



Universal VLSI Protoboard

ni logic Pvt. Ltd.,

25/B5, Bandal Complex,

Paud Road, Kothrud,

Pune – 411 038.

Maharashtra.

Tele-Fax: + 91 - 20 - 2528 6948

info@ni2designs.com

www.ni2designs.com

Universal VLSI Protoboard

User Manual

Table of contents

1. Introduction
 - a. Introduction to Programmable logic and PLDs
 - b. Product Introduction
2. Features and Specifications
3. Diagrams
 - a. I/O Connections for Xilinx FPGA
 - b. I/O Connections for Altera FPGA
 - c. I/O Connections for CPLDs
 - d. Baseboard legend diagram
4. Precautions
5. Design Flow
 - a. PLD design flow
 - b. Xilinx webpack design flow
 - c. Altera Quartus-II design flow
6. Configuration/Downloading
 - a. Types of Modes
 - b. Jumper setting for mode selection
7. Pin Assignments
8. Jumper settings
9. Sample Codes
10. Glossary

Chapter 1: Introduction

1.a What is Programmable Logic?

In the world of digital electronic systems, there are three basic kinds of devices: memory, microprocessors, and logic. Memory devices store random information such as the contents of a spreadsheet or database. Microprocessors execute software instructions to perform a wide variety of tasks such as running a word processing program or video game. Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform.

Fixed Logic versus Programmable Logic

Logic devices can be classified into two broad categories - fixed and programmable. As the name suggests, the circuits in a fixed logic device are permanent, they perform one function or set of functions - once manufactured, they cannot be changed. On the other hand, programmable logic devices (PLDs) are standard, off-the-shelf parts that offer customers a wide range of logic capacity, features, speed, and voltage characteristics - and these devices can be changed at any time to perform any number of functions.

With fixed logic devices, the time required to go from design, to prototypes, to a final manufacturing run can take from several months to more than a year, depending on the complexity of the device. And, if the device does not work properly, or if the requirements change, a new design must be developed. The up-front work of designing and verifying fixed logic devices involves substantial "non-recurring engineering" costs, or NRE. These NRE costs can run from a few hundred thousand to several million dollars.

With programmable logic devices, designers use inexpensive software tools to quickly develop, simulate, and test their designs. Then, a design can be quickly programmed into a device, and immediately tested in a live circuit. There are no NRE costs and the final design is completed much faster than that of a custom, fixed logic device.

Another key benefit of using PLDs is that during the design phase customers can change the circuitry as often as they want until the design operates to their satisfaction. That's because PLDs are based on re-writable memory technology - to change the design, the device is simply reprogrammed. Once the design is final, customers can go into immediate production by simply programming as many PLDs as they need with the final software design file.

CPLDs and FPGAs

The two major types of programmable logic devices are field programmable gate arrays (FPGAs) and complex programmable logic devices (CPLDs). Of the two, FPGAs offer the highest amount of logic density, the most features, and the highest performance. The largest FPGA provides millions of "system gates" (the relative density of logic). These advanced devices also offer features such as built-in hardwired IP cores (such as the IBM Power PC, PCI cores, microcontrollers, peripherals, etc), substantial amounts of memory, clock management systems, and support for many of the latest, very fast device-to-device signaling technologies. FPGAs are used in a wide variety of applications ranging from data processing and storage, to instrumentation, telecommunications, and digital signal processing.

CPLDs, by contrast, offer much smaller amounts of logic - up to about 10,000 gates. But CPLDs offer very predictable timing characteristics and are therefore ideal for critical control applications. Low power CPLDs are also available and are very inexpensive, making them ideal for cost-sensitive, battery-operated, portable applications such as mobile phones and digital handheld assistants.

The PLD Advantage

Fixed logic devices and PLDs both have their advantages. Fixed logic devices, for example, are often more appropriate for large volume applications because they can be mass-produced more economically. For certain applications where the very highest performance is required, fixed logic devices may also be the best choice.

However, programmable logic devices offer a number of important advantages over fixed logic devices, including:

- ? PLDs offer customers much more flexibility during the design cycle because design iterations are simply a matter of changing the programming file, and the results of design changes can be seen immediately in working parts.
- ? PLDs do not require long lead times for prototypes or production parts - the PLDs are already on a distributor's shelf and ready for shipment.
- ? PLDs do not require customers to pay for large NRE costs and purchase expensive mask sets - PLD suppliers incur those costs when they design their programmable devices and are able to amortize those costs over the multi-year lifespan of a given line of PLDs.
- ? PLDs allow customers to order just the number of parts they need, when they need them, allowing them to control inventory. Customers who use fixed logic devices often end up with excess inventory which must be scrapped, or if demand for their product surges, they may be caught short of parts and face production delays.
- ? PLDs can be reprogrammed even after a piece of equipment is shipped to a customer. In fact, thanks to programmable logic devices, a number of equipment manufacturers now tout the ability to add new features or upgrade products that already are in the field. To do this, they simply upload a new programming file to the PLD, via the Internet, creating new hardware logic in the system.

Conclusion

The value of programmable logic has always been its ability to shorten development cycles for electronic equipment manufacturers and help them get their product to market faster. As PLD suppliers continue to integrate more functions inside their devices, reduce costs, and increase the availability of time-saving IP cores, programmable logic is certain to expand its popularity with digital designers.

1.b Product Information

The **Universal VLSI Protoboard** is a low cost universal platform for testing and verifying designs based on the Xilinx and Altera PLDs. Using this protoboard the user can verify his PLD designs with practical concepts of HDL designing.

Universal VLSI Protoboard supports multiple vendor devices from Xilinx and Altera, who are world leader in PLD manufacturing. It supports **Spartan-2** and **XC9500** series of devices from Xilinx; and **ACEX 1K** and **MAX7000s** series of devices from Altera.

The **Universal VLSI Protoboard** has not limited himself for I/O requirement in designs; it supports packages like PLCC84, TQ144 and PQ208.

Chapter 2: Features & Specifications

2.a Features and Specifications

Technical specs *

- ? Multi-vendor device support for Xilinx and Altera PLDs.
- ? Packages supported PLCC84, TQ144 and PQ208.
- ? Up to 140 user I/Os.
- ? Four Multiplexed 7-Segment displays.
- ? Interface to RS232 with 9-pin D-type connector.
- ? User selectable configuration modes, using either SPROM / JTAG / Slave Serial.
- ? Byte-blaster cable interface for configuration of Altera FPGAs.
- ? On board 8-MHz Clock oscillator (user selectable).
- ? Configurable 24 switches as I/P or O/P.
- ? 16 digital LED indicated outputs
- ? Power on Reset and configuration reset key.
- ? Power on Reset and configuration reset key.
- ? Support for different I/O Standards.
- ? Complaint with Xilinx ISE series software.
- ? Complaint with Altera Quartus design software.

2.b Individual Module Specification *

- ? **Xilinx CPLD Module**
 - o XC9572 PC84-15C containing 72 macrocells with 1,600 usable gates.
- ? **Xilinx FPGA Module**
 - o 30,000 gate density XC2S30 PQ208-5 **FPGA from Xilinx**.
- ? **Altera CPLD Module**
 - o EPM7128SLC84-15C device containing 128 macrocells with 2,500 usable gates.
- ? **Altera FPGA Module**
 - o 50,000 gate density EP1K50 TQ144-3C **FPGA from Altera**.

***Note: Above module are optional with the product.**

Power supply

The 3.3V and +2.5V are generated onboard through regulators, other supply voltages are applied from external power supply.

Here is the list of voltages on board used.

+5V

-5V

+12V

+3.3V

+2.5V

*** Note:** The above specifications and features of product are subject to change with new versions of product.

Connectors

Header Name	Ident
FPGA Header	JH1, JH2, JH3, JH4
Power supply	JP6

Jumpers

Jumpers are provided on baseboard for selection of

1. Configuration mode pins
2. Bypassing the PROM
3. Selecting configurable Input or Output
4. Selecting the O/P LEDs.

- | | |
|------------|-----------------------|
| o JP1 | Mode selection header |
| o JP2 | PROM bypass |
| o JP3 | PROM bypass |
| o S0-S7 | SW1 |
| o S8-S15 | SW2 |
| o S16-S23 | SW3 |
| o JL0-JL7 | O/P LEDs |
| o JL8-JL15 | O/P LEDs |

Downloading cable

For Xilinx PLDs*

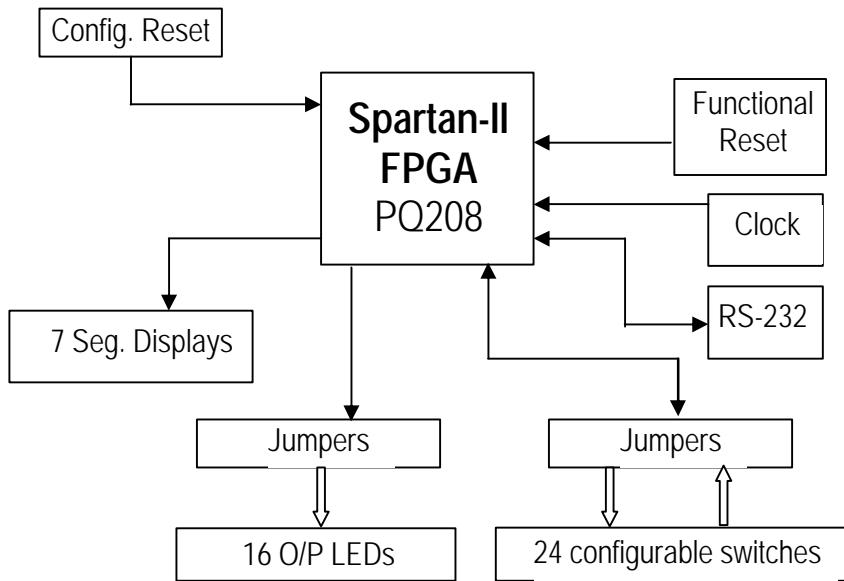
For configuration of Xilinx FPGA and CPLD from PC, a 9 pins D Type connector is provided on baseboard. The **Universal Protoboard** can be connected to PC's parallel port with cable provided having 25 pins D Type connector on other end.

For Altera PLDs*

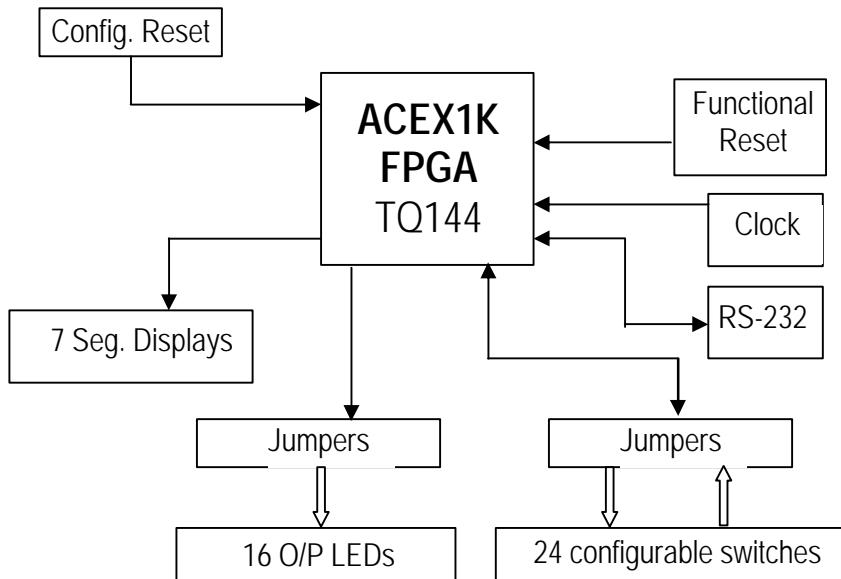
Altera PLD adaptors have onboard JTAG header. User has to connect the programming JTAG cable provided on this header and other end to PCs parallel port to configure the PLDs.

Note: *Kindly remove the cables by its headers only. Removing the cables by handling its wire may cause damage to its joints.*

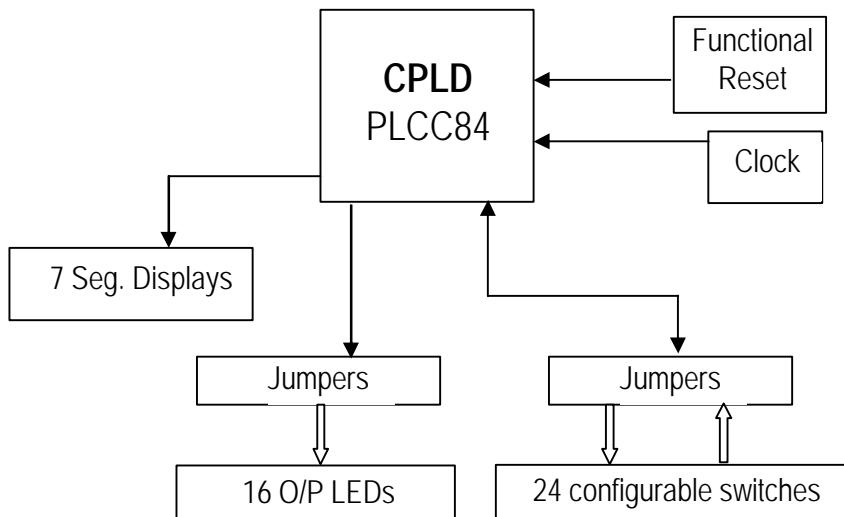
Chapter 3: Diagrams



3.a System Connection Diagram (Xilinx FPGA)

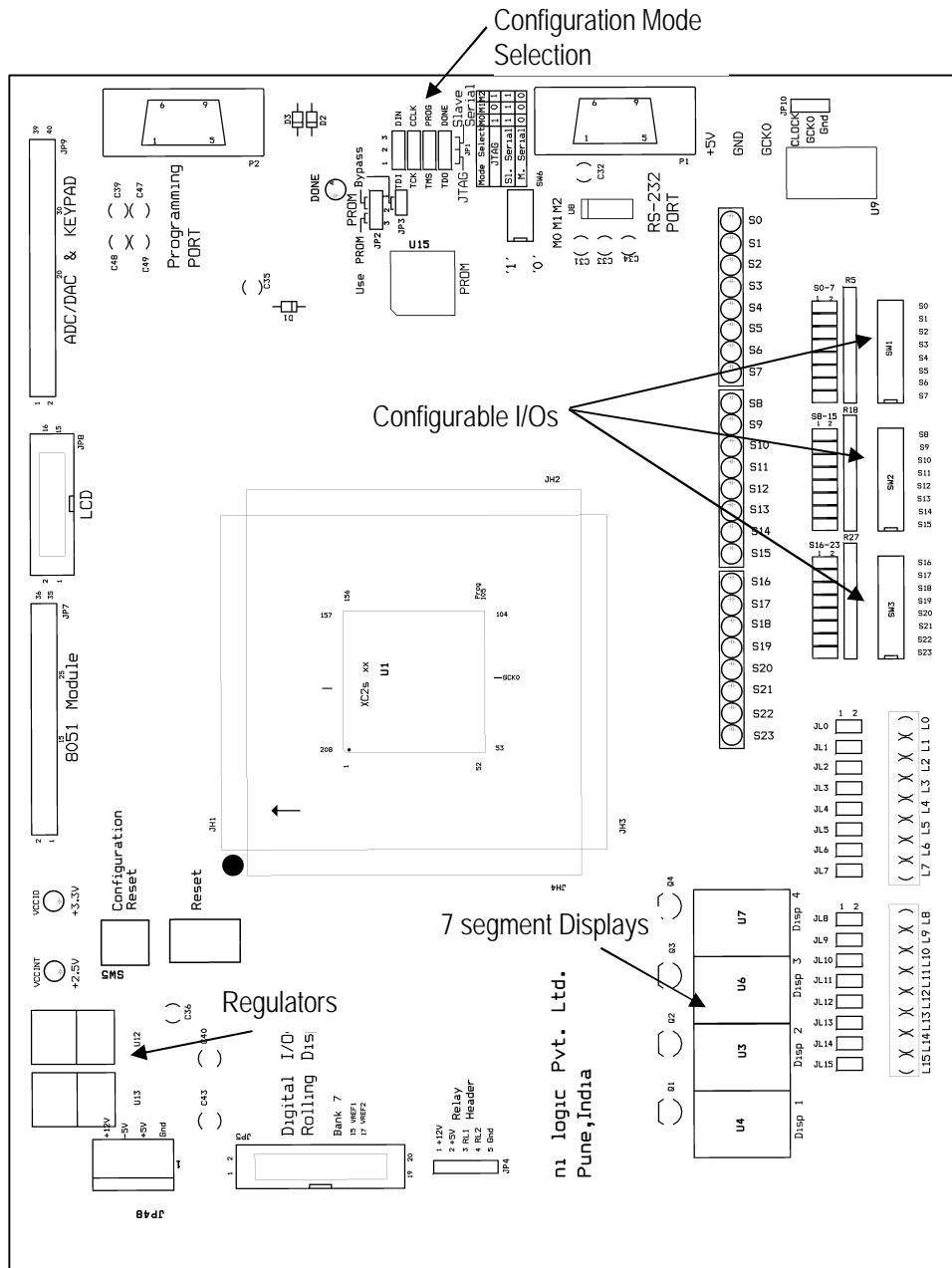


3.b System Connection Diagram (Altera FPGA)



3.c System Connection Diagram (CPLDs)

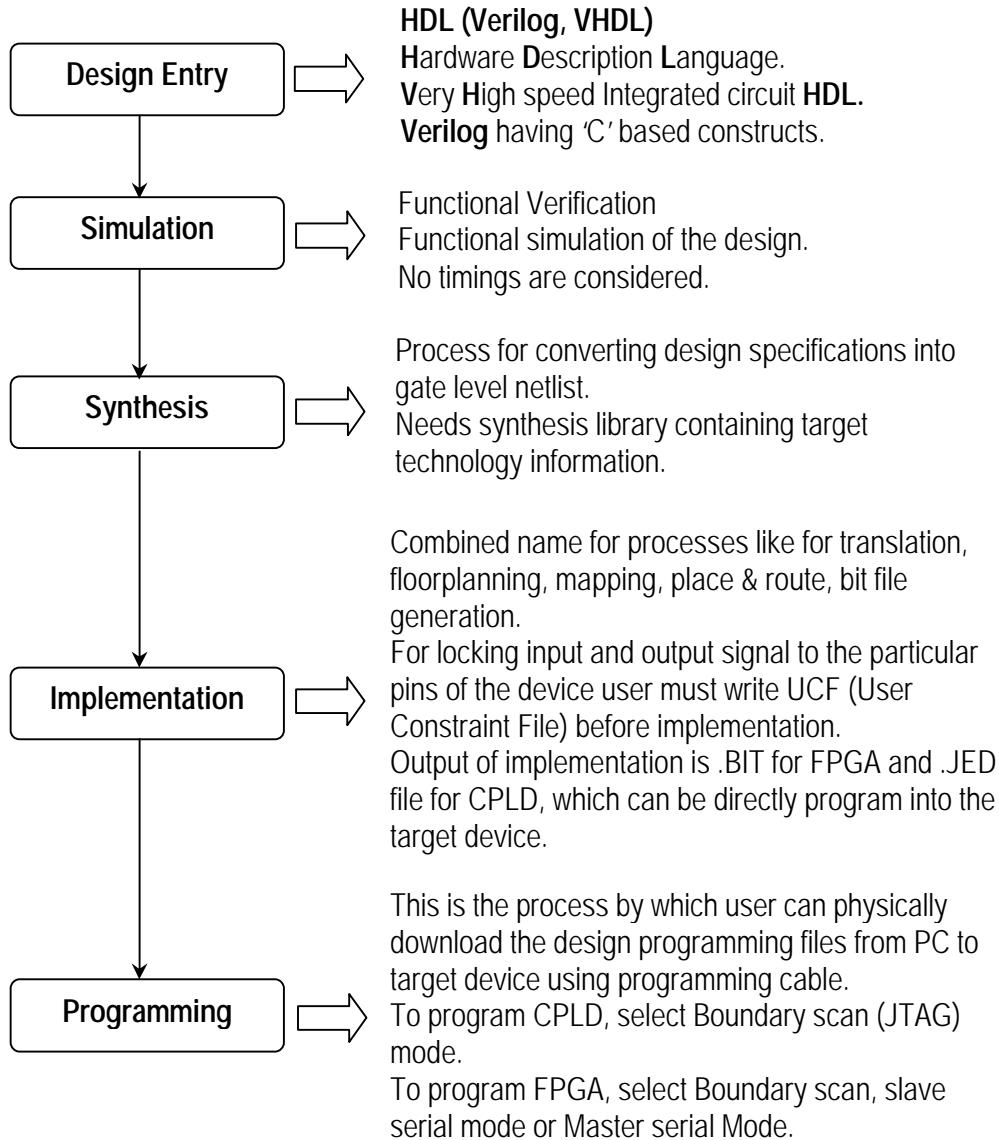
3.d Universal Protoboard component legend diagram



Chapter 4: Precautions

- ? Verify the power on LED status after applying power to the trainer.
- ? Connect the 9 pin D connector of the cable to the trainer only after confirming the above
- ? During downloading make sure that the jumper selections are proper.
- ? Select the proper configuration mode during programming, else programming can fail.
- ? Take care for adaptor position before plugging on the board.
- ? Insert the PLD adaptor by looking at the **circle** marks given on the baseboard and adaptor card.
- ? Before implementation, it is necessary to lock the pins in user constraint file (UCF) as per the design and I/Os used.
- ? For downloading the bit stream, the downloading circuit requires a stable supply; hence it is recommended to use power supply given along with the trainer board.
- ? Kindly remove the cables from holding its headers only. Removing the cables from holding its wires may cause damage to cable joints.

Chapter 5: Design Flow



Using EDA Tools

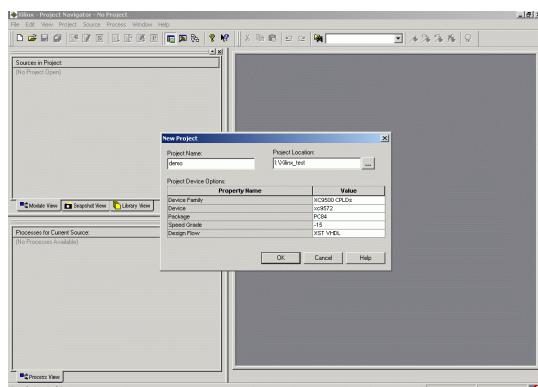
In this chapter we will see how a project can be created in Xilinx and Altera EDA tools, and how we can proceed to use Universal Protoboard to perform our experiments.

We take the example of half adder and implement on both vendor devices.

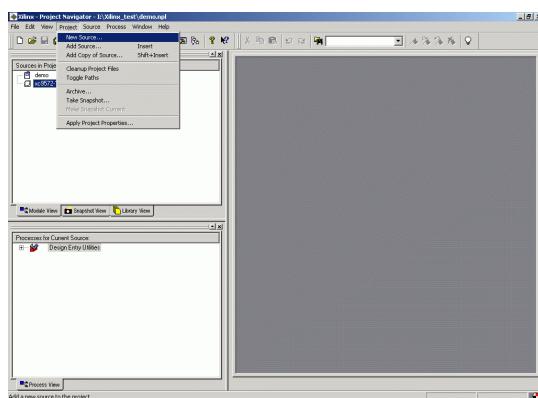
Design flow for Xilinx ISE series softwares

Step 1: Open ISE webpack software.

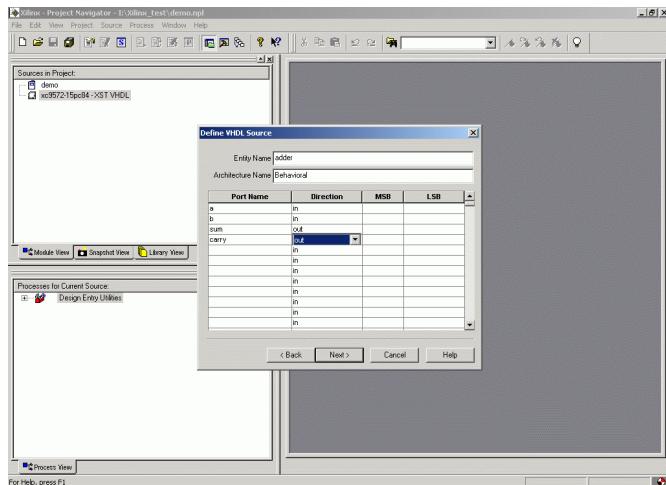
Step 2: Create new project



Step 3: Go to project menu and select new source



Step 4: Select VHDL source file, name it **adder**, click next, and enter entity I/Os as **A, B, Sum & Carry**.



Step 5: Write VHDL code for **half adder**.

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD.TEXTIO.TEXTIO_ALL;
use IEEE.STD.TEXTIO.TEXTIOITEMS_ALL;

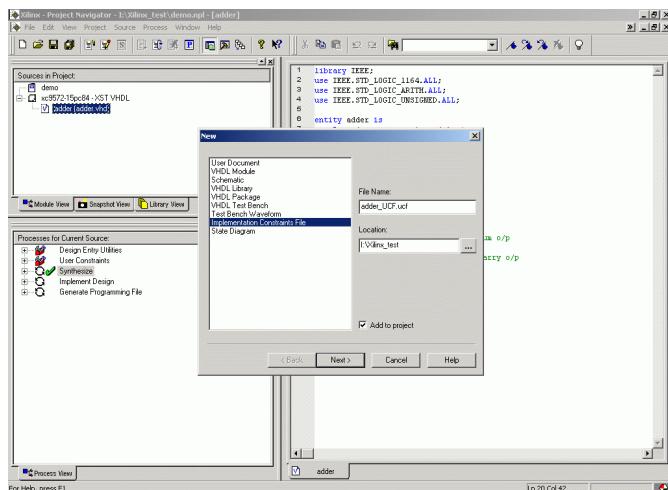
entity adder is
    Port ( a : in std_logic;
           b : in std_logic;
           sum : out std_logic;
           carry : out std_logic);
end adder;

architecture Behavioral of adder is
begin
    sum <= a xor b; -- XOR Gate Sum o/p
    carry <= a and b; -- AND Gate carry o/p
end Behavioral;

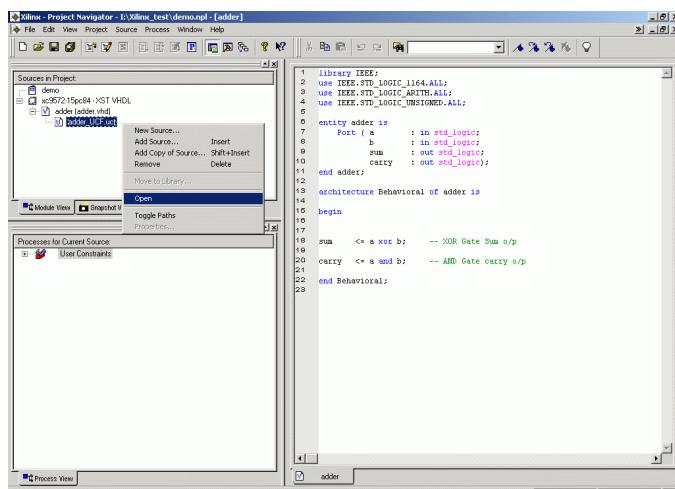
```

The screenshot shows the Xilinx Project Navigator interface with the project "adder" selected. The "Sources in Project" pane shows a file named "adder.vhd" under the "demo" folder. The "Processes for Current Source" pane lists "Design Entry Utilities", "User Constraints", "Synthesis", "Implement Design", and "Generate Programming File". The main window displays the VHDL code for the half adder.

Step 6: Create new source file for implementation constraint file. Name it **adder_UCF**, and associate with the corresponding design file.

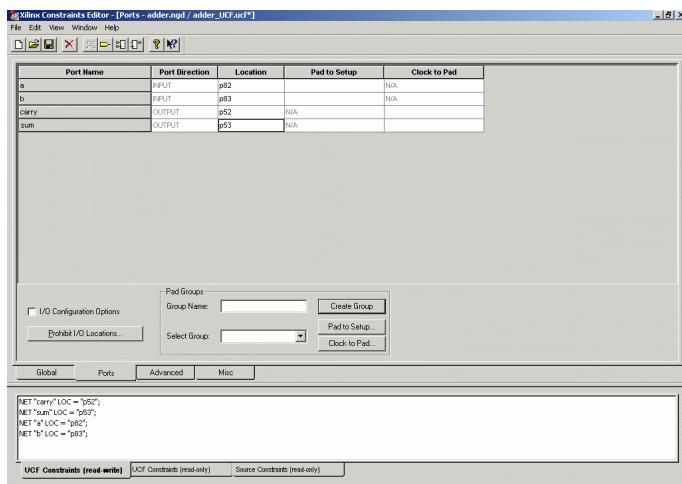


Step 7: To assign the pin location of the design, open the UCF file, to run the constraint editor where we have to lock the I/Os of design to a particular pin number.

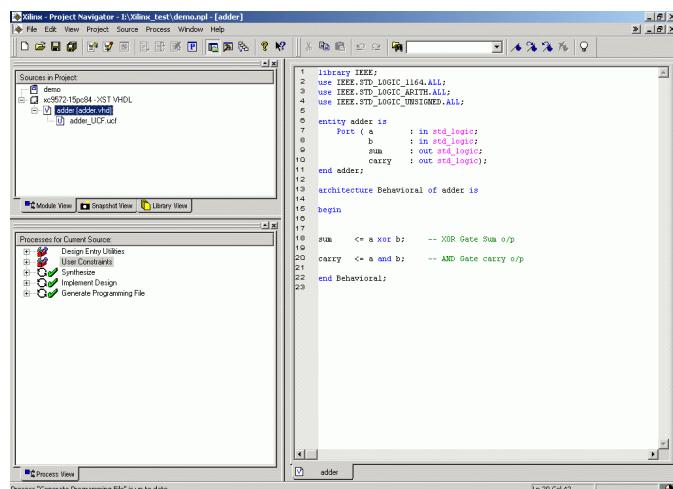


Step 8: Once the constraint editor is open, goto **ports** tab, and assign the pins by referring the Pin assignment chapter.

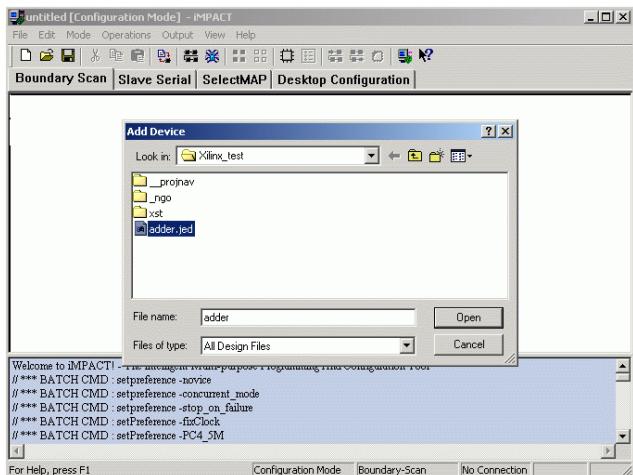
Eg: **net A loc =p82;**



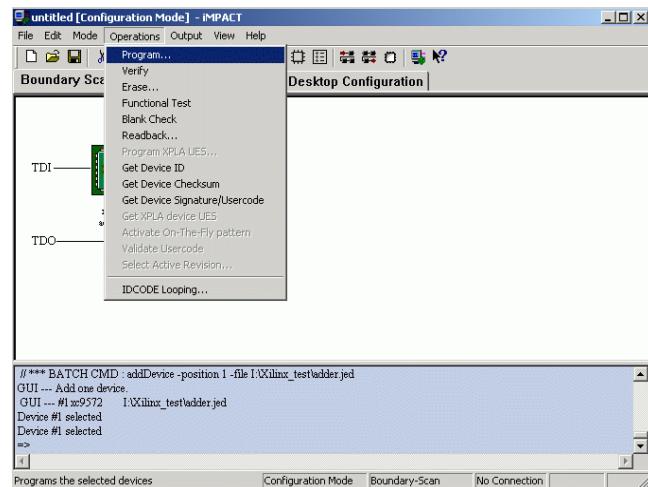
Step 9: Save the UCF file and come back to project navigator. Now selecting the **adder** design file, run the **synthesis** process, there after **Implementation** and finally run generate **programming file** option.



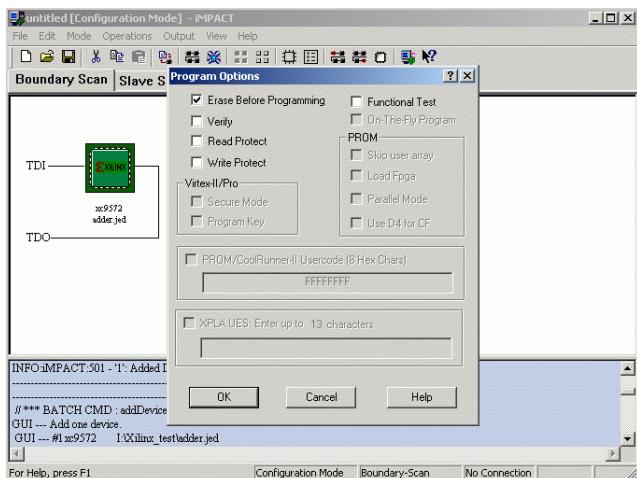
Step 10: After the all the three processes are successfully over, run the configure device (impact) option to open the impact programmer. After opening impact add Xilinx device design file, for CPLD it is *.jed file, and for FPGA it is *.bit file.



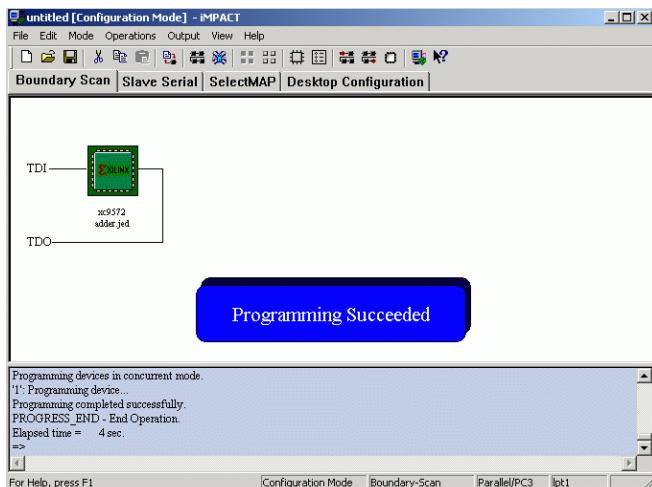
Step 11: Go to **operations** option and select program option.



Step 12: Keep the erase option enabled and click OK.



Step 13: After erasing the CPLD, the programming would start and will configure the particular device.



Now check the functionality on the board and verify it by applying different inputs.

Design Flow for Quartus-II series of software of Altera.

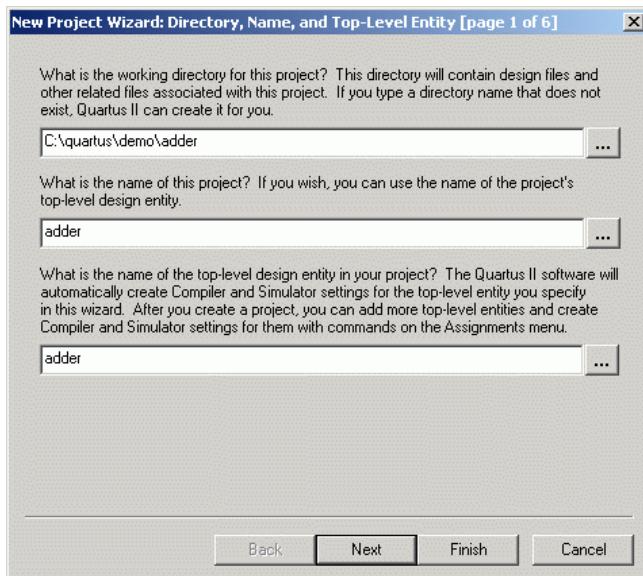
Install Quartus-II (version 3.0 & above) software on your machine, the supported platforms are windows NT/XP/2000.

We take the same example for implementing on the Altera MAX7000s CPLD.

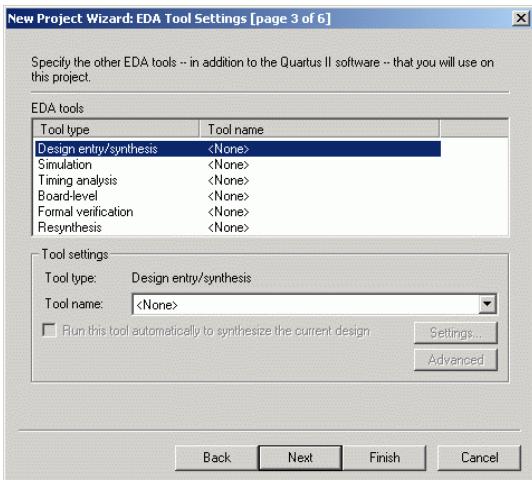
Step 1: Start Quartus-II (version 3.0 & above) software.

Step 2: For new project creation, go to **File** option and select new project wizard. In the opened window, specify project location and design and entity name. For eg. Entity name **adder**, and top design name also **adder**.

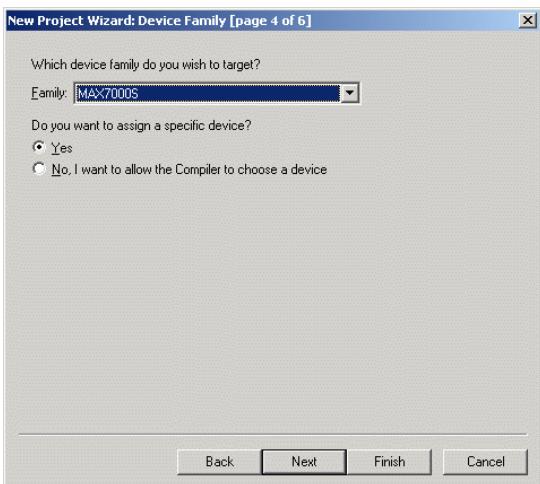
Click "next".



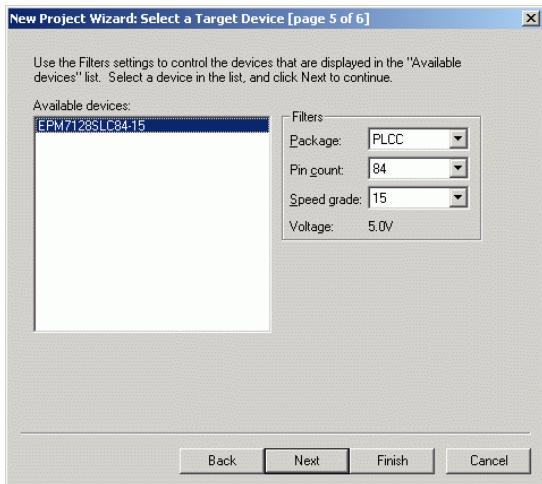
Step 3: Click “next” button till you reach EDA tool settings window, there keep all options as none, which in default will select the inbuilt design tools and softwares for the design processing. **Click “next”.**



Step 3: Select MAX700S device family in the next window. **Click “next”.**

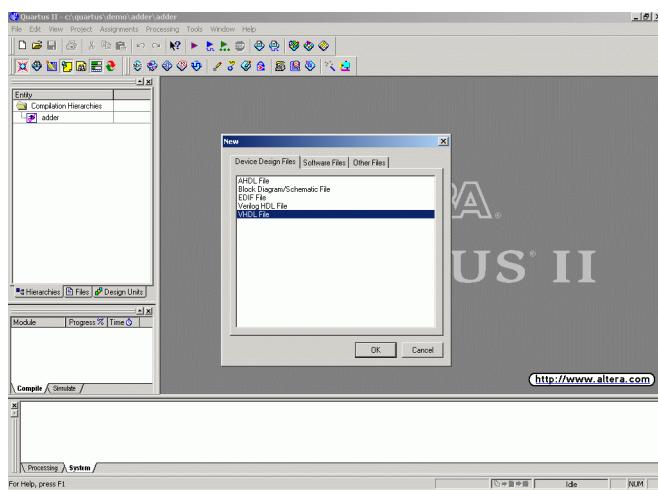


Step 4: In the next window, select the device as **EPM7128SLC84-15**. Click “next”.

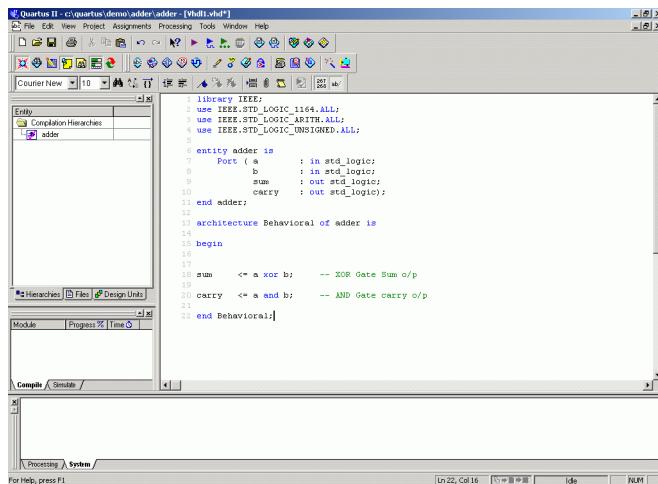


Step 5: Click “Finish”. And the new project would be created. Now we need to make and add new design file in the project. So goto “File” menu, and click “New”, and select **VHDL File**, in the “device design files” tab.

Click “OK”.



Step 6: Write the VHDL code for half adder design, and save the file as "adder.vhd"



```

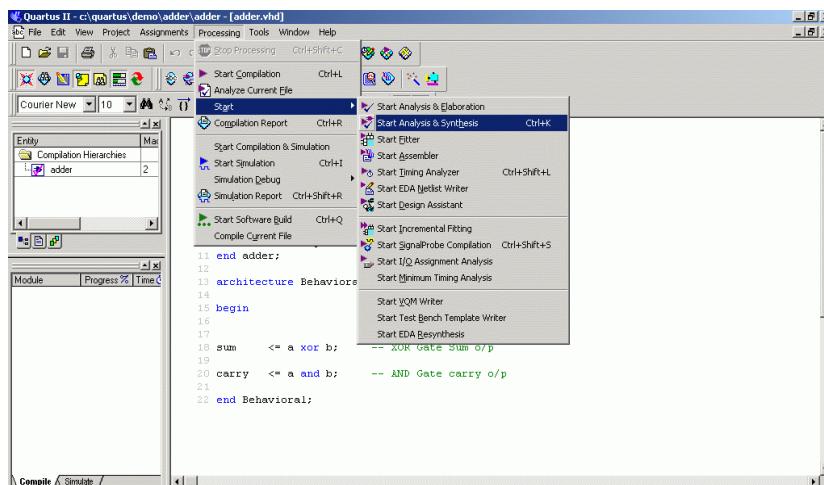
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity adder is
    Port ( a : in std_logic;
           b : in std_logic;
           sum : out std_logic;
           carry : out std_logic);
end adder;

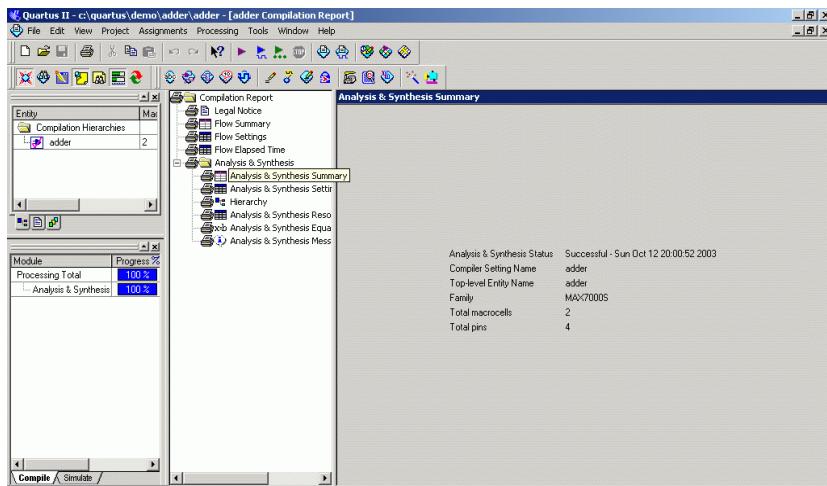
architecture Behavioral of adder is
begin
    sum <= a xor b; -- XOR Gate Sum o/p
    carry <= a and b; -- AND Gate carry o/p
end Behavioral;

```

Step 7: Now goto processing menu- then – start - , click start analysis and synthesis.

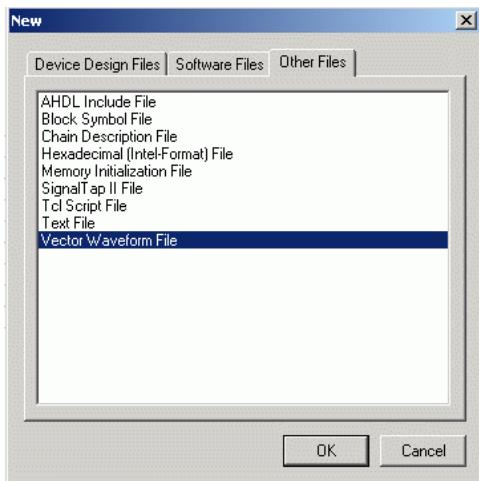


Step 8: Read the synthesis reports

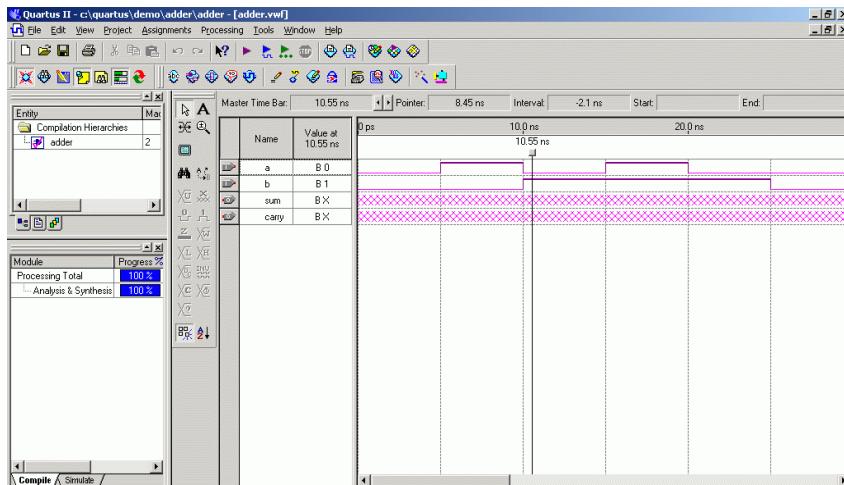


Step 9: For performing simulation, we need to create stimuli file from where we can apply input signals and watch the o/p waveforms.

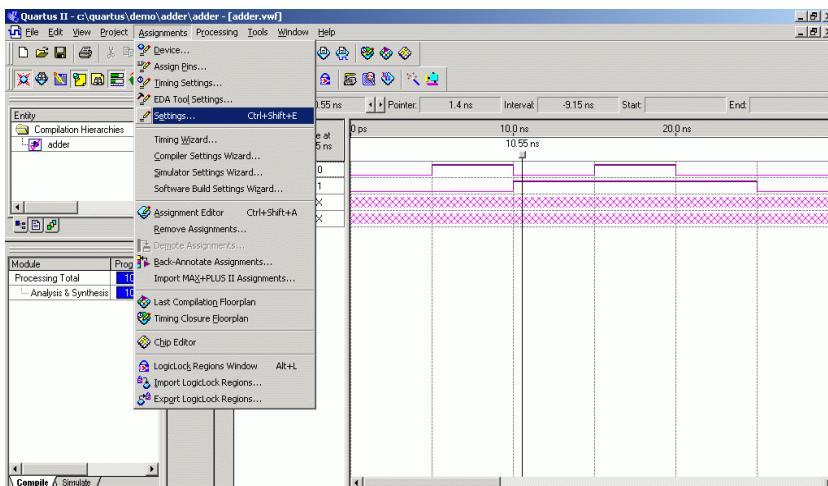
Goto **file** menu, and click **new** file, goto **other files** tab, and select “**vector waveform file**” option.



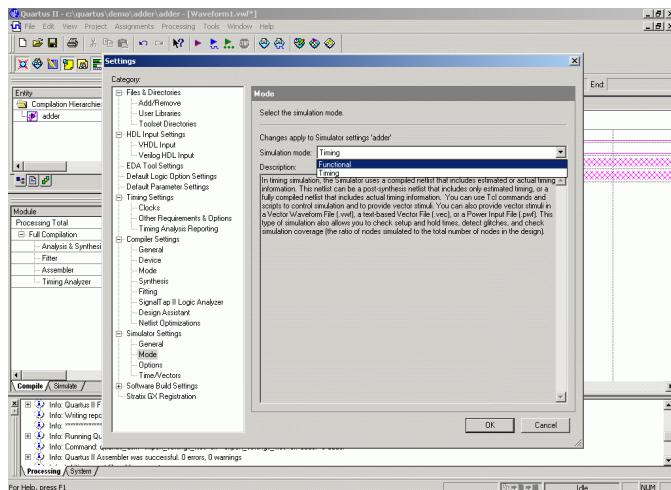
Step 10: Add the entity signals in the waveform window, and apply different sets of value to check the functionality. Save the file as the same name of entity, **adder.vwf**



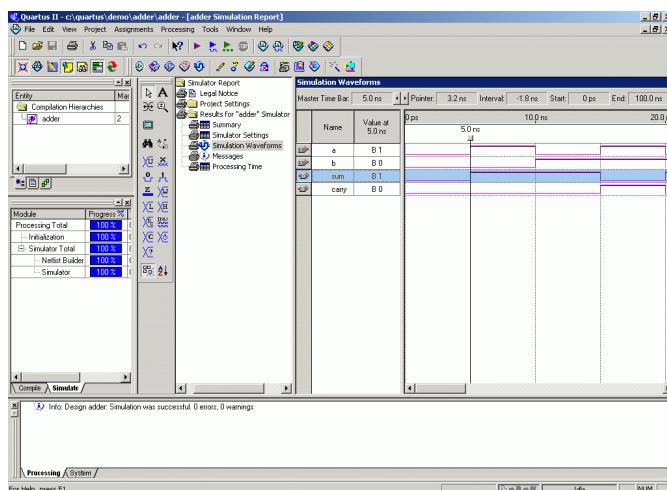
Step 11: In Altera Quartus-II software you can perform **Functional** and **Timing** simulation. For simulation mode settings, for to **assignments** menu, and click **settings**.



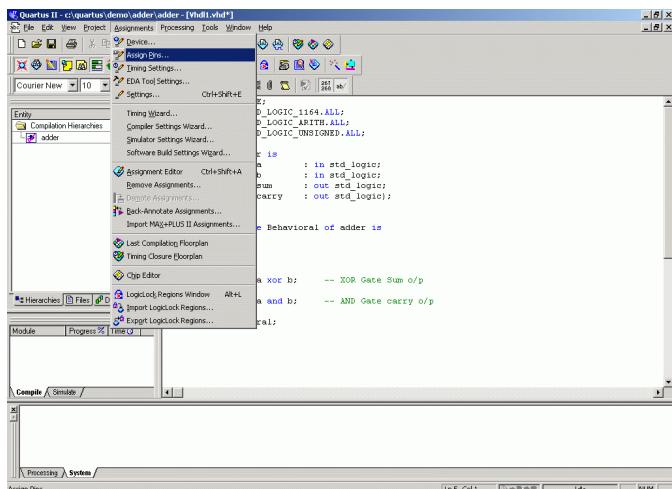
Step 12: Now goto **simulator setting**, then to **mode**, and in the right-hand side window, select the simulation mode to **Functional**.



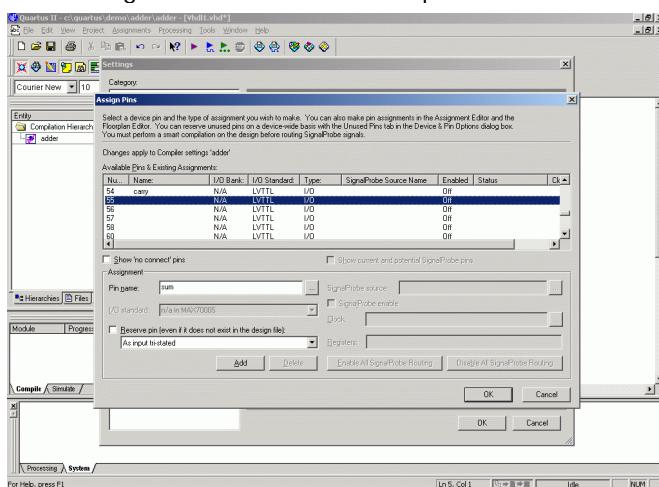
Step 13: After clicking OK, come back to main window, and goto processing window, and click start simulation, the Quartus-II will start the simulation the result would appear in couple of minutes. Observe the results, if found bugs, then change VHDL code and start simulation again.



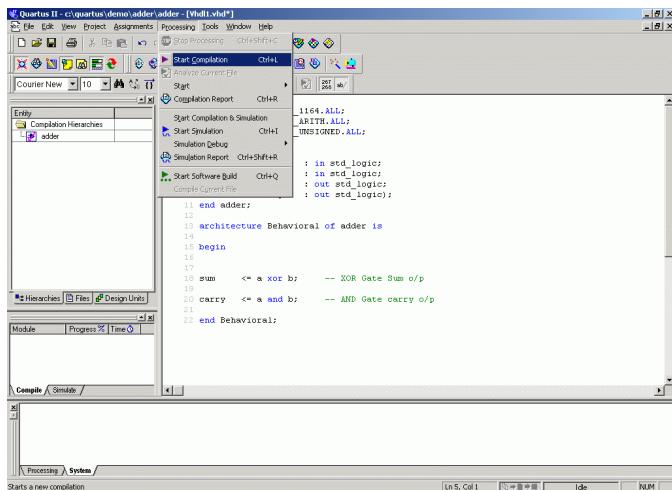
Step 14: Once the simulation results found correct, then we need to implement the design in the target device. For this we need to lock our design I/Os with the Kit I/O pin details. Goto **assignment** menu, click “**assign pin**” option.



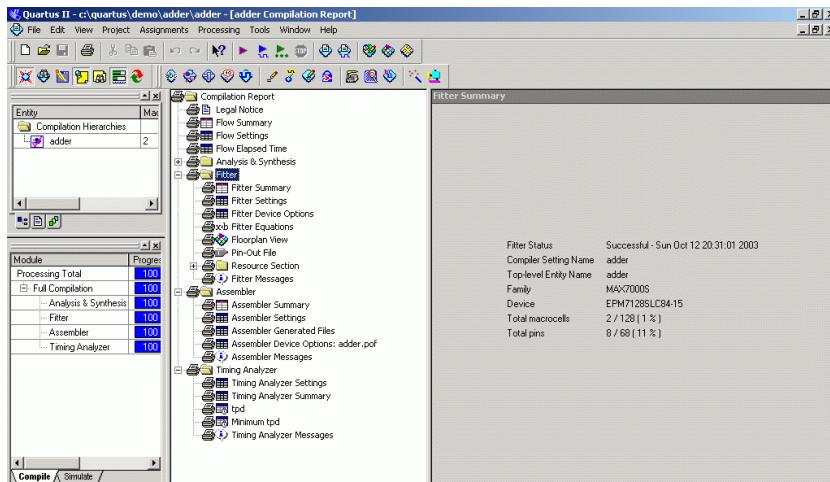
Step 15: Looking at the pin assignment chapter, lock the MAX7000s CPLD I/O with the particular pin no., for this select the I/O number on the LHS, name the design I/O in the bottom pin name option, and then click add, the particular signal will be locked to that pin number



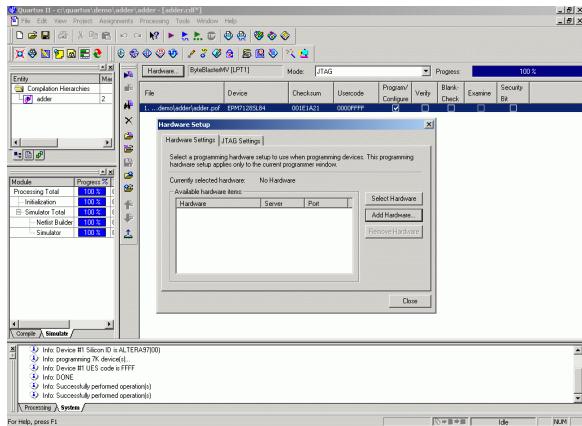
Step 16: Once the pin assignment is over, come back to main window. Now we need to implement the design on the particular device. So goto **processing** menu, and click **start compilation** process. Which will fit the design in CPLD and generate the programming file.



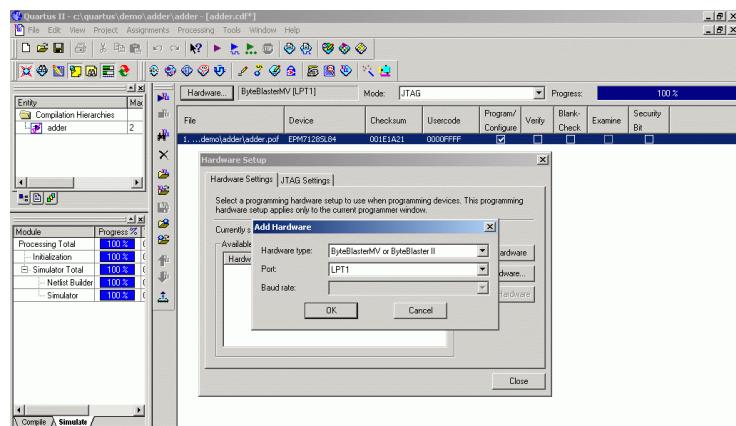
Step 17: Once the **compilation** process is over, user can check the **reports** and see the floorplan window.



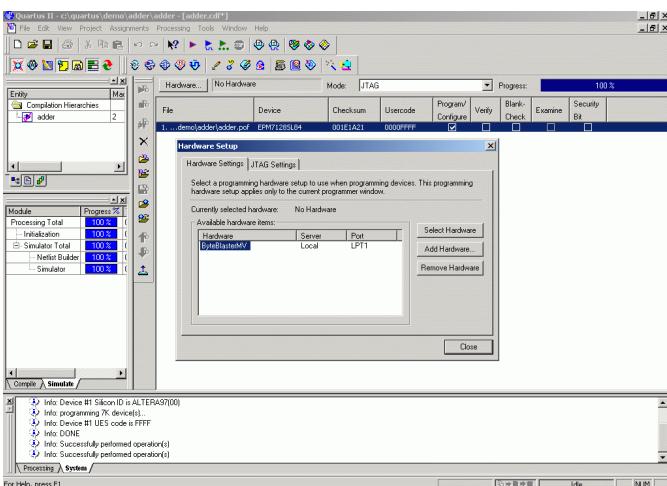
Step 18: Now we need to program CPLD, for this goto tools menu, and click programmer. Which will open the programmer; the software will automatically add the programming file (adder.pof). In the opened window select the **program/configure** option. Now we need to select the programming hardware, for which click the hardware tab on the LHS of programming window.



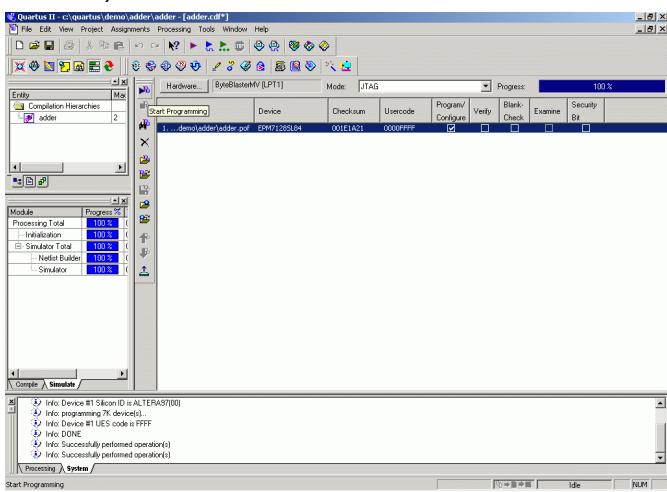
Step 19: In the opened window Click **add hardware** tab, and select the hardware type as **ByteBlasterMV or ByteBlaster II** and port as **LPT1**.



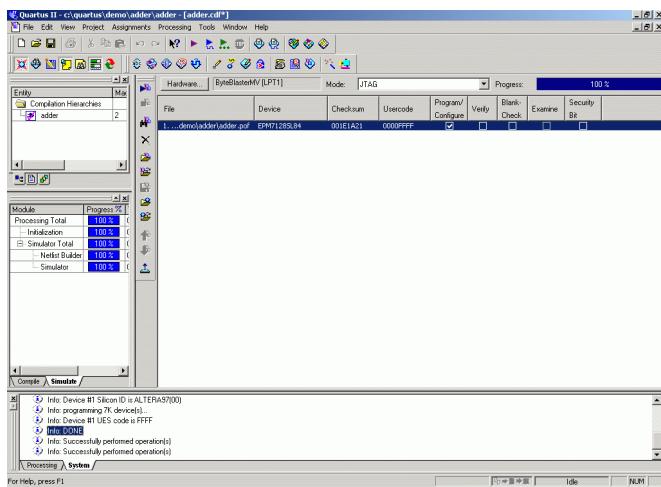
Step 20: Come back to **hardware setup** window, and click the **select hardware tab** and close the window.



Step 21: Now click the **start-programming** button (play symbol) on the top LHS of the programming window (keep the program/configure option selected).



Step 22: The programmer will start programming and in couple of seconds the device would be configured. Check the DONE indication in the bottom console window.



Step 23: Check the design functionality on the board, by applying signal from switches or other points.

Chapter 6: Configuration/Downloading

For Xilinx devices.

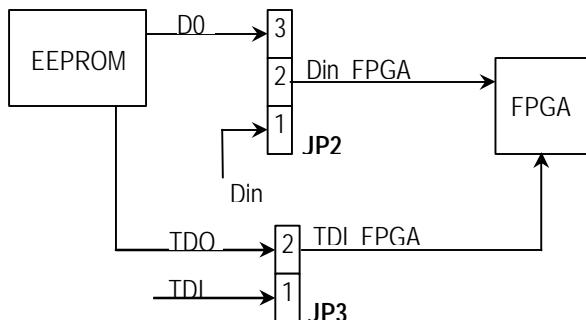
Mode Selection Header JP1	
JTAG Short 1-2	Slave Serial Short 2-3
TDI	DIN
TCK	CCLK
TMS	PROG
TDO	DONE

Mode Selection Switch			
	M0	M1	M2
JTAG	1	0	1
Slave Serial	1	1	1
Master Serial	0	0	0

The serial PROM is connected with FPGA through jumpers. While configuring the PLD from PC, the PROM has to be bypassed (JTAG & Slave serial modes).

Jumper selection for PROM

	Bypass PROM	Use PROM
JP2	Short 1-2	Short 2-3
JP3	Short 1-2	Keep open



Note: *Altera* device adaptors can be programmed only in JTAG mode.
For more programming details, refer the respective device datasheet.

Chapter 7: Pin Assignment

For Xilinx Devices

- ? Spartan-II FPGA (XC2S30PQ208)
- ? XC9500 CPLD (XC95xx PC84)

Clock and Reset			O/P LEDs				7 segment Display				
	FPGA	CPLD		FPGA	CPLD		FPGA	CPLD		FPGA	CPLD
Reset	206	74	L 15	45	35	L 7	61	45	Seg A	27	34
Clock	80	10	L 14	46	36	L 6	62	46	Seg B	29	33
			L 13	47	37	L 5	63	47	Seg C	30	32
			L 12	48	39	L 4	67	48	Seg D	31	31
			L 11	49	40	L 3	68	50	Seg E	33	26
			L 10	57	41	L 2	69	51	Seg F	34	25
			L 9	58	43	L 1	70	52	Seg G	35	24
			L 8	59	44	L 0	71	53	Seg DP	36	23
RS-232 Port									DISP En 1	37	19
	FPGA	CPLD							DISP En 2	41	17
RXD	193	NA							DISP En 3	42	15
TXD	192	NA							DISP En 4	43	14

NA: Not Available

For Xilinx Devices

- ? Spartan-II FPGA (XC2S30PQ208)
- ? XC9500 CPLD (XC95xx PC84)

Configurable Switches								
I/O	FPGA	CPLD	I/O	FPGA	CPLD	I/O	FPGA	CPLD
S23	73	54	S15	87	65	S7	99	75
S22	74	55	S14	88	66	S6	100	76
S21	75	56	S13	89	67	S5	101	77
S20	81	57	S12	90	68	S4	102	79
S19	82	58	S11	94	69	S3	108	80
S18	83	61	S10	95	70	S2	109	81
S17	84	62	S9	96	71	S1	110	82
S16	86	63	S8	98	72	S0	111	83

NA: Not Available

Note: Lock the listed I/Os of entity in **User Constraints File (UCF)** with the above pin numbers before going for implementation process.

For Altera Devices

- ? ACEX 1K FPGA (EP1k50 TQ144)
- ? MAX7000S CPLD (EPM7125SLC84)

Clock and Reset			O/P LEDs						7 segment Display		
	FPGA	CPLD		FPGA	CPLD		FPGA	CPLD		FPGA	CPLD
Reset	122	1	L 15	31	35	L 7	43	46	Seg A	17	25
Clock	55	83	L 14	32	36	L 6	44	48	Seg B	18	27
			L 13	33	37	L 5	46	49	Seg C	19	28
			L 12	36	39	L 4	47	50	Seg D	20	29
			L 11	37	40	L 3	48	51	Seg E	21	30
			L 10	38	41	L 2	49	52	Seg F	22	31
			L 9	39	44	L 1	51	54	Seg G	23	33
			L 8	41	45	L 0	59	55	Seg DP	26	34
RS-232 Port									DISP En 1	27	20
	FPGA	CPLD							DISP En 2	28	21
RXD	124	NA							DISP En 3	29	4
TXD	140	NA							DISP En 4	30	5

NA: Not Available

For Altera Devices

- ? ACEX 1K FPGA (EP1k50 TQ144)
- ? MAX7000S CPLD (EPM7125SLC84)

Configurable Switches								
I/O	FPGA	CPLD	I/O	FPGA	CPLD	I/O	FPGA	CPLD
S23	60	56	S15	70	67	S7	83	77
S22	62	57	S14	72	68	S6	86	79
S21	63	58	S13	73	69	S5	87	80
S20	64	60	S12	78	70	S4	88	81
S19	65	61	S11	79	73	S3	89	18
S18	67	63	S10	80	74	S2	90	17
S17	68	64	S9	81	75	S1	91	16
S16	69	65	S8	82	76	S0	92	15

NA: Not Available

Note: Lock the listed I/Os of entity in **Quartus-II** assignment organizer before downloading the configuration file in the respective PLD.

Chapter 8: Header and Jumper settings

Clock

Jumper Setting	Clock
1-2	8 MHz
2-3	GND

Clock

1
2
Gnd

Note: The GCK0 pin of FPGA is used for applying clock.

Mode Selection Header

JP1	
JTAG Short 1-2	Slave Serial Short 2-3
TDI	DIN
TCK	CCLK
TMS	PROG
TDO	DONE

Mode Selection Switch

	M0	M1	M2
JTAG	1	0	1
Slave Serial	1	1	1
Master Serial	0	0	0

Jumper selection for PROM

	Bypass PROM	Use PROM
JP2	Short 1-2	Short 2-3
JP3	Short 1-2	Keep open

Configurable I/Os

I/O number	Input	Output
S23-S0	Short 1-2	Keep open

Output LEDs

JL0-JL7
JL8-JL15

Header Name	Ident
FPGA Header	JH1, JH2, JH3, JH4
Power supply	JP6

Chapter 9: Sample Codes

User can use sample codes to work on Universal trainer board, provided along with the kit.

The list of sample codes provided is mentioned below, for more information and getting new examples and code, user can visit our website www.ni2designs.com

? Digital logic code

- o Combination circuits
 - ☒ Basic logic gates
 - ☒ Binary to gray converter
 - ☒ 7 segment decoder
 - ☒ 3:8 decoder
 - ☒ Demultiplexer
 - ☒ Multiplexer
 - ☒ Parity generator
 - ☒ Full adder – behavioral model
 - ☒ Full adder- structural model
 - ☒ Half adder
 - ☒ 4 bit ALU
 - ☒ Model of IC 74xx245
 - ☒ Model of IC 74181 (4 bit ALU)
- o Sequential Logic
 - ☒ 4 bit binary counter
 - ☒ 4 bit universal binary counter
 - ☒ D F/F with asynchronous reset
 - ☒ D F/F with synchronous reset
 - ☒ SRAM model (16 bytes)
 - ☒ 4 bit shift register with enable, load and parallel o/p's

Chapter 10: Glossary of Terms

ASIC (Application Specific Integrated Circuit)

A custom integrated circuit designed specifically for one end product or a closely related family of end products.

Concurrency

The ability of an electronic circuit to do several (or at least two) different things at the same time. Contrast with computer programs, which usually execute only one instruction at a time unless the program is running on a processor with multiple, concurrent execution units.

CPLD (Complex Programmable Logic Device)

A programmable IC which is more complex than the original Programmable Logic Devices such as AMD's (originally MMI's) PALs but somewhat less complex than Field Programmable Logic Arrays.

EDIF (Electronic Design Interchange Format)

A standard representation format for describing electronic circuits, used to allow the interchange of circuit design information between EDA tools.

FPGA (Field Programmable Gate Array)

An integrated circuit containing a large number of logic cells or gates that can be programmably configured after the IC has been manufactured. Some FPGAs use fuses for this programming and others store the configuration in an on chip EEPROM or RAM memory. Fuse programmed parts cannot be reprogrammed so they can only be configured once. EEPROM based FPGAs can be erased and reprogrammed so they can be configured many times. RAM based FPGAs can be reconfigured quickly, even while the circuit is in operation.

HDL (Hardware Description Language)

A synthetic computer based language used for the formal description of electronic circuits. An HDL can describe a circuit's operation, its design, and a set of tests to verify circuit operation through simulation. The two most popular digital HDLs are VHDL and Verilog. An analog HDL called AHDL is under development by many vendors. HDLs make it easier to develop very large designs through formal software engineering methods that define ways

to divide a large team project into smaller pieces that can be implemented by individual team members.

Moore's Law

An empirical law developed and later revised by Intel's Gordon Moore which predicts that the IC industry is capable of doubling the number of transistors on a silicon chip every 18 months (originally every year) resulting in declining IC prices and increasing performance. Most design cycles in the electronics industry including embedded system development firmly rely on Moore's law.

Net List (or Netlist)

A computer file (sometimes a printed listing) containing a list of the signals in an electronic design and all of the circuit elements (transistors, resistors, capacitors, ICs, etc.) connected to that signal in the design.

PLCC (Plastic Leaded Chip Carrier)

A low cost IC package (usually square). PLCCs have interconnection leads on either two (usually only for memory chips) or all four sides (for logic and ASIC chips).

PLD (Programmable Logic Device)

The generic term for all programmable logic ICs including PLAs (programmable logic arrays), PALs, CPLDs (complex PLDs), and FPGAs (field programmable gate arrays).

PROM (Programmable Read Only Memory)

An integrated circuit that stores programs and data in many embedded systems. PROM stores retains information even when the power is off but it can only be programmed or initialized once.

RTL (Register Transfer Level or Register Transfer Logic)

A register level description of a digital electronic circuit. Registers store intermediate information between clock cycles in a digital circuit, so an RTL description describes what intermediate information is stored, where it is stored within the design, and how that information moves through the design as it operates.

Simulation

Modeling of an electronic circuit (or any other physical system) using computer based algorithms and programming. Simulations can model designs at many levels of abstraction (system, gate, transistor, etc.). Simulation allows

engineers to test designs without actually building them and thus can help speed the development of complex electronic systems. However, the simulations are only as good as the mathematical models used to describe the systems; inaccurate models lead to inaccurate simulations. Therefore, accurate component models are essential for accurate simulations.

Synthesis (also Logic Synthesis)

A computer process that transforms a circuit description from one level of abstraction to a lower level, usually towards some physical implementation. Synthesis is to hardware design what compilation is to software development. In fact, logic synthesis was originally called hardware compilation.

User Constraints File (UCF)

A user created ASCII file for storing timing constraints and location constraints for a design implementation.

Verilog

A hardware description language developed by Gateway Design Automation (now part of Cadence) in the 1980s which became very popular with ASIC and IC designers.

VHDL (VHSIC Hardware Description Language)

A hardware description language developed in the 1980s by IBM, Texas Instruments, and Intermetrics under US government contract for the Department of Defense's VHSIC (Very High Speed Integrated Circuit) program. VHDL enjoys a growing popularity with ASIC designers as VHDL development tools mature.



www.ni2designs.com

ni logic pvt. ltd.

25/B-5, Bandal Complex
Bhusari Colony, Paud Road, Kothrud,
Pune – 411 038
Maharashtra.

Tele-fax: +91-20-2528 6948

Email: info@ni2designs.com

www.ni2designs.com