FPGA-based Voice Visualization

Supervisor: Dr. Benjamin Carrion Schafer

Student: Gao Zhendong

Date : April 13, 2015

Gao Zhendong 2015/5/26

1



Background and Objectives

Tools and Platform Used

VHDL Program Structure

- System Block Diagram
- AC97 Audio Codec Module
- LCD Module
- VGA Module

Verification and Testing

Summary and Conclusions

Acknowledgement

Background and Objectives

Background:

- Sound processing is used to many fields.
- FPGA are popular platforms for high-performance HW designs.
- The project can increase my practice ability and research capabilities which may broaden my future career.

Goals of the project:

- Learn and master the use of VHDL for configuring FPGA with Xilinx ISE .
- Use FPGA to visualize in <u>real-time</u> the quantized speech waveform on the LCD.
- Use FPGA to visualize in <u>real-time</u> the quantized speech waveform on VGA monitor and changed visualization type using switches
 - Welcome message, Histogram, moving dots, sine waveform.

Tools and Platform Used

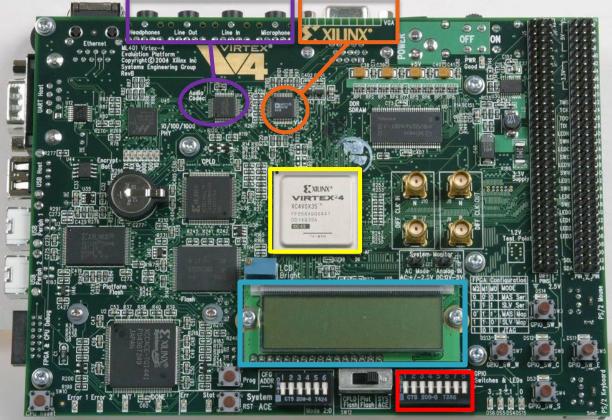
- Xilinx Virtex 4 FPGA—is an integrated circuit design that allows designers to quickly develop and prototype their HW designs.
- Xilinx ML402 board—contains single FPGA and peripherals
- ISE software—is a platform for VHDL synthesis and simulation.
- VHDL—is a popular Hardware description language





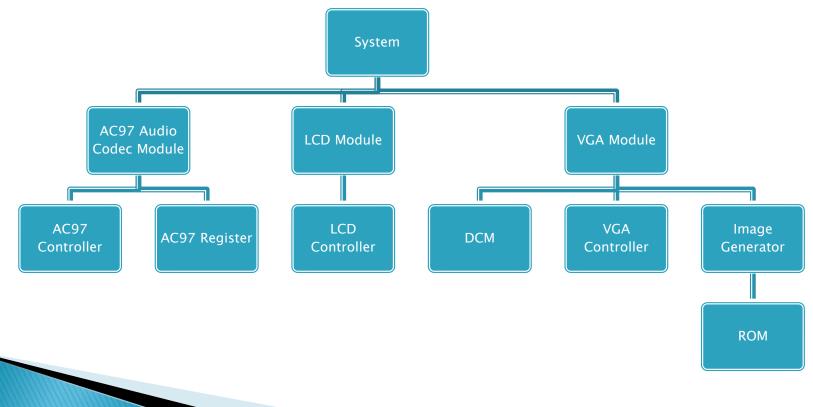
Tools and Platform Used

- AC97 Audio codec
- Video DAC
- VGA Interface
- LCD
- Switches
- FPGA



VHDL Program Structure

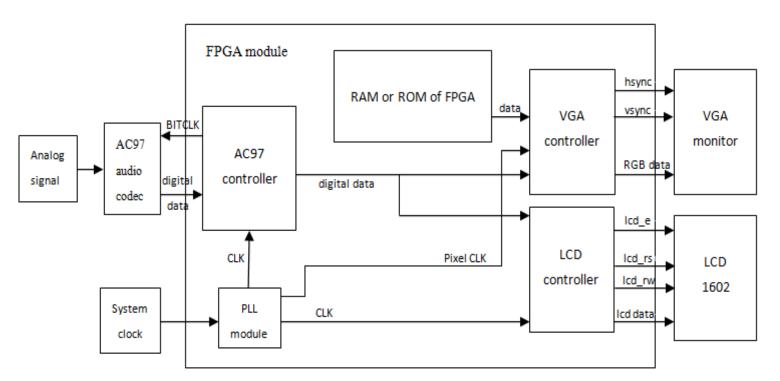
- Hierarchal design
- Top-down approach
- 9 VHDL files (include Top Module)
- 16 process
- ~6546 lines of VHDL Code



System Block Diagram

The complete system:

- Signal acquisition chip (AC97 audio codec)
- FPGA module (including PLL module)
- LCD module, ROM module and VGA controller module.
- DCM, VGA controller, image generator and ROM are categorized into VGA module.



AC97 Audio Codec Module

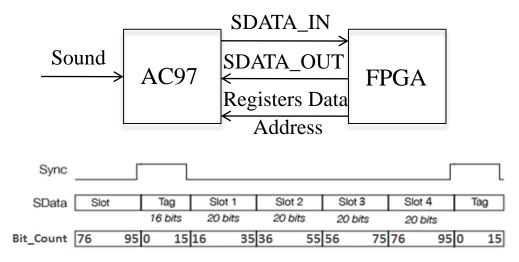
The project uses two parts (AC97 controller and AC97 cmd) to achieve AC97's work.

AC97 controller:

- Define slot 0 to slot 4 of AC97 link output frame and slot 3 and 4 of AC97 link input frame.
- Bit_count is used to calculate the length of each slot and assign the value in different slot.

AC97 cmd:

- Uses state machine to configure the registers of AC97.
- 12 states for configuring the registers.
- Refer to register map on AC97 data sheet.



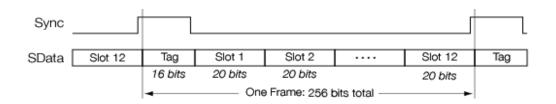
AC97 Audio Codec Module - Frame

AC97 Frames:

- It is a serial interface. Data is transmitted to and from the codec one bit at a time
- Frames are divided into twelve slots of 20 bits each, plus a 16-bit tag field.
- Slots 1 and 2 are used to read and write to configuration registers in the codec.
- Slot3 is used to send PCM data to the left channel DAC or ADC.
- Slot4 is used to send PCM data to the right channel DAC or ADC.
- Do not need to use other slots in this project.

Tag bits:

- Bit 15 is a valid flag for entire frame
- Bit 14-3 are valid flags for the individual slots in the frame.
- Bit 2-0 are zero in this project.



LCD Module

- Each character need to split into the upper 4-bits and lower 4-bits in order to be displayed.
- Users can display characters according to different ASCLL.
- There are some important interfaces of LCD needed to be mentioned briefly.

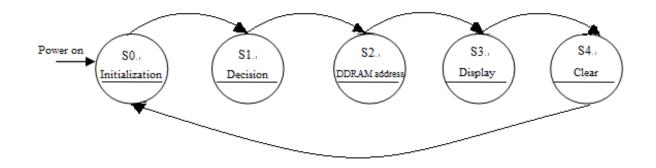
Interface Name	Mode	Description	Pin
rw	output	This interface is read/write select signal. LCD will read data during high level signal and write data during low level signal.	LCD Pin 5
rs	output	This interface is register select signal. High level signal means that data are being sent. Low level signal means that instructions are being sent.	LCD Pin 4
е	output	It is enable signal of LCD.	LCD Pin 6
lcd_data	BIDIR	It sends ASCLL values to LCD. Then LCD displays corresponding content on LCD screen.	LCD pins 7-14

LCD Module

• LCD used to receive the sampled data from AC97 and decide the range of value in order to display rough waveform on LCD screen.

A FSM is used to initialize and activate LCD and decide the range of temp:

- S0 is initialization step.
- S1 is decision of range and definition of mark signal.
- S2 is used to define the DDRAM address.
- S3 displays characters on LCD screen.
- S4 is used to clear the LCD screen and start from the initialization step (S0).



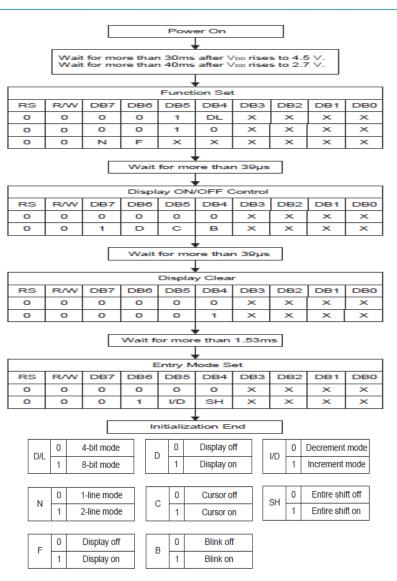
LCD Module – Initialization

Initialization processes have three steps:

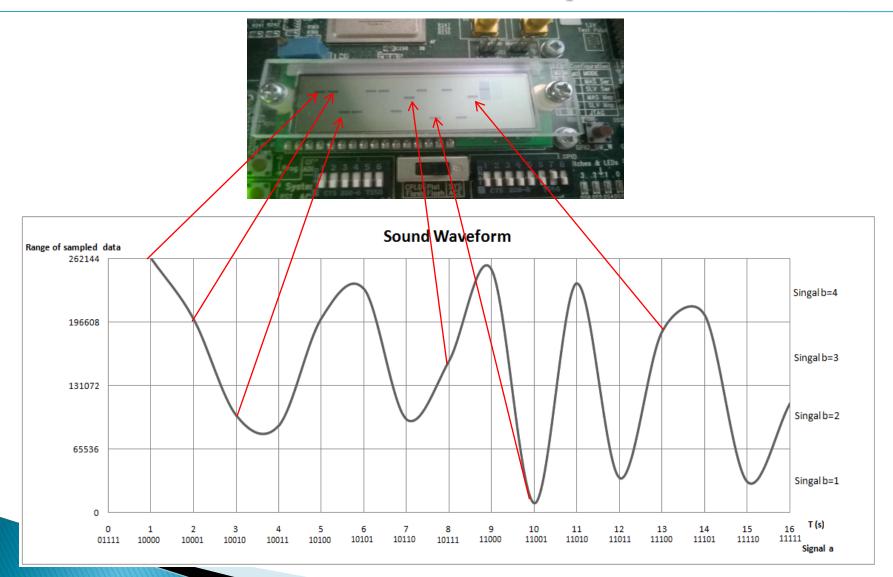
- Function set. (0010 0010 1100)
- Display ON/OFF. (0000 1111)
- Entry mode. (0000 0000)

Display mode:

- LCD works in 4-bit mode.
- The whole content of LCD cannot be shifted and LCD displays 2-line with a flashing cursor on the screen.



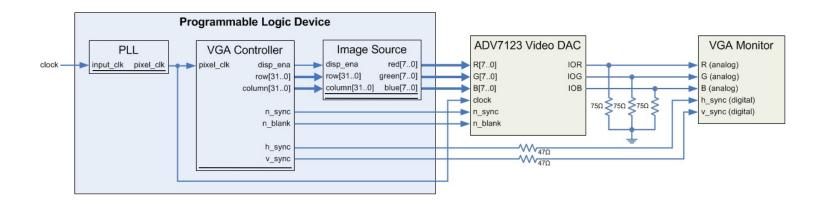
LCD Module – Data Representation



VGA Module – Block Diagram

VGA module has four main parts:

- PLL (DCM) part.
- VGA Controller part.
- Image Generator part.
 - Switch is 000.
 - Switch is 001.
 - Switch is 010.
 - Switch is 011.



VGA Module - DCM Part

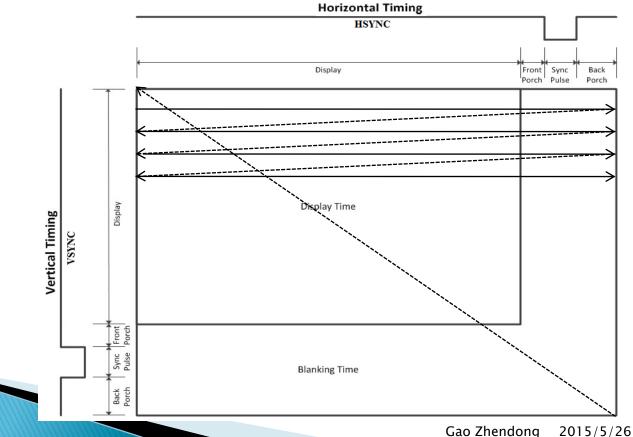
- Change the system clock to the desired pixel clock
- frequency depends on the monitor resolution and fps rate.
- The resolution is 1440x900 60 Hz, so the pixel clock is 106.47MHz.
- Core Generator define the DCM module as an IP using Xilinx's Core Generator.

dcm1 cLKIN_IN CLKFX_OUT CLKIN_IBUFG_OU RST_IN CLK0_OUT ac97 dcm		iency: 100 MHz) values		
	M	D	Output Freq (MHz)	Period Jitter (unit interval)	Period Jitter (pk-to-pk ns)
	16	15	106.667	0.030	0.278

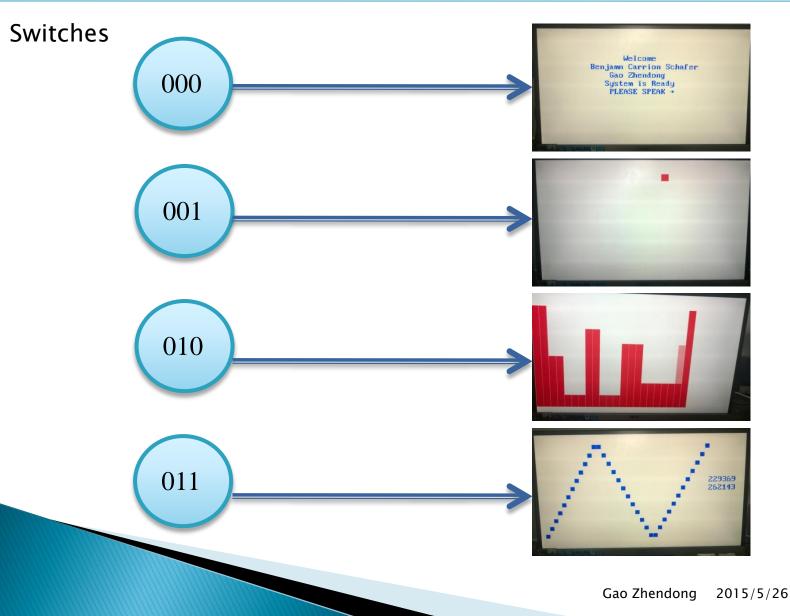
VGA Module – Controller

- Define the monitor resolution, fps rate, HSYNC and VSYNC signal.
- According to resolution, find the Front Porch, Sync Pulse and Back Porch.

Resolution (pixels)	Refresh Rate (Hz)	Pixel Clock (MHz)		Horizontal	(pixel clocks)		Vertical (rows)					
			Display	Front Porch	Sync Pulse	Back Porch	Display	Front Porch	Sync Pulse	Back Porch		
1440x900	60	106.47	1440	80	152	232	900	1	3	28		
1600x1200	60	162	1600	64	192	304	1200	1	3	46		

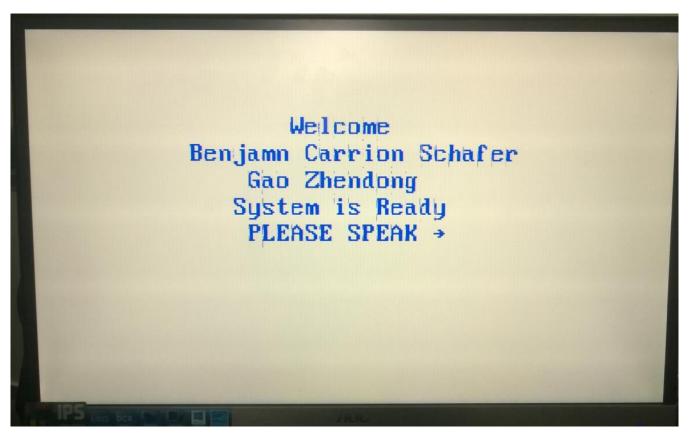


VGA Module - Imager Generator



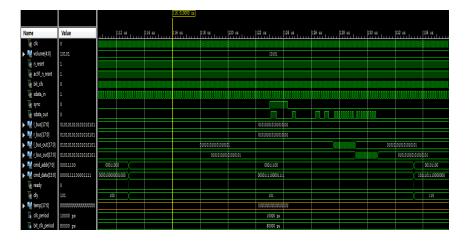
ROM

- Used to store monitor display messages.
- Send the stored display to imager generator part.



Verification and Testing

Simulation based



Prototyping



Simulation of AC97

				i	116.010000 us									
			li ce	144.1		la e	lan.	l. an	lini.	kee.	kan	li ee	kee li	
Name	Value		112 us	114 us 1	116 us	118 us	120 us	122 us	124 us	126 us	128 us	130 us	132 us	134 us
Ug clk	0													
volume[4:0]	10101							10101						
10 n_reset	1											_		
ac97_n_reset	1													
U bit_clk	0													
le sdata_in	1					HUUUUUUUUU								
le sync	0									Slot 2	S ot 3 Slot	4		
🕼 sdata_out	0								Slot 1					
I_bus[17:0]	01010101010101010101							010101010101010						
• 💑 r_bus[17:0]	01010101010101010101							010101010101010	101	/				
I_bus_out[17:0]	01010101010101010101					0101010101010101			//	X			10101010101010101	
r_bus_out[17:0]	01010101010101010101					01010101	0101010101		<u> </u>				010101010101010	
🛚 🔣 cmd_addr[7:0]	00011100	000110						00011100						00101100
• 🔣 cmd_data[15:0]	0000111100001111	00001000000	01000					000011110000111	/				^	101110000
Un ready	0						20 us							
🕼 dly	101	100						101						110
• 🔣 temp[17:0]	000000000000000000000000000000000000000							บบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบ	טטט					
🔓 clk_period	10000 ps							10000 ps						
🔓 bit_clk_period	80000 ps							80000 ps						
• 🔣 l_bus[17:0]	01010101010	1010101								01010101	0101010101			
• 式 r_bus[17:0]	01010101010	1010101								01010101	0101010101			
🛚 式 l_bus_out[1	7:0] 01010101010	1010101	0101010	10101010101							0101	01010101 <mark>0101</mark>	01	
🛛 🔣 r_bus_out[1		1010101		10101010101010	0101							0101	01010101010101	
🛛 🔣 cmd_addr[7					00101	1100		X					000110	10
• Mathematical Control Cont		00000			101110111	1000000							000000000	000000
Te ready	o													
ါ်စ္ dly	o				11	0							0	
temp[17:0]	01010101010	1010101			ມບບບບບບບ	บบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบบ							01010101010	1010101

Simulation of LCD

Name	Value	2	,000 us	2	2,500 us	3,000 us	3, 500 u	5	4,000	U5	4,500 us	5	,000 us	5, 500 us		6, 0	00 us	6, 500	15	7,000	U5	7,500 us	8,000 u
1 clk	1																						
🔓 reset	1																						
🕨 🕌 [cd[3:0]	1101		000	111	1	0000				1100	0000 🔶		0010)	1101			100		0001		0010	1101
🄓 lcd_e	0																						
la lcd_rs	1																						
la lcd_rw	0																						
🕨 🕌 temp[17:0]	01010101010101010101	IJ									010	10	1010101010101										
🕨 🕌 a[4:0]	10000				01111						100)00								100			
]] c_state	s1				s0			1		s2	X		53			1		52				53	X
ll _e b	10				0											10							
🌡 clk_period	10000 ps											100	00 ps										
🌡 bit_clk_period	80000 ps											800	00 ps										

Simulation of Switch (000)

Name	Value	5, 983, 200 ns	5, 983, 300	15	5, 983, 400 ns	5, 983, 500 ns	5, 983, 600 ns	5, 983, 700 ns	5, 983, 800 ns	5, 983, 900 ns	5, 984, 000 ns	5,984,100 ns	5, 9
1) ₆ clk	0			Ш									Ī
🄓 pixel_clk	0												
🔓 row	101001111						1010	01111					
🄓 column	101100000	xXXXXXXXXXXXXX										XXXXXXXXXXX	XX
🔓 disp_ena	1												
🕨 📑 red[7:0]	11111111		1111111			x0000000X	11111111	(000····X		11111111		0000000	1
🕨 📲 green[7:0]	11111111		1111111			00000000	11111111	(000)		11111111		00000000	1
🕨 🕌 blue[7:0]	11111111						111	11111					
l_{1} font_on	0												
🕨 📑 stage[2:0]	000						(00					

Simulation of Switch (001)

Name	Value		4 ms		5 m5	8 ms	10 ms	12 ms	14 ms	16 ms	18 ms 20
lie cik 1	L										
1 pixel_clk	L										
	01010101									(1)	
•	0110011111										
ါ <mark>ြ</mark> disp_ena ၀)										
▶ 📑 stage[2:0]	001						001				
▶ 🕌 red[7:0]	1111111									0	
▶ 🚮 green[7:0] 11	1111111									0)	
▶ 🚮 blue[7:0] 11	1111111									0	
	01010101010101010101						0101010101010	10101			
	1010001010						1010001010	D			
	001011000						100101100	0			
196 · · ·	10011	110010				110011				110100	
Vo pixels_x1 1	l	0	X			1			X	10	

Simulation of Switch (010)

pixels_x1_1	0				0			
pixels_x1_3	1100100				1100100			
🔓 pixels_x1_5	11001000				11001000			
🔓 pixels_x1_7	100101100				100101100			
🔓 pixels_x1_8	101011110				101011110			
🔓 pixels_x1_10	111000010				111000010			
pixels_x1_12	1000100110				1000100110			
🔓 pixels_x1_15	1010111100				1010111100			
pixels_x1_17	1100100000				1100100000			
🔓 pixels_x1_19	1110000100				1110000100			
pixels_x1_22	10000011010				10000011010			
pixels_x2_1	110010				110010			
🔓 pixels_x2_3	10010110				10010110			
🔓 pixels_x2_5	11111010				11111010			
pixels_x2_7	101011110				101011110			
pixels_x2_8	110010000				110010000			
🔓 pixels_x2_10	111110100				111110100			
pixels_x2_12	1001011000				1001011000			
🔓 pixels_x2_15	1011101110				1011101110			
pixels_x2_17	1101010010				1101010010			
🔓 pixels_x2_19	1110110110				1110110110			
pixels_x2_22	10001001100				10001001100			
Name	Value	0 us	500 us	 1,000 us	1,500 us	2,000 us	2,500 us	3,000 us
🔓 pixel_clk	0							
1 row	0							
🔓 column	100100							
🕨 📑 stage[2:0]	010					010	\supset	
▶ 📑 red[7:0]	11111111							
🕨 📑 green[7:0]	11111111							
▶ 📑 blue[7:0]	11111111							
🕨 📑 temp[17:0]	000000000000000000000000000000000000000	0000				0101010	10101010101	
16 а	1					1		
D pixels_y1_2	100000000000000000000000000000000000000	(1000)					0	
pixels_y2_2	100000000000000000000000000000000000000	(1000…)				101	0001010	

Simulation of Switch (011)

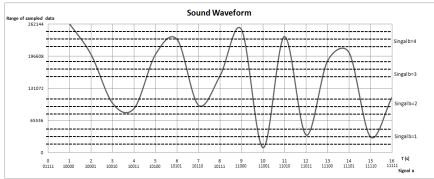
Name	Value		1, 362, 600 ns	1, 362, 650 ns	1, 362, 700 ns	1, 362, 750	15	1, 362, 800 ns	1, 362, 850 ns	1, 362, 900 ns	1, 362, 950 ns	1, 363, 000 ns	1, 363, 050 ns	1, 363, 100 r
🗓 pixel_clk	0	Γ					\Box							
🔓 row	1001100								10011	00				
🗓 column	1001110010	X					X:::X							
🕨 📲 stage[2:0]	011								011					
🕨 🕌 red[7:0]	0000000		1111111			000000	0			X111***X			0000000	
🕨 🕌 green[7:0]	0000000		1111111			000000	0			X111***X			0000000	
🕨 🕌 blue[7:0]	11111111								11111	.11				
🔓 font_on	1													
🕨 <table-of-contents> temp[17:0]</table-of-contents>	01010101010101010101								01010101010	01010101				

Prototyping (Video)



Summary and Conclusions

- FPGA based speech visualization
 - Top-down approach
 - High flexibility and reliability
- Visualization on:
 - LCD
 - VGA
- Learnt:
 - Mastered VHDL and Xilinx ISE
 - Interface of FPGA with different peripherals
 - Audio Codec
 - LCD
 - VGA
- Future work:
 - Divide more sampled data's range
 - Use RAM to store sampled data
 - Design an equalizer or oscilloscope





- I would like to extend my sincere gratitude to my supervisor, Dr. Benjamin Carrion Schafer, for his patience, instructive advice and useful suggestions on my thesis. I am deeply grateful of his help in the completion of this thesis.
- I am also deeply indebted to all the other students, tutors and teachers in Hong Kong Polytechnic University for their direct and indirect help to me.
- Thank you for listening to my presentation.

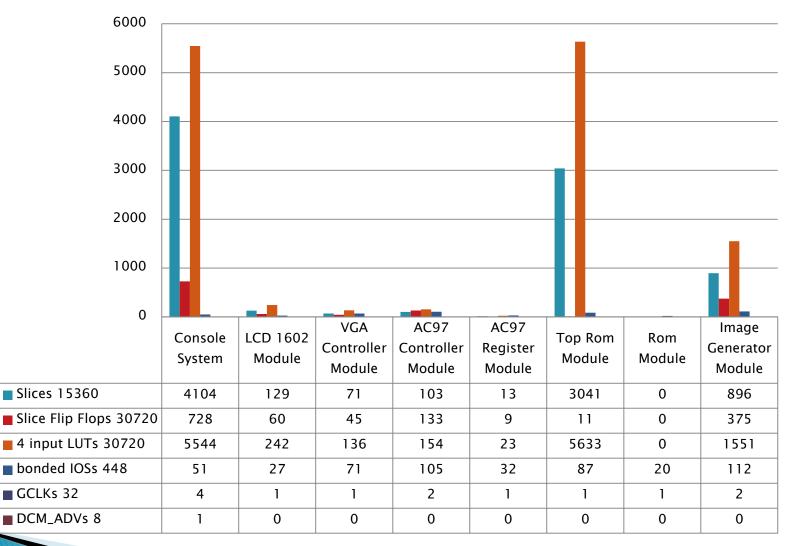
Thank you!



Synthesis Report of Console System

Logic Utilization	Used	Available	Utilization
Number of Slices	4104	15360	26%
Number of Slice Flip Flops	728	30720	2%
Number of 4 input LUTs	5544	30720	18%
Number of IOs	51		
Number of bonded IOBs	51	448	11%
Number of FIFO16/RAMB16s	1	192	0%
Number used as RAMB16s	1		
Number of GCLKs	4	32	12%
Number of DCM_ADVs	1	8	12%

Diagram of the Device Utilization



LCD Module Synthesis Report

Synthesis Report of LCD 1602

Logic Utilization	Used	Available	Utilization
Number of Slices	129	15360	0%
Number of Slice Flip Flops	60	30720	0%
Number of 4 input LUTs	242	30720	0%
Number of IOs	27		
Number of bonded IOBs	27	448	6%
Number of GCLKs	1	32	3%



ROM

Synthesis Report of TOP ROM

Logic Utilization	Used	Available	Utilization
Number of Slices	3041	15360	19%
Number of Slice Flip Flops	11	30720	0%
Number of 4 input LUTs	5633	30720	18%
Number of IOs	87		
Number of bonded IOBs	87	448	19%
IOB Flip Flops	1		
Number of FIFO16/RAMB16s	1	192	0%
Number used as RAMB16s	1		
Number of GCLKs	1	32	3%

VGA Module Synthesis report

Synthesis Report of VGA Controller

Logic Utilization	Used	Available	Utilization
Number of Slices	71	15360	0%
Number of Slice Flip Flops	45	30720	0%
Number of 4 input LUTs	136	30720	0%
Number of IOs	71		
Number of bonded IOBs	71	448	15%
Number of GCLKs	1	32	3%

Synthesis Report of Image Generator

Logic Utilization	Used	Available	Utilization
Number of Slices	896	15360	5%
Number of Slice Flip Flops	375	30720	1%
Number of 4 input LUTs	1551	30720	2%
Number of IOs	112		
Number of bonded IOBs	112	448	25%
Number of GCLKs	1	32	6%



AC'97 Audio Codec Module Synthesis Report

Synthesis Report of AC97 Controller

Logic Utilization	Used	Available	Utilization
Number of Slices	103	15360	0%
Number of Slice Flip Flops	133	30720	0%
Number of 4 input LUTs	154	30720	0%
Number of IOs	105		
Number of bonded IOBs	105	448	23%
Number of GCLKs	2	32	6%

Synthesis Report of AC97 Register

Logic Utilization	Used	Available	Utilization
Number of Slices	13	15360	0%
Number of Slice Flip Flops	9	30720	0%
Number of 4 input LUTs	21	30720	0%
Number of IOs	32		
Number of bonded IOBs	32	448	7%
Number of GCLKs	1	32	3%