PREFACE

About This Manual

This manual gives operational details for all the interfaces.

Manual Contents

This manual contains following chapters:

CHAPTER 1
Introduction

SPARTAN-3 based VTU ProtoBoard **(MXUNIV-MB-0208-003-VTU-KIT)** provides easy to use development platform, useful to physically verify DSP algorithms or simple digital designs around SPARTAN -3 FPGA.

Features

Figure 1 shows the SPARTAN-3 ProtoBoard, which includes the following components and features:

- **SPARTAN -3 FPGA :** 400 k logic cell SPARTAN -3 FPGA in PQ208 Plastic Quad Flat Package (MXS3FK-004-DSP)
  - Three families Spartan 3 /Spartan 3L/Spartan 3 XA.
  - Very low cost, high-performance logic solution for high-volume, consumer-oriented applications.
    - Densities as high as 74,880 logic cells.
    - Three power rails for core (1.2V), I/O's (1.2V to 3.3V) and Auxiliary purposes (2.5V).
    - 326 MHz system clock rate.
    - 90 nm process technology.
  - Select IO™ Signaling.
    - Up to 784 I/O pins.
    - 622 Mb/s data transfer rate per IO.
    - 18 single-ended signal standards.
    - 6 differential I/O standards including LVDS, RSDS.
    - Termination by Digitally Controlled Impedance.
    - Double data Rate (DDR) support.
  - Logic Resources
    - Abundant Logic cells with shift register capability.
    - Wide Multiplexers.
    - Fast look-ahead carry logic.
    - Dedicated 18 x 18 Multipliers.
  - SelectRAM ™ Hierarchical Memory.
    - Up to 1,872 Kbits of total block RAM.
    - Up to 520 Kbits of Distributed RAM.
  - Digital Clock Manager (up to 4DCMs)
    - Clock skew elimination.
    - Frequency synthesis
    - High resolution phase shifting.
  - Eight global clock lines and abundant routing.
  - Micro Blaze™ processor, PCI and other cores.
- **Analog Interface**: – 12 bit AD7891 ADC and 12 bit AD7541 DAC.
  - Analog Input – Four channels using ADC using AD7891, (500Ksps, 12 bit).
  - Additional Stereo Jacks are provided for Audio Input and Audio Output.
  - Thermister interface is given to ADC channel 5.
  - Analog Output- Four channels using four DAC's-AD7541. (12 bit, 100 ns conversion time).
- **Function Generator** (using IC 8038)
  - Provides Sine, Square and Triangular waveforms outputs.
  - Frequency variable from 60-200 KHz.
  - One Anti-aliasing filter at the input of Analog to Digital converter.
  - One Reconstruction filter at the output of Digital to Analog converter.

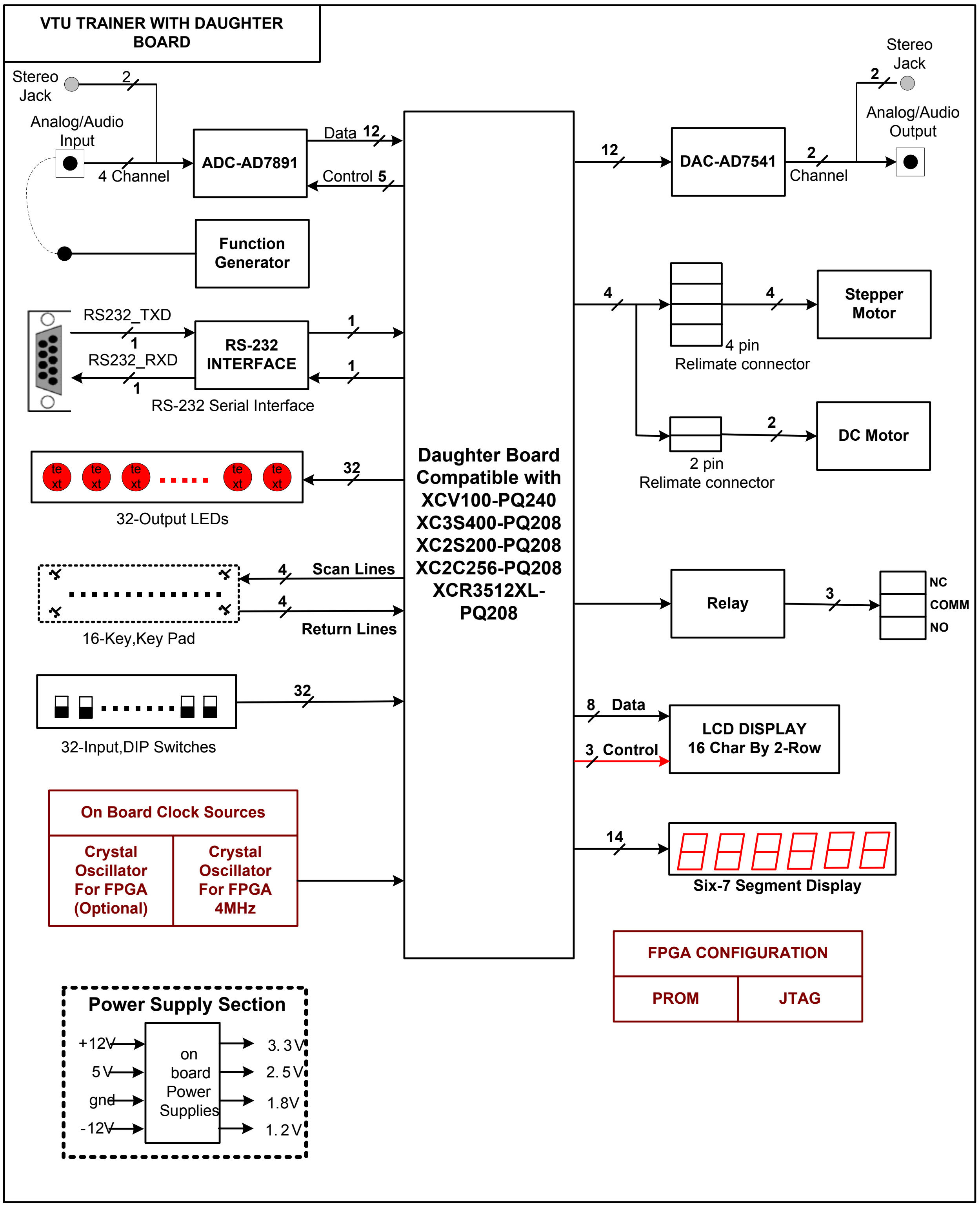


Figure 1: Block Diagram

- **Seven Segment Display:** Six-character, seven-segment LED display.
- **DIP Switches:** 32 DIP switches.
- **LEDs:** 70 onboard LEDS
  - 32 output LEDs (OL 0 – OL 31).
  - 32 input LEDs (IL 0 – IL 31).
  - Done LED.(DONE)
  - 4 Power ON LEDs (LED3V3, LED2V5, LED1V8, LED1V2).
  - 1 Relay LEDs (RNC).
- **Push Button Switches:** 31 momentary-contact push button switches.
- **LCD interface:** 16 character 2 row LCD.
- **Serial Interface:** One RS-232 channel using MAX3223, 9 pin two channel serial interfaces.
  - DB9 9-pin female connector (DCE connector).
  - RS-232 transceiver/level translator using MAX3223 in SSOP package.
  - Uses straight-through serial cable to connect to computer or workstation serial port.
- **Stepper Motor Interface:** Stepper Motor interface using 12VDC, Steps/Rev-200 motor with step angle of 1.8^0
- **Relay Interface:** NC contact are provided using Relay-12VDC.
- **User selectable configuration modes**.
- **Clock Oscillator:** 4 MHz crystal clock oscillator. Socket for an auxiliary crystal oscillator clock source.
- **JTAG port:** JTAG download cable (parallel III) interface.
- **Power Supplies:** 5 volts regulated power supply provided along with the board.
  - On board 3.3V, 2.5V, 1.8V, 1.2V regulators.
  - FPGA supplies viz. Vccint & Vcco are generated on board.

CHAPTER 2
ADC - DAC Interface

SPARTAN-3 DSP ProtoBoard has a high speed, 12 bit ADC (AD 7891) and DAC (AD 7541), surface mounted on top side of the board. A detailed interface is as shown in Figure 2

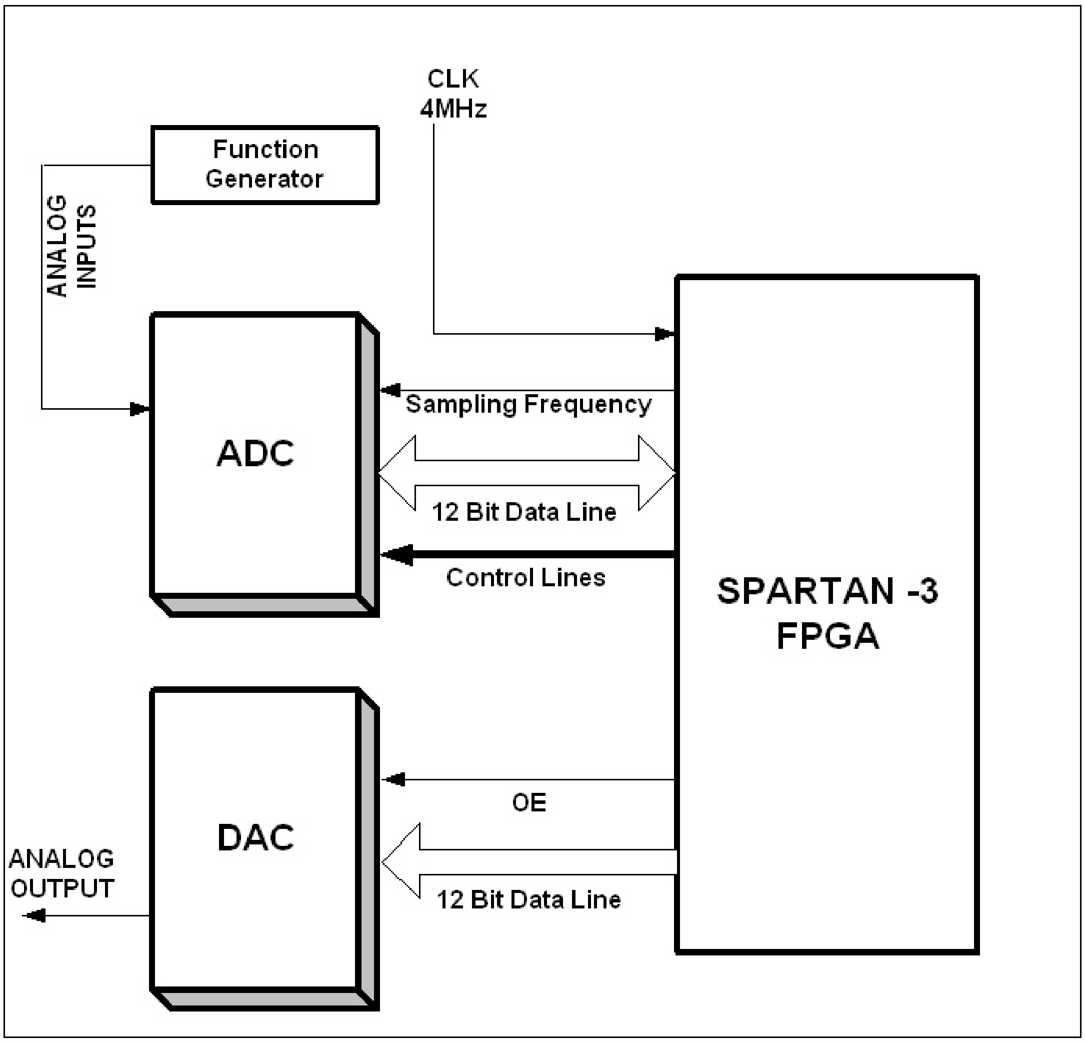


Figure 2: FPGA – ADC DAC Interface

2.1 ANALOG INPUT

Five analog input channels (In-Channel 1 to In-Channel 4 and In-Channel 5 for Thermister), as shown in Figure 3, are provided using ADC - AD7891, with following specifications

- Input range - <u>+10V to -10 Volts.</u>
- In-Channel 1 and In-Channel 2 can take external analog inputs either from the PUT terminal or audio inputs from the stereo jacks provided.
- In-Channel 3 takes an external analog input from the PUT terminal.

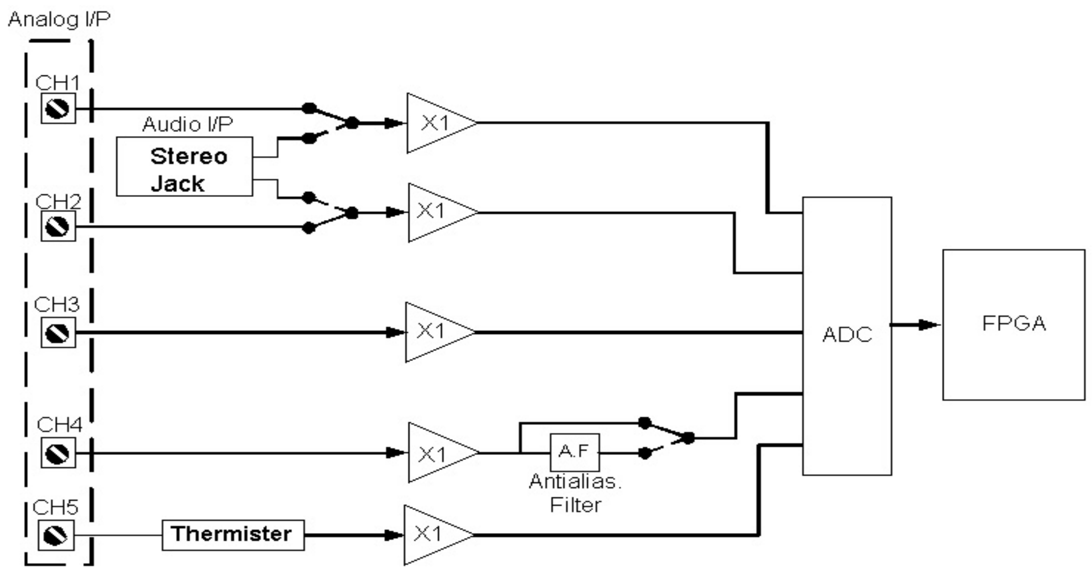


Figure 3: Input Channels of ADC

- In-Channel 4 takes an external analog input from the PUT terminal, user has the option of cascading the onboard **Anti-Aliasing Filter** (a low pass Analog filter) to his input. Figure 4 shows Anti Aliasing Filter
- Channel 5 takes input from Thermister.

Note - AD7891 ADC has eight single ended channels out of which only five channels are used as analog inputs.

2.2 Analog Input Connector:

- **CH1 to CH4:** Four PUTs are provided to connect single ended ANALOG INPUTS

NOTE: Anti Aliasing Filter is provided for channel 4.

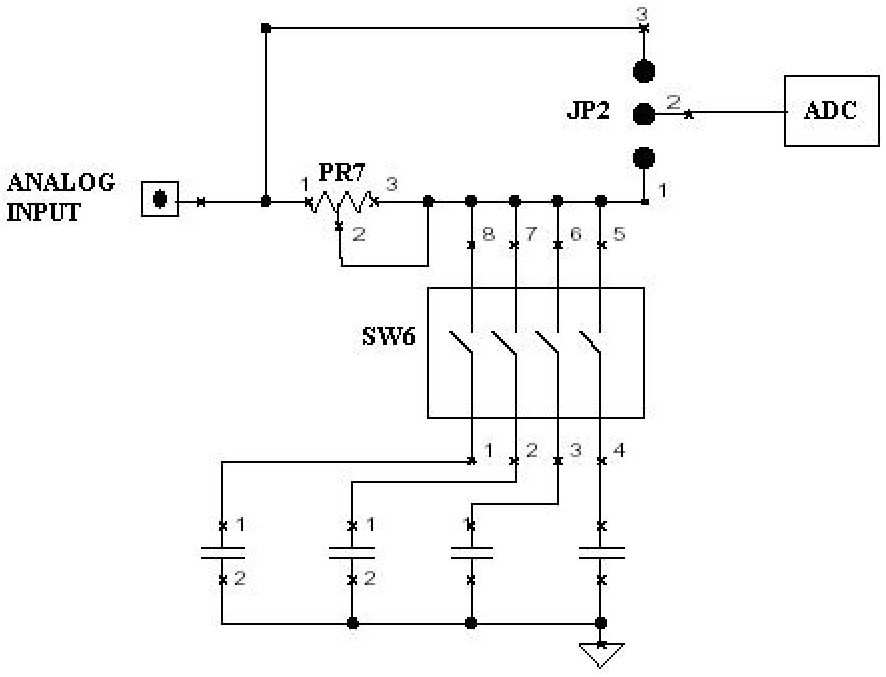


Figure 4 : Anti- Aliasing Filter

Table 1: Analog Inputs

Ch1		Ch2		Ch3		Ch4	
Ch1 In	GND	Ch2 In	GND	Ch3 In	GND	Ch4 In	GND
1	2	1	2	1	2	1	2

Table 2: ADC Interface to SPARTAN 3 FPGA

Data Bits	**FPGA pin**
"ADC_D<0>"	loc = "p51";
"ADC_D<1>"	loc = "p52";
"ADC_D<2>"	loc = "p64";
"ADC_D<3>"	loc = "p65";
"ADC_D<4>"	loc = "p67";
"ADC_D<5>"	loc = "p68";
"ADC_D<6>"	loc = "p71";
"ADC_D<7>"	loc = "p72";
"ADC_D<8>"	loc = "p74";
"ADC_D<9>"	loc = "p78";
"ADC_D<10>"	loc = "p79";
"ADC_D<11>"	loc = "p80";

Table 3: Control Inputs to ADC

Control Inputs	**FPGA PIN**
"ADC_CS_BAR"	loc = "p61";

"ADC_RD_BAR"	loc = "p62";
"ADC_WR_BAR"	loc = "p63";
"ADC_EOC"	loc = "p57";
"ADC_MODE"	loc = "p81";
"ADC_CONVST_BAR"	loc = "p58";

2.3 ANALOG OUTPUT

Two analog output channels are provided on-board DAC – AD7541

- Output Range +10 V to -10 Volts, single ended.
- Analog output on Out-Channel 1 and Out-Channel 2 can be routed either to Stereo Jacks or PUT terminals.
- Out-Channel 2, user has the option of either connecting its output directly to PUT terminal or through a **"Reconstruction filter"** (Low pass analog filter), as shown in Figure 5.

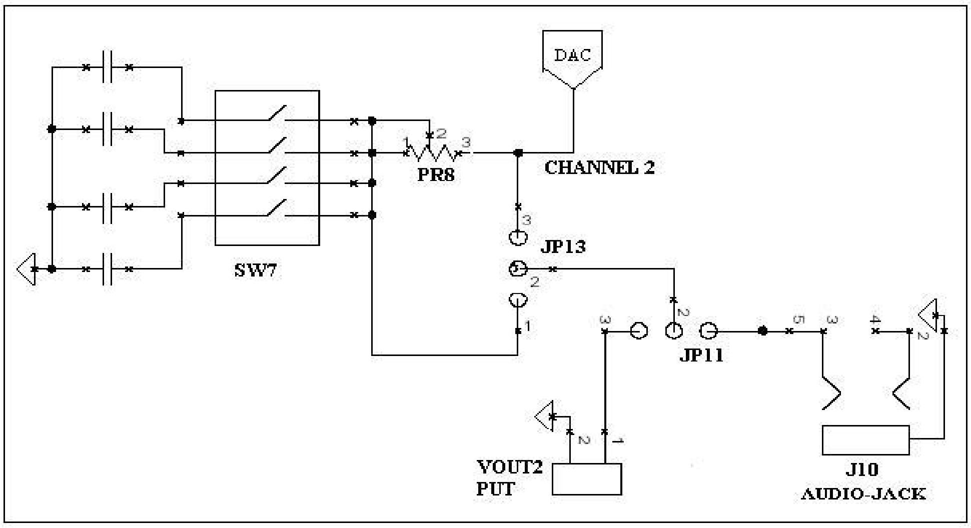


Figure 5: Reconstruction Filter

2.4 Analog Output Connector:

- **Vout1 and Vout2:** Two PUTs are provided to provide ANALOG OUTPUTS

Table 4: Analog Outputs

Vout1		Vout2	
Vout1	GND	Vout2	GND
1	2	1	2

Table 5: DAC Interface to FPGA

Data Bits	**FPGA PIN**
"DAC_D<0>"	loc = "p93";
"DAC_D<1>"	loc = "p76";
"DAC_D<2>"	loc = "p102";
"DAC_D<3>"	loc = "p107";
"DAC_D<4>"	loc = "p100";
"DAC_D<5>"	loc = "p96";

"DAC_D<6>"	loc = "p90";
"DAC_D<7>"	loc = "p101";
"DAC_D<8>"	loc = "p95";
"DAC_D<9>"	loc = "p106";
"DAC_D<10>"	loc = "p97";
"DAC_D<11>"	loc = "p87";
"DAC2_EN"	loc = "p94";
"DAC1_EN"	loc = "p77";

2.6 Stereo Jack Connector:
Stereo jack connectors are provided taking in / giving out signals to/from for audio systems.

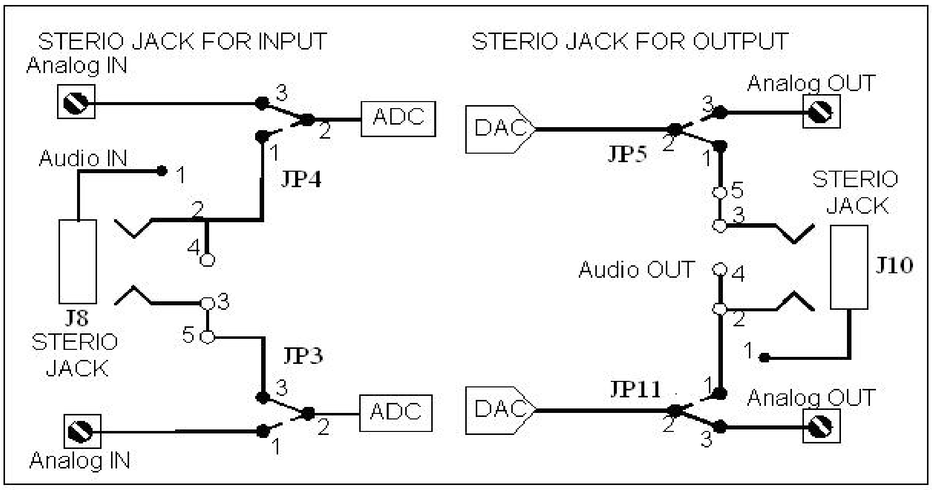


Figure 6: Stereo Jack

2.7 Function Generator
Function Generator - IC8038 is used on board to generate sine, square, triangular waves in the frequency range of 60 Hz to 200 KHz.
Output of function generator can be used as analog input to ADC for performing different DSP applications.

- Function Generator outputs are available as Sine, Square and Triangular wave at three test points – **SINE**, **SQUARE** and **TRIANGULAR** respectively.
- **Frequency Setting** – function generator frequency can be varied in 2-steps
  - Coarse Frequency – using switch SW5.
  - Fine Frequency – using potentiometer PR1 f*or frequency range selection*

SW5 is 4-way DIP Switch and is used to select the frequency range of the Function generator.

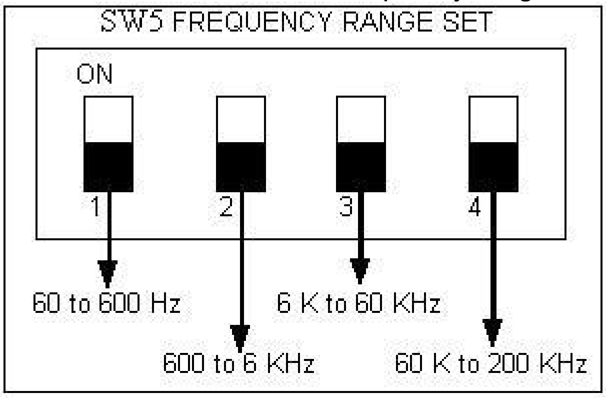


Figure 7: Frequency Range Set

Only one switch must be ON at a time for correct operation of the function generator.

- **Amplitude Setting** – Amplitude of the generated waveform(s) can be adjusted using potentiometers as follows
  - PR1 for Square wave,
  - PR2 for Triangular Wave
  - PR4 for Sine wave

2.8 Potentiometer Adjustments

Details of the potentiometers used in ADC DAC Interface are given in table 6

Table 6: Potentiometer adjustments

Adjustments	**Potentiometer**
Frequency Adjustment	PR6
Square Wave Amplitude Adjustment	PR1
Triangular Wave Amplitude Adjustment	PR2
Sine Wave Amplitude Adjustment	PR4
Offset adjustment of Sine wave	PR5
Offset adjustment of Triangular wave	PR3
Time constant(R) adjustment of anti-Aliasing Filter	PR7
Time constant(R) adjustment of Reconstruction Filter	PR8
DAC-1 Offset adjustment	PR9
DAC-2 Offset adjustment	PR10
DAC Reference Voltage Adjustment	PR11

2.9 Jumper Settings of ADC- DAC Interface

- ***Analog Input to ADC***

Jumper JP3 is for In-Channel 1 and JP2 for In-Channel 2

Table 7: Jumper Setting for ADC Input

JUMPER SETTING JP3 / JP4	
1-2	Audio I/P from Stereo Jack
2-3	Analog I/P from PUT

- ***Anti aliasing filter selection JP4***

Anti-aliasing filter provided with channel 4 is selected using JP4

Table 8: Jumper setting for Anti Aliasing Filter

JUMPER SETTING JP2	
1-2	ADC IN with Anti-Aliasing Filter
2-3	ADC IN without Anti-Aliasing Filter

- ***Analog Output***

Jumper JP5 is for Out-Channel 1 and JP for Out-Channel 2

Table 9: Jumper Setting for Analog Output

JUMPER SETTING JP5/JP11	
1-2	Audio O/P to Stereo Jack
2-3	Analog O/P to PUT

- ***Reconstruction Filter selection JP7***

Reconstruction Filter provided with channel4 is selected using JP13.

Table 10: Jumper Setting for Reconstruction Filter

JUMPER SETTING JP13	
1-2	DAC2 OUT with Reconstruction Filter
2-3	DAC2 OUT without Reconstruction Filter

CHAPTER 3
Serial Interface

The SPARTAN -3 DSP Protoboard supports RS-232 serial interface. The details of interface are described below.

3.1 RS- 232 Interface

The RS-232 transmit and receive signals appear on the female DB9 connector, indicated as in Figure 8. The connector is a DCE-style port and connects to the DB9 DTE-style serial port connector available on most personal computers and workstations. Use a standard straight-through serial cable to connect the SPARTAN -3 protoboard to the PC's serial port.

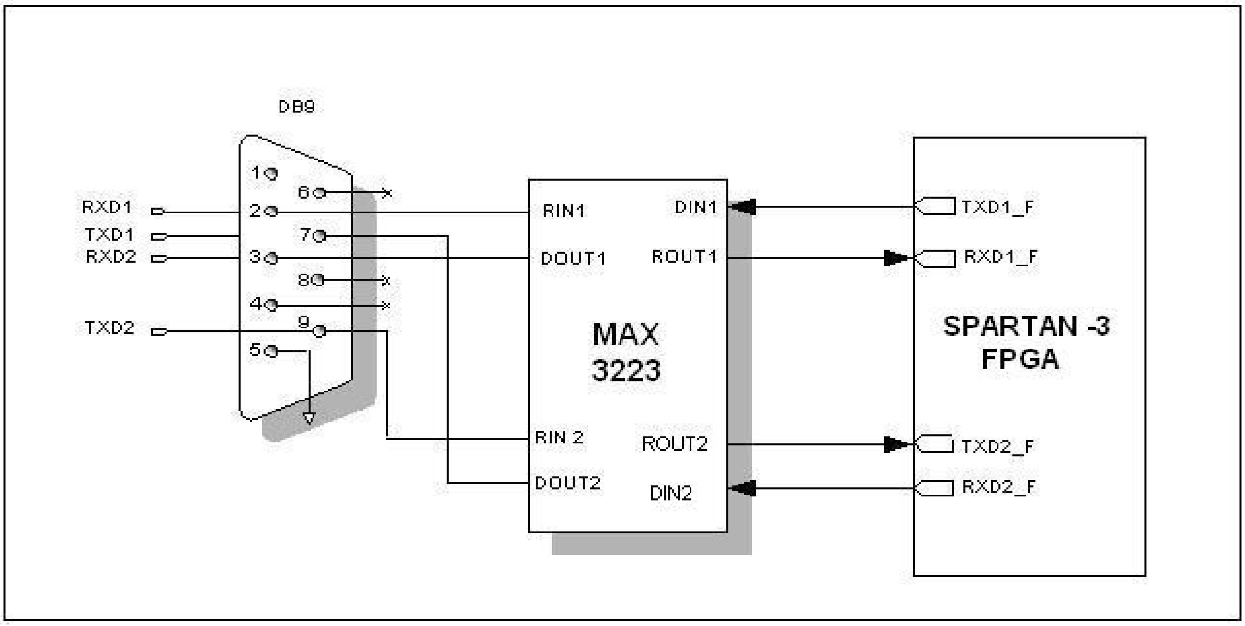


Figure 8: FPGA – RS232 Interface

Figure 8 shows the connection between the FPGA and the DB9 connector, including the Maxim MAX3223 RS-232 voltage converter. The FPGA supplies serial output data as LVTTL or LVCMOS levels to the Maxim device, which in turn converts the logic value to the appropriate RS-232 voltage level. Likewise, the Maxim device converts the RS-232 serial input data to LVTTL levels for the FPGA.
Hardware flow control is not supported on the connector. The port's DCD, DTR, and DSR signals are left unconnected. Similarly, the port's CTS and Ring Indicator are used as an auxiliary RS232 channel signals
The FPGA connections to the Maxim RS-232 translator appear in Table 11.

Table 11: RS232 Interface to SPARTAN -3 FPGA

Control Bit	FPGA Pin
"RS232_RXD"	loc = "p111";
"RS232_TXD"	loc = "p109";

For more details on RS232 UART application please refer the following application note AN2141 from Maxim.

Seven Segment LED Display

The SPARTAN-3 protoboard has a Six-character, seven segment LED display controlled by FPGA user-I/O pins. Each digit shares eight common control signals to light individual LED segments. Each individual character has a separate cathode control input.

To light an individual signal, drive the individual segment control signal High along with the associated cathode control signal for the individual character. The control signal is high, enabling the control inputs for the left-most character. The segment control inputs, A through G and DP, drive the individual segments that comprise the character. A High value lights the individual segment, a Low turns off the segment.

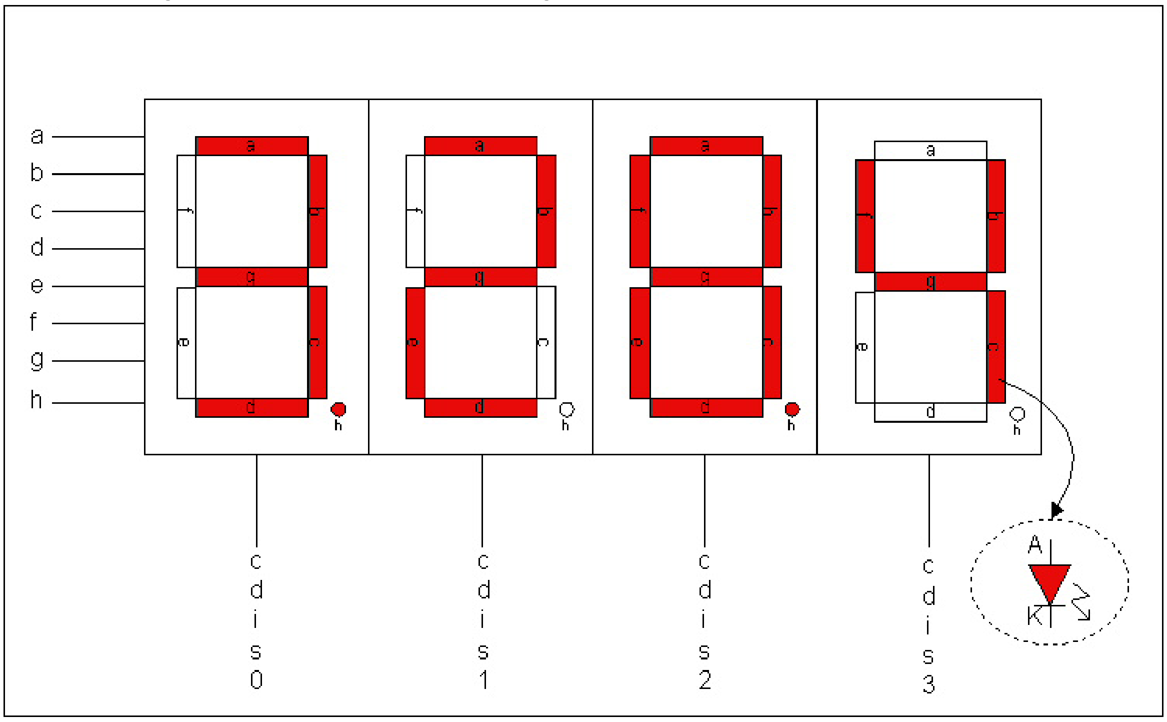
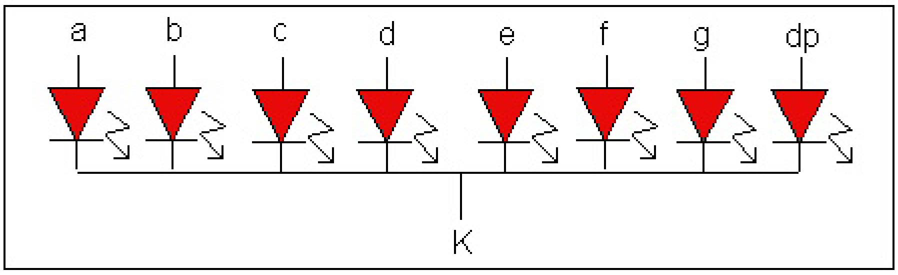


Figure 9: Seven Segment Display

The two types of the seven segment displays are as shown below

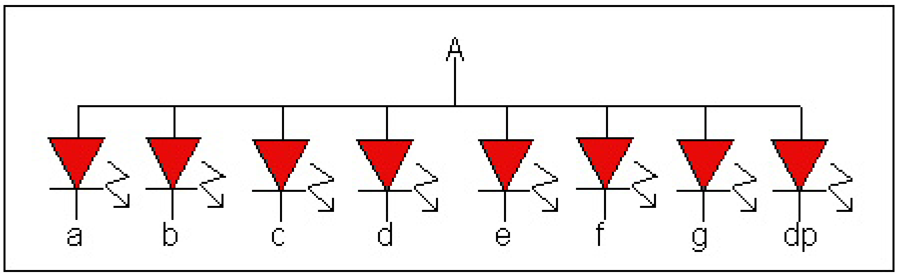
- **Common Cathode Display:** In this type of display the cathode of all the LEDs are tied together and the anode terminals decides the status of the LED, either ON or OFF.
- To turn ON the LED i.e. segment value of driven segment should be 1 and 0 for turn OFF.



COMMON CATHODE DISPLAY

- **Common Anode Display:** In this type of display all the anode terminals of LEDs are tied together and the cathode terminals decide the status of the LED either ON or OFF.

- To turn ON the LED i.e. segment value of driven segment should be 0 and 1 for turn OFF.



COMMON ANODE DISPLAY

Interface details for the seven segment display with SPARTAN-3 display is as follows

Table 12: Seven Segment Display Interface to SPARTAN-3 FPGA

Control Bit	FPGA Pin
"SEGA"	loc = "p120";
"SEGB"	loc = "p124";
"SEGC"	loc = "p123";
"SEGD"	loc = "p126";
"SEGDP"	loc = "p122";
"SEGE"	loc = "p125";
"SEGF"	loc = "p130";
"SEGG"	loc = "p128";
"DIS<0>"	loc = "p117";
"DIS<1>"	loc = "p119";
"DIS<2>"	loc = "p115";
"DIS<3>"	loc = "p116";
"DIS<4>"	loc = "p114";
"DIS<5>"	loc = "p113";

CHAPTER 5

Stepper Motor And Relay Interface

5.1 Stepper Motor Interface

The stepper motor provided onboard interface with the following specification-

- *Stepper Motor-SMO-02*
    - Input Voltage – 12 VDC
    - Step Angle – 1.8^0
    - Steps/Revolution – 200

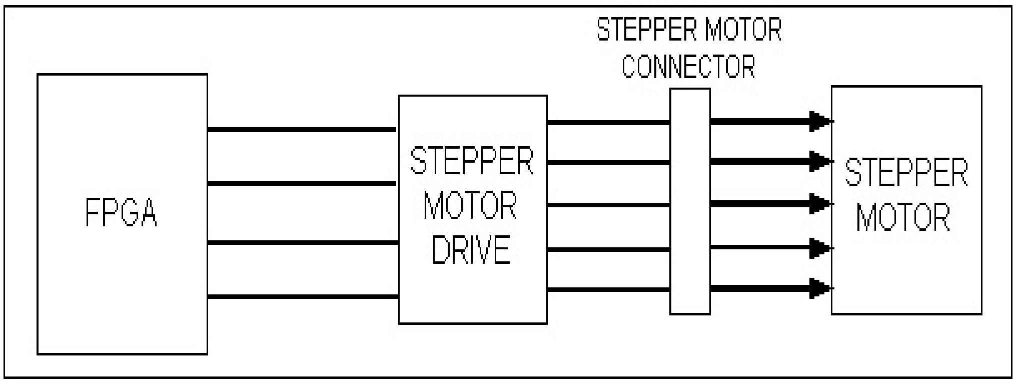


Figure 10: Stepper Motor Interface

- *Stepper Motor Connector Details*

By giving control signals to these pins the Speed and Direction of Rotation of Stepper Motor can be controlled

Table 13: Stepper motor Connector: J14

A1_Coil	A2_Coil	+12 V	B1_coil	B2_coil
1	2	3	4	5

5.2 Relay Interface

A relay is provided on board with following specification.

- *Relay*
    - Input Voltage – 12 V DC
    - Single Contact

Do not connect any external power supply at the TB.
Connect load of 12V only.
LED is provided indicate NC condition. Green LED (LED1) indicates NC condition.

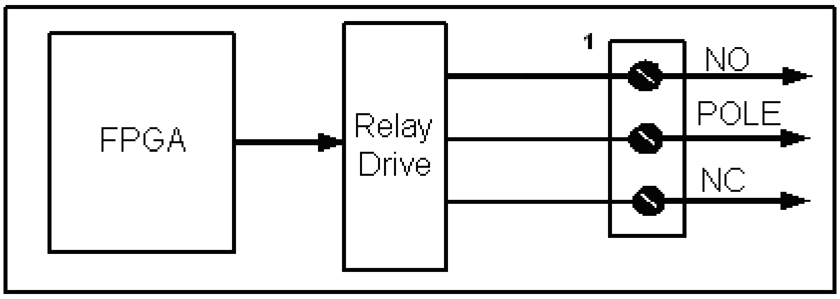


Figure 11: Relay Interface

Table 14: Stepper Motor and Relay Interface to FPGA

Function	FPGA Pin Number
"COIL_A<0>"	loc = "p33";
"COIL_A<1>"	loc = "p36";
"COIL_B<0>"	loc = "p31";
"COIL_B<1>"	loc = "p29";
"RELAY<0>"	loc = "p34";

CHAPTER 6
DC Motor

6.1 DC Motor Interface

The DC motor provided onboard interface with the following specification-

- *DC MOTOR Interface*
    - Input Voltage – 12 VDC

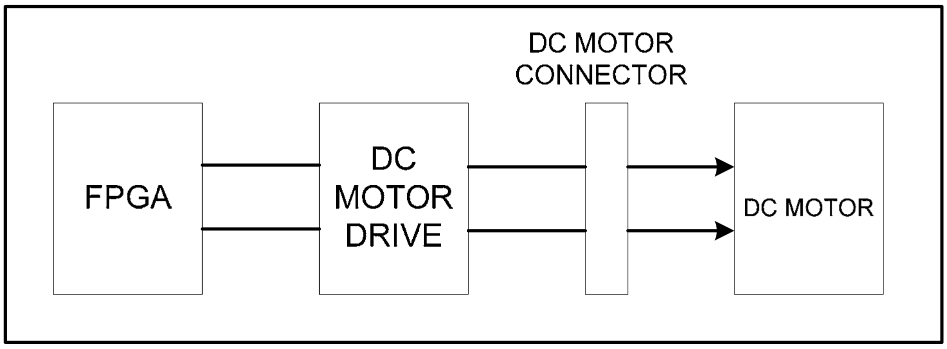


Figure 12: DC Motor Interface

- *DC Motor Connector Details*

By giving control signals to these pins the Speed and Direction of Rotation of DC Motor can be controlled

Table 15: DC motor Connector: J14

B1_coil	B2_coil
4	5

CHAPTER 7

LCD Interface

SPARTAN-3 proto board includes a LCD Module, which is a dot matrix liquid crystal display that displays alphanumeric, Kana (Japanese) characters and symbols. Built in controller provides connectivity between LCD and FPGA.
This LCD has a built in Dot Matrix controller, with font 5x7 or 5x10 dots, display data RAM for 80 characters (80 x 8 bit) and a character generator ROM which provides 160 characters with 5x7 font and 32 characters with font of 5x10.
All the functions required for LCD are provided internally. Internal refresh is provided by the Controller.
The Interface details of the LCD display are as shown in figure 13.

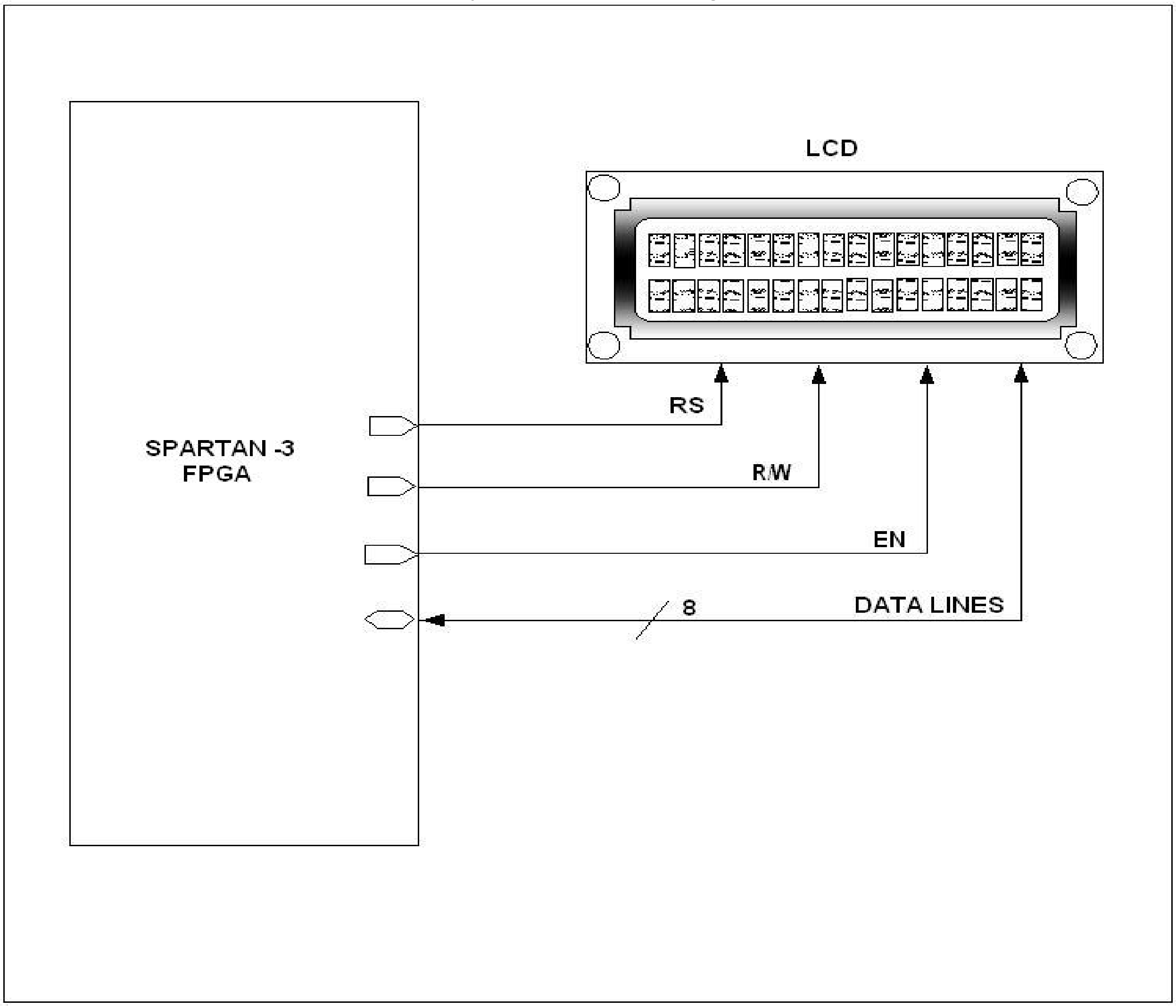


Figure 13: LCD Interface to SPARTAN-3 FPGA

7.1 Data Lines Connection

LCD has 8 bit bidirectional data bus interface to FPGA. When Enable signal is at low level, these data bus remains in high impedance state.

Interface details of the data lines with SPARTAN-3 FPGA are as in Table 16

Table 16: Data Line Interface to SPARTAN-3 FPGA

Data Bit	FPGA Pin
"LCD_D<0>"	loc = "p46";
"LCD_D<1>"	loc = "p43";
"LCD_D<2>"	loc = "p44";
"LCD_D<3>"	loc = "p40";
"LCD_D<4>"	loc = "p42";
"LCD_D<5>"	loc = "p37";
"LCD_D<6>"	loc = "p39";
"LCD_D<7>"	loc = "p35";

7.2 Control Line Interface:

The control lines of LCD comprises of RS, R/W# and E The significance of the above mentioned control signals is as follows

- **RS:** Register select signal used to select Data register or a Command/Status register.
  - High on RS selects the data register.
  - Low on RS selects the Command/Status register.

- **R/W#:** Read/Write select control line.
  - High on R/W # selects the read operation
  - Low on R/W # selects the write operation.

- **E:** Enable signal used to enable or disable the data bus.
  - Low on the enable signal puts the data bus into a high impedance state.
  - High on the enable signal selects the data bus.

The control line interface of LCD with FPGA is as shown in table 17

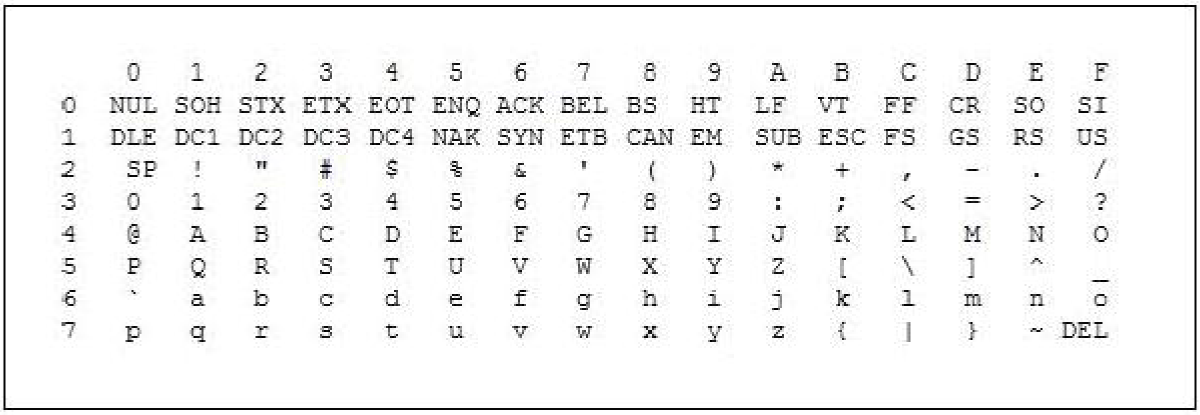
Table 17: Control Line Interface to SPARTAN-3 FPGA

Control Bit	FPGA Pin
"LCD_E"	loc = "p45";
"LCD_RS"	loc = "p48";
"LCD_RW"	loc = "p50";

Note: PR12 is used to adjust the contrast of LCD Display

7.3 ASCII CODE

The ASCII code for 5x 7 LCD Display is given in Figure 14



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Figure 14: ASCII Code for 5 x 7 LCD display

CHAPTER 8

Switches And LEDs

8.1 DIP Switches

The SPARTAN -3 proto board has 32 DIP switches.
The switches connect to an associated FPGA pin, as shown in Table 18
A 4.7KΩ series resistor provides nominal input protection.

Table 18: DIP switch Interface to SPARTAN-3 FPGA

Control Bit	FPGA Pin
"IL<0>"	loc = "p191";
"IL<1>"	loc = "p190";
"IL<2>"	loc = "p196";
"IL<3>"	loc = "p194";
"IL<4>"	loc = "p198";
"IL<5>"	loc = "p197";
"IL<6>"	loc = "p200";
"IL<7>"	loc = "p199";
"IL<8>"	loc = "p5";
"IL<9>"	loc = "p7";
"IL<10>"	loc = "p3";
"IL<11>"	loc = "p4";
"IL<12>"	loc = "p205";
"IL<13>"	loc = "p2";
"IL<14>"	loc = "p203";
"IL<15>"	loc = "p204";
"IL<16>"	loc = "p27";
"IL<17>"	loc = "p28";
"IL<18>"	loc = "p24";
"IL<19>"	loc = "p26";
"IL<20>"	loc = "p21";
"IL<21>"	loc = "p22";
"IL<22>"	loc = "p19";
"IL<23>"	loc = "p20";
"IL<24>"	loc = "p10";
"IL<25>"	loc = "p9";
"IL<26>"	loc = "p12";
"IL<27>"	loc = "p11";
"IL<28>"	loc = "p15";
"IL<29>"	loc = "p13";
"IL<30>"	loc = "p18";

"IL<31>"	loc = "p16";

When in the UP or ON position, a switch connects the FPGA pin to V_{CCO}, a logic High. When DOWN or in the OFF position, the switch connects the FPGA pin to ground, a logic Low. The switches typically exhibit about 2 ms of mechanical bounce and there is no active debouncing circuitry, although such circuitry could easily be added to the FPGA design programmed on the board.

8.2 Key Switches

The SPARTAN-3 proto board has 16 momentary-contact Key switches(4X4 Matrix), indicated as in Figure1. These are located along the lower edge of the board, toward the left edge. The switches are labelled K0 through K15. The switches connect to an associated FPGA pin, as shown in Table 19. Pressing a key generates logic High on the associated FPGA pin. There is no active debouncing circuitry on the key switches.

Table 19: KEY switch Interface to SPARTAN-3 FPGA

Return Lines	
net "RL<0>"	loc = "p189";
net "RL<1>"	loc = "p182";
net "RL<2>"	loc = "p183";
net "RL<3>"	loc = "p180";
Scan Lines	
net "SL<0>"	loc = "p178";
net "SL<1>"	loc = "p184";
net "SL<2>"	loc = "p185";

8.3 LEDS

The SPARTAN-3 proto board has 16 multicoloured LEDs located above the key switches, indicated in Figure 1. The LEDs are labelled LED15 through LED0.Table 20 shows the FPGA connections to the LEDs.
A series current limiting resistor of 270Ω is associated with every LED. To light an individual LED, drive the associated FPGA control signal High

Table 20: LED Interface to SPARTAN-3 FPGA

Control Bit	**FPGA Pin**
"OL<0>"	loc = "p139";
"OL<1>"	loc = "p140";
"OL<2>"	loc = "p137";
"OL<3>"	loc = "p138";
"OL<4>"	loc = "p133";
"OL<5>"	loc = "p135";
"OL<6>"	loc = "p131";
"OL<7>"	loc = "p132";
"OL<8>"	loc = "p149";
"OL<9>"	loc = "p150";
"OL<10>"	loc = "p147";
"OL<11>"	loc = "p148";
"OL<12>"	loc = "p144";

Control Bit	**FPGA Pin**
"OL<13>"	loc = "p146";
"OL<14>"	loc = "p141";
"OL<15>"	loc = "p143";
"OL<16>"	loc = "p181";
"OL<17>"	loc = "p176";
"OL<18>"	loc = "p172";
"OL<19>"	loc = "p175";
"OL<20>"	loc = "p169";
"OL<21>"	loc = "p168";
"OL<22>"	loc = "p167";
"OL<23>"	loc = "p171";
"OL<24>"	loc = "p165";
"OL<25>"	loc = "p166";
"OL<26>"	loc = "p161";
"OL<27>"	loc = "p162";
"OL<28>"	loc = "p155";
"OL<29>"	loc = "p156";
"OL<30>"	loc = "p152";
"OL<31>"	loc = "p154";

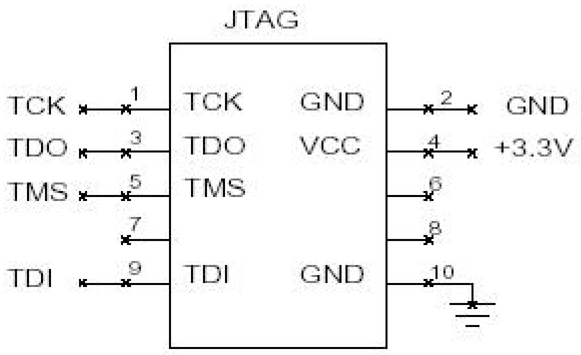
CHAPTER 9

Connector Details

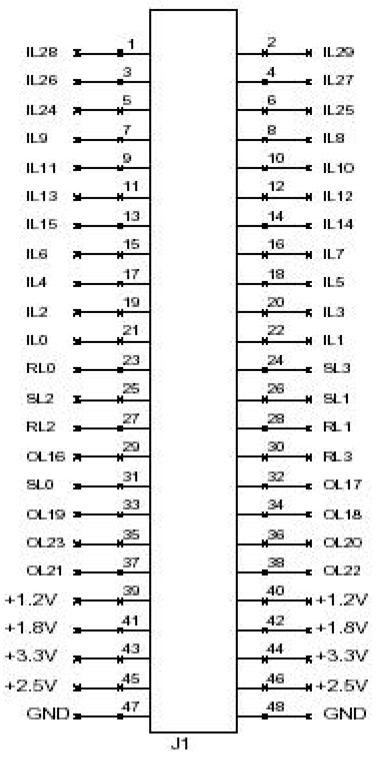
SPARTAN-3 Proto Board has 5 connectors.

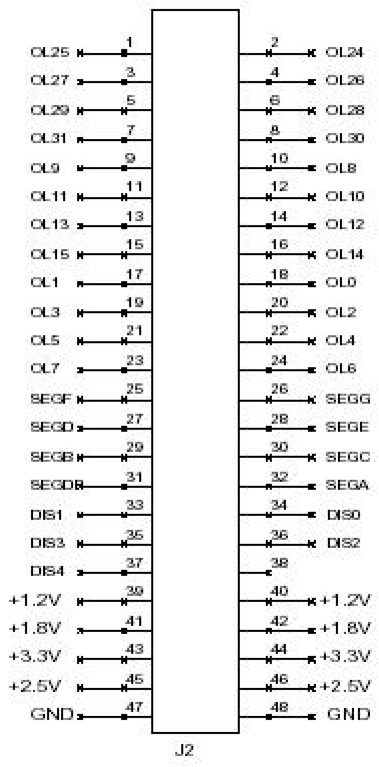
For XC3S400 (PQ208) Daughter Board IOs are provided on four connectors as follows,

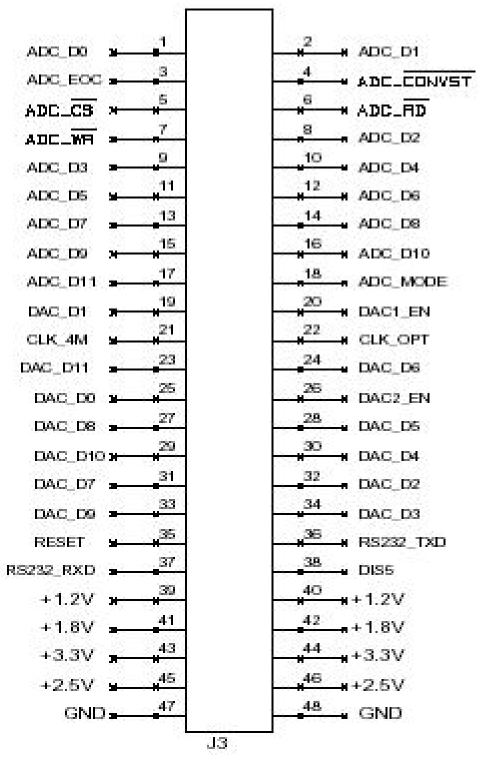
- **J1:** A 48 Pin FRC Female connector.
- **J2 :** A 48 Pin FRC Female connector.
- **J3:** A 48 Pin FRC Female connector.
- **J4:** A 48 Pin FRC Female connector.
- **JTAG Connector:** A 10 pin FRC male connector provided for JTAG cable connection.

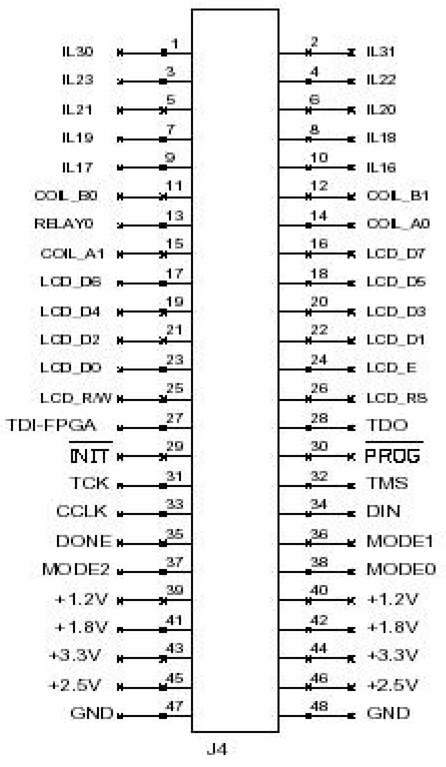


JTAG Connector


IL28 1 2 IL29
IL26 3 4 IL27
IL24 5 6 IL25
IL9 7 8 IL8
IL11 9 10 IL10
IL13 11 12 IL12
IL15 13 14 IL14
IL6 15 16 IL7
IL4 17 18 IL5
IL2 19 20 IL3
IL0 21 22 IL1
RL0 23 24 SL3
SL2 25 26 SL1
RL2 27 28 RL1
OL16 29 30 RL3
SL0 31 32 OL17
OL19 33 34 OL18
OL23 35 36 OL20
OL21 37 38 OL22
+1.2V 39 40 +1.2V
+1.8V 41 42 +1.8V
+3.3V 43 44 +3.3V
+2.5V 45 46 +2.5V
GND 47 48 GND
J1


OL25 1 2 OL24
OL27 3 4 OL26
OL29 5 6 OL28
OL31 7 8 OL30
OL9 9 10 OL8
OL11 11 12 OL10
OL13 13 14 OL12
OL15 15 16 OL14
OL1 17 18 OL0
OL3 19 20 OL2
OL5 21 22 OL4
OL7 23 24 OL6
SEGF 25 26 SEGG
SEGD 27 28 SEGE
SEGB 29 30 SEGC
SEGDP 31 32 SEGA
DIS1 33 34 DIS0
DIS3 35 36 DIS2
DIS4 37 38
+1.2V 39 40 +1.2V
+1.8V 41 42 +1.8V
+3.3V 43 44 +3.3V
+2.5V 45 46 +2.5V
GND 47 48 GND
J2


ADC_D0 1 2 ADC_D1
ADC_EOC 3 4 ADC_CONVST
ADC_CS 5 6 ADC_RD
ADC_WR 7 8 ADC_D2
ADC_D3 9 10 ADC_D4
ADC_D5 11 12 ADC_D6
ADC_D7 13 14 ADC_D8
ADC_D9 15 16 ADC_D10
ADC_D11 17 18 ADC_MODE
DAC_D1 19 20 DAC1_EN
CLK_4M 21 22 CLK_OPT
DAC_D11 23 24 DAC_D6
DAC_D0 25 26 DAC2_EN
DAC_D8 27 28 DAC_D5
DAC_D10 29 30 DAC_D4
DAC_D7 31 32 DAC_D2
DAC_D9 33 34 DAC_D3
RESET 35 36 RS232_TXD
RS232_RXD 37 38 DIS5
+1.2V 39 40 +1.2V
+1.8V 41 42 +1.8V
+3.3V 43 44 +3.3V
+2.5V 45 46 +2.5V
GND 47 48 GND
J3


IL30 1 2 IL31
IL23 3 4 IL22
IL21 5 6 IL20
IL19 7 8 IL18
IL17 9 10 IL16
COIL_B0 11 12 COIL_B1
RELAY0 13 14 COIL_A0
COIL_A1 15 16 LCD_D7
LCD_D6 17 18 LCD_D5
LCD_D4 19 20 LCD_D3
LCD_D2 21 22 LCD_D1
LCD_D0 23 24 LCD_E
LCD_R/W 25 26 LCD_RS
TDI-FPGA 27 28 TDO
INIT 29 30 PROG
TCK 31 32 TMS
CCLK 33 34 DIN
DONE 35 36 MODE1
MODE2 37 38 MODE0
+1.2V 39 40 +1.2V
+1.8V 41 42 +1.8V
+3.3V 43 44 +3.3V
+2.5V 45 46 +2.5V
GND 47 48 GND
J4

The positional details of these connectors on proto board is as shown in figure 15

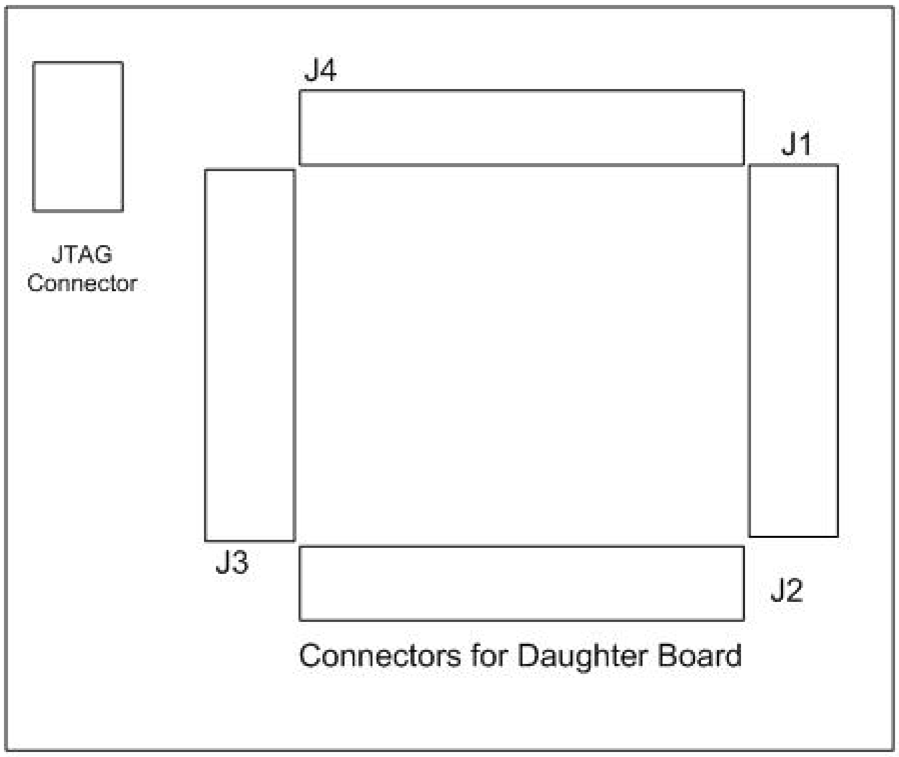


Figure 15: positional Details of On Board Connectors

CHAPTER 10

Clock and Reset Sources

The SPARTAN-3 Proto board has a dedicated 4 MHz oscillator source (OSC1) and an optional socket for another clock oscillator source (OSC2).
This dedicated clock source can be used to derive other frequencies using DCM (digital clock mangers) available in SPARTAN-3 FPGA.
SPARTAN-3 Proto board has on board reset circuitry (a key switch) that is used to reset (active high) the hardware present on the board.

The interface details of clock and reset with FPGA is given in Table 21.

Table 21: IO Clock-Reset Interface to FPGA

Control Bit	**FPGA Pin**
"CLK_4M"	loc = "p85";
"CLK_OPT"	loc = "p86";
"RESET"	loc = "p108";

CHAPTER 11

SPARTAN-3 Configuration Details

SPARTAN-3 Proto board supports two configuration modes as stated below
- Boundary Scan mode
- Master Serial Mode

11.1 Boundary Scan mode:

In boundary scan mode of configuration the SPARTAN-3 FPGA is directly configured via a JTAG port using the dedicated configuration pins TCK, TMS, TDI and TDO.
The jumper setting for selection of boundary scan mode is discussed in jumper setting section (section 11.3).

11.2 Master Serial Mode

In master serial mode the SPARTAN -3 is configured through a FLASH PROM.

In Master Serial mode, the FPGA automatically loads the configuration bit stream in bit-serial form from configuration flash synchronized by the configuration clock (CCLK) generated by the FPGA. Upon power-up or reconfiguration, the FPGA's mode select pins are used to select the Master Serial configuration mode. Master Serial Mode provides a simple configuration interface. Only a serial data line, a clock line, and two control lines (INIT and DONE) are required to configure an FPGA. Data from the PROM is read out sequentially on a single data line (DIN). The serial Bit stream data must be set up at the FPGA's DIN input pin a short time before each rising edge of the FPGA's internally generated CCLK signal.

The jumper setting for selection of Master Serial Mode is discussed in jumper setting section (section 11.3).

11.3 Jumper Setting

- **Mode Selection Jumpers:** M0, M1, M2 are the mode selection jumpers used to select the configuration mode either Boundary Scan or Master Serial Mode.

Table 22: Mode Selection Jumper Settings

Configuration Mode	1	2	• Connecting 1-2 selects Logic –0, • Disconnecting 1-2 selects Logic- 1
MODE0 (M0)	●	●	
MODE1 (M1)	●	●	
MODE2 (M2)	●	●	

Table 23: Mode Selection Table

Configuration Mode	MODE0	MODE1	MODE2
Boundary Scan Mode	1	0	1
Master Serial Mode	0	0	0

Table 24: Configuration Selection

JUMPER SETTING JP1	
1-2	Configuration of PROM + FPGA
2-3	Configuration of FPGA

- **JTAG Chain Selection Jumper:** JP1 jumper is used to select the JTAG chain for configuration. When jumper is connected between 1-2, then PROM and FPGA both get added in the JTAG Chain where as connecting jumper between 2-3 brings only FPGA in Chain.

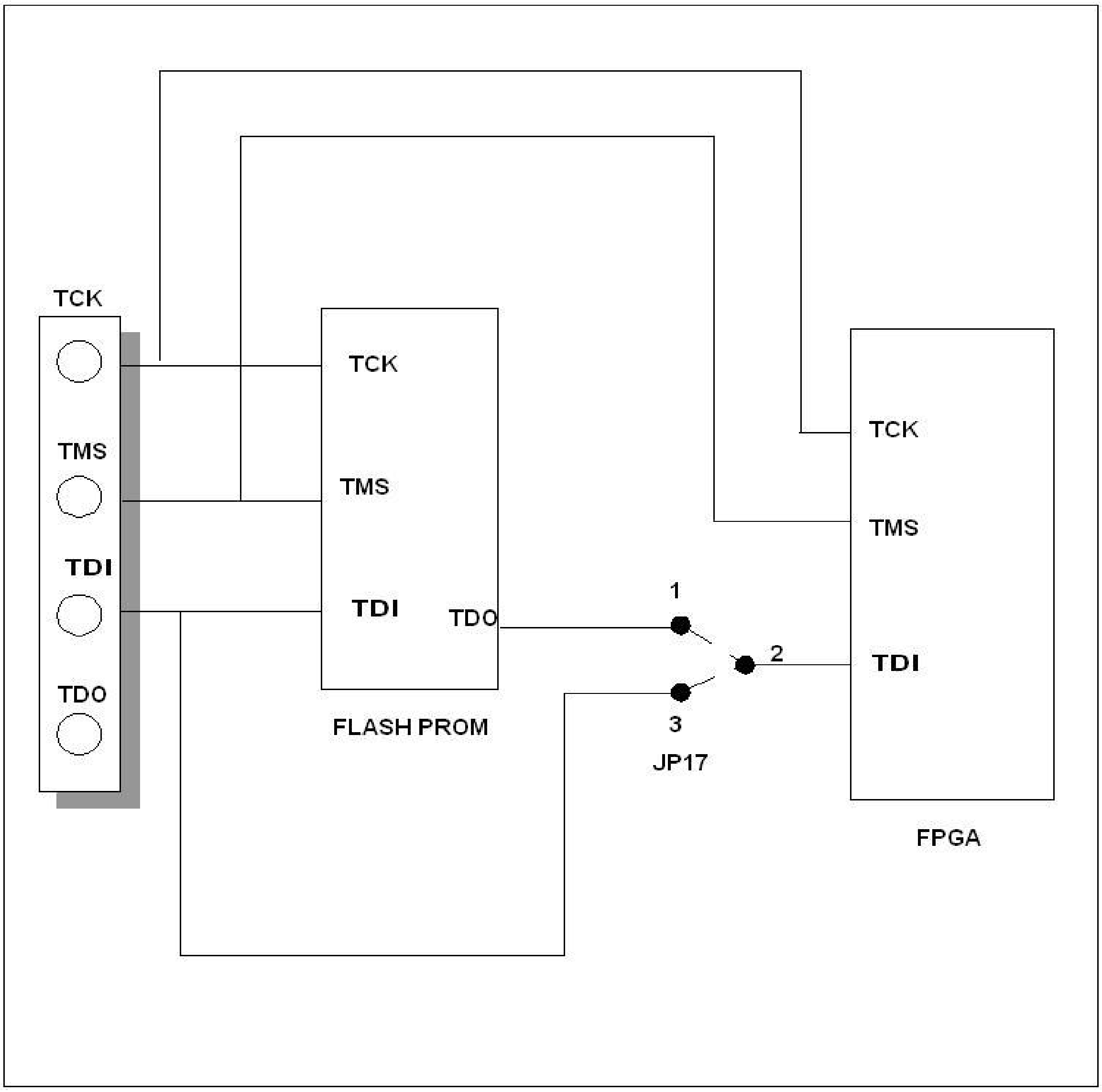


Figure 16: JTAG Mode Selection

13.4 JTAG Header:

An on board JTAG connector (10 pin FRC male) **JTAG** is provided for configuring the FPGA through parallel port of PC via a parallel III cable.
The details of this connector are as shown in figure 17.

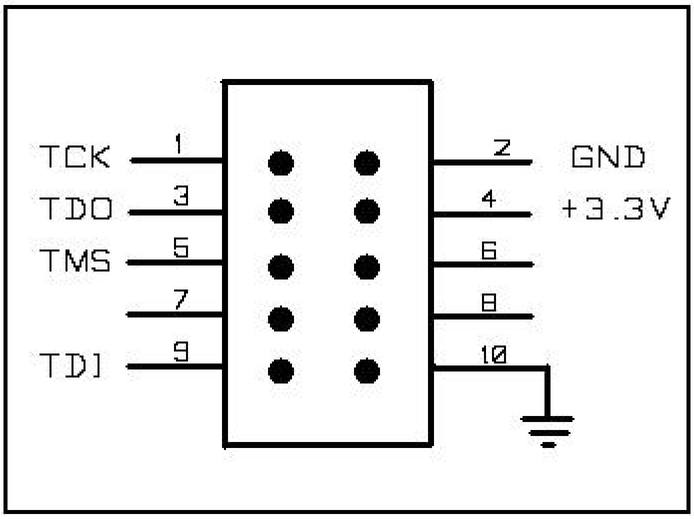


Figure 17: JTAG Connector Details

CHAPTER 12

Power Supplies

SPARTAN-3 Proto board is provided with a regulated power supply of + 5V DC output. This supply is used to generate the required on board supply voltages.

Output of the regulated power supply is given to power connector present on board.
The power LED (Red LED) lights up when power is properly applied to the board.

12.1 Voltage Regulators

The list of various voltage regulators present on board are as given in Table 25.

Table 25: Power Supply Details

Voltage	**Source**
+ 5 V DC	Generated from external power supply provided along with the development Board.
+ 3.3 V DC	A regulated 3.3 V DC supply is generated onboard using regulator LM317 with input voltage of 5 Volts
+2.5 V DC	A regulated 2.5 V DC supply is generated onboard using regulator LM317 with input voltage of 5 Volts
+1.8 V DC	A regulated 1.8 V DC supply is generated onboard using a linear low drop out regulator LT1963 with input voltage of 3.3 Volts.
+1.2 V DC	A regulated 1.2 V DC supply is generated onboard using a linear low drop out regulator LT1963 with input voltage of 3.3 Volts.

Overall, the 5V DC switching power adapter powers the board. A 3.3V regulator, powered by the 5V DC supply, provides power to the inputs of the 1.8V and 1.2V regulators.

Similarly, the 3.3V regulator feeds all the VCCO voltage supply inputs to the FPGA’s I/O banks and powers most of the components on the board.

The 2.5V regulator feeds to the FPGA’s VCCAUX supply inputs.

The FPGA configuration interface on the board is powered by 3.3V. Consequently, the 2.5V supply has a current shunt resistor to prevent reverse current.

Finally, a 1.2V regulator feeds to the FPGA’s VCCINT voltage inputs, which power the FPGA’s core logic.
Test points (pads for probing) are provided on board for every voltage output (TP5V, TP3V3, TP2V5, TP1V8, TP1V2)

APPENDIX A

Consolidated UCF For The Complete Board

Clock And Reset

net "CLK_4M"	loc = "p76";
net "CLK_OPT"	loc = "p77";

net "RESET"	loc = "p108";

Test LEDs

net "OL<0>"	loc = "p139";
net "OL<1>"	loc = "p140";
net "OL<2>"	loc = "p137";
net "OL<3>"	loc = "p138";
net "OL<4>"	loc = "p133";
net "OL<5>"	loc = "p135";
net "OL<6>"	loc = "p131";
net "OL<7>"	loc = "p132";
net "OL<8>"	loc = "p149";
net "OL<9>"	loc = "p150";
net "OL<10>"	loc = "p147";
net "OL<11>"	loc = "p148";
net "OL<12>"	loc = "p144";
net "OL<13>"	loc = "p146";
net "OL<14>"	loc = "p141";
net "OL<15>"	loc = "p143";
net "OL<16>"	loc = "p181";
net "OL<17>"	loc = "p176";
net "OL<18>"	loc = "p172";
net "OL<19>"	loc = "p175";
net "OL<20>"	loc = "p169";
net "OL<21>"	loc = "p168";
net "OL<22>"	loc = "p167";
net "OL<23>"	loc = "p171";
net "OL<24>"	loc = "p165";
net "OL<25>"	loc = "p166";
net "OL<26>"	loc = "p161";
net "OL<27>"	loc = "p162";
net "OL<28>"	loc = "p155";
net "OL<29>"	loc = "p156";
net "OL<30>"	loc = "p152";
net "OL<31>"	loc = "p154";

LCD Interface

Data Lines	
net "LCD_D<0>"	loc = "p46";
net "LCD_D<1>"	loc = "p43";

net "LCD_D<2>"	loc = "p44";
net "LCD_D<3>"	loc = "p40";
net "LCD_D<4>"	loc = "p42";
net "LCD_D<5>"	loc = "p37";
net "LCD_D<6>"	loc = "p39";
net "LCD_D<7>"	loc = "p35";
Control Signals	
net "LCD_E"	loc = "p45";
net "LCD_RS"	loc = "p48";
net "LCD_RW"	loc = "p50";

RS232 Interface

net "RS232_RXD"	loc = "p111";
net "RS232_TXD"	loc = "p109";

Seven Segment Interface

7 Segments	
net "SEGA"	loc = "p120";
net "SEGB"	loc = "p124";
net "SEGC"	loc = "p123";
net "SEGD"	loc = "p126";
net "SEGDP"	loc = "p122";
net "SEGE"	loc = "p125";
net "SEGF"	loc = "p130";
net "SEGG"	loc = "p128";
Enable Signals	
net "DIS<0>"	loc = "p117";
net "DIS<1>"	loc = "p119";
net "DIS<2>"	loc = "p115";
net "DIS<3>"	loc = "p116";
net "DIS<4>"	loc = "p114";
net "DIS<5>"	loc = "p113";

Keys

Return Lines	
net "RL<0>"	loc = "p189";
net "RL<1>"	loc = "p182";
net "RL<2>"	loc = "p183";
net "RL<3>"	loc = "p180";
Scan Lines	
net "SL<0>"	loc = "p178";
net "SL<1>"	loc = "p184";
net "SL<2>"	loc = "p185";

Input Switches

net "IL<0>"	loc = "p191";
net "IL<1>"	loc = "p190";
net "IL<2>"	loc = "p196";

net "IL<3>"	loc = "p194";
net "IL<4>"	loc = "p198";
net "IL<5>"	loc = "p197";
net "IL<6>"	loc = "p200";
net "IL<7>"	loc = "p199";
net "IL<8>"	loc = "p5";
net "IL<9>"	loc = "p7";
net "IL<10>"	loc = "p3";
net "IL<11>"	loc = "p4";
net "IL<12>"	loc = "p205";
net "IL<13>"	loc = "p2";
net "IL<14>"	loc = "p203";
net "IL<15>"	loc = "p204";
net "IL<16>"	loc = "p27";
net "IL<17>"	loc = "p28";
net "IL<18>"	loc = "p24";
net "IL<19>"	loc = "p26";
net "IL<20>"	loc = "p21";
net "IL<21>"	loc = "p22";
net "IL<22>"	loc = "p19";
net "IL<23>"	loc = "p20";
net "IL<24>"	loc = "p10";
net "IL<25>"	loc = "p9";
net "IL<26>"	loc = "p12";
net "IL<27>"	loc = "p11";
net "IL<28>"	loc = "p15";
net "IL<29>"	loc = "p13";
net "IL<30>"	loc = "p18";
net "IL<31>"	loc = "p16";

ADC Interface

Data Lines	
net "ADC_D<0>"	loc = "p51";
net "ADC_D<1>"	loc = "p52";
net "ADC_D<2>"	loc = "p64";
net "ADC_D<3>"	loc = "p65";
net "ADC_D<4>"	loc = "p67";
net "ADC_D<5>"	loc = "p68";
net "ADC_D<6>"	loc = "p71";
net "ADC_D<7>"	loc = "p72";
net "ADC_D<8>"	loc = "p74";
net "ADC_D<9>"	loc = "p78";
net "ADC_D<10>"	loc = "p79";
net "ADC_D<11>"	loc = "p80";
Control Signals	
net "ADC_CS_BAR"	loc = "p61";
net "ADC_RD_BAR"	loc = "p62";
net "ADC_WR_BAR"	loc = "p63";
net "ADC_EOC"	loc = "p57";
net "ADC_MODE"	loc = "p81";
net "ADC_CONVST_BAR"	loc = "p58";

DAC Interface

Data Lines	
net "DAC_D<0>"	loc = "p93";
net "DAC_D<1>"	loc = "p85";
net "DAC_D<2>"	loc = "p102";
net "DAC_D<3>"	loc = "p107";
net "DAC_D<4>"	loc = "p100";
net "DAC_D<5>"	loc = "p96";
net "DAC_D<6>"	loc = "p90";
net "DAC_D<7>"	loc = "p101";
net "DAC_D<8>"	loc = "p95";
net "DAC_D<9>"	loc = "p106";
net "DAC_D<10>"	loc = "p97";
net "DAC_D<11>"	loc = "p87";
Control Signals	
net "DAC2_EN"	loc = "p94";
net "DAC1_EN"	loc = "p86";

STEPPER MOTOR

STEPPER MOTOR	
net "COIL_A<0>"	loc = "p33";

net "COIL_A<1>"	loc = "p36";
net "COIL_B<0>"	loc = "p31";
net "COIL_B<1>"	loc = "p29";

DC MOTOR AND RELAY

DC MOTOR	
net "COIL_B<0>"	loc = "p31";
net "COIL_B<1>"	loc = "p29";
Relay	
net "RELAY<0>"	loc = "p34";

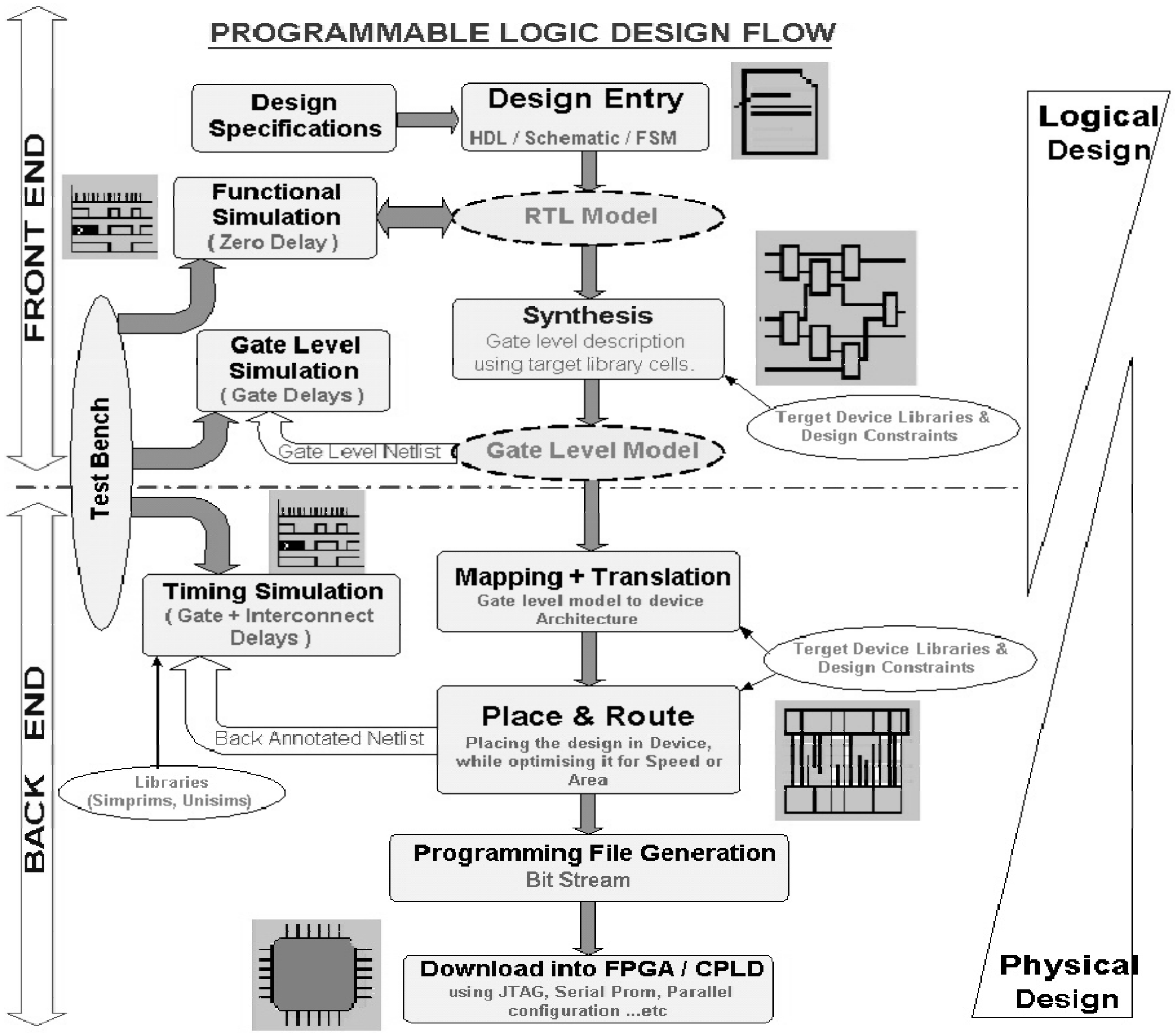
Operating Instructions To Start A New Design

B.1 Starting The ISE Software:

- Start ISE from the Start menu by selecting **Start ->˜ Programs ->˜ Xilinx ISE Project Navigator**.

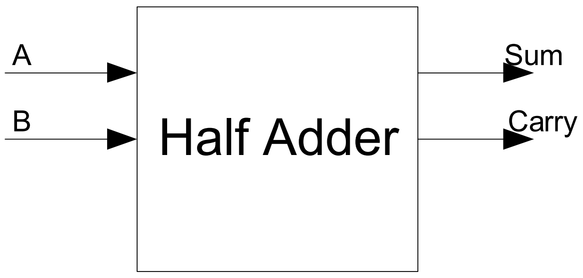
B.2 Design Flow

- DESIGN ENTRY
- SIMULATION
- SYNTHESIS
- IMPLEMENTATION
- DEVICE PROGRAMMING


PROGRAMMABLE LOGIC DESIGN FLOW
FRONT END
BACK END
Design Specifications
Design Entry
HDL / Schematic / FSM
Logical Design
Functional Simulation
(Zero Delay)
RTL Model
Synthesis
Gate level description using target library cells.
Test Bench
Gate Level Simulation
(Gate Delays)
Gate Level Netlist
Gate Level Model
Terget Device Libraries & Design Constraints
Timing Simulation
(Gate + Interconnect Delays)
Mapping + Translation
Gate level model to device Architecture
Terget Device Libraries & Design Constraints
Back Annotated Netlist
Place & Route
Placing the design in Device, while optimising it for Speed or Area
Libraries
(Simprims, Unisims)
Programming File Generation
Bit Stream
Download into FPGA / CPLD
using JTAG, Serial Prom, Parallel configuration ...etc
Physical Design

Sample Design of Half Adder is used to explain the Design Flow.

B.3 Design Description



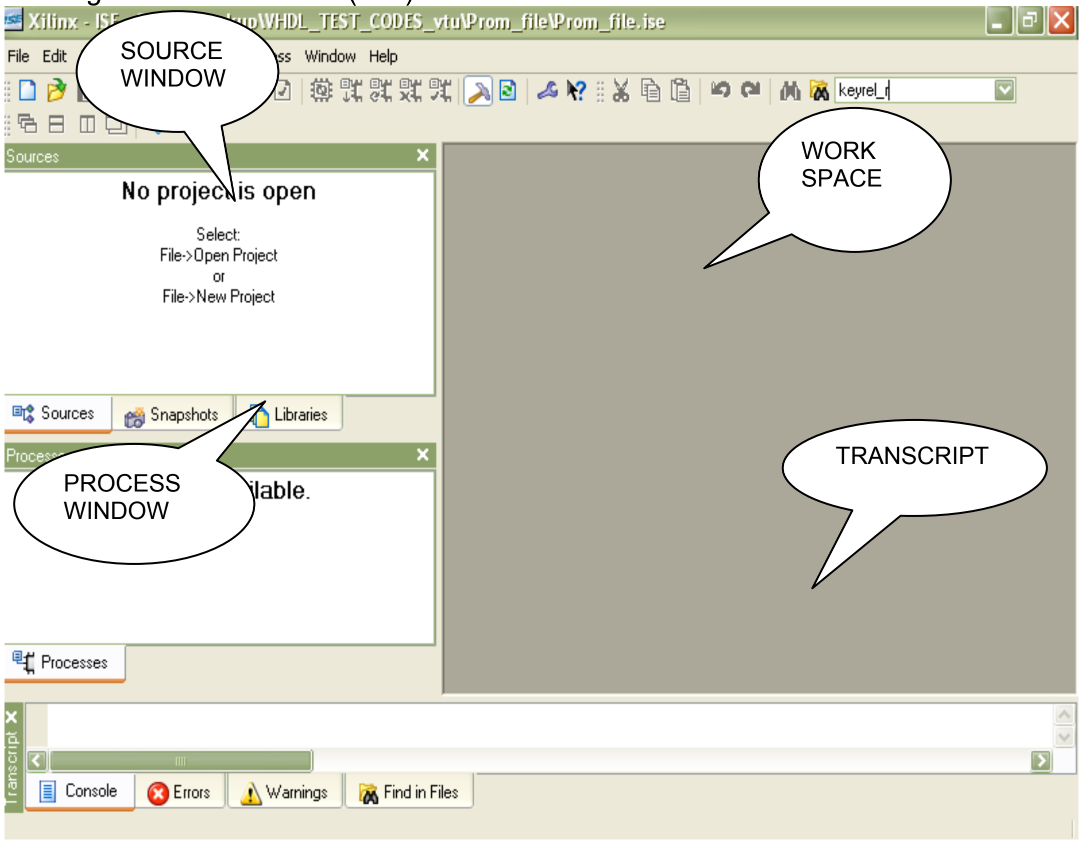
B.4 Truth Table of Half adder: -

Inputs		**Output**	
A	**B**	**Sum**	**Carry**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

B.5 VHDL Code for Half adder

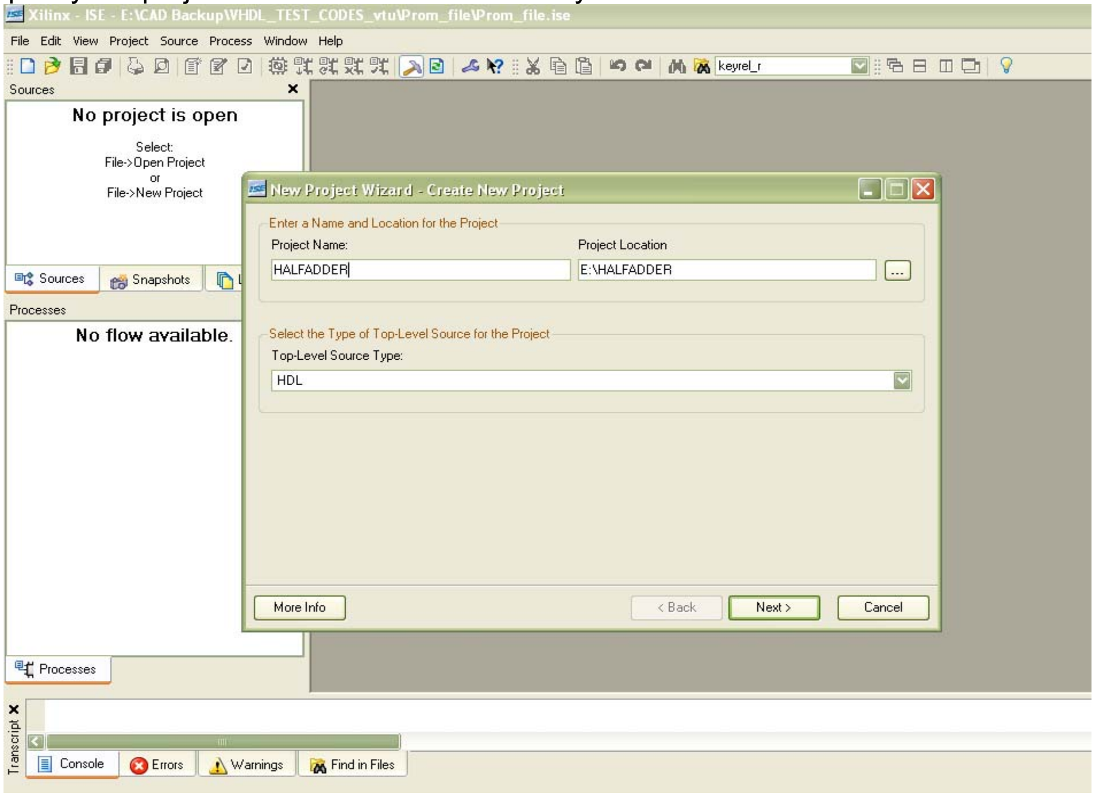
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity half_adder is
  Port ( a : in std_logic;
         b : in std_logic;
         c : in std_logic;
       sum : out std_logic;
     carry : out std_logic);
end full_adder;

architecture Behavioral of half_adder is

begin
            sum <= a xor b ;
            carry <= a and b;
      end Behavioral;
```

B.6 Steps to implement the Half adder in the FPGA using Xilinx ISE(8.1i)

Step 1 :	Start the Xilinx Project Navigator by using the desktop shortcut or by using the *Start* → *Programs* → *Xilinx ISE* (8.1i).  Xilinx - ISE ...\VHDL_TEST_CODES_vtu\Prom_file\Prom_file.ise File Edit ... ss Window Help keyrel_r Sources Sources Snapshots Libraries Process Processes Transcript Console Errors Warnings Find in Files SOURCE WINDOW WORK SPACE PROCESS WINDOW TRANSCRIPT No project is open Select: File->Open Project or File->New Project ...ilable.

Step 2

Create a new project
In the window go to *FILE →New project*.
Specify the project name and location and say *NEXT*



Xilinx - ISE - E:\CAD Backup\VHDL_TEST_CODES_vtu\Prom_file\Prom_file.ise

File Edit View Project Source Process Window Help
keyrel_r
Sources
Processes
Sources Snapshots
Console Errors Warnings Find in Files
Transcript

No project is open

Select:
File->Open Project
or
File->New Project

No flow available.

New Project Wizard - Create New Project

Enter a Name and Location for the Project

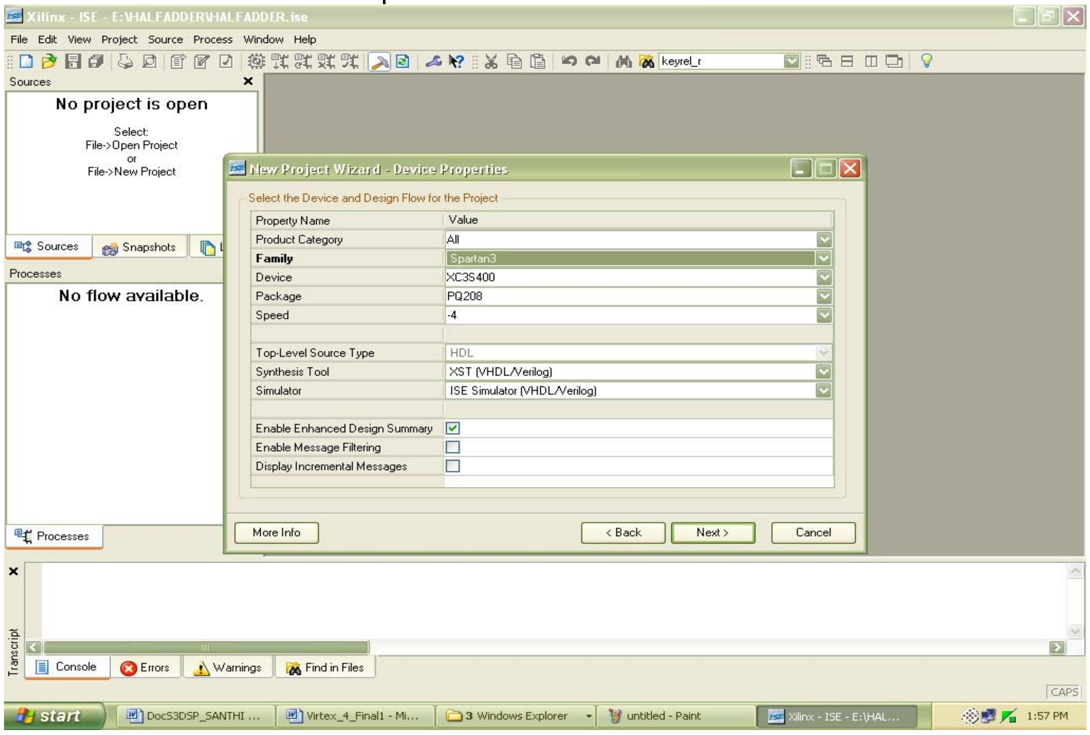
Project Name:	Project Location
HALFADDER	E:\HALFADDER

Select the Type of Top-Level Source for the Project

Top-Level Source Type: HDL

More Info
< Back
Next >
Cancel

Select Device. Use the pull-down arrow to select the Value for each Property Name. Click in the field to access the pull-down list.



Xilinx - ISE - E:\HALFADDER\HALFADDER.ise

No project is open

Select:
File->Open Project
or
File->New Project

No flow available.

New Project Wizard - Device Properties

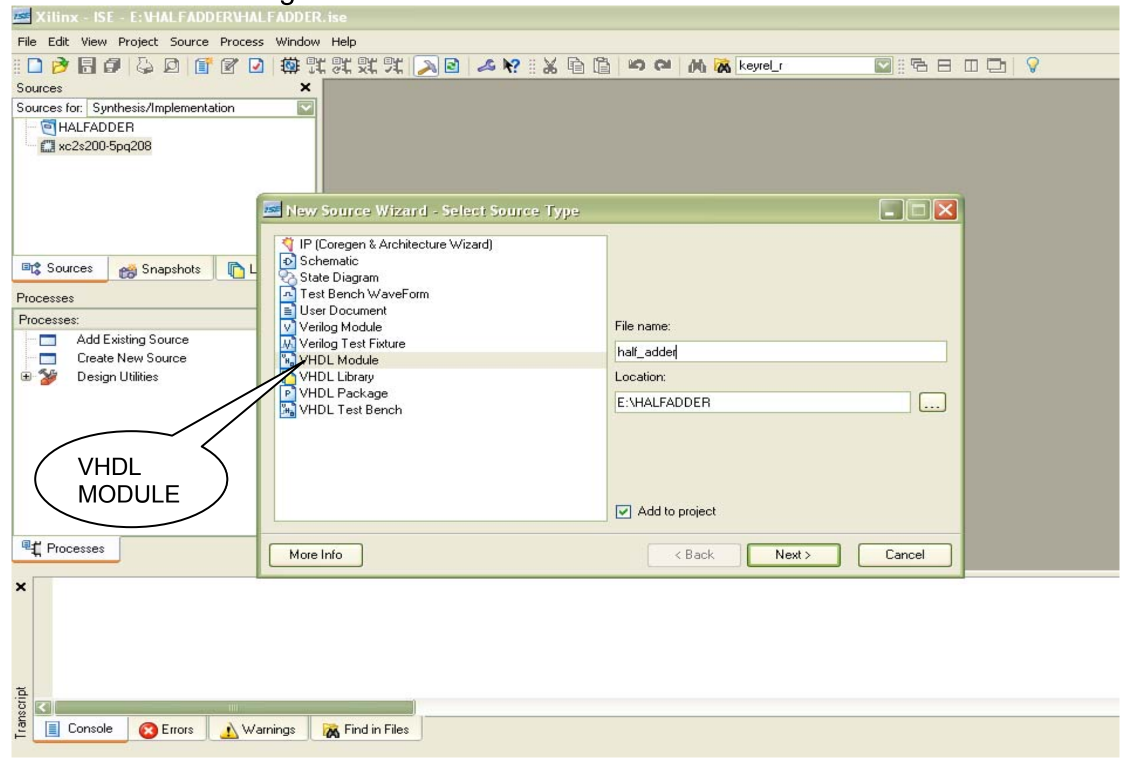
Select the Device and Design Flow for the Project

Property Name	Value
Product Category	All
Family	Spartan3
Device	XC3S400
Package	PQ208
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISE Simulator (VHDL/Verilog)
Enable Enhanced Design Summary	✓
Enable Message Filtering	☐
Display Incremental Messages	☐

Say *FINISH.* Project summary is seen.

Step 3:

Creating a new VHD file
Click on the symbol of FPGA device and then right click→ Click on *new source* →*VHDL module* and give the File name



Xilinx - ISE - E:\HALFADDER\HALFADDER.ise

Sources for: Synthesis/Implementation
HALFADDER
xc2s200-5pq208

Processes:
Add Existing Source
Create New Source
Design Utilities

New Source Wizard - Select Source Type

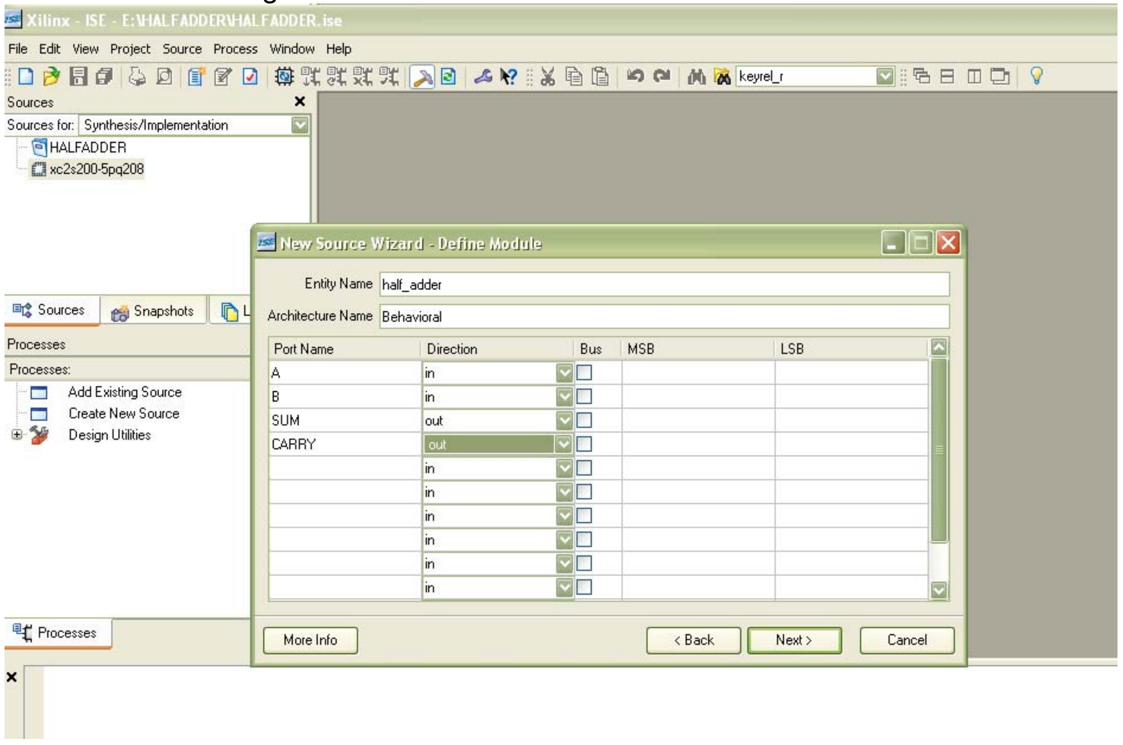
IP (Coregen & Architecture Wizard)
Schematic
State Diagram
Test Bench WaveForm
User Document
Verilog Module
Verilog Test Fixture
VHDL Module
VHDL Library
VHDL Package
VHDL Test Bench

File name: half_adder
Location: E:\HALFADDER
☑ Add to project

Then say *Next→Define ports*.In this case

- *a* and *b* are the input ports defined as *in*
- *sum* and *carry* are output ports defined as *out*

after this say *Next* twice and then *Finish*
Skeleton of the design is shown in the VHDL editor.



Xilinx - ISE - E:\HALFADDER\HALFADDER.ise

Sources for: Synthesis/Implementation
HALFADDER
xc2s200-5pq208

Processes:
Add Existing Source
Create New Source
Design Utilities

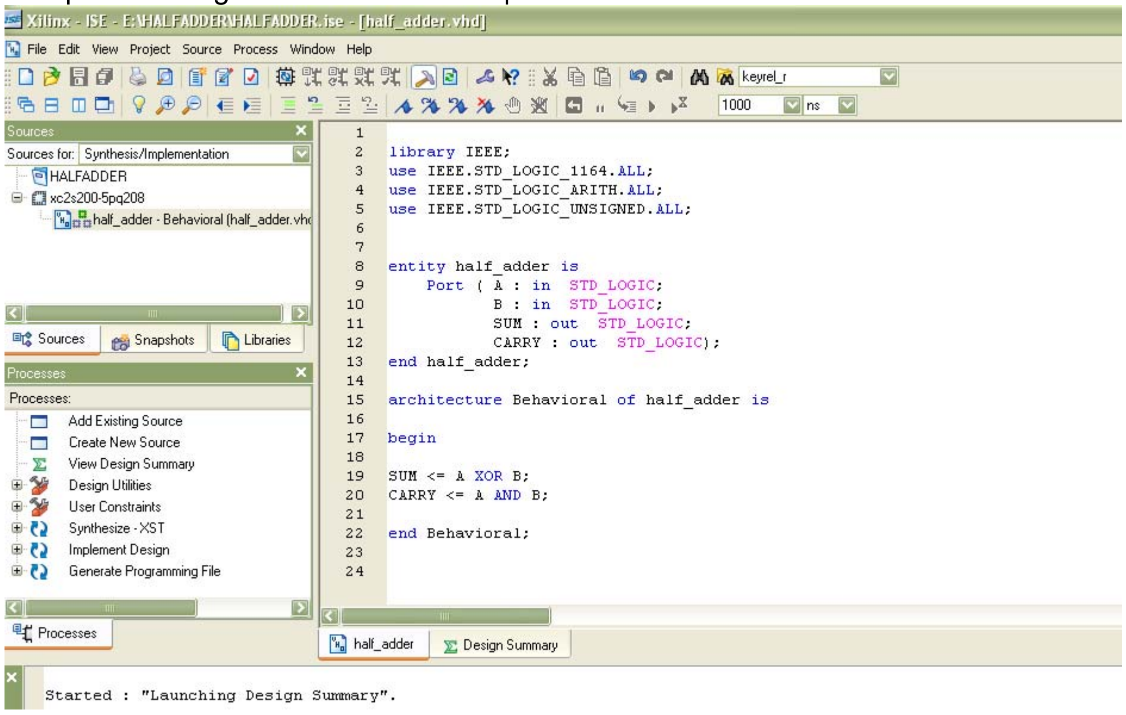
New Source Wizard - Define Module

Entity Name: half_adder
Architecture Name: Behavioral

Port Name	Direction	Bus	MSB	LSB
A	in	☐		
B	in	☐		
SUM	out	☐		
CARRY	out	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		

Step 4:

Writing the Behavioural VHDL Code in VHDL Editor
Sample code is given below for this experiment.



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


entity half_adder is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           SUM : out  STD_LOGIC;
           CARRY : out  STD_LOGIC);
end half_adder;

architecture Behavioral of half_adder is

begin

SUM <= A XOR B;
CARRY <= A AND B;

end Behavioral;
```

```
Started : "Launching Design Summary".
```

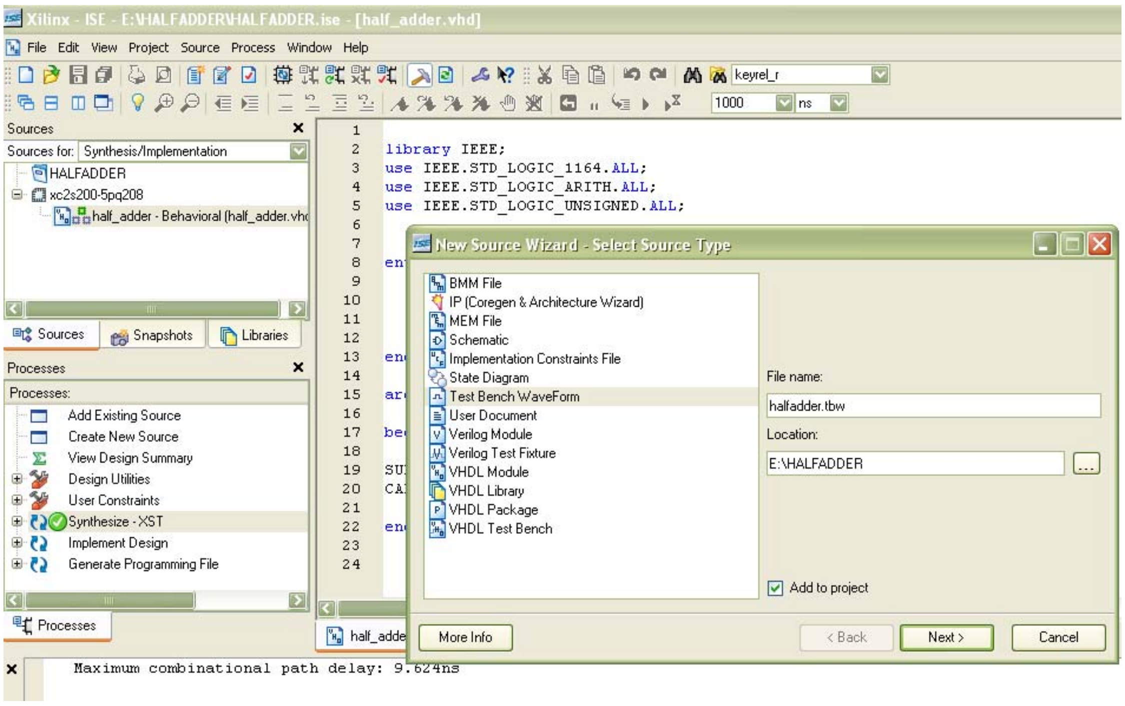
Step 5

Check Syntax
Run the *Check syntax → Process window→ synthesize→check syntax* >, and remove errors if present.

Step 6

Creating a test bench file

Verify the operation of your design before you implement it as hardware. Simulation can be done using ISE simulator. For this click on the symbol of FPGA device and then right click→ *Click on new source →Test Bench Waveform and give the name → Select entity→Finish*.



Xilinx - ISE - E:\HALFADDER\HALFADDER.ise - [half_adder.vhd]

Sources for: Synthesis/Implementation

- HALFADDER
- xc2s200-5pq208
- half_adder - Behavioral (half_adder.vhd)

Processes:

- Add Existing Source
- Create New Source
- View Design Summary
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File

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

New Source Wizard - Select Source Type

- BMM File
- IP (Coregen & Architecture Wizard)
- MEM File
- Schematic
- Implementation Constraints File
- State Diagram
- Test Bench WaveForm
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench

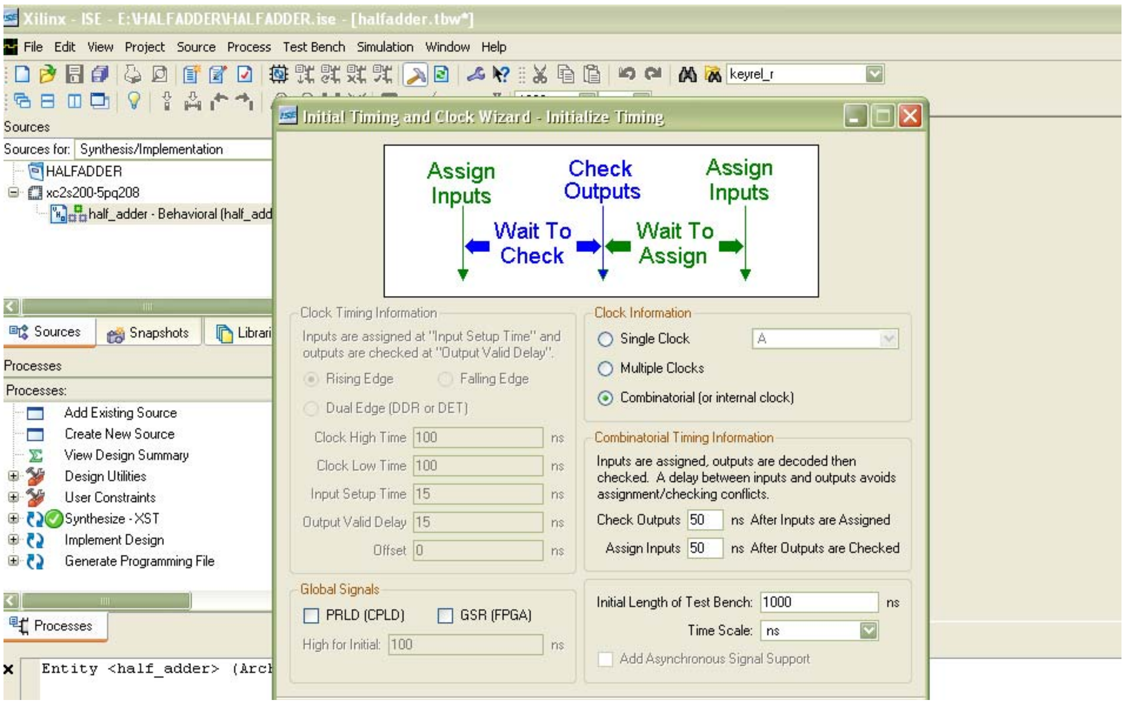
File name: halfadder.tbw

Location: E:\HALFADDER

[✓] Add to project

```
Maximum combinational path delay: 9.624ns
```

Select the desired parameters for simulating your design. In this case *combinational circuit and Simulation time.*



Xilinx - ISE - E:\HALFADDER\HALFADDER.ise - [halfadder.tbw*]

Initial Timing and Clock Wizard - Initialize Timing

Clock Timing Information: Inputs are assigned at "Input Setup Time" and outputs are checked at "Output Valid Delay".

Rising Edge / Falling Edge / Dual Edge (DDR or DET)

Clock High Time 100 ns

Clock Low Time 100 ns

Input Setup Time 15 ns

Output Valid Delay 15 ns

Offset 0 ns

Clock Information: Single Clock / Multiple Clocks / [✓ Combinatorial (or internal clock)]

Combinatorial Timing Information: Inputs are assigned, outputs are decoded then checked. A delay between inputs and outputs avoids assignment/checking conflicts.

Check Outputs 50 ns After Inputs are Assigned

Assign Inputs 50 ns After Outputs are Checked

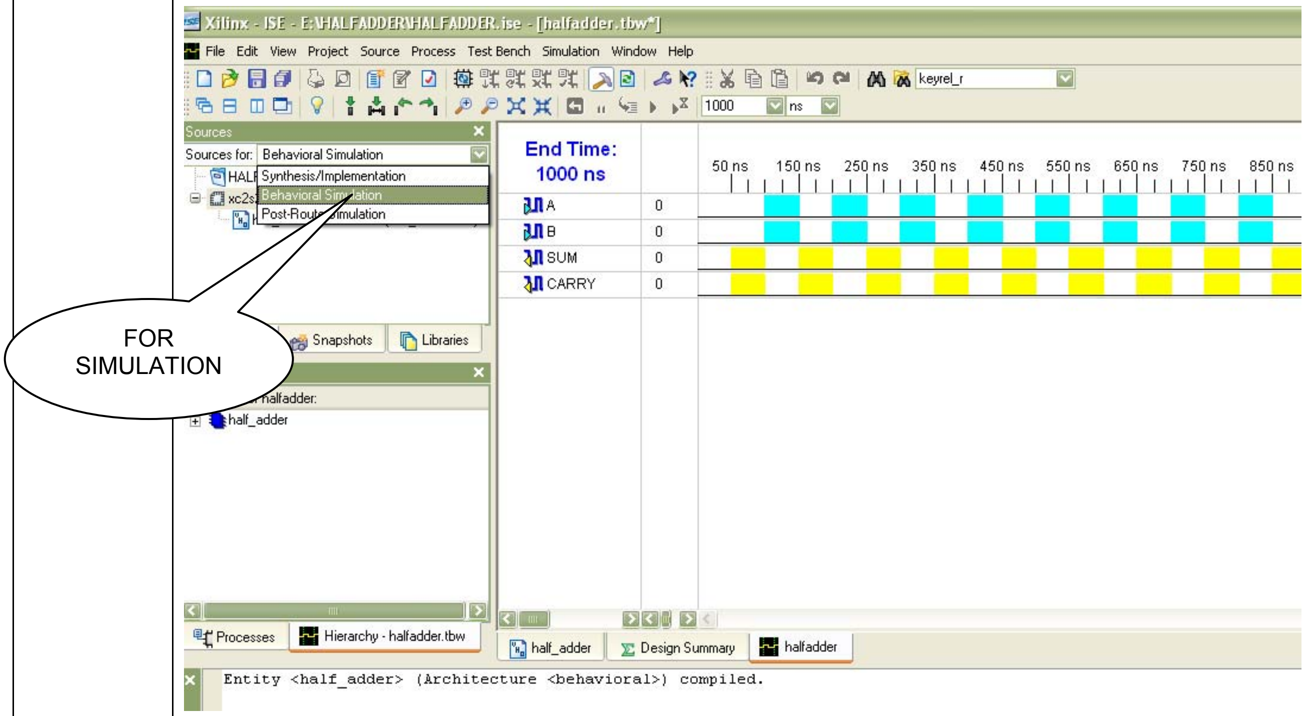
Global Signals: PRLD (CPLD) GSR (FPGA)

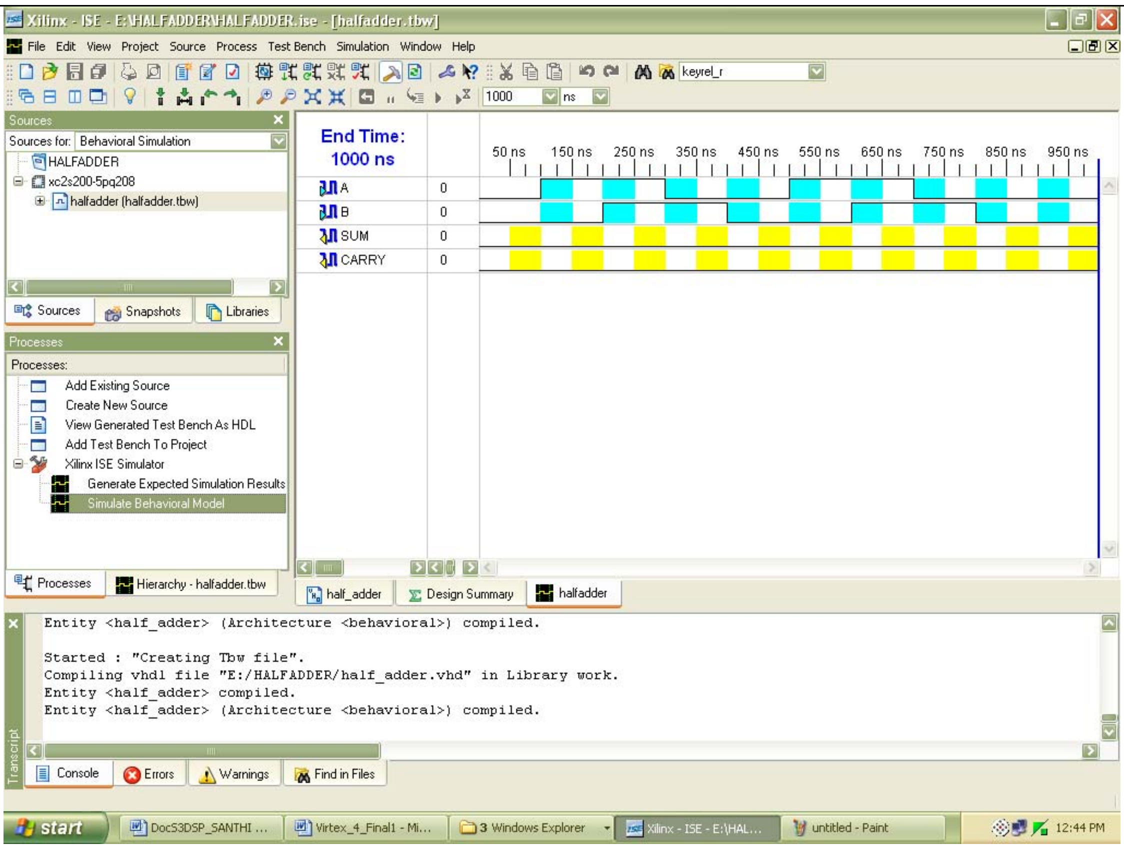
High for Initial: 100 ns

Initial Length of Test Bench: 1000 ns

Time Scale: ns

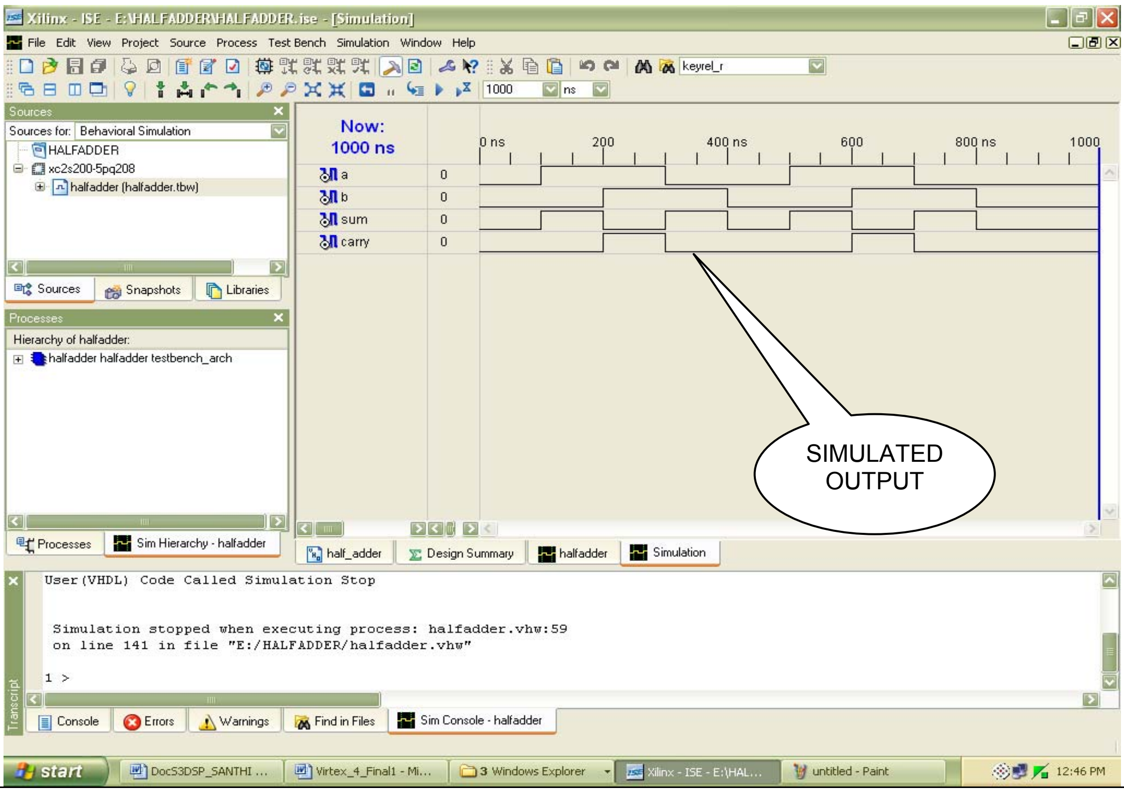
Add Asynchronous Signal Support

Step 7:	*Simulate the code* *Simulation Tools* *ISE tool supports the following simulation tools:* • *HDL Bencher is an automated test bench creation tool. It is fully integrated with Project Navigator.* • *ModelSim from Model Technology, Inc., is integrated in Project Navigator to simulate the design at all steps (Functional and Timing). ModelSim XE, the Xilinx Edition of Model Technology, Inc.'s ModelSim application, can be installed from the MTI CD included in your ISE Tool* In source Window from the Drop-down menu select *Behavioural Simulation* to view the created test Bench file.  Xilinx - ISE - E:\HALFADDER\HALFADDER.ise - [halfadder.tbw*] File Edit View Project Source Process Test Bench Simulation Window Help Sources for: Behavioral Simulation Synthesis/Implementation Behavioral Simulation Post-Route Simulation End Time: 1000 ns A 0 B 0 SUM 0 CARRY 0 Processes Hierarchy - halfadder.tbw half_adder Design Summary halfadder FOR SIMULATION Entity <half_adder> (Architecture <behavioral>) compiled. Click on test bench file. Test bench file will open in main window. Assign all the signals and save File. From the source of process window. Click on *Simulate Behavioral Model* in Process window.


Xilinx - ISE - E:\HALFADDER\HALFADDER.ise - [halfadder.tbw]
File Edit View Project Source Process Test Bench Simulation Window Help
Sources for: Behavioral Simulation
HALFADDER
xc2s200-5pq208
halfadder (halfadder.tbw)
Sources Snapshots Libraries
Processes:
Add Existing Source
Create New Source
View Generated Test Bench As HDL
Add Test Bench To Project
Xilinx ISE Simulator
Generate Expected Simulation Results
Simulate Behavioral Model
Processes Hierarchy - halfadder.tbw
End Time: 1000 ns
A 0
B 0
SUM 0
CARRY 0
half_adder Design Summary halfadder
Console Errors Warnings Find in Files

```
Entity <half_adder> (Architecture <behavioral>) compiled.

Started : "Creating Tbw file".
Compiling vhdl file "E:/HALFADDER/half_adder.vhd" in Library work.
Entity <half_adder> compiled.
Entity <half_adder> (Architecture <behavioral>) compiled.
```

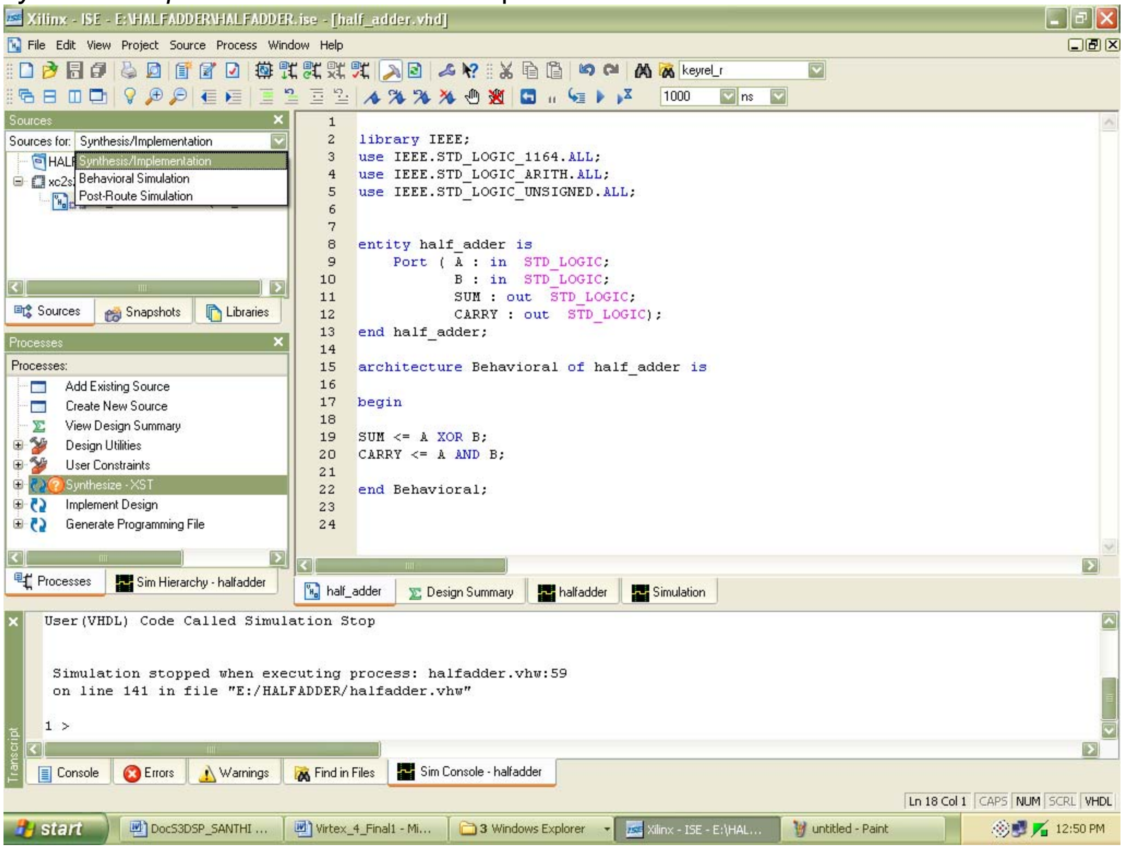
Verify your design in wave window by seeing behaviour of output signal with respect to input signal. Close the ISE simulator window


Xilinx - ISE - E:\HALFADDER\HALFADDER.ise - [Simulation]
File Edit View Project Source Process Test Bench Simulation Window Help
Sources for: Behavioral Simulation
HALFADDER
xc2s200-5pq208
halfadder (halfadder.tbw)
Hierarchy of halfadder:
halfadder halfadder testbench_arch
Processes Sim Hierarchy - halfadder
Now: 1000 ns
a 0
b 0
sum 0
carry 0
SIMULATED OUTPUT
half_adder Design Summary halfadder Simulation
Console Errors Warnings Find in Files Sim Console - halfadder

```
User(VHDL) Code Called Simulation Stop


 Simulation stopped when executing process: halfadder.vhw:59
 on line 141 in file "E:/HALFADDER/halfadder.vhw"

1 >
```

Step 8: *Synthesize the design using XST.*
Translate your design into gates and optimize it for the target architecture. This is the

synthesis phase.
Again for synthesizing your design, from the source window select, *synthesis/Implementation* from the drop-down menu.

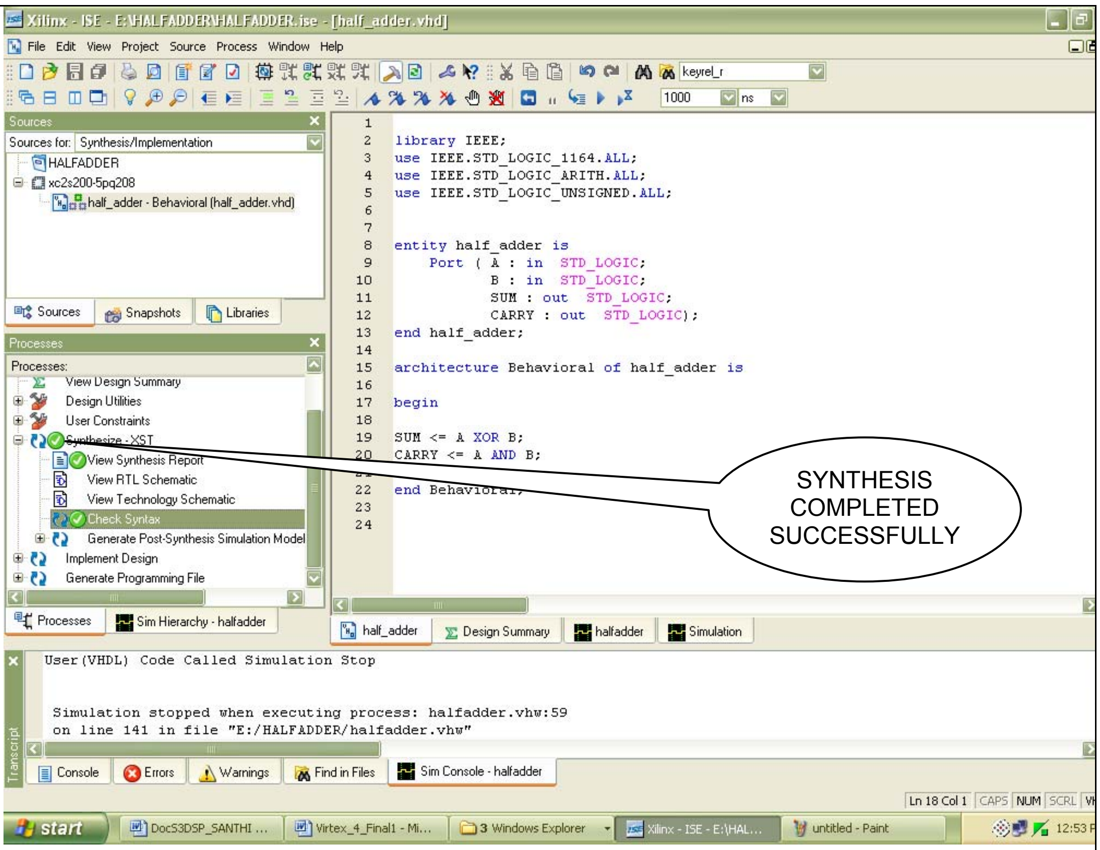


```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity half_adder is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           SUM : out  STD_LOGIC;
           CARRY : out  STD_LOGIC);
end half_adder;

architecture Behavioral of half_adder is

begin

SUM <= A XOR B;
CARRY <= A AND B;

end Behavioral;
```

```
User(VHDL) Code Called Simulation Stop


 Simulation stopped when executing process: halfadder.vhw:59
 on line 141 in file "E:/HALFADDER/halfadder.vhw"

1 >
```

Highlight file in the Sources in Project window. To run synthesis, right-click on Synthesize, and the Run option, *or* double-click on Synthesize in the Processes for Current Source window. Synthesis will run, and

- a green check ✓will appear next to Synthesize when it is successfully completed.
- a red cross ✘indicates an error was generated and
- a yellow exclamation ! mark indicates that a warning was generated, (warnings are OK).
  Check the synthesis report.
  If there are any errors correct it and rerun synthesis..

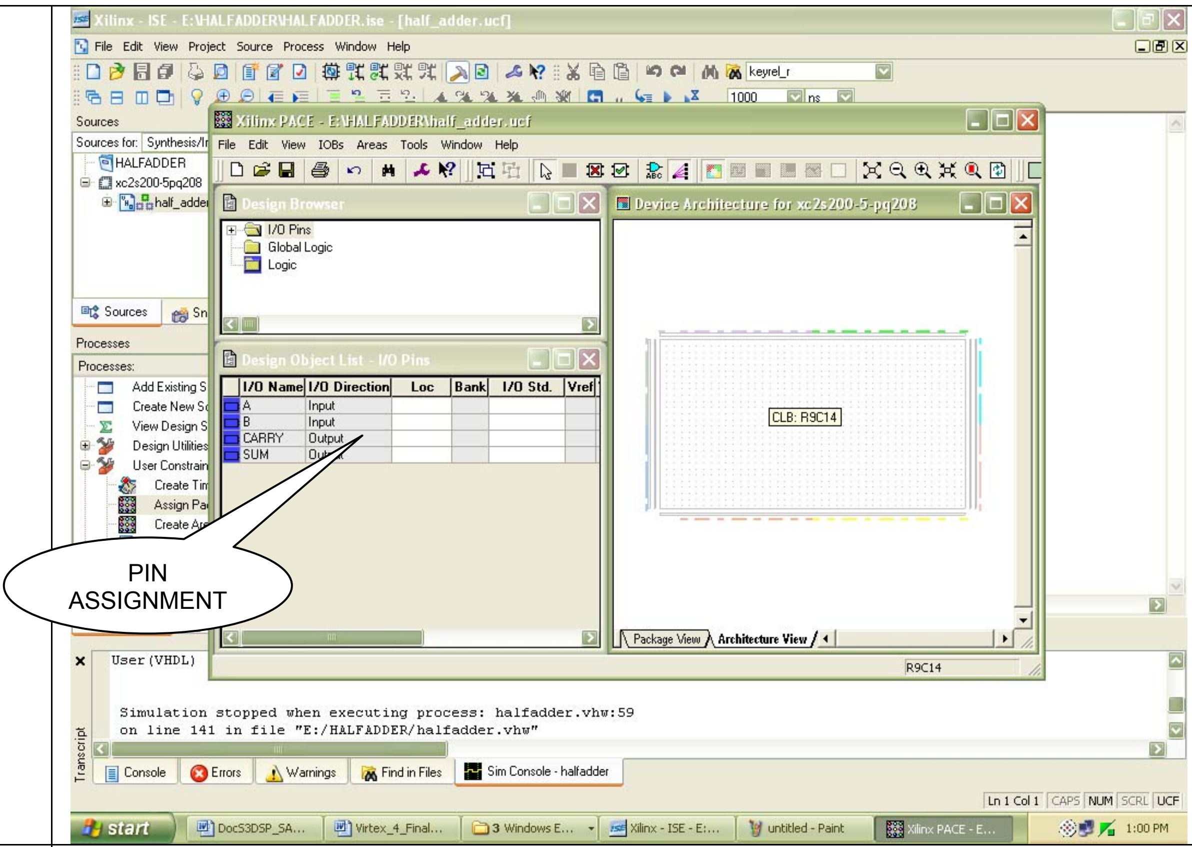

Xilinx - ISE - E:\HALFADDER\HALFADDER.ise - [half_adder.vhd]
File Edit View Project Source Process Window Help
Sources for: Synthesis/Implementation
HALFADDER
xc2s200-5pq208
half_adder - Behavioral (half_adder.vhd)
Processes:
View Design Summary
Design Utilities
User Constraints
Synthesize - XST
View Synthesis Report
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Generate Programming File
SYNTHESIS COMPLETED SUCCESSFULLY

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


entity half_adder is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           SUM : out  STD_LOGIC;
           CARRY : out  STD_LOGIC);
end half_adder;

architecture Behavioral of half_adder is

begin

SUM <= A XOR B;
CARRY <= A AND B;

end Behavioral;
```

```
User(VHDL) Code Called Simulation Stop

 Simulation stopped when executing process: halfadder.vhw:59
 on line 141 in file "E:/HALFADDER/halfadder.vhw"
```

Step 9:

Create Constraints File(UCF)
Click on the symbol of FPGA device and then right click→ Click on new source →Implementation Constraints File and give the name → Select entity→Finish.
Click on *User Constraint and* in that Double Click on *Assign Package Pins* option in Process window. Xilinx PACE window opens. Enter all the pin assignments in PACE., depending upon target device and number of input and outputs used in your design.
(sample code is given below for given design.)


Xilinx - ISE - E:\HALFADDER\HALFADDER.ise - [half_adder.ucf]
Xilinx PACE - E:\HALFADDER\half_adder.ucf
Design Browser
I/O Pins
Global Logic
Logic
Device Architecture for xc2s200-5-pq208
CLB: R9C14
Package View
Architecture View
R9C14
PIN ASSIGNMENT

Design Object List - I/O Pins

I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref
A	Input				
B	Input				
CARRY	Output				
SUM	Output				

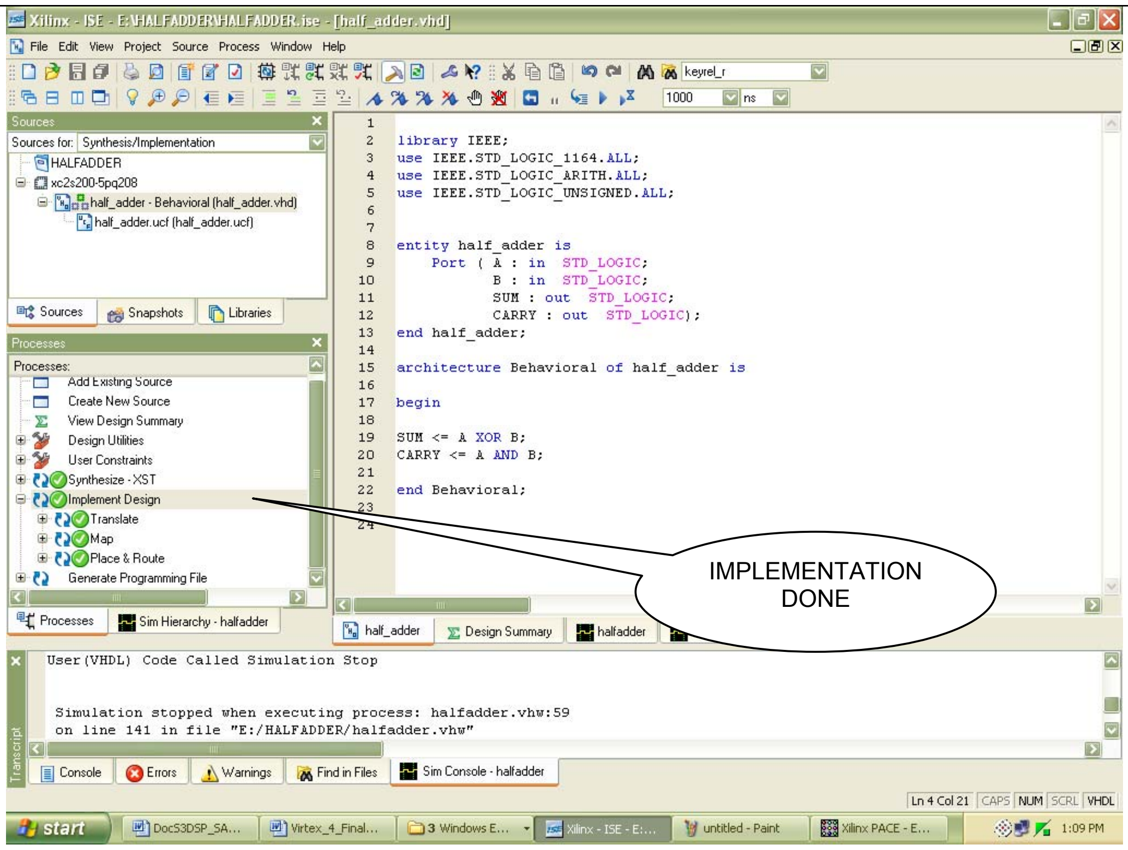
```
User (VHDL)

Simulation stopped when executing process: halfadder.vhw:59
on line 141 in file "E:/HALFADDER/halfadder.vhw"
```

Step 10: *Implementing a Design*

Once synthesis is complete, you can place and route your design to fit into a Xilinx device, and you can also get some post place-and-route timing information about the design. The implementation stage consists of taking the synthesized netlist through translation, mapping, and place and route.

To check your design as it is implemented, reports are available for each stage in the implementation process. Use the Xilinx Constraints Editor to add timing and location constraints for the implementation of your design. This procedure runs you through the basic flow for implementation.

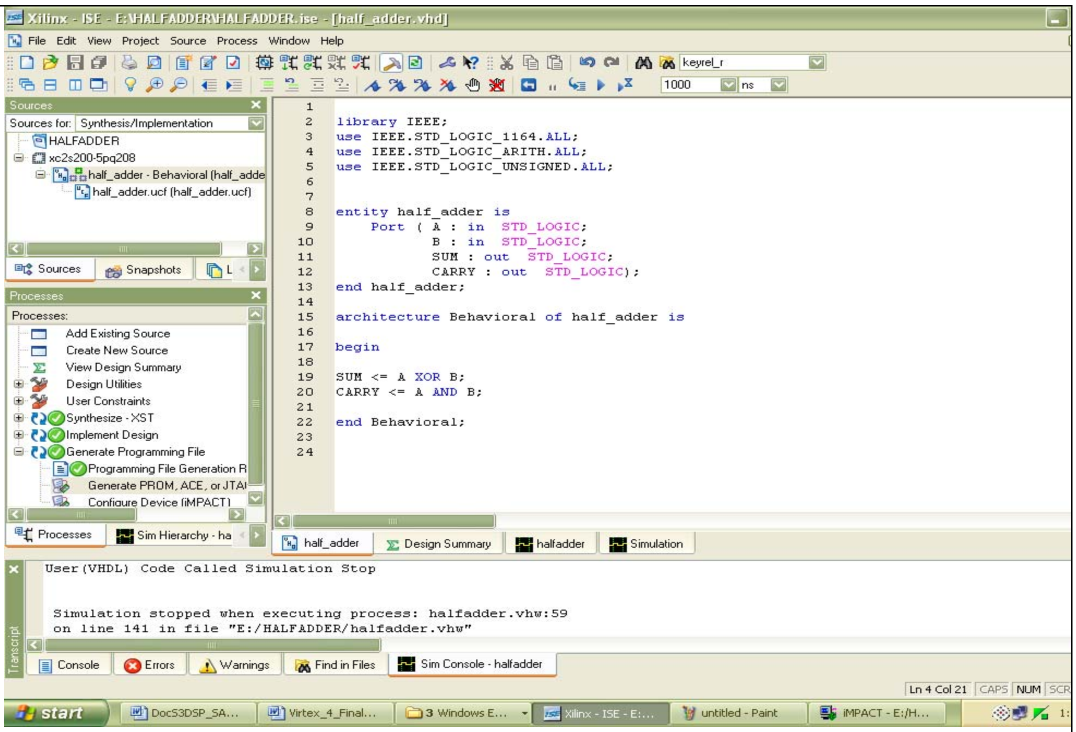
Right-click on *Implement Design*, and choose the Run option, *or* double left-click on *Implement Design.*



```

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


entity half_adder is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           SUM : out  STD_LOGIC;
           CARRY : out  STD_LOGIC);
end half_adder;

architecture Behavioral of half_adder is

begin

SUM <= A XOR B;
CARRY <= A AND B;

end Behavioral;
```

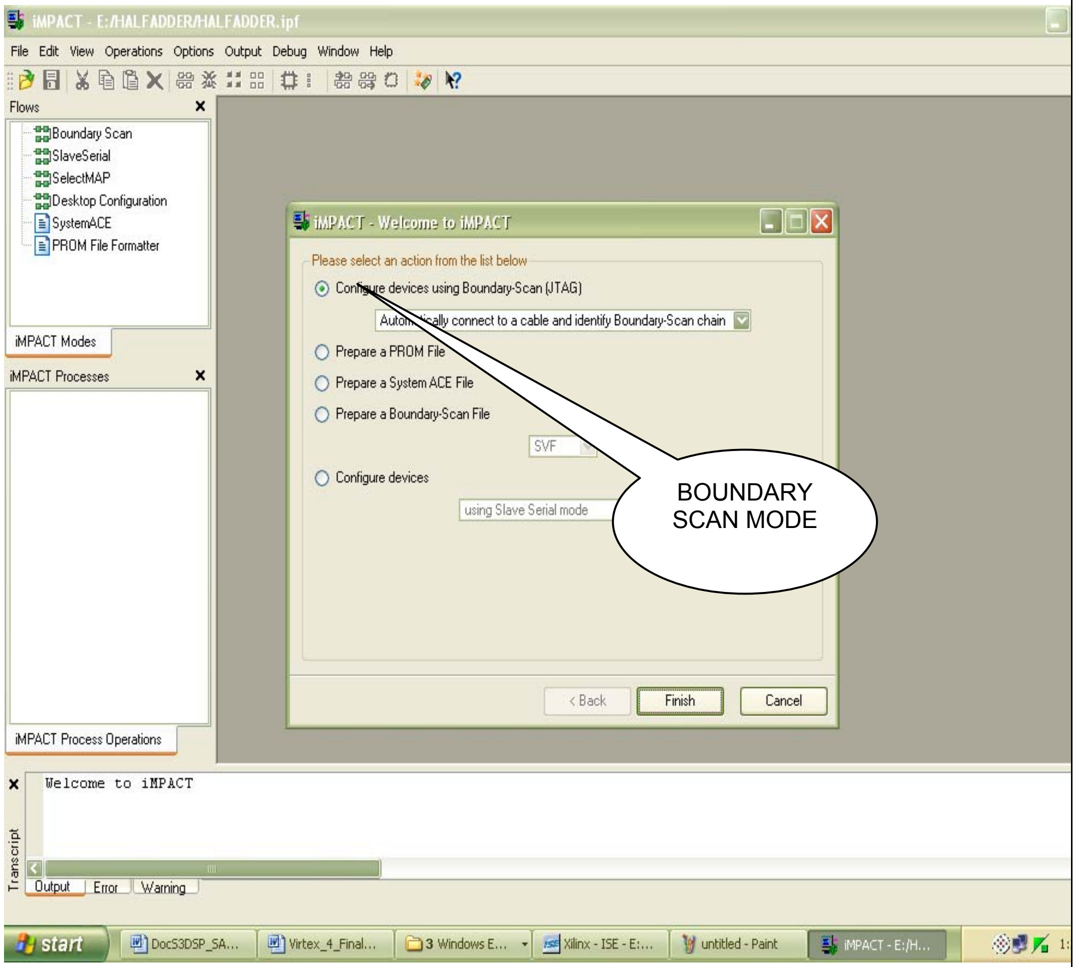
```
User(VHDL) Code Called Simulation Stop


 Simulation stopped when executing process: halfadder.vhw:59
 on line 141 in file "E:/HALFADDER/halfadder.vhw"
```

Step 11:	*Generating Programming File* Right-click on *Generate Programming File,* choose the Run option, *or* double left-click on *Generate Programming File***.** This will generate the Bit stream
Step 12	*Downloading in* Boundary Scan Mode. *Note : Xilinx provides 2-tools for downloading purpose, viz.* • *iMPACT - is a command line and GUI based tool* • *PROM File Formatter*


Xilinx - ISE - E:\HALFADDER\HALFADDER.ise - [half_adder.vhd]
File Edit View Project Source Process Window Help
Sources
Sources for: Synthesis/Implementation
HALFADDER
xc2s200-5pq208
half_adder - Behavioral (half_adde
half_adder.ucf (half_adder.ucf)
Sources
Snapshots
Processes
Add Existing Source
Create New Source
View Design Summary
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Programming File Generation R
Generate PROM, ACE, or JTAG
Configure Device (iMPACT)
Sim Hierarchy - ha
half_adder
Design Summary
halfadder
Simulation
Console
Errors
Warnings
Find in Files
Sim Console - halfadder
Ln 4 Col 21
start

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


entity half_adder is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           SUM : out  STD_LOGIC;
           CARRY : out  STD_LOGIC);
end half_adder;

architecture Behavioral of half_adder is

begin

SUM <= A XOR B;
CARRY <= A AND B;

end Behavioral;
```

```
User(VHDL) Code Called Simulation Stop


 Simulation stopped when executing process: halfadder.vhw:59
 on line 141 in file "E:/HALFADDER/halfadder.vhw"
```

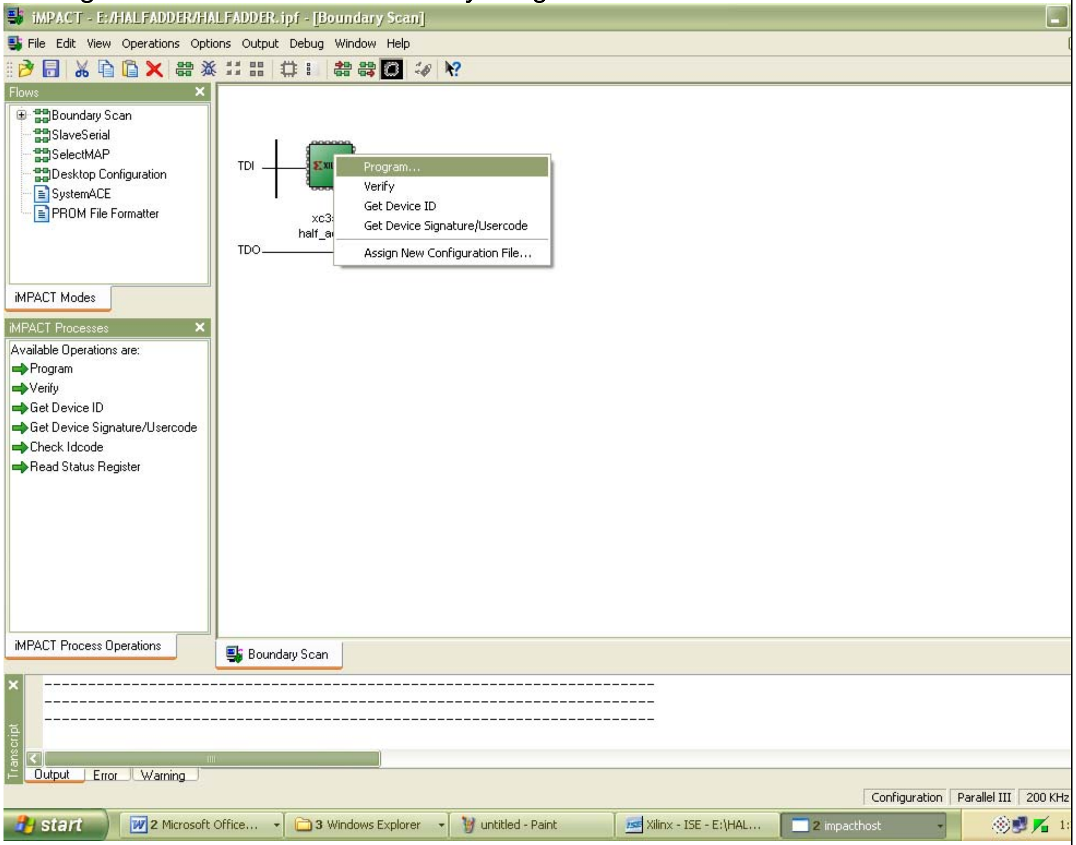


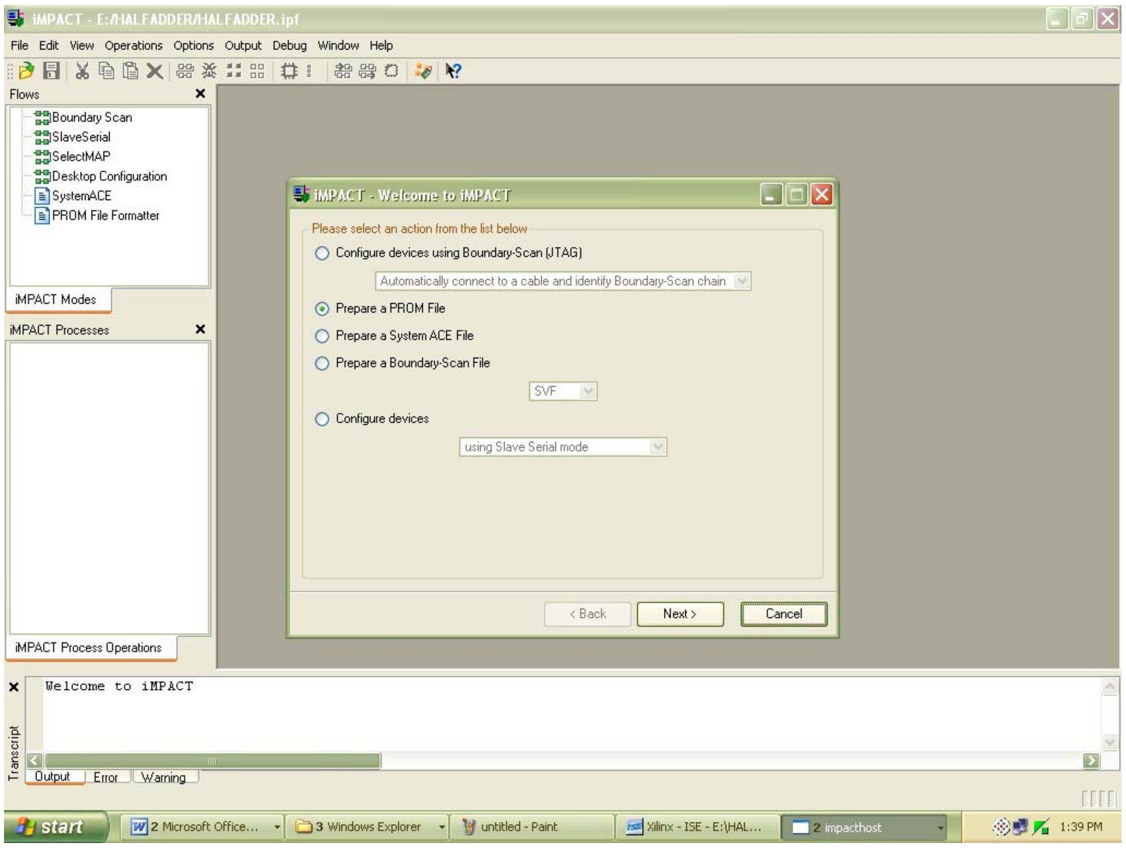
Please select an action from the list below

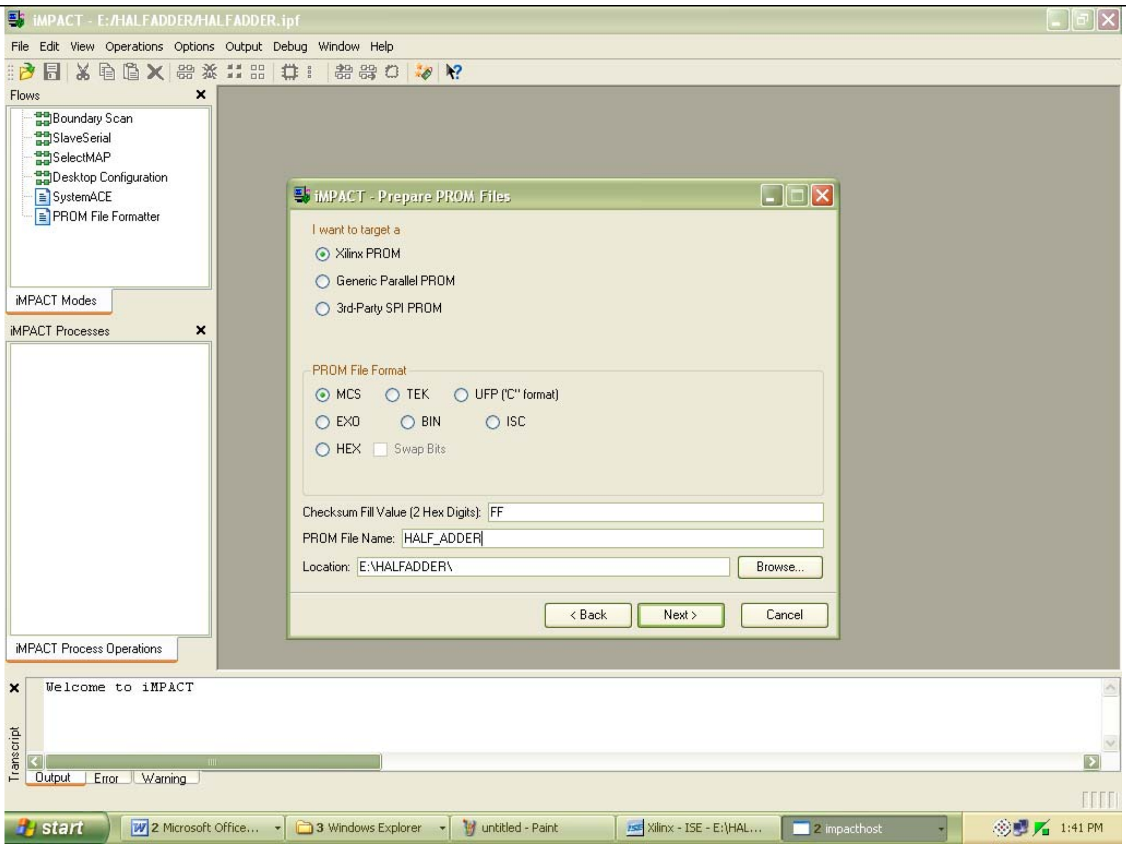
- [✓ Configure devices using Boundary-Scan (JTAG)]
  - Automatically connect to a cable and identify Boundary-Scan chain
- Prepare a PROM File
- Prepare a System ACE File
- Prepare a Boundary-Scan File
  - SVF
- Configure devices
  - using Slave Serial mode

```
Welcome to iMPACT
```

Procedure for downloading using iMPACT

	• Boundary Scan Mode 1. Right click on “*Configure Device (iMPACT)” -> and Say RUN* or Double click on “*Configure Device (iMPACT)”.* 2. Right click in workspace and say *Initialize chain* .The device is seen. 3. Right click on the device and say *Program.*  iMPACT - E:/HALFADDER/HALFADDER.ipf - [Boundary Scan] File Edit View Operations Options Output Debug Window Help Flows Boundary Scan SlaveSerial SelectMAP Desktop Configuration SystemACE PROM File Formatter iMPACT Modes iMPACT Processes Available Operations are: Program Verify Get Device ID Get Device Signature/Usercode Check Idcode Read Status Register iMPACT Process Operations TDI TDO xc3 half_a Program... Verify Get Device ID Get Device Signature/Usercode Assign New Configuration File... Boundary Scan Transcript Output Error Warning Configuration Parallel III 200 KHz start 2 Microsoft Office... 3 Windows Explorer untitled - Paint Xilinx - ISE - E:\HAL... 2 impacthost If the device is programmed properly, it says *Programming Succeeded* or else. *Programming Failed.* The *DONE* Led glows green if programming succeeds.
	Note: Before downloading make sure that Protoboard is connected to PC's parallel port with the cable provided and power to the Protoboard is ON.
Step 13:	Apply input through DIP Switches, output is displayed on LEDs
Step 14:	Configuration through PROM: *Generating PROM file*: FPGA can also be configured in Master Serial Mode through PROM. For this you need to program the PROM through a .mcs file.

	Right click on "*Generate PROM,ACE or JTAG file" -> and Say RUN* or Double click on "*Generate PROM,ACE or JTAG file"*  iMPACT - Welcome to iMPACT. Please select an action from the list below: ( ) Configure devices using Boundary-Scan (JTAG) — Automatically connect to a cable and identify Boundary-Scan chain; [✓ Prepare a PROM File]; ( ) Prepare a System ACE File; ( ) Prepare a Boundary-Scan File — SVF; ( ) Configure devices — using Slave Serial mode. Transcript: Welcome to iMPACT Specify the PROM file name and location where it is to be generated.



iMPACT - Prepare PROM Files

I want to target a
[✓ Xilinx PROM] Generic Parallel PROM 3rd-Party SPI PROM

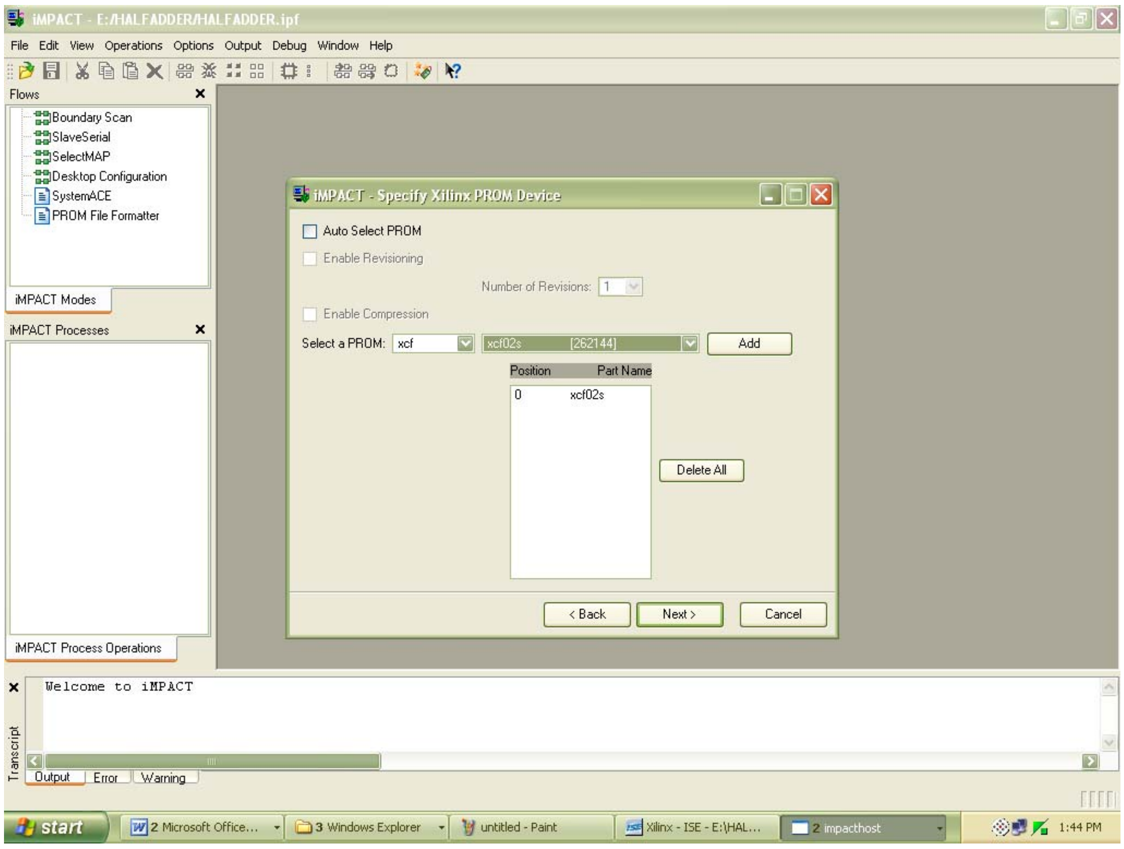
PROM File Format
[✓ MCS] TEK UFP ('C' format) EXO BIN ISC HEX Swap Bits

Checksum Fill Value (2 Hex Digits): FF
PROM File Name: HALF_ADDER
Location: E:\HALFADDER\

< Back Next > Cancel Browse...

```
Welcome to iMPACT
```

Specify the desired parameters of the PROM on board and say *ADD* then *FINISH*



iMPACT - Specify Xilinx PROM Device

Auto Select PROM
Enable Revisioning
Number of Revisions: 1
Enable Compression
Select a PROM: xcf xcf02s [262144] Add

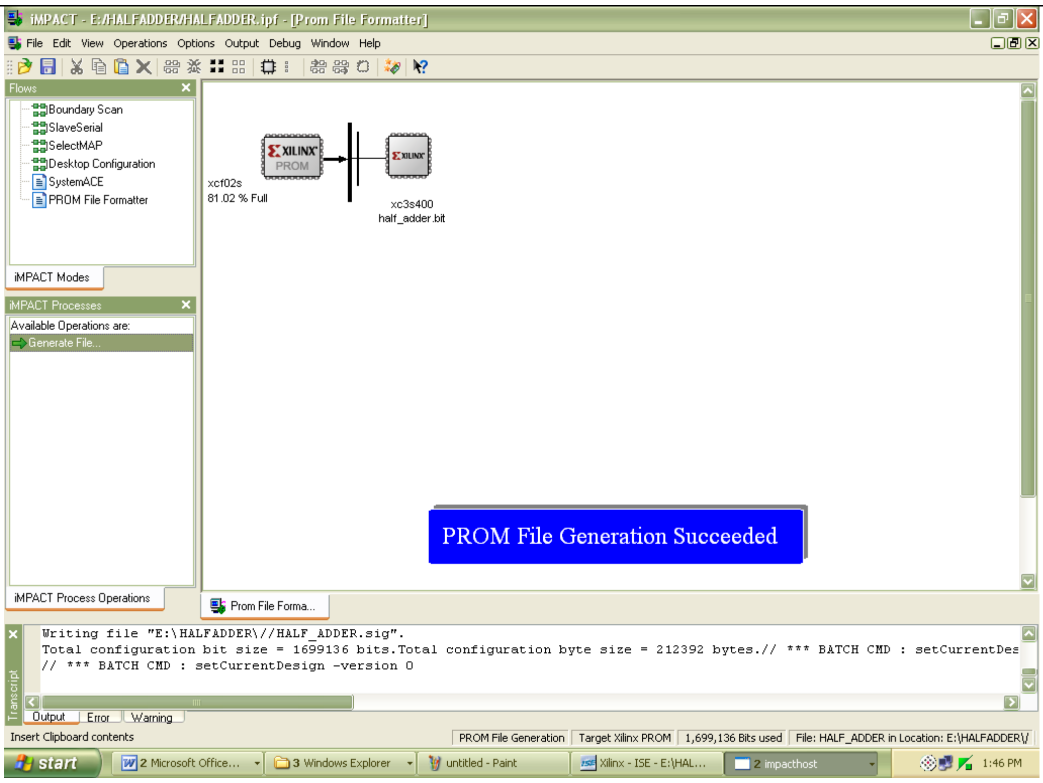
Position	Part Name
0	xcf02s

Delete All

< Back Next > Cancel

```
Welcome to iMPACT
```

Say *Generate File* from the Process Window.

iMPACT - E:/HALFADDER/HALFADDER.ipf - [Prom File Formatter]

PROM File Generation Succeeded

```
Writing file "E:\HALFADDER\//HALF_ADDER.sig".
Total configuration bit size = 1699136 bits.Total configuration byte size = 212392 bytes.// *** BATCH CMD : setCurrentDes
// *** BATCH CMD : setCurrentDesign -version 0
```

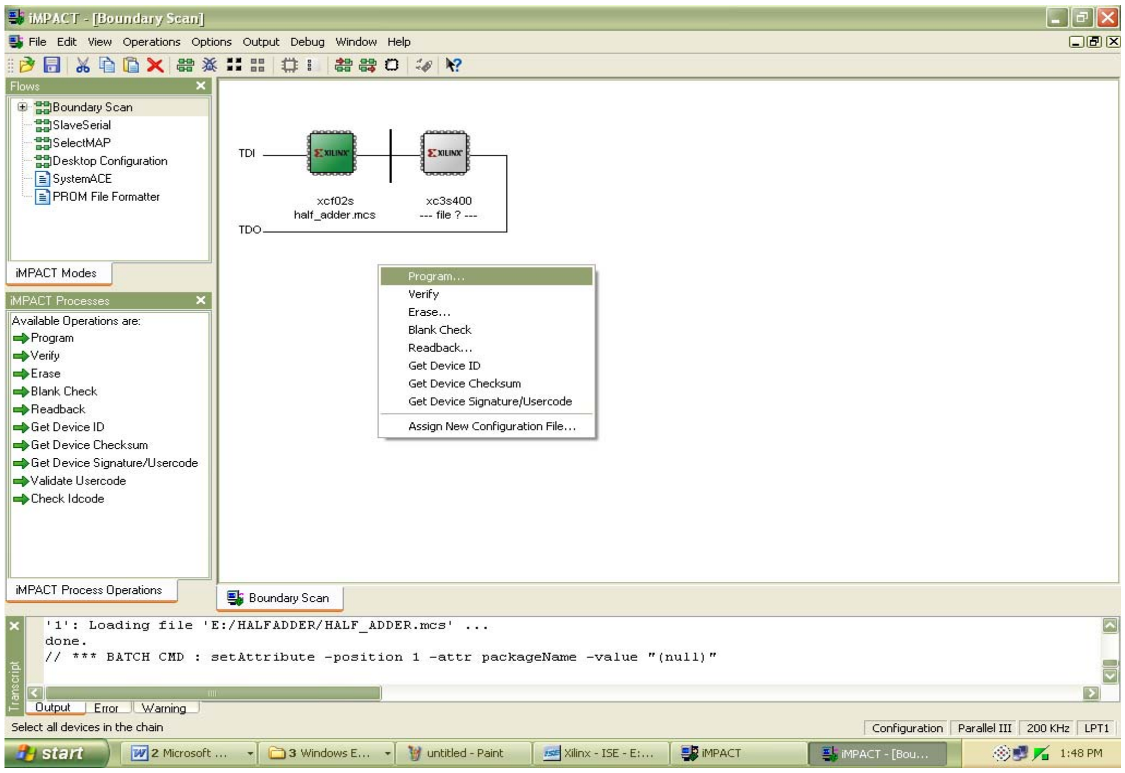
Insert Clipboard contents | PROM File Generation | Target Xilinx PROM | 1,699,136 Bits used | File: HALF_ADDER in Location: E:\HALFADDER\/

PROGRAMMING THE PROM

Note: Check the Jumper setting on the board. Refer the Chapter jumper Setting

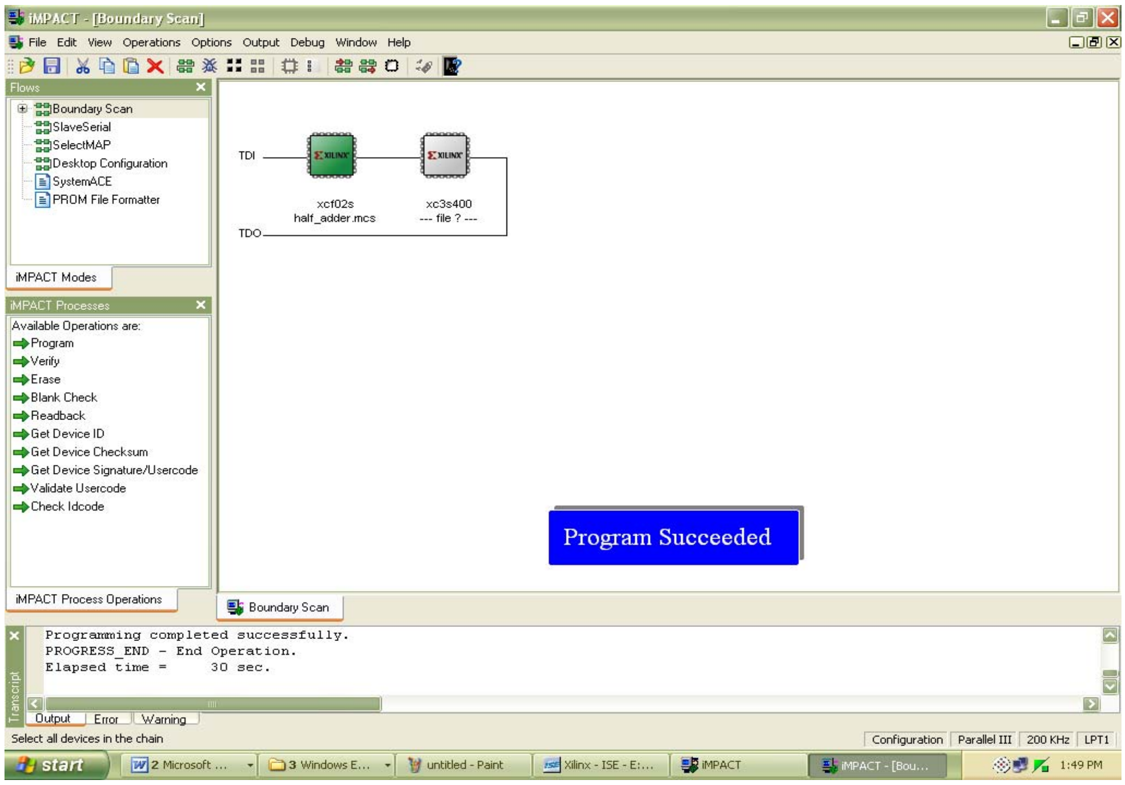
Similar to Step 12.Initialize chain through iMPACT. PROM and FPGA devices on board are seen .Assign the generated mcs file and bit file as desired.

Right click the PROM symbol and say *PROGRAM.*



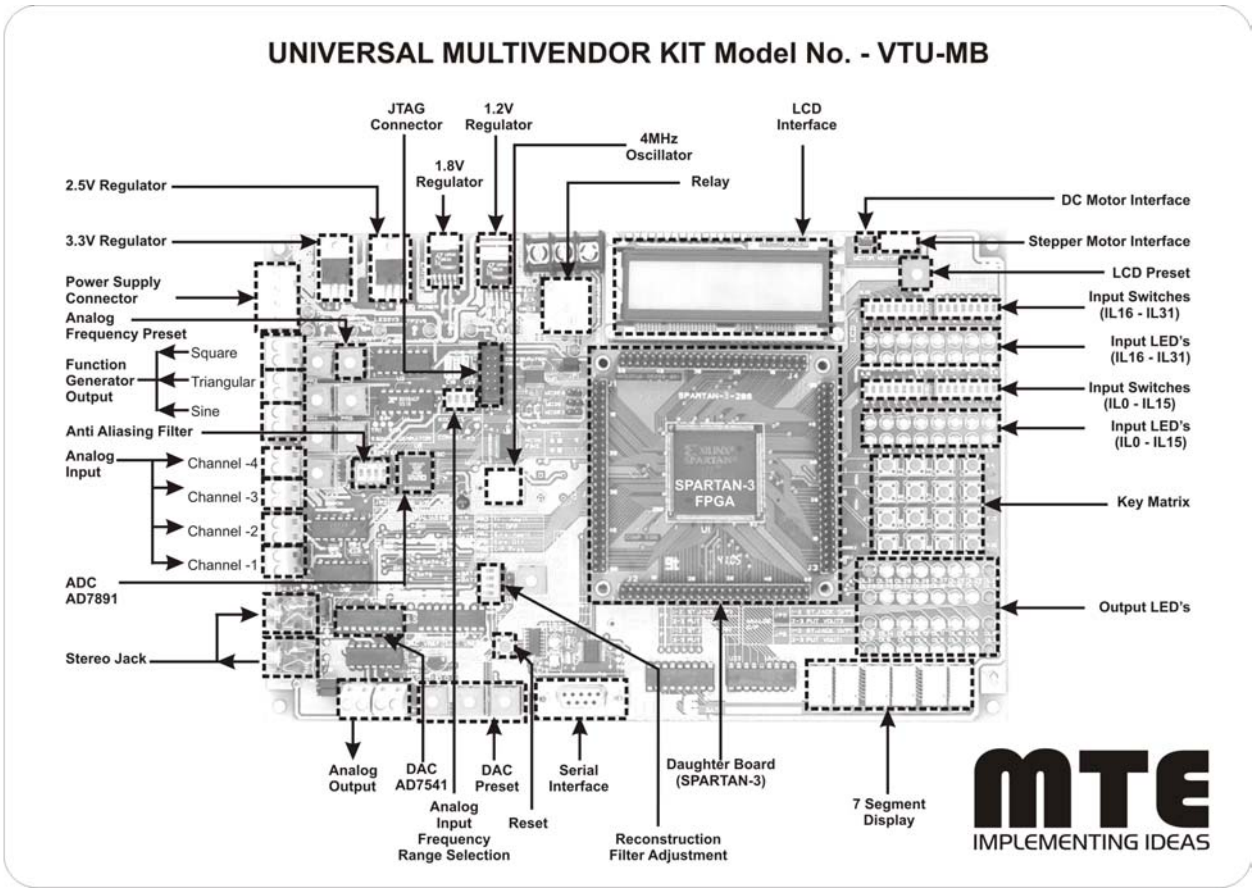
```
'1': Loading file 'E:/HALFADDER/HALF_ADDER.mcs' ...
done.
// *** BATCH CMD : setAttribute -position 1 -attr packageName -value "(null)"
```

Now, whenever the board is powered on in master serial mode, FPGA is configured through PROM automatically.



```
Programming completed successfully.
PROGRESS_END - End Operation.
Elapsed time =      30 sec.
```

Component Diagram Details



Above figure represents component diagram for New VTU Development board.