out std_logic;
         disp    : out std_logic_vector(6 downto 0)  ---Seven segment display for displaying
 );                                                    ---  the hex value.
end sevenseg;

architecture Behavioral of sevenseg is
begin
dispcom         <= '0'; --for common cathode seven segment display
-------------------- SEVEN SEGMENT DISPLAY MODULE --------------------
process(bin)
begin
        case bin is
                when "0000" =>
                        disp <= "0111111";  --0--
                when "0001" =>
                        disp <= "0000110";  --1--
                when "0010" =>
                        disp <= "1011011";  --2--
```

```
                when "0011" =>
                        disp <= "1001111";  --3--
                when "0100" =>
                        disp <= "1100110";  --4--
                when "0101" =>
                        disp <= "1101101";  --5--
                when "0110" =>
                        disp <= "1111101";  --6--
                when "0111" =>
                        disp <= "0000111";  --7--
                when "1000" =>
                        disp <= "1111111";  --8--
                when "1001" =>
                        disp <= "1100111";  --9--
                when "1010" =>
                        disp <= "1110111";  --a--
                when "1011" =>
                        disp <= "1111100";  --b--
                when "1100" =>
                        disp <= "0111001";  --c--
                when "1101" =>
                        disp <= "1011110";  --d--
                when "1110" =>
                        disp <= "1111001";  --e--
                when "1111" =>
                        disp <= "1110001";  --f--
                when others =>
                        disp <= "0000000";
        end case;
end process;

end Behavioral;
```

User Constraints File :

```
NET "bin<3>"        LOC = "p71";
NET "bin<2>"        LOC = "p72";
NET "bin<1>"        LOC = "p91";
NET "bin<0>"        LOC = "p101";
NET "disp<0>"       LOC = "p179";
NET "disp<1>"       LOC = "p178";
NET "disp<2>"       LOC = "p144";
NET "disp<3>"       LOC = "p146";
NET "disp<4>"       LOC = "p147";
NET "disp<5>"       LOC = "p180";
```

```
NET "disp<6>"          LOC = "p181";
NET "dispcom"          LOC = "p145";
```

EXPERIMENT 6A) VHDL CODE FOR CLOCKWISE ROTATION OF STEPPER MOTOR

```
----------------------------------------------------------------------------------------------------
--FOR ON BOARD :
-- Connect the 5 pin RMC cable of the stepper motor to the 5 pin
-- connector P10(VPTB-10) and the jumper J13 of VPTB-10 must be closed in +5V
-- position.Observe the clockwise rotation of the stepper motor by rolling
-- a small bit of paper on its top.
----------------------------------------------------------------------------------------------------
--FOR VVSI -32 :
-- Connect external io connector P6 (VPTB-10) to the external io connector P3
-- (VVSI-32).Connect the 5 pin RMC cable of the stepper motor to the 5 pin
-- connector P4(VVSI-32) and the jumpers J5 +5V of VPTB-10 & J2[VCC MVCC] of VVSI-32
-- must be closed.Observe the clockwise rotation of the stepper motor by rolling
-- a small bit of paper on its top.
----------------------------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity stepperclk is
port(
                clk : in  std_logic;                    -- System clock 20 MHz
                db  : out bit_vector(4 downto 1)-- Data Bus
   );
end stepperclk;

architecture Behavioral of stepperclk is
--signal declarations--
signal delay : std_logic_vector(20 downto 1); -- Delay
begin
-------------------- STEPPER MOTOR MODULE -----------------------------
stepmotor: process(clk)
        begin
                if rising_edge(clk) then
                     delay <= delay + 1;
                        case delay(20 downto 18) is -- Set your Delay Time
                                when "000" =>  --       FOR CLOCKWISE DIRECTION     --
                                        db <= "1001";   -- 1st Data
                                when "010" =>
```

```
                                db <= "0101";   -- 2nd Data
                            when "100" =>
                                db <= "0110";   -- 3rd Data
                            when "110" =>
                                db <= "1010";   -- 4th Data
                            when others =>
                        end case;
                end if;
    end process stepmotor;
end Behavioral;
```

User Constraints File :

For ON BOARD :

```
NET "clk"           LOC = "P184"  ;
NET "db<1>"         LOC = "P76"  ;
NET "db<2>"         LOC = "P77"  ;
NET "db<3>"         LOC = "P78"  ;
NET "db<4>"         LOC = "P82"  ;
```

For VVSI-32 :

```
NET "clk"           LOC = "P184"  ;
NET "db<1>"         LOC = "P5"  ;#5 pin of ext ioc P6
NET "db<2>"         LOC = "P4"  ;#4 pin of ext ioc P6
NET "db<3>"         LOC = "P2"  ;#2 pin of ext ioc P6
NET "db<4>"         LOC = "P3"  ;#3 pin of ext ioc P6
```

EXPERIMENT 6B) VHDL CODE FOR ANTI-CLOCKWISE ROTATION OF STEPPER MOTOR

```
--------------------------------------------------------------------------------
--FOR ON BOARD:
-- Connect the 5 pin RMC cable of the stepper motor to the 5 pin
-- connector P10(VPTB-10) and the jumper J13 of VPTB-10 must be closed in +5V
-- position.Observe the anti-clockwise rotation of the stepper motor by rolling
-- a small bit of paper on its top.
--------------------------------------------------------------------------------
--FOR VVSI-32:
-- Connect external io connector P6 (VPTB-10) to the external io connector P3
-- (VVSI-32).Connect the 5 pin RMC cable of the stepper motor to the 5 pin
-- connector P4(VVSI-32) and the jumpers J5 +5V of VPTB-10 & J2[VCC MVCC] of VVSI-32
-- must be closed.Observe the anti-clockwise rotation of the stepper motor by rolling
-- a small bit of paper on its top.
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity stepperanti is
port(
                clk : in  std_logic;                -- System clock 20 MHz
                db  : out bit_vector(4 downto 1)          -- Data Bus
   );
end stepperanti;

architecture Behavioral of stepperanti is
--signal declarations--
signal delay : std_logic_vector(20 downto 1); -- Delay
begin
----------------------STEPPER MOTOR MODULE----------------------
stepmotor: process(clk)
begin
          if rising_edge(clk) then
                   delay <= delay + 1;
                            case delay(20 downto 18) is   -- Set your Delay Time

                                     when "000" =>           -- FOR ANTICLOCKWISE DIRECTION
                                              db <= "1010";   -- 4th Data
                                     when "010" =>
                                              db <= "0110";   -- 3rd Data
                                     when "100" =>
                                              db <= "0101";   -- 2nd Data
```

```
                        when "110" =>
                                db <= "1001";   -- 1st Data
                        when others =>
                    end case;
          end if;
        end process stepmotor;
end Behavioral;
```

User Constraints File :

For ON BOARD :

```
NET "clk"           LOC = "P184"  ;
NET "db<1>"         LOC = "P76"  ;
NET "db<2>"         LOC = "P77"  ;
NET "db<3>"         LOC = "P78"  ;
NET "db<4>"         LOC = "P82"  ;
```

For VVSI-32 :

```
NET "clk"           LOC = "P184"  ;
NET "db<1>"         LOC = "P5"  ;#5 pin of ext ioc P6
NET "db<2>"         LOC = "P4"  ;#4 pin of ext ioc P6
NET "db<3>"         LOC = "P2"  ;#2 pin of ext ioc P6
NET "db<4>"         LOC = "P3"  ;#3 pin of ext ioc P6
```

Experiment 7) VHDL CODE FOR TRAFFIC LIGHT CONTROL

```
--------------------------------------------------------------------------------
-- SYSTEM CLOCK FREQUENCY IS 20 MHZ (Default)
-- Connect VLIM connector P8 (VPTB-10) and VLIM connector (TLC) using 26 pin FRC
-- cable.In VPTB-10:
-- SW34 - rst ;
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity traffic is
port(
        clk   : in  std_logic;        --system clock
        rst   : in  std_logic;         --active high,asynchronous reset
        dir   : out std_logic_vector(2 downto 0);
        p1    : out std_logic_vector(4 downto 0); --road1
        p2    : out std_logic_vector(4 downto 0); --road2
```

```
            p3    : out std_logic_vector(4 downto 0); --road3
            p4    : out std_logic_vector(4 downto 0); --road4
            pl    : out std_logic_vector(3 downto 0)--pedestrian
   );
end traffic;

architecture Behavioral of traffic is
--signal declarations--
signal sig : std_logic_vector(30 downto 0) := (others => '0');
begin
        dir       <= "111";
process(clk,rst)
begin
 if (rst = '0') then  --At reset condition
        p1 <= "00100";          --road1 is red
        p2 <= "00100";          --road2 is red
        p3 <= "00100";          --road3 is red
        p4 <= "00100";          --road4 is red
        pl <= "1111";           --pedestrian is red
        sig <= (others => '0');
 elsif rising_edge(clk) then
   sig <= sig + 1;
                case sig(29 downto 24) is
                        when "000000" =>
                                        p1 <= "10011";      --road1 is Green
                                        p2 <= "00100";      --all other roads are red
                                        p3 <= "00100";
                                        p4 <= "00100";
                                        pl <= "1111";
                        when "000100" =>
                                        p1 <= "01000";     --road1 is Yellow
                                        p2 <= "00100";     --all other roads are red
                                        p3 <= "00100";
                                        p4 <= "00100";
                                        pl <= "1111";
                        when "001000" =>
                                        p1 <= "00100";    --all other roads are red
                                        p2 <= "10011";    --road2 is Green
                                        p3 <= "00100";
                                        p4 <= "00100";
                                        pl <= "1111";
                        when "001100" =>
                                        p1 <= "00100";   --all other roads are red
                                        p2 <= "01000";   --road2 is Yellow
                                        p3 <= "00100";
```

```
                        p4 <= "00100";
                        pl <= "1111";
                when "010000" =>
                        p1 <= "00100";
                        p2 <= "00100";      --all other roads are red
                        p3 <= "10011";      --road3 is Green
                        p4 <= "00100";
                        pl <= "1111";
                when "010100" =>
                        p1 <= "00100";
                        p2 <= "00100";      --all other roads are red
                        p3 <= "01000";      --road3 is yellow
                        p4 <= "00100";
                        pl <= "1111";
                when "011000" =>
                        p1 <= "00100";
                        p2 <= "00100";
                        p3 <= "00100";      --all other roads are red
                        p4 <= "10011";      --road4 is Green
                        pl <= "1111";
                when "011100" =>
                        p1 <= "00100";
                        p2 <= "00100";
                        p3 <= "00100";      --all other roads are red
                        p4 <= "01000";      --road4 is Yellow
                        pl <= "1111";
                when "100000" =>
                        p1 <= "00100";
                        p2 <= "00100";
                        p3 <= "00100";
                        p4 <= "00100";      --all other roads are red
                        pl <= "0000";       --Pedestrain is green which means
--the pedestrian can cross within the roads 1,2,3,4 respectively since no vehicles
--will pass over for this duration of time.
                when "100100" =>
                        sig  <= (others => '0');
                when others =>
        end case;
 end if;
end process;

end Behavioral;
```

User Constraints File :

```
NET "clk"          LOC = "p184"  ;
NET "rst"          LOC = "p175"  ;
NET "p1<0>"        LOC = "p31"  ;#pao
NET "p1<1>"        LOC = "p33"  ;#pa1
NET "p1<2>"        LOC = "p34"  ;#pa2
NET "p1<3>"        LOC = "p35"  ;#pa3
NET "p1<4>"        LOC = "p36"  ;#pa4
NET "p2<0>"        LOC = "p39"  ;#pa5
NET "p2<1>"        LOC = "p40"  ;#pa6
NET "p2<2>"        LOC = "p41"  ;#pa7
NET "p2<3>"        LOC = "p75"  ;#pc0
NET "p2<4>"        LOC = "p74"  ;#pc1
NET "p3<0>"        LOC = "p69"  ;#pc2
NET "p3<1>"        LOC = "p68"  ;#pc3
NET "p3<2>"        LOC = "p65"  ;#pc4
NET "p3<3>"        LOC = "p64"  ;#pc5
NET "p3<4>"        LOC = "p63"  ;#pc6
NET "p4<0>"        LOC = "p62"  ;#pc7
NET "p4<1>"        LOC = "p50"  ;#pb4
NET "p4<2>"        LOC = "p55"  ;#pb5
NET "p4<3>"        LOC = "p56"  ;#pb6
NET "p4<4>"        LOC = "p60"  ;#pb7
NET "pl<0>"        LOC = "p45"  ;#pb0
NET "pl<1>"        LOC = "p47"  ;#pb1
NET "pl<2>"        LOC = "p48"  ;#pb2
NET "pl<3>"        LOC = "p49"  ;#pb3
NET "dir<2>"       LOC = "p30"  ;
NET "dir<1>"       LOC = "p42"  ;
NET "dir<0>"       LOC = "p61"  ;
```

Experiment 8) VHDL CODE FOR 4 BIT UP/DOWN COUNTER IN BEHAVIORAL MODELLING

```
--------------------------------------------------------------------------------
--In VPTB-10 board :
--1 sw :  dir ; SW34 : rst ;
--13 led : led<3> (msb) ; 14 led : led<2> ; 15 led : led<1> ; 16 led : led<0> (lsb) ;
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity counter is
```

```
port(
        clk  : in  std_logic;           --system clock(default 20 MHz)--
        rst  : in  std_logic;           --reset--
        dir  : in  std_logic;           --up/down count direction--
        led  : out std_logic_vector(3 downto 0)--led display for counter output--
   );
end counter;

architecture Behavioral of counter is
--signal declarations--
signal cntr : std_logic_vector(28 downto 0) := (others => '0');
begin
-------------------- 4 BIT UP/DOWN COUNTER MODULE --------------------
process(clk)
begin
if rising_edge(clk) then
        if (rst = '0') then             --when rst switch is low,counter is reset.
          cntr <= (others => '0');
          led <= cntr(28 downto 25);
        else
                if (dir = '1') then
                                cntr <= cntr + 1;       --up count--
                                led <= cntr(28 downto 25);
                else
                                cntr <= cntr - 1        ;  --down count--
                                led <= cntr(28 downto 25);
                end if;
        end if;
end if;
end process;

end Behavioral;
```

User Constraints File :

```
NET "clk"           LOC = "P184";
NET "rst"           LOC = "P175";
NET "dir"           LOC = "P71";
NET "led<3>"        LOC = "P123";
NET "led<2>"        LOC = "P126";
NET "led<1>"        LOC = "P127";
NET "led<0>"        LOC = "P128";
```

VHDL CODE FOR SIMULATION OF 4 BIT UP/DOWN COUNTER IN BEHAVIORAL MODELLING

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

entity counter is
port(
          clk  : in  std_logic;     --system clock(default 20 MHz)--
           rst  : in  std_logic;     --active low reset--
           dir  : in  std_logic;     --up/down count direction--
           led  : out std_logic_vector(3 downto 0)--led display--
   );
end counter;

architecture Behavioral of counter is
--signal declarations--
signal cntr : std_logic_vector(3 downto 0) := "0000";
begin
-------------------- 4 BIT UP/DOWN COUNTER MODULE --------------------
process(clk,rst,dir)
begin
if rising_edge(clk) then
          if (rst = '0') then          --active low reset--
            cntr <= "0000";
            led <= cntr(3 downto 0);
          else
                    if (dir = '1') then
                                        cntr <= cntr + 1;          --up count--
                                        led <= cntr(3 downto 0);
                    else
                                        cntr <= cntr - 1;          --down count--
                                        led <= cntr(3 downto 0);
                    end if;
          end if;
end if;
end process;
end Behavioral;
```

EXPERIMENT 9) VHDL CODE FOR DISPLAYING TEXT MESSAGE "VI MICROSYSTEMS" IN LCD

```
--SYSTEM CLOCK IS 20 MHZ
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity lcd is
port(
                clk  : in  std_logic;      --system clock 20MHz (Default)
                rs   : out std_logic;      --register select
                cs   : out std_logic;      --chip select
                data : out std_logic_vector(7 downto 0)--lcd data lines
        );
end lcd;

architecture arch_lcd of lcd is
--array and constant declarations--
type main1 is array(1 to 5) of std_logic_vector(7 downto 0);
constant cmd : main1:=(x"38",x"06",x"01",x"0f",x"80"); --lcd initialisation commands
type main2 is array(1 to 15) of std_logic_vector(7 downto 0);
constant text : main2:=(x"56",x"49",x"20",x"4d",x"49",x"43",x"52",x"4f",x"53",
                        x"59",x"53",x"54",x"45",x"4d",x"53");--"VI MICROSYSTEMS"--
--signal declarations--
signal sig : std_logic_vector(25 downto 0) := (others => '0');
signal i,j : integer := 1;
begin
----------------------LCD DISPLAY MODULE------------------------
process(clk)
begin
 if rising_edge(clk) then
        sig <= sig + 1;
                case sig(25 downto 20) is
                        when "000000" =>        rs   <= '0';
                        when "000001" =>       data <= cmd(i);          --lcd initialization
                        when "000010" =>       cs   <= '1';
                        when "000011" =>       cs   <= '0';
                                               i    <= i + 1;
                                               if (i <= 5) then
                                                  sig(25 downto 20)<="000001";
                                               else
                                                 sig(25 downto 20)<="000100";
                                                 j <= 1;
```

```
                                    end if;
                when "000100" =>    rs    <= '1';
                when "000101" =>    data <= text(j);--"VI MICROSYSTEMS"
                when "000110" =>    cs    <= '1';
                when "000111" =>    cs    <= '0';
                                    j     <= j + 1;
                                    if (j < 15) then
                                        sig(25 downto 20)<="000101";
                                    else
                                        sig(25 downto 20)<="000000";
                                        i <= 1;
                                    end if;
                when others   => null;
        end case;
 end if;
end process;
end architecture;
```

User Constraints File :

```
NET "clk"           LOC = "p184"  ;
NET "rs"            LOC = "p83"  ;
NET "cs"            LOC = "p89"  ;
NET "data<0>"       LOC = "p90"  ;
NET "data<1>"       LOC = "p93"  ;
NET "data<2>"       LOC = "p94"  ;
NET "data<3>"       LOC = "p96"  ;
NET "data<4>"       LOC = "p97"  ;
NET "data<5>"       LOC = "p98"  ;
NET "data<6>"       LOC = "p99"  ;
NET "data<7>"       LOC = "p100"  ;
```

EXPERIMENT 10) VHDL CODE TO GENERATE PWM SIGNAL FOR DC MOTOR CONTROL

```
-------------------------------------------------------------------------------------------------------
--FOR ON BOARD :
-- The jumpers J1 (VPTB-10) must be shorted in +5V position.Connect the 2 pin RMC cable
-- of the DC motor to the 2 pin RMC connectors P1(VPTB-10). Press sw14 of the matrix
-- keypad for increasing the duty cycle and sw15 of the keypad for decreasing the duty
-- cycle of the dc motor.
-------------------------------------------------------------------------------------------------------
--FOR VVSI - 32 :
-- Connect the external io connector P6 (VPTB-10) to the external io connector P3
-- (VVSI-32).The jumpers J5 +5V (VPTB-10) and J1 [MVCC VCC](VVSI-32) must be shorted.
-- Connect the 2 pin RMC cable of the DC motor to the 2 pin RMC connectors P5(VVSI- 32).
```

```
-- Press sw13 of the matrix keypad for increasing the duty cycle and sw14 of the keypad
-- for decreasing the duty cycle of the dc motor.
----------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity dcmotor is
port(
                clk     : in  STD_LOGIC;    --System clock 20 MHz
                pulse   : out STD_LOGIC;
                dc_out : out STD_LOGIC;
                a       : out STD_LOGIC_VECTOR(3 downto 0);
                sw1     : in  STD_LOGIC;
                sw2     : in  STD_LOGIC
        );
end dcmotor;

architecture Behavioral of dcmotor is
--signal declarations--
signal cnt1 : integer := 200;
signal cnt  : integer := 10;
begin
a       <= "0111";
dc_out <= '0';
-----------------------------------------------------------------
process(clk)
variable h  : integer := 0;
begin
 if rising_edge(clk) then
        h := h + 1;
                if      (h < cnt1)      then
                        pulse <= '1';
                elsif(h > cnt1) and (h < 2000) then
                        pulse <= '0';
                elsif(h > 2000) then
                        h := 0;
                end if;
 end if;
end process;
-----------------------------------------------------------------
process(clk)
variable h,f : integer := 0;
begin
```

```
 if rising_edge(clk) then
                    if (sw1 = '0') then
                              f := f + 1;
                                        if (f > 1000000) then
                                                  cnt <= cnt + 1;
                                                  f := 0;
                                        if (cnt >= 90) then
                                                  cnt <= 90;
                                                  end if;
                                        end if;
                    end if;
                    ---------------------
                    if (sw2 = '0') then
                              f := f + 1;
                               if (f > 1000000) then
                                        cnt <= cnt - 1;
                                  f := 0;
                                         if (cnt <= 10) then
                                        cnt <= 10;
                                          end if;
                               end if;
                    end if;
                    cnt1 <= cnt * 20;
 end if;
end process;
end Behavioral;
```

User Constraints File :

For ON BOARD :

```
NET "clk"           LOC = "P184";
NET "sw1"           LOC = "p132" | PULLUP ;
NET "sw2"           LOC = "p134" | PULLUP ;
NET "a<0>"          LOC = "p129" | PULLUP ;
NET "a<1>"          LOC = "p133" | PULLUP ;
NET "a<2>"          LOC = "p135" | PULLUP ;
NET "a<3>"          LOC = "p138" | PULLUP ;
NET "pulse"         LOC = "P139";
NET "dc_out"        LOC = "P206";
```

For VVSI-32 :

```
NET "clk"           LOC = "P184";
NET "sw1"           LOC = "p132" | PULLUP ;
```

```
NET "sw2"           LOC = "p134" | PULLUP ;
NET "a<0>"          LOC = "p129" | PULLUP ;
NET "a<1>"          LOC = "p133" | PULLUP ;
NET "a<2>"          LOC = "p135" | PULLUP ;
NET "a<3>"          LOC = "p138" | PULLUP ;
NET "pulse"         LOC = "P9";
NET "dc_out"        LOC = "P8";
```

EXPERIMENT 11) VHDL CODE FOR 4:1 MULTIPLEXER IN BEHAVIORAL MODELLING:

```
--------------------------------------------------------------------------------
--MULTIPLEXER  Many input One output ( 2^N inputs ,N select inputs,1 output ) --
--8x1 MULTIPLEXER -- 4 input lines,2 select lines, 1 output line --
--In VPTB-10 board the first 2 switches are for the select lines the next 4 switches
--are the input lines of the multiplexer and the 1st LED is the output line
--of the multiplexer.
--In VPTB-10 board :
--1 sw  : sel<1> ; 2 sw  : sel<0> ;
--3 sw  : inp<0> ; 4 sw  : inp<1> ; 5 sw  : inp<2> ; 6 sw  : inp<3> ;
--1 led : muxout ;
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mux is
port(
     inp    : in  std_logic_vector(3 downto 0);--mux input lines
     sel    : in  std_logic_vector(1 downto 0);--mux sel input lines
     muxout : out std_logic                    --mux output line
   );
end mux;

architecture Behavioral of mux is
begin
----------------- 4:1 MULTIPLEXER MODULE ---------------------
process(inp,sel)
begin
 case sel is
          when "00" =>
                                        muxout <= inp(0);       -- mux O/P=1 I/P--
          when "01" =>
                                        muxout <= inp(1);       -- mux O/P=2 I/P--
          when "10" =>
```

```
                                    muxout <= inp(2);       -- mux O/P=3 I/P--
        when "11" =>
                                    muxout <= inp(3);       -- mux O/P=4 I/P--
  when others =>
 end case;
end process;

end Behavioral;
```

User Constraints File :

```
NET "sel<1>"        LOC = "P71";
NET "sel<0>"        LOC = "P72";
NET "inp<0>"        LOC = "P91";
NET "inp<1>"        LOC = "P101";
NET "inp<2>"        LOC = "P110";
NET "inp<3>"        LOC = "P118";
NET "muxout"        LOC = "P102";
```

EXPERIMENT 11) VHDL CODE FOR 1:4 DEMULTIPLEXER IN BEHAVIORAL MODELLING:

```
----------------------------------------------------------------------------------
-- DEMULTIPLEXER -- One input many outputs (1 input , 2^N outputs ,N select input)
-- 1X4 DEMULTIPLEXER -- 1 input line, 2 select inputs, 4 output lines--
-- In VPTB-10 board :
-- 1 sw  : dmuxin ; 2 sw  : sel<1> ; 3 sw  : sel<0> ;
-- 1 led : oup<0> ; 2 led : oup<1> ; 3 led : oup<2> ; 4 led : oup<3> ;
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity demux is
port(
     dmuxin : in  std_logic; --demux input line
     sel    : in  std_logic_vector(1 downto 0);--demux select inputs
     oup    : out std_logic_vector(3 downto 0) --demux outputs
   );
end demux;

architecture Behavioral of demux is
begin
process(dmuxin,sel)
```

```
begin
        case sel is
                when "00" =>
                        oup(0) <= dmuxin;   --1 dmux o/p = dmux i/p--
                        oup(1) <= '0';
                        oup(2) <= '0';
                        oup(3) <= '0';
                when "01"     =>
                        oup(0) <= '0';
                        oup(1) <= dmuxin;     --2 dmux o/p = dmux i/p--
                        oup(2) <= '0';
                        oup(3) <= '0';
                when "10"     =>
                        oup(0) <= '0';
                        oup(1) <= '0';
                        oup(2) <= dmuxin;     --3 dmux o/p = dmux i/p--
                        oup(3) <= '0';
                when "11"     =>
                        oup(0) <= '0';
                        oup(1) <= '0';
                        oup(2) <= '0';
                        oup(3) <= dmuxin;     --4 dmux o/p = dmux i/p--
                when others =>
        end case;
end process;
end Behavioral;
```

User Constraints File :

```
NET "dmuxin"        LOC = "P71";
NET "sel<1>"        LOC = "P72";
NET "sel<0>"        LOC = "P91";
NET "oup<0>"        LOC = "P102";
NET "oup<1>"        LOC = "P106";
NET "oup<2>"        LOC = "P107";
NET "oup<3>"        LOC = "P108";
```

Experiment 12) VHDL CODE FOR 3:8 DECODER IN BEHAVIORAL MODELLING

```
----------------------------------------------------------------------------------
-- DECODER --  n INPUT 2^n OUTPUT --
-- 3:8 DECODER -- 3 INPUT 8 OUPUT --
-- In VPTB-10 board :
-- 14 sw  : inp<2> ; 15 sw : inp<1> ; 16 sw : inp<0>  ;
-- 9  led : oup<7> ; 10 led : oup<6> ; 11 led : oup<5> ; 12 led : oup<4> ;
-- 13 led : oup<3> ; 14 led : oup<2> ; 15 led : oup<1> ; 16 led : oup<0> ;
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decdr is
port(
            inp : in  std_logic_vector(2 downto 0);  --decoder inputs--
            oup : out std_logic_vector(7 downto 0)   --decoder outputs--
    );
end decdr;
architecture Behavioral of decdr is
begin
process(inp)
begin
          case inp is
                    when "000" =>
                              oup <= "00000001";
                    when "001" =>
                              oup <= "00000010";
                    when "010" =>
                              oup <= "00000100";
                    when "011" =>
                              oup <= "00001000";
                    when "100" =>
                              oup <= "00010000";
                    when "101" =>
                              oup <= "00100000";
                    when "110" =>
                              oup <= "01000000";
                    when "111" =>
                              oup <= "10000000";
                    when others =>
                              oup <= "--------";--don't care--
          end case;
end process;
end Behavioral;
```

User Constraints File :

```
NET "inp<2>"        LOC = "p169";
NET "inp<1>"        LOC = "p194";
NET "inp<0>"        LOC = "p174";
NET "oup<7>"        LOC = "p116";
NET "oup<6>"        LOC = "p119";
NET "oup<5>"        LOC = "p120";
NET "oup<4>"        LOC = "p122";
NET "oup<3>"        LOC = "p123";
NET "oup<2>"        LOC = "p126";
NET "oup<1>"        LOC = "p127";
NET "oup<0>"        LOC = "p128";
```

Experiment 12) VHDL CODE FOR 8:3 ENCODER IN BEHAVIORAL MODELLING

```
--------------------------------------------------------------------------------
-- ENCODER -- 2^n INPUT n OUTPUT --
-- 8:3 ENCODER -- 8 INPUT 3 OUPUT --
--------------------------------------------------------------------------------
--In VPTB-10 board :
--9  sw  : inp<7> ; 10 sw : inp<6> ; 11 sw : inp<5> ; 12 sw : inp<4> ;
--13 sw  : inp<3> ; 14 sw : inp<2> ; 15 sw : inp<1> ; 16 sw : inp<0> ;
--14 led : oup<2> ; 15 led : oup<1> ; 16 led : oup<0> ;
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity encdr is
port(
            inp : in   std_logic_vector (7 downto 0); --encoder inputs
           oup : out  std_logic_vector (2 downto 0) --encoder outputs
   );
end encdr;

architecture Behavioral of encdr is
begin
process(inp)
begin
         case inp is
                  when "00000001" =>
                           oup <= "000";
                  when "00000010" =>
```

```
                    oup <= "001";
                when "00000100" =>
                    oup <= "010";
                when "00001000" =>
                    oup <= "011";
                when "00010000" =>
                    oup <= "100";
                when "00100000" =>
                    oup <= "101";
                when "01000000" =>
                    oup <= "110";
                when "10000000" =>
                    oup <= "111";
                when others =>
                    oup <= "---";
        end case;
end process;
end Behavioral;
```

User Constraints File :

```
NET "inp<7>"        LOC = "p136";
NET "inp<6>"        LOC = "p142";
NET "inp<5>"        LOC = "p148";
NET "inp<4>"        LOC = "p154";
NET "inp<3>"        LOC = "p159";
NET "inp<2>"        LOC = "p169";
NET "inp<1>"        LOC = "p194";
NET "inp<0>"        LOC = "p174";
NET "oup<2>"        LOC = "p126";
NET "oup<1>"        LOC = "p127";
NET "oup<0>"        LOC = "p128";
```

EXPERIMENT 12) VHDL CODE FOR 4 BIT SERIAL IN PARALLEL OUT SHIFT REGISTER IN BEHAVIORAL MODELLING

```
----------------------------------------------------------------------------------
-- In VPTB-10 board :
-- SW34 : rst ; 1 sw : si ;
-- 13 led : po<3> ; 14 led : po<2> ; 15 led : po<1> ; 16 led : po<0> ;
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity shiftreg is
port(
     clk  : in  std_logic;          --system clock(Default 20 MHz)
     rst  : in  std_logic;          --active low reset--
     si   : in  std_logic;          --serial input--
     po   : out std_logic_vector(3 downto 0)--parallel output--
   );
end shiftreg;

architecture Behavioral of shiftreg is
--signal declarations--
signal reg    : std_logic_vector(3 downto 0) := "0000";
signal clk_sr : std_logic := '0';
begin
po <= reg;
----------------------------------------
-- SLOW CLOCK MODULE   --
----------------------------------------
process(clk)
variable i    : integer := 0;
variable temp : std_logic := '0';
begin
 if rising_edge(clk) then
          i := i + 1;
           if (i = 10000000) then          -- required clk = system clk/coutn val
                    temp := not(temp);     --                = 20000000 / 20000000
                    clk_sr <= temp;        --                = 1 Hz
                    i := 0;
           end if;
 end if;
end process;
```

```
----------------------------------------------------------------
--    SERIAL IN PARALLEL OUT SHIFT REGISTER  MODULE       --
----------------------------------------------------------------
process(clk_sr,rst)
begin
  if rising_edge(clk_sr) then
                if (rst = '0') then                          --synchronous,active low reset--
                        reg <= "0000";
                else
                        reg <= reg(2 downto 0) & si;
                end if;
  end if;
end process;
end Behavioral;
```

User Constraints File :

```
NET "clk"           LOC = "p184";
NET "rst"           LOC = "p175";
NET "si"            LOC = "p71";
NET "po<3>"         LOC = "p123";
NET "po<2>"         LOC = "p126";
NET "po<1>"         LOC = "p127";
NET "po<0>"         LOC = "p128";
```

CHAPTER-10
SYLLABUS PROGRAMS IN VERILOG

Experiment1a) VERILOG CODE FOR 4 BIT ADDITION IN DATA FLOW MODELLING

```
//////////////////////////////////////////////////////////////////////////////////
// In VPTB-10 :
// 9  sw : a[3] ; 10 sw : a[2] ; 11 sw : a[1] ; 12 sw : a[0] ;
// 13 sw : b[3] ; 14 sw : b[2] ; 15 sw : b[1] ; 16 sw : b[0] ;
// 12 led : oup[4](carry) ;
// 13 led : oup[3](msb) ; 14 led : oup[2] ;
// 15 led : oup[1]          ; 16 led  : oup[0](lsb) ;
//////////////////////////////////////////////////////////////////////////////////
module addition4bit(a,b,oup); //module declaration with input,output list
//input ports
input  [3:0] a;   // 1st i/p value
input  [3:0] b;  // 2nd i/p value
//output ports
output [4:0] oup; // result

// 4 bit addition

//expression is evaluvated as soon as one of the operands in the RHS (here a , b) changes
//and  the result is assigned to the LHS net.
//Multiple Statements using the assign keyword are executed at the same time.

assign oup = a + b;
endmodule
```

User Constraints file :

```
NET "a[3]"          LOC = "P136";
NET "a[2]"          LOC = "P142";
NET "a[1]"          LOC = "P148";
NET "a[0]"          LOC = "P154";
NET "b[3]"          LOC = "P159";
NET "b[2]"          LOC = "P169";
NET "b[1]"          LOC = "P194";
NET "b[0]"          LOC = "P174";
NET "oup[4]"        LOC = "P122";
NET "oup[3]"        LOC = "P123";
NET "oup[2]"        LOC = "P126";
NET "oup[1]"        LOC = "P127";
```

```
NET "oup[0]"          LOC = "P128";
```

Experiment1b) VERILOG CODE FOR 4 BIT SUBTRACTION IN DATA FLOW MODELLING

```
//////////////////////////////////////////////////////////////////////////////////
// In VPTB-10 :
// 9  sw : a[3] ; 10 sw : a[2] ; 11 sw : a[1] ; 12 sw : a[0] ;
// 13 sw : b[3] ; 14 sw : b[2] ; 15 sw : b[1] ; 16 sw : b[0] ;
// 12 led : oup[4](borrow) ;
// 13 led : oup[3](msb) ; 14 led : oup[2] ;
// 15 led : oup[1]          ; 16 led : oup[0](lsb) ;
//////////////////////////////////////////////////////////////////////////////////
module subtract4bit(a,b,oup); //module declaration with input,output list
// input ports
input [3:0] a; // 1st i/p value
input [3:0] b; // 2nd i/p value
// output ports
output [4:0] oup; // result

// 4 bit subtraction

// expression is evaluvated as soon as one of the operands in the RHS (here a , b) changes
// and  the oup is assigned to the LHS net.Multiple Statements using the assign keyword
// are executed at the same time.

assign oup = a - b;

endmodule
```

User Constraints file :

```
NET "a[3]"          LOC = "P136";
NET "a[2]"          LOC = "P142";
NET "a[1]"          LOC = "P148";
NET "a[0]"          LOC = "P154";
NET "b[3]"          LOC = "P159";
NET "b[2]"          LOC = "P169";
NET "b[1]"          LOC = "P194";
NET "b[0]"          LOC = "P174";
NET "oup[4]"        LOC = "P122";
NET "oup[3]"        LOC = "P123";
NET "oup[2]"        LOC = "P126";
NET "oup[1]"        LOC = "P127";
```

```
NET "oup[0]"          LOC = "P128";
```

Experiment1c) VERILOG CODE FOR 4 BIT MULTIPLICATION IN DATA FLOW MODELLING

```
//////////////////////////////////////////////////////////////////////////////////
// In VPTB-10 :
// 9  sw : a[3] ; 10 sw : a[2] ; 11 sw : a[1] ; 12 sw : a[0] ;
// 13 sw : b[3] ; 14 sw : b[2] ; 15 sw : b[1] ; 16 sw : b[0] ;
// 9  led : oup[7](msb) ; 10 led : oup[6] ;
// 11 led : oup[5]          ; 12 led : oup[4] ;
// 13 led : oup[3]          ; 14 led : oup[2] ;
// 15 led : oup[1]          ; 16 led : oup[0](lsb) ;
//////////////////////////////////////////////////////////////////////////////////
module mult4bit(a,b,oup); //module declaration with input,output list
// input ports
input [3:0] a;  // 1st input value
input [3:0] b;  // 2nd input value
// output ports
output [7:0] oup; // result

// 4 bit multiplication

//Expression is evaluvated as soon as one of the operands in the RHS (here a , b) changes
//and the result is assigned to the LHS net.
//Multiple Statements using the assign keyword are executed at the same time.

assign oup = a * b;

endmodule
```

User Constraints file :

```
NET "a[3]"          LOC = "P136";
NET "a[2]"          LOC = "P142";
NET "a[1]"          LOC = "P148";
NET "a[0]"          LOC = "P154";
NET "b[3]"          LOC = "P159";
NET "b[2]"          LOC = "P169";
NET "b[1]"          LOC = "P194";
NET "b[0]"          LOC = "P174";
NET "oup[7]"        LOC = "P116";
NET "oup[6]"        LOC = "P119";
NET "oup[5]"        LOC = "P120";
NET "oup[4]"        LOC = "P122";
```

```
NET "oup[3]"        LOC = "P123";
NET "oup[2]"        LOC = "P126";
NET "oup[1]"        LOC = "P127";
NET "oup[0]"        LOC = "P128";
```

Experiment1d) VERILOG CODE FOR 4 BIT DIVISION IN BEHAVIORAL MODELLING

```
//////////////////////////////////////////////////////////////////////////////////
// In VPTB-10 :
// 9  sw : a[3](msb) ; 10 sw : a[2] ; 11 sw : a[1] ; 12 sw : a[0](lsb) ;
// 13 sw : b[3](msb) ; 14 sw : b[2] ; 15 sw : b[1] ; 16 sw : b[0](lsb) ;
// 13 led : result[3](msb); 14 led : result[2] ;
// 15 led : result[1]       ; 16 led : result[0](lsb) ;
//////////////////////////////////////////////////////////////////////////////////
module division(clk,a,b,result); //module declaration
// input ports
input  clk;       // system clock 20 MHz
input  [3:0] a;  // 1st input value
input  [3:0] b;  // 2nd input value
// output ports
output [3:0] result;
// internal variables
reg  [3:0] result;
reg  [2:0] delay;
reg  [3:0] reg1,reg2;
reg  [3:0] cnt;

//All variables on the left hand side of the always block must be declared as registers.In
//verilog registers represent data storage elements.

/// 4 bit division block   ///

//Whenever inputs in the sensitivity list changes the always block is executed.Statements
//within the always block are executed sequentially.

always @ (posedge clk)
begin
 case (delay)
          3'b000 : begin
                        reg1 <= a;       //store a[3:0] in reg1
                        reg2 <= b;       //store b[3:0] in reg2
                        delay <= 3'b001;
                        end
          3'b001 : begin
```

```
                    if (reg2 == 0)  // check b[3:0] is zero or not
                            delay <= 3'b011;
                    else
                            delay <= 3'b010;
                end
        3'b010 :    if (reg1 >= reg2)   // if reg1 >= reg2 ,do repeated subtraction and increment
                    begin               // cnt by 1 every time till if condition becomes false.
                      reg1 <= reg1 - reg2;
                      cnt  <= cnt + 1;
                      delay <= 3'b010;
                    end
                   else
                    begin
                      result <= cnt;     // store cnt  in result.
                      delay  <= 3'b100;
                    end
        3'b011 : begin
                    result <= 4'b0000; //  result is zero if reg2 or b equal to 0
                    delay <= 3'b100;
                 end
        3'b100 : begin
                    cnt <= 4'b0000;     // for new set of inputs reinitialise cnt to zero,
                    delay <= 3'b000;  //  get back to first state and  perform division.
                 end
        default :
                    delay <= 3'b000;
 endcase
end
endmodule
```

User Constraints file :

```
NET "clk"           LOC = "P184";
NET "a[3]"          LOC = "P136";
NET "a[2]"          LOC = "P142";
NET "a[1]"          LOC = "P148";
NET "a[0]"          LOC = "P154";
NET "b[3]"          LOC = "P159";
NET "b[2]"          LOC = "P169";
NET "b[1]"          LOC = "P194";
NET "b[0]"          LOC = "P174";
NET "result[3]"     LOC = "P123";
NET "result[2]"     LOC = "P126";
NET "result[1]"     LOC = "P127";
NET "result[0]"     LOC = "P128";
```

Experiment 2) VERILOG CODE FOR SWITCH LED INTERFACE IN DATAFLOW MODELLING

```
module swled(sw,led);//module declaration with input,output list
input  [16:1] sw;  //input declaration
output [16:1] led; //output declaration

//expression is evaluvated as soon as one of the operands in the RHS changes and
//assigned to the LHS net.
//Statements using the assign keyword are executed at the same time.

assign led = sw;

end module
```

User Constraints file :

```
NET "led[10]"        LOC = "p119"  ;
NET "led[11]"        LOC = "p120"  ;
NET "led[12]"        LOC = "p122"  ;
NET "led[13]"        LOC = "p123"  ;
NET "led[14]"        LOC = "p126"  ;
NET "led[15]"        LOC = "p127"  ;
NET "led[16]"        LOC = "p128"  ;
NET "led[1]"         LOC = "p102"  ;
NET "led[2]"         LOC = "p106"  ;
NET "led[3]"         LOC = "p107"  ;
NET "led[4]"         LOC = "p108"  ;
NET "led[5]"         LOC = "p109"  ;
NET "led[6]"         LOC = "p112"  ;
NET "led[7]"         LOC = "p113"  ;
NET "led[8]"         LOC = "p115"  ;
NET "led[9]"         LOC = "p116"  ;
NET "sw[10]"         LOC = "p142"  ;
NET "sw[11]"         LOC = "p148"  ;
NET "sw[12]"         LOC = "p154"  ;
NET "sw[13]"         LOC = "p159"  ;
NET "sw[14]"         LOC = "p169"  ;
NET "sw[15]"         LOC = "p194"  ;
NET "sw[16]"         LOC = "p174"  ;
NET "sw[1]"          LOC = "p71"  ;
NET "sw[2]"          LOC = "p72"  ;
NET "sw[3]"          LOC = "p91"  ;
NET "sw[4]"          LOC = "p101"  ;
```

```
NET "sw[5]"          LOC = "p110"  ;
NET "sw[6]"          LOC = "p118"  ;
NET "sw[7]"          LOC = "p124"  ;
NET "sw[8]"          LOC = "p130"  ;
NET "sw[9]"          LOC = "p136"  ;
```

Experiment 3) VERILOG CODE FOR MATRIX KEYPAD INTERFACE IN BEHAVIORAL MODELLING

```
////////////////////////////////////////////////////////////////////////////////
// Press the push buttons of the matrix keypad in the VPTB-10 board and observe
// the corresponding value in the seven segment display.
////////////////////////////////////////////////////////////////////////////////
module matrixkey(clk,a,b,dispcom,disp);//module declaration with io port list
//input ports
input clk;                    // system clock 20 MHz Default
input  [3:0] b;               // column matrix
//output ports
output [3:0] a;               // row matrix
output dispcom;               // for common cathode seven segment display
output [6:0] disp;            // seven segment for displaying value of matrix key
//internal variables
reg [3:0] a;
reg [6:0] disp = 7'b0000000;
reg [12:0] sig = 0;           // for delay

//All variables on the left hand side of the always block must be declared as registers.In verilog
//registers represent data storage elements.

//---------------- 4x4 MATRIX KEYPAD BLOCK ----------------//

//Whenever inputs in the sensitivity list changes the always block is executed.Statements within
//the always block are executed sequentially.

always @ (posedge clk)
begin
 sig <= sig + 1;
                case (sig[10:8])
                        3'b000 :
                                a <= 4'b0111; // --first row
                        3'b001 :
                                if (b == 4'b0111)
                                        disp <= 7'b1001111; // -- 3
                                else if (b == 4'b1011)
                                        disp <= 7'b1011011; // -- 2
```

```
                        else if (b == 4'b1101)
                                disp <= 7'b0000110; // -- 1
                        else if (b == 4'b1110)
                                disp <= 7'b0111111; // -- 0
                3'b010 :
                        a <= 4'b1011; // --second row
                3'b011 :
                        if (b == 4'b1110)
                                disp <= 7'b1100110; // -- 4
                        else if (b == 4'b1101)
                                disp <= 7'b1101101; // -- 5
                        else if (b == 4'b1011)
                                disp <= 7'b1111101; // -- 6
                        else if (b == 4'b0111)
                                disp <= 7'b0000111; // -- 7
                3'b100 :
                        a <= 4'b1101; // --third row
                3'b101 :
                        if (b == 4'b1110)
                                disp <= 7'b1111111; // -- 8
                        else if (b == 4'b1101)
                                disp <= 7'b1100111; // -- 9
                        else if (b == 4'b1011)
                                disp <= 7'b1110111; // -- A
                        else if (b == 4'b0111)
                                disp <= 7'b1111100; // -- b
                3'b110 :
                        a <= 4'b1110; // --fourth row
                3'b111 :
                        if (b == 4'b1110)
                                disp <= 7'b0111001; // -- C
                        else if (b == 4'b1101)
                                disp <= 7'b1011110; // -- d
                        else if (b == 4'b1011)
                                disp <= 7'b1111001; // -- E
                        else if (b == 4'b0111)
                                disp <= 7'b1110001; // -- F
                default :
                                disp <= 7'b0000000;
        endcase
end

assign dispcom = 1'b0;

endmodule
```

User Constraints file :

```
NET "clk"          LOC = "p184"  ;
NET "a[0]"         LOC = "p129" | PULLUP ;
NET "a[1]"         LOC = "p133" | PULLUP ;
NET "a[2]"         LOC = "p135" | PULLUP ;
NET "a[3]"         LOC = "p138" | PULLUP ;
NET "b[0]"         LOC = "p132" | PULLUP ;
NET "b[1]"         LOC = "p134" | PULLUP ;
NET "b[2]"         LOC = "p137" | PULLUP ;
NET "b[3]"         LOC = "p204" | PULLUP ;
NET "disp[0]"      LOC = "p179";
NET "disp[1]"      LOC = "p178";
NET "disp[2]"      LOC = "p144";
NET "disp[3]"      LOC = "p146";
NET "disp[4]"      LOC = "p147";
NET "disp[5]"      LOC = "p180";
NET "disp[6]"      LOC = "p181";
NET "dispcom"      LOC = "p145";
```

Experiment 4a) VERILOG CODE FOR BUZZER INTERFACE IN BEHAVIORAL MODELLING

```
module buzzer(clk,buzz); // module declaration
// input ports
input clk;
// output ports
output buzz;
// register ports for internal variables
reg [25:0] sig = 0;
reg buzz;

//All variables on the left hand side of the always block must be declared as registers.In verilog
//registers represent data storage elements.

// BUZZER  BLOCK //

//Whenever inputs in the sensitivity list changes the always block is executed.Statements within
//the always block are executed sequentially.

always @ (posedge clk)
begin
 sig <= sig + 1;
  case (sig[25:24])
                2'b00 :
```

```
                                buzz <= 1'b0;
                2'b10 :
                                buzz <= 1'b1;
 endcase
end
endmodule
```

User Constraints file :

```
NET "clk"      LOC    = "p184"        ;
NET "buzz"  LOC      = "p190" ;
```

Experiment 4b) VERILOG CODE FOR RELAY INTERFACE IN BEHAVIORAL MODELLING

```
module relay(clk,relay); // module declaration
// input  ports
input clk;        // System clock 20 MHz Default
// output ports
output relay;
// internal variables
reg [25:0] sig = 0;
reg relay;

//All variables on the left hand side of the always block must be declared as registers.In verilog
//registers represent data storage elements.

// RELAY BLOCK //

//Whenever inputs in the sensitivity list changes the always block is executed.Statements within
the
//the always block are executed sequentially.

always @ (posedge clk)
begin
 sig <= sig + 1;
        case (sig[25:24])
                2'b00 :
                                        relay <= 1'b0;
                2'b10 :
                                        relay <= 1'b1;
        endcase
end
endmodule
```

User Constraints file :

```
NET "clk"            LOC   = "p184";
NET "relay"          LOC   = "p193" ;
```

Experiment 5) VERILOG CODE FOR SEVEN SEGMENT LED INTERFACE IN BEHAVIORAL MODELLING

```
//////////////////////////////////////////////////////////////////////////////////
// VPTB-10 board :
// 5 sw : bin[3](msb) ; 6 sw : bin[2] ; 7 sw : bin[1] ; 8 sw : bin[0](lsb) ;
//////////////////////////////////////////////////////////////////////////////////
module sevenseg(bin,dispcom,disp); //module declaration
//input ports
input  [3:0] bin;    // Input binary value from 0 to 15
//output ports
output [6:0] disp;   // Seven segment display for displaying the hex value
output dispcom;
//internal variables
reg    [6:0] disp;

//**************  Seven segment display module  ***************//

always @ (bin) //when the input in the sensitivity list changes always block is
begin          //executed.
 case (bin)
                    4'b0000 : disp = 7'b0111111; // 0
                    4'b0001 : disp = 7'b0000110; // 1
                    4'b0010 : disp = 7'b1011011; // 2
                    4'b0011 : disp = 7'b1001111; // 3
                    4'b0100 : disp = 7'b1100110; // 4
                    4'b0101 : disp = 7'b1101101; // 5
                    4'b0110 : disp = 7'b1111101; // 6
                    4'b0111 : disp = 7'b0000111; // 7
                    4'b1000 : disp = 7'b1111111; // 8
                    4'b1001 : disp = 7'b1100111; // 9
                    4'b1010 : disp = 7'b1110111; // a
                    4'b1011 : disp = 7'b1111100; // b
                    4'b1100 : disp = 7'b0111001; // c
                    4'b1101 : disp = 7'b1011110; // d
                    4'b1110 : disp = 7'b1111001; // e
                    4'b1111 : disp = 7'b1110001; // f
                    default : disp = 7'b0000000;
 endcase
end
```

```
assign dispcom = 1'b0; // For common cathode seven segment display

endmodule
```

User Constraints File :

```
NET "bin[3]"        LOC = "p71";
NET "bin[2]"        LOC = "p72";
NET "bin[1]"        LOC = "p91";
NET "bin[0]"        LOC = "p101";
NET "disp[0]"       LOC = "p179";
NET "disp[1]"       LOC = "p178";
NET "disp[2]"       LOC = "p144";
NET "disp[3]"       LOC = "p146";
NET "disp[4]"       LOC = "p147";
NET "disp[5]"       LOC = "p180";
NET "disp[6]"       LOC = "p181";
NET "dispcom"       LOC = "p145";
```

EXPERIMENT 6A) VERILOG CODE FOR CLOCKWISE ROTATION OF STEPPER MOTOR

```
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// FOR ON BOARD :
// Connect the 5 pin RMC cable of the stepper motor to the 5 pin
// connector P10(VPTB-10) and the jumper J13 of VPTB-10 must be closed in +5V
// position.Observe the clockwise rotation of the stepper motor by rolling
// a small bit of paper on its top.
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// FOR VVSI -32 :
// Connect external io connector P6 (VPTB-10) to the external io connector P3
// (VVSI-32).Connect the 5 pin RMC cable of the stepper motor to the 5 pin
// connector P4(VVSI-32) and the jumpers J5 +5V of VPTB-10 & J2[VCC MVCC] of VVSI-32
// must be closed.Observe the clockwise rotation of the stepper motor by rolling
// a small bit of paper on its top.
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
module stepperclk(clk,db); // module declaration
// input declaration
input clk;              // system clock 20 MHz
// output declaration
output [4:1] db;        // to stepper motor data bus
// internal variables
reg [4:1] db;
reg [20:1] sig = 0;
```

```
//*************          stepper motor block    *****************//
always @ (posedge clk)
begin
  sig <= sig + 1;
        case (sig[20:18])  // Set your Delay Time
                3'b000 :          //--      FOR CLOCKWISE DIRECTION     --
                        db <= 4'b1001;  // 1st Data
                3'b010 :
                        db <= 4'b0101;   // 2nd Data
                3'b100 :
                        db <= 4'b0110;   // 3rd Data
                3'b110 :
                        db <= 4'b1010;   // 4th Data
        endcase
end
endmodule
```

User Constraints File :

For ON BOARD :

```
NET "clk"      LOC = "P184"  ;
NET "db[1]"   LOC = "P76"  ;
NET "db[2]"   LOC = "P77"  ;
NET "db[3]"   LOC = "P78"  ;
NET "db[4]"   LOC = "P82"  ;
```

For VVSI-32 :

```
NET "clk"      LOC = "P184"  ;
NET "db[1]"   LOC = "P5"  ;#5 pin of ext ioc P6
NET "db[2]"   LOC = "P4"  ;#4 pin of ext ioc P6
NET "db[3]"   LOC = "P2"  ;#2 pin of ext ioc P6
NET "db[4]"   LOC = "P3"  ;#3 pin of ext ioc P6
```

EXPERIMENT 6B) VERILOG CODE FOR ANTI-CLOCKWISE ROTATION OF STEPPER MOTOR

```
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// FOR ON BOARD:
// Connect the 5 pin RMC cable of the stepper motor to the 5 pin
// connector P10(VPTB-10) and the jumper J13 of VPTB-10 must be closed in +5V
// position.Observe the anti-clockwise rotation of the stepper motor by rolling
// a small bit of paper on its top.
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// FOR VVSI-32:
// Connect external io connector P6 (VPTB-10) to the external io connector P3
// (VVSI-32).Connect the 5 pin RMC cable of the stepper motor to the 5 pin
// connector P4(VVSI-32) and the jumpers J5 +5V of VPTB-10 & J2[VCC MVCC] of VVSI-32
// must be closed.Observe the anti-clockwise rotation of the stepper motor by rolling
// a small bit of paper on its top.
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module stepperanti(clk,db); // module declaration
// input declaration
input clk;                  // system clock 20 MHz
// output declaration
output [4:1] db;            // to stepper motor data bus
// internal variables
reg [4:1] db;
reg [20:1] sig = 0;

//*************                stepper motor block    *****************//
always @ (posedge clk)
begin
  sig <= sig + 1;
        case (sig[20:18])  // Set your Delay Time
                3'b000 :  //-- FOR ANTICLOCKWISE DIRECTION     --
                        db <= 4'b1010;   // 1st Data
                3'b010 :
                        db <= 4'b0110;   // 2nd Data
                3'b100 :
                        db <= 4'b0101;   // 3rd Data
                3'b110 :
                        db <= 4'b1001;   // 4th Data
        endcase
end
endmodule
```

User Constraints File :

For ON BOARD :

```
NET "clk"      LOC = "P184"  ;
NET "db[1]"   LOC = "P76"  ;
NET "db[2]"   LOC = "P77"  ;
NET "db[3]"   LOC = "P78"  ;
NET "db[4]"   LOC = "P82"  ;
```

For VVSI-32 :

```
NET "clk"      LOC = "P184"  ;
NET "db[1]"   LOC = "P5"  ;#5 pin of ext ioc P6
NET "db[2]"   LOC = "P4"  ;#4 pin of ext ioc P6
NET "db[3]"   LOC = "P2"  ;#2 pin of ext ioc P6
NET "db[4]"   LOC = "P3"  ;#3 pin of ext ioc P6
```

Experiment 7) VERILOG CODE FOR TRAFFIC LIGHT CONTROL

```
//////////////////////////////////////////////////////////////////////////////////
// SYSTEM CLOCK FREQUENCY IS 20 MHZ (Default)
// Connect VLIM connector P8 (VPTB-10) and VLIM connector (TLC) using 26 pin FRC
// cable.In VPTB-10:
// SW34 - rst ;
//////////////////////////////////////////////////////////////////////////////////
module traffic(clk, rst, dir, p1, p2, p3, p4, pt); //module declaration
// input ports
input clk;                                  // system clock frequency is 20 mhz
input rst;                                  // for reset
// output ports
output [2:0] dir;
output [4:0] p1;                            // d5,d4,d3,d2,d1
output [4:0] p2;                            // d10,d9,d8,d7,d6
output [4:0] p3;                            // d15,d14,d13,d12,d11
output [4:0] p4;                            // d20,d19,d18,d17,d16
output [3:0] pt;                            // dl1,dl2,dl3,dl4,dl5,dl6,dl7,dl8 (Pedestrain)
// registered ports
reg [4:0] p1;
reg [4:0] p2;
reg [4:0] p3;
reg [4:0] p4;
reg [3:0] pt;
reg [30:0] sig;
```

```
//************ TRAFFIC LIGHT CONTROLLLER BLOCK ***************//

always @ (posedge clk or negedge rst)
begin
if (rst == 1'b0)          //when reset switch is zero TLC is reset.
        begin
                p1 <= 5'b00100;         //at reset condition
                p2 <= 5'b00100;
                p3 <= 5'b00100;
                p4 <= 5'b00100;
                pt <= 4'b1111;
                sig <= 8'h00000000;
        end
else
        begin
          sig <= sig + 1;
            case (sig[29:24])
                6'b000000 : begin
                        p1 <= 5'b10011;                 // d5,d4,d3,d2,d1       (Green)
                        p2 <= 5'b00100;
                        p3 <= 5'b00100;
                        p4 <= 5'b00100;
                        pt <= 4'b1111;
                        end
                6'b000100 : begin
                         p1 <= 5'b01000;                 //Yellow
                         p2 <= 5'b00100;
                         p3 <= 5'b00100;
                         p4 <= 5'b00100;
                         pt <= 4'b1111;
                         end
                6'b001000 : begin
                         p1 <= 5'b00100;                 // d10,d9,d8,d7,d6
                         p2 <= 5'b10011;                 //Green
                         p3 <= 5'b00100;
                         p4 <= 5'b00100;
                         pt <= 4'b1111;
                         end
                6'b001100 : begin
                         p1 <= 5'b00100;
                         p2 <= 5'b01000;                 //Yellow
                         p3 <= 5'b00100;
                         p4 <= 5'b00100;
                         pt <= 4'b1111;
                         end
```

```
                6'b010000 : begin
                        p1 <= 5'b00100;             // d15,d14,d13,d12,d11
                        p2 <= 5'b00100;
                        p3 <= 5'b10011;             //Green
                        p4 <= 5'b00100;
                        pt <= 4'b1111;
                        end
                6'b010100 : begin
                        p1 <= 5'b00100;
                        p2 <= 5'b00100;
                        p3 <= 5'b01000;             //yellow
                        p4 <= 5'b00100;
                        pt <= 4'b1111;
                        end
                6'b011000 : begin
                        p1 <= 5'b00100;             // d20,d19,d18,d17,d16
                        p2 <= 5'b00100;
                        p3 <= 5'b00100;
                        p4 <= 5'b10011;             //Green
                        pt <= 4'b1111;
                        end
                6'b011100 : begin
                        p1 <= 5'b00100;
                        p2 <= 5'b00100;
                        p3 <= 5'b00100;
                        p4 <= 5'b01000;             //Yellow
                        pt <= 4'b1111;
                        end
                6'b100000 : begin
                        p1 <= 5'b00100;             // dl1,dl2,dl3,dl4,dl5,dl6,dl7,dl8
                        p2 <= 5'b00100;
                        p3 <= 5'b00100;
                        p4 <= 5'b00100;
                        pt <= 4'b0000;              //Pedestrain
                        end
                6'b100100 : sig <= 8'h00000000;
                        default : begin
                        end
            endcase
        end
end

assign dir = 3'b111;

endmodule
```

User Constraints File :

```
NET "clk"      LOC = "p184"  ;
NET "rst"      LOC = "p175"  ;
NET "p1[0]"   LOC = "p31"  ;#pao
NET "p1[1]"   LOC = "p33"  ;#pa1
NET "p1[2]"   LOC = "p34"  ;#pa2
NET "p1[3]"   LOC = "p35"  ;#pa3
NET "p1[4]"   LOC = "p36"  ;#pa4
NET "p2[0]"   LOC = "p39"  ;#pa5
NET "p2[1]"   LOC = "p40"  ;#pa6
NET "p2[2]"   LOC = "p41"  ;#pa7
NET "p2[3]"   LOC = "p75"  ;#pc0
NET "p2[4]"   LOC = "p74"  ;#pc1
NET "p3[0]"   LOC = "p69"  ;#pc2
NET "p3[1]"   LOC = "p68"  ;#pc3
NET "p3[2]"   LOC = "p65"  ;#pc4
NET "p3[3]"   LOC = "p64"  ;#pc5
NET "p3[4]"   LOC = "p63"  ;#pc6
NET "p4[0]"   LOC = "p62"  ;#pc7
NET "p4[1]"   LOC = "p50"  ;#pb4
NET "p4[2]"   LOC = "p55"  ;#pb5
NET "p4[3]"   LOC = "p56"  ;#pb6
NET "p4[4]"   LOC = "p60"  ;#pb7
NET "pt[0]"    LOC = "p45"  ;#pb0
NET "pt[1]"    LOC = "p47"  ;#pb1
NET "pt[2]"    LOC = "p48"  ;#pb2
NET "pt[3]"    LOC = "p49"  ;#pb3
NET "dir[2]"   LOC = "p30"  ;
NET "dir[1]"   LOC = "p42"  ;
NET "dir[0]"   LOC = "p61"  ;
```

Experiment 8) VERILOG CODE FOR 4 BIT UP/DOWN COUNTER IN BEHAVIORAL MODELLING

```
//////////////////////////////////////////////////////////////////////////////////
// In VPTB-10 board :
// 1  sw :  dir ; SW34 : rst ;
// 13 led : led[3] (msb) ; 14 led : led[2] ; 15 led : led[1] ; 16 led : led[0] (lsb) ;
//////////////////////////////////////////////////////////////////////////////////
module counter4b(clk,rst,dir,led);//module declaration
//----------Output Ports-------------
output [3:0] led;  // counter output
//----------Input Ports--------------
input  clk, rst ,dir;// clock,reset & direction inputs of counter
```

```
//---------Internal Variables--------
reg [28:0] cntr = 0;
reg [3:0]  led;

//All variables on the left hand side of the always block must be declared as registers
//using the reg keyword.In verilog register represent data storage elements.

//----------------- 4 BIT UP/DOWN COUNTER BLOCK -----------------//

//Whenever the input in the sensitivity list changes,the statements in the always block
//are executed.

always @ (posedge clk)
begin
  if (rst == 1'b0)                  // when rst switch is low,counter is reset.
                  begin
                   cntr <= 0;
                   led  <= cntr[28:25];
                  end
  else
                  begin
                           if (dir)                          // when dir is high
                             begin
                                  cntr <= cntr + 1;         // up count
                                  led <= cntr[28:25];
                              end
                            else                            // when dir is low
                             begin
                                  cntr <= cntr - 1          ;  // down count
                                  led <= cntr[28:25];
                              end
                  end
 end

endmodule
```

User Constraints File :

```
NET "clk"    LOC = "P184";
NET "rst"    LOC = "P175";
NET "dir"    LOC = "P71";
NET "led[3]" LOC = "P123";
NET "led[2]" LOC = "P126";
NET "led[1]" LOC = "P127";
NET "led[0]" LOC = "P128";
```

VERILOG CODE FOR SIMULATION OF 4 BIT UP/DOWN COUNTER IN BEHAVIORAL MODELLING

```
module counter4b(clk,rst,dir,led);//module declaration
//----------Output Ports-------------
output [3:0] led;  // counter output
//----------Input Ports--------------
input  clk, rst ,dir;// clock,reset & direction inputs of counter
//---------Internal Variables--------
reg [3:0] cntr = 0;
reg [3:0]  led;

//All variables on the left hand side of the always block must be declared as registers
//using the reg keyword.In verilog register represent data storage elements.

//----------------- 4 BIT UP/DOWN COUNTER BLOCK -----------------//

//Whenever the input in the sensitivity list changes,the statements in the always block
//are executed.

always @ (posedge clk)
begin
  if (rst == 1'b0)                  // when rst switch is low, counter is reset.
                  begin
                   cntr <= 0;
                   led  <= cntr[3:0];
                  end
  else
                  begin
                           if (dir)          // when dir is high
                                   begin
                                           cntr <= cntr + 1;         // up count
                                           led <= cntr[3:0];
                                   end
                           else               // when dir is low
                                   begin
                                           cntr <= cntr - 1          ;  // down count
                                           led <= cntr[3:0];
                                   end
                  end
end

endmodule
```

EXPERIMENT 9) VERILOG CODE FOR DISPLAYING TEXT MESSAGE "VI MICROSYSTEMS" IN LCD

```
module lcddisp(clk,rs,cs,d); //module port declaration
// input ports
input  clk;              // system clock 20 MHz default
// output ports
output rs;               // register select
output cs;               // chip select
output [7:0]d;           // data bus lines
// internal variables
reg [3:0] i = 0 ;
reg [3:0] j = 0 ;
reg [25:0] sig;
reg [7:0] cmd [0:4] ; // array for storing lcd initialisation commands
reg [7:0] data[0:14]; // array for storing text "VI MICROSYSTEMS"

reg rs;
reg cs;
reg [7:0] d;

//All variables on the LHS of the always block must be declared as register using reg keyword.
//In verilog register represents data storage elements.

//-------------------- LCD DISPLAY BLOCK --------------------//

//When the input in the sensitivity list changes the always block is executed.Statements
//within always block are executed sequentially.

always @ (posedge clk)

  begin

  cmd[0]      <= 8'h38;  //lcd initialisation commands
  cmd[1]      <= 8'h06;
  cmd[2]      <= 8'h01;
  cmd[3]      <= 8'h0f;
  cmd[4]      <= 8'h80;

  data[0]     <= 8'h56;  // V
  data[1]     <= 8'h49;  // I
  data[2]     <= 8'h20;  //
  data[3]     <= 8'h4d;  // M
  data[4]     <= 8'h49;  // I
  data[5]     <= 8'h43;  // C
```

```
data[6]      <= 8'h52;  // R
data[7]      <= 8'h4f;  // O
data[8]      <= 8'h53;  // S
data[9]      <= 8'h59;  // Y
data[10]     <= 8'h53;  // S
data[11]     <= 8'h54;  // T
data[12]     <= 8'h45;  // E
data[13]     <= 8'h4d;  // M
data[14]     <= 8'h53;  // S

sig <= sig + 1;
        case (sig[25:20])
            6'b000000 :
                        rs   <= 1'b0;
            6'b000001 :
                        d <= cmd[i];                        // lcd initialisation
             6'b000010 :
                        cs   <= 1'b1;
             6'b000011 : begin
                        cs   <= 1'b0;
                        i    <= i + 1;
                        if (i < 4)
                                sig[25:20] <= 6'b000001;
                        else
                          begin
                                sig[25:20] <= 6'b000100;
                                j <= 0;
                         end
                      end
            6'b000100 :
                        rs   <= 1'b1;
             6'b000101 :
                        d    <= data[j];                    // "VI MICROSYSTEMS"
             6'b000110 :
                        cs   <= 1'b1;
             6'b000111 : begin
                        cs   <= 1'b0;
                        j    <= j + 1;
                        if (j < 14)
                                sig[25:20] <= 6'b000101;
                        else
                         begin
                                sig[25:20] <= 6'b000000;
                                i <= 0;
                        end
```

```
                    end
        default :   sig <= 0;
       endcase
end
endmodule
```

User Constraints File :

```
NET "clk"     LOC = "p184"  ;
NET "rs"      LOC = "p83"  ;
NET "cs"      LOC = "p89"  ;
NET "d[0]"    LOC = "p90"  ;
NET "d[1]"    LOC = "p93"  ;
NET "d[2]"    LOC = "p94"  ;
NET "d[3]"    LOC = "p96"  ;
NET "d[4]"    LOC = "p97"  ;
NET "d[5]"    LOC = "p98"  ;
NET "d[6]"    LOC = "p99"  ;
NET "d[7]"    LOC = "p100"  ;
```

EXPERIMENT 10) VERILOG CODE TO GENERATE PWM SIGNAL FOR DC MOTOR CONTROL

```
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// FOR ON BOARD :
// The jumpers J1 (VPTB-10) must be shorted in +5V position.Connect the 2 pin RMC cable
// of the DC motor to the 2 pin RMC connectors P1(VPTB-10). Press sw14 of the matrix
// keypad for increasing the duty cycle and sw15 of the keypad for decreasing the duty
// cycle of the dc motor.
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// FOR VVSI - 32 :
// Connect the external io connector P6 (VPTB-10) to the external io connector P3
// (VVSI-32).The jumpers J5 +5V (VPTB-10) and J1 [MVCC VCC](VVSI-32) must be shorted.
// Connect the 2 pin RMC cable of the DC motor to the 2 pin RMC connectors P5(VVSI- 32).
// Press sw13 of the matrix keypad for increasing the duty cycle and sw14 of the keypad
// for decreasing the duty cycle of the dc motor.
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module dcmotor(clk,sw1,sw2,pulse,dc_out,a); //module declaration
// input ports
input clk;                          //System clock 20 MHz Default
input sw1,sw2;
// output ports
output pulse;
output dc_out;
```

```
output [3:0] a;
// ---- internal variables ---- //
reg pulse;

reg [12:0] cnt1 = 200;
reg [7:0] cnt = 10;
reg [12:0] h;
reg [20:0] f;

always @ (posedge clk)
begin
  h <= h + 1;
 if (h < cnt1)
          pulse <= 1'b1;
 else if ((h > cnt1) && (h < 2000))
          pulse <= 1'b0;
 else     if (h > 2000)
          h <= 0;
end

always @ (posedge clk)
begin
 if (sw1 == 1'b0)
  begin
          f <= f + 1;
                  if (f > 1000000)
                    begin
                            cnt <= cnt + 1;
                            f           <= 0;
                    end
                    if (cnt >= 90)
                                    cnt <= 90;
  end
///-------------------------------------
 if (sw2 == 1'b0)
  begin
          f <= f + 1;
                  if (f > 1000000)
                    begin
                            cnt <= cnt - 1;
                            f           <= 0;
                    end
                    if (cnt <= 10)
                                    cnt <= 10;
  end
```

```
 cnt1 <= cnt * 20;
end

assign a          = 4'b0111;
assign dc_out = 1'b0;

endmodule
```

User Constraints File :

For ON BOARD :

```
NET "clk"      LOC = "P184";
NET "sw1"      LOC = "p132" | PULLUP ;
NET "sw2"      LOC = "p134" | PULLUP ;
NET "a[0]"     LOC = "p129" | PULLUP ;
NET "a[1]"     LOC = "p133" | PULLUP ;
NET "a[2]"     LOC = "p135" | PULLUP ;
NET "a[3]"     LOC = "p138" | PULLUP ;
NET "pulse"    LOC = "P139";
NET "dc_out" LOC = "P206";
```

For VVSI-32 :

```
NET "clk"      LOC = "P184";
NET "sw1"      LOC = "p132" | PULLUP ;
NET "sw2"      LOC = "p134" | PULLUP ;
NET "a[0]"     LOC = "p129" | PULLUP ;
NET "a[1]"     LOC = "p133" | PULLUP ;
NET "a[2]"     LOC = "p135" | PULLUP ;
NET "a[3]"     LOC = "p138" | PULLUP ;
NET "pulse"    LOC = "P9";
NET "dc_out" LOC = "P8";
```

EXPERIMENT 11) VERILOG CODE FOR 4:1 MULTIPLEXER IN BEHAVIORAL MODELLING

```
////////////////////////////////////////////////////////////////////////////////
// MULTIPLEXER  Many input One output ( 2^N inputs ,N select inputs,1 output ) --
// 8x1 MULTIPLEXER -- 4 input lines,2 select lines, 1 output line --
// In VPTB-10 board the first 2 switches are for the select lines the next 4 switches
// are the input lines of the multiplexer and the 1st LED is the output line
// of the multiplexer.
// In VPTB-10 board :
// 1 sw  : sel[1] ; 2 sw  : sel[0] ;
// 3 sw  : inp[0] ; 4 sw  : inp[1] ; 5 sw  : inp[2] ; 6 sw  : inp[3] ;
// 1 led : muxout ;
////////////////////////////////////////////////////////////////////////////////
module mux4x1(muxout,inp,sel);//module declaration with list of input,outputs
//output ports
output muxout; // mux output line
//input ports
input [3:0] inp; // mux input lines
input [1:0] sel; // select lines
//internal variables
reg    muxout;          //register declaration

//All variables on the LHS of the always block must be declared as register using keyword reg
//In verilog register represents data storage elements.

//---------------------- 4:1 MULTIPLEXER BLOCK -----------------------//

//When the inputs in the senstivity list changes the always block is executed.Statements
//within always block are executed sequentially.

always @ (sel or inp)
begin
        case(sel)
                2'b00 :
                        muxout = inp[0];  // 1 input assigned to muxout for select value "00"
                2'b01 :
                        muxout = inp[1];  // 2 input assigned to muxout for select value "01"
                2'b10 :
                        muxout = inp[2];  // 3 input assigned to muxout for select value "10"
                2'b11 :
                        muxout = inp[3];  // 4 input assigned to muxout for select value "11"
                default :
                        muxout = 1'b0;
        endcase
```

```
end
endmodule
```

User Constraints File :

```
NET "sel[1]"        LOC = "P71";
NET "sel[0]"        LOC = "P72";
NET "inp[0]"        LOC = "P91";
NET "inp[1]"        LOC = "P101";
NET "inp[2]"        LOC = "P110";
NET "inp[3]"        LOC = "P118";
NET "muxout"        LOC = "P102";
```

EXPERIMENT 11) VERILOG CODE FOR 1:4 DEMULTIPLEXER IN BEHAVIORAL MODELLING

```
//////////////////////////////////////////////////////////////////////////////////
// DEMULTIPLEXER -- One input many outputs (1 input , 2^N outputs ,N select input)
// 1X4 DEMULTIPLEXER -- 1 input line, 2 select inputs, 4 output lines--
// In VPTB-10 board :
// 1 sw  : dmuxin ; 2 sw  : sel[1] ; 3 sw  : sel[0] ;
// 1 led : oup[0] ; 2 led : oup[1] ; 3 led : oup[2] ; 4 led : oup[3] ;
//////////////////////////////////////////////////////////////////////////////////
module demux1x4(dmuxin,sel,oup);// module declaration
//input ports
input dmuxin;           //  demux input line
input [1:0] sel;    //  select input lines
//output ports
output [3:0] oup;       //  demux output lines
//internal variables
reg     [3:0] oup;      //  register declaration

//All variables on the LHS of the always block must be declared as register using the
//reg keyword. In verilog register represents data storage elements.

//---------------- 1:4 DEMULTIPLEXER BLOCK ----------------//

//When anyone of the inputs in the sensitivity list changes the always block is executed.
//Statements within always block are executed sequentially.

always @ (sel or dmuxin)
begin
case (sel)
 2'b00 :
            begin
```

```
                oup[0] = dmuxin;      // demux input assigned to oup[0] for select value "00"
                oup[1] = 1'b0;
                oup[2] = 1'b0;
                oup[3] = 1'b0;
            end

2'b01 :   begin
                oup[0] = 1'b0;
                oup[1] = dmuxin;      // demux input assigned to oup[1] for select value "01"
                oup[2] = 1'b0;
                oup[3] = 1'b0;
            end

2'b10 :   begin
                oup[0] = 1'b0;
                oup[1] = 1'b0;
                oup[2] = dmuxin;      // demux input assigned to oup[2] for select value "10"
                oup[3] = 1'b0;
            end

2'b11 :  begin
                oup[0] = 1'b0;
                oup[1] = 1'b0;
                oup[2] = 1'b0;
                oup[3] = dmuxin;      // demux input assigned to oup[3] for select value "11"
           end

default : begin                        // default statement is optional.
                oup[0] = 1'b0;
                oup[1] = 1'b0;
                oup[2] = 1'b0;
                oup[3] = 1'b0;         // for default case all o/p lines remain at 0.
            end
endcase
end
endmodule
```

User Constraints File :

```
NET "dmuxin"        LOC = "P71";
NET "sel[1]"        LOC = "P72";
NET "sel[0]"        LOC = "P91";
NET "oup[0]"        LOC = "P102";
NET "oup[1]"        LOC = "P106";
NET "oup[2]"        LOC = "P107";
NET "oup[3]"        LOC = "P108";
```

Experiment 12) VERILOG CODE FOR 3:8 DECODER IN BEHAVIORAL MODELLING

```
//////////////////////////////////////////////////////////////////////////////////
// DECODER --  n INPUT 2^n OUTPUT --
// 3:8 DECODER -- 3 INPUT 8 OUPUT --
// In VPTB-10 board :
// 14 sw  : inp[2] ; 15 sw : inp[1] ; 16 sw : inp[0]  ;
// 9  led : oup[7] ; 10 led : oup[6] ; 11 led : oup[5] ; 12 led : oup[4] ;
// 13 led : oup[3] ; 14 led : oup[2] ; 15 led : oup[1] ; 16 led : oup[0] ;
//////////////////////////////////////////////////////////////////////////////////

module decdr3to8(oup,inp);//module declaration with i/p ,o/p list
//input ports
output [7:0] oup;//decoder output lines
//output ports
input  [2:0] inp;//decoder input  lines
//internal variables
reg    [7:0] oup;//register declaration

//All variables on the left hand side of the always must be declared as register.
//In verilog register represent data storage elements.

//---------------------- 3:8 DECODER BLOCK ----------------------//

//Whenever inputs in the sensitivity list changes,the statements within the always block are
//executed sequentially.

always @ (inp)
begin
        case (inp)      // Output value is assigned with respect to the input as per truth table
                        // of the decoder.
                3'b000 : oup = 8'b00000001;
                3'b001 : oup = 8'b00000010;
                3'b010 : oup = 8'b00000100;
                3'b011 : oup = 8'b00001000;
```

```
                3'b100 : oup = 8'b00010000;
                3'b101 : oup = 8'b00100000;
                3'b110 : oup = 8'b01000000;
                3'b111 : oup = 8'b10000000;
                default: oup = 8'b00000000;  //default statement is optional
        endcase
end
endmodule
```

User Constraints File :

```
NET "inp[2]"        LOC = "p169";
NET "inp[1]"        LOC = "p194";
NET "inp[0]"        LOC = "p174";
NET "oup[7]"        LOC = "p116";
NET "oup[6]"        LOC = "p119";
NET "oup[5]"        LOC = "p120";
NET "oup[4]"        LOC = "p122";
NET "oup[3]"        LOC = "p123";
NET "oup[2]"        LOC = "p126";
NET "oup[1]"        LOC = "p127";
NET "oup[0]"        LOC = "p128";
```

Experiment 12) VERILOG CODE FOR 8:3 ENCODER IN BEHAVIORAL MODELLING

```
//////////////////////////////////////////////////////////////////////////////////
// ENCODER -- 2^n INPUT n OUTPUT --
// 8:3 ENCODER -- 8 INPUT 3 OUPUT --
//////////////////////////////////////////////////////////////////////////////////
// In VPTB-10 board :
// 9  sw  : inp[7] ; 10 sw : inp[6] ; 11 sw : inp[5] ; 12 sw : inp[4] ;
// 13 sw  : inp[3] ; 14 sw : inp[2] ; 15 sw : inp[1] ; 16 sw : inp[0] ;
// 14 led : oup[2] ; 15 led : oup[1] ; 16 led : oup[0] ;
//////////////////////////////////////////////////////////////////////////////////
module encdr8to3(inp,oup);//module declaration with i/p ,o/p list
//input ports
input  [7:0] inp;         // encoder input lines
//output ports
output [2:0] oup;         // encoder output lines
//registered ports
reg       [2:0] oup;      // register declaration

//All variables on the LHS of the always block must be declared as register using reg
//keyword.In verilog register represents data storage elements.
```

```
//----------------- 8:3 ENCODER BLOCK -----------------//

//Whenever inputs in the sensitivity list changes, statements within the always block are
//executed sequentially.

always @ (inp)
begin
// i/p is given through the switches inp[7] to inp[0] & output is assigned as per
// the truth table of encoder
case (inp)
  8'b00000001 :       oup = 3'b000;
  8'b00000010 :       oup = 3'b001;
  8'b00000100 :       oup = 3'b010;
  8'b00001000 :       oup = 3'b011;
  8'b00010000 :       oup = 3'b100;
  8'b00100000 :       oup = 3'b101;
  8'b01000000 :       oup = 3'b110;
  8'b10000000 :       oup = 3'b111;
  default         :   oup = 3'b000;
endcase
end
endmodule
```

User Constraints File :

```
NET "inp[7]"        LOC = "p136";
NET "inp[6]"        LOC = "p142";
NET "inp[5]"        LOC = "p148";
NET "inp[4]"        LOC = "p154";
NET "inp[3]"        LOC = "p159";
NET "inp[2]"        LOC = "p169";
NET "inp[1]"        LOC = "p194";
NET "inp[0]"        LOC = "p174";
NET "oup[2]"        LOC = "p126";
NET "oup[1]"        LOC = "p127";
NET "oup[0]"        LOC = "p128";
```

EXPERIMENT 12) VERILOG CODE FOR 4 BIT SERIAL IN PARALLEL OUT SHIFT REGISTER IN BEHAVIORAL MODELLING

```
//////////////////////////////////////////////////////////////////////////////////
// In VPTB-10 board :
// SW34 : rst ; 1 sw : si ;
// 13 led : po[3] ; 14 led : po[2] ; 15 led : po[1] ; 16 led : po[0] ;
//////////////////////////////////////////////////////////////////////////////////
module shiftreg(clk,rst,si,po);//module declaration with io port list
//input ports
input  clk;       // system clock 20 MHz
input  rst;       // reset
input  si;        // serial input
//output ports
output [3:0] po;  // shift register o/p
//internal variables
reg clk_gen;
reg [23:0] i=0;
reg [3:0] poreg;

//All variables on the left hand side of the always must be declared as register.
//In verilog register represent data storage elements.

//----------------- slow clk generation block -----------------//

//When the input in the sensitivity list changes the always block is executed.Statements
//within always block are executed sequentially.

always @ (posedge clk)
 begin
  i <= i + 1;
  if ( i == 10000000)
   begin
                clk_gen <= ~clk_gen;
                i       <= 0;
   end
  else
                clk_gen <= clk_gen;
 end

//-----------shift register block-----------//

always @(posedge clk_gen)

  if (rst == 1'b0)                  // When rst sw is zero ,shift register is reset.Otherwise shifts in
```

```
        poreg <= 4'b0000 ; // serial data input.
 else
        poreg <= {poreg[2:0],si};

assign po = poreg;

endmodule
```

User Constraints File :

```
NET "clk"          LOC = "p184";
NET "rst"          LOC = "p175";
NET "si"           LOC = "p71";
NET "po[3]"        LOC = "p123";
NET "po[2]"        LOC = "p126";
NET "po[1]"        LOC = "p127";
NET "po[0]"        LOC = "p128";
```