



Embedded Systems Design Flow using Altera's FPGA Development Board (DE2-115 T-Pad)

SPRING 2012

Ankita Goel Hamid Mahmoodi

Table of Contents

Chapter 1: Introduction to the DE2-115 Development and Education Board	3
1.1 Overview of DE2-115	
1.2 Block Diagram of the DE2-115 Board	5
1.3 Getting Started	7
1.4 Control Panel Demonstration	15
Chapter 2: Hardware Design Flow Using Verilog in Quartus II	16
2.1 Introduction to Quartus II System Development Software	
2.2 Design Flow (Hardware Only)	
2.3 Binary Adder Example	
Step by Step Binary Adder Tutorial	20
2.4 Introduction to System Builder	
Chapter 3: Hardware and Software Co-design Flow	
3.1 Introduction to Nios II Soft-Core Processor	
3.2 Co-design Flow	42
3.3 Overview of System Integration Software SOPC Builder and Q Sys	43
3.4 Introduction to Nios II SBT for Eclipse	44
Binary Adder Tutorial Using Nios II	
Chapter 4 : Video Generation for Text Display on T-Pad	51
Introduction	51
Hardware	51
Software	52
Step by Step ALU on T-Pad Tutorial	53
Hardware Setup	53
Step 1 : System Setup by using System Builder	53
Step 2: Quartus II – Hardware Setup	54
Step 3: SOPC Builder Hardware Setup	58
Software Setup	63
Basic Software Algorithm	64
Downloading the design to the board:	66

Link to the Video Demonstration:	67
Chapter 5 – Integrating Touch Interface of T-Pad	68
Introduction	68
Step by Step ALU on T-Pad with Touch Interface Tutorial	70
Software Setup	80
SOFTWARE Algorithm	81
Downloading the design to the board	83
Link of Video Demonstration	85
Chapter 6: Video Generation for Text and Image Display on T-Pad	86
Introduction	86
Step by Step ALU with image in background Tutorial	89
Hardware Setup	89
Software Setup	
Downloading the design to the Board	104

<u>Chapter 1: Introduction to the DE2-115</u> <u>Development and Education Board</u>

1.1 Overview of DE2-115

This device (FPGA Board) is specifically designed for to create, implement, and test digital designs using programmable logic. Figure below shows the I/O ports in DE2-115. It shows the layout of the board and indicates the location and connections of various components.



Figure 2-1 The DE2-115 board (top view)

The following hardware is provided on the DE2-115 board:

→Altera Cyclone® IV 4CE115 FPGA device

→Altera Serial Configuration device – EPCS64

 \rightarrow USB Blaster (on board) for programming; both JTAG and Active Serial (AS) programming modes are supported

- \rightarrow 2MB SRAM
- →Two 64MB SDRAM
- →8MB Flash memory
- \rightarrow SD Card socket
- \rightarrow 4 Push buttons
- \rightarrow 18 Slide switches
- \rightarrow 18 Red user LEDs
- →9 Green user LEDs
- \rightarrow 50MHz oscillator for clock sources
- \rightarrow 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- →VGA DAC (8-bit high-speed triple DACs) with VGA-out connector
- →TV Decoder (NTSC/PAL/SECAM) and TV-in connector
- \rightarrow 2 Gigabit Ethernet PHY with RJ45 connectors
- →USB Host/Slave Controller with USB type A and type B connectors
- →RS-232 transceiver and 9-pin connector
- \rightarrow PS/2 mouse/keyboard connector
- →IR Receiver
- \rightarrow 2 SMA connectors for external clock input/output
- \rightarrow One 40-pin Expansion Header with diode protection
- →One High Speed Mezzanine Card (HSMC) connector
- →16x2 LCD module

1.2 Block Diagram of the DE2-115 Board

Figure gives the block diagram of the DE2-115 board. To provide maximum flexibility for the user, all connections are made through the Cyclone IV E FPGA device. Thus, the user can configure the FPGA to implement any system design.





Following is more detailed information about the blocks of the Figure below:

FPGA device:

•Cyclone IV EP4CE115F29 device

- •114,480 LEs
- •432 M9K memory blocks
- •3,888 Kbits embedded memory
- •4PLLs

FPGA configuration:

•JTAG and AS mode configuration •EPCS64 serial configuration device •On-board USB Blaster circuitry

Memory devices:

128MB (32Mx32bit) SDRAM
2MB (1Mx16) SRAM
8MB (4Mx16) Flash with 8-bit mode
32Kb EEPROM

SD Card socket:

•Provides SPI and 4-bit SD mode for SD Card access

Connectors:

- •Two Ethernet 10/100/1000 Mbps ports
- •High Speed Mezzanine Card (HSMC)
- •Configurable I/O standards (voltage levels:3.3/2.5/1.8/1.5V)
- •USB type A and B
- \rightarrow Provide host and device controllers compliant with USB 2.0
- \rightarrow Support data transfer at full-speed and low-speed
- \rightarrow PC driver available
- •40-pin expansion port
- →Configurable I/O standards (voltage levels: 3.3/2.5/1.8/1.5V)
- •VGA-out connector
- \rightarrow VGA DAC (high speed triple DACs)
- •DB9 serial connector for RS-232 port with flow control
- •PS/2 mouse/keyboard

Clock:

•Three 50MHz oscillator clock inputs •SMA connectors (external clock input/output)

Audio:

•24-bit encoder/decoder (CODEC) •Line-in, line-out, and microphone-in jacks

Display:

•16x2 LCD module

Switches and indicators:

•18 slide switches and 4 push-buttons switches

18 red and 9 green LEDsEight 7-segment displays

Other features: •Infrared remote-control receiver module •TV decoder (NTSC/PAL/SECAM) and TV-in connector

Power:Desktop DC inputSwitching and step-down regulators LM3150MH

1.3 Getting Started

After getting the overview of the kit, next step is to download the necessary software development tools and drivers for the DE2-115 that will connect to your host computer via USB.

Required Downloads:

The majority of resources listed below are found on the DE2-115 and T-Pad System CDs. These CDs can be downloaded from Terasic's website free of charge. Students should first download these files onto their personal computers. Each student will need to become a Terasic member. This is done on first download attempt.

Resources on the System CD are not available for single file download directly from Terasic website. Specific files, unavailable for download, is available from System cd.

To download Quartus II and Nios II:

https://www.altera.com/download/dnl-index.jsp

To download system CDs:

1. DE2-115 resource site:

http://www.terasic.com.tw/cgi-bin/page/archive.pl? %E2%80%A8Language=English&CategoryNo=139&No=502&PartNo=4

2. TPad resource site:

http://www.terasic.com.tw/cgi-bin/page/archive.pl? %E2%80%A8Language=English&CategoryNo=139&No=550&PartNo=4

Downloading Quartus II and Nios II

Step 1) Go to the link below:

https://www.altera.com/download/dnl-index.jsp

Step 2) Click on the icon "Download Windows Version" and run The Altera Software Installer will open

Altera Software Installer (11.0 Build	157)	
Introduction Welcome to the Altera Software I	installer	
	The Altera Software Installer guides you through the pr To continue, click Next. » For more information about Altera software, go to the » For technical support, go to the <u>www.altera.com/mys</u> » For online documentation on software installations, cl » For Altera Software Installer command-line options, c	rocess of installing Altera software. e <u>Design Software Support</u> website. s <u>upport</u> website. ick <u>here</u> . lick <u>here</u> .
> Introduction		
> License		
> Installer Setup		
> Destination Select		
> Components Select		
> Summary		
> Installation		
	Allow Altera Software Installer to report statistics to	o Altera to improve quality.
		Next > Cancel

Step 3) Click Next then Agree to the Terms and Conditions, then click Next

Altera Software Installer (11.0 Build	157) 📃 🔍 🖂
Installer Setup Select the location of the installation	on files.
	Specify the Installation Files Directory Select this option if you want the Altera Software Installer to retrieve installation files from your hard disk or local network. Installer Source Directory: Browse
> Introduction	Oownload Installation Files from the Internet Select this option if you want the Altera Software to retrieve installation files from the Internet.
> License	Specify how you are connected to the Internet: No proxy (recommended)
> Installer Setup	Manual proxy configuration
> Destination Select	Web Proxy: Port: Port:
> Components Select	Password:
> Summary	
> Installation	
	< Back Next > Cancel

Step 4) Select the Destination where the Altera folder is going to be located and the name of the folder

Click next

👯 Altera Software Installer (11.0 Build 1	157)	
Select Destination Specify the software destination dir	ectory.	
	Destination Directory:	
	c:\altera\11.0	Browse
	Available Space:	134 G
	Temporary Directory for installation files:	
	C:\Users\CACONN~1\AppData\Local\Temp	Browse
	Available Space:	134 G
~	Remove saved installation files after completion.	
> Introduction	Download only and install later.	
	Program Folder	
> License	Specify the Program Folder:	
Sinstaller Setun	Altera	
	Existing Folders:	
> Destination Select		Â
> Components Select		≡
> Summary		
> Installation		
		+
	<pre>Back Next</pre>	> Cancel

Step 5) Select everything except for the Components that say "Paid". The Paid version is a 30-day trial after that you will not be able to use it. Click next

👯 Altera Software Installer (11.0 Build	157)			
Select components Select the software components	you want to install.			
	Components Image: Component size Image: Component size	ubscription Edition (Paid) Veb Edition (Free)	Install Size	Download Size
	ModelSim-	Altera Starter Edition (Free) Altera Edition (Paid) edded Design Suite	2.7 G 2.7 G	326 M 326 M
	DSP Builder	j	184 M	40 M
	Stand-Alon	e Quartus II Programmer (Free)) 722 M	96 M
 > Introduction > License > Installer Setup > Destination Select 				
	Description			
> Components Select	Installs DSP Builder. interfaces between	DSP Builder is a digital signal proc the Quartus II software and MAT	essing (DSP) de LAB or Simulink	evelopment tool that 🔺
> Summary	and Simulink system	thm development, simulation, and n-level design tools from The Math	Works with VHI	pabilities of MATLAB DL svnthesis,
> Installation	Space Required:	Install: Down 11 G 3.3 G	load:	Select/Deselect All
	Space Available:	134 G 134 G	i	
			< Back	Next > Cancel

Step 6) Click next for the DSP Builder setup

😍 Altera Software Installer (11.0 Build	157)	_ - X
DSP Builder 11.0 Setup Specify the location of your MATLA	B installation.	
	 Skip MATLAB setup. Select from the following list of MATLAB versions found on your system. Cannot find a version of MATLAB 	
> Introduction		
> License		
> Installer Setup		
> Destination Select		
> Components Select		
> Summary		
> Installation	 Specify other MATLAB installation directory. (Please note only versions from R2009a and above are supported) 	Browse
	< Back Next >	Cancel

Step 7) A summary of what will be installed to the computer will appear



💘 Altera Software Installer (11.0 Build	± 157) 📃 🖂 🖂 🖂
Installation Please wait while the Altera softw	vare is installed.
	Installation successful.
	Components Downloaded Quartus II Web Edition (Free) Quartus II Web Edition (Free) Arria GX Family Arria II GX Family Cyclone Family E E
> Introduction > License	Cyclone II Family Cyclone II Family Cyclone IV E Family Cyclone IV GX Family Legacy Families MAX II Family
> Installer Setup	MAX V Family Stratix Family Stratix II Family
> Components Select	Image: Construction of the second
> Summary > Installation	Create Desktop Shortcut Image: Optimized Content Image: Optimized Content <tr< th=""></tr<>
	Rate your installation experience
	< Back Finish Cancel

Step 8) After the installation is complete click finish.

Using these steps, Quartus and Nios software can be downloaded and ready to be used on the board.

1.4 Control Panel Demonstration

To get familiarized with the board, *Control Panel* can be used which automatically uses Quartus II to run a demonstration on the DE2-115. A video link demonstrating the same is given below:

http://www.youtube.com/watch?v=EtDDd07yUnw

Step 1: Connect the DE2-115 to your host computer through the USB port. Turn on the power by pressing the big red push-button. Make sure that SW-19 is set to *Run*.

Step 2: open *<system cd>\DE2_115\DE2_115_tools* within this file you will find control_panel.exe. With the DE2-115 connected to your host computer, execute this Control Panel file by double-clicking its icon.

Note: If your Operating System is running on 64 bit, click on win7_64bits and then click in the DE2_115 Control Panel

Step 3: It may take a few minutes for the program load. Control Panel provides a GUI for you to play with all the peripherals on the DE2-115.

Once the Control Panel is open, follow the pattern shown in the picture below and type your name into the LCD Display:



A link describing the DE2-115 board is given below: <u>http://www.youtube.com/watch?v=720t8fNcJKM</u>

<u> Chapter 2: Hardware Design Flow Using</u> <u>Verilog in Quartus II</u>

2.1 Introduction to Quartus II System Development Software

This chapter is an introduction to the Quartus II software that will be used for analysis and synthesis of the DE2-115 Development and Education Board. Throughout this chapter hardware description languages like Verilog will be used for coding. The Altera Quartus II design software provides a complete, multiplatform design environment for system-on-a-programmable-chip (SOPC) designs. Also an example will be implemented in a tutorial using the hardware description language (Verilog) and the DE2-115. Below are some suggested readings before going into the next section.

Quartus II Development Software Reading Resources:

(In suggested chronological reading/watching order)

- 1) Introduction to Quartus II Software
- \rightarrow Version 11.0 (Latest):

http://www.altera.com/literature/manual/quartus2_introduction.pdf

• *NOTE:* The link to the newer version of the later version (11.0) provides a very

brief overview, whereas the older version (listed below) gives more in depth information.

\rightarrow Version 10.0:

http://www.altera.com/literature/manual/archives/intro_to_quartus2.pdf

• *focus:* Emphasis is placed on the following sections, although a greater

knowledge base is achieved by reviewing the entire document:

a) Design Flow- Introduction (Page No. 11), Graphical User Interface Design Flow (Page No. 12)

b) Design Entry (Page No. 29) Introduction, Creating a Project(Page No. 30),
Creating a Design(Page No. 31), *later this document can be used for a specific method of design entry (like Verilog, Block Diagram, VHDL, etc.)*c) Programming & Configuration (Page No. 93) Introduction, Creating and Using Programming Files

2) Using Verilog for Quartus II Design:

<system cd>\DE2_115_tutorials\tut_quartus_intro_verilog.pdf

• *focus*: This tutorial guides through the simulation process so that the project can be implemented without needing access to the DE2-115.(familiar with quartus and Verilog) (PG No 1-21)

3) Quartus II Handbook: <u>http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf</u>

• *NOTE:* This resource is in depth and is only necessary to briefly overview the material in order to know where information can be found on an *as needed* basis.

2.2 Design Flow (Hardware Only)



2.3 Binary Adder Example

Now that you are getting familiar with Quartus II and the DE2-11 a tutorial discussing the basic steps for using Quartus II is discussed below.

In this example, the components from the DE2-115 Board that will be used are:

 \rightarrow 7 Segment Hex Display,

 \rightarrow Switches,

 \rightarrow 8 Red LEDs, and the

 \rightarrow LCD Display

As shown in the picture above the switches and LED's are synchronized and represent a 4 bit binary number. The values of these binary numbers are displayed on the 7 segment display and LCD. Moreover the addition of these two binary numbers is also displayed on the seven segment display and LCD.

*To learn more in detail about the 7 Segment Hex Display also there is a short video about 7 segment display () and LCD refer to the last 5 pages of this example



OThe Binary Adder tutorial teaches how to

- Connect the conputer with the DE2-115.
- Create a new project using Quartus II.
- Create a Verilog file.
- Put I/O pin locations in the assignment editor.
- Synthesize your design.
- Use system builder.
- The youtube video for the complete procedure can be accessed from the link given below: http://www.youtube.com/watch?v=PB9wk5Wl_Ec
- 2. The example can also be implemented by using the written instructions given below:

Step by Step Binary Adder Tutorial

Step 1: Install the USB driver for the FPGA development board. This step will only be done for the first time the FPGA board is used.

- a) On the FPGA board, connect the power plug to an outlet. Connect the USB cable from your computer to the FPGA board in port J9 (closest to the power outlet).
- b) Open the start Menu and Search Windows for "Device Manager"-> Scroll down to "Other Devices"-> A new window called "USB Blaster Properties" will open.

S Device Manager	
File Action View Help	٦
A MININT-K86E44K	
▶ 🙀 Batteries	
A - 🚯 Bluetooth Radios	
🔤 🚯 Broadcom 2070 Bluetooth	
⊳ - 1 M Computer	
Disk drives	
🔉 📲 Display adapters	
DVD/CD-ROM drives	
▷ - 😋 IDE ATA/ATAPI controllers	
▷ 満 Imaging devices	
⊳ Keyboards	
> - 🖉 Mice and other pointing devices	
> - 🔤 Modems	
Monitors	
Network adapters	
△ · In the devices	
Portable Devices	
Ports (COM & LT)	
P To rocessors	
ping Source and game controllers	
h 着 Universal Serial Bus controllers	
USB Virtualization	
× •	

 c) Under the tab "Driver" select "Update Driver" -> A new window will pop up and you'll select "Browse my computer for driver software

USB-Blaster Properties	
General Driver Details	😡 📱 Update Driver Software - USB-Blaster
USB-Blaster	How do you want to search for driver software?
Device type: Other devices Manufacturer: Unknown Location: Port_#0003.Hub_#0004 Device status The drivers for this device are not installed. (Code 28)	Search automatically for updated driver software Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.
There is no driver selected for the device information set or element. To find a driver for this device, click Update Driver. Update Driver	Browse my computer for driver software Locate and install driver software manually.
OK Cancel	Cancel

 d) In the field "Search for Drivers in this location" browse your computer to create the following path: C: -> Altera -> 11.0 -> Quartus -> Drivers -> USB Blaster then select "Browse"



- e) You may need to click "allow" to complete the process.
- Step 2: Open the Quartus II software
 - a) Select "Create New Project Wizard"



b) In the first step (1 of 5) you will need to create a directory for your project and

name your new project.

😲 New Project Wizard	×
Directory, Name, Top-Level Entity [page 1 of 5]	
What is the working directory for this project?	
C:\Users\WECRL\Desktop	
What is the name of this project?	
Adder	
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.	
Adder	
Use Existing Project Settings	
< Back Next > Finish Cancel He	IP J

- c) In step 2 of 5, you will add any previously created files to your project. Make sure to go to the lower portion of screen and select "Add User Libraries".
 - A new window opens. Go to "Global Library Name" and to the right of Global libraries click on "..."
 - ii. Go to "Computer" then go to the "C drive" (where the Altera folder is located)
 - iii. Go to on the Altera folder then go to the "quartus" folder
 - iv. Go to on the "libraries" folder
 - v. Add the "MegaFunctions" library and click "Select folder" then "OK"
- d) In step 3 of 5, "Family & Device Settings" you will adjust the family and device you want to target for compilation.
 - i. Device family is Cyclone IV E.

ii. Target device is "Specific" and select our device from "Available

Esmilue Cyclope IV E	Device ramily			Show in 'Available devices' list			
Family: Cyclone IV E			•	Package:	Any	-	
Devices: All				Pin count:	Any	-	
				Speed grade:	Any	-	
Target device				speed grade: Any			
Auto device selected b	by the Fitter			Show adva	anced devices		
Coosific douise coloste	ui 'acciuale aldelieu é' ai b			HardCopy	compatible only		
S opecane de vice selecte	an Avaiable devices in						
Name	Core Voltage	LEs	User I/Os	Memory	Bits Embedded multiplier 9-bit elements		
EP4CE115F23C8L	1.0V	114480	281	3981312	532	4	
EP4CE115F23C9L	1.0V	114480	281	3981312	532	4	
EP4CE115F23I7	1.2V	114480	281	3981312	532	4	
EP4CE115F23I8L	1.0V	114480	281	3981312	532	4	
EP4CE115F29C7	1.2V	114480	529	3981312	532	4	
EP4CE115F29C8	1.2V	114480	529	3981312	532	4	
EP4CE115F29C8L	1.0V	114480	529	3981312	532	4	
						•	
< [
Companion device							
Companion device							

Devices"→EP4CE115F29C7. Click "Next"

- e) In step 4 of 5, EDA Tool Settings do not make any adjustments. Click "Next"
- f) In step 5 of 5, Summary, click "Finish to create your new project.
- Step 3: You will need to create a new Verilog file for your project.
 - a) Under "File" select "New"

File	Edit	View	Project	Assignments	Proces
	New			Ctrl+N	
2	Open.			Ctrl+O	
	Close			Ctrl+F4	
1	New P	roject V	/izard		
1	Open i	Project.		Ctrl+J	
	Save F	Project			
	Close I	Project			
	Save			Ctrl+S	
	Save A	As			
ø	Save A	All		Ctrl+Shit	ft+S
	File Pro	operties			
	Create	e / Upda	te		•
	Export	t			
	Conve	rt Progr	amming Fil	es	
	Page S	Setup			
D.	Print P	review			
6	Print	-		Ctrl+P	
	Recen	t Files			•
	Recen	t Projec	ts		•
	Exit			Alt+F4	

b) Under "Design Files" select "Verilog HDL File"



- c) Click "OK"
- d) A new Verilog file will open. An asterisk will appear near the file name whenever unsaved changes have been made.

~ This tutorial focuses on Verilog (a hardware description language), In order to program the Altera DE2-115

Step 4: Copy the Verilog Code from the file Binary_Adder.txt file into Quartus II

Note: Binary_Adder.txt is located in the Codes folder

Step 5: You will use the DE2-115 manual to determine ports and PIN assignments. Assignments->assignment editor (Ctrl+Shift+A) set all components to their appropriate locations and voltage

Assig	nments Processing Tools	Window Help 💎							
2	Device								
_∕∕	Settings	Ctrl+Shift+E							
	TimeQuest Timing Analyzer Wiz	ard							
4	Assignment Editor	Ctrl+Shift+A							
ම	Pin Planner	Ctrl+Shift+N							
	Remove Assignments								
۰.	Back-Annotate Assignments								
	Import Assignments								
	Export Assignments								
	Assignment Groups								
	LogicLock Regions Window	Alt+L							
20	Design Partitions Window	Alt+D							

	tatu	From	То	Assignment Name	Value	Enabled	Entity	Comment	Tag
1	$\mathbf{\mathbf{V}}$		CLOCK_50	Location	PIN_Y2	Yes			
2	\checkmark		CLOCK_50	I/O Standard	3.3-V LVTTL	Yes			
3	\checkmark		IEDR[0]	Location	PIN_G19	Yes			
4	\checkmark		LEDR[0]	I/O Standard	2.5 V	Yes			
5	\checkmark		LEDR[1]	Location	PIN_F19	Yes			
6	\checkmark		LEDR[1]	I/O Standard	2.5 V	Yes			
7	\checkmark		LEDR[2]	Location	PIN_E19	Yes			
8	\checkmark		LEDR[2]	I/O Standard	2.5 V	Yes			
9	\checkmark		LEDR[3]	Location	PIN_F21	Yes			
10	\checkmark		LEDR[3]	I/O Standard	2.5 V	Yes			
11	\checkmark		KEY[0]	Location	PIN_M23	Yes			
12	\checkmark		KEY[0]	I/O Standard	3.3-V LVTTL	Yes			
13	\checkmark		KEY[1]	Location	PIN_M21	Yes			
14	×		KEY[1]	I/O Standard	3.3-V LVTTL	Yes			
15	\checkmark		KEY[2]	Location	PIN_N21	Yes			
16	\checkmark		KEY[2]	I/O Standard	3.3-V LVTTL	Yes			
17	\checkmark		KEY[3]	Location	PIN_R24	Yes			
18	\checkmark		KEY[3]	I/O Standard	3.3-V LVTTL	Yes			
19	\checkmark		🕩 SW[0]	Location	PIN_AB28	Yes			
20	\checkmark		🕩 SW[0]	I/O Standard	3.3-V LVTTL	Yes			
21	\checkmark		D SW[1]	Location	PIN_AC28	Yes			
22	\checkmark		D SW[1]	I/O Standard	3.3-V LVTTL	Yes			
23	\checkmark		🕩 SW[2]	Location	PIN_AC27	Yes			
24	\checkmark		🕩 SW[2]	I/O Standard	3.3-V LVTTL	Yes			
25	\checkmark		🕩 SW[3]	Location	PIN_AD27	Yes			
26	\checkmark		🕩 SW[3]	I/O Standard	3.3-V LVTTL	Yes			
27	\checkmark		HEX0[0]	Location	PIN_G18	Yes			
28	\checkmark		HEX0[0]	I/O Standard	2.5 V	Yes			
29	\checkmark		HEX0[1]	Location	PIN_F22	Yes			
30	\checkmark		HEX0[1]	I/O Standard	2.5 V	Yes			
31	\checkmark		HEX0[2]	Location	PIN_E17	Yes			
32	\checkmark		IEX0[2]	I/O Standard	2.5 V	Yes			
33	\checkmark		HEX0[3]	Location	PIN_L26	Yes			
34	\checkmark		HEX0[3]	I/O Standard	3.3-V LVTTL	Yes			
35	\checkmark		IEX0[4]	Location	PIN_L25	Yes			
36	\checkmark		IEX0[4]	I/O Standard	3.3-V LVTTL	Yes			
37	\checkmark		• HEX0[5]	Location	PIN_J22	Yes			
38	\checkmark		• HEX0[5]	I/O Standard	3.3-V LVTTL	Yes			
39	\checkmark		• HEX0[6]	Location	PIN_H22	Yes			

40	1		T/O Standard	2.2.1/1.1/171	Vec		
41	2		1/O Stanuaru		Vec		
42	2		Location L/O Standard	2.2.11/0771	Vec		
43	2	HEX1[0]	Location		Vec		
44	2		I/O Standard	3.3-VIVTTI	Vec		
45	2		Location		Vec		
46	1		Location L/O Standard	3.3-1/1/1771	Vec		
47	1		1/O Stanuaru		Vec		
10	1		Location L/O Standard	2.2 VIVTT	Voc		
40	1		1/O Stanuaru		Vec		
49	×.		Location L/O Standard	2.2 VIVTT	Ves		
50	×.		1/O Stanuaru	5.5-V LVIIL	Vec		
51	×.	Status: Ok	Location	PIN_025	Tes		
52	×.		I/O Standard	3.3-V LVIIL	res		
53	×.		Location	PIN_024	res	_	
54	•	■ HEX1[6]	I/O Standard	3.3-V LVI IL	res		
55	•	HEX2[0]	Location	PIN_AA25	Yes		
56	Υ,	D HEX2[0]	I/O Standard	3.3-V LVTTL	Yes		
57	۷,	D HEX2[1]	Location	PIN_AA26	Yes		
58	۷,	HEX2[1]	I/O Standard	3.3-V LVTTL	Yes		
59	۷,	HEX2[2]	Location	PIN_Y25	Yes		
60	, ₹	HEX2[2]	I/O Standard	3.3-V LVTTL	Yes		
61	. ✓	IEX2[3]	Location	PIN_W26	Yes		
62	<	•D> HEX2[3]	I/O Standard	3.3-V LVTTL	Yes		
63	✓	• HEX2[4]	Location	PIN_Y26	Yes		
64	✓	HEX2[4]	I/O Standard	3.3-V LVTTL	Yes		
65	✓	HEX2[5]	Location	PIN_W27	Yes		
66	\checkmark	HEX2[5]	I/O Standard	3.3-V LVTTL	Yes		
67	✓	HEX2[6]	Location	PIN_W28	Yes		
68	\checkmark	HEX2[6]	I/O Standard	3.3-V LVTTL	Yes		
69	\checkmark	HEX3[0]	Location	PIN_V21	Yes		
70	\checkmark	HEX3[0]	I/O Standard	3.3-V LVTTL	Yes		
71	\checkmark	HEX3[1]	Location	PIN_U21	Yes		
72	\checkmark	HEX3[1]	I/O Standard	3.3-V LVTTL	Yes		
73	\checkmark	HEX3[2]	Location	PIN_AB20	Yes		
74	<	HEX3[2]	I/O Standard	3.3-V LVTTL	Yes		
75	<	HEX3[3]	Location	PIN_AA21	Yes		
76	✓	HEX3[3]	I/O Standard	3.3-V LVTTL	Yes		
77	✓	HEX3[4]	Location	PIN_AD24	Yes		
78	\checkmark	HEX3[4]	I/O Standard	3.3-V LVTTL	Yes		

118	\checkmark	HEX6[3]	I/O Standard	3.3-V LVTTL	Yes		
119	~	HEX6[4]	Location	PIN_AB15	Yes		
120	×	HEX6[4]	I/O Standard	3.3-V LVTTL	Yes		
121	~	HEX6[5]	Location	PIN_AA15	Yes		
122	\checkmark	HEX6[5]	I/O Standard	3.3-V LVTTL	Yes		
123	\checkmark	HEX6[6]	Location	PIN_AC17	Yes		
124	\checkmark	HEX6[6]	I/O Standard	3.3-V LVTTL	Yes		
125	\checkmark	HEX7[0]	Location	PIN_AD17	Yes		
126	\checkmark	HEX7[0]	I/O Standard	3.3-V LVTTL	Yes		
127	\checkmark	HEX7[1]	Location	PIN_AE17	Yes		
128	\checkmark	HEX7[1]	I/O Standard	3.3-V LVTTL	Yes		
129	\checkmark	HEX7[2]	Location	PIN_AG17	Yes		
130	\checkmark	HEX7[2]	I/O Standard	3.3-V LVTTL	Yes		
131	\checkmark	HEX7[3]	Location	PIN_AH17	Yes		
132	\checkmark	HEX7[3]	I/O Standard	3.3-V LVTTL	Yes		
133	\checkmark	HEX7[4]	Location	PIN_AF17	Yes		
134	\checkmark	HEX7[4]	I/O Standard	3.3-V LVTTL	Yes		
135	\checkmark	HEX7[5]	Location	PIN_AG18			
136	\checkmark	IEX7[5]	I/O Standard	3.3-V LVTTL	Status: Ok		
137	\checkmark	• HEX7[6]	Location	PIN_AA14	Yes		
138	\checkmark	IEX7[6]	I/O Standard	3.3-V LVTTL	Yes		
139	\checkmark	 IEDR2[0]	Location	PIN_F15	Yes	 	
140	\checkmark	@ LEDR2[0]	I/O Standard	2.5 V	Yes	 	
141	\checkmark	IEDR2[1]	Location	PIN_G15	Yes		
142	\checkmark	IEDR2[1]	I/O Standard	2.5 V	Yes		
143	✓	IEDR2[2]	Location	PIN_G16	Yes		
144	\checkmark	IEDR2[2]	I/O Standard	2.5 V	Yes	 	
145	\checkmark	IEDR2[3]	Location	PIN_H15	Yes		
146	\checkmark	IEDR2[3]	I/O Standard	2.5 V	Yes		
147	\checkmark	D SW2[0]	Location	PIN_AA23	Yes		
148	\checkmark	D SW2[0]	I/O Standard	3.3-V LVTTL	Yes		
149	✓.	➡ SW2[1]	Location	PIN_AA22	Yes	 	
150	✓.	SW2[1]	I/O Standard	3.3-V LVTTL	Yes	 	
151	✓.	■ SW2[2]	Location	PIN_Y24	Yes		
152	\checkmark	■ SW2[2]	I/O Standard	3.3-V LVTTL	Yes		
153	\checkmark	➡ SW2[3]	Location	PIN_Y23	Yes		
154	\checkmark	D SW2[3]	I/O Standard	3.3-V LVTTL	Yes		

Step 6: For any project it is required to create pin assignment from the DE2-115 manual.

- a) Under "Assignments" select "Assignment Editor"
- b) Add each port under "Assignment Name" –each port will need two assignments:
 - i. PIN location
 - ii. I/O requirement.

Note: This process is very lengthy and in the future can be bypassed using "System Builder" (PG No. 15).

- Step 7: When the Verilog code is finished, and all assignments are done, you will be ready to compile your design and program the device.
- Step 8: At the top of the screen, select the "Play" button to begin the automatic compilation process. Watch in the lower left screen as the compilation process occurs. This may take several minutes.



Step 9: When it has compiled, double click on "Program Device".

- a) Push the large red button on the FPGA board to turn on the power.
- b) Programmer will open, and at the top left "USB Blaster" will appear. If it does not, click on "Hardware Setup". Select "USB Blaster" and click ok.
- c) When "USB Blaster" appears next to "Hardware Setup" select "Start" and watch the upper right corner as the design is implemented.
- d) When the "Progress" bar has reach 100% you may test your design on the FPGA board.



2.4 Introduction to System Builder

Alternate way to do pin assignments with the help of System Builder

System builder is a GUI that creates pin assignment by selecting the components that will be needed for a project. System builder saves time by creating the pin assignments for you and letting you choose what components you need. For Example:-

- 1) Open DE2_115_tools->DE2_115_system_builder to find DE2_115_SystemBuilder.exe
- 2) Name the project under Project Name: in this Tutorial we name or project Binary_Adder



3) Check all Components that you will be using: in this Tutorial we are using CLOCK,

LEDx27, Buttonx4, 7-Segementx8, Switchx18, and of course the LCD.



4) Click Generate

5) Create a directory for your project and then click save

Save As					— X
Save in:	Binary_Add	r.	-	+ 🗈 💣 🗊 -	
e.	Name	*		Date modified	Туре
Recent Places	🍌 db 🕤 Binary_Adı	der.qpf		6/23/2011 3:03 PM 6/23/2011 3:01 PM	File folde QPF File
Desktop					
Computer					
Network					
	•				+
	File name:	Binary_Adder.qpf		-	Save
	Save as type:	Quartus Project File (*.e	apf)	-	Cancel

- 6) To open this project open the .qpf file
- Delete the verilog code that System Builder created then copy the code from Binary_Adder_System_Builder(is located in the codes folder)
- 8) At the top of the screen, select the "Play" button to begin the automatic compilation process. Watch in the lower left screen as the compilation process occurs. This may take several minutes.



- 9) When it has compiled, double click on "Program Device".
 - a) Push the large red button on the FPGA board to turn on the power.
 - b) Programmer will open, and at the top left "USB Blaster" will appear. If it does not, click on "Hardware Setup". Select "USB Blaster" and click ok.
 - c) When "USB Blaster" appears next to "Hardware Setup" select "Start" and watch the upper right corner as the design is implemented.
 - d) When the "Progress" bar has reach 100% you may test your design on the FPGA board.



7 Segment Hex Display

□if(!reset n)begin number1 = 0; number2 = 0; end else begin number1 = SW; number2 = SW2; sum = (number1 + number2); Begin // XX X# XXXX hex4 = DISPLAYNUMBERS(number1%10); // XX #X XXXX hex5 = DISPLAYNUMBERS(number1/10); // X# XX XXXX hex6 = DISPLAYNUMBERS(number2%10); // #X XX XXXX hex7 = DISPLAYNUMBERS(number2/10); // XX XX XXX# hex0 = DISPLAYNUMBERS(sum%10); // XX XX XX#X hex1 = DISPLAYNUMBERS(sum/10); end -end -end _function [7:0] DISPLAYNUMBERS;

input [3:0] value; begin if (value == 0) DISPLAYNUMBERS = 7'b1000000;//0 else if (value == 1) DISPLAYNUMBERS = 7'b1111001;//1 else if (value == 2) DISPLAYNUMBERS = 7'b0100100;//2 else if (value == 3) DISPLAYNUMBERS = 7'b0110000;//3 else if (value == 4) DISPLAYNUMBERS = 7'b0011001;//4 else if (value == 5) DISPLAYNUMBERS = 7'b0010010;//5 else if (value == 6) DISPLAYNUMBERS = 7'b0000010;//6 else if (value == 7) DISPLAYNUMBERS = 7'b1111000;//7 else if (value == 8) DISPLAYNUMBERS = 7'b0000000;//8 else if (value == 9) DISPLAYNUMBERS = 7'b0011000;//9 end endfunction assign LEDR = number1; //Links switch orientation with respective LEDs assign LEDR2 = number2; assign HEX0 = hex0; //Links sum value to display output signal assign HEX1 = hex1; assign HEX4 = hex4; //Links 1st number value to display output signal assign HEX5 = hex5; assign HEX6 = hex6; //Links 2nd number value to display output signal assign HEX7 = hex7; assign HEX3 = 7'b1111111; assign HEX2 = 7'b1111111; endmodule



SFSU - Embedded Systems Tutorial
In this project we used four 7-segment displays to show the values of switches being turned on in binary. In a 7- segment display a high logic level will turn off the led and a low logic level to a segment will turn the led on. To represent an LED with a seven-bit value we use the values zero through six. To display a zero to a segment we set the hex value to be equal to 7b'1000000. This is because a zero will have all led on but the center led (number 6 on the figure above). The code also uses a function to simplify the task of representing a bit value to a hex value. Since the function DISPLAYNUMBERS only has one output it seemed like a function instead of a task. In the function we have only one input value that represents a 4 bit switch value, this value is passed through a series of if else statements to determine the hex value. At the end of this program we assign all appropriate values to the represented LEDs.

There is a quick example of getting the LED's, Switches, Keys, and 7 segment Hex Display to function properly in the link below that goes more in detail about the 7 segment display.

http://www.youtube.com/watch?v=SNCZGqWEtJg

<u>16 x 2 LCD</u>





8-Bit Ineterface

To display characters to an LCD there is a series of steps that need to be done before to initializing the LCD module. Since Verilog doesn't read code sequentially we created a case statement that will allow the initialization to be done in order. This is done by changing the state of the case to the next step in every case statement. The steps performed are RESET1, RESET2, RESET3, FUNCTION SET, DISPLAY OFF, DISPLAY CLEAR, RETURN HOME, CHANGE

LINE, DROP LCD, HOLD, DISPLAY ON, and MODE SET AND PRINT STRING. These reset needs to be done three time to because we need to initialize enable to high and register select and read/write to low signals. These steps are also done to communicate with the LCD to determine if it will be an 8 or 4 bit data bus, this is done by setting the data bus equal to the hex value eight(8'h38). Before we can write to the screen we need to clear the LCD display, this is done by changing the data bus equal to 8'h01 (Start of heading). Finally when we need to display the screen we set enable and read/write to high and reset to low, this is done because this allows us to write data to the LCD. In the print string case statement we added an else if (index ==line1) because without this the LCD wouldn't know when the next line begin or the first line starts.

<u>Chapter 3: Hardware and Software Co-</u> <u>design Flow</u>

3.1 Introduction to Nios II Soft-Core Processor

1) Introduction to the Altera Nios II Soft Processor:

<system cd>\DE2_115_tutorials\tut_nios2_introduction.pdf

• *focus*: All of the information in this resource is needed for creating systems and should be read carefully, as familiarity will greatly help students in avoiding time consuming mistakes.

Nios II is an embedded processor architecture designed specifically for Altera's FPGA boards. An example of a Nios II processor system could be found on <u>page 11</u> from Altera's Nios II Processor Reference Handbook. When implementing your board there is three different types of CPU's to choose from which are the NIOS II/fast, NIOS II/standard, and NIOS II/economy. The main differences between the CPU's are the balance between performance and cost.



NOTE: This figure taken from Altera's Nios II Processor Reference Handbook: <u>http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf</u> page 11

Figure 2-1. Nios II Processor Core Block Diagram



NOTE: This figure taken from Altera's Nios II Processor Reference Handbook: <u>http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf</u> page 18

2) Nios II Hardware Development:

http://www.altera.com/literature/tt/tt_nios2_hardware_tutorial.pdf

• *focus:* This resource is an excellent overview of the basic requirements to creating a system using QSys in Quartus II, instantiating the design in the project files, implementation, and then creating the necessary software.

3) Nios II Processor Reference: <u>http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf</u>

• *NOTE:* This resource has a lot of detailed information which is not necessary to complete most projects, but it is good to be familiar with document in the case of troubleshooting.

3.2 Co-design Flow





Figure 1–2 shows the Nios II system development flow between hardware and software. This flow consists of three types of development: hardware design steps, software design steps, and system design steps.

NOTE: This figure taken from Altera's Nios II Hardware Development Tutorial: <u>http://www.altera.com/literature/tt/tt_nios2_hardware_tutorial.pdf</u>

3.3 Overview of System Integration Software SOPC Builder and Q Sys



NOTE: This diagram was taken from Altera's Nios II Software Developer's Handbook, <u>http://www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf</u>

System Integration Software

This software allows the designer to marry hardware and software. In order to use the Nios II soft-core processor, a system must be designed using either SOPC builder or QSys (both are accessed from Quartus II-> Menu -> Tools). QSys is a newer version of SOPC builder and it is encouraged that students begin with QSys. This development tool primarily generates the .sopcinfo file which is used in Nios II SBT for Eclipse to create the software project to run on top of the FPGA design, utilizing the Nios II soft-core processor.

After creating a system to suit the students' project needs, "Generation" (synonymous to "Compilation") automatically creates the necessary hardware files for low-level abstraction. A main niosII module is created in this process, which is instantiated from the top-level hardware file. This process is described as *System Integration*

Although much of the reading presented here applies to SOPC Builder, the information applies also to QSys and an effort should be made to use QSys in place of SOPC Builder.

- Introduction to the Altera SOPC Builder: <system cd>\DE2_115_tutorials\tut_sopc_introduction_verilog.pdf
- 2) QSys System Design: <u>http://www.altera.com/literature/tt/tt_qsys_intro.pdf</u>

• QSys main reference page: <u>http://www.altera.com/products/software/quartus-</u> <u>ii/subscription-edition/qsys/qts-qsys.html?GSA_pos=10&WT.oss_r=1&WT.oss=qSys</u>

3) SOPC Builder User Guide: <u>http://www.altera.com/literature/ug/ug_sopc_builder.pdf</u>

3.4 Introduction to Nios II SBT for Eclipse

Eclipse allows the user to use the software that was executed by a Nios II processorbased system in an FPGA. The user can configure the FPGA on the development board with the pre-generated Nios II standard hardware system by downloading the FPGA configuration file to the board.

 Nios II Software Developer's Handbook: <u>http://www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf</u>
 NOTE: Link is placed here for reference, but is not necessary for review in this stage.

Binary Adder Tutorial Using Nios II

A link to the video describing the Binary Adder Tutorial:

http://www.youtube.com/watch?v=bKA3mNYTl2g

http://www.youtube.com/watch?v=bM4uHq9hlmQ

The major steps were:

- 1) Create hardware system in system builder
- 2) Build new system in QSys system
- 3) Instantiate the Nios II module in top level entity
- 4) Add IP variation file
- 5) Adjust .sdc
- 6) Place design on FPGA

7) Develop Software in Nios II SBT for Eclipse.

Hardware:

- Clock
- Red LEDs
- Switches
- 7 segment Hex
- LCD

NIOS II Binary Adder

Step 1: System Builder

- 1) Open DE2_115_tools->DE2_115_system_builder to find DE2_115_SystemBuilder.exe
- 2) Name the project under Project Name: Binary_Adder_Nios



3) Check all Components that you will be using: in this Tutorial we are using CLOCK, LEDx27, 7-Segementx8, Switchx18, and of course the LCD.

NITERAL terasic		System Configuration Project Name:					
ROGRAM WWW.terresic.com		Binary_Adder					
DE2-115 FPGA Bo		P CLOCK P LED x27 P Button x4 P Ps2 VGA SDRAM.128MB SRAM.248 SRAM.248 SRAM.248 Receiver IO Voltage, 33 V(Petk Name None None	RS-232 P 7-Segement x 8 Switch x 18 S Switch x 18 S DCARD VICD FLASH, 8MB F UCD FLASH, 8MB F USB F ENFRM USB EBHRM12 EEHROM, 32kb Default) v				
Load Setting	Generate	IO Voltage: 2.5 V (Prefix Name:	Default) 🔹				
Save Setting	Exit	None	•				

- 4) Click Generate
- 5) Create a directory for your project and then click save



6) To open this project open the .qpf file

Step 2: Building Qsys System

- 1) Open Qsys under tools tab
- Start by adding a Nios II Processor Core: Under "Component Library"-> Processors -> Nios II Processor -> Add
 - a. Select "Nios II/s"
 - b. Set "Hardware multiplication type" = "None"
 - c. Disable "Hardware divide"
 - d. "Finish"
 - e. Rename Nios to "cpu"
- 3) On-Chip Memory: Under "Component Library"-> Memories and Memory Controllers ->
 - On-Chip -> On-Chip Memory (RAM or ROM)-> Click "Add"
 - a. Block Type list = "Auto"
 - b. Total Memory size = "204800" to specify 2KB of memory
 - c. Do not change any other default settings.
 - d. "Finish"
 - e. Under the "System Contents" tab, right-click the on-chip memory and rename as "onchip_mem"
- 4) JTAG UART: Component Library -> Interface Protocols -> Serial -> JTAG UART -> Add
 - a. Do not change any default settings
 - b. Rename to "jtag_uart"
- 5) Interval Timer: Component Library -> Peripherals -> Microcontroller Peripherals -> Interval Timer-> Add
 - a. Under "Hardware Options" set "Presets" to "Full-Featured"
 - b. Do not change any other default settings
 - c. Rename to "sys_clk_timer"
- 6) System ID Peripheral: Component Library-> Peripherals -> Debug and Performance -> System ID Peripheral-> Add
 - a. Do not change any default settings
 - b. Rename as "sysid"
- 7) PIO's: Component Library-> Peripherals -> Microcontroller Peripherals -> PIO -> Add
 - a. Under "Basic Settings" enter the value of "4" for the box labeled "Width"
 - b. Do not change any other default settings
 - c. Finish
 - d. Rename as "pio_led"
 - e. For this example us two "pio_led"
 - f. Repeat these steps for two "pio_sw" with 4 bits of width and change to input.
 - g. Repeat these steps for pio_hex0 through 7 with widths of 7 bits.
- LCD: Component Library-> Peripherals -> Display-> Character LCD -> Add
 a. Finish
- 9) Go to the "Connections" column and connect the following ports: (Figure Below)
 - a. For all the components connect the clock input and outputs to clock_50
 - b. For all the components connect the Avalon memory mapped slave to the On-chip memory AMMS.
 - c. Open the **Nios II processor named CPU** and change the reset vector and exception vectors to onchip_memory2

- 10) Go to the "Export" column and connect the following ports:
 - a. Click on "click to export" on the external connection row to activate connection for all of the led's, switches and 7 segment display.
 - b. Click on "click to export" on the external row for the LCD
- 11) Under Generation click generate
 - a. Save as "Nios"
 - b. Once generation is complete coping code from HDL example



Step 2: Quartus HDL Connections

- 1) Add IP Variation File: Menu bar: Assignments -> Settings
 - a. Under "Category" -> "Files" -> (...) Browse -> Choose script files for type to find(*.tcl, *.sdc, *.qip)

- b. Locate and choose the file nios2/synthesis/nios.qip
- c. Add to project, click okay and close
- Copy code under structural coding in Quartus (Code located in the Codes folder under Binary_Adder_Quartus)
 - a. Notice LCD_BLON is set to 1'b1;
 - b. Notice LCD_ON is set to 1'b1;
 - c. Notice all connections in parenthesis



- 3) Compile and Run
 - a. Compile and Run

Step 3: Develop the Software for Nios II SBT for Eclipse

- 1) This step relies on the .sopcinfo file created when generating the Qsys system
- 2) Open Nios II SBT for Eclipse
 - a) Indicate workspace as your project directory, and create a new file called "Software" and click "Okay"
 - b) Set perspective to Nios II: Menu -> Window -> Open Perspective -> Other -> Nios II
 - c) Menu -> File -> New -> Nios II Application and BSP from Template



- i) Under "Target Hardware Information" select file <directory>\nios.sopcinfo
- ii) Under "Application Project" type "Binary Adder" as "Project Name"
- iii) Under "Project Template" select "helloWorld"
- iv) Click "Finish"

Nios II Application and BSP from Terror	mplate	
Nios II Software Examples Please specify a .sopcinfo file		
Target hardware information SOPC Information File name:		
Application project Project name: If Use default location Project location:		
Project template Templates Blank Project Board Diagnostics Count Binary Hello Kinecc/OS-II Hello Microf Small Memory Test Memory Test Memory Test Simple Socket Server Simple Socket Server Web Server Web Server	Template description Hello World prints: 'Hello from Nios II' to STDOUT. This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readmeth file in the project directory. The BSP for this template is based on the Altera HAL operating system. For information about how this software example relates to	A H
?	< Back Next > Finish	Cancel

3) Include C++ code (Code located in the Codes folder under Binary_Adder_Nios2)

```
🖻 *hello_world.c 🛛
           #include <stdio.h>
#include <stdlib.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
         void lcd_display(int a, int b);
            int main()
                         int value,value2;
                          int value, 
                                               3.2
                    IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX6_BASE, segments[value%10]);
IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX7_BASE, segments[value/10]);
                     IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX4_BASE, segments[value2%10]);
IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX5_BASE, segments[value2/10]);
                     IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX0_BASE, segments[(value+value2)%10]);
IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX1_BASE, segments[(value+value2)/10]);
                      lcd display(value2,value);
                           }
                               11-
                     return 0;
            void lcd_display(int a, int b){
    FILE *pLCD;
    char text[32];
    sprintf(text, " $-2.2i + $-2.2i = $-2.2i \r", a,b,a+b);
                             pLCD = fopen(LCD_NAME, "w");
                              if(pLCD){
                                    fwrite(text, 32, 1, pLCD);
fclose(pLCD);
            }else{
                        printf("Failed to Display\n");
            3
            3
```

- 4) Build project
- 5) Run as Hardware

<u> Chapter 4 : Video Generation for Text</u> <u>Display on T-Pad</u>

Introduction

In this chapter, the ALU will be displayed on T-Pad. Switches perform different operation of the ALU. With switches, different numbers can be displayed and also their ALU operations can be performed.

Hardware



The T-Pad features an 8-inch Amorphous-TFT-LCD panel. The LCD Screen module offers resolution of (800x600) to provide users the best display quality for developing applications. The LCD panel supports 18-bit parallel RGB data interface.

The hardware is implemented using Altera IP cores on SOPC builder. A phase locked loop (Alt PLL) has been used to generate the required clocking for the whole system. In this system a 100Mhz clock for the Nios-II/f have been used, another 100Mhz with -65 phase shift is used to clock the SDRAM in addition to the required 40Mhz clock for the VGA controller. The figure above shows the block diagram of the hardware that is implemented in the SOPC builder.

Video Pipeline

A Scatter Gather DMA is used to connect to the VGA Controller as shown in the figure below. A summary of how video is fed to the VGA Controller is given in the paragraph below.



The Scatter Gather DMA is used for high speed data transfer between two components. It is used to transfer and merge noncontiguous memory to continuous address space and vice versa. It works in three modes.

- 1. Memory to Memory
- 2. Memory to Data Stream
- 3. Data Stream to Memory

In this chapter, the SGDMA is used to transfer data from SDRAM to the VGA Stream. So that is option 2 from the above. A timing adapter is used to adjust the timing between the two different streams of data. In short, it is used to connect two components that require different number of cycles to receive or send data. A FIFO is a First In First Out queue. It is a dual clock FIFO that is used to match the system clock to the VGA clock to normalize the flow of pixels to the VGA sink.

A RGB converter is required to convert the RGB format from BGR0 to BGR. The VGA Controller requires 18 bit parallel RGB interface. To make the format coming from memory (24bit RGB) compatible with the VGA sink that is connected to the tPad, we insert RGB Converter. All these components contribute to generate a video pipeline which enables us to display a video on the tPad.

Software

The LCD screen is initialized and a blank screen can be seen. Switches are toggled to change the number values and their operation, the result is displayed on the LCD Screen and updated every time switch is toggled.



Step by Step ALU on T-Pad Tutorial

Hardware Setup

Step 1 : System Setup by using System Builder

Open System Builder, select Clock, LED, VGA and switches as shown in figure below.



Select HSMC Source as LTC – 8" LCD/Touch Camera as shown below.



 \rightarrow Select a project name, for this example we are using "tpad_alu_display" as our project name Click on Generate and open the folder containing these files.

 \rightarrow Open the folder where the project files are saved and open tpad_alu_display.qpf file. This file will be opened in quartus II.

Step 2: Quartus II – Hardware Setup

 \rightarrow In Quartus II, the Verilog code will look like this (in blue):

//-----

// This code is generated by Terasic System Builder

//-----

module tpad_alu_display(

//////// CLOCK ////////

CLOCK_50,

CLOCK2_50,

CLOCK3_50,

//////// LED /////////

LEDG,

LEDR,

//////// SW ////////

SW, //////// VGA ///////// VGA_B, VGA_BLANK_N, VGA_CLK, VGA_G, VGA_HS, VGA_R, VGA_SYNC_N, VGA_VS, //////// I2C for HSMC //////// I2C_SCLK, I2C_SDAT, /////// HSMC, HSMC connect to LTC - 8" LCD/Touch/Camera //////// CAMERA_D, CAMERA_FVAL, CAMERA_LVAL, CAMERA_PIXCLK, CAMERA_RESET_N, CAMERA_SCLK, CAMERA_SDATA, CAMERA_STROBE, CAMERA_TRIGGER, CAMERA_XCLKIN, LCD_B, LCD_DEN, LCD_DIM, LCD_G, LCD_NCLK, LCD_R, TOUCH_BUSY,

TOUCH_CS_N,

TOUCH_DCLK,

TOUCH_DIN,

TOUCH_DOUT,

TOUCH_PENIRQ_N

);

//----// PARAMETER declarations
//-----// PORT declarations

//-----

//////// CLOCK /////////

input	CLOCK_50;
input	CLOCK2_50;
input	CLOCK3_50;

//////// LED ////////

output	[8:0]	LEDG;
output	[17:0]	LEDR;
//////// sw ////	//////	
input	[17:0]	SW;
//////// VGA ///	//////	
output	[7:0]	VGA_B;
output		VGA_BLANK_N;
output		VGA_CLK;
output	[7:0]	VGA_G;
output		VGA_HS;
output	[7:0]	VGA_R;
output		VGA_SYNC_N;
output		VGA_VS;

//////// I2C for HSMC ////////

output		I2C_SCLK;
inout		I2C_SDAT;
//////// HSMC, HSM	IC connect to LTC - 3	8" LCD/Touch/Camera ////////
input	[11:0]	CAMERA_D;
input		CAMERA_FVAL;
input		CAMERA_LVAL;
input		CAMERA_PIXCLK;
output		CAMERA_RESET_N;
output		CAMERA_SCLK;
inout		CAMERA_SDATA;
input		CAMERA_STROBE;
output		CAMERA_TRIGGER;
output		CAMERA_XCLKIN;
output	[5:0]	LCD_B;
output		LCD_DEN;
output		LCD_DIM;
output	[5:0]	LCD_G;
output		LCD_NCLK;
output	[5:0]	LCD_R;
input		TOUCH_BUSY;
output		TOUCH_CS_N;
output		TOUCH_DCLK;
output		TOUCH_DIN;
input		TOUCH_DOUT;
input		TOUCH_PENIRQ_N;
//		
// REG/WIRE declaration	ons	
//======		
//		
// Structural coding		

//-----

Endmodule

Step 3: SOPC Builder Hardware Setup

Open SOPC Builder Window and add:

→CPU

- \rightarrow On Chip memory
- \rightarrow Character Buffer with DMA
- \rightarrow Pixel Buffer
- \rightarrow Pixel Buffer with DMA
- → Pixel RGB Resampler
- → Pixel Scaler
- \rightarrow Video Clipper
- → Alpha Blender
- → Dual Clock FIFO
- → VGA Controller
- →JTAG UART
- →sysid
- →Touch Panel SPI
- →Touch Panel penirq
- →Touch Panel Busy

 \rightarrow Altpll_0

Step 3a: Go to the "Connections" column and connect the following ports:

- c. For all the components connect the clock input and outputs to clock_50
- d. For all the components connect the Avalon memory mapped slave to the On-chip memory AMMS.
- e. Open the **Nios II processor named CPU** and change the reset vector and exception vectors to onchip_memory2

Step 3b: For assignment of base addresses in SOPC Builder:

 \rightarrow Click on "Auto assign base addresses" on the main menu bar and "Auto assign IRQ's" as shown in figure below:



The complete SOPC Builder system is shown below:

Use	Connections	Name	Description	Clock	Base	End	IRQ	Tags
V		E CPU	Nios II Processor	[clk]				
		instruction_master	Avalon Memory Mapped Master	altpll_sys				
		data_master	Avalon Memory Mapped Master	[clk]	IRQ 0	IRQ 31	\leftarrow	
	$\uparrow \uparrow \longrightarrow$	jtag_debug_module	Avalon Memory Mapped Slave	[clk]		0x00100fff		
\checkmark		Onchip_Memory	On-Chip Memory (RAM or ROM)	[clk1]				
	\searrow	s1 /	Avalon Memory Mapped Slave	altpll_sys		0x000f1fff		
V		Char_Buffer_with_D	Character Buffer for VGA Display	[clock_reset]				
	→ →	avalon_char_control	Avalon Memory Mapped Slave	altpll_sys		0x00101867		
	·→	avalon_char_buffer_s	Avalon Memory Mapped Slave	[clock_reset]		0x001017ff		
_		avalon_char_source	Avalon Streaming Source	[clock_reset]				
1		Pixel_Buffer	SRAM/SSRAM Controller	[clock_reset]				
	\rightarrow	avalon_sram_slave	Avalon Memory Mapped Slave	altpll_sys	= 0x00000000	0x0007ffff	_	
		E Pixel_Buffer_DMA	Pixel Buffer DMA Controller	[clock_reset]				
		avaion_pixei_dma_ma /	Avalon Memory Mapped Master	altpli_sys				
		avaion_control_slave	Avalon Memory Mapped Slave	[clock_reset]	- 0x00101800	0x00101801		
		avaion_pixei_source	Avaion Streaming Source	[clock_reset]				
~		Pixel_RGB_Resampler	KGB Resampler	[Clock_reset]				
		avalon_rgb_slitk	Avalon Streaming Sink	anpii_sys				
		Divel Scaler	Scalar	[clock_reset]				
		avalon scaler sink	Avalon Streaming Sink	altoli svs				
		avalon scaler source	Avalon Streaming Source	[clock_reset]				
		E video clipper	Clipper	[clock_reset]				
		avalon clipper sink	Avalon Streaming Sink	altoll sys				
		avalon clipper source	Avalon Streaming Source	[clock reset]				
		E Alpha Blender	Alpha Blender	[clock reset]				
		avalon foreground sink	Avalon Streaming Sink	altpll sys				
		avalon_background_si	Avalon Streaming Sink	[clock_reset]				
		avalon_blended_source	Avalon Streaming Source	[clock_reset]				
V		Dual_Clock_FIFO	Dual-Clock FIFO					
_		avalon_dc_buffer_sink	Avalon Streaming Sink	altpll_sys				
	\sim	avalon_dc_buffer_so /	Avalon Streaming Source	altpll_pclk				
V		UGA_Controller	/GA Controller	[clock_reset]				
		avaion_background_s	si Avaion Streaming Sink	[CIUCK_res	eu			
		avalon_blended_sour	ce Avalon Streaming Source	[clock_res	iet]			
1		Dual_Clock_FIFO	Dual-Clock FIFO					
		avalon_dc_buffer_sir	Avalon Streaming Sink	altpli_sys	5 Iz			
		E VGA Controller	VGA Controller	[clock_res	et]			
	·	→ avalon_vga_sink	Avalon Streaming Sink	altpll_pcl	k			
1	1	E SW1	PIO (Parallel I/O)	[clk]				
-		→ s1	Avalon Memory Mapped Slave	altpll_sys	s = 0x001018	0x0010181f		
		⊟ SW2	PIO (Parallel I/O)	[clk]				
	1	S1	PIO (Parallel I/O)	artpii_sys	s 0x001018	20 0x0010182f		
	' <u> </u>	→ s1	Avalon Memory Mapped Slave	altoll svs	s 0x001018	30 0x0010183f		
		□ SW4	PIO (Parallel VO)	[clk]				
		→ s1	Avalon Memory Mapped Slave	clk_50		350 0x0010185f		
1	9 L	⊟ jtag_uart	JTAG UART	[clk]				<u></u> Ц
-		→ avalon_jtag_slave	Avaion Memory Mapped Slave	altpll_sys	S = 0x001018	0x0010186f	P	
	' 	control slave	Avalon Memory Mapped Slave	altoll eve		70 0x00101877		
	1	touch panel spi	SPI (3 Wire Serial)	[clk]	- OXOOIDIG	04001010//		
	Ĩ 	→ spi_control_port	Avalon Memory Mapped Slave	altpll_io	■ 0x000800	00 0x0008001f	⊳	——桁
		touch_panel_penirq	n PIO (Parallel VO)	[clk]				_
		→ s1	Avalon Memory Mapped Slave	altpll_io	= 0x000800	020 0x0008002f		
1		touch_panel_busy	PIO (Parallel VO)	[clk]				
	1	- si E altoli 0	Avalon Memory Mapped Slave	[inclk_inter	- 0x000800	SU UXUUUSU03E		
		→ pll slave	Avalon Memory Mapped Slave	clk 50	- 0x000800	40 0x0008004f		

Note: If you wish to open the complete already designed hardware in SOPC builder, you may open the file "Video_system.sopcinfo" which is attached to this tutorial.

Step 3c: Click on Generate.

ile Edit Module System View Tools	Nios II	Help									
System Contents System Generation											
Component Library	Targ	jet	Clock Sett	ings							
Project	Devic	ce Family Cyclone IV E 🚽	Name		Source		N	IHz		A	
New component Hemories and Memory Contro Horizasic Technologies Inc. Library			clk_ext clk_sdra clk_pixel clk_svs	clk_ext E clk_staram a clk_pixel a clk.svs a		External altpil.c0 altpil.c1 altpil.c2			50.0 100.0 40.0 100.0		
Bridges and Adapters	Use	Connections	1	Module Name	Description	Clock	Base	End	Tags	IRQ	
Interface Protocols E-Legacy Components	7			∃ altpll pl_slave	Avaion ALTPLL Avaion Memory Mapped Slave	clk_ext	@ 0x0a411060	0x0a41106f			
Memories and Memory Contro Peripherals Debug and Performance Display	V		⁶	 cpu instruction_master data_master jtag_debug_module 	Nios II Processor Avalon Memory Mapped Master Avalon Memory Mapped Master Avalon Memory Mapped Slave	clk_sys	IRQ 0 # 0x0a410800) IRQ 31 0x0a410fff		5	
Interval Timer	V V		C	∃ jtag_uart avalon_jtag_slave ∃ sysid	JTAG UART Avalon Memory Mapped Slave System ID Peripheral	clk_sys		0x0a411077)—Q	
· ● PIO (Parallel VO) ⊕-Multiprocessor Coordinati ⊕-PLL	V			controLslave ∃ sdram s1	Avaion Memory Mapped Slave SDRAM Controller Avaion Memory Mapped Slave	clk_sys	 0x0a411078 0x00000000 	0x0a41107f 0x07ffffff			
Processor Additions Processors SLS			^e	tri_state_bridge_flash avalon_slave tristate_master	Avalon-MM Tristate Bridge Avalon Memory Mapped Slave Avalon Memory Mapped Tristate Ma	clk_sys					
			E	∃ cfi_flash s1 ⊐_sram	Flash Memory Interface (CFI) Avalon Memory Mapped Tristate Sla TERASIC SRAM	ve clk_sys	≓ 0x09800000	0x09ffffff			
New Edit Add	Ren	nove Edit 🗶 🔺		Z Address Map	Fiters Fiter: Default						
 Error: cpu: Reset vector points to the Info: peripheral_bridge: Only asset Info: cot_flash: Fissh memory capacit Info: cot_panel_pen_irgn: PIO Info: touch_panel_busy: PIO inputs 	memory t byteen ty: 8.0 Mi nputs are are not h	cfi_flash which is not connected ables corresponding to the data v Bytes (8388608 bytes). e not hardwired in test bench. Undefine hardwired in test bench. Undefine	I to the Nios I widths of (sy defined value ed values will	Il processor. Please reconnect rs_clk_timer.s1, touch_panel_s es will be read from PIO inputs I be read from PIO inputs during	/re-enable the memory or reconfigure pi.spi_control_port, touch_panel_pen_ during simulation. simulation.	the processor. irq_n.s1, touch_p	anel_busy.s1} when a	ccessing them.			

Step 3(d): After you generate the system. Following code is generated:

system (

// 1) global signals:

.clk_0(),

.clocks_VGA_CLK_40_out(),

.clocks_VGA_CLK_out(),

.clocks_sys_clk_out(),

.reset_n(),

// the_SW

.in_port_to_the_SW(),

// the_video_vga_controller

.VGA_BLANK_from_the_video_vga_controller(),

.VGA_B_from_the_video_vga_controller(),

.VGA_CLK_from_the_video_vga_controller(),

.VGA_DATA_EN_from_the_video_vga_controller(),

.VGA_G_from_the_video_vga_controller(),

.VGA_HS_from_the_video_vga_controller(),

.VGA_R_from_the_video_vga_controller(),

```
.VGA_SYNC_from_the_video_vga_controller(),
.VGA_VS_from_the_video_vga_controller()
```

Step 3(e): This code should be copied and pasted in the main Verilog (shown previously) under REG/WIRE declarations section. The modifications are shown in green:

system (
// 1) global signals:
.clk_0(CLOCK_50),
.clocks_VGA_CLK_40_out(),
.clocks_VGA_CLK_out(),
.clocks_sys_clk_out(),
.reset_n(SW[17]),
// the_SW
.in_port_to_the_SW(),
<pre>// the_video_vga_controller</pre>
.VGA_BLANK_from_the_video_vga_controller(),
.VGA_B_from_the_video_vga_controller(B),
.VGA_CLK_from_the_video_vga_controller(LCD_NCLK),
.VGA_DATA_EN_from_the_video_vga_controller(LCD_DEN),
.VGA_G_from_the_video_vga_controller(G),
.VGA_HS_from_the_video_vga_controller(),
.VGA_R_from_the_video_vga_controller(R),
.VGA_SYNC_from_the_video_vga_controller(),
.VGA_VS_from_the_video_vga_controller()

)

Step 3(f): Compile and run the system.

With this step, the hardware simulation is complete.

Software Setup

This step relies on the .sopcinfo file created when generating the SOPC Builder system

Step 1: Open Nios II SBT for Eclipse

- a) Indicate workspace as your project directory, and create a new file called "Software" and click "Okay"
- b) Set perspective to Nios II: Menu -> Window -> Open Perspective -> Other -> Nios II
- c) Menu -> File -> New -> Nios II Application and BSP from Template

(🔿 🛚	Nios II - dsaf/hell	o_world.c - Eclipse	
File	Edit Source	Refactor Navigate Search	Project Run Nios II Window Help
	New	Alt+Shift+N ▶	 Nios II Application and BSP from Template
	Open File		🕅 Nios II Application
	Close Close All	Ctrl+W Ctrl+Shift+W	Nios II Board Support Package Nios II Library Project
	Save Save As	Ctrl+S	Other Ctrl+N

- i) Under "Target Hardware Information" select file <directory>\nios.sopcinfo
- ii) Under "Application Project" type "Binary Adder" as "Project Name"
- iii) Under "Project Template" select "helloWorld"
- iv) Click "Finish"



Basic Software Algorithm

\rightarrow Initialize the screen

```
screen_x = 319; screen_y = 239;
```

char text[16];

color = 0x0000; // black color

VGA_box (0, 0, screen_x, screen_y, color); // fill the screen with background

 \rightarrow Values of switches are pointed by allocating their base address

volatile int * switch1_ptr = (int *) 0x00101810; volatile int * switch2_ptr = (int *) 0x00101820; volatile int * switch3_ptr = (int *) 0x00101830; volatile int * switch4_ptr = (int *) 0x00101850;

 \rightarrow According to the switch position, the operation of ALU is decided.

00 : Addition 01: Subtraction 10: Logical OR 11 : Logical And

if (sel1&sel2)

```
{
sprintf(text_top_VGA, "My ALU");
sprintf(text,"%d + %d = %d ",number1,number2,number1 + number2);
}
else if (!sel1&sel2)
{
sprintf(text_top_VGA, "My ALU");
else if (sel1&!sel2)
{
sprintf(text_top_VGA, "My ALU");
```

```
sprintf (text,"%d & %d = %d ",number1,number2,number1 & number2);
}
else
{
sprintf( text_top_VGA, "My ALU");
sprintf (text,"%d | %d = %d ",number1,number2,number1 | number2);
}
```

 \rightarrow Characters are written on the screen through "VGA_text" function.

```
void VGA_text(int x, int y, char * text_ptr)
{
    int offset;
    volatile char * character_buffer = (char *) 0x00101000; // VGA character buffer
    offset = (y << 7) + x;
    while ( *(text_ptr) )
    {
            *(character_buffer + offset) = *(text_ptr); // write to the character buffer
            ++text_ptr;
            ++offset;
        }
}</pre>
```

You can obtain the software code by opening the main.c file which is attached with this tutorial.

Downloading the design to the board:

ammer - C:/Us	ers/NEC	CRL/Desktop/version	on_1.5/tPad_Pictu	re_Viewer - tPad	I_Picture_View	er - [tPad_	Picture_Vi	ewer_time_lin	nited.cdf]	_		_	
t View Pro	cessing	Tools Window											
lware Setup	No Ha	rdware						Mode: JTAG	3		 Pro 	gress:	
le real-time ISP	to allow	background program	nming (for MAX II de	vices)									
Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP		
h Chur	tPad	EP4CE115F29	00AC43C8	FFFFFFF	V								
" Stop													
uto Detect													
Delete													
dd File													
ange File													
ave File	[
d Device													
աստ		ALLER											
	т		4 <u>7</u>										
Down		→ ∦ ▶											

Step 1 – For Hardware, compile the respective .sof file on the board as shown below:

Step 2 – For software, Run the software program under target as Nios II Hardware shown below:

Nios II C/C++ - main.c - Nios II IDE	and a second second second second							
ile Edit Refactor Navigate Search P	roject Tools Run Window Help							
Nios II C/C++ Projects 🛛 👘 🗖	🖻 main.c 🛙							
(#include "stdio.h"							
	#include "stdlib.h"							
tP-d Distance Viennes	#include "10.n"							
trew	#include "sys/aic_aiaim.n"							
Go Into	#include "altera avalon sgdma descriptor.h"							
Open in New Window	#include "altera_avalon_sgdma_regs.h"							
openinter tindow	_ #include "alt_types.h"							
Rebuild Index	<pre>#include "alt_video_display.h"</pre>							
Active Build Configuration	#include "fat_file.h"							
Run As	📩 Nios II Hardware							
Debug As	I Instruction Set Simulator							
Build Project	🖼 Nios II ModelSim							
Clean Project	A Lauterbach Nios II Instruction Set Simulator							
Сору								
💼 Paste	#include "skinl.h"							
Delete	#include Fonces.h							
Move	#define WIDTH 800							
Rename	#define HEIGHT 600							
Na Import	#define NUM_FRAME 8							
- A Export	#define UI_MARGIN 20							
Exportin	#define BACK_BUTTON_X_MIN_(50 - UI_MARGIN)							
🗞 Refresh	#define BACK_BUITON_X_MAA_(BACK_BUITON_X_MIN + BACKWARd_D_W + 01_MARGIN)							
Close Project	#define BACK BUTTON Y MAX (BACK BUTTON Y MIN + backward b h + UI MARGIN)							
Team	////function delaration							
Compare With	int write_buffer(alt_video_display *display_global,char *pic_name,int frame_write							
Restore from Local History	<pre>void draw_toolbox(alt_video_display *display, int frame_write_index);</pre>							
	void draw play_button(alt_video_display *display,int frame_write_index);							
Properties	- typedef struct							
System Library Properties	unsigned int frame buff[HEIGHT][WIDTH];							
	<pre>}video_frame_buffer;</pre>							
	<pre>void jpeg_decode(unsigned int *frame, FILE * infile, char * filename)</pre>							
	int width height dwidth dheight.							
	int $dx = 0$, $dv = 0$;							
	float dscaling = 0.0, output scale factor, scale factor x, scale factor y;							
	int jpeg_scaled_width=0, jpeg_scaled_height=0, output_is_scaled=0;							
	int rownum=0. colnum=0. outrow=0. outcol=0:							

Link to the Video Demonstration:

http://www.youtube.com/watch?v=gSJPt2jvn9E

<u>Chapter 5 – Integrating Touch Interface of</u> <u>T-Pad</u>

Introduction

In this chapter, the ALU will be displayed on T-Pad. Different operation of the ALU is performed by touch interface. With switches, different numbers can be displayed and their ALU operations are performed by touching the buttons on the screen.

Hardware



The T-Pad features an 8-inch Amorphous-TFT-LCD panel. The LCD Screen module offers resolution of (800x600) to provide users the best display quality for developing applications. The LCD panel supports 18-bit parallel RGB data interface.

In this chapter, touch features on the LCD Display are used. Hardware implementation to exploit the touch features on the TPad:

a) A touch_panel_spi

b) A touch_panel_busy

c) A touch_panel_penirq_n

A Serial Peripheral Interface (SPI) and a Parallel I/0 (PIO) peripheral implement the touch screen interface. The SPI peripheral communicates with the Analog Devices AD7843, touch screen digitizer chip to signal pen_move events. A single PIO captures pen interrupt events, transitions on the pen_down line from the AD7843 chip to indicate pen_down and pen_up events. The Nios II processor in the system runs the software that drives the SPI and PIO peripherals. The main commands, which we use in the project to implement the touch interface, are touch_panel_spi which implements the SPI interface are touch_panel_spi which implements pen interrupt interface.

The T-Pad has SPI for recognizing touch on a resistive screen. The touch is communicated with the processor using Serial Peripheral Interface. We need to designate two parallel input ports, one with interrupt for pen down, for recognizing that the screen is touched. The PIO with interrupt is known as pen_irq. The PIO without the interrupt is used to indicate if the touch interface is busy or not. If busy, the touch will not sense any interrupt i.e., touch on the screen.

Software

The LCD screen is initialized and ALU Options will be displayed. Switches are toggled to change the number values and for a specific ALU operation, screen is touched. The result is displayed on the LCD Screen and updated every time switch is toggled and/or screen is touched.



Step by Step ALU on T-Pad with Touch Interface Tutorial

<u>Step 1</u>: Open System Builder, select Clock, LED, VGA ans switches as shown in figure below.



Step 2 : Select HSMC Source as LTC – 8" LCD/Touch Camera as shown below.

ADIERA.		System Configuration Project Name					
		tpad_alu_display					
DE2-115 FPGA B	bard	CLOCK LED x 27 LED x 27 LED x 27 Sutton x 4 P PS2 VGA SDRAM.128MB SAM.2MB Audio Ethemet 1 TV Decoder F IR Receiver	RS-232 7-Segement x 8 Switch x 18 SD CARD LCD FLASH, 8MB FLASH, 8MB SMA USB Ethernel 2 EJTAG EEPROM, 32Kb				
		GPIO Header IO Voltage: [3.3 V (Prefix Name: None	Default) •				
Load Setting .	Generate	10 Voltage: 2.5 V (Outsuit				
Save Setting	Exit	LTC - 8" LCD/Touch	Camera ·				

Step 3 : Select a project name, for this example we are using "tpad_alu_display" as our project name Click on Generate and open the folder containing these files.

Step 4 : Open the folder where the project files are saved and open "tpad_alu_display.qpf" file. This file will be opened in quartus II.

Step 5: In Quartus II, the Verilog code will look like this (in blue):


//////// I2C for HSMC ////////

I2C_SCLK,

I2C_SDAT,

/////// HSMC, HSMC connect to LTC - 8" LCD/Touch/Camera ////////

CAMERA_D,

CAMERA_FVAL,

CAMERA_LVAL,

CAMERA_PIXCLK,

CAMERA_RESET_N,

CAMERA_SCLK,

CAMERA_SDATA,

CAMERA_STROBE,

CAMERA_TRIGGER,

CAMERA_XCLKIN,

LCD_B,

LCD_DEN,

LCD_DIM,

LCD_G,

LCD_NCLK,

LCD_R,

TOUCH_BUSY,

TOUCH_CS_N,

TOUCH_DCLK,

TOUCH_DIN,

TOUCH_DOUT,

TOUCH_PENIRQ_N

);

//-----

// PARAMETER declarations

//-----

// PORT declarations

//-----

//////// CLOCK /////////

input	CLOCK_50;
input	CLOCK2_50;
input	CLOCK3_50;

//////// LED ////////

output	[8:0]	LEDG;
output	[17:0]	LEDR;

//////// SW ////////

input	[17:0]	SW;

//////// VGA ////////

output	[7:0]	VGA_B;
output		VGA_BLANK_N;
output		VGA_CLK;
output	[7:0]	VGA_G;
output		VGA_HS;
output	[7:0]	VGA_R;
output		VGA_SYNC_N;
output		VGA_VS;

//////// I2C for HSMC ////////

output	I2C_SCLK;
inout	I2C_SDAT;

/////// HSMC, HSMC connect to LTC - 8" LCD/Touch/Camera ////////

input	[11:0]	CAMERA_D;
input		CAMERA_FVAL;
input		CAMERA_LVAL;
input		CAMERA_PIXCLK;
output		CAMERA_RESET_N;
output		CAMERA_SCLK;
inout		CAMERA_SDATA;
input		CAMERA_STROBE;
output		CAMERA_TRIGGER;
output		CAMERA_XCLKIN;
output	[5:0]	LCD_B;
output		LCD_DEN;
output		LCD_DIM;
output	[5:0]	LCD_G;
output		LCD_NCLK;
output	[5:0]	LCD_R;
input		TOUCH_BUSY;
output		TOUCH_CS_N;
output		TOUCH_DCLK;
output		TOUCH_DIN;
input		TOUCH_DOUT;
input		TOUCH_PENIRQ_N;

//-----

// REG/WIRE declarations

//-----

//-----

// Structural coding

//-----

Endmodule

Step 6: Open SOPC Builder Window and add:

→CPU

- \rightarrow On Chip memory
- \rightarrow Character Buffer with DMA
- \rightarrow Pixel Buffer
- \rightarrow Pixel Buffer with DMA
- \rightarrow Pixel RGB Resampler
- \rightarrow Pixel Scaler
- \rightarrow Video Clipper
- \rightarrow Alpha Blender
- \rightarrow Dual Clock FIFO
- \rightarrow VGA Controller
- →JTAG UART
- →SYSID
- \rightarrow Touch Panel SPI
- →Touch Panel penirq
- →Touch Panel Busy
- \rightarrow Altpll_0

Step 7: Go to the "Connections" column and connect the following ports:

- f. For all the components connect the clock input and outputs to clock_50
- g. For all the components connect the Avalon memory mapped slave to the On-chip memory AMMS.
- h. Open the **Nios II processor named CPU** and change the reset vector and exception vectors to onchip_memory2

Step 8: For assignment of base addresses in SOPC Builder:

 \rightarrow Click on "Auto assign base addresses" on the main menu bar and "Auto assign IRQ's" as shown in figure below:

ile Edit Module Sy	stem View Tools	Nios II	Help			
System Contents	Auto-Assign Ba	se Addr	esses			
Component Librar	Auto-Assign IR0	۵s		Clock Settings		
component Elerar	Insert Avalon-S	T Adapte	ers 🥏			-
Project			O, CHARLE	✓ Name		Source
New comp	onent			clk_ext		External
- Memories and I	Memory Controllers			clk_sd.em		altpll.c0
	la sias la s			clk_pixel		altpll.c1
ilibrariu	biogles inc.			cik svs		altoil.c2
Avalon Verifics	ation Suite	Ilea	Connections	Modula Name	Description	Clock
Bridges and Ar	Intere	0.00	Connections			CIOCK
Interface Proto	cole			E cpu	Nios II Processor	
E legacy Compo	nents			instruction_master	Avalon Memory Mapped Master	cik_sys
Memories and I	Memory Controller			data_master	Avaion Memory Mapped Master	
-Peripherals	including controllent			jtag_debug_module	Avaion Memory Mapped Slave	
Debug and	Performance			E jtag_uart	JTAG DART	-11
Display		1		avaion_tag_slave	Avaion memory mapped Slave	cik_sys
FPGA Peric	oherals				Avalor Memory Mannad Slave	ally ant
-Microcontro	oller Peripherals	100		pi_siave	Avaion menory mapped Slave	CIK_EXL
Inte	erval Timer				Avalan Memory Manned Slave	olk ovo
0 PIC	(Parallel VO)	877			SDBAM Controller	CIK_SYS
Multiproces	sor Coordination				Avalon Memory Manned Slave	cik svs
PLL			11 1	E tri state bridge flas	Avalon-MM Tristate Bridge	cin_bju
+ Processor Add	litions			avalon slave	Avalon Memory Manned Slave	cik svs
Processors					Avalon Memory Mapped Tristate Master	ciii_cijo
-SLS				E cfi flash	Elash Memory Interface (CEI)	
· Video and Imag	e Processing				Avalon Memory Mapped Tristate Slave	cik svs
				E sram	TERASIC SRAM	
				avalon slave	Avalon Memory Mapped Slave	clk svs
				descriptor mem	On-Chip Memory (RAM or ROM)	- /
				s1	Avalon Memory Mapped Slave	cik sys
				sqdma pixel	Scatter-Gather DMA Controller	
				csr	Avalon Memory Mapped Slave	clk_sys
				descriptor_read	Avalon Memory Mapped Master	
				descriptor_write	Avalon Memory Mapped Master	
				m_read	Avalon Memory Mapped Master	
				out	Avalon Streaming Source	
				timing_adapter	Avalon-ST Timing Adapter	
				in	Avalon Streaming Sink	clk_sys
				out	Avalon Streaming Source	
		V		🖃 fifo	On-Chip FIFO Memory	
					Avalon Streaming Sink	clk_sys
				in_csr	Avalon Memory Mapped Slave	clk_sys
	Sec. 1		1 1	J 7	أما منا	f

The complete SOPC Builder system is shown below:

Use	Connections	Name	Description	Clock	Base	End	IRQ	Tags
V		E CPU	Nios II Processor	[clk]				
			Avalon Memory Mapped Master	altpll_sys				
			Avalon Memory Mapped Master	[clk]	IRQ 0	IRQ 31	\leftarrow	
		→ jtag_debug_module	Avaion Memory Mapped Slave	[clk]		0x00100fff		
1		Onchip_Memory	On-Chip Memory (RAM or ROM)	[clk1]				
		→ s1	Avaion Memory Mapped Slave	altpll_sys		0x000f1fff		
V		Char_Buffer_with_D	Character Buffer for VGA Display	[clock_reset]				
		→ avalon_char_control	Avaion Memory Mapped Slave	altpll_sys		0x00101867		
		→ avalon_char_buffer_s.	Avaion Memory Mapped Slave	[clock_reset]		0x001017ff		
			Avalon Streaming Source	[clock_reset]				
V		Pixel_Buffer	SRAM/SSRAM Controller	[clock_reset]				
		→ avalon_sram_slave	Avaion Memory Mapped Slave	altpll_sys		0x0007ffff		
V		Pixel_Buffer_DMA	Pixel Buffer DMA Controller	[clock_reset]				
			Avaion Memory Mapped Master	altpll_sys				
		→ avalon_control_slave	Avaion Memory Mapped Slave	[clock_reset]		0x0010180f		
			Avalon Streaming Source	[clock_reset]				
V		Pixel_RGB_Resampler	RGB Resampler	[clock_reset]				
		→ avalon_rgb_sink	Avalon Streaming Sink	altpll_sys				
			Avalon Streaming Source	[clock_reset]				
V		Pixel_Scaler	Scaler	[clock_reset]				
		→ avalon_scaler_sink	Avalon Streaming Sink	altpll_sys				
			Avalon Streaming Source	[clock_reset]				
V		video_clipper	Clipper	[clock_reset]				
		→ avalon_clipper_sink	Avalon Streaming Sink	altpll_sys				
		→ avalon_clipper_source	Avalon Streaming Source	[clock_reset]				
V		Alpha_Blender	Alpha Blender	[clock_reset]				
		→ avalon_foreground_sin	k Avalon Streaming Sink	altpll_sys				
		→ avalon_background_si.	Avalon Streaming Sink	[clock_reset]				
		avalon_blended_sourc	Avalon Streaming Source	[clock_reset]				
1		Dual_Clock_FIFO	Dual-Clock FIFO					
		→ avalon_dc_buffer_sink	Avalon Streaming Sink	altpll_sys				
		avalon_dc_buffer_so	Avalon Streaming Source	altpll_pclk				
V		VGA_Controller	VGA Controller	[clock_reset]				

	avaion_background_si.	Avaion Streaming Sink	[ciock_reset]					
	avalon_blended_source	Avalon Streaming Source	[clock_reset]					
1	Dual_Clock_FIFO	Dual-Clock FIFO						
	avalon_dc_buffer_sink	Avalon Streaming Sink	altpll_sys					
	avalon_dc_buffer_so	Avalon Streaming Source	altpll_pclk					
V	□ VGA_Controller	VGA Controller	[clock_reset]					
	avalon_vga_sink	Avalon Streaming Sink	altpll_pclk					
1	□ SW1	PIO (Parallel I/O)	[clk]					
	→ s1	Avaion Memory Mapped Slave	altpll_sys	÷.	0x00101810	0x0010181f		
V	⊡ SW2	PIO (Parallel I/O)	[clk]					
	→ s1	Avaion Memory Mapped Slave	altpll_sys	÷.	0x00101820	0x0010182f		
1	E SW3	PIO (Parallel I/O)	[clk]					
	∽	Avaion Memory Mapped Slave	altpll_sys	÷.	0x00101830	0x0010183f		
V	⊡ SW4	PIO (Parallel I/O)	[Clk]					
	▶	Avaion Memory Mapped Slave	clk_50	шP.	0x00101850	0x0010185f		
V	⊟ jtag_uart	JTAG UART	[clk]					
	Avalon_jtag_slave	Avaion Memory Mapped Slave	altpll_sys	шP.	0x00101868	0x0010186f	⊳ —6	
V	⊡ sysid	System ID Peripheral	[clk]				T	
	└────── control_slave	Avalon Memory Mapped Slave	altpll_sys	÷P.	0x00101870	0x00101877		
1	touch_panel_spi	SPI (3 Wire Serial)	[clk]					
	Spi_control_port	Avalon Memory Mapped Slave	altpll_io	÷.	0x00080000	0x0008001f	≻ –1	
V	touch_panel_penirq_n	PIO (Parallel I/O)	[clk]				-	
	►	Avalon Memory Mapped Slave	altpll_io	÷.	0x00080020	0x0008002f		
1	touch_panel_busy	PIO (Parallel I/O)	[Clk]					
	∽s1	Avaion Memory Mapped Slave	altpll_io	÷.	0x00080030	0x0008003f		
V	□ altpll_0	Avalon ALTPLL	[inclk_interfa					
	└ pll_slave	Avalon Memory Mapped Slave	clk_50	шP.	0x00080040	0x0008004f		

Note: If you wish to open the complete already designed hardware in SOPC builder, you may open the file "Video_system.sopcinfo" which is attached to this tutorial.

Step 9 : Click on Generate.

omponent Library										
omponent Library Tar										
conent Library Target Ci				lock Settings						
Project _ Dev	Device Family: Cyclone IV E 👻			e	Source	MHz				A
New component				clk ext External			6	50.0		A Per
Memories and Memory Contro				sdram	altpli.c0		1	00.0		
E-Terasic Technologies Inc.			clk_	pixel	altpll.c1		4	0.0		
ibrary			cik	RVS.	altoll.c2		It	0.0		÷
-Avaion Verification Suite	e .	Connections		Module Name	Description	Clock	Base	End	Taos	RO
Interface Protocols	2				Aurolan ALTER L					
Legacy Components				→ pil slave	Avalon Memory Manned Slave	clk ext	0x0a411060	0x0a41106f		
Memories and Memory Contro	7	(E cou	Nios I Processor	cin_cat	- 0404111000	- CAULTIOUL		
-Peripherals E	-			instruction master	Avaion Memory Mapped Master	cik svs				
Debug and Performance		<u>}</u>		-< data master	Avalon Memory Mapped Master		IRO	0 IRO 31		←
Display				→ jtag_debug_module	Avalon Memory Mapped Slave		0x0a410800	0x0a410fff		
FPGA Peripherals	1			⊟ jtag_uart	JTAG UART					
Microcontroller Peripheral				→ avalon_itag_slave	Avalon Memory Mapped Slave	clk_sys		0x0a411077		i i i i i i i i i i i i i i i i i i i
• • Interval Timer	1			sysid	System ID Peripheral					
• PIO (Parallel VO)				→ control_slave	Avalon Memory Mapped Slave	clk_sys		0x0a41107f		
Multiprocessor Coordinati	7			sdram	SDRAM Controller					
PLL Deserves à didàises				→ s1	Avalon Memory Mapped Slave	clk_sys	0x00000000	0x07ffffff		
Processor Additions	/			tri_state_bridge_flash	Avalon-MM Tristate Bridge					
USIS T				→ avalon_slave	Avalon Memory Mapped Slave	clk_sys				
				tristate_master	Avaion Memory Mapped Tristate Master					
	4			E ch_nash	Flash Memory Interface (CFI)					
× -	2				TEDACIC CDAM	CIK_SYS	0x09800000	UNUSILLIL		
lew Edit Add Re	Rem	ove Edit		Address Map	Filters Filter: Default					

Step 10 : After you generate the system. Following code is generated:

system (

- // 1) global signals:
 .clk_0(),
- .clocks_VGA_CLK_40_out(),
- .clocks_VGA_CLK_out(),
- .clocks_sys_clk_out(),

.reset_n(),

// the_SW

.in_port_to_the_SW(),

// the_video_vga_controller

.VGA_BLANK_from_the_video_vga_controller(),

.VGA_B_from_the_video_vga_controller(),

.VGA_CLK_from_the_video_vga_controller(),

.VGA_DATA_EN_from_the_video_vga_controller(),
.VGA_G_from_the_video_vga_controller(),
.VGA_HS_from_the_video_vga_controller(),
.VGA_R_from_the_video_vga_controller(),
.VGA_SYNC_from_the_video_vga_controller(),
.VGA_VS_from_the_video_vga_controller()

Step 11: This code should be copied and pasted in the main Verilog (shown previously) under REG/WIRE declarations section. The modifications are shown in green:

system (

// 1) global signals: .clk_0(CLOCK_50), .clocks_VGA_CLK_40_out(), .clocks_VGA_CLK_out(), .clocks_sys_clk_out(), .reset_n(SW[17]), // the_SW .in_port_to_the_SW(), // the_video_vga_controller .VGA_BLANK_from_the_video_vga_controller(),

.VGA_B_from_the_video_vga_controller(B),

.VGA_CLK_from_the_video_vga_controller(LCD_NCLK),

.VGA_DATA_EN_from_the_video_vga_controller(LCD_DEN),

.VGA_G_from_the_video_vga_controller(G),

.VGA_HS_from_the_video_vga_controller(),

.VGA_R_from_the_video_vga_controller(R),

```
.VGA_SYNC_from_the_video_vga_controller(),
.VGA_VS_from_the_video_vga_controller()
```

)

Step 12: Compile and run the system.

With this step, the hardware simulation is complete.

Software Setup

 \rightarrow This step relies on the .sopcinfo file created when generating the SOPC System Buider system.

→Open Nios II SBT for Eclipse

→Indicate workspace as your project directory, and create a new file called "Software" and click "Okay"

→Set perspective to Nios II: Menu -> Window -> Open Perspective -> Other -> Nios II

→Menu -> File -> New -> Nios II Application and BSP from Template

🔵 N	ios II - dsaf/hello	o_world.c - Eclipse			
File	Edit Source	Refactor Navigate Search	Projec	t Run Nios II Window Help	
	New	Alt+Shift+N ►	2	Nios II Application and BSP from Template	
	Open File		6	Nios II Application	
	Close	Ctrl+W	C	Nios II Board Support Package	
	Close All	Ctrl+Shift+W	2	Nios II Library	
	-		C2	Project	
	Save	Ctrl+S	-	Other	Ctrl+N
	Save As			outen	CUITIN

→Under "Target Hardware Information" select file <directory>\nios.sopcinfo
→Under "Application Project" type "Binary Adder" as "Project Name"
→Under "Project Template" select "helloWorld"
→Click "Finish"

Nios II Application and BSP from Ter	nplate	- 0 <mark>- X -</mark>
Nios II Software Examples Please specify a .sopcinfo file		
Target hardware information SOPC Information File name: CPU name:		
Application project Project name Use default location Project location: Project template Template Bank Project Board Diagnostics Count Binary Hello MicroC/05-II Hello World Hello World Small Memory Test Small Simple Socket Server Simple Socket Server Simple Socket Server Web Server (RGMII) Web Server (RGMII)	Template description Hello World prints 'Hello from Nios II to STDOUT. This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware For details, click Finish to create the project and refer to the readmetht file in the project directory. The BSP for this template is based on the Altera HAL operating system. For information about how this software example relates to	· · · · · · · · · · · · · · · · · · ·
(?)	< Back Next > Finish	Cancel

SOFTWARE Algorithm

```
\rightarrow Values of switches are pointed by allocating their base address
   volatile int * switch1_ptr = (int *) 0x0b081040;
   volatile int * switch2_ptr = (int *) 0x0b081060;
→ For displaying different options on the LCD Display :
         sprintf(szText," + ");
         vid_print_string_alpha(rcPlus.left+5, rcPlus.top, COLOR_WHITE, COLOR_BLACK, tahomabold_32, display,
    szText);
         vid_draw_round_corner_box ( rcPlus.left, rcPlus.top, rcPlus.right, rcPlus.bottom, 10, COLOR_WHITE,
    DO_NOT_FILL, display);
         sprintf(szText," - ");
         vid_print_string_alpha(rcMinus.left+10, rcMinus.top, COLOR_WHITE, COLOR_BLACK, tahomabold_32,
    display, szText);
         vid_draw_round_corner_box ( rcMinus.left, rcMinus.top, rcMinus.right, rcMinus.bottom, 10, COLOR_WHITE,
    DO_NOT_FILL, display);
         sprintf(szText," & ");
         vid_print_string_alpha(rcAnd.left+5, rcAnd.top, COLOR_WHITE, COLOR_BLACK, tahomabold_32, display,
    szText):
         vid_draw_round_corner_box ( rcAnd.left, rcAnd.top, rcAnd.right, rcAnd.bottom, 10, COLOR_WHITE,
    DO_NOT_FILL, display);
         sprintf(szText," | ");
         vid_print_string_alpha(rcOr.left+10, rcOr.top, COLOR_WHITE, COLOR_BLACK, tahomabold_32, display,
    szText);
         vid_draw_round_corner_box ( rcOr.left, rcOr.top, rcOr.right, rcOr.bottom, 10, COLOR_WHITE, DO_NOT_FILL,
    display);
```

➔ For touch display, different cases are referred for each option selected, which is discussed in the next section.

```
alt_touchscreen_get_pen(screen, (&pen_data.pen_down), (&pen_data.x), (&pen_data.y));
if (PtInRect(&rcPlus, pen_data.x, pen_data.y)){
    select = 0;
}
if (PtInRect(&rcMinus, pen_data.x, pen_data.y)){
    select = 1;
}
if (PtInRect(&rcAnd, pen_data.x, pen_data.y)){
    select = 2;
}
if (PtInRect(&rcOr, pen_data.x, pen_data.y)){
    select = 3;
}
```

→ For different ALU options, case statements are used.

```
switch (select)
```

```
{
```

case 0:

result = number1 + number2;

sprintf (szText,"%d (+) %d = %d ",number1,number2,result);

printf ("%d + %d = %d ",number1,number2,result);

vid_print_string_alpha(400, 300, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display, szText);

break;

case 1:

result = number1 - number2;

sprintf (szText,"%d (-) %d = %d ",number1,number2,result);

printf ("%d - %d = %d ",number1,number2,result);

vid_print_string_alpha(400, 300, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display, szText);

break;

case 2:

result = number1 & number2;

sprintf (szText,"%d (&) %d = %d ",number1,number2,result);

printf ("%d & %d = %d ",number1,number2,result);

vid_print_string_alpha(400, 300, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display,

szText); break; case 3: result = number1 | number2; sprintf (szText,"%d (|) %d = %d ",number1,number2,result); printf ("%d | %d = %d ",number1,number2,result); vid_print_string_alpha(400, 300, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display, szText); break; }



You can obtain the software code by opening the main.c file which is attached with this tutorial.

Downloading the design to the board

a) -For Hardware, compile the respective .sof file on the board as shown below:

rammer - C:/Users/N dit View Processin	ECRL/Desktop/versio	on_1.5/tPad_Pictur	re_Viewer - tPad	L_Picture_View	er - [tPad_	Picture_Vie	ewer_time_lin	nited.cdf]				- 0
rdware Setup) No	Hardware w background program	ming (for MAX II de	vices)				Mode: JTAG			 Pro 	gress:	
Start File	e Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP		
Stop tPad	EP4CE115F29	00AC43C8	FFFFFFF	V								
ute Detect												
uto Detect												
Delete												
Add File												
ange File												
Save File												
Saverne												
d Device												
🛍 Up	AULE											
Down												
	10 10											

b) – For software, Run the software program under target as Nios II Hardware shown below:



Link of Video Demonstration http://www.youtube.com/watch?v=nvzwhp5aRSE

Chapter 6: Video Generation for Text and Image Display on T-Pad

Introduction

In this chapter, the ALU will be displayed on T-Pad with an image in the background. Terasic T-Pad provides a touch screen, which enables us to incorporate a video component. The strong multimedia capabilities of the T-Pad are used to develop an application that would ease the process of viewing an image from a SD Card. SD Card is used to access the images/pictures because every *Digital single-lens reflex (DSLR)* camera used in the modern day stores clicked images on it. Moreover, these images are generally in JPEG format and hence the user can store a large quantity of images on the card.

The ALU with image at the background supports following functionalities

- 1. Mounting a SD Card and reading files from it.
- 2. Displaying pictures on the touch screen display.
- 3. A simple ALU on the top of the image.
- 4. Intuitive touch to perform various functions of the ALU.

Hardware Description:



The hardware can be broken down in the following subsystems.

- 1. Memory Subsystem
- 2. Video Pipeline Subsystem
- 3. Touch Panel Subsystem

Memory Subsystem

The FPGA provides multiple options for memory storage. It provides on chip memory, off chip SRAM FLASH and SDRAM and a SD Card SPI interface. In this chapter, SDRAM is used as a source for the Scatter Gather DMA for VGA controller. SRAM is not used for this particular design. On-chip memory of the Cyclone 4 FPGA is used to store local data for the application program run on Nios II processor. The stack for the application is built in the on chip memory itself for faster access. Flash memory is included in the system for program code storage. Flash programmer in Nios II IDE is used to program the code into the flash. When this is done, the FPGA will boot up with the Nios II processor for image display and the application will load automatically.

The SD Card controller needs to be included to provide appropriate control and data signals for SPI interface, which connects the SD Card socket to the processor. Image from the SD Card will be read out using simple memory pointers. The SDRAM is used as a frame buffer to store images for the VGA controller to read. The SD Card cannot feed images to the VGA controller and hence SDRAM is required as a buffer to connect to the VGA controller.

Software Design

For software implementation the VGA module, touch screen and SD Card are initialized first. Then a home screen appears on the T-Pad which allows the user to touch and initialize the SD Card to read the images. After the image is displayed by the pixel buffer on the T-Pad, the character buffer displays the ALU options. After the user selects/touch one of the options of ALU, the character buffer is refreshed and the result is displayed. After the result is displayed, the software waits for the touch input to any other option and respective results are shown.

For initializing the touch screen and the VGA, Hardware Abstraction Layer is used provided by Altera. The SD card controller is initialized using SPI. Once these components are initialized, the home screen appears on the T-Pad's screen. After the user selects a particular mode, images are read one by one from the SD Card. Then using JPEG library, we decode the image data into hex format. This hex data is passed to the buffer and pushed into video pipeline. Image corresponding to this data is then displayed. The latency with which the image is displayed depends on the size of the hex data and therefore ultimately on the image size. The flow chart for the software is shown below:



Step by Step ALU with image in background Tutorial

Hardware Setup

<u>Step 1</u>: Open System Builder, select Clock, SDRAM, SRAM, FLASH, SD CARD, VGA and LTC – 8" LCD/Touch/Camera as shown in figure below.

Terasic DE2-115 System Builder V 1.0.0				
		-System Configuration- Project Name: picture_alu		
DE2-115 FPGA Bo	bard	 ✓ CLOCK □ LED x 27 □ Button x 4 □ PS2 ✓ VGA ✓ SDRAM, 128MB ✓ SRAM, 2MB □ Audio □ Ethernet 1 □ TV Decoder □ IR Receiver GPIO Header IO Voltage: 3.3 V (I Prefix Name: None HSMC 	□ RS-232 □ 7-Segement x 8 □ Switch x 18 ☑ SD CARD ☑ LCD ☑ FLASH, 8MB □ SMA □ USB □ Ethernet 2 □ EJTAG □ EEPROM, 32Kb	
Load Setting	Generate	IO Voltage: 2.5 V (I Prefix Name:	Default) 🔽	
Save Setting	Exit	LTC - 8" LCD/Touch/	Camera 🔄 🏭	

<u>Step 2</u> : Select a project name, for this example we are using "picture_alu" as our project name. Click on Generate and open the folder containing these files.

<u>Step 3</u>: Open the folder where the project files are saved and open picture_alu.qpf file. This file is generated when we generate in the above step. Open the respective folder for this file. This file will be opened in Quartus II.

<u>Step 4</u>: In Quartus II, the Verilog code will look like this (in blue):

//_____

// This code is generated by Terasic System Builder

module picture_alu(

///////// CLOCK /////////

CLOCK_50,

CLOCK2_50,

CLOCK3_50,

///////// LCD /////////

LCD_BLON,

LCD_DATA,

LCD_EN,

LCD_ON,

LCD_RS,

LCD_RW,

//////// SDCARD ////////

SD_CLK,

SD_CMD,

SD_DAT,

SD_WP_N,

VGA_B,

VGA_BLANK_N,

VGA_CLK,

VGA_G,

VGA_HS,

VGA_R,

VGA_SYNC_N,

VGA_VS,

///////// I2C for HSMC ////////

I2C_SCLK,

I2C_SDAT,

//////// SDRAM ////////

DRAM_ADDR,

DRAM_BA,

DRAM_CAS_N,

DRAM_CKE,

DRAM_CLK,

DRAM_CS_N,

DRAM_DQ,

DRAM_DQM,

DRAM_RAS_N,

DRAM_WE_N,

//////// SRAM ////////

SRAM_ADDR,

SRAM_CE_N,

SRAM_DQ,

SRAM_LB_N,

SRAM_OE_N,

SRAM_UB_N,

SRAM_WE_N,

//////// Flash /////////

FL_ADDR,

 FL_CE_N ,

FL_DQ,

FL_OE_N,

FL_RST_N,

 FL_RY ,

FL_WE_N,

FL_WP_N,

/////// HSMC, HSMC connect to LTC - 8" LCD/Touch/Camera ////////

CAMERA_D,

CAMERA_FVAL,

CAMERA_LVAL,

CAMERA_PIXCLK,

CAMERA_RESET_N,

CAMERA_SCLK,

CAMERA_SDATA,

CAMERA_STROBE,

CAMERA_TRIGGER,

CAMERA_XCLKIN,

LCD_B,

LCD_DEN,

LCD_DIM,

LCD_G,

LCD_NCLK,

LCD_R,

TOUCH_BUSY,

TOUCH_CS_N,

TOUCH_DCLK,

TOUCH_DIN,

TOUCH_DOUT,

TOUCH_PENIRQ_N

);

// PARAMETER declarations

// PORT declarations

///////// CLOCK /////////

input	<i>CLOCK_50;</i>
input	<i>CLOCK2_50;</i>
input	CLOCK3_50;

///////// LCD /////////

output		LCD_BLON;
inout	[7:0]	LCD_DATA;
output		LCD_EN;
output		LCD_ON;
output		LCD_RS;
output		LCD_RW;

//////// SDCARD ////////

output		SD_CLK;
inout		SD_CMD;
inout	[3:0]	SD_DAT;
input		SD_WP_N;

///////// VGA /////////

output	[7:0]	VGA_B;
output		VGA_BLANK_N;
output		VGA_CLK;
output	[7:0]	VGA_G;
output		VGA_HS;

output	[7:0]	VGA_R;
output		VGA_SYNC_N;
output		VGA_VS;

//////////////////////////////////////	for HSMC	////////
111111111111111111111111111111111111111	jor mone	///////////////////////////////////////

output	I2C_SCLK;
inout	I2C_SDAT;

//////// SDRAM /////////

output	[12:0]	DRAM_ADDR;
output	[1:0]	DRAM_BA;
output		DRAM_CAS_N;
output		DRAM_CKE;
output		DRAM_CLK;
output		DRAM_CS_N;
inout	[31:0]	DRAM_DQ;
output	[3:0]	DRAM_DQM;
output		DRAM_RAS_N;
output		DRAM_WE_N;

//////// SRAM /////////

output	[19:0]	SRAM_ADDR;
output		SRAM_CE_N;
inout	[15:0]	SRAM_DQ;
output		SRAM_LB_N;
output		SRAM_OE_N;
output		SRAM_UB_N;
output		SRAM_WE_N;

///////// Flash /////////

output	[22:0]	FL_ADDR;
output		FL_CE_N;

inout	[7:0]	$FL_DQ;$
output		FL_OE_N;
output		FL_RST_N;
input		FL_RY;
output		FL_WE_N;
output		FL_WP_N;

/////// HSMC, HSMC connect to LTC - 8" LCD/Touch/Camera ////////

input	[11:0]	CAMERA_D;
input		CAMERA_FVAL;
input		CAMERA_LVAL;
input		CAMERA_PIXCLK;
output		CAMERA_RESET_N;
output		CAMERA_SCLK;
inout		CAMERA_SDATA;
input		CAMERA_STROBE;
output		CAMERA_TRIGGER;
output		CAMERA_XCLKIN;
output	[5:0]	LCD_B;
output		LCD_DEN;
output		LCD_DIM;
output	[5:0]	$LCD_G;$
output		LCD_NCLK;
output	[5:0]	LCD_R;
input		TOUCH_BUSY;
output		TOUCH_CS_N;
output		TOUCH_DCLK;
output		TOUCH_DIN;
input		TOUCH_DOUT;
input		TOUCH_PENIRQ_N;

endmodule

This step initiates all the ports selected on system builder.

<u>Step 5</u>: Open SOPC Builder Window and add the following components from library (detailed procedure is explained in chapter no.3):

→alt_pll

→CPU

→System id

→SD RAM

→Tri State Bridge Flash

 \rightarrow Flash

 \rightarrow Sram

 \rightarrow On chip Memory

→SGDMA Controller

→Timing Adapter

→On Chip FIFO Memory

→Timing Adapter

 \rightarrow Pixel Converter

- →VGA Sink
- → Peripheral Bridge
- →Sd Card Controller
- →Interval Timer
- \rightarrow Touch Panel
- →Touch Panel SPI
- \rightarrow Touch Panel penirq
- \rightarrow Touch Panel Busy

<u>Step 6</u>: Go to the "Connections" column and connect the following ports:

- i. Pll_slave(system clock) is connected to cpu, jtag_uart, sysid, sdram, tri_state bridge flas, cfi_flash, sram, on chip memory, SGDMA Controller, Fifo and peripheral bridge
- j. CPU's jtag_debug_module to SDRAM, Tristate Bridge, SRAM.
- k. SDRAM to SGDMA Controller's "m_read".
- 1. Descriptor Memory to SGDMA Controller's "Descriptor Read"
- m. Descriptor Memory to SGDMA Controller's "Descriptor Write"
- n. SGDMA Controller's out to Timing Adapter's "in".
- o. Timing Adapter's "out" to FIFO's "in".
- p. FIFO's "out" to Pixel Converter's "in".
- q. Pixel Converter's "out" to VGA Sink.
- r. SD Card Controller to System Clock Timer, Touch Panel SPI, Touch_Panel_irq_n and Touch_Panel_busy.
- s. Open the **Nios II processor named CPU** and change the reset vector and exception vectors to onchip_memory2

Step 7: For assignment of base addresses in SOPC Builder:

 \rightarrow Click on "Auto assign base addresses" on the main menu bar and "Auto assign IRQ's" as shown in figure below:

Altera SOPC Builder - nios_simple.	sopc* (C:\Users\NECRL\Desktop\versi	on_1.5\nios_simple.sopc)		-						
File Edit Module System View Tools	Nios II Help									
System Contents Auto-Assign Ba	se Addresses									
Auto-Assian IRG		Clock Settings								
Insert Avalor ST	L Adaptare	- Cover Solution								
Project	Adapters VE -	Name		Source						
		clk_ext		External						
Memories and Memory Controllers		clk_su.en		altpll.c0						
SD Card		clk_pixel		altpli.c1						
		clk_svs		altoll.c2						
Library	line Connections	Hedula Nama	Department	Clock						
Available and Adapters	Use connections		Description	CIUCK						
Interface Protocols		E cpu	Nios II Processor							
E l enacy Components		instruction_master	Avaion Memory Mapped Master	CIK_SYS						
Memories and Memory Controller		data_master	Avaion Memory Mapped Master							
Peripherals		jrag_debug_module	Avaion Memory Mapped Slave							
Debug and Performance		E jtag_uart	Austea Nemeri Manned Stave	ally ava						
Display		avaion_iag_siave	Avalon ALTEL	CIK_SYS						
FPGA Peripherals			Avalon Memory Manned Slave	clk ext						
Microcontroller Peripherals		- eveid	System ID Perinheral	cin_cxt						
···· Interval Timer		control slave	Avalon Memory Mapped Slave	cik svs						
 PIO (Parallel I/O) 		sdram	SDRAM Controller							
Multiprocessor Coordination			Avalon Memory Mapped Slave	clk sys						
⊕-PLL		tri_state_bridge_flash	Avalon-MM Tristate Bridge							
Processor Additions		avalon_slave	Avaion Memory Mapped Slave	clk_sys						
Processors		tristate_master	Avalon Memory Mapped Tristate Master							
		⊟ cfi_flash	Flash Memory Interface (CFI)							
			Avalon Memory Mapped Tristate Slave	clk_sys						
		🖃 sram	TERASIC_SRAM							
		avalon_slave	Avalon Memory Mapped Slave	clk_sys						
		descriptor_mem	On-Chip Memory (RAM or ROM)							
		→ s1	Avalon Memory Mapped Slave	clk_sys						
		sgdma_pixel	Scatter-Gather DMA Controller							
			Avalon Memory Mapped Slave	clk_sys						
		descriptor_read	Avalon Memory Mapped Master							
		descriptor_write	Avalon Memory Mapped Master							
		m_read	Avalon Memory Mapped Master							
		Out	Avaion Streaming Source							
		in in in	Avalon Streaming Sink	clk eve						
		out	Avalon Streaming Source	UK_SYS						
		FI fifo	On-Chin FIEO Memory							
			Avalon Streaming Sink	cik svs						
4		in csr	Avalon Memory Mapped Slave	clk sys						
				1727-177						

The complete SOPC Builder system is shown below:



		out	Avalon Streaming Source				
V		timing_adapter	Avalon-ST Timing Adapter				
		in	Avalon Streaming Sink	clk_sys			
		out	Avalon Streaming Source				
V		⊟ fifo	On-Chip FIFO Memory				
	$ \diamond \phi \diamond \phi + \rightarrow$	in	Avalon Streaming Sink	clk_sys			
	$\diamond \bullet \bullet \diamond \diamond \bullet + + + + \bullet \bullet$	in_csr	Avalon Memory Mapped Slave	clk_sys	🖹 0x0a411040	0x0a41105f	
		out	Avalon Streaming Source	clk_pixel			
V		☐ fifo_to_pixel_converter	Avalon-ST Timing Adapter				
		in	Avalon Streaming Sink	clk_pixel			
		out	Avalon Streaming Source				
V		pixel_converter	Pixel Converter (BGR0> BGR)				
	$ \diamond \diamond \diamond \bullet \bullet \diamond + \diamond$	in	Avalon Streaming Sink	clk_pixel			
		out	Avalon Streaming Source				
1		vga_source	VGA_SINK				
		avalon_streaming_sink	Avalon Streaming Sink	clk_pixel			
1		peripheral_bridge	Avalon-MM Clock Crossing Bridge				
	$\diamond \bullet \bullet \diamond \bullet \bullet$	s1	Avalon Memory Mapped Slave	clk_sys		0x080007ff	
	$ $ $ $ \succ	m1	Avalon Memory Mapped Master	clk_periph			
V		sd_card_controller	SD Card Controller (SPI)				
	$\diamond \diamond - \diamond \diamond \diamond - \bullet $	avalon_slave	Avalon Memory Mapped Slave	clk_periph	⊜ ▲ 0x00000000	0x000003ff	
1		sys_clk_timer	Interval Timer				
	$\diamond \diamond \diamond \diamond \diamond \bullet \bullet$	s1	Avalon Memory Mapped Slave	clk_periph	🛋 0x00000400	0x0000041f	
V		touch_panel_spi	SPI (3 Wire Serial)				
	$\diamond \diamond - \diamond \diamond \diamond - \phi \rightarrow \phi$	spi_control_port	Avalon Memory Mapped Slave	clk_periph	🛋 0x00000420	0x0000043f	
1		touch_panel_pen_irq_n	PIO (Parallel VO)				
	$\diamond \diamond \diamond \diamond \diamond \bullet \bullet$	s1	Avalon Memory Mapped Slave	clk_periph	🛋 0x00000470	0x0000047f	
V		touch_panel_busy	PIO (Parallel I/O)				
	$\diamond - \diamond - \diamond - \diamond - \diamond - \bullet - \bullet \rightarrow$	s1	Avalon Memory Mapped Slave	clk_periph	i ▲ 0x00000480	0x0000048f	

If you wish to open the complete already designed hardware in SOPC builder, you may open the file "nios_simple.sopcinfo" which is attached to this tutorial.

To generate the hardware in sopc builder, click on generate shown in the figure below:

System Centers System Generation System Centers Sys) NIOS II	Help										
Component Library Target Cock Settings Project Name Survice Mit Memories and Memory Control Ke, at (k, at (k, pkel Survice) 100.0 Project Interact Prelocion Interact Prelocion 100.0 Interact Prelocion Interact Prelocion I	System Contents System Generation												
Project Name Source MH2 Image: Source Source	Component Library	Targ	et	Clock Se	ck Settings								
Improve component 640.ext External 640.ext 100.0 Improve component 640.ext 100.0 Improve component Improve component 100.0 Improve component	Project	Devic	e Family: Cyclone IV E 🚽	Name	Name Source				MHz		Add		
memory controls and Memory Controls memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory memory				clk_ext	clk_ext External					0.0		Remo	
Barana Technologies Inc. Intervent Avaion Varification See Technologies Inc. Intervent Avaion Varification See Technologies Inc. Intervent Avaion Varification See Technologies Inc. Intervent Avaion Varification Intervent Avaion Varification Technologies Inc. Intervent	Memories and Memory Contro			clk_sd	ram	altpli.c	:0		1	00.0	E	Kenio	
Library	Terasic Technologies Inc.			clk_pix	el	altpll.c	altpli.c1			40.0			
	Library	cik.				altoll.c	altoll.c?				100.0		T
We have be finded by the problem Image: the problem	Bridges and Adapters Use Connections Instructure Protocols			Module Name	Description	Clock Base			End	Taos	PO		
B. Legacy, Components Image: State Components					Avalog ALTRU		Clock			laga			
Hemory Control Peropherals Porcessor and Memory Lapped Silve Avabin Memory Mapped Silve Avabin Memory M	E-Legacy Components	ontro		>	pl slave	Avalon Memory Mappe	d Slave	clk ext	0x0a411060	0x0a41106f			
Perpherais	-Memories and Memory Contro				E cpu	Nios Il Processor		ongon					
Image: Construction of the second Addression of the se	Peripherals				instruction_master	Avalon Memory Mappe	d Master	clk_sys					
	Debug and Performance		≻		data_master	Avalon Memory Mappe	d Master		IRQ	0 IRQ 31			
Improvemental Imp	Display				jtag_debug_module	Avalon Memory Mappe	d Slave			0x0a410fff			
	Microcontroller Perinheral				⊟ jtag_uart	JTAG UART				70 0x0a411077			
● PO(Parale I/O) IV I system U-reprined in System U-reprind U-reprind U-reprined in System U-reprind U-reprined in Syst	- a Interval Timer				avalon_tag_slave	Avalon Memory Mappe	d Slave	clk_sys	= 0x0a411070				
Mudiarcessor Coordinate PL. PL	PIO (Parallel VO)	v			E sysia	System ID Peripheral	d Claus	ally and		0.000 41107.5			
⊕ PLL S1 Availab Memory Mapped Slave clk_sys ● 0x00000000 0x07££££££ ⊕ Processor Additions IV B tri_state_bridge flash Availon-IIIM Tristate Bridge III IIII IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	Multiprocessor Coordinati				G edram	SDRAM Controller	u Slave	CIK_SYS	= 0x04411078	ORDERIIO/I			
Processor Additions V Int_state_bridge_flash Avaion-MM Tristate Bridge	⊕-PLL		<u> </u>	\longrightarrow	s1	Avalon Memory Mappe	d Slave	clk sys	a 0x0000000	00 0x07ffffff			
	Processor Additions				tri_state_bridge_flash	Avalon-MM Tristate Bri	dge						
Avalon_slave Avalon Memory Mapped Slave cik_sys	Processors				avalon_slave	Avalon Memory Mappe	d Slave	clk_sys					
SLS Avalon Memory Mapped Tristate Master	÷ SLS +				tristate_master	Avalon Memory Mappe	d Tristate Master						
Cfi_flash Flash Memory Interface (CFI)	4				□ cfi_flash	Flash Memory Interface	e (CFI)						
Avaion Memory Mapped Tristate Slave clk_sys 0x09800000 0x095fffff	3 I X			\rightarrow	s1	Avalon Memory Mappe	d Tristate Slave	clk_sys	= 0x09800000	11111110809			
New. Edt. Add. Remove Edt. X A V X Address Nap. Fiters. Fiter Defaut	New Edit Add	Ren	iove Edit		Z Address Map	Filters Filter: D	efault						

After the sopc builder system is generated:

We get the following code:

wire reset_n;

wire [7:0] wire_HC_B;

wire [7:0] wire_HC_G;

wire [7:0] wire_HC_R;

assign reset_n = 1'b1;

assign $HC_DIM = 1'b1;$

nios_simple nios_simple_ins(

// 1) global signals:

.clk_ext(CLOCK_50),

.reset_n(reset_n),

//.altpll_25(),

//.altpll_io(),

.clk_sdram(DRAM_CLK),

//.clk_sdram(DRAM_CLK),

//.clk_pixel(HC_NCLK),

////VGA SINK

.vga_b_from_the_vga_source	$(wire_HC_B),$
.vga_clk_from_the_vga_source	(HC_NCLK),
.vga_de_from_the_vga_source	(HC_DEN),
.vga_g_from_the_vga_source	(wire_HC_G),
.vga_hs_from_the_vga_source	(),
.vga_r_from_the_vga_source	(wire_HC_R),
.vga_vs_from_the_vga_source	() ,

// the_sdram

.zs_addr_from_the_sdram(DRAM_ADDR),

.zs_ba_from_the_sdram(DRAM_BA),

.zs_cas_n_from_the_sdram(DRAM_CAS_N),

.zs_cke_from_the_sdram(DRAM_CKE),

.zs_cs_n_from_the_sdram(DRAM_CS_N),

.zs_dq_to_and_from_the_sdram(DRAM_DQ),

.zs_dqm_from_the_sdram(DRAM_DQM),

 $.zs_ras_n_from_the_sdram(DRAM_RAS_N),$

.zs_we_n_from_the_sdram(DRAM_WE_N),

// the_sd_card_controller

.spi_clk_from_the_sd_card_controller (SD_CLK),

.spi_cs_n_from_the_sd_card_controller	(SD_DAT[3]),
.spi_data_in_to_the_sd_card_controller	(SD_DAT[0]),
.spi_data_out_from_the_sd_card_control	ller (SD_CMD),
////cfi flash	
.tri_state_bridge_flash_address (Fi	L_ADDR),

.tri_state_bridge_flash_data	(FL_DQ),
.write_n_to_the_cfi_flash	(FL_WE_N),
.read_n_to_the_cfi_flash	(FL_OE_N),
.select_n_to_the_cfi_flash	(FL_CE_N),
////touch panel interface	
.MISO_to_the_touch_panel_spi	(HC_ADC_DOUT),
.MOSI_from_the_touch_panel_spi	(HC_ADC_DIN),
.SCLK_from_the_touch_panel_spi	(HC_ADC_DCLK),
.SS_n_from_the_touch_panel_spi	(HC_ADC_CS_N),
.in_port_to_the_touch_panel_pen_	irq_n (HC_ADC_PENIRQ_N),
.in_port_to_the_touch_panel_busy	(HC_ADC_BUSY),
////SRAM	
.SRAM_ADDR_from_the_sram	(SRAM_ADDR),
.SRAM_CE_n_from_the_sram	(SRAM_CE_N),
.SRAM_DQ_to_and_from_the_srat	m (SRAM_DQ),
.SRAM_LB_n_from_the_sram	(SRAM_LB_N),
.SRAM_OE_n_from_the_sram	(SRAM_OE_N),
.SRAM_UB_n_from_the_sram	(SRAM_UB_N),
.SRAM_WE_n_from_the_sram	(SRAM_WE_N)

);

////

assign $HC_B = wire_HC_B[7:2];$

assign $HC_G = wire_HC_G[7:2];$

assign $HC_R = wire_HC_R[7:2];$

// Flash Config

assign FL_RST_N = reset_n;

assign $FL_WP_N = 1'b1;$

// FLASH_RY,

/// LCD config assign LCD_BLON = 0; // not supported

assign LCD_ON = 1'b1; // alwasy on

This code should be copied and pasted in the main Verilog (shown previously) under REG/WIRE declarations section.

With this step our hardware configuration is done.

After this step, open Nios II IDE Eclipse and write software to configure software of the demonstration.

Software Setup

Basic Algorithm:

1) Display is initialized.

```
display_global = alt_video_display_init( "/dev/sgdma_pixel", // Name of video controller
WIDTH, // Width of display
HEIGHT, // Height of display
32, // Color depth (32 or 16)
SDRAM_BASE+SDRAM_SPAN/2, // Where we want our frame buffers
DESCRIPTOR_MEM_BASE, // Where we want our descriptors
NUM_FRAME );
```

if(display_global)

2) Touch Panel is initialized.

hTouch = Touch_Init(TOUCH_PANEL_SPI_BASE, TOUCH_PANEL_PEN_IRQ_N_BASE, TOUCH_PANEL_PEN_IRQ_N_IRQ);

3) Welcome screen is displayed. The following commands are used to display characters on the screen.

sprintf(szText,"\nCalculator ");

vid_print_string_alpha(400, 2, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display_global, szText);

sprintf(szText,"\nTouch Anywhere to Begin");

vid_print_string_alpha(220, 200, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display_global, szText);

//vid_draw_round_corner_box (300, 400, 500, 500,10, 0x5555, 0x0000, display_global);

4) After the touch is selected , pixel buffer and character buffer gets updated.

alt_video_display_clear_screen(display_global, 0x0);

result = write_buffer(display_global,name_list[pic_index],frame_write_index);

alt_video_display_register_written_buffer(display_global); ////direct the display buffer to buffer_being_written

 $while (alt_video_display_buffer_is_available (display_global)!=0); ////update \ display_global- \ >buffer_being_displayed \ >buffer_beindisplayed \ >buffer_beindisplayed \ >buffer_beindisplayed \ >b$

printf("\nLook at the the screen");

x=0; y=0;

sprintf(text_disp,"ADDITION"); vid_print_string_alpha(50, 200, COLOR_WHITE, COLOR_BLACK, tahomabold_20,

display_global, text_disp);

sprintf(text_disp,"SUBTRACTION"); vid_print_string_alpha(200, 200, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display_global, text_disp);

sprintf(text_disp,"LOGICAL AND");

vid_print_string_alpha(400, 200, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display_global, text_disp);

sprintf(text_disp,"LOGICAL OR");

vid_print_string_alpha(600, 200, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display_global, text_disp);

Touch_EmptyFifo(hTouch);

Touch_EmptyFifo(hTouch) is used to empty the touch input

5) The screen waits for the touch input in trigger area which will perform ALU functions.

For example:

if (touch == 0 && (x >= 50 && x<=150)) //Trigger area

{

printf("I am in addition loop\n x=%d y = %d", x, y);// display statement on console for debugging

```
touch = 1;
x=0;
y=0;
```

result = write_buffer(display_global,name_list[pic_index],frame_write_index);

 $sprintf(text_calc,"10 + 5 = 15");$

vid_print_string_alpha(350, 410, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display_global, text_calc); sprintf(text_disp,"ADDITION");

vid_print_string_alpha(50, 200, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display_global, text_disp); sprintf(text_disp,"SUBTRACTION");

vid_print_string_alpha(200, 200, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display_global, text_disp); sprintf(text_disp,"LOGICAL AND");

vid_print_string_alpha(400, 200, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display_global, text_disp); sprintf(text_disp,"LOGICAL OR");

vid_print_string_alpha(600, 200, COLOR_WHITE, COLOR_BLACK, tahomabold_20, display_global, text_disp);

usleep(100000);

Touch_EmptyFifo(hTouch);

goto here; // label to take it back to the main loop

6) The screen performs respective functions according to which trigger area is touched.

Alternatively you can obtain the software code by opening the main.c file which is attached with this tutorial.

Downloading the design to the Board

a) –For Hardware, compile the .sof file on the board as shown below:

🔖 Programmer - C:/U	sers/NECR	L/Desktop/versio	n_1.5/tPad_Pictur	e_Viewer - tPad	_Picture_View	er - [tPad_	Picture_Vie	ewer_time_lin	nited.cdf]		_		_ 0	X
File Edit View Pro	ocessing	Tools Window												
Hardware Setup	No Hard to allow ba	ware ackground program	ning (for MAX II de	vices)				Mode: JTAG	}		▼ Pro	gress:		
Mu Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP			
Stop	tPad_	EP4CE115F29	00AC43C8	FFFFFFF										
pole Auto Detect														
X Delete														
Add File														
Change File														
Save File														*
Add Device														-
	TDI		`I											
- Down													 	-

b) – For software, Run the software program under target as Nios II Hardware shown below:



Video Demonstration of this tutorial is available on YouTube by clicking here:

http://www.youtube.com/watch?v=_epPtQ-ITuQ