SDSU | San Diego State University

“Voice Interactive 7-Segment Display System”
Final Project Report

Project Images

GitHub Link

Abdul Karim Tamim

COMPE-475 - Microprocessors
Prof. Ken Arnold
Dec 02, 2024

Completed Tasks Overview For This Project

- ☑ Essential Parts and Components (FPGA, Microprocessor, Parallel Volatile SRAM, Parallel Non-Volatile Flash, Speech Recognition Board Module***)
- ☑ Locate the address, data, and control pins of your parts
- ☑ Draw block diagrams (high-level design, FPGA Verilog Modules, top-level FPGA)
- ☑ Noise Margin
    - ☑ Parallel Interface Volatile Memory SRAM with Basys 3 Artix 7 FPGA (FPGA ⇄ SRAM)
    - ☑ Parallel Interface Non-volatile Memory Flash with Basys 3 Artix 7 FPGA (FPGA ⇄ Flash)
    - ☑ Parallel Interface Volatile Memory SRAM with the Processor (Processor ⇄ SRAM)
    - ☑ Parallel Interface Non-Volatile Memory Flash with the Processor (Processor ⇄ Flash)
    - ☑ Basys 3 Artix 7 FPGA with the Processor (FPGA ⇄ Processor)
- ☑ Loading Analysis
    - ☑ Data (CPU)
    - ☑ Data (SRAM)
    - ☑ Data (Flash)
    - ☑ Data (FPGA)
    - ☑ Address (CPU)
        - ☑ SRAM
        - ☑ Flash
        - ☑ FPGA
    - ☑ CE, OE, WE (CPU)
        - ☑ SRAM
        - ☑ Flash
        - ☑ FPGA
    - ☑ WP (CPU)
        - ☑ WE (Flash)
- ☑ Timing Analysis
    - ☑ Include the Specs of Each Part/Component for calculating the Timing Analysis
    - ☑ Derating Delay Analysis and Calculations
        - ☑ Processor Derating Delay
        - ☑ SRAM Derating Delay
        - ☑ Flash Derating Delay

- ☑ Signal Delays for Complete Power-Off
  - ☑ SRAM
  - ☑ Flash
- ☑ SRAM Signal Timing Delays for Complete Power On/Off
  - ☑ SRAM
  - ☑ Flash
- ☑ Timing Analysis Between Connected Signals
  - ☑ Signal Paths Calculations
  - ☑ Avoid Bus contention from one memory to another
    - ☑ Ensure no overlap occurs
  - ☑ Determine Longest Path In The System
  - ☑ Determine the Last Signal to Go Active

☑ Memory Address Mapping Tables
- ☑ Program Address Map Table
- ☑ Data Memory Address Table

☑ Logic Implementation of SpeakUp Click Speech Recognition Module on the Basys-3 Artix-7 FPGA

☑ Verilog Code
- ☑ Verilog Top Module, including:
  - ☑ Verilog Module to Generate a Sine Signal Using Fake Data
  - ☑ Verilog Noise Generator LFSR Module
  - ☑ Verilog Program Address Map Module
  - ☑ Verilog Data Memory Address Module
  - ☑ Verilog Digit to the 7-Segment Display Module
  - ☑ Verilog UART Receiver Module
  - ☑ Verilog Filter Signal Module
    - ☑ Generate a Square Wave
    - ☑ Filter the Square Wave to a Rounded Signal
- ☑ Verilog Simulation Testbench Modules, including:
  - ☑ Verilog Program Address Map Testbench Module
  - ☑ Verilog Data Memory Address Testbench Module
  - ☑ Verilog Digit to the 7-Segment Display Testbench Module
  - ☑ Verilog UART Receiver Testbench Module
  - ☑ Verilog Filter Signal Testbench Module
- ☑ Verilog Constraints File
- ☑ Verilog Memory Files

Table of Content

1 Project Description

Field-programmable gate arrays (FPGAs) are integrated circuits that can be reconfigured to meet designers' needs. FPGAs contain an array of programmable logic blocks, and chip adoption is driven by their flexibility, hardware-timed speed and reliability, and parallelism. This project is a voice recognition system that processes spoken numbers between 0 and 9999 and displays the recognized number on a 4-digit 7-segment display. The system will utilize the Basys-3 Artix-7 FPGA to handle signal processing and interface with a voice recognition module. The design will include an external microprocessor and memories of volatile (SRAM) and non-volatile (Flash) components to handle real-time data and store previous inputs.

The project will utilize Verilog, a hardware description language, which will be used for designing, and implementing the logical modules for the project. Verilog also allows for simulation, timing, and test analysis verification. While several software applications support Verilog development and implementation, this project will primarily utilize Vivado for Verilog code writing and simulation. Xilinx's Vivado offers a comprehensive integrated development environment (IDE) that is specifically designed for FPGA design and verification. due to its numerous capabilities, which include advanced synthesis and analysis tools, it is an ideal choice for implementing complicated digital systems like the voice recognition project.

This report walks through the calculations of the design verification analysis obtained from the utilized components of this project. These analyses include noise margin analysis, which is a measure of design margins to ensure circuits function properly within specified conditions. The noise margin is divided into two-part calculations that require many I/O electrical characteristics values, the first part is the Noise Margin High (NMH), also known as Logic One Case, which is the range of tolerance for which you can still correctly receive a logical high signal. The second part is the Noise Margin Low (NML), also known as the Logic Zero Case, which stipulates the range of tolerance for a logical low signal on the wire.

Loading analysis is another important design verification analysis of this project, which is an evaluation of how many circuit elements a given output can drive without degrading the signal. Loading analysis requires many I/O electrical characteristic values, just like the noise margin, that can be used to calculate the total input currents and capacitance, as well as the margin. Mainly for the loading analysis, the processor or CPU derives the other components, such as the external memories and FPGA. Source signals like the data bus signals, address bus signals, and control bus signals are derived from the CPU and sent to the other components.

Timing Analysis is another important design verification analysis of this project, which is utilized to ensure that a circuit meets all timing specifications and will operate correctly at speed. To ensure this the longest propagation delay of the circuit must be tested. The timing analysis is entirely important for the SRAM and Flash memories, as it helps analyze the time it takes to access data, including the delay from when an address is sent to the memory device to when the valid data is available at the output.

2 Initial Part Numbers and Links to Data Sheets

A. Basys-3 Artix-7 FPGA

- Basys 3 Artix-7 FPGA Trainer Board
- Artix 7 FPGAs Data Sheet: DC and AC Switching Characteristics
- XC7A35T-1CPG236C
- 1286-1041-ND
- Schematic
- Basys 3™ FPGA Board Reference Manual

B. Microprocessor

- AM3351BZCE30
- Data Sheet
- Schematic
- Reference Manual

C. PARALLEL Interface Volatile Memory (SRAM)

- AS7C38098A-10BIN
- Data Sheet

D. PARALLEL Interface Non-volatile Memory (Flash)

- S29AL008J 55 FOR 10
- Data Sheet

E. Speech Recognition Board Module ***

- SpeakUp Click by MikroElektronika
- User Manual

3 Information and Pin Configuration/Description

3.1 Basys 3 Artix 7 FPGA

Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod port(s)	10	Programming mode jumper
3	Analog signal Pmod port (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

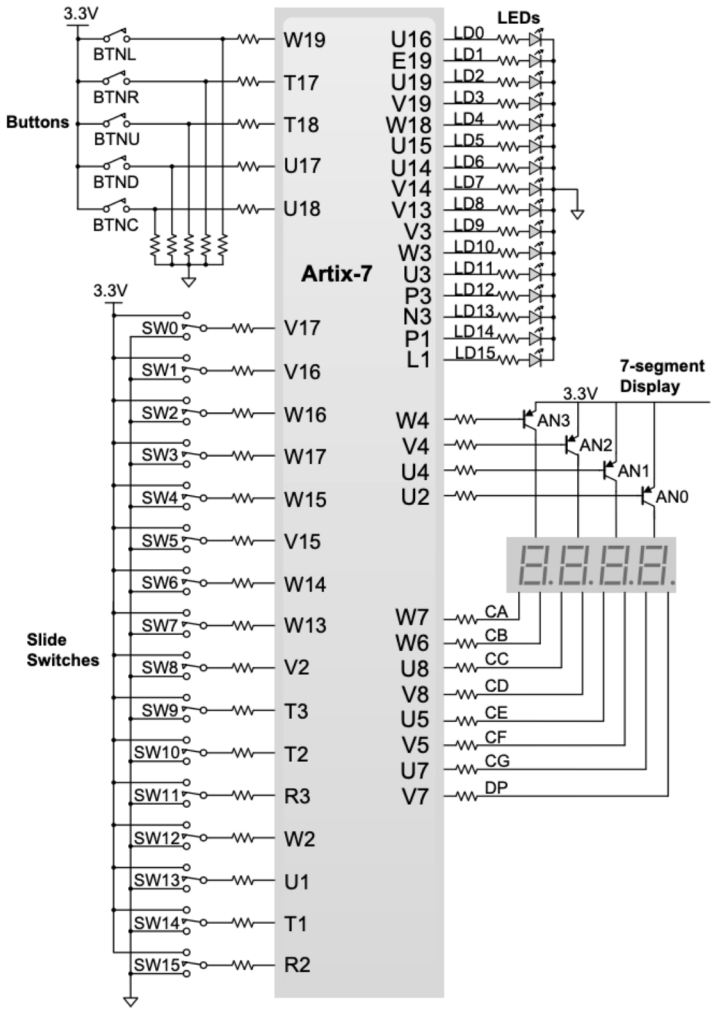


Figure 16. General purpose I/O devices on the Basys 3.

3.2 Processor

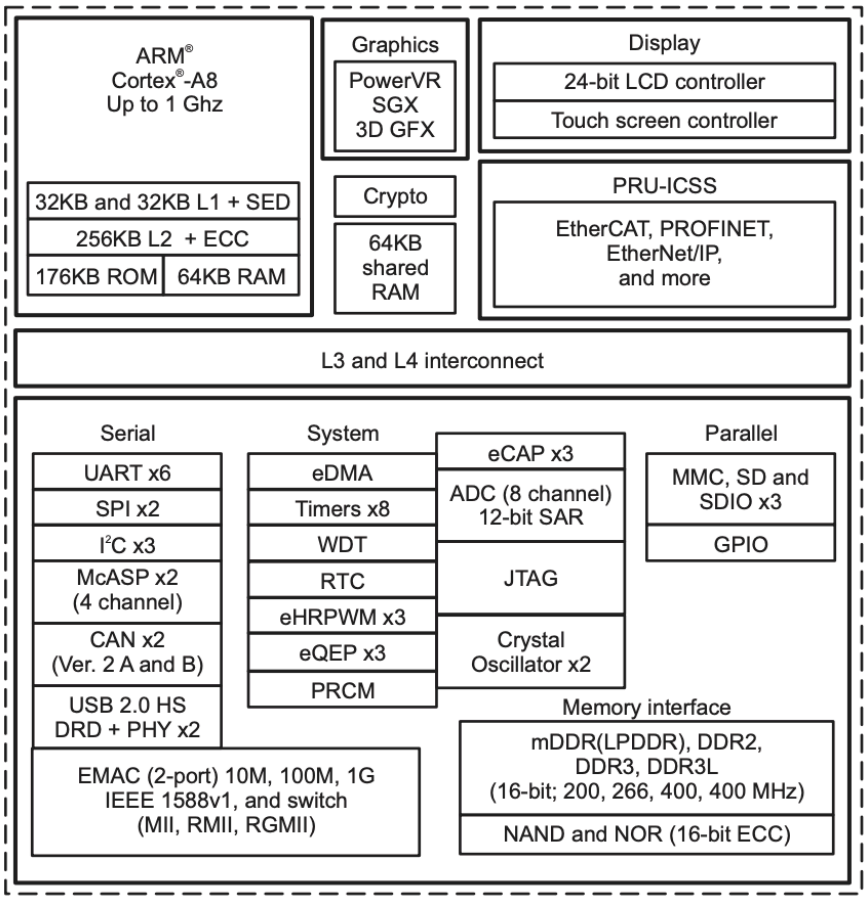


Figure 1-1. AM335x Functional Block Diagram

7.1.4.1.2 Typical GPMC Setup

Table 7-39 lists some of the I/Os of the GPMC module.

Table 7-39. GPMC Signals

Signal Name	I/O	Description
GPMC_FCLK	Internal	Functional and interface clock. Acts as the time reference.
GPMC_CLK	O	External clock provided to the external device for synchronous operations
GPMC_A[27:17]	O	Address
GPMC_AD[15: 0]	I/O	Data-multiplexed with addresses A[16:1] on memory side
GPMC_CSxn	O	Chip-select (where x = 0, or 1)
GPMC_ADVn_ALE	O	Address valid enable
GPMC_OE_REn	O	Output enable (read access only)
GPMC_WEn	O	Write enable (write access only)
GPMC_WAIT[1:0]	I	Ready signal from memory device. Indicates when valid burst data is ready to be read

Figure 7-201. DDR2/3/mDDR Memory Controller Signals

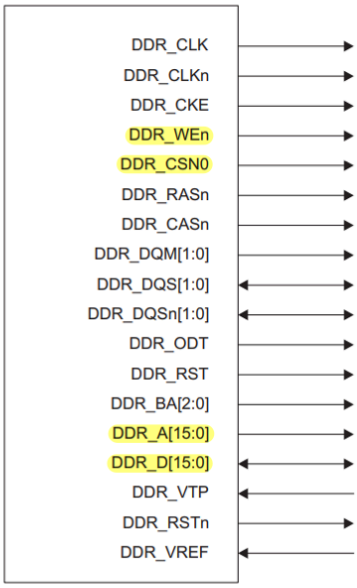
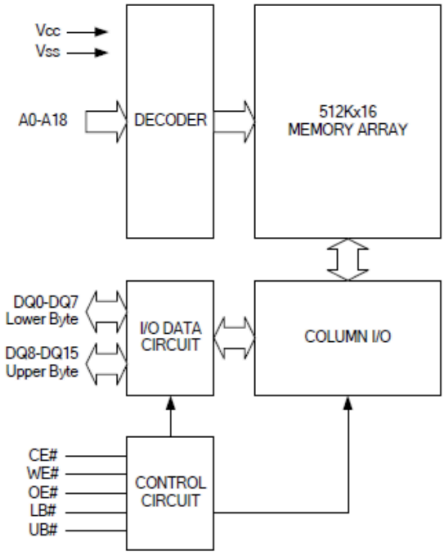


Table 7-208. DDR2/3/mDDR Memory Controller Signal Descriptions

Pin	Description
DDR_D[15:0]	Bidirectional data bus. Input for data reads and output for data writes.
DDR_A[15:0]	External address output.
DDR_CSN0	Chip select output.
DDR_DQM[1:0]	Active-low output data mask.
DDR_CLK/DDR_CLKn	Differential clock outputs. All DDR2/3/mDDR interface signals are synchronous to these clocks.
DDR_CKE	Clock enable. Used to select Power-Down and Self-Refresh operations.
DDR_CASn	Active-low column address strobe.
DDR_RASn	Active-low row address strobe.
DDR_WEn	Active-low write enable.

3.3 Parallel Volatile SRAM

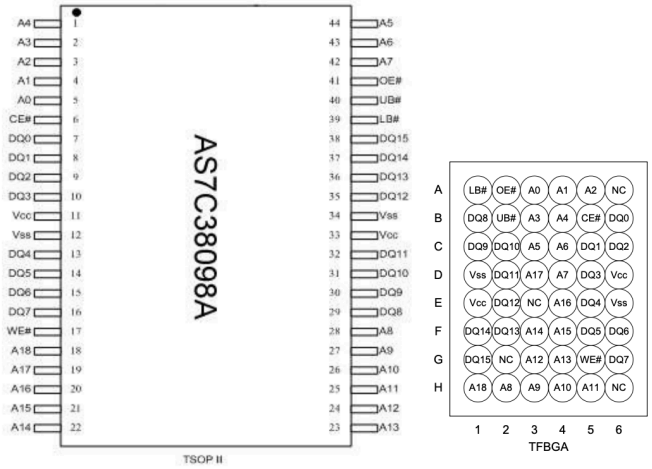
FUNCTIONAL BLOCK DIAGRAM



PIN DESCRIPTION

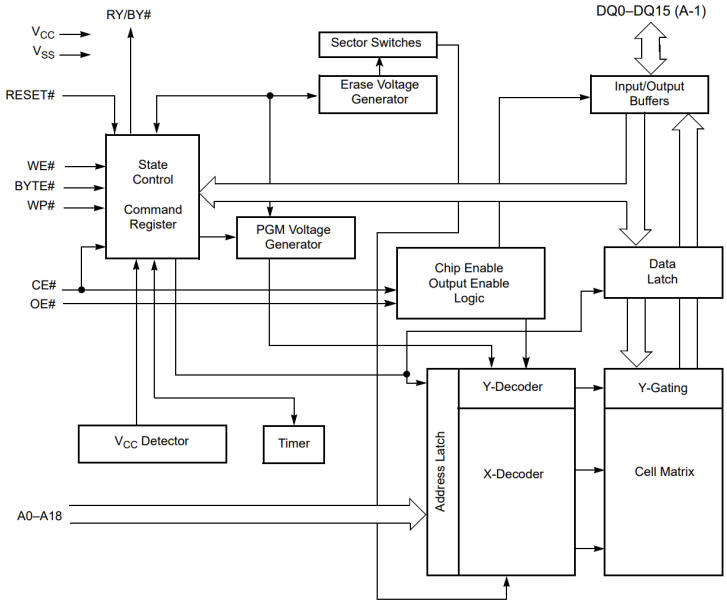
SYMBOL	DESCRIPTION
A0 - A18	Address Inputs
DQ0 – DQ15	Data Inputs/Outputs
CE#	Chip Enable Input
WE#	Write Enable Input
OE#	Output Enable Input
LB#	Lower Byte Control
UB#	Upper Byte Control
V_{CC}	Power Supply
V_{SS}	Ground

PIN CONFIGURATION



3.4 Parallel Non-Volatile Flash

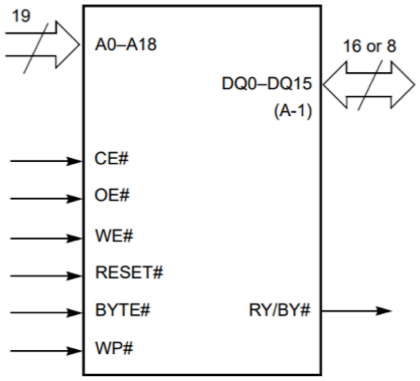
Block Diagram



Pin Configuration

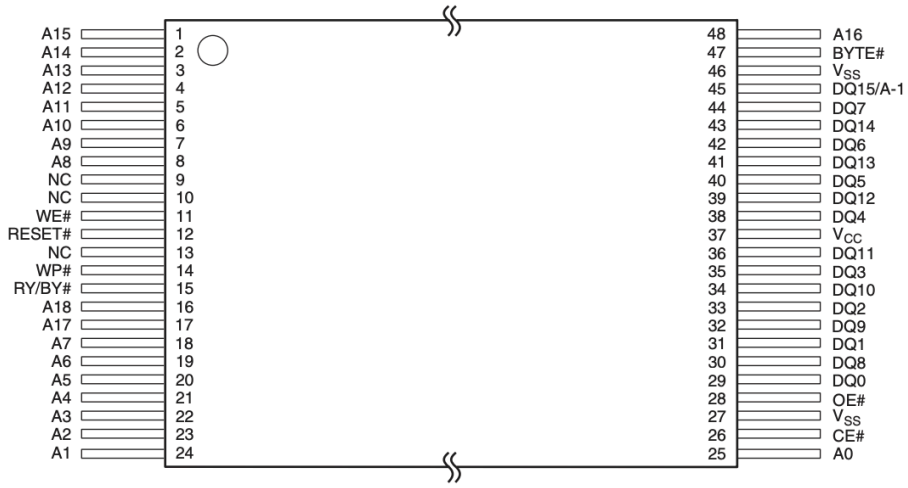
A0–A18	19 addresses
DQ0–DQ14	15 data inputs/outputs
DQ15/A-1	DQ15 (data input/output, word mode), A-1 (LSB address input, byte mode)
BYTE#	Selects 8-bit or 16-bit mode
CE#	Chip enable
OE#	Output enable
WE#	Write enable
WP#	Write protect: The WP# contains an internal pull-up; when unconnected, WP is at V_{IH}.
RESET#	Hardware reset
RY/BY#	Ready/Busy output
V_{CC}	3.0 volt-only single power supply (see Product Selector Guide on page 4 for speed options and voltage supply tolerances)
V_{SS}	Device ground
NC	Pin not connected internally

Logic Symbol



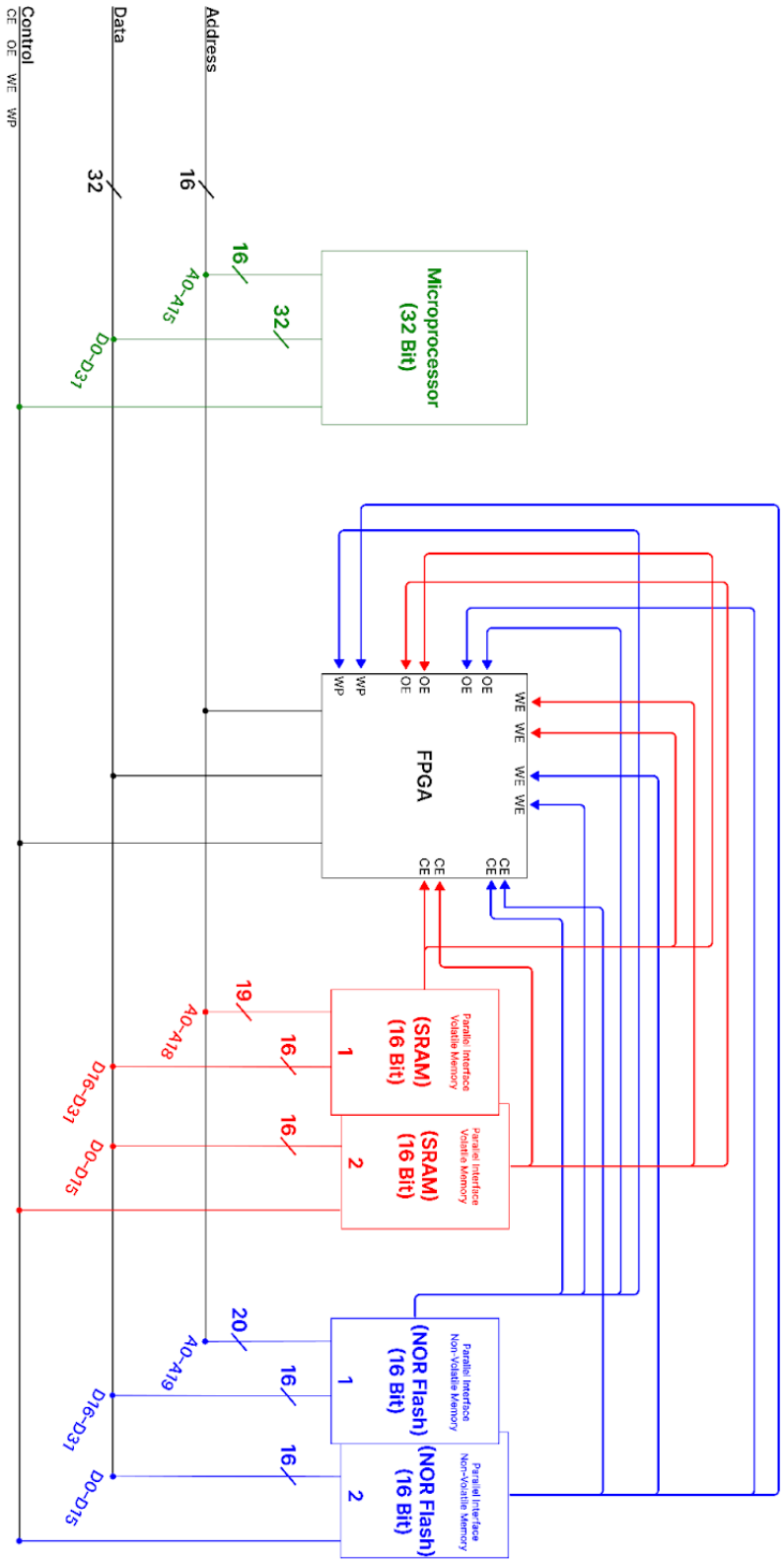
Connection Diagrams

Figure 1. 48-pin Standard TSOP (TS048)

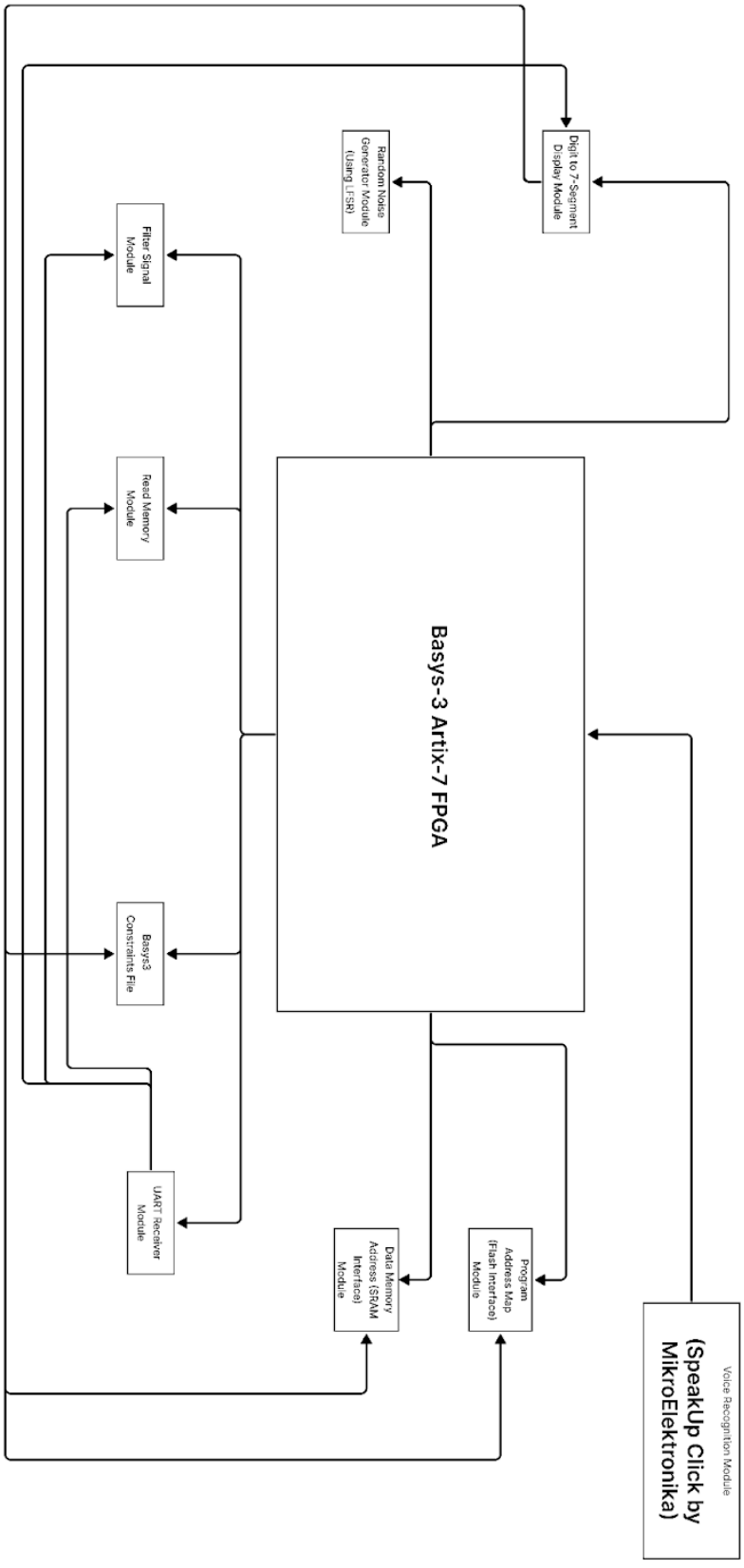


4 Block Diagrams

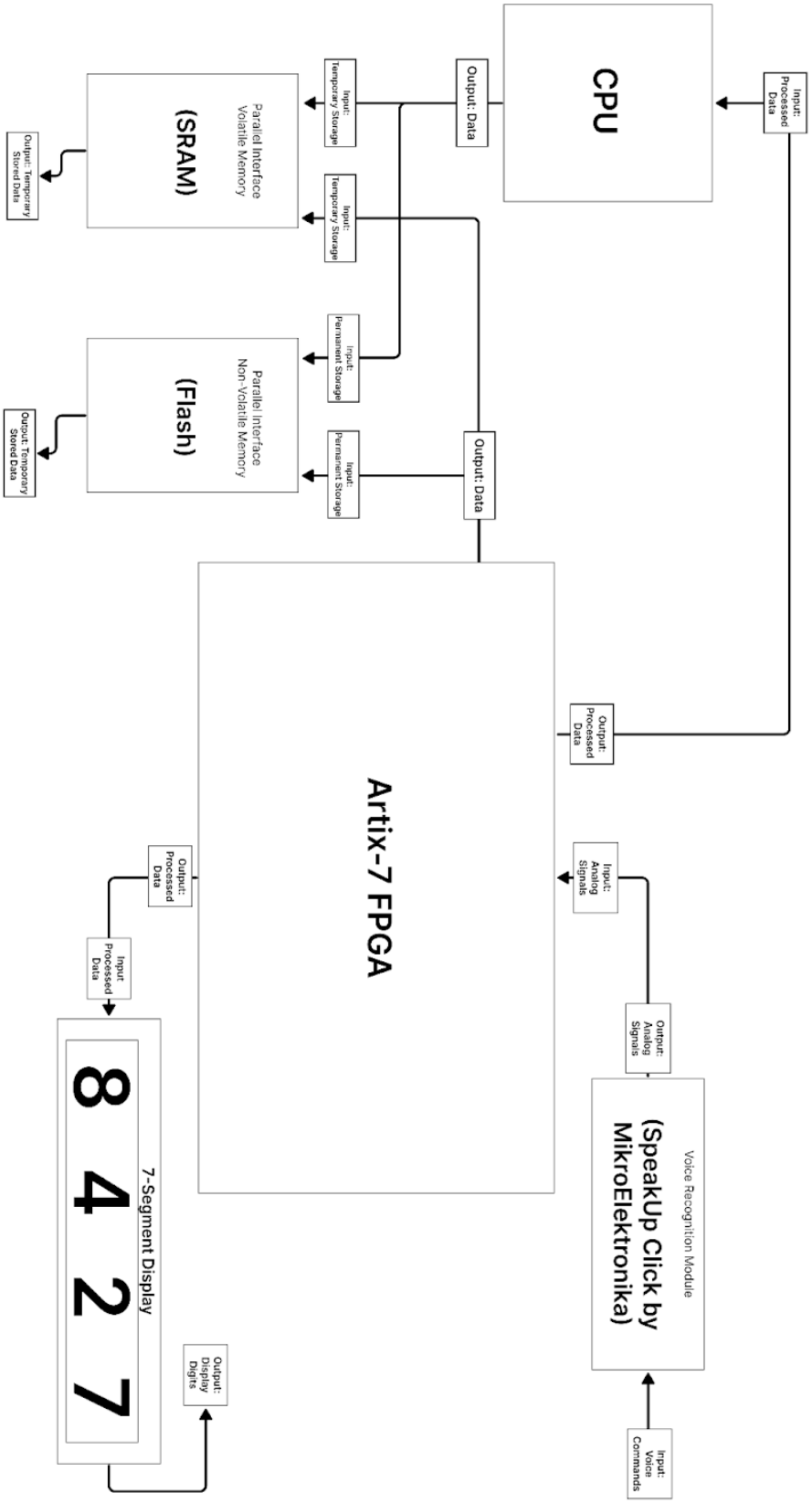
4.1 Block Diagram 1: High-level Design



4.2 Block Diagram 2: FPGA Verilog Modules



4.3 Block Diagram 3: Top-level FPGA



5 Noise Margin

The Noise Margin is a measure of design margins to ensure circuits function properly within specified conditions. To calculate the Noise Margin of the memory ICs, the SRAM, Flash, and the processor, we review their datasheets and look for the values of Vil max (Input Low Voltage), Vol max (Output Low Voltage), Voh min (Output High Voltage), and Vih min (Input High Voltage), which can typically be found in the "DC Electrical Characteristics" section. This section contains all the necessary information for calculating the noise margin for the logic zero and the logic one cases, using their corresponding equations. Below is a walkthrough of the calculations of the noise margin between the FPGA, SRAM, Flash, and the processor.

5.1 Parallel Interface Volatile Memory SRAM with Basys 3 Artix 7 FPGA

- Initial Values

Source	**Output**		**Input**		**Unit**
	Vol max	**Voh min**	**Vil max**	**Vih min**	
SRAM	0.4	2.4	0.8	2.2	V
FPGA	0.4	2.9	0.8	2.0	V

- Calculated Output Summary

FPGA ⇄ SRAM				
- *From FPGA to SRAM*				
Input	**Output**	**Noise Margin**		**Unit**
		Logic Zero	Logic One	
FPGA	SRAM	**0.4**	**0.7**	**V**
- *From SRAM to FPGA*				
Input	**Output**	**Noise Margin**		**Unit**
		Logic Zero	Logic One	
SRAM	FPGA	**0.4**	**0.4**	**V**

5.1.1 From FPGA to SRAM

Calculations:

DC ELECTRICAL CHARACTERISTICS

PARAMETER	SYMBOL	TEST CONDITION		MIN.	TYP. *4	MAX.	UNIT
Supply Voltage	V_{CC}		-10	2.7	3.3	3.6	V
Input High Voltage	V_{IH} *1			2.2	-	V_{CC}+0.3	V
Input Low Voltage	V_{IL} *2			- 0.3	-	0.8	V
Input Leakage Current	I_{LI}	$V_{CC} \geqq V_{IN} \geqq V_{SS}$		- 1	-	1	µA
Output Leakage Current	I_{LO}	$V_{CC} \geqq V_{OUT} \geqq V_{SS}$, Output Disabled		- 1	-	1	µA
Output High Voltage	V_{OH}	I_{OH} = -8mA		2.4	-	-	V
Output Low Voltage	V_{OL}	I_{OL} =4mA		-	-	0.4	V
Average Operating Power supply Current	Icc	CE# = V_{IL} , $I_{I/O}$ = 0mA ;f=max	-10	-	100	130	mA
	Icc_1	CE# ≧V_{CC} - 0.2V, Other pin is at 0.2V or Vcc-0. $I_{I/O}$ = 0mA;f=max	-10		80	110	mA
Standby Power Supply Current	Isb	CE# ≧Vih Other pin is at Vil or Vih				40	mA
Standby Power Supply Current	I_{SB1}	CE# ≧V_{CC} - 0.2V; Other pin is at 0.2V or Vcc-0.2V			3	25	mA

"SRAM Initial Values"

Table 8: **SelectIO DC Input and Output Levels**(1)(2)

I/O Standard	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V, Min	V, Max	V, Min	V, Max	V, Max	V, Min	mA, Max	mA, Min
HSTL_I	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_I_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_II	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSTL_II_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSUL_12	–0.300	V_{REF} – 0.130	V_{REF} + 0.130	V_{CCO} + 0.300	20% V_{CCO}	80% V_{CCO}	0.10	–0.10
LVCMOS12	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 3	Note 3
LVCMOS15	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	25% V_{CCO}	75% V_{CCO}	Note 4	Note 4
LVCMOS18	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.450	V_{CCO} – 0.450	Note 5	Note 5
LVCMOS25	–0.300	0.7	1.700	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVCMOS33	–0.300	0.8	2.000	3.450	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVTTL	–0.300	0.8	2.000	3.450	0.400	2.400	Note 5	Note 5
MOBILE_DDR	–0.300	20% V_{CCO}	80% V_{CCO}	V_{CCO} + 0.300	10% V_{CCO}	90% V_{CCO}	0.10	–0.10
PCI33_3	–0.400	30% V_{CCO}	50% V_{CCO}	V_{CCO} + 0.500	10% V_{CCO}	90% V_{CCO}	1.50	–0.50
SSTL135	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	13.00	–13.00
SSTL135_R	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	8.90	–8.90
SSTL15	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	V_{CCO}/2 – 0.175	V_{CCO}/2 + 0.175	13.00	–13.00

"Basys 3 Artix-7 FPGA Initial Values"

- $Noise\ Margin\ Logic\ Zero\ Case\ =\ NM_0 = V_{il\,max}\ -\ V_{ol\,max}$

Input: SRAM ($V_{il\,max}$) Output: FPGA ($V_{ol\,max}$)

$$V_{il\,max}\ =\ 0.8\,V$$

$$V_{ol\,max}\ =\ 0.4\,V$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ 0.8\ -\ 0.4$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ \underline{\mathbf{0.4\ V}}$$

- $Noise\ Margin\ Logic\ One\ Case\ =\ NM_1\ =\ V_{oh\,min}\ -\ V_{ih\,min}$

Output: FPGA ($V_{oh\,min}$) Input: SRAM ($V_{ih\,min}$)

$$V_{oh\ min}\ =\ V_{CCO}\ -\ 0.4 \qquad (\text{Where } V_{CCO}\ =\ 3.3\,V)$$

$$V_{oh\ min}\ =\ 3.3\ -\ 0.4\ =\ 2.9V$$

$$V_{ih\,min}\ =\ 2.2\,V$$

$$Logic\ One\ Case\ =\ 2.9\ -\ 2.2$$

$$Logic\ One\ Case\ =\ \underline{\mathbf{0.7\ V}}$$

- Therefore, after performing the calculations, we obtain a noise margin logic zero case of 0.4V, and a noise margin logic one case of 0.7 V from FPGA to the SRAM.

5.1.2 From SRAM to FPGA

Calculations:

DC ELECTRICAL CHARACTERISTICS

PARAMETER	SYMBOL	TEST CONDITION		MIN.	TYP. *4	MAX.	UNIT
Supply Voltage	V_{CC}		-10	2.7	3.3	3.6	V
Input High Voltage	V_{IH} *1			2.2	-	V_{CC}+0.3	V
Input Low Voltage	V_{IL} *2			- 0.3	-	0.8	V
Input Leakage Current	I_{LI}	$V_{CC} \geqq V_{IN} \geqq V_{SS}$		- 1	-	1	µA
Output Leakage Current	I_{LO}	$V_{CC} \geqq V_{OUT} \geqq V_{SS}$, Output Disabled		- 1	-	1	µA
Output High Voltage	V_{OH}	I_{OH} = -8mA		2.4	-	-	V
Output Low Voltage	V_{OL}	I_{OL} =4mA		-	-	0.4	V
Average Operating Power supply Current	Icc	CE# = V_{IL} , $I_{I/O}$ = 0mA ;f=max	-10	-	100	130	mA
	Icc_1	CE# ≧V_{CC} - 0.2V, Other pin is at 0.2V or Vcc-0. $I_{I/O}$ = 0mA;f=max	-10		80	110	mA
Standby Power Supply Current	Isb	CE# ≧Vih Other pin is at Vil or Vih				40	mA
Standby Power Supply Current	I_{SB1}	CE# ≧V_{CC} - 0.2V; Other pin is at 0.2V or Vcc-0.2V			3	25	mA

"SRAM Initial Values"

Table 8: **SelectIO DC Input and Output Levels**(1)(2)

I/O Standard	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V, Min	V, Max	V, Min	V, Max	V, Max	V, Min	mA, Max	mA, Min
HSTL_I	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_I_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_II	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSTL_II_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSUL_12	–0.300	V_{REF} – 0.130	V_{REF} + 0.130	V_{CCO} + 0.300	20% V_{CCO}	80% V_{CCO}	0.10	–0.10
LVCMOS12	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 3	Note 3
LVCMOS15	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	25% V_{CCO}	75% V_{CCO}	Note 4	Note 4
LVCMOS18	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.450	V_{CCO} – 0.450	Note 5	Note 5
LVCMOS25	–0.300	0.7	1.700	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVCMOS33	–0.300	0.8	2.000	3.450	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVTTL	–0.300	0.8	2.000	3.450	0.400	2.400	Note 5	Note 5
MOBILE_DDR	–0.300	20% V_{CCO}	80% V_{CCO}	V_{CCO} + 0.300	10% V_{CCO}	90% V_{CCO}	0.10	–0.10
PCI33_3	–0.400	30% V_{CCO}	50% V_{CCO}	V_{CCO} + 0.500	10% V_{CCO}	90% V_{CCO}	1.50	–0.50
SSTL135	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	13.00	–13.00
SSTL135_R	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	8.90	–8.90
SSTL15	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	V_{CCO}/2 – 0.175	V_{CCO}/2 + 0.175	13.00	–13.00

"Basys 3 Artix-7 FPGA Initial Values"

- $Noise\ Margin\ Logic\ Zero\ Case\ =\ NM_0 = V_{il\,max}\ -\ V_{ol\,max}$

Input: FPGA ($V_{il\,max}$) Output: SRAM ($V_{ol\,max}$)

$$V_{il\,max}\ =\ 0.8\,V$$

$$V_{ol\,max}\ =\ 0.4\,V$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ 0.8\ -\ 0.4$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ \underline{\mathbf{0.4\ V}}$$

- $Noise\ Margin\ Logic\ One\ Case\ =\ NM_1\ =\ V_{oh\,min}\ -\ V_{ih\,min}$

Output: SRAM ($V_{oh\,min}$) Input: FPGA ($V_{ih\,min}$)

$$V_{oh\ min}\ =\ 2.4V$$

$$V_{ih\,min}\ =\ 2.0\,V$$

$$Logic\ One\ Case\ =\ 2.4\ -\ 2.0$$

$$Logic\ One\ Case\ =\ \underline{\mathbf{0.4\ V}}$$

- Therefore, after performing the calculations, we obtain a noise margin logic zero case of 0.4V, and a noise margin logic one case of 0.4 V from SRAM to the FPGA.

5.2 Parallel Interface Non-volatile Memory Flash with Basys 3 Artix 7 FPGA

- Initial Values

Source	**Output**		**Input**		**Unit**
	Vol max	**Voh min**	**Vil max**	**Vih min**	
Flash	0.45	2.55	0.8	2.1	V
FPGA	0.4	2.9	0.8	2.0	V

- Calculated Output Summary

FPGA ⇄ Flash				
- *From FPGA to Flash*				
Input	**Output**	**Noise Margin**		**Unit**
		Logic Zero	Logic One	
FPGA	Flash	**0.4**	**0.8**	**V**
- *From Flash to FPGA*				
Input	**Output**	**Noise Margin**		**Unit**
		Logic Zero	Logic One	
Flash	FPGA	**0.35**	**0.55**	**V**

5.2.1 From Basys 3 Artix-7 FPGA to Flash

Calculations:

Parameter	Description	Test Conditions		Min	Typ	Max	Unit
I_{LI}	Input Load Current	$V_{IN} = V_{SS}$ to V_{CC}, $V_{CC} = V_{CC\ max}$				±1.0	µA
	WP# Input Load Current	$V_{CC} = V_{CC\ max}$, WP# = V_{SS} to V_{CC}				±25	
I_{LIT}	A9 Input Load Current	$V_{CC} = V_{CC\ max}$; A9 = 12.5 V				35	
I_{LO}	Output Leakage Current	$V_{OUT} = V_{SS}$ to V_{CC}, $V_{CC} = V_{CC\ max}$				±1.0	
I_{CC1}	V_{CC} Active Read Current (Note 1)	CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$, Byte Mode	5 MHz		7	12	mA
			1 MHz		2	4	
		CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$, Word Mode	5 MHz		7	12	
			1 MHz		2	4	
I_{CC2}	V_{CC} Active Erase/Program Current (Notes 2, 3, 4)	CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$			20	30	mA
I_{CC3}	V_{CC} Standby Current (Note 4)	OE# = V_{IH}, CE#, RESET# = V_{CC} ± 0.3 V/-0.1V, WP# = V_{CC} or open, $V_{CC} = V_{CC\ max}$ (Note 5)			0.2	5	µA
I_{CC4}	V_{CC} Standby Current During Reset (Note 4)	$V_{CC} = V_{CC\ max}$; RESET# = V_{SS} ± 0.3 V/-0.1V WP# = V_{CC} or open, (Note 5)			0.2	5	µA
I_{CC5}	Automatic Sleep Mode (Notes 3, 4)	$V_{CC} = V_{CC\ max}$, $V_{IH} = V_{CC}$ ± 0.3 V, $V_{IL} = V_{SS}$ ± 0.3 V/-0.1 V, WP# = V_{CC} or open, (Note 5)			0.2	5	µA
V_{IL}	Input Low Voltage			-0.1		0.8	V
V_{IH}	Input High Voltage			0.7 x V_{CC}		V_{CC} + 0.3	
V_{ID}	Voltage for Autoselect and Temporary Sector Unprotect	V_{CC} = 2.7–3.6 V		8.5		12.5	
V_{OL}	Output Low Voltage	I_{OL} = 4.0 mA, $V_{CC} = V_{CC\ min}$				0.45	
V_{OH1}	Output High Voltage	I_{OH} = -2.0 mA, $V_{CC} = V_{CC\ min}$		0.85 x V_{CC}			
V_{OH2}		I_{OH} = -100 µA, $V_{CC} = V_{CC\ min}$		V_{CC}–0.4			
V_{LKO}	Low V_{CC} Lock-Out Voltage			2.1		2.5	

Notes
1. The I_{CC} current listed is typically less than 2 mA/MHz, with OE# at V_{IH}. Typical V_{CC} is 3.0 V.

"Flash Initial Values"

Table 8: **SelectIO DC Input and Output Levels**(1)(2)

I/O Standard	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V, Min	V, Max	V, Min	V, Max	V, Max	V, Min	mA, Max	mA, Min
HSTL_I	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_I_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_II	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSTL_II_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSUL_12	–0.300	V_{REF} – 0.130	V_{REF} + 0.130	V_{CCO} + 0.300	20% V_{CCO}	80% V_{CCO}	0.10	–0.10
LVCMOS12	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 3	Note 3
LVCMOS15	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	25% V_{CCO}	75% V_{CCO}	Note 4	Note 4
LVCMOS18	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.450	V_{CCO} – 0.450	Note 5	Note 5
LVCMOS25	–0.300	0.7	1.700	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVCMOS33	–0.300	0.8	2.000	3.450	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVTTL	–0.300	0.8	2.000	3.450	0.400	2.400	Note 5	Note 5
MOBILE_DDR	–0.300	20% V_{CCO}	80% V_{CCO}	V_{CCO} + 0.300	10% V_{CCO}	90% V_{CCO}	0.10	–0.10
PCI33_3	–0.400	30% V_{CCO}	50% V_{CCO}	V_{CCO} + 0.500	10% V_{CCO}	90% V_{CCO}	1.50	–0.50
SSTL135	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	13.00	–13.00
SSTL135_R	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	8.90	–8.90
SSTL15	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	V_{CCO}/2 – 0.175	V_{CCO}/2 + 0.175	13.00	–13.00

"Basys 3 Artix-7 FPGA Initial Values"

- $Noise\ Margin\ Logic\ Zero\ Case\ =\ NM_0 = V_{il\,max}\ -\ V_{ol\,max}$

Input: Flash ($V_{il\,max}$) Output: FPGA ($V_{ol\,max}$)

$$V_{il\,max}\ =\ 0.8\,V$$

$$V_{ol\,max}\ =\ 0.4\,V$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ 0.8\ -\ 0.4$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ \underline{\mathbf{0.4\ V}}$$

- $Noise\ Margin\ Logic\ One\ Case\ =\ NM_1\ =\ V_{oh\,min}\ -\ V_{ih\,min}$

Output: FPGA ($V_{oh\,min}$) Input: Flash ($V_{ih\,min}$)

$$V_{oh\ min}\ =\ V_{CCO}\ -\ 0.4 \qquad (\text{Where } V_{CCO}\ =\ 3.3\,V)$$

$$V_{oh\ min}\ =\ 3.3\ -\ 0.4\ =\ 2.9V$$

$$V_{ih\,min}\ =\ 0.7\ *\ V_{CC} \qquad (\text{Where } V_{CC}\ =\ 3.0\,V)$$

$$V_{ih\,min}\ =\ 0.7\ *\ 3.0\ =\ 2.1\,V$$

$$Logic\ One\ Case\ =\ 2.9\ -\ 2.1$$

$$Logic\ One\ Case\ =\ \underline{\mathbf{0.8\ V}}$$

- Therefore, after performing the calculations, we obtain a noise margin logic zero case of 0.4V, and a noise margin logic one case of 0.8 V from FPGA to the Flash.

5.2.2 From Flash to Basys 3 Artix 7 FPGA

Calculations:

Parameter	Description	Test Conditions		Min	Typ	Max	Unit
I_{LI}	Input Load Current	$V_{IN} = V_{SS}$ to V_{CC}, $V_{CC} = V_{CC\ max}$				±1.0	µA
	WP# Input Load Current	$V_{CC} = V_{CC\ max}$, WP# = V_{SS} to V_{CC}				±25	
I_{LIT}	A9 Input Load Current	$V_{CC} = V_{CC\ max}$; A9 = 12.5 V				35	
I_{LO}	Output Leakage Current	$V_{OUT} = V_{SS}$ to V_{CC}, $V_{CC} = V_{CC\ max}$				±1.0	
I_{CC1}	V_{CC} Active Read Current (Note 1)	CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$, Byte Mode	5 MHz		7	12	mA
			1 MHz		2	4	
		CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$, Word Mode	5 MHz		7	12	
			1 MHz		2	4	
I_{CC2}	V_{CC} Active Erase/Program Current (Notes 2, 3, 4)	CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$			20	30	mA
I_{CC3}	V_{CC} Standby Current (Note 4)	OE# = V_{IH}, CE#, RESET# = V_{CC} ± 0.3 V/-0.1V, WP# = V_{CC} or open, $V_{CC} = V_{CC\ max}$ (Note 5)			0.2	5	µA
I_{CC4}	V_{CC} Standby Current During Reset (Note 4)	$V_{CC} = V_{CC\ max}$; RESET# = V_{SS} ± 0.3 V/-0.1V WP# = V_{CC} or open, (Note 5)			0.2	5	µA
I_{CC5}	Automatic Sleep Mode (Notes 3, 4)	$V_{CC} = V_{CC\ max}$, $V_{IH} = V_{CC}$ ± 0.3 V, $V_{IL} = V_{SS}$ ± 0.3 V/-0.1 V, WP# = V_{CC} or open, (Note 5)			0.2	5	µA
V_{IL}	Input Low Voltage			-0.1		0.8	V
V_{IH}	Input High Voltage			0.7 x V_{CC}		V_{CC} + 0.3	
V_{ID}	Voltage for Autoselect and Temporary Sector Unprotect	V_{CC} = 2.7–3.6 V		8.5		12.5	
V_{OL}	Output Low Voltage	I_{OL} = 4.0 mA, $V_{CC} = V_{CC\ min}$				0.45	
V_{OH1}	Output High Voltage	I_{OH} = -2.0 mA, $V_{CC} = V_{CC\ min}$		0.85 x V_{CC}			
V_{OH2}		I_{OH} = -100 µA, $V_{CC} = V_{CC\ min}$		V_{CC}–0.4			
V_{LKO}	Low V_{CC} Lock-Out Voltage			2.1		2.5	

Notes
1. The I_{CC} current listed is typically less than 2 mA/MHz, with OE# at V_{IH}. Typical V_{CC} is 3.0 V.

"Flash Initial Values"

Table 8: **SelectIO DC Input and Output Levels**(1)(2)

I/O Standard	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V, Min	V, Max	V, Min	V, Max	V, Max	V, Min	mA, Max	mA, Min
HSTL_I	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_I_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_II	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSTL_II_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSUL_12	–0.300	V_{REF} – 0.130	V_{REF} + 0.130	V_{CCO} + 0.300	20% V_{CCO}	80% V_{CCO}	0.10	–0.10
LVCMOS12	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 3	Note 3
LVCMOS15	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	25% V_{CCO}	75% V_{CCO}	Note 4	Note 4
LVCMOS18	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.450	V_{CCO} – 0.450	Note 5	Note 5
LVCMOS25	–0.300	0.7	1.700	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVCMOS33	–0.300	0.8	2.000	3.450	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVTTL	–0.300	0.8	2.000	3.450	0.400	2.400	Note 5	Note 5
MOBILE_DDR	–0.300	20% V_{CCO}	80% V_{CCO}	V_{CCO} + 0.300	10% V_{CCO}	90% V_{CCO}	0.10	–0.10
PCI33_3	–0.400	30% V_{CCO}	50% V_{CCO}	V_{CCO} + 0.500	10% V_{CCO}	90% V_{CCO}	1.50	–0.50
SSTL135	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	13.00	–13.00
SSTL135_R	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	8.90	–8.90
SSTL15	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	V_{CCO}/2 – 0.175	V_{CCO}/2 + 0.175	13.00	–13.00

"Basys 3 Artix-7 FPGA Initial Values"

- $Noise\ Margin\ Logic\ Zero\ Case\ =\ NM_0 = V_{il\,max}\ -\ V_{ol\,max}$

Input: FPGA ($V_{il\,max}$) Output: Flash ($V_{ol\,max}$)

$$V_{il\,max}\ =\ 0.8\,V$$

$$V_{ol\,max}\ =\ 0.45\,V$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ 0.8\ -\ 0.45$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ \underline{\mathbf{0.35\ V}}$$

- $Noise\ Margin\ Logic\ One\ Case\ =\ NM_1\ =\ V_{oh\,min}\ -\ V_{ih\,min}$

Output: Flash ($V_{oh\,min}$) Input: FPGA ($V_{ih\,min}$)

$V_{oh\ min}\ =\ 0.85\ *\ V_{CC}$ (Where $V_{CC}\ =\ 3.0V$)

$$V_{oh\ min}\ =\ 0.85\ *\ 3.0\ =\ 2.55\,V$$

$$V_{ih\,min}\ =\ 2.0\,V$$

$$Logic\ One\ Case\ =\ 2.55\ -\ 2.0$$

$$Logic\ One\ Case\ =\ \underline{\mathbf{0.55\ V}}$$

- Therefore, after performing the calculations, we obtain a noise margin logic zero case of 0.35V, and a noise margin logic one case of 0.55V from Flash to the FPGA.

5.3 Parallel Interface Volatile Memory SRAM with the Processor

- Initial Values

Source	Output		Input		Units
	Vol max	**Voh min**	**Vil max**	**Vih min**	
SRAM	0.4	2.4	0.8	2.2	V
Processor	0.45	2.9	0.8	2.0	V

- Calculated Output Summary

Processor ⇄ SRAM				
- *From Processor to SRAM*				
Input	**Output**	**Noise Margin**		**Units**
		Logic Zero	Logic One	
Processor	SRAM	**0.35**	**0.7**	**V**
- *From SRAM to Processor*				
Input	**Output**	**Noise Margin**		**Units**
		Logic Zero	Logic One	
SRAM	Processor	**0.4**	**0.4**	**V**

5.3.1 From Processor to SRAM

Calculations:

DC ELECTRICAL CHARACTERISTICS

PARAMETER	SYMBOL	TEST CONDITION		MIN.	TYP. *4	MAX.	UNIT
Supply Voltage	V_{CC}		-10	2.7	3.3	3.6	V
Input High Voltage	V_{IH} *1			2.2	-	V_{CC}+0.3	V
Input Low Voltage	V_{IL} *2			- 0.3	-	0.8	V
Input Leakage Current	I_{LI}	$V_{CC} \geqq V_{IN} \geqq V_{SS}$		- 1	-	1	µA
Output Leakage Current	I_{LO}	$V_{CC} \geqq V_{OUT} \geqq V_{SS}$, Output Disabled		- 1	-	1	µA
Output High Voltage	V_{OH}	I_{OH} = -8mA		2.4	-	-	V
Output Low Voltage	V_{OL}	I_{OL} =4mA		-	-	0.4	V
Average Operating Power supply Current	Icc	CE# = V_{IL} , $I_{I/O}$ = 0mA ;f=max	-10	-	100	130	mA
	Icc_1	CE# ≧V_{CC} - 0.2V, Other pin is at 0.2V or Vcc-0. $I_{I/O}$ = 0mA;f=max	-10		80	110	mA
Standby Power Supply Current	Isb	CE# ≧Vih Other pin is at Vil or Vih				40	mA
Standby Power Supply Current	I_{SB1}	CE# ≧V_{CC} - 0.2V; Other pin is at 0.2V or Vcc-0.2V			3	25	mA

"SRAM Initial Values"

5.7 DC Electrical Characteristics

over recommended ranges of supply voltage and operating temperature (unless otherwise noted)(1)

PARAMETER			MIN	NOM	MAX	UNIT
DDR_RESETn,DDR_CSn0,DDR_CKE,DDR_CK,DDR_CKn,DDR_CASn,DDR_RASn,DDR_WEn,DDR_BA0,DDR_BA1,DDR_BA2,DDR_A0,DDR_A1,DDR_A2,DDR_A3,DDR_A4,DDR_A5,DDR_A6,DDR_A7,DDR_A8,DDR_A9,DDR_A10,DDR_A11,DDR_A12,DDR_A13,DDR_A14,DDR_A15,DDR_ODT,DDR_D0,DDR_D1,DDR_D2,DDR_D3,DDR_D4,DDR_D5,DDR_D6,DDR_D7,DDR_D8,DDR_D9,DDR_D10,DDR_D11,DDR_D12,DDR_D13,DDR_D14,DDR_D15,DDR_DQM0,DDR_DQM1,DDR_DQS0,DDR_DQSn0,DDR_DQS1,DDR_DQSn1 Pins (mDDR - LVCMOS Mode)						
All other LVCMOS pins (VDDSHVx = 3.3 V; x = 1 to 6)						
V_{IH}	High-level input voltage		2			V
V_{IL}	Low-level input voltage				0.8	V
V_{HYS}	Hysteresis voltage at an input		0.265		0.44	V
V_{OH}	High-level output voltage, driver enabled, pullup or pulldown disabled	I_{OH} = 6 mA	VDDSHVx – 0.45			V
V_{OL}	Low-level output voltage, driver enabled, pullup or pulldown disabled	I_{OL} = 6 mA			0.45	V

"Processor Initial Values"

- $Noise\ Margin\ Logic\ Zero\ Case\ =\ NM_0 = V_{il\,max}\ -\ V_{ol\,max}$

Input: SRAM ($V_{il\,max}$) Output: Processor ($V_{ol\,max}$)

$$V_{il\,max}\ =\ 0.8\,V$$

$$V_{ol\,max}\ =\ 0.45\,V$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ 0.8\ -\ 0.4$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ \underline{\mathbf{0.35\ V}}$$

- $Noise\ Margin\ Logic\ One\ Case\ =\ NM_1\ =\ V_{oh\,min}\ -\ V_{ih\,min}$

Output: Processor ($V_{oh\,min}$) Input: SRAM ($V_{ih\,min}$)

$V_{oh\ min}\ =\ V_{DDSSHVx} - 0.4$ (Where $V_{DDSSHVx}\ =\ 3.3\,V$)

$$V_{oh\ min}\ =\ 3.3\ -\ 0.4\ =\ 2.9V$$

$$V_{ih\,min}\ =\ 2.2\,V$$

$$Logic\ One\ Case\ =\ 2.9\ -\ 2.2$$

$$Logic\ One\ Case\ =\ \underline{\mathbf{0.7\ V}}$$

- Therefore, after performing the calculations, we obtain a noise margin logic zero case of 0.35 V, and a noise margin logic one case of 0.7 V from the processor to the SRAM.

5.3.2 From SRAM to Processor

Calculations:

DC ELECTRICAL CHARACTERISTICS

PARAMETER	SYMBOL	TEST CONDITION		MIN.	TYP. *4	MAX.	UNIT
Supply Voltage	V_{CC}		-10	2.7	3.3	3.6	V
Input High Voltage	V_{IH} *1			2.2	-	V_{CC}+0.3	V
Input Low Voltage	V_{IL} *2			- 0.3	-	0.8	V
Input Leakage Current	I_{LI}	$V_{CC} \geqq V_{IN} \geqq V_{SS}$		- 1	-	1	µA
Output Leakage Current	I_{LO}	$V_{CC} \geqq V_{OUT} \geqq V_{SS}$, Output Disabled		- 1	-	1	µA
Output High Voltage	V_{OH}	I_{OH} = -8mA		2.4	-	-	V
Output Low Voltage	V_{OL}	I_{OL} =4mA		-	-	0.4	V
Average Operating Power supply Current	Icc	CE# = V_{IL} , $I_{I/O}$ = 0mA ;f=max	-10	-	100	130	mA
	I_{CC1}	CE# ≧V_{CC} - 0.2V, Other pin is at 0.2V or Vcc-0. $I_{I/O}$ = 0mA;f=max	-10		80	110	mA
Standby Power Supply Current	Isb	CE# ≧Vih Other pin is at Vil or Vih				40	mA
Standby Power Supply Current	I_{SB1}	CE# ≧V_{CC} - 0.2V; Other pin is at 0.2V or Vcc-0.2V			3	25	mA

"SRAM Initial Values"

5.7 DC Electrical Characteristics

over recommended ranges of supply voltage and operating temperature (unless otherwise noted)(1)

PARAMETER			MIN	NOM	MAX	UNIT
DDR_RESETn,DDR_CSn0,DDR_CKE,DDR_CK,DDR_CKn,DDR_CASn,DDR_RASn,DDR_WEn,DDR_BA0,DDR_BA1,DDR_BA2,DDR_A0,DDR_A1,DDR_A2,DDR_A3,DDR_A4,DDR_A5,DDR_A6,DDR_A7,DDR_A8,DDR_A9,DDR_A10,DDR_A11,DDR_A12,DDR_A13,DDR_A14,DDR_A15,DDR_ODT,DDR_D0,DDR_D1,DDR_D2,DDR_D3,DDR_D4,DDR_D5,DDR_D6,DDR_D7,DDR_D8,DDR_D9,DDR_D10,DDR_D11,DDR_D12,DDR_D13,DDR_D14,DDR_D15,DDR_DQM0,DDR_DQM1,DDR_DQS0,DDR_DQSn0,DDR_DQS1,DDR_DQSn1 Pins (mDDR - LVCMOS Mode)						
All other LVCMOS pins (VDDSHVx = 3.3 V; x = 1 to 6)						
V_{IH}	High-level input voltage		2			V
V_{IL}	Low-level input voltage				0.8	V
V_{HYS}	Hysteresis voltage at an input		0.265		0.44	V
V_{OH}	High-level output voltage, driver enabled, pullup or pulldown disabled	I_{OH} = 6 mA	VDDSHVx – 0.45			V
V_{OL}	Low-level output voltage, driver enabled, pullup or pulldown disabled	I_{OL} = 6 mA			0.45	V

"Processor Initial Values"

- $Noise\ Margin\ Logic\ Zero\ Case\ =\ NM_0 = V_{il\,max}\ -\ V_{ol\,max}$

Input: Processor ($V_{il\,max}$) Output: SRAM ($V_{ol\,max}$)

$$V_{il\,max}\ =\ 0.8\ V$$

$$V_{ol\,max}\ =\ 0.4\ V$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ 0.8\ -\ 0.4$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ \underline{\mathbf{0.4\ V}}$$

- $Noise\ Margin\ Logic\ One\ Case\ =\ NM_1\ =\ V_{oh\,min}\ -\ V_{ih\,min}$

Output: SRAM ($V_{oh\,min}$) Input: Processor ($V_{ih\,min}$)

$$V_{oh\ min}\ =\ 2.4\ V$$

$$V_{ih\,min}\ =\ 2.0\ V$$

$$Logic\ One\ Case\ =\ 2.4\ -\ 2.0$$

$$Logic\ One\ Case\ =\ \underline{\mathbf{0.4\ V}}$$

- Therefore, after performing the calculations, we obtain a noise margin logic zero case of 0.4 V, and a noise margin logic one case of 0.4 V from the SRAM to the processor.

5.4 Parallel Interface Non-Volatile Memory Flash with the Processor

- Initial Values

Source	**Output**		**Input**		**Units**
	Vol max	**Voh min**	**Vil max**	**Vih min**	
Flash	0.45	2.55	0.8	2.1	V
Processor	0.45	2.85	0.8	2.0	V

- Calculations Outputs

Processor ⇄ Flash				
- *From Processor to Flash*				
Input	**Output**	**Noise Margin**		**Units**
		Logic Zero	Logic One	
Processor	Flash	**0.35**	**0.75**	**V**
- *From Flash to Processor*				
Input	**Output**	**Noise Margin**		**Units**
		Logic Zero	Logic One	
Flash	Processor	**0.35**	**0.55**	**V**

5.4.1 From Processor to Flash

Calculations:

Parameter	Description	Test Conditions		Min	Typ	Max	Unit
I_{LI}	Input Load Current	$V_{IN} = V_{SS}$ to V_{CC}, $V_{CC} = V_{CC\ max}$				±1.0	µA
	WP# Input Load Current	$V_{CC} = V_{CC\ max}$, WP# = V_{SS} to V_{CC}				±25	
I_{LIT}	A9 Input Load Current	$V_{CC} = V_{CC\ max}$; A9 = 12.5 V				35	
I_{LO}	Output Leakage Current	$V_{OUT} = V_{SS}$ to V_{CC}, $V_{CC} = V_{CC\ max}$				±1.0	
I_{CC1}	V_{CC} Active Read Current (Note 1)	CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$, Byte Mode	5 MHz		7	12	mA
			1 MHz		2	4	
		CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$, Word Mode	5 MHz		7	12	
			1 MHz		2	4	
I_{CC2}	V_{CC} Active Erase/Program Current (Notes 2, 3, 4)	CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$			20	30	mA
I_{CC3}	V_{CC} Standby Current (Note 4)	OE# = V_{IH}, CE#, RESET# = V_{CC} ± 0.3 V/-0.1V, WP# = V_{CC} or open, $V_{CC} = V_{CC\ max}$ (Note 5)			0.2	5	µA
I_{CC4}	V_{CC} Standby Current During Reset (Note 4)	$V_{CC} = V_{CC\ max}$; RESET# = V_{SS} ± 0.3 V/-0.1V WP# = V_{CC} or open, (Note 5)			0.2	5	µA
I_{CC5}	Automatic Sleep Mode (Notes 3, 4)	$V_{CC} = V_{CC\ max}$, $V_{IH} = V_{CC}$ ± 0.3 V, $V_{IL} = V_{SS}$ ± 0.3 V/-0.1 V, WP# = V_{CC} or open, (Note 5)			0.2	5	µA
V_{IL}	Input Low Voltage			-0.1		0.8	V
V_{IH}	Input High Voltage			0.7 x V_{CC}		V_{CC} + 0.3	
V_{ID}	Voltage for Autoselect and Temporary Sector Unprotect	V_{CC} = 2.7–3.6 V		8.5		12.5	
V_{OL}	Output Low Voltage	I_{OL} = 4.0 mA, $V_{CC} = V_{CC\ min}$				0.45	
V_{OH1}	Output High Voltage	I_{OH} = -2.0 mA, $V_{CC} = V_{CC\ min}$		0.85 x V_{CC}			
V_{OH2}		I_{OH} = -100 µA, $V_{CC} = V_{CC\ min}$		V_{CC}–0.4			
V_{LKO}	Low V_{CC} Lock-Out Voltage			2.1		2.5	

Notes

1. The I_{CC} current listed is typically less than 2 mA/MHz, with OE# at V_{IH}. Typical V_{CC} is 3.0 V.

"Flash Initial Values"

5.7 DC Electrical Characteristics

over recommended ranges of supply voltage and operating temperature (unless otherwise noted)(1)

PARAMETER			MIN	NOM	MAX	UNIT
DDR_RESETn,DDR_CSn0,DDR_CKE,DDR_CK,DDR_CKn,DDR_CASn,DDR_RASn,DDR_WEn,DDR_BA0,DDR_BA1,DDR_BA2,DDR_A0,DDR_A1,DDR_A2,DDR_A3,DDR_A4,DDR_A5,DDR_A6,DDR_A7,DDR_A8,DDR_A9,DDR_A10,DDR_A11,DDR_A12,DDR_A13,DDR_A14,DDR_A15,DDR_ODT,DDR_D0,DDR_D1,DDR_D2,DDR_D3,DDR_D4,DDR_D5,DDR_D6,DDR_D7,DDR_D8,DDR_D9,DDR_D10,DDR_D11,DDR_D12,DDR_D13,DDR_D14,DDR_D15,DDR_DQM0,DDR_DQM1,DDR_DQS0,DDR_DQSn0,DDR_DQS1,DDR_DQSn1 Pins (mDDR - LVCMOS Mode)						
All other LVCMOS pins (VDDSHVx = 3.3 V; x = 1 to 6)						
V_{IH}	High-level input voltage		2			V
V_{IL}	Low-level input voltage				0.8	V
V_{HYS}	Hysteresis voltage at an input		0.265		0.44	V
V_{OH}	High-level output voltage, driver enabled, pullup or pulldown disabled	I_{OH} = 6 mA	VDDSHVx – 0.45			V
V_{OL}	Low-level output voltage, driver enabled, pullup or pulldown disabled	I_{OL} = 6 mA			0.45	V

"Processor Initial Values"

- $Noise\ Margin\ Logic\ Zero\ Case\ =\ NM_0 = V_{il\,max}\ -\ V_{ol\,max}$

Input: Flash ($V_{il\,max}$) Output: Processor ($V_{ol\,max}$)

$$V_{il\,max}\ =\ 0.8\,V$$

$$V_{ol\,max}\ =\ 0.45\,V$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ 0.8\ -\ 0.45$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ \underline{\mathbf{0.35\ V}}$$

- $Noise\ Margin\ Logic\ One\ Case\ =\ NM_1\ =\ V_{oh\,min}\ -\ V_{ih\,min}$

Output: Processor ($V_{oh\,min}$) Input: SRAM ($V_{ih\,min}$)

$V_{oh\ min}\ =\ V_{DDSSHVx} - 0.45$ (Where $V_{DDSSHVx} =\ 3.3\,V$)

$$V_{oh\ min}\ =\ 3.3\ -\ 0.45\ =\ 2.85\,V$$

$V_{oh\ min}\ =\ 0.7\ *\ V_{CC}$ (Where $V_{CC} =\ 3.0V$)

$$V_{ih\,min}\ =\ 0.7\ *\ 3.0\ =\ 2.1\,V$$

$$Logic\ One\ Case\ =\ 2.85\ -\ 2.1$$

$$Logic\ One\ Case\ =\ \underline{\mathbf{0.75\ V}}$$

- Therefore, after performing the calculations, we obtain a noise margin logic zero case of 0.35 V, and a noise margin logic one case of 0.75 V from the processor to the Flash.

5.4.2 From Flash to Processor

Calculations:

Parameter	Description	Test Conditions		Min	Typ	Max	Unit
I_{LI}	Input Load Current	$V_{IN} = V_{SS}$ to V_{CC}, $V_{CC} = V_{CC\ max}$				±1.0	µA
	WP# Input Load Current	$V_{CC} = V_{CC\ max}$, WP# = V_{SS} to V_{CC}				±25	
I_{LIT}	A9 Input Load Current	$V_{CC} = V_{CC\ max}$; A9 = 12.5 V				35	
I_{LO}	Output Leakage Current	$V_{OUT} = V_{SS}$ to V_{CC}, $V_{CC} = V_{CC\ max}$				±1.0	
I_{CC1}	V_{CC} Active Read Current (Note 1)	CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$, Byte Mode	5 MHz		7	12	mA
			1 MHz		2	4	
		CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$, Word Mode	5 MHz		7	12	
			1 MHz		2	4	
I_{CC2}	V_{CC} Active Erase/Program Current (Notes 2, 3, 4)	CE# = V_{IL}, OE# = V_{IH}, $V_{CC} = V_{CC\ max}$			20	30	mA
I_{CC3}	V_{CC} Standby Current (Note 4)	OE# = V_{IH}, CE#, RESET# = V_{CC} ± 0.3 V/-0.1V, WP# = V_{CC} or open, $V_{CC} = V_{CC\ max}$ (Note 5)			0.2	5	µA
I_{CC4}	V_{CC} Standby Current During Reset (Note 4)	$V_{CC} = V_{CC\ max}$; RESET# = V_{SS} ± 0.3 V/-0.1V WP# = V_{CC} or open, (Note 5)			0.2	5	µA
I_{CC5}	Automatic Sleep Mode (Notes 3, 4)	$V_{CC} = V_{CC\ max}$, $V_{IH} = V_{CC}$ ± 0.3 V, $V_{IL} = V_{SS}$ ± 0.3 V/-0.1 V, WP# = V_{CC} or open, (Note 5)			0.2	5	µA
V_{IL}	Input Low Voltage			-0.1		0.8	V
V_{IH}	Input High Voltage			0.7 x V_{CC}		V_{CC} + 0.3	
V_{ID}	Voltage for Autoselect and Temporary Sector Unprotect	V_{CC} = 2.7–3.6 V		8.5		12.5	
V_{OL}	Output Low Voltage	I_{OL} = 4.0 mA, $V_{CC} = V_{CC\ min}$				0.45	
V_{OH1}	Output High Voltage	I_{OH} = -2.0 mA, $V_{CC} = V_{CC\ min}$		0.85 x V_{CC}			
V_{OH2}		I_{OH} = -100 µA, $V_{CC} = V_{CC\ min}$		V_{CC}–0.4			
V_{LKO}	Low V_{CC} Lock-Out Voltage			2.1		2.5	

Notes

1. The I_{CC} current listed is typically less than 2 mA/MHz, with OE# at V_{IH}. Typical V_{CC} is 3.0 V.

"Flash Initial Values"

5.7 DC Electrical Characteristics

over recommended ranges of supply voltage and operating temperature (unless otherwise noted)(1)

PARAMETER			MIN	NOM	MAX	UNIT
DDR_RESETn,DDR_CSn0,DDR_CKE,DDR_CK,DDR_CKn,DDR_CASn,DDR_RASn,DDR_WEn,DDR_BA0,DDR_BA1,DDR_BA2,DDR_A0,DDR_A1,DDR_A2,DDR_A3,DDR_A4,DDR_A5,DDR_A6,DDR_A7,DDR_A8,DDR_A9,DDR_A10,DDR_A11,DDR_A12,DDR_A13,DDR_A14,DDR_A15,DDR_ODT,DDR_D0,DDR_D1,DDR_D2,DDR_D3,DDR_D4,DDR_D5,DDR_D6,DDR_D7,DDR_D8,DDR_D9,DDR_D10,DDR_D11,DDR_D12,DDR_D13,DDR_D14,DDR_D15,DDR_DQM0,DDR_DQM1,DDR_DQS0,DDR_DQSn0,DDR_DQS1,DDR_DQSn1 Pins (mDDR - LVCMOS Mode)						
All other LVCMOS pins (VDDSHVx = 3.3 V; x = 1 to 6)						
V_{IH}	High-level input voltage		2			V
V_{IL}	Low-level input voltage				0.8	V
V_{HYS}	Hysteresis voltage at an input		0.265		0.44	V
V_{OH}	High-level output voltage, driver enabled, pullup or pulldown disabled	I_{OH} = 6 mA	VDDSHVx – 0.45			V
V_{OL}	Low-level output voltage, driver enabled, pullup or pulldown disabled	I_{OL} = 6 mA			0.45	V

"Processor Initial Values"

- $Noise\ Margin\ Logic\ Zero\ Case\ =\ NM_0 = V_{il\,max}\ -\ V_{ol\,max}$

Input: Processor ($V_{il\,max}$) Output: Flash ($V_{ol\,max}$)

$$V_{il\,max}\ =\ 0.8\,V$$

$$V_{ol\,max}\ =\ 0.45\,V$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ 0.8\ -\ 0.45$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ \underline{\mathbf{0.35\ V}}$$

- $Noise\ Margin\ Logic\ One\ Case\ =\ NM_1\ =\ V_{oh\,min}\ -\ V_{ih\,min}$

Output: Flash ($V_{oh\,min}$) Input: Processor ($V_{ih\,min}$)

$V_{oh\ min}\ =\ 0.85\ *\ V_{CC}$ (Where $V_{CC}\ =\ 3.0\,V$)

$$V_{oh\ min}\ =\ 0.85\ *\ 3.0\ =\ 2.55\,V$$

$$V_{ih\,min}\ =\ 2.0\,V$$

$$Logic\ One\ Case\ =\ 2.55\ -\ 2.0$$

$$Logic\ One\ Case\ =\ \underline{\mathbf{0.55\ V}}$$

- Therefore, after performing the calculations, we obtain a noise margin logic zero case of 0.35 V, and a noise margin logic one case of 0.55 V from the Flash to the processor.

5.5 Basys 3 Artix 7 FPGA with the Processor

- Initial Values

Source	Output		Input		Units
	Vol max	**Voh min**	**Vil max**	**Vih min**	
Processor	0.45	2.85	0.8	2.0	V
FPGA	0.4	2.9	0.8	2.0	V

- Calculated Output Summary

FPGA ⇄ Processor				
- *From FPGA to Processor*				
Input	**Output**	**Noise Margin**		**Units**
		Logic Zero	Logic One	
FPGA	Processor	**0.4**	**0.9**	**V**
- *From Processor to FPGA*				
Input	**Output**	**Noise Margin**		**Units**
		Logic Zero	Logic One	
Processor	FPGA	**0.35**	**0.85**	**V**

5.5.1 From Basys 3 Artix 7 FPGA to Processor

Calculations:

5.7 DC Electrical Characteristics

over recommended ranges of supply voltage and operating temperature (unless otherwise noted)(1)

PARAMETER			MIN	NOM	MAX	UNIT
DDR_RESETn,DDR_CSn0,DDR_CKE,DDR_CK,DDR_CKn,DDR_CASn,DDR_RASn,DDR_WEn,DDR_BA0,DDR_BA1,DDR_BA2,DDR_A0,DDR_A1,DDR_A2,DDR_A3,DDR_A4,DDR_A5,DDR_A6,DDR_A7,DDR_A8,DDR_A9,DDR_A10,DDR_A11,DDR_A12,DDR_A13,DDR_A14,DDR_A15,DDR_ODT,DDR_D0,DDR_D1,DDR_D2,DDR_D3,DDR_D4,DDR_D5,DDR_D6,DDR_D7,DDR_D8,DDR_D9,DDR_D10,DDR_D11,DDR_D12,DDR_D13,DDR_D14,DDR_D15,DDR_DQM0,DDR_DQM1,DDR_DQS0,DDR_DQSn0,DDR_DQS1,DDR_DQSn1 Pins (mDDR - LVCMOS Mode)						
All other LVCMOS pins (VDDSHVx = 3.3 V; x = 1 to 6)						
V_{IH}	High-level input voltage		2			V
V_{IL}	Low-level input voltage				0.8	V
V_{HYS}	Hysteresis voltage at an input		0.265		0.44	V
V_{OH}	High-level output voltage, driver enabled, pullup or pulldown disabled	I_{OH} = 6 mA	VDDSHVx – 0.45			V
V_{OL}	Low-level output voltage, driver enabled, pullup or pulldown disabled	I_{OL} = 6 mA			0.45	V

"Processor Initial Values"

Table 8: **SelectIO DC Input and Output Levels(1)(2)**

I/O Standard	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V, Min	V, Max	V, Min	V, Max	V, Max	V, Min	mA, Max	mA, Min
HSTL_I	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_I_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_II	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSTL_II_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSUL_12	–0.300	V_{REF} – 0.130	V_{REF} + 0.130	V_{CCO} + 0.300	20% V_{CCO}	80% V_{CCO}	0.10	–0.10
LVCMOS12	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 3	Note 3
LVCMOS15	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	25% V_{CCO}	75% V_{CCO}	Note 4	Note 4
LVCMOS18	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.450	V_{CCO} – 0.450	Note 5	Note 5
LVCMOS25	–0.300	0.7	1.700	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVCMOS33	–0.300	0.8	2.000	3.450	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVTTL	–0.300	0.8	2.000	3.450	0.400	2.400	Note 5	Note 5
MOBILE_DDR	–0.300	20% V_{CCO}	80% V_{CCO}	V_{CCO} + 0.300	10% V_{CCO}	90% V_{CCO}	0.10	–0.10
PCI33_3	–0.400	30% V_{CCO}	50% V_{CCO}	V_{CCO} + 0.500	10% V_{CCO}	90% V_{CCO}	1.50	–0.50
SSTL135	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	13.00	–13.00
SSTL135_R	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	8.90	–8.90
SSTL15	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	V_{CCO}/2 – 0.175	V_{CCO}/2 + 0.175	13.00	–13.00

"Basys 3 Artix-7 FPGA Initial Values"

- $Noise\ Margin\ Logic\ Zero\ Case\ =\ NM_0 = V_{il\,max}\ -\ V_{ol\,max}$

Input: Processor ($V_{il\,max}$) Output: FPGA ($V_{ol\,max}$)

$$V_{il\,max}\ =\ 0.8\,V$$

$$V_{ol\,max}\ =\ 0.4\,V$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ 0.8\ -\ 0.4$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ \underline{\mathbf{0.4\ V}}$$

- $Noise\ Margin\ Logic\ One\ Case\ =\ NM_1\ =\ V_{oh\,min}\ -\ V_{ih\,min}$

Output: FPGA ($V_{oh\,min}$) Input: Processor ($V_{ih\,min}$)

$$V_{oh\ min}\ =\ V_{CCO}\ -\ 0.4 \qquad (\text{Where } V_{CCO}\ =\ 3.3\,V)$$

$$V_{oh\ min}\ =\ 3.3\ -\ 0.4\ =\ 2.9V$$

$$V_{ih\,min}\ =\ 2.0\,V$$

$$Logic\ One\ Case\ =\ 2.9\ -\ 2.0$$

$$Logic\ One\ Case\ =\ \underline{\mathbf{0.9\ V}}$$

- Therefore, after performing the calculations, we obtain a noise margin logic zero case of 0.4V, and a noise margin logic one case of 0.9 V from FPGA to the Processor.

5.5.2 From Processor to Basys 3 Artix 7 FPGA

Calculations:

5.7 DC Electrical Characteristics

over recommended ranges of supply voltage and operating temperature (unless otherwise noted)(1)

PARAMETER			MIN	NOM	MAX	UNIT
DDR_RESETn,DDR_CSn0,DDR_CKE,DDR_CK,DDR_CKn,DDR_CASn,DDR_RASn,DDR_WEn,DDR_BA0,DDR_BA1,DDR_BA2,DDR_A0,DDR_A1,DDR_A2,DDR_A3,DDR_A4,DDR_A5,DDR_A6,DDR_A7,DDR_A8,DDR_A9,DDR_A10,DDR_A11,DDR_A12,DDR_A13,DDR_A14,DDR_A15,DDR_ODT,DDR_D0,DDR_D1,DDR_D2,DDR_D3,DDR_D4,DDR_D5,DDR_D6,DDR_D7,DDR_D8,DDR_D9,DDR_D10,DDR_D11,DDR_D12,DDR_D13,DDR_D14,DDR_D15,DDR_DQM0,DDR_DQM1,DDR_DQS0,DDR_DQSn0,DDR_DQS1,DDR_DQSn1 Pins (mDDR - LVCMOS Mode)						
All other LVCMOS pins (VDDSHVx = 3.3 V; x = 1 to 6)						
V_{IH}	High-level input voltage		2			V
V_{IL}	Low-level input voltage				0.8	V
V_{HYS}	Hysteresis voltage at an input		0.265		0.44	V
V_{OH}	High-level output voltage, driver enabled, pullup or pulldown disabled	I_{OH} = 6 mA	VDDSHVx – 0.45			V
V_{OL}	Low-level output voltage, driver enabled, pullup or pulldown disabled	I_{OL} = 6 mA			0.45	V

"Processor Initial Values"

Table 8: **SelectIO DC Input and Output Levels(1)(2)**

I/O Standard	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V, Min	V, Max	V, Min	V, Max	V, Max	V, Min	mA, Max	mA, Min
HSTL_I	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_I_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_II	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSTL_II_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSUL_12	–0.300	V_{REF} – 0.130	V_{REF} + 0.130	V_{CCO} + 0.300	20% V_{CCO}	80% V_{CCO}	0.10	–0.10
LVCMOS12	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 3	Note 3
LVCMOS15	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	25% V_{CCO}	75% V_{CCO}	Note 4	Note 4
LVCMOS18	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.450	V_{CCO} – 0.450	Note 5	Note 5
LVCMOS25	–0.300	0.7	1.700	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVCMOS33	–0.300	0.8	2.000	3.450	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVTTL	–0.300	0.8	2.000	3.450	0.400	2.400	Note 5	Note 5
MOBILE_DDR	–0.300	20% V_{CCO}	80% V_{CCO}	V_{CCO} + 0.300	10% V_{CCO}	90% V_{CCO}	0.10	–0.10
PCI33_3	–0.400	30% V_{CCO}	50% V_{CCO}	V_{CCO} + 0.500	10% V_{CCO}	90% V_{CCO}	1.50	–0.50
SSTL135	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	13.00	–13.00
SSTL135_R	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	8.90	–8.90
SSTL15	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	V_{CCO}/2 – 0.175	V_{CCO}/2 + 0.175	13.00	–13.00

"Basys 3 Artix-7 FPGA Initial Values"

- $Noise\ Margin\ Logic\ Zero\ Case\ =\ NM_0 = V_{il\,max}\ -\ V_{ol\,max}$

Input: FPGA ($V_{il\,max}$) Output: Processor ($V_{ol\,max}$)

$$V_{il\,max}\ =\ 0.8\,V$$

$$V_{ol\,max}\ =\ 0.45\,V$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ 0.8\ -\ 0.45$$

$$Noise\ Margin\ Logic\ Zero\ Case\ =\ \underline{\mathbf{0.35\ V}}$$

- $Noise\ Margin\ Logic\ One\ Case\ =\ NM_1\ =\ V_{oh\,min}\ -\ V_{ih\,min}$

Output: Processor ($V_{oh\,min}$) Input: FPGA ($V_{ih\,min}$)

$$V_{oh\ min}\ =\ V_{DDSHVx}\ -\ 0.45 \qquad (Where\ V_{DDSHVx}\ =\ 3.3\,V)$$

$$V_{oh\ min}\ =\ 3.3\ -\ 0.45\ =\ 2.85$$

$$V_{ih\,min}\ =\ 2.0\,V$$

$$Logic\ One\ Case\ =\ 2.85\ -\ 2.0$$

$$Logic\ One\ Case\ =\ \underline{\mathbf{0.85\ V}}$$

- Therefore, after performing the calculations, we obtain a noise margin logic zero case of 0.35 V, and a noise margin logic one case of 0.85 V from SRAM to the FPGA.

6 Loading Analysis

Loading analysis evaluates how many circuit elements a given output can drive without degrading the signal. Loading analysis requires many I/O electrical characteristic values, just like the noise margin, that can be used to calculate the total input currents and capacitance, as well as the margin. Mainly for the loading analysis, the processor or CPU derives the other components, such as the external memories and FPGA. Source signals like the data bus signals, address bus signals, and control bus signals are derived from the CPU and sent to the other components.

6.1 Loading Analysis Equations

After obtaining the I/O electrical characteristic values, it is necessary to calculate the total input current and capacitance from the source to the derived load, which is calculated by multiplying the input current/capacitance by the quantity. Next, we move to calculate the total input current/capacitance for all the derived loads, but adding all the totals for each input current and capacitance. Finally, we calculate the margin by subtracting the output currents and capacitance load from the total input current/capacitance of all the loads. These equations, along with the loading analysis table can be found below for a more detailed and comprehensive evaluation of the system's response.

- Equations of the total input current and capacitance from the source to the derived load,

$$I_{IL}\,Total^{1} = Quantity * I_{IL}$$
$$I_{IH}\,Total^{1} = Quantity * I_{IH}$$
$$C_{in}\,Total^{1} = Quantity * C_{in}$$

- Equations of the total input current/capacitance for all the derived loads,

$$I_{IL}\,Total^{2} = Total\ of\ First\ Load + Total\ of\ Second\ Load + \ldots$$
$$I_{IH}\,Total^{2} = Total\ of\ First\ Load + Total\ of\ Second\ Load + \ldots$$
$$C_{in}\,Total^{2} = Total\ of\ First\ Load + Total\ of\ Second\ Load + \ldots$$

- Equations of the margin,

$$I_{IL}\,Margin = I_{OL} - I_{IL}\,Total^{2}$$
$$I_{IH}\,Margin = I_{OH} - I_{IH}\,Total^{2}$$
$$C_{in}\,Margin = C_{L} - C_{in}\,Total^{2}$$

6.2 Loading Analysis Table

Source						**Load**			**Unit Load**			**Total[1]**		
			uA	uA	pF				uA	uA	pF	uA	uA	pF
Signal	**Pin #**	**Source**	**IOL**	**IOH**	**CL**	**Load**	**Signal**	Qty	**IIL**	**IIH**	**Cin**	**IIL**	**IIH**	**Cin**
WP	4	CPU	6000	6000	30	Flash	WP	1	-1	1	6	-1	1	6
						Artix-7	WP	1	15	15	8	15	15	8
						wire cap		15			1			15
											Total[2]	**14**	**16**	**29**
											Margin	**5986**	**5984**	**1**
CE, WE, RD	6	CPU	6000	6000	30	SRAM	CE, WE, (OE/RD)	1	-1	1	8	-1	1	8
						Flash	CE, WE, (OE/RD)	1	-1	1	6	-1	1	6
						Artix-7	CE, WE, (OE/RD)	1	15	15	8	15	15	8
						wire cap		20			1			20
											Total[2]	**13**	**17**	**42**
											Margin	**5987**	**5983**	**-12**
A1		CPU	6000	6000	30	Flash	A-1	1	-1	1	6	-1	1	6
						Artix-7		1	15	15	8	15	15	8
						wire cap		15			1			15
											Total[2]	**14**	**16**	**29**
											Margin	**5986**	**5984**	**1**
A2…7		CPU	6000	6000	30	Flash	A2…7	1	-1	1	6	-1	1	6
						Artix-7		1	15	15	8	15	15	8
						wire cap		15			1			15
											Total[2]	**14**	**16**	**29**
											Margin	**5986**	**5984**	**1**
A8…15		CPU	6000	6000	30	SRAM	A8…15	1	-1	1	8	-1	1	8

						Artix-7		1	15	15	8	15	15	8
						wire cap		15			1			15
											Total²	**14**	**16**	**31**
											Margin	**5986**	**-5984**	**-1**
AD0..15		CPU	6000	6000	30	SRAM	DQ0…15	1	-1	1	8	-1	1	8
						Flash	DQ0…15	1	-1	1	6	-1	1	6
						Artix-7		1	15	15	8	15	15	8
						wire cap		20			1			20
											Total²	**13**	**17**	**42**
											Margin	**5987**	**5983**	**-12**
DQ0…15	7-38	SRAM	4000	-8000	30	CPU	AD0..15	1	-243	-19	5.5	-243	-19	5.5
						Flash	DQ0…15	1	-1	1	6	-1	1	6
						Artix-7		1	15	15	8	15	15	8
						wire cap		20			1			20
											Total²	**-229**	**-3**	**39.5**
											Margin	**3771**	**-8003**	**-9.5**
DQ0…15	29-45	Flash	4000	-2000	30	SRAM	DQ0…15	1	-1	1	8	-1	1	8
						CPU	AD0..15	1	-243	-19	5.5	-243	-19	5.5
						Artix-7		1	15	15	8	15	15	8
						wire cap		20			1			20
											Total²	**-229**	**-3**	**41.5**
											Margin	**3771**	**-2003**	**-11.5**
		Artix-7	12000	12000	0	CPU	AD0..15	1	-243	-19	5.5	-243	-19	5.5
						Flash	DQ0…15	1	-1	1	6	-1	1	6
						SRAM	DQ0…15	1	-1	1	8	-1	1	8
						wire cap		20			1			20
											Total²	**-245**	**-17**	**39.5**
											Margin	**11755**	**11983**	**-39.5**

7 Memory Address Mapping Tables

Memory address mapping tables consist of two main address mapping tables, a program address map and a data memory address map. A program address map, commonly referred to as a memory address map, is a table that illustrates the memory addresses assigned to each chip in a computer system. It is a visual representation of the address space of each chip. Each byte of memory is assigned an address. The CPU utilizes these addresses to locate particular memory regions. Program address memory is normally read-only or non-volatile, such as Flash memory, since it stores the code the processor will execute, usually pre-loaded before operation.

A data memory address refers to a specific location in the computer's memory where data can be stored or retrieved. It is used to access and manipulate data during the execution of a program. A memory address is represented in hexadecimal format, a base-16 numbering system. This format uses numbers 0-9 and letters A-F, offering a more compact representation than binary. Hexadecimal makes it easier for programmers to read and work with memory addresses. Data memory addresses, such as SRAM, are volatile because they store data that changes during program execution, such as intermediate results, variables, and stack/heap data. Both tables are shown below along with the address ranges and external memories. A visual representation of both tables is also provided in the Verilog code, and a simulation of both tables is also provided in modules **8.2 ProgramAddressMap.v** and **8.3 DataMemoryAddress.v**.

7.1 Program Address Map Table

Address Range (hex) (Binary)	**Address bits** A31-A12	**Chip Select Active for Memory IC**	**Size**
0000_0000 - 07FF_FFFF 0000_0000_0000_0000_0000_0000_0000_0000 - 0000_0111_1111_1111_1111_1111_1111_1111	0000_0000_ 0000_0000_ 0000	Flash 0	128 MB
0800_0000 - 0FFF_FFFF 0000_1000_0000_0000_0000_0000_0000_0000 - 0000_1111_1111_1111_1111_1111_1111_1111	0000_1000_ 0000_0000_ 0000	Flash 1	128 MB
2000_0000 - 4AFF_FFFF 0010_0000_0000_0000_0000_0000_0000_0000 - 0100_1010_1111_1111_1111_1111_1111_1111	0010_0000_ 0000_0000_ 0000	Not Used	682 MB

7.2 Data Memory Address Table

Address Range (hex) (Binary)	**Address bits** A31-A12	**Active Select: Memory / I/O**	**Size**
1000_0000 - 13FF_FFFF 0001_0000_0000_0000_0000_0000_0000_0000 - 0001_0011_1111_1111_1111_1111_1111_1111	0001_0000_0000_0000_0000	SRAM 0	64 MB
1400_0000 - 17FF_FFFF 0001_0100_0000_0000_0000_0000_0000_0000 - 0001_0111_1111_1111_1111_1111_1111_1111	0001_0100_0000_0000_0000	SRAM 1	64 MB
2000_0000 - 44E0_FFFF 0010_0000_0000_0000_0000_0000_0000_0000 - 0100_0100_1110_0000_1111_1111_1111_1111	0010_0000_0000_0000_0000	Not Used	593 MB
44E1_0000 - 44E1_1FFF 0100_0100_1110_0001_0000_0000_0000_0000 - 0100_0100_1110_0001_0001_1111_1111_1111	0100_0100_1110_0001_0000	Control Module	128 KB
44E1_2000 - 4802_1FFF 0100_0100_1110_0001_0010_0000_0000_0000 - 0100_1000_0000_0010_0001_1111_1111_1111	0100_0100_1110_0001_0010	Not Used	56 MB
4802_2000 - 4802_2FFF 0100_1000_0000_0010_0010_0000_0000_0000 - 0100_1000_0000_0010_0010_1111_1111_1111	0100_1000_0000_0010_0010	UART1	4 KB
4802_3000 - 4AFF_FFFF 0100_1000_0000_0010_0011_0000_0000_0000 - 0100_1010_1111_1111_1111_1111_1111_1111	0100_1000_0000_0010_0011	Not Used	44 MB

8 SpeakUp Click Speech Recognition Module to Basys-3 Artix-7 FPGA

MikroElektronika's SpeakUp Click is a speech recognition board that enables voice commands to operate various devices. The board is perfect for voice-activated control systems because its primary function is the recognition of up to 200 speech commands, which may be programmed and assigned to various activities. This project will use this board to transmit voice commands spoken by the user to the Basys-3 Artix-7 FPGA. The focus of this project is to output this command, specifically numbers between 0 and 9999, and display the recognized number on a 4-digit 7-segment display on the FPGA. The following section provides a detailed logical method of how 10,000 numbers will be captured and received in only 45 commands, rather than 10,000 commands. Module **9.2 DigitTo7SegmentDisplay.v** was implemented to evaluate this method, and the **9.10 Simulation Waveform**'s output confirms the operational efficiency of his approach.

8.1 Commands and Efficient Logic Handling

Command	Number Group	ID	Number
N/A	N/A	0	0
One	Ones	1	1
Two	Ones	2	2
Three	Ones	3	3
Four	Ones	4	4
Five	Ones	5	5
Six	Ones	6	6
Seven	Ones	7	7
Eight	Ones	8	8
Nine	Ones	9	9
Ten	Tens	10	10
Eleven	Tens	11	11
Twelve	Tens	12	12
Thirteen	Tens	13	13
Fourteen	Tens	14	14

Fifteen	Tens	15	15
Sixteen	Tens	16	16
Seventeen	Tens	17	17
Eighteen	Tens	18	18
Nineteen	Tens	19	19
Twenty	Tens	20	20
Thirty	Tens	21	30
Forty	Tens	22	40
Fifty	Tens	23	50
Sixty	Tens	24	60
Seventy	Tens	25	70
Eighty	Tens	26	80
Ninety	Tens	27	90
One-Hundred	Hundreds	28	100
Two-Hundred	Hundreds	29	200
Three-Hundred	Hundreds	30	300
Four-Hundred	Hundreds	31	400
Five-Hundred	Hundreds	32	500
Six-Hundred	Hundreds	33	600
Seven-Hundred	Hundreds	34	700
Eight-Hundred	Hundreds	35	800
Nine-Hundred	Hundreds	36	900
One-Thousand	Thousands	37	1000
Two-Thousand	Thousands	38	2000
Three-Thousand	Thousands	39	3000

Four-Thousand	Thousands	40	4000
Five-Thousand	Thousands	41	5000
Six-Thousand	Thousands	42	6000
Seven-Thousand	Thousands	43	7000
Eight-Thousand	Thousands	44	8000
Nine-Thousand	Thousands	45	9000
Done	N/A	46	N/A
Start	N/A	47	N/A

8.2 Numeric Speech Examples to 7-Segment Display Output

Examples of how commands will be implemented in Verilog and the output of the 7-segment display are shown in this section. It is clear that to guarantee default initialization for any unspoken or unrecognized command, a 0 will be automatically assigned to the ID if a command number is not spoken.

1. **Number Spoken**: Nine-Thousand Two-Hundred Fifty-Four
   Verilog Logic: Thousands[ID] + Hundreds[ID] + Tens[ID] + Ones[ID] =
   Thousands[47] + Hundreds[31] + Tens[23] + Ones[5] =
   9000 + 200 + 50 + 4 = 9254
   Basys-3 7-Segment Display Output: **9254**
2. **Number Spoken**: Eight-Thousand
   Verilog Logic: Thousands[ID] + Hundreds[ID] + Tens[ID] + Ones[ID] =
   Thousands[45] + Hundreds[0] + Tens[0] + Ones[0] =
   8000 + 0 + 0 + 0 = 8000
   Basys-3 7-Segment Display Output: 8000
3. **Number Spoken**: Eight-Hundred Seventy-Six
   Verilog Logic: Thousands[ID] + Hundreds[ID] + Tens[ID] + Ones[ID] =
   Thousands[0] + Hundreds[36] + Tens[25] + Ones[6] =
   0 + 800 + 70 + 6 = 876
   Basys-3 7-Segment Display Output: **876**
4. **Number Spoken**: Nineteen
   Verilog Logic: Thousands[ID] + Hundreds[ID] + Tens[ID] + Ones[ID] =
   Thousands[0] + Hundreds[0] + Tens[19] + Ones[0] =
   0 + 0 + 19 + 0= 19
   Basys-3 7-Segment Display Output: **19**

9 Timing Analysis

The purpose of timing analysis is to determine the sequence of events in each of the bus cycles so that we can delimit, among other things, the time available for each of the components to respond to changes. This time is compared to the requirements as specified in the manufacturers' data sheets to determine if they are compatible, and by what margin.

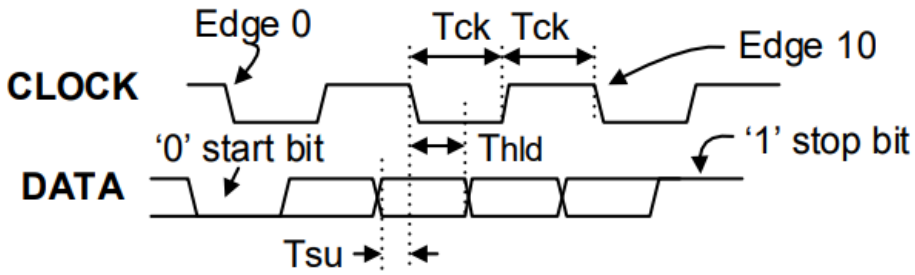
Timing diagrams are a critical method for accurately and unambiguously representing the time-related operations of digital circuits. We will use them to understand and document the correct sequence of operations for microcomputer systems. Timing analysis, using these diagrams, allows the designer to determine safe and reliable limits to the proper operation of the various circuits in the system.

9.1 Timing Analysis Specs

9.1.1 Basys 3 Artix-7 FPGA

Table 17: **IOB High Range (HR) Switching Characteristics *(Cont'd)***

I/O Standard	T_{IOPI}						T_{IOOP}						T_{IOTP}						Units
	Speed Grade						Speed Grade						Speed Grade						
	1.0V				0.95V	0.9V	1.0V				0.95V	0.9V	1.0V				0.95V	0.9V	
	-3	-2/ -2LE	-1	-1Q/ -1M	-1LI	-2LE	-3	-2/ -2LE	-1	-1Q/ -1M	-1LI	-2LE	-3	-2/ -2LE	-1	-1Q/ -1M	-1LI	-2LE	
LVCMOS33_S4	1.26	1.34	1.41	1.52	1.41	1.62	3.80	3.93	4.18	4.18	4.18	4.41	3.82	3.96	4.20	4.20	4.20	4.05	ns
LVCMOS33_S8	1.26	1.34	1.41	1.52	1.41	1.62	3.52	3.65	3.90	3.90	3.90	4.13	3.54	3.68	3.91	3.91	3.91	3.77	ns
LVCMOS33_S12	1.26	1.34	1.41	1.52	1.41	1.62	3.09	3.21	3.46	3.46	3.46	3.69	3.10	3.24	3.48	3.48	3.48	3.33	ns
LVCMOS33_S16	1.26	1.34	1.41	1.52	1.41	1.62	3.40	3.52	3.77	3.78	3.77	4.00	3.42	3.55	3.79	3.79	3.79	3.64	ns
LVCMOS33_F4	1.26	1.34	1.41	1.52	1.41	1.62	3.26	3.38	3.64	3.64	3.64	3.86	3.28	3.41	3.65	3.65	3.65	3.50	ns
LVCMOS33_F8	1.26	1.34	1.41	1.52	1.41	1.62	2.74	2.87	3.12	3.12	3.12	3.35	2.76	2.90	3.13	3.13	3.13	2.99	ns
LVCMOS33_F12	1.26	1.34	1.41	1.52	1.41	1.62	2.56	2.68	2.93	2.93	2.93	3.16	2.57	2.71	2.95	2.95	2.95	2.80	ns
LVCMOS33_F16	1.26	1.34	1.41	1.52	1.41	1.62	2.56	2.68	2.93	3.06	2.93	3.16	2.57	2.71	2.95	3.07	2.95	2.80	ns



Symbol	Parameter	Min	Max
T_{CK}	Clock time	30us	50us
T_{SU}	Data-to-clock setup time	5us	25us
T_{HLD}	Clock-to-data hold time	5us	25us

Figure 8. PS/2 device-to host timing diagram.

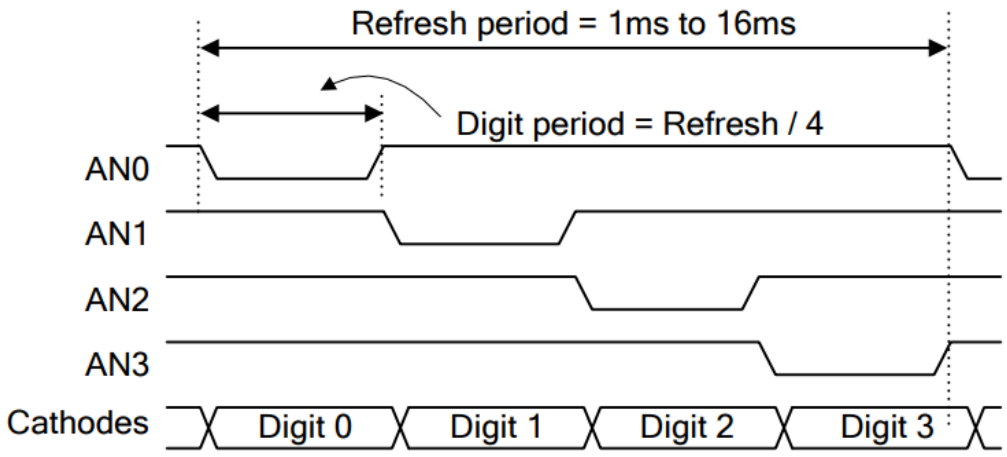


Figure 19. Four digit scanning display controller timing diagram.

9.1.2 Processor

Table 7-40. Useful Timing Parameters on the Memory Side

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tCES	CSn setup time to clock	0
tACS	Address setup time to clock	3
tIACC	Synchronous access time	80
tBACC	Burst access time valid clock to output delay	5,2
tCEZ	Chip-select to High-Impedance	7
tOEZ	Output enable to High-Impedance	7
tAVC	ADVn setup time	6
tAVD	AVDn pulse	6
tACH	Address hold time from clock	3

Table 7-42. AC Characteristics for Asynchronous Read Access

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tCE	Read Access time from CSn low	80
tAAVDS	Address setup time to rising edge of ADVn	3
tAVDP	ADVn low time	6
tCAS	CSn setup time to ADVn	0
tOE	Output enable to output valid	6
tOEZ	Output enable to High-Impedance	7

Table 7-44. AC Characteristics for Asynchronous Single Write (Memory Side)

AC Characteristics on the Memory Side	Description	Duration (ns)
tWC	Write cycle time	60
tAVDP	ADVn low time	6
tWP	Write pulse width	25
tWPH	Write pulse width high	20
tCS	CSn setup time to WEn	3
tCAS	CSn setup time to ADVn	0
tAVSC	ADVn setup time	3

Table 7-43. GPMC Timing Parameters for Asynchronous Read Access

Parameter Name on GPMC side	Formula	Duration (ns)	Number of Clock Cycles (F = 104 MHz)	GPMC Register Configurations
ClkActivationTime	n/a (asynchronous mode)			
AccessTime	round max (tCE)	80	9	ACCESSTIME = 9h
PageBurstAccessTime	n/a (single access)			
RdCycleTime	AccessTime + 1cycle + tOEZ	96, 615	11	RDCYCLETIME = Bh
CsOnTime	tCAS	0	0	CSONTIME = 0
CsReadOffTime	AccessTime + 1 cycle	89, 615	10	CSRDOFFTIME = Ah
AdvOnTime	tAAVDS	3	1	ADVONTIME = 1
AdvRdOffTime	tAAVDS + tAVDP	9	1	ADVRDOFFTIME = 1
OeOnTime	OeOnTime ≥ AdvRdOffTime (multiplexed mode)	-	3, for instance	OEONTIME = 3h
OeOffTime	AccessTime + 1cycle	89, 615	10	OEOFFTIME = Ah

Figure 7-45. Asynchronous Single Read Access (Timing Parameters in Clock Cycles)

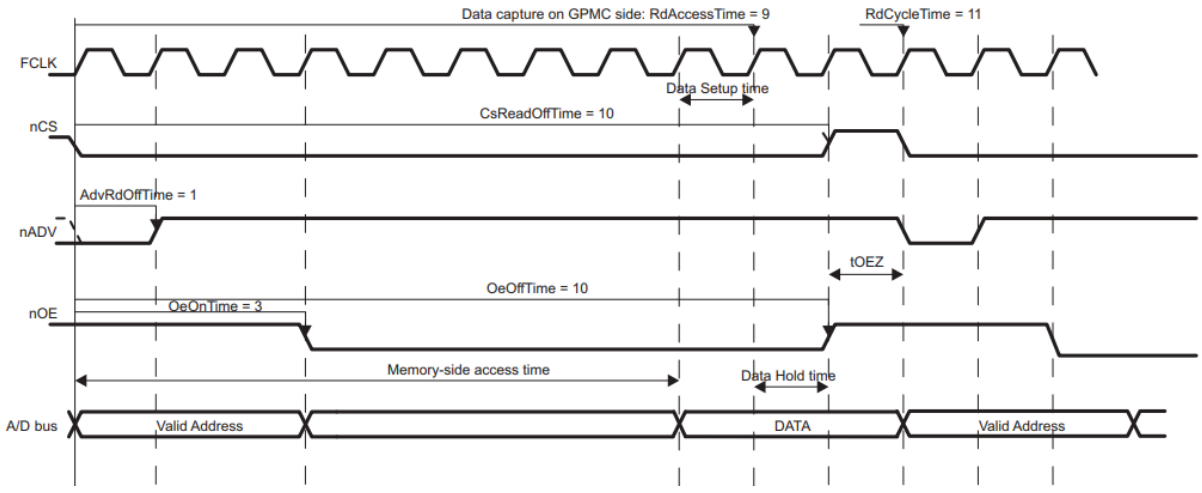


Figure 26-14. GPMC XIP Timings

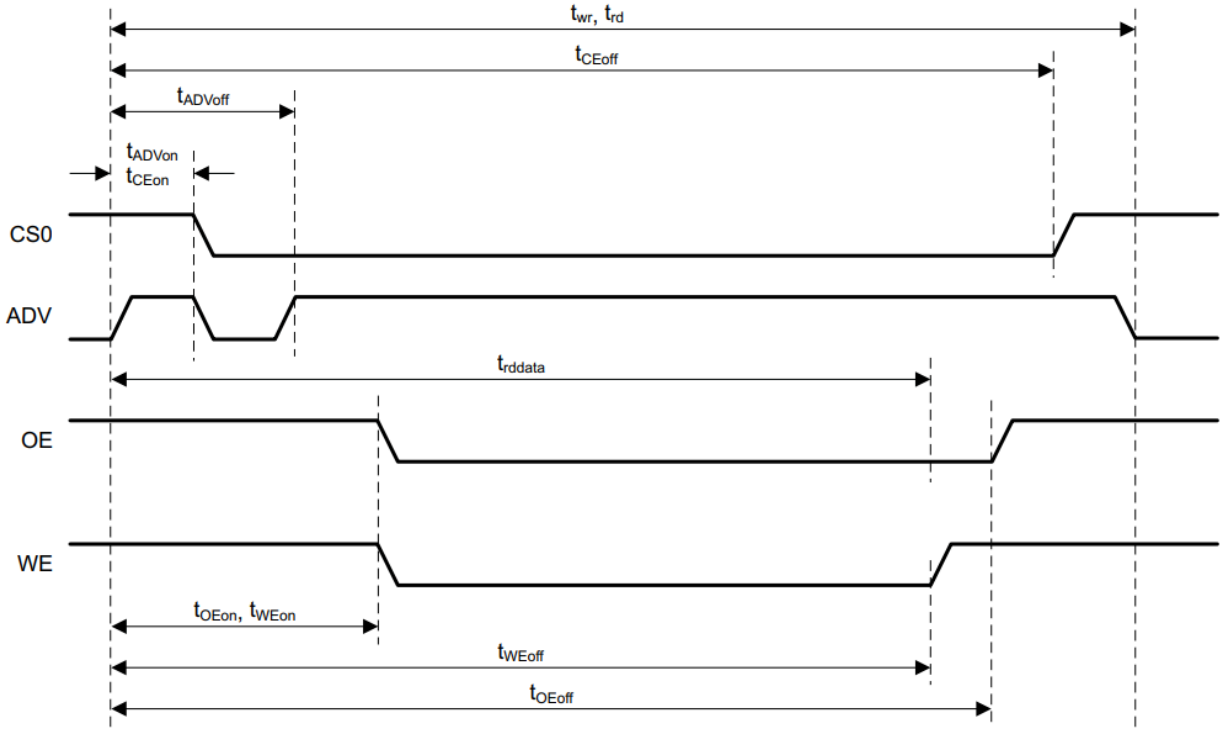


Table 26-8. XIP Timings Parameters

Parameter	Description	Value (clock cycles)
t_{wr}	write cycle period	17
t_{rd}	read cycle period	17
t_{CEon}	CE low time	0
t_{CEoff}	CE high time	16
t_{ADVon}	ADV low time	1
t_{ADVoff}	ADV high time	2
t_{OEon}	OE low time	3
t_{WEon}	WE low time	3
t_{rddata}	data latch time	15
t_{OEoff}	OE high time	16
t_{WEoff}	WE high time	15
$t_{CSEXTRADELAY}$	CS Extra Delay	1/4

9.1.3 SRAM

AC ELECTRICAL CHARACTERISTICS

(1) READ CYCLE

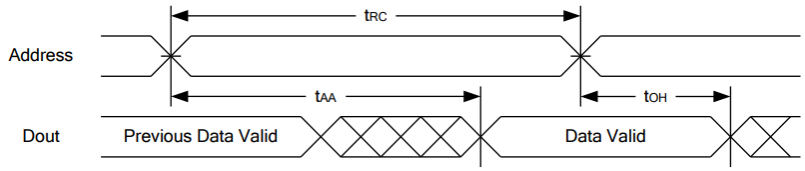
PARAMETER	SYM.	AS7C38098A-10		UNIT
		MIN.	MAX.	
Read Cycle Time	t_{RC}	10	-	ns
Address Access Time	t_{AA}	-	10	ns
Chip Enable Access Time	t_{ACE}	-	10	ns
Output Enable Access Time	t_{OE}	-	4.5	ns
Chip Enable to Output in Low-Z	t_{CLZ}*	2	-	ns
Output Enable to Output in Low-Z	t_{OLZ}*	0	-	ns
Chip Disable to Output in High-Z	t_{CHZ}*	-	4	ns
Output Disable to Output in High-Z	t_{OHZ}*	-	4	ns
Output Hold from Address Change	t_{OH}	2	-	ns
LB#, UB# Access Time	t_{BA}	-	4.5	ns
LB#, UB# to High-Z Output	t_{BHZ}*	-	4	ns
LB#, UB# to Low-Z Output	t_{BLZ}*	0	-	ns

(2) WRITE CYCLE

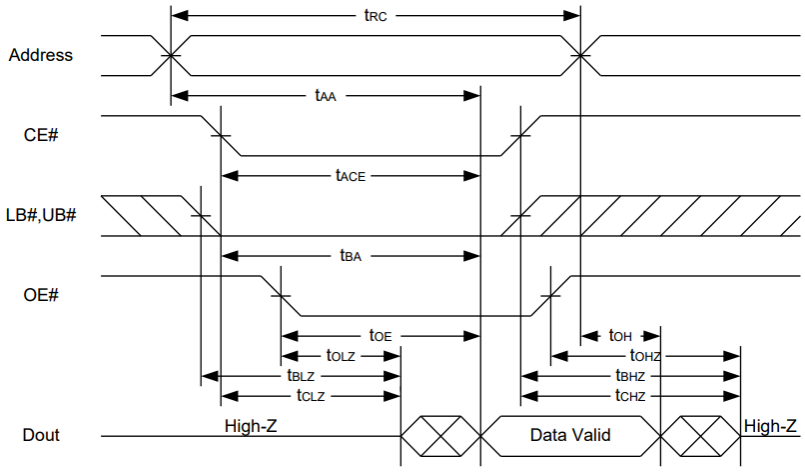
PARAMETER	SYM.	AS7C38098A-10		UNIT
		MIN.	MAX.	
Write Cycle Time	t_{WC}	10	-	ns
Address Valid to End of Write	t_{AW}	8	-	ns
Chip Enable to End of Write	t_{CW}	8	-	ns
Address Set-up Time	t_{AS}	0	-	ns
Write Pulse Width	t_{WP}	8	-	ns
Write Recovery Time	t_{WR}	0	-	ns
Data to Write Time Overlap	t_{DW}	6	-	ns
Data Hold from End of Write Time	t_{DH}	0	-	ns
Output Active from End of Write	t_{OW}*	2	-	ns
Write to Output in High-Z	t_{WHZ}*	-	4	ns
LB#, UB# Valid to End of Write	t_{BW}	8	-	ns

TIMING WAVEFORMS

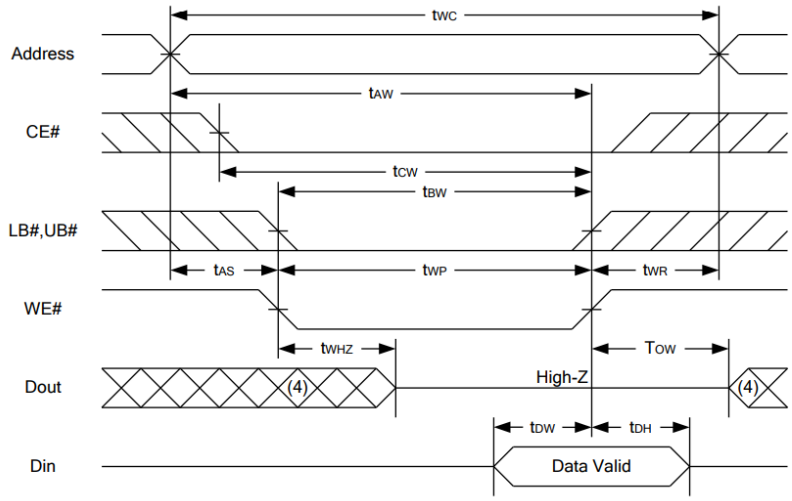
READ CYCLE 1 (Address Controlled) (1,2)



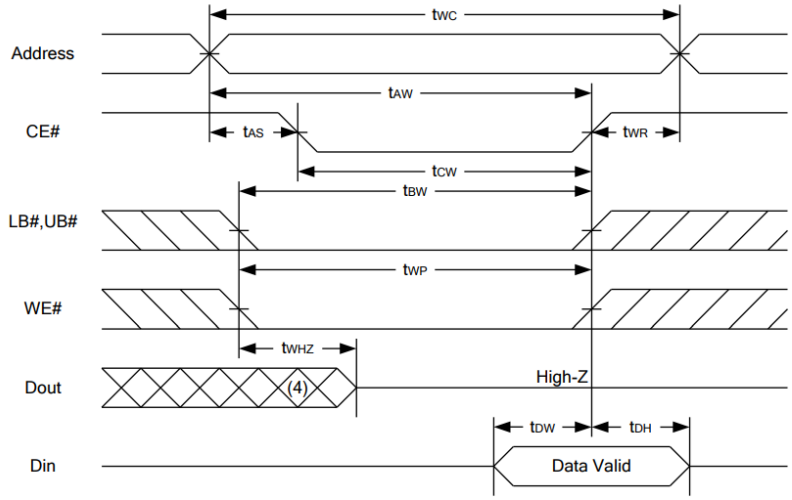
READ CYCLE 2 (CE# and OE# Controlled) (1,3,4,5)



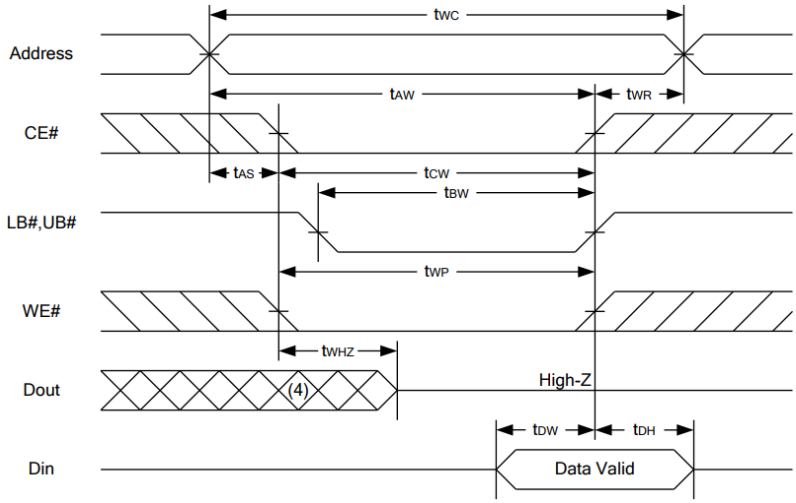
WRITE CYCLE 1 (WE# Controlled) (1,2,3,5,6)



WRITE CYCLE 2 (CE# Controlled) (1,2,5,6)



WRITE CYCLE 3 (LB#,UB# Controlled) (1,2,5,6)



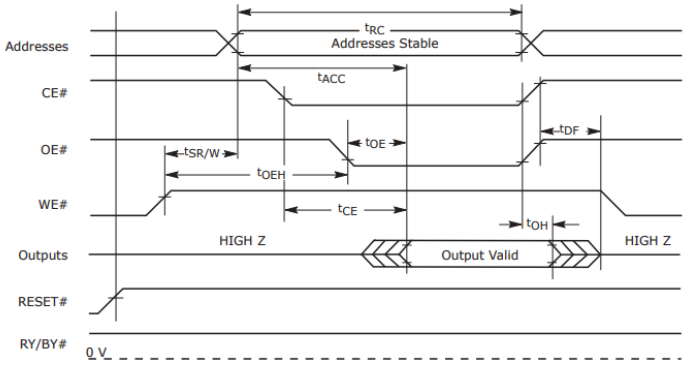
9.1.4 Flash

17. AC Characteristics

17.1 Read Operations

Parameter		Description		Test Setup		Speed Options		Unit
JEDEC	Std					55	70	
t_{AVAV}	t_{RC}	Read Cycle Time (Note 1)			Min	55	70	ns
t_{AVQV}	t_{ACC}	Address to Output Delay		CE# = V_{IL} OE# = V_{IL}	Max	55	70	
t_{ELQV}	t_{CE}	Chip Enable to Output Delay		OE# = V_{IL}	Max	55	70	
t_{GLQV}	t_{OE}	Output Enable to Output Delay			Max	30	30	
t_{EHQZ}	t_{DF}	Chip Enable to Output High Z (Note 1)			Max	16		
t_{GHQZ}	t_{DF}	Output Enable to Output High Z (Note 1)			Max	16		
	$t_{SR/W}$	Latency Between Read and Write Operations			Min	20		
	t_{OEH}	Output Enable Hold Time (Note 1)	Read		Min	0		
			Toggle and Data# Polling		Min	10		
t_{AXQX}	t_{OH}	Output Hold Time From Addresses, CE# or OE#, Whichever Occurs First (Note 1)			Min	0		

Figure 14. Read Operations Timings

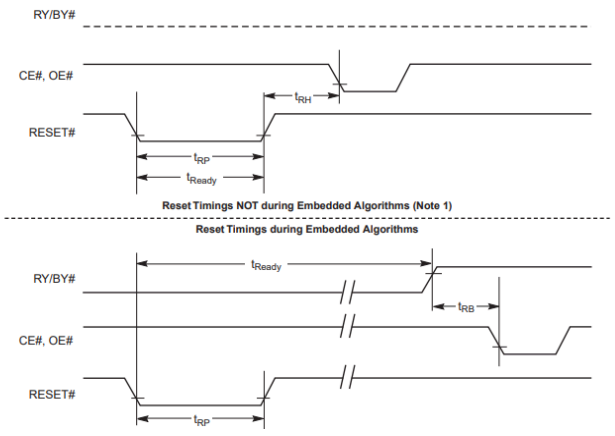


17.2 Hardware Reset (RESET#)

Parameter		Description	Test Setup	All Speed Options	Unit
JEDEC	Std				
	t_{READY}	RESET# Pin Low (During Embedded Algorithms) to Read or Write (See Note)	Max	35	µs
	t_{READY}	RESET# Pin Low (NOT During Embedded Algorithms) to Read or Write (See Note)	Max	500	ns
	t_{RP}	RESET# Pulse Width	Min	500	
	t_{RH}	RESET# High Time Before Read (See Note)		50	
	t_{RPD}	RESET# Low to Standby Mode		35	µs
	t_{RB}	RY/BY# Recovery Time		0	ns

Note
Not 100% tested.

Figure 15. RESET# Timings



17.3 Word/Byte Configuration (BYTE#)

Parameter		Description		Speed Options		Unit
JEDEC	Std			55	70	
	t_{ELFL}/t_{ELFH}	CE# to BYTE# Switching Low or High	Max	5		ns
	t_{FLQZ}	BYTE# Switching Low to Output HIGH Z	Max	16		
	t_{FHQV}	BYTE# Switching High to Output Active	Min	55	70	

Figure 16. BYTE# Timings for Read Operations

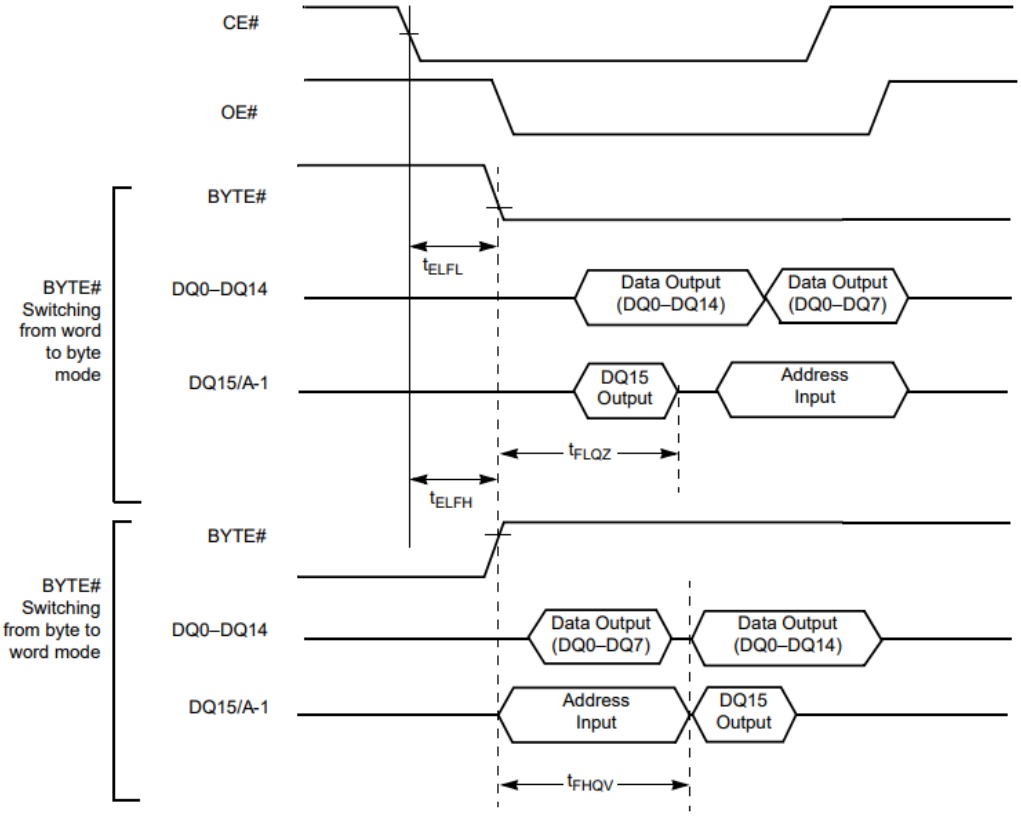
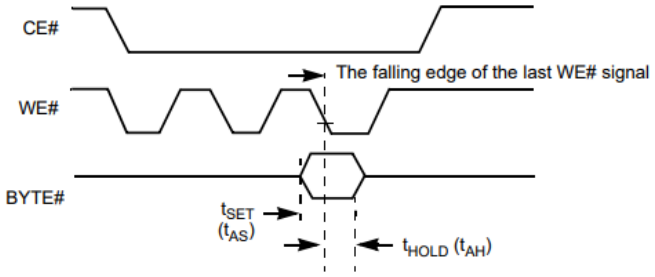


Figure 17. BYTE# Timings for Write Operations



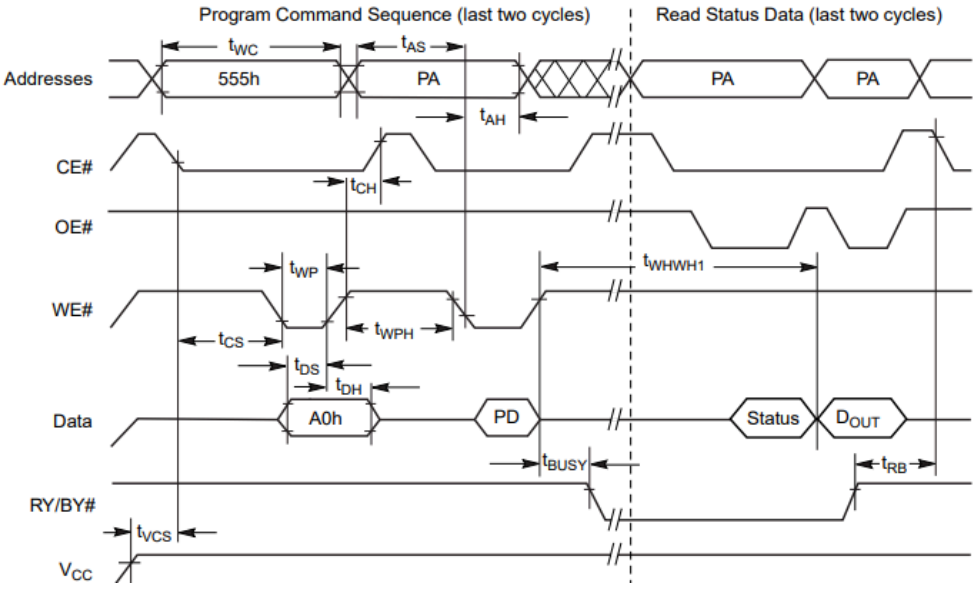
17.4 Erase/Program Operations

Parameter		Description			Speed Options		Unit
JEDEC	Std				55	70	
t_{AVAV}	t_{WC}	Write Cycle Time (Note 1)		Min	55	70	ns
t_{AVWL}	t_{AS}	Address Setup Time		Min	0		ns
t_{WLAX}	t_{AH}	Address Hold Time		Min	45		ns
t_{DVWH}	t_{DS}	Data Setup Time		Min	35	35	ns
t_{WHDX}	t_{DH}	Data Hold Time		Min	0		ns
	t_{OES}	Output Enable Setup Time		Min	0		ns
t_{GHWL}	t_{GHWL}	Read Recovery Time Before Write (OE# High to WE# Low)		Min	0		ns
t_{ELWL}	t_{CS}	CE# Setup Time		Min	0		ns
t_{WHEH}	t_{CH}	CE# Hold Time		Min	0		ns
t_{WLWH}	t_{WP}	Write Pulse Width		Min	35	35	ns
t_{WHWL}	t_{WPH}	Write Pulse Width High		Min	25		ns
	$t_{SR/W}$	Latency Between Read and Write Operations		Min	20		ns
t_{WHWH1}	t_{WHWH1}	Programming Operation (Note 2)	Byte	Typ	6		µs
			Word	Typ	6		
t_{WHWH2}	t_{WHWH2}	Sector Erase Operation (Note 2)		Typ	0.5		sec
	t_{VCS}	V_{CC} Setup Time (Note 1)		Min	50		µs
	t_{RB}	Recovery Time from RY/BY#		Min	0		ns
	t_{BUSY}	Program/Erase Valid to RY/BY# Delay		Max	90		

Notes

1. *Not 100% tested.*
2. *See Erase and Programming Performance on page 44 for more information.*

Figure 18. Program Operation Timings



9.2 Derating Delay Analysis and Calculations

Derating delay is modifying nominal delay values to account for temperature, voltage, and process changes (PVT). These changes affect memory access time, read/write timings, and setup/hold requirements. Tools add a derating factor to delays, increasing or decreasing according to environmental circumstances and manufacturing tolerances. Time derating in the processor, SRAM, and Flash ensures that memory operates within the set time margins in all cases. The spec value of the capacitance load and current output, as well as their actual values and differences, will be utilized to compute the access time spec that should be applied under the given conditions.

9.2.1 Processor Derating Delay

Delta T = (delta V * delta C) / (delta I)

$$\Delta T = \frac{\Delta V * \Delta C}{\Delta I}$$

Spec Values	Actual Values	Difference
$C_L = 3\,pF$	$30\,pF$	$27\,pF\ (\Delta C)$
$I_o = 1\,mA$	$250\,mA$	$249\,mA\ (\Delta I)$

To calculate the change in voltage, we subtract the values of maximum input low voltage, Vilmax, and minimum input low voltage, Vilmin, as specified in the datasheet. Performing the calculation will result in,

$$V_{ilmax} = 8\,V\,, V_{ilmin} = 0\,V$$

$$\Delta V = V_{ilmax} - V_{ilmin}$$

$$\Delta V = 8 - 0$$

$$\Delta V = 8\,V$$

Using the equation above, we can calculate the change in time delay ΔT. Given the values above, performing the calculation will result in,

$$\Delta T = \frac{8\,V * 27\,pF}{249\,mA}$$

$$\Delta T = 0.867\,ns$$

Therefore, 0.867 ns should be added to all the output delay specs for the driving device. The access time used should be,

$$T_{aa}(actual) = T_{aa}(spec) + \Delta T = 80\,ns + 0.867\,ns = \underline{\mathbf{80.867\ ns}}$$

9.2.2 SRAM Derating Delay

Delta T = (delta V * delta C) / (delta I)

$$\Delta T = \frac{\Delta V * \Delta C}{\Delta I}$$

Spec Values	Actual Values	Difference
$C_L = 5\,pF$	$30\,pF$	$25\,pF\ (\Delta C)$
$I_o = -1/2\,mA$	$50\,mA$	$50.5\,mA\ (\Delta I)$

To calculate the change in voltage, we subtract the values of maximum input low voltage, Vilmax, and minimum input low voltage, Vilmin, as specified in the datasheet. Performing the calculation will result in,

$$V_{ilmax} = 0.8\,V\,, V_{ilmin} = -0.3\,V$$

$$\Delta V = V_{ilmax} - V_{ilmin}$$

$$\Delta V = 0.8 - (-0.3)$$

$$\Delta V = 1.1\,V$$

Using the equation above, we can calculate the change in time delay ΔT. Given the values above, performing the calculation will result in,

$$\Delta T = \frac{1.1\,V * 25\,pF}{50.5\,mA}$$

$$\Delta T = 0.545\,ns$$

Therefore, 0.545 ns should be added to all the output delay specs for the driving device. The access time used should be,

$$T_{aa}(actual) = T_{aa}(spec) + \Delta T = 10\,ns + 0.545\,ns = \underline{\mathbf{10.545\ ns}}$$

9.2.3 Flash Derating Delay

Delta T = (delta V * delta C) / (delta I)

$$\Delta T = \frac{\Delta V * \Delta C}{\Delta I}$$

Spec Values	Actual Values	Difference
$C_L = 5.5\,pF$	$30\,pF$	$24.5\,pF\ (\Delta C)$
$I_o = -1/2\,mA$	$200\,mA$	$200.5\,mA\ (\Delta I)$

To calculate the change in voltage, we subtract the values of maximum input low voltage, Vilmax, and minimum input low voltage, Vilmin, as specified in the datasheet. Performing the calculation will result in,

$$V_{ilmax} = 0.8\,V, V_{ilmin} = -0.1\,V$$

$$\Delta V = V_{ilmax} - V_{ilmin}$$

$$\Delta V = 0.8 - (-0.1)$$

$$\Delta V = 0.9\,V$$

Using the equation above, we can calculate the change in time delay ΔT. Given the values above, performing the calculation will result in,

$$\Delta T = \frac{0.9\,V * 24.5\,pF}{200.5\,mA}$$

$$\Delta T = 0.110\,ns$$

Therefore, 0.110 ns should be added to all the output delay specs for the driving device. The access time used should be,

$$T_{aa}(actual) = T_{aa}(spec) + \Delta T = 55\,nS + 0.110\,nS = \underline{\mathbf{55.110\ ns}}$$

9.3 Signal Delays for Complete Power-Off

9.3.1 SRAM

Signal	Description	Power Off Delay	Unit
CE0	Chip Enable 0	**8.447**	ns
CE1	Chip Enable 1	**8.982**	ns
OE0	Output Enable 0	**8.458**	ns
OE1	Output Enable 1	**7.830**	ns
WE0	Write Enable 0	**9.195**	ns
WE1	Write Enable 1	**8.749**	ns

9.3.2 Flash

Signal	Description	Power Off Delay	Unit
CS0	Chip Select 0	**6.439**	ns
CS1	Chip Select 1	**6.303**	ns
WP	Write Protect	**6.152**	ns

9.4 SRAM Signal Timing Delays for Complete Power On/Off

9.4.1 SRAM

Signal	Description	Power-Off Delay	Derating Delay	Power-On/Off Delay	Unit
CE0	Chip Enable 0	8.447	10.545	**18.992**	ns
CE1	Chip Enable 1	8.982	10.545	**19.527**	ns
OE0	Output Enable 0	8.458	10.545	**19.003**	ns
OE1	Output Enable 1	7.830	10.545	**18.375**	ns
WE0	Write Enable 0	9.195	10.545	**19.740**	ns
WE1	Write Enable 1	8.749	10.545	**19.294**	ns

9.4.2 Flash

Signal	Description	Power-Off Delay	Derating Delay	Power-On/Off Delay	Unit
CS0	Chip Select 0	6.439	55.110	**61.549**	ns
CS1	Chip Select 1	6.303	55.110	**61.413**	ns
WP	Write Protect	6.152	55.110	**61.262**	ns

9.5 Timing Analysis Between Connected Signals

#	Signal Path	Calculated Time	Time	Unit
1	CPU ⟶ FPGA	1.544 + 0.478	2.032	ns
2	CPU ⟶ FPGA ⟶ Idle Time ⟶ CE0 ⟶ OE0 ⟶ tCEZ ⟶ tOEZ ***Note:** Idle Time = max(tCEZ, tOEZ)	1.544 + 0.478 + 7 + 18.992 + 19.003 + 7 + 7	61.017	ns
3	CPU ⟶ FPGA ⟶ Idle Time ⟶ CE0 ⟶ WE0 ⟶ tCEZ ⟶ tCW ***Note:** Idle Time = max(tCEZ, tCW)	1.544 + 0.478 + 8 + 18.992 + 19.740 + 7 + 8	63.754	ns
4	CPU ⟶ FPGA ⟶ Idle Time ⟶ CE1 ⟶ OE1 ⟶ tCEZ ⟶ tOEZ ***Note:** Idle Time = max(tCEZ, tOEZ)	1.544 + 0.478 + 7 + 19.527 + 18.375 + 7 + 7	60.924	ns
5	CPU ⟶ FPGA ⟶ Idle Time ⟶ CE1 ⟶ WE1 ⟶ tCEZ ⟶ tCW ***Note:** Idle Time = max(tCEZ, tCW)	1.544 + 0.478 + 8 + 19.527 + 19.294 + 7 + 8	63.843	ns
6	CPU ⟶ FPGA ⟶ Idle Time ⟶ CS0 ⟶ WP ⟶ tCS ⟶ tWP ***Note:** Idle Time = max(tCS, tWP)	1.544 + 0.478 + 25 + 61.549 + 61.262 + 3 + 25	177.833	ns
7	CPU ⟶ FPGA ⟶ Idle Time ⟶ CS1 ⟶ WP ⟶ tCS ⟶ tWP ***Note:** Idle Time = max(tCS, tWP)	1.544 + 0.478 + 25 + 61.413 + 61.262 + 3 + 25	177.697	ns

- ****Note****: The signal paths above are **FREE FROM CONTENTION** since the Idle time and deasserting time are calculated along with each path.
- Longest Path In The System According to the Timing Analysis Calculations above:
  - **CPU ⟶ FPGA ⟶ Idle Time ⟶ CS1 ⟶ WP ⟶ tCS ⟶ tWP = 177.833 ns**
- Last Signal to Go Active According to the Timing Analysis Calculations above:
  - **WP (Write Protect)**

10 Verilog Modules and Testbench Simulations

10.1 Design Sources

10.1.1 Top.v

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
//////////////////////////////////////////////////////////////////////////////////

module Top #(parameter N = 4)   // Default width = 4
            (input clk,                                         // System Clock
             input cpu_clk,                                     // CPU Clock
             input reset,                                       // Avtive High reset
             input nRESET,                                      // Active Low reset
             output wire signed [7:0] out,                      // Declare output as
signed for ---> read_memory.v module <---
             output wire [N-1:0] lfsr_4bit,                     // Declare lfsr_4bit for
---> RandomNoiseLFSR.v module <---
             output wire [N*2-1:0] lfsr_8bit,                   // Declare lfsr_8bit for
---> RandomNoiseLFSR.v module <---
             output wire [N*8-1:0] lfsr_32bit,                  // Declare lfsr_32bit for
---> RandomNoiseLFSR.v module <---
             input [N*8-1:0] address,                           // 32 bit input addrerss
for ---> ProgramAddressMap and DataMemoryAddress modules <---
             input read,                                        // 1 bit input read signal
for ---> DataMemoryAddress module <---
             input write,                                       // 1 bit input write
signal for ---> DataMemoryAddress module <---
             output wire [N*4-1:0] Control_Module,              // 16 bit output Control
Module for ---> DataMemoryAddress module <---
             output wire [N*4-1:0] UART1,                       // 16 bit output UART1
for ---> DataMemoryAddress module <---
             output wire CE0,                                   // 1 bit output first SRAM
Chip Enable for ---> DataMemoryAddress module <---
             output wire CE1,                                   // 1 bit output second
SRAM Chip Enable for ---> DataMemoryAddress module <---
             output wire OE0,                                   // 1 bit output first SRAM
Output Enable for ---> DataMemoryAddress module <---
             output wire OE1,                                   // 1 bit output second
SRAM Output Enable for ---> DataMemoryAddress module <---
             output wire WE0,                                   // 1 bit output first SRAM
Write Enable for ---> DataMemoryAddress module <---
             output wire WE1,                                   // 1 bit output second
SRAM Write Enable for ---> DataMemoryAddress module <---
```

```
            output wire CS0,                                    // 1 bit output first
Flash CS0 for ---> ProgramAddressMap module <---
            output wire CS1,                                    // 1 bit output second
Flash CS1 for ---> ProgramAddressMap module <---
            output wire WP,                                     // 1 bit output Chip
Enable for ---> ProgramAddressMap module <---
            input rx,                                           // Serial data input bit
(1-bit) for ---> UART_Receiver module <---
            output wire rx_busy,                                // Signal to indicate
whether transmission is -> busy <- or not (tx_busy = 1: transmitter is busy and
currently involved in transmitting data | tx_done = 0: transmitter is not busy or has
completed its transmission) for ---> UART_Receiver module <---
            output wire [7:0] data,                             // 8-bit data to transmit
for ---> UART_Receiver module <---
            output wire [N*3+1:0] Seven_Segment_Display,   // 14 bit output testing
for seven segment display for ---> DigitTo7SegmentDisplay.v module <---
            input [31:0] noisy_data,                            // 32-bit noisy input data
for  ---> Low_Pass_Filter module <---
            output wire [31:0] filtered_data                 // 32-bit filtered output
data for ---> Low_Pass_Filter module <---
            );

    // Instantiation of the read_memory module
    read_memory rm (.clk(clk),
                    .out(out)
                    );

    // Instantiation of the RandomNoiseLFSR module
    RandomNoiseLFSR #(.N(N)) lfsr1 (.clk(clk),
                                    .reset(reset),
                                    .lfsr(lfsr_4bit)
                                   );

    RandomNoiseLFSR #(.N(N*2)) lfsr2 (.clk(clk),
                                    .reset(reset),
                                    .lfsr(lfsr_8bit)
                                    );

    RandomNoiseLFSR #(.N(N*8)) lfsr3 (.clk(clk),
                                     .reset(reset),
                                     .lfsr(lfsr_32bit)
                                     );

    // Instantiation of the ProgramAddressMap module
    ProgramAddressMap #(.N(N*8)) dataAdd (.clk(clk),
                                          .nRESET(nRESET),
                                          .address(address),
```

```
                                        .CS0(CS0),
                                        .CS1(CS1),
                                        .WP(WP)
                                       );

    // Instantiation of the DataMemoryAddress module
    DataMemoryAddress #(.N(N*8)) progAdd (.clk(clk),
                                        .nRESET(nRESET),
                                        .address(address),
                                        .read(read),
                                        .write(write),
                                        .Control_Module(Control_Module),
                                        .UART1(UART1),
                                        .CE0(CE0),
                                        .CE1(CE1),
                                        .OE0(OE0),
                                        .OE1(OE1),
                                        .WE0(WE0),
                                        .WE1(WE1)
                                       );

    // Instantiation of the UART_Receiver module
    UART_Receiver Receiver (.clk(clk),
                            .rx(rx),
                            .data(data),
                            .rx_busy(rx_busy)
                            );

    // Instantiation of the DigitTo7SegmentDisplay module
    wire [7:0] ID = {data[0], data[1], data[2], data[3], data[4], data[5], data[6],
data[7]};

    DigitTo7SegmentDisplay display (.clk(clk),
                                    .reset(reset),
                                    .ID(ID[5:0]),
                                    .Seven_Segment_Display(Seven_Segment_Display)
                                    );

    // Instantiate the Low_Pass_Filter module
    Low_Pass_Filter filter (.clk(clk),
                            .reset(reset),
                            .noisy_data(noisy_data),
                            .filtered_data(filtered_data)
                            );

endmodule
```

10.1.2 UART_Receiver.v

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
//////////////////////////////////////////////////////////////////////////////////

module UART_Receiver (input clk,           // Generated baud clock from
UART_Baud_Rate_Gen module
                      input rx,                 // Serial data input bit (1-bit)
                      output reg [7:0] data,    // 8-bit data to transmit
                      output reg rx_busy        // Signal to indicate whether
transmission is -> busy <- or not (tx_busy = 1: transmitter is busy and currently
involved in transmitting data | tx_done = 0: transmitter is not busy or has completed
its transmission)
                      );

    // State Machine States
    localparam [1:0] IDLE = 2'b00,
                     START = 2'b01,
                     DATA = 2'b10,
                     END = 2'b11;

    localparam clk_per_bit = 2;

    reg [1:0] state = IDLE;        // Current_state of the machine and next_state of
the machine
    reg [2:0] bit_index = 0;       // The index of the bit to be transmitted. Since we
are 8-bis of data, this bit_index is capable of transmitting all 8 since it is 3-bits,
which can range from 0 to 7
    reg [7:0] rx_data;             // Data to be transmitted
    reg [15:0] clk_counter = 0;

    initial begin
        rx_data <= 8'b0;
    end

    always @(posedge clk) begin
        case (state)
            IDLE: begin
                rx_busy <= 0;
                clk_counter <= 0;

                if (rx == 1'b0) begin
                    state <= START;
                end
```

```
            end

            START: begin
                if (clk_counter == (clk_per_bit - 1) / 2) begin
                    if (rx == 1'b0) begin
                        state <= DATA;
                        rx_busy <= 1;
                        clk_counter <= 0;
                    end
                    else begin
                        state <= IDLE;
                    end
                end
                else begin
                    clk_counter <= clk_counter + 1;
                end
            end

            DATA: begin
                rx_data[bit_index] <= rx;
                if (bit_index < 7) begin
                    bit_index <= bit_index + 1;
                end
                else begin
                    state <= END;
                    clk_counter <= 0;
                    bit_index <= 0;
                end
            end

            END: begin
                if (rx == 1'b1) begin
                    data <= rx_data;
                    state <= IDLE;
                    rx_busy <= 0;
                end
                else begin
                    state <= IDLE;
                end
            end

            // Default state
            default: state <= IDLE;

        endcase
    end
endmodule
```

10.1.3 DigitTo7SegmentDisplay.v

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
//////////////////////////////////////////////////////////////////////////////////


module DigitTo7SegmentDisplay (input clk,
                               input reset,
                               input [5:0] ID,
                               output reg [13:0] Seven_Segment_Display
                               );

    // State Machine States
    localparam [1:0] IDLE = 2'b00,
                     START = 2'b01,
                     LOGIC = 2'b10,
                     DONE = 2'b11;

    reg [1:0] state = IDLE;        // Current state of the machine

    reg [3:0] ones;               // 4-bit wide
    reg [6:0] tens;               // 7-bit wide
    reg [9:0] hundreds;           // 10-bit wide
    reg [13:0] thousands;         // 14-bit wide

    // Number Groups
    reg [3:0] Ones [1:9];              // 10 entries, each 4-bit wide
    reg [6:0] Tens [10:27];            // 10 entries, each 7-bit wide
    reg [9:0] Hundreds [28:36];        // 10 entries, each 10-bit wide
    reg [13:0] Thousands [38:45];      // 10 entries, each 14-bit wide
    reg DoSt [46:47];                  // done = 46, Start = 47

    initial begin
        Ones[1] = 1;
        Ones[2] = 2;
        Ones[3] = 3;
        Ones[4] = 4;
        Ones[5] = 5;
        Ones[6] = 6;
        Ones[7] = 7;
        Ones[8] = 8;
        Ones[9] = 9;
        Tens[10] = 10;
        Tens[11] = 11;
```

```
        Tens[12] = 12;
        Tens[13] = 13;
        Tens[14] = 14;
        Tens[15] = 15;
        Tens[16] = 16;
        Tens[17] = 17;
        Tens[18] = 18;
        Tens[19] = 19;
        Tens[20] = 20;
        Tens[21] = 30;
        Tens[22] = 40;
        Tens[23] = 50;
        Tens[24] = 60;
        Tens[25] = 70;
        Tens[26] = 80;
        Tens[27] = 90;
        Hundreds[28] = 100;
        Hundreds[29] = 200;
        Hundreds[30] = 300;
        Hundreds[31] = 400;
        Hundreds[32] = 500;
        Hundreds[33] = 600;
        Hundreds[34] = 700;
        Hundreds[35] = 800;
        Hundreds[36] = 900;
        Thousands[37] = 1000;
        Thousands[38] = 2000;
        Thousands[39] = 3000;
        Thousands[40] = 4000;
        Thousands[41] = 5000;
        Thousands[42] = 6000;
        Thousands[43] = 7000;
        Thousands[44] = 8000;
        Thousands[45] = 9000;
        DoSt[46] = 0;
        DoSt[47] = 1;
    end

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            Seven_Segment_Display <= 0;
        end
        else begin
            case (state)
                IDLE: begin
                    if (ID > 0) begin
                        state <= START;
```

```
        end
    end

    START: begin
        if (ID >= 1 && ID <= 46) begin
            ones <= 0;
            tens <= 0;
            hundreds <= 0;
            thousands <= 0;
            state <= LOGIC;
        end
        else begin
            state <= IDLE;
        end
    end

    LOGIC: begin
        if (ID >= 1 && ID <= 9) begin
            ones <= Ones[ID];
        end
        else if (ID >= 10 && ID <= 27) begin
            tens <= Tens[ID];
        end
        else if (ID >= 28 && ID <= 36) begin
            hundreds <= Hundreds[ID];
        end
        else if (ID >= 37 && ID <= 45) begin
            thousands <= Thousands[ID];
        end
        else begin
            if (ID == 46) begin
                state <= DONE;
            end
            else begin
                state <= LOGIC;
            end
        end
    end

    DONE: begin
        Seven_Segment_Display <= thousands + hundreds + tens + ones;

        if (ID == 47) begin
            state <= START;
        end
        else begin
            state <= IDLE;
```

```
                    end
                end

                default: state <= IDLE;
            endcase
        end
    end

endmodule
```

10.1.4 ProgramAddressMap.v

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
//////////////////////////////////////////////////////////////////////////////////

module ProgramAddressMap #(parameter N = 32)                    // 32 bit address
                          (input clk,
                           input nRESET,                        // Active Low reset
                           input [N-1:0] address,               // 32 bit input
addrerss
                           output reg CS0,                      // 1 bit output first
Flash Chip Select
                           output reg CS1,                      // 1 bit output second
Flash Chip Select
                           output reg WP                        // 1 bit output Write
Protect
                           );

    wire [19:0] upper_bits;

    assign upper_bits = address[31:12];

    always @(posedge clk or negedge nRESET) begin
        if (!nRESET) begin
            // Inactive when outside Flash address range
            CS0 <= 1;
            CS1 <= 1;
            WP <= 1;
        end
        else begin
            case (upper_bits)
                20'b0000_0000_0000_0000_0000: begin
                    CS0 <= 0;             // Active low first Flash chip select
                    CS1 <= 1;
                    WP <= 0;              // Active low Write Protect
                end

                20'b0000_1000_0000_0000_0000: begin
                    CS0 <= 1;
```

```
                    CS1 <= 0;             // Active low second Flash chip select
                    WP <= 0;              // Active low Write Protect
                end

                20'b0010_0000_0000_0000_0000: begin
                    // Inactive when outside Flash address range
                    CS0 <= 1;
                    CS1 <= 1;
                    WP <= 1;
                end
            endcase
        end
    end

endmodule
```

10.1.5 DataMemoryAddress.v

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
//////////////////////////////////////////////////////////////////////////////////

module DataMemoryAddress #(parameter N = 32)            // 32 bit address
                          (input clk,
                           input nRESET,                    // Active Low reset
                           input [N-1:0] address,           // 32 bit input addrerss
                           input read,                      // 1 bit input read signal
                           input write,                     // 1 bit input write
signal
                           output reg Control_Module,       // 1 bit output Input Port
                           output reg UART1,                // 1 bit output Output
Port
                           output reg CE0,                  // 1 bit output first SRAM
Chip Enable
                           output reg CE1,                  // 1 bit output second
SRAM Chip Enable
                           output reg OE0,                  // 1 bit output first SRAM
Output Enable
                           output reg OE1,                  // 1 bit output second
SRAM Output Enable
                           output reg WE0,                  // 1 bit output first SRAM
Write Enable
                           output reg WE1                   // 1 bit output second
SRAM Write Enable
                           );

    wire [19:0] upper_bits;

    assign upper_bits = address[31:12];

    always @(posedge clk or negedge nRESET) begin
        if (!nRESET) begin
            Control_Module <= 1;
            UART1 <= 1;
            CE0 <= 1;
            CE1 <= 1;
            OE0 <= 1;
            OE1 <= 1;
            WE0 <= 1;
            WE1 <= 1;
        end
        else begin
            case (upper_bits)
```

```
                20'b0001_0000_0000_0000_0000: begin
                    CE0 <= 0;
                    CE1 <= 1;

                    if (read) begin
                        OE0 <= 0;
                        WE0 <= 1;
                    end
                    else if (write) begin
                        WE0 <= 0;
                        OE0 <= 1;
                    end
                    else begin
                        OE0 <= 1;
                        WE0 <= 1;
                    end
                end

                20'b0001_0100_0000_0000_0000: begin
                    CE1 <= 0;
                    CE0 <= 1;

                    if (read) begin
                        OE1 <= 0;
                        WE1 <= 1;
                    end
                    else if (write) begin
                        WE1 <= 0;
                        OE1 <= 1;
                    end
                    else begin
                        OE1 <= 1;
                        WE1 <= 1;
                    end
                end

                20'b0010_0000_0000_0000_0000: begin
                    Control_Module <= 1;
                    UART1 <= 1;
                    CE0 <= 1;
                    CE1 <= 1;
                    OE0 <= 1;
                    OE1 <= 1;
                    WE0 <= 1;
                    WE1 <= 1;
                end

                20'b0100_0100_1110_0001_0000: begin
                    Control_Module <= 0;
```

```
                end

                20'b0100_0100_1110_0001_0010: begin
                    Control_Module <= 1;
                    UART1 <= 1;
                    CE0 <= 1;
                    CE1 <= 1;
                    OE0 <= 1;
                    OE1 <= 1;
                    WE0 <= 1;
                    WE1 <= 1;
                end

                20'b0100_1000_0000_0010_0010: begin
                    UART1 <= 0;
                end

                20'b0100_1000_0000_0010_0011: begin
                    Control_Module <= 1;
                    UART1 <= 1;
                    CE0 <= 1;
                    CE1 <= 1;
                    OE0 <= 1;
                    OE1 <= 1;
                    WE0 <= 1;
                    WE1 <= 1;
                end
            endcase
        end
    end

endmodule
```

10.1.6 RandomNoiseLFSR.v

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
//////////////////////////////////////////////////////////////////////////////////

module RandomNoiseLFSR #(parameter N = 4)   // Default width = 4
                        (input clk,
                         input reset,
                         output reg [N-1:0] lfsr
                         );

    wire xor_first_bit;

    // Tap selection based on parameter N
    generate
        if (N == 4) begin
            // For a 4-bit LFSR
            assign xor_first_bit = (lfsr ^ (lfsr >> 1)) & 1; // Taps: 4th and 3rd bits
        end
        else if (N == 8) begin
            // For an 8-bit LFSR
            assign xor_first_bit = (lfsr ^ (lfsr >> 8) ^ (lfsr >> 6) ^ (lfsr >> 5) ^
(lfsr >> 4)) & 1;  // Taps: 8th, 6th, 5th, 4th bits
        end
        else if (N == 32) begin
            // For an 8-bit LFSR
            assign xor_first_bit = (lfsr ^ (lfsr >> 32) ^ (lfsr >> 30) ^ (lfsr >> 26)
^ (lfsr >> 25)) & 1;  // Taps: 32nd, 30th, 26th, 25th bits
        end
        else begin
            assign xor_first_bit = 1'b0;
        end
    endgenerate

    always @(posedge clk or posedge reset) begin
        if (reset)
            if (N == 4) begin
                lfsr <= (1 << 4) - 1;            // Initial value when N = 4
            end
```

```
            else if (N == 8) begin
                lfsr <= (1 << 8) - 1;   // Initial value when N = 8
            end
            else if (N == 32) begin
                lfsr <= (1 << 32) - 1;
            end
            else begin
                lfsr <= {N{1'b0}};      // Default reset value for other N
            end
        else begin
            // Shift operation based on N
            if (N == 4) begin
                lfsr <= {(lfsr >> 1) | (xor_first_bit << 3)};   // Shift left and
insert xor_first_bit
            end
            else if (N == 8) begin
                lfsr <= {(lfsr >> 1) | (xor_first_bit << 7)};   // Shift left and
insert xor_first_bit
            end
            else if (N == 32) begin
                lfsr <= {(lfsr >> 1) | (xor_first_bit << 31)};   // Shift left and
insert xor_first_bit
            end
        end
    end

endmodule
```

10.1.7 read_memory.v

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
//////////////////////////////////////////////////////////////////////////////////

module read_memory (input clk,
                    output reg signed [7:0] out         // Declare output as signed
                    );

    reg signed [7:0] memory [0:251];        // Memory array to hold signed sine wave
values
    reg [7:0] i = 0;                        // Memory index pointer

    initial begin
        $readmemb("SineWave.mem", memory);      // Load memory from file
    end

    always @(posedge clk) begin
        out <= memory[i];           // Output the current memory value
        i <= (i == 251) ? 0 : i + 1;            // Wrap around index
    end

endmodule
```

10.1.8 Low_Pass_Filter.v

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
//////////////////////////////////////////////////////////////////////////////////


module Low_Pass_Filter(input clk,
                       input reset,
                       input [31:0] noisy_data,             // 32-bit noisy input data
                       output reg [31:0] filtered_data      // 32-bit filtered output
data
                       );

    parameter N = 16;       // Moving average window size (16 samples)

    // Register to store the last N samples (32-bit data)
    reg [31:0] data_window [0:N-1];         // Array to hold samples

    // Sum of the last N samples for averaging (48-bit to avoid overflow)
    reg [47:0] sum;  // 48-bit sum to handle the large 32-bit values

    // Counter to track how many samples we've seen
    reg [4:0] count;  // 5-bit counter to track up to N (16 in this case)

    integer i;

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            // Reset the sum, counter, and the data window
            sum <= 0;
            count <= 0;
            for (i = 0; i < N; i = i + 1) begin
                data_window[i] <= 32'd0;
            end
            filtered_data <= 32'd0;
        end else begin
            // Shift the data window by one and insert the new sample at the beginning
            sum <= sum - data_window[count] + noisy_data;  // Update sum (subtract
old, add new)
```

```
            data_window[count] <= noisy_data;  // Store the new data sample

            // Increment counter and wrap around if we exceed N samples
            if (count < N-1) begin
                count <= count + 1;
            end else begin
                count <= 0;
            end

            // Compute the filtered output (moving average)
            if (count == N-1) begin
                filtered_data <= sum / N;  // Calculate average after N samples
            end
        end
    end

endmodule
```

10.2 Simulation Sources

10.2.1 system_testbench.v

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
//////////////////////////////////////////////////////////////////////////////////


module system_testbench;

    // Parameters
    parameter N = 4;

    reg clk;
    reg reset;
    reg nRESET;                         // Active Low reset
    wire signed [7:0] out;              // Sine wave output from the --->
read_memory.v module <---
    wire [N-1:0] lfsr_4bit;             // 4 bit output from ---> RandomNoiseLFSR.v
module <---
    wire [N*2-1:0] lfsr_8bit;           // 8 bit output from ---> RandomNoiseLFSR.v
module <---
    wire [N*8-1:0] lfsr_32bit;          // 32 bit output from ---> RandomNoiseLFSR.v
module <---
    reg [N*8-1:0] address;              // 32 bit input addrerss for --->
ProgramAddressMap and DataMemoryAddress modules <---
    reg read;                           // 1 bit input read signal for --->
DataMemoryAddress module <---
    reg write;                          // 1 bit input write signal for --->
DataMemoryAddress module <---
    wire Control_Module;                // 1 bit output Input Port for --->
DataMemoryAddress module <---
    wire UART1;                         // 1 bit output Output Port for --->
DataMemoryAddress module <---
    wire CE0;                           // 1 bit output first SRAM Chip Enable for
---> DataMemoryAddress module <---
    wire CE1;                           // 1 bit output second SRAM Chip Enable for
---> DataMemoryAddress module <---
    wire OE0;                           // 1 bit output first SRAM Output Enable for
---> DataMemoryAddress module <---
```

```
    wire OE1;                           // 1 bit output second SRAM Output Enable for
---> DataMemoryAddress module <---
    wire WE0;                           // 1 bit output first SRAM Write Enable for
---> DataMemoryAddress module <---
    wire WE1;                           // 1 bit output second SRAM Write Enable for
---> DataMemoryAddress module <---
    wire CS0;                           // 1 bit output first Flash CS0 for --->
ProgramAddressMap module <---
    wire CS1;                           // 1 bit output second Flash CS1 for --->
ProgramAddressMap module <---
    wire WP;                            // 1 bit output Write Protect for --->
ProgramAddressMap module <---
    reg rx;                             // Serial data input bit (1-bit) for --->
UART_Receiver module <---
    wire rx_busy;                       // Signal to indicate whether transmission is
-> busy <- or not (tx_busy = 1: transmitter is busy and currently involved in
transmitting data | tx_done = 0: transmitter is not busy or has completed its
transmission) for ---> UART_Receiver module <---
    wire [7:0] data;                    // 8-bit data to transmit for --->
UART_Receiver module <---
    wire [13:0] Seven_Segment_Display;  // 14 bit output testing for seven segment
display for ---> DigitTo7SegmentDisplay.v module <---
    reg [31:0] noisy_data;              // 32-bit noisy input data for  --->
Low_Pass_Filter module <---
    wire [31:0] filtered_data;          // 32-bit filtered output data for --->
Low_Pass_Filter module <---

    reg [31:0] memory [0:125];      // Memory array to store 125 data points (32-bit)

    parameter ClockPeriod = 10;     // ClockPeriod is 10 ns

    // Clock generation
    initial clk = 0;            // Initialize clock signal to active low (0) at the
start of simulation
    always #(ClockPeriod / 2) clk = ~clk;       // Continuously toggle the clock every
(10 / 2 = 5 ns) to create a waveform

    // Instantiate the design under test (DUT)
    Top #(.N(N)) top
         (.clk(clk),
          .reset(reset),
          .nRESET(nRESET),
```

```
        .out(out),
        .lfsr_4bit(lfsr_4bit),
        .lfsr_8bit(lfsr_8bit),
        .lfsr_32bit(lfsr_32bit),
        .address(address),
        .read(read),
        .write(write),
        .Control_Module(Control_Module),
        .UART1(UART1),
        .CE0(CE0),
        .CE1(CE1),
        .OE0(OE0),
        .OE1(OE1),
        .WE0(WE0),
        .WE1(WE1),
        .CS0(CS0),
        .CS1(CS1),
        .WP(WP),
        .rx(rx),
        .rx_busy(rx_busy),
        .data(data),
        .Seven_Segment_Display(Seven_Segment_Display),
        .noisy_data(noisy_data),
        .filtered_data(filtered_data)
       );

integer i;
integer mem_index;              // Index to read data from the memory array

// Task to send an ID (in reverse) with start and stop bits
task send_ID(input [7:0] ID);
    begin
        rx <= 0;                                  // Start bit (LOW)
        @(posedge clk);                     // First Check (IDLE state)
        @(posedge clk);                     // Second Check (START state)

        // Send each bit of the ID in reverse order
        for (i = 7; i >= 0; i = i - 1) begin
            rx <= ID[i];
            @(posedge clk);
        end
```

```
            rx <= 1;  @(posedge clk);          // Stop bit (HIGH)

            rx <= 1; @(posedge clk);           // Default IDLE state (HIGH)
        end
    endtask


    initial begin
        // Initialize reset
        reset = 1;      // Start with reset enabled
        nRESET = 0;     // Start with nREST disabled

        read = 0;
        write = 0;

        noisy_data = 32'd0;
        mem_index = 0;      // Start at the beginning of the memory array

        // Wait for a few cycles and then release reset
        #5;
        reset = 0;      // Deactivte reset
        nRESET = 1;     // Activte nREST

        // Sending multiple ID numbers using the task to UART_Receiver module
        rx <= 1; @(posedge clk);           // Default IDLE state (HIGH)

        send_ID(8'd0);          // Start with ID = 0 (IDEL state)
        send_ID(8'd5);          // Change ID to 5 (move to START state)
        send_ID(8'd13);         // Record number
        send_ID(8'd35);         // Record number
        send_ID(8'd44);         // Record number
        send_ID(8'd46);         // 46 Indicates no more numbers (move to DONE state)
        send_ID(8'd47);         // 47 indicates there is another number to display (go
back to START state)
        send_ID(8'd30);         // Record number
        send_ID(8'd38);         // Record number
        send_ID(8'd46);         // 46 Indicates no more numbers (move to DONE state)
        send_ID(8'd0);          // 0 Indicates no more numbers (back to IDEL state)

        // ProgramAddressMap module Testing
        #100;
```

```
        address = 32'h0000_0BCD;      // An address between 0x0000_0000 to 0x07FF_FFFF
(Flash 0)

        #100;
        address = 32'h0800_0CBA;      // An address between 0x0800_0000 to 0x0FFF_FFFF
(Flash 1)

        #100;
        address = 32'h2000_0DEF;      // An address between 0x2000_0000 - 0x4AFF_FFFF
(Not Used)

        // DataMemoryAddress module Testing
        #90;
        address = 32'h1000_08AD;      // An address between 0x1000_0000 to 0x13FF_FFFF
(SRAM 0)

        #10;
        read = 1;

        #10;
        read = 0;
        write = 1;

        #10;
        write = 0;

        #90;
        address = 32'h1400_0F32;      // An address between 0x1400_0000 to 0x17FF_FFFF
(SRAM 1)

        #10;
        read = 1;

        #10;
        read = 0;
        write = 1;

        #10;
        write = 0;

        #100;
```

```
        address = 32'h2000_0FFA;      // An address between 0x2000_0000 to 0x44E0_FFFF
(Not Used)

        #100;
        address = 32'h44E1_0ABC;      // An address between 0x44E1_0000 to 0x44E1_1FFF
(Control Module)

        #100;
        address = 32'h44E1_28AD;      // An address between 0x44E1_2000 to 0x4802_1FFF
(Not Used)

        #100;
        address = 32'h4802_2C58;      // An address between 0x4802_2000 to 0x4802_2FFF
(UART1)

        #100;
        address = 32'h4802_3BBB;      // An address between 0x4802_3000 to 0x4AFF_FFFF
(Not Used)

        // Read the noisy data from the .mem file into the memory array
        $readmemh("NoisyWave.mem", memory);  // Read the data from the .mem file

        // Apply the noisy data to the filter
        for (i = 0; i < 125; i = i + 1) begin
            #20 noisy_data = memory[mem_index];     // Feed the noisy data into the
filter
            mem_index = mem_index + 1;      // Move to the next data point
        end

        // Simulate for 200ns
        #200;
        $finish;
    end

endmodule
```

10.2.2 DigitTo7SegmentDisplay_tb.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
//////////////////////////////////////////////////////////////////////////////////


module DigitTo7SegmentDisplay_tb;

    reg clk;                // System Clock
    reg rx;            // Serial data input bit (1-bit)
    wire [7:0] data;         // 8-bit data to transmit
    wire rx_busy;        // Signal to indicate whether transmission is -> busy <- or
not (tx_busy = 1: transmitter is busy and currently involved in transmitting data |
tx_done = 0: transmitter is not busy or has completed its transmission)
    reg reset;
    wire [13:0] Seven_Segment_Display;  // 14 bit output testing for seven segment
display for ---> DigitTo7SegmentDisplay.v module <---

    UART_Receiver
          Receiver
          (
          .baud_clk(clk),
          .rx(rx),
          .data(data),
          .rx_busy(rx_busy)
          );

    wire [7:0] ID = {data[0], data[1], data[2], data[3], data[4], data[5], data[6],
data[7]};

    // Instantiation of the DigitTo7SegmentDisplay module
    DigitTo7SegmentDisplay display (.clk(clk),
                                    .reset(reset),
                                    .ID(ID[5:0]),
                                    .Seven_Segment_Display(Seven_Segment_Display)
                                    );

    parameter CLK_PERIOD = 10; // 100 MHz clock
```

```
// Clock Generation
initial begin
    clk = 0;
    forever #(CLK_PERIOD / 2) clk = ~clk; // Toggle clock
end

integer i;

// Task to send an ID (in reverse) with start and stop bits
task send_ID(input [7:0] ID);
    begin
        rx <= 0;                                  // Start bit (LOW)
        @(posedge clk);                    // First Check (IDLE state)
        @(posedge clk);                    // Second Check (START state)

        // Send each bit of the ID in reverse order
        for (i = 7; i >= 0; i = i - 1) begin
            rx <= ID[i];
            @(posedge clk);
        end

        rx <= 1;  @(posedge clk);          // Stop bit (HIGH)

        rx <= 1; @(posedge clk);           // Default IDLE state (HIGH)
    end
endtask

initial begin
    clk <= 0;
    rx <= 1; @(posedge clk);           // Default IDLE state (HIGH)

    // Initialize reset
    reset = 1;      // Start with reset enabled

    // Wait for a few cycles and then release reset
    #5;
    reset = 0;      // Deactivte reset

    // Sending multiple ID numbers using the task
    send_ID(8'd0);   // Start with ID = 0 (IDEL state)
    send_ID(8'd5);   // Change ID to 5 (move to START state)
    send_ID(8'd13);  // Record number
```

```
        send_ID(8'd35);  // Record number
        send_ID(8'd44);  // Record number
        send_ID(8'd46);  // 46 Indicates no more numbers (move to DONE state)
        send_ID(8'd47);  // 47 indicates there is another number to display (go back
to START state)
        send_ID(8'd30);  // Record number
        send_ID(8'd38);  // Record number
        send_ID(8'd46);  // 46 Indicates no more numbers (move to DONE state)
        send_ID(8'd0);   // 0 Indicates no more numbers (back to IDEL state)

        // Simulate for 50ns
        #500;
    end

endmodule
```

10.2.3 ProgramAddressMap_tb.v

```
`timescale 1ns / 1ns
////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
////////////////////////////////////////////////////////////////////////////////


module ProgramAddressMap_tb;

    // Parameters
    parameter N = 4;

    reg clk;
    reg nRESET;                         // Active Low reset
    reg [N*8-1:0] address;              // 32 bit input addrerss for --->
ProgramAddressMap and DataMemoryAddress modules <---
    wire CS0;                           // 1 bit output first Flash CS0 for --->
ProgramAddressMap module <---
    wire CS1;                           // 1 bit output second Flash CS1 for --->
ProgramAddressMap module <---
    wire WP;                            // 1 bit output Write Protect for --->
ProgramAddressMap module <---

    parameter ClockPeriod = 10;     // ClockPeriod is 10 ns

    // Clock generation
    initial clk = 0;            // Initialize clock signal to active low (0) at the
start of simulation
    always #(ClockPeriod / 2) clk = ~clk;       // Continuously toggle the clock every
(10 / 2 = 5 ns) to create a waveform

    // Instantiation of the ProgramAddressMap module
    ProgramAddressMap #(.N(N*8)) dataAdd (.clk(clk),
                                          .nRESET(nRESET),
                                          .address(address),
                                          .CS0(CS0),
                                          .CS1(CS1),
                                          .WP(WP)
                                         );
```

```
    initial begin
        // Initialize reset
        nRESET = 0;     // Start with nREST disabled

        // Wait for a few cycles and then release reset
        #5;
        nRESET = 1;     // Activte nREST

        // Flash Addresses Testing
        #30;
        address = 32'h0000_0BCD;      // An address between 0x0000_0000 to 0x07FF_FFFF

        #30;
        address = 32'h0800_0CBA;      // An address between 0x0800_0000 to 0x0FFF_FFFF

        #30;
        address = 32'h2000_0DEF;      // An address between 0x2000_0000 - 0x4AFF_FFFF

        // Simulate for 50ns
        #50;
        $finish;
    end

endmodule
```

10.2.4 DataMemoryAddress_tb.v

```
`timescale 1ns / 1ns
////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
////////////////////////////////////////////////////////////////////////////////

module DataMemoryAddress_tb;
    // Parameters
    parameter N = 4;

    reg clk;
    reg nRESET;                         // Active Low reset
    reg [N*8-1:0] address;              // 32 bit input addrerss for --->
ProgramAddressMap and DataMemoryAddress modules <---
    reg read;                           // 1 bit input read signal for --->
DataMemoryAddress module <---
    reg write;                          // 1 bit input write signal for --->
DataMemoryAddress module <---
    wire Control_Module;                // 1 bit output Input Port for --->
DataMemoryAddress module <---
    wire UART1;                         // 1 bit output Output Port for --->
DataMemoryAddress module <---
    wire CE0;                           // 1 bit output first SRAM Chip Enable for
---> DataMemoryAddress module <---
    wire CE1;                           // 1 bit output second SRAM Chip Enable for
---> DataMemoryAddress module <---
    wire OE0;                           // 1 bit output first SRAM Output Enable for
---> DataMemoryAddress module <---
    wire OE1;                           // 1 bit output second SRAM Output Enable for
---> DataMemoryAddress module <---
    wire WE0;                           // 1 bit output first SRAM Write Enable for
---> DataMemoryAddress module <---
    wire WE1;                           // 1 bit output second SRAM Write Enable for
---> DataMemoryAddress module <---

    parameter ClockPeriod = 10;     // ClockPeriod is 10 ns

    // Clock generation
    initial clk = 0;            // Initialize clock signal to active low (0) at the
start of simulation
```

```
    always #(ClockPeriod / 2) clk = ~clk;       // Continuously toggle the clock every
(10 / 2 = 5 ns) to create a waveform

    // Instantiation of the DataMemoryAddress module
    DataMemoryAddress #(.N(N*8)) progAdd (.clk(clk),
                                          .nRESET(nRESET),
                                          .address(address),
                                          .read(read),
                                          .write(write),
                                          .Control_Module(Control_Module),
                                          .UART1(UART1),
                                          .CE0(CE0),
                                          .CE1(CE1),
                                          .OE0(OE0),
                                          .OE1(OE1),
                                          .WE0(WE0),
                                          .WE1(WE1)
                                         );

    initial begin
        // Initialize reset
        nRESET = 0;     // Start with nREST disabled

        read = 0;
        write = 0;

        // Wait for a few cycles and then release reset
        #5;
        nRESET = 1;     // Activte nREST

       // DataMemoryAddress module Testing
        #50;
        address = 32'h1000_08AD;      // An address between 0x1000_0000 to 0x13FF_FFFF
(SRAM 0)

        #10;
        read = 1;

        #10;
        read = 0;
        write = 1;
```

```
        #10;
        write = 0;

        #50;
        address = 32'h1400_0F32;      // An address between 0x1400_0000 to 0x17FF_FFFF
(SRAM 1)

        #10;
        read = 1;

        #10;
        read = 0;
        write = 1;

        #10;
        write = 0;

        #50;
        address = 32'h2000_0FFA;      // An address between 0x2000_0000 to 0x44E0_FFFF
(Not Used)

        #50;
        address = 32'h44E1_0ABC;      // An address between 0x44E1_0000 to 0x44E1_1FFF
(Control Module)

        #50;
        address = 32'h44E1_28AD;      // An address between 0x44E1_2000 to 0x4802_1FFF
(Not Used)

        #50;
        address = 32'h4802_2C58;      // An address between 0x4802_2000 to 0x4802_2FFF
(UART1)

        #50;
        address = 32'h4802_3BBB;      // An address between 0x4802_3000 to 0x4AFF_FFFF
(Not Used)

        // Simulate for 50ns
        #50;
        $finish;
    end
endmodule
```

10.2.5 Low_Pass_Filter_tb.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: San Diego State University
// Engineer: Abdul Karim Tamim
//////////////////////////////////////////////////////////////////////////////////


module Low_Pass_Filter_tb;

    reg clk;
    reg reset;
    reg [31:0] noisy_data;          // 32-bit noisy data input to the filter
    wire [31:0] filtered_data;      // 32-bit filtered output data

    reg [31:0] memory [0:125];      // Memory array to store 125 data points (32-bit)

    integer i;
    integer mem_index;              // Index to read data from the memory array

    parameter ClockPeriod = 10;     // ClockPeriod is 10 ns

    // Clock generation
    initial clk = 0;            // Initialize clock signal to active low (0) at the
start of simulation
    always #(ClockPeriod / 2) clk = ~clk;       // Continuously toggle the clock every
(10 / 2 = 5 ns) to create a waveform

    // Instantiate the Low_Pass_Filter module
    Low_Pass_Filter filter (
        .clk(clk),
        .reset(reset),
        .noisy_data(noisy_data),
        .filtered_data(filtered_data)
    );

    // Clock generation (50 MHz clock)
    always begin
        #10 clk = ~clk;  // Toggle every 10ns, creating a 50 MHz clock
    end
```

```
    // Stimulus process
    initial begin
        // Initialize signals
        clk = 0;
        reset = 1;
        noisy_data = 32'd0;
        mem_index = 0;  // Start at the beginning of the memory array

        #5;
        // Apply reset
        reset = 0;

        // Read the noisy data from the .mem file into the memory array
        $readmemh("NoisyWave.mem", memory);  // Read the data from the .mem file

        // Apply the noisy data to the filter
        for (i = 0; i < 125; i = i + 1) begin
            #20 noisy_data = memory[mem_index];     // Feed the noisy data into the
filter
            mem_index = mem_index + 1;      // Move to the next data point
        end

        // End simulation after all data has been fed into the filter
        #2500;
        $finish;
    end

endmodule
```

10.3 Memory Files

10.3.1 SineWave.mem

```
00000000
00010110
00101011
01000000
01010010
01100001
01101110
01110111
01111101
01111111
01111101
01110111
01101110
01100001
01010010
01000000
00101011
00010110
00000000
11101010
11010101
11000000
10101110
10011111
10010010
10001001
10000011
10000001
10000011
10001001
10010010
10011111
10101110
11000000
11010101
11101010
00000000 . . .
```

Access Full File: SineWave.mem

10.3.2 NoisyWave.mem

```
40edeebb
4028b97a
402151e0
bdd37e58
3f5a4810
c0577f36
c09f2fce
c0a3639a
c1074c26
c138b451
c0fdb366
c104037d
c0daaa86
c10f82d8
c12eb6fb
c11ee56d
c160fe3f
c0f6fdfd
c0a8a749
c0460403
c05ca133
bfa6abb3
bf9b07f1
bf826245
40519792
41021864
40e13962
40981d7c
40ba363a
41117b74
41397538
41262058
4102020d
4123afdc
40d76ff6
41124020
41010a43
410a7490 . . .
```

Access Full File: NoisyWave.mem

10.4 Constraints

10.4.1 system_constraints.xdc

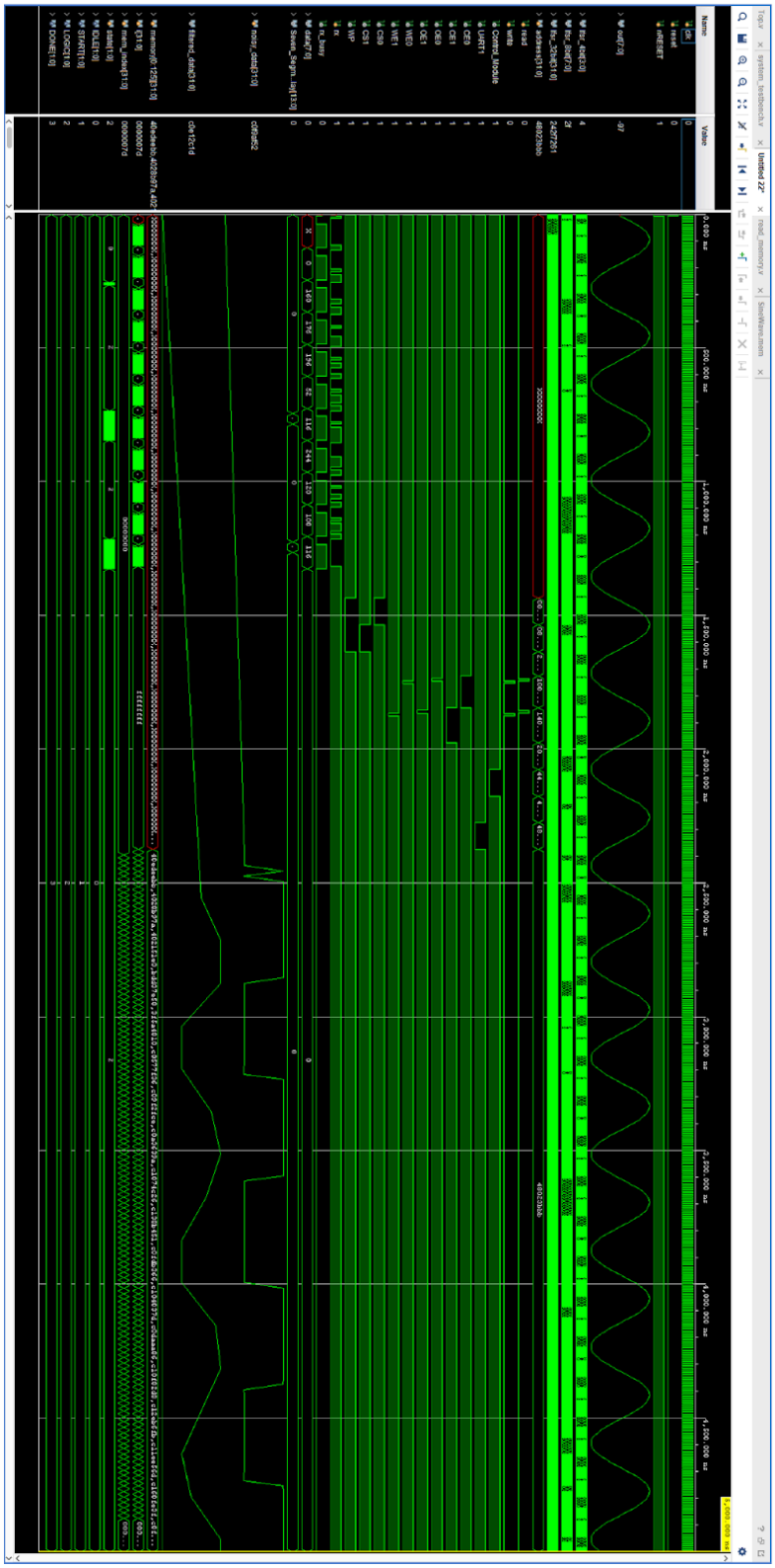
```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
    set_property IOSTANDARD LVCMOS33 [get_ports clk]
       create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports
clk]

# Create the clock with a 300 MHz frequency (period = 3.33 ns)
create_clock -add -name cpu_clk -period 3.33 [get_ports cpu_clk]

set_property PACKAGE_PIN W19 [get_ports nRESET]
    set_property IOSTANDARD LVCMOS33 [get_ports nRESET]

##Sch name = JA1
set_property PACKAGE_PIN G17 [get_ports {address[0]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[0]}]
set_property PACKAGE_PIN G18 [get_ports {address[1]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[1]}]
set_property PACKAGE_PIN G19 [get_ports {address[2]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[2]}]
set_property PACKAGE_PIN H1 [get_ports {address[3]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[3]}]
set_property PACKAGE_PIN H2 [get_ports {address[4]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[4]}]
set_property PACKAGE_PIN H19 [get_ports {address[5]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[5]}]
set_property PACKAGE_PIN J1 [get_ports {address[6]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[6]}]
set_property PACKAGE_PIN J2 [get_ports {address[7]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[7]}]
set_property PACKAGE_PIN J3 [get_ports {address[8]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[8]}]
set_property PACKAGE_PIN K18 [get_ports {address[9]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[9]}]
set_property PACKAGE_PIN K19 [get_ports {address[10]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[10]}]
set_property PACKAGE_PIN L1 [get_ports {address[11]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[11]}]
set_property PACKAGE_PIN L2 [get_ports {address[12]}]
       set_property IOSTANDARD LVCMOS33 [get_ports {address[12]}]
```

```
set_property PACKAGE_PIN L3 [get_ports {address[13]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[13]}]
set_property PACKAGE_PIN N17 [get_ports {address[14]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[14]}]
set_property PACKAGE_PIN N18 [get_ports {address[15]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[15]}]
set_property PACKAGE_PIN N19 [get_ports {address[16]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[16]}]
set_property PACKAGE_PIN P1 [get_ports {address[17]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[17]}]
set_property PACKAGE_PIN P3 [get_ports {address[18]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[18]}]
set_property PACKAGE_PIN P17 [get_ports {address[19]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[19]}]
set_property PACKAGE_PIN P18 [get_ports {address[20]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[20]}]
set_property PACKAGE_PIN P19 [get_ports {address[21]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[21]}]
set_property PACKAGE_PIN T2 [get_ports {address[22]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[22]}]
set_property PACKAGE_PIN T3 [get_ports {address[23]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[23]}]
set_property PACKAGE_PIN T17 [get_ports {address[24]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[24]}]
set_property PACKAGE_PIN T18 [get_ports {address[25]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[25]}]
set_property PACKAGE_PIN U1 [get_ports {address[26]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[26]}]
set_property PACKAGE_PIN U2 [get_ports {address[27]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[27]}]
set_property PACKAGE_PIN U3 [get_ports {address[28]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[28]}]
set_property PACKAGE_PIN U4 [get_ports {address[29]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[29]}]
set_property PACKAGE_PIN U5 [get_ports {address[30]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[30]}]
set_property PACKAGE_PIN U7 [get_ports {address[31]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {address[31]}]

# Switches
set_property PACKAGE_PIN R2 [get_ports {read}]
      set_property IOSTANDARD LVCMOS33 [get_ports {read}]
```

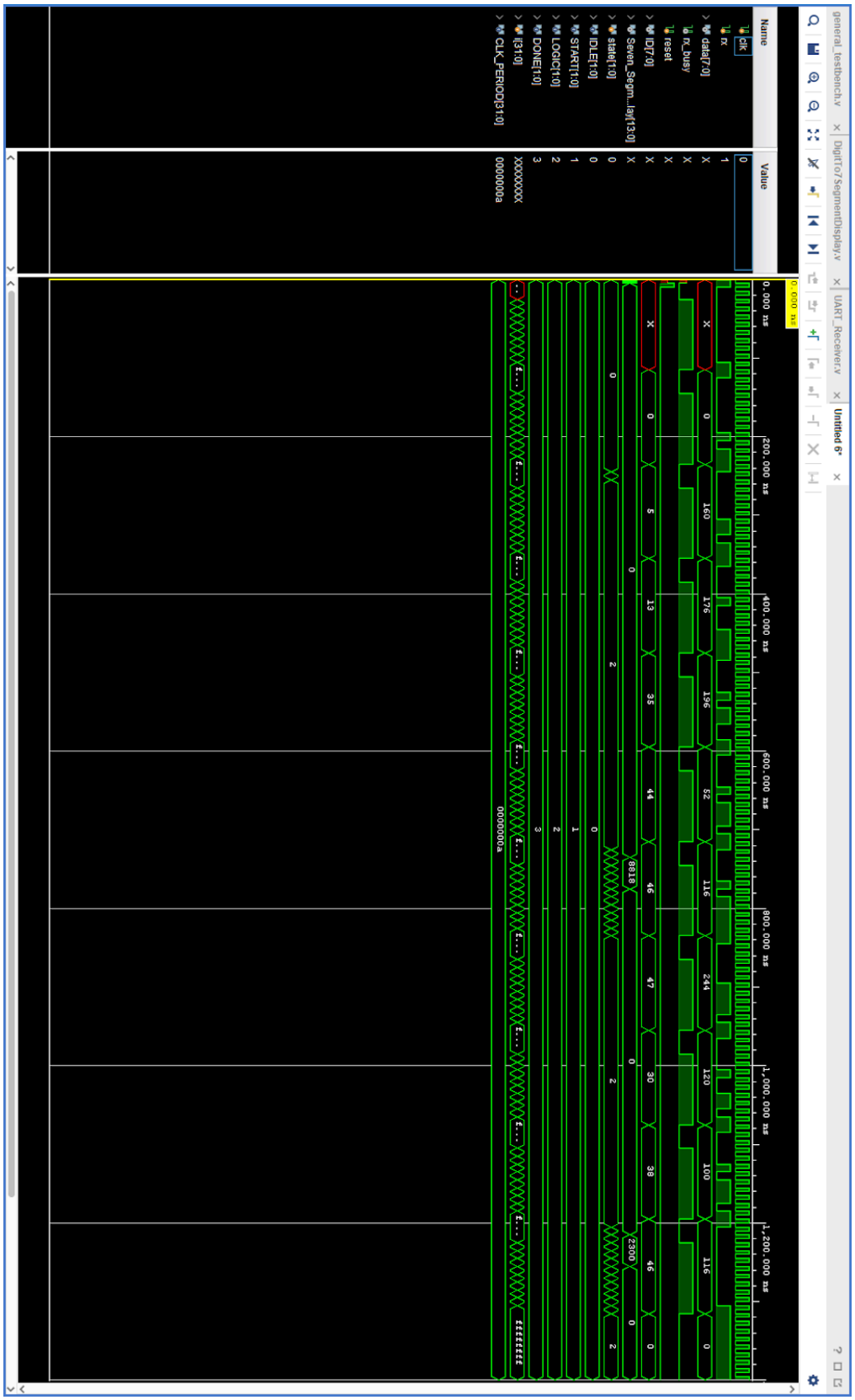
```
set_property PACKAGE_PIN T1 [get_ports {write}]
       set_property IOSTANDARD LVCMOS33 [get_ports {write}]

##Pmod Header JB
##Sch name = JB2
set_property PACKAGE_PIN A16 [get_ports {CE0}]
       set_property IOSTANDARD LVCMOS33 [get_ports {CE0}]
##Sch name = JB8
set_property PACKAGE_PIN A17 [get_ports {CE1}]
       set_property IOSTANDARD LVCMOS33 [get_ports {CE1}]
##Sch name = JB3
set_property PACKAGE_PIN B15 [get_ports {OE0}]
       set_property IOSTANDARD LVCMOS33 [get_ports {OE0}]
##Sch name = JB4
set_property PACKAGE_PIN B16 [get_ports {OE1}]
       set_property IOSTANDARD LVCMOS33 [get_ports {OE1}]
##Sch name = JB9
set_property PACKAGE_PIN C15 [get_ports {WE0}]
       set_property IOSTANDARD LVCMOS33 [get_ports {WE0}]
##Sch name = JB10
set_property PACKAGE_PIN C16 [get_ports {WE1}]
       set_property IOSTANDARD LVCMOS33 [get_ports {WE1}]

##Pmod Header JC
##Sch name = JC1
set_property PACKAGE_PIN K17 [get_ports {CS0}]
       set_property IOSTANDARD LVCMOS33 [get_ports {CS0}]
##Sch name = JC7
set_property PACKAGE_PIN L17 [get_ports {CS1}]
       set_property IOSTANDARD LVCMOS33 [get_ports {CS1}]
##Sch name = JC2
set_property PACKAGE_PIN M18 [get_ports {WP}]
       set_property IOSTANDARD LVCMOS33 [get_ports {WP}]
```

10.4 Simulation Waveform Images

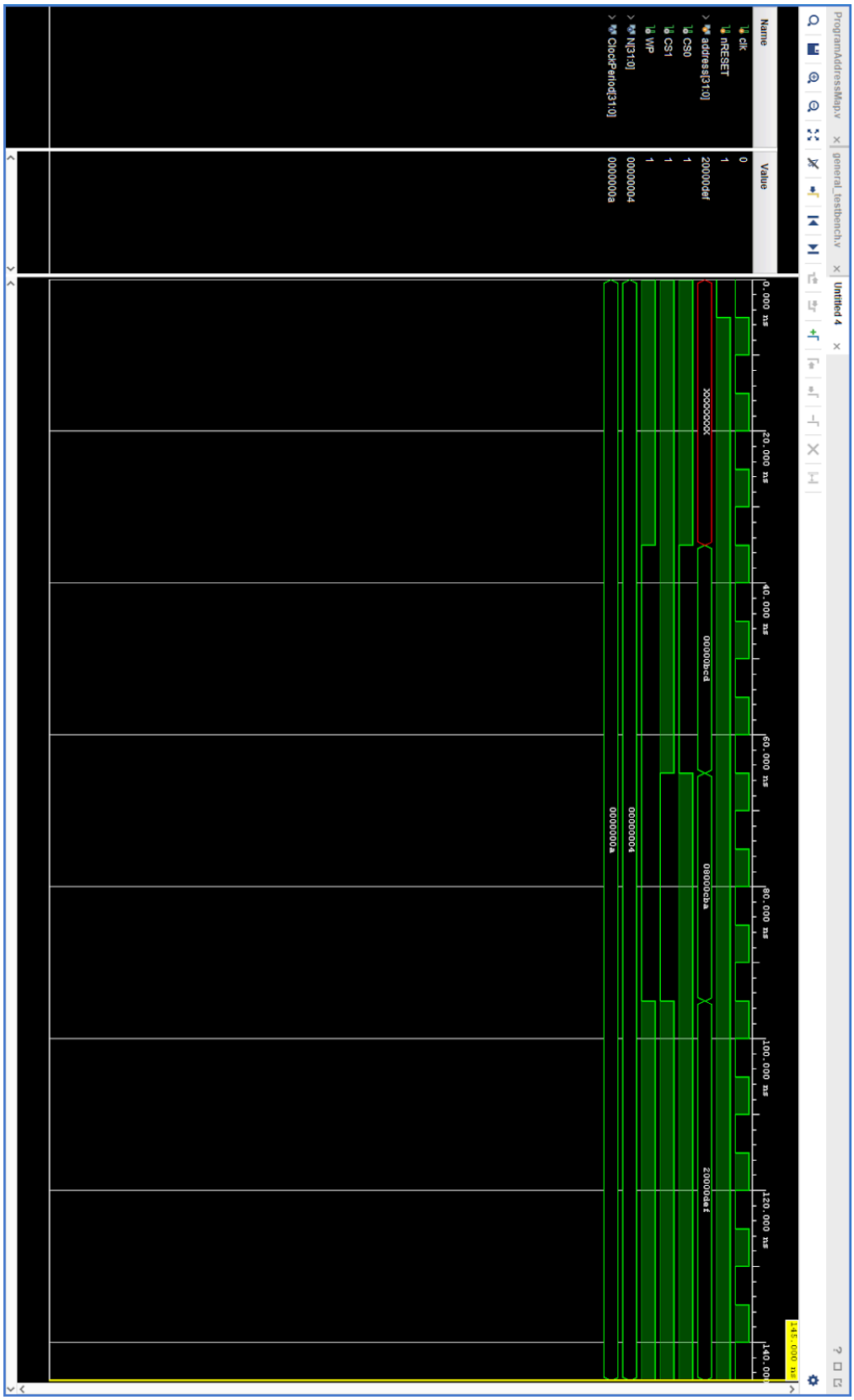
10.4.1 Top Module Simulation Waveform



10.4.2 DigitTo7SegmentDisplay Module Simulation Waveform

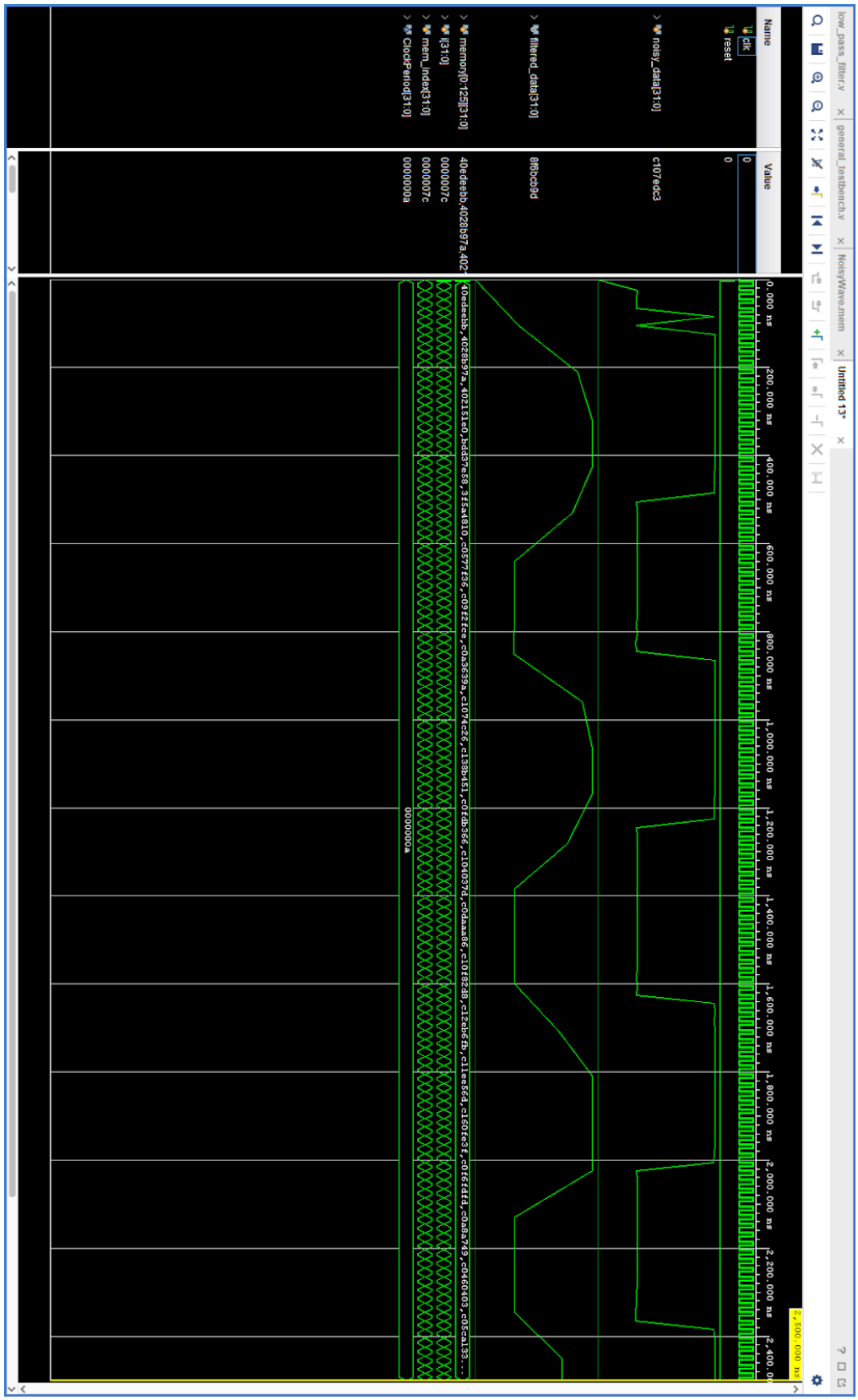


10.4.3 ProgramAddressMap Module Simulation Waveform



10.4.4 DataMemoryAddress Module Simulation Waveform

10.4.5 LowPassFilter Module Simulation Waveform



11 Python Files

11.1 csv_to_mem.py

```
import csv
import math

def csv_to_mem(csv_filename, mem_filename):
    # Open the CSV file for reading
    with open(csv_filename, 'r') as csvfile:
        reader = csv.reader(csvfile)

        # Open the .mem file for writing
        with open(mem_filename, 'w') as memfile:
            for row in reader:
                angle = float(row[0])
                sine_value = float(row[1])

                # Scale the sine value to fit in an 8-bit signed integer (-128 to 127)
                scaled_value = round(sine_value * 127)

                # Convert the scaled value to 8-bit two's complement binary
                if scaled_value < 0:
                    bin_value = format((1 << 8) + scaled_value, '08b')  # Two's
complement for negative values
                else:
                    bin_value = format(scaled_value, '08b')

                # Write the binary value to the .mem file
                memfile.write(bin_value + '\n')

# Convert your SineWave.csv to SineWave.mem
csv_to_mem('SineWave.csv', 'SineWave.mem')
```

11.2 SineWave.csv

```
0,0
10,0.1736481777
20,0.3420201433
30,0.5
40,0.6427876097
50,0.7660444431
60,0.8660254038
70,0.9396926208
80,0.984807753
90,1
100,0.984807753
110,0.9396926208
120,0.8660254038
130,0.7660444431
140,0.6427876097
150,0.5
160,0.3420201433
170,0.1736481777
180,0
190,-0.1736481777
200,-0.3420201433
210,-0.5
220,-0.6427876097
230,-0.7660444431
240,-0.8660254038
250,-0.9396926208
260,-0.984807753
270,-1
280,-0.984807753
290,-0.9396926208
300,-0.8660254038
310,-0.7660444431
320,-0.6427876097
330,-0.5
340,-0.3420201433
350,-0.1736481777
360,0
```

12 What I Would Have Done Differently

Working on this project allowed me to delve deeper into the world of hardware engineering. Reflecting on the experience, one thing I would have done differently is actually building the hardware component to create a functional voice-interactive 7-segment display system using the Basys-3 Artix-7 FPGA. Unfortunately, the microprocessor I selected has more pins than the Basys-3 board can accommodate, making it extremely difficult, if not impossible, to connect all the necessary pins and develop a fully functional system. In hindsight, I would have chosen a processor with fewer pins, compatible with the Basys-3 board, to focus on the hardware implementation of the project.

13 Conclusion

Overall, this project turned out to be another successful achievement for me. I thoroughly enjoyed every moment I spent working on it. This class and project provided me with a valuable opportunity to learn about computer hardware and gain insight into how things work internally within a computer. Unlike previous courses that primarily focused on software implementation, this class emphasized hardware concepts, such as programming in Verilog, simulating module implementations, calculating noise margins, performing loading and timing analyses, and understanding memory addressing. These were entirely new concepts for me, and this class gave me the chance to explore them in depth.

One of the most valuable skills I also gained was learning how to read and interpret datasheets. Datasheets are often extensive and contain a wealth of technical information about components, but this class taught me how to efficiently navigate and extract the specific details I need. This skill will undoubtedly prove beneficial for future projects.

Finally, I would like to extend my gratitude to **Professor Ken Arnold** for his dedication to teaching us everything we needed to successfully complete this project and course. His guidance and motivation inspired me to consistently perform at my best, meet deadlines, and seek his feedback during office hours. His support and encouragement played a significant role in turning this project into a success, and I truly appreciate the time he devoted to helping us succeed.

14 Total Hours Spent On The Project

After reviewing my Google Calendar, I calculated that I spent a total of **165 hours** on this project over the course of the semester. All this hard work and dedication paid off, resulting in a successful project in the end.