Be Boulder.

University of Colorado Boulder

PROJECT #2 GUIDE

UCB ECEN 5863 FALL 2023 PROJECT #2: MICROSEMI SMARTFUSION2 & XILINX PYNQ

TABLE OF CONTENTS

Introduction 2

Learning Objectives 3

I. Procedure 4

Module I: SmartTime For SmartFusion2 4

Module II: SmartFusion2 System on a Chip 4

Module III: Load-testing FPGAs with Counters 4

Bonus Module: Xilinx Pynq: Embedded Software Development on an SoC Using Python 5

II. Deliverables 5

Technical Report 5

Recorded Observations, Test Data, and Images 6

FPGA Project directory zipped for each of 4 Modules 6

Software Files for all the modules in a zip file 6

III. Evaluation 6

INTRODUCTION

Microsemi SmartFusion®2 SoC FPGAs integrate a fourth-generation, flash-based FPGA fabric, an ARM Cortex-M3 processor, and high-performance communications interfaces on a single chip. The SmartFusion2 family is the industry's lowest-power, most reliable, and highest-security programmable logic solution. SmartFusion2 SoC FPGAs offer up to 3.6X the gate density and up to 2X the performance of previous Flash-based FPGA families, and also include multiple memory blocks and multiply-accumulate blocks for DSP processing. The 166-MHz ARM Cortex-M3 processor is enhanced with an embedded trace macrocell (ETM), a memory protection unit (MPU), an 8-KB instruction cache, and additional peripherals, including controller area network (CAN), gigabit Ethernet, and a high-speed universal serial bus (USB). High-speed serial interfaces include PCI Express (PCIe), 10-Gbps Attachment Unit Interface (XAUI)/XGMII extended sublayer (XGXS), plus native serialization/deserialization (SerDes) communication. The DDR2/DDR3 memory controllers available in the devices provide high-speed memory interfaces.

SmartFusion2 SoC FPGAs provide you with:

Full Design Customizability

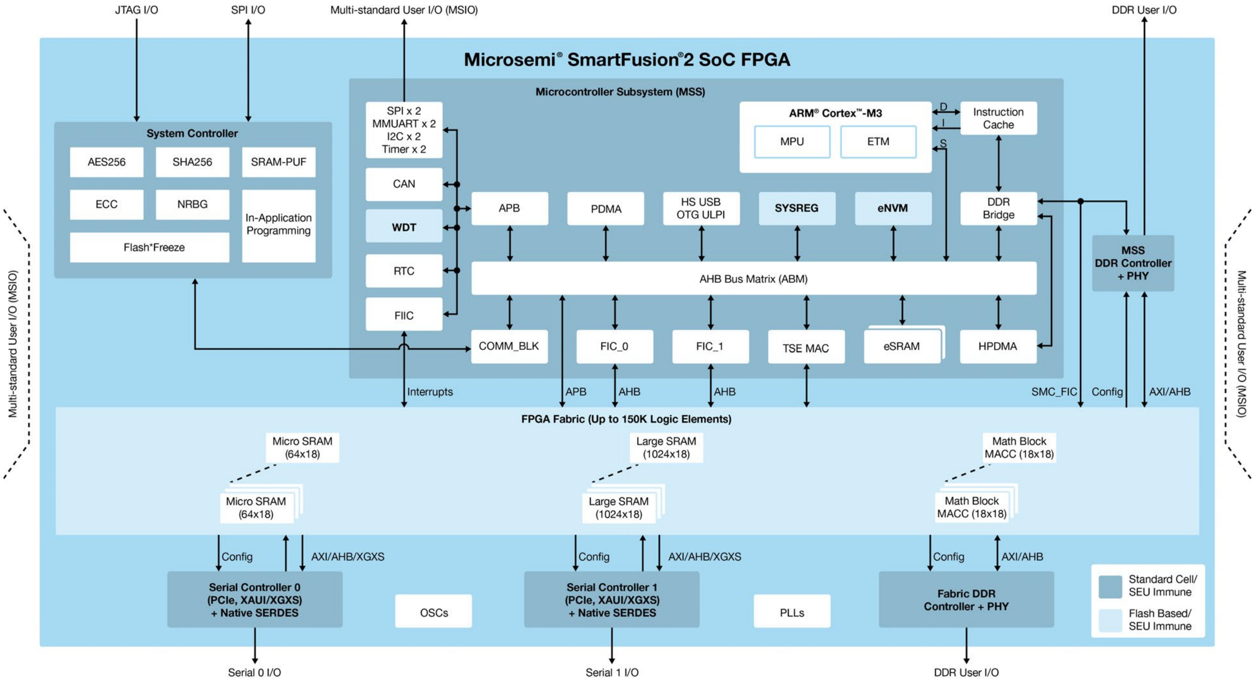
- Create a product with exactly the features you need
- Innovate and differentiate for a competitive edge
- Incorporate last-minute changes with an on-chip FPGA fabric
- Future-proof your design with in-application programming (IAP) capability for field upgrades
- Experiment with hardware acceleration for select algorithms in FPGA fabric

Intellectual Property (IP) Protection

- Inherent flash FPGA IP security with a complete SoC
- Snoop-free interface between microcontroller and FPGA
- High design security since no bitstream download is required at power-up
- Secure in-system programming (ISP) with 256-bit Advanced Encryption Standard (AES) via JTAG
- FlashLock ® to prevent FPGA contents from being read back
- Protection against overbuilding with customer programmable device key

Ease-of-Use Increases Productivity

- A single platform for your entire line of products
- Libero SoC integrated design environment for both FPGA and embedded designers
- Simple GUI-based configuration of complex programmable analog
- Industry leading compile and debug from Keil™, IAR Systems® and GNU
- Operating system support includes uClinux from Emcraft Systems, FreeRTOS, SAFERTOS and uc/OS-III from Micrium



MICROCONTROLLER SUBSYSTEM (MSS)

- Hardware industry-standard 166 MHz, 32-bit ARM Cortex-M3 CPU
- Multi-layer AHB communication matrix with up to 16 Gbps throughput
- Triple-speed Ethernet (TSE) 10/100/1000 Mbps MAC
- Two of each: SPI, I2C, UART, 32-bit timers
- Up to 512 KB flash and 64 KB of SRAM
- USB 2.0 high speed on-the-go (OTG) controller with ULPI interface
- 8-channel DMA controller

FPGA FABRIC

- Efficient 4-input look-up tables (LUTs) with carry chains for high performance and low power
- Embedded SRAMs and FIFOs
- Up to 574 I/Os supporting LVDS, PCI, PCI-X and LVTTL/LVCMOS standards

LEARNING OBJECTIVES

For this project, the objective is for students to:

- Become familiar with the Microsemi FPGA development flow, particularly in the case of a SoC with software development flow included.

- Appreciate the capabilities of the Microsemi SmartFusion2 System on a chip, and compare and contrast it to the Xilinx Zynq SoC.
- Learn how to build hardware using the SmartDesign design tool.
- Learn to evaluate system timing using SmartTime.
- Learn how to integrate software with hardware in the same device using SoftConsole.
- Design and build a several hardware examples using the SmartFusion2.
- Consider hardware and software tradeoff possibilities available in the SmartFusion2 system architecture.

Do not be afraid to ask questions as you work on this project. It involves many processes and actions that may seem complicated and confusing at first, but will become clearer as you work through the modules.

The modules will get progressively more difficult and take more time. Try to finish Modules 1 and 2 in the first week, so that you will have the second week to complete Module 3 and write the report.

I. PROCEDURE

MODULE I: SMARTTIME FOR SMARTFUSION2

1. Run the SmartTime tutorial found in SmartTimeTutorial. Pdf. Use the source files found in the accompanying zip file SmartTimeTutorial.zip.
2. For each of the example circuits, Run SmartTime and Record the fMAX and estimate the % utilization of the FPGA logic.

MODULE II: SMARTFUSION2 SYSTEM ON A CHIP

1. Download and follow the directions in the document SmartFusion2MakerDemo.pdf. You will use the Digi-key SmartFusion2 Maker Kit for this part of the project. Use the source files found in the accompanying zip file M2S010-MKR-KIT_FirstProj-master.zip
2. Record the fMAX, input arrival time and clock to output time in the timing report. What is the smallest timing margin listed in the timing report?
3. Estimate the % utilization of the FPGA logic.
4. From the Soft Console Debug windows, record the values of R13, R15, and memory addresses 0x00000000 and 0x00000004.
5. Record your observations of the board behavior once the FPGA is programmed. Does it behave as you expected?

MODULE III: LOAD-TESTING FPGAS WITH COUNTERS

1. Create a **16-bit counter** in Verilog or VHDL. Make it into a module or component. Instantiate this module or component into a top-level design, and verify with an RTL simulation that it works. Record your simulation waveform.
2. Estimate the **number of 16-bit counters** that will fit into both the **Altera MAX10 DE10-lite and the Microsemi SmartFusion2 Maker kit** boards. Generate these in your design within the Verilog or VHDL code **(Hint: USE FOR...GENERATE)** and use the terminal count from one counter to control the count enable of the next counter in a chain. Attach the last terminal count to an I/O pin to force the

synthesizer to compile. Try to compile this design in Quartus and Libero respectively, and report the results. How many counters can you fit?

NOTE: These compilations may take a long time, many minutes to hours, be patient.

3. Now, **using Quartus** and estimate the maximum number of counters for the DE1-SoC board.
4. Remove counters until each design compiles without error, and report the results. Verify at least one counter output by driving external pins (preferably the LEDs) on one board and using RTL simulation for all boards.
5. Report the number of counters that can be included in each design and still close timing.
6. Record the Fmax in the timing report for each of the 3 boards.
7. Estimate the % utilization of the FPGA logic.
8. Compare all the results from all the boards.

9. BONUS: 5 points each for the team with the verified most number of counters for each of the board designs

BONUS MODULE: XILINX PYNQ: EMBEDDED SOFTWARE DEVELOPMENT ON AN SOC USING PYTHON

1. **Install Xilinx's Vivado tool (use 15.1 webinstall edition**). You can try more current versions of the tool, but 15.1 has be verified to work.
2. Follow the directions in http://pynq.readthedocs.io/en/latest/getting_started.html
3. Configure 2 buttons using Asynchronous interrupt handlers as 2 Morse code symbols and print the input characters on terminal. You can refer to https://github.com/Xilinx/PYNQ/blob/master/docs/source/overlay_design_methodology/pynq_and_asyncio.rst
and https://github.com/Xilinx/PYNQ/blob/master/boards/Pynq-Z1/base/notebooks/board/asyncio_buttons.ipynb found at Xilinx/PYNQ/boards/Pynq-Z1/base/ /notebooks/board

II. DELIVERABLES

Provide zipped project files for all modules, and the report and test results in a separate directory. Suggested Project target dates are:

Module 1 – 2023/10/18

Module 2 – 2023/10/23

Module 3 – 2023/03/29

All deliverables are due on 2023/10/30. Please submit them to Canvas.

Each partner pair must provide a technical report that explains the project objectives and test results. At a minimum, it should include:

1. Executive Summary
2. Objectives
3. Procedure
4. Module Test Results
5. Lessons Learned
6. Conclusions
7. Appendix: References

The report should be 5 pages in length in a Word document. Be sure the report addresses the questions posed in the module procedures.

RECORDED OBSERVATIONS, TEST DATA, AND IMAGES

Include observations recorded in a lab notebook, test data taken, and any digital pictures of the proceedings.

FPGA PROJECT DIRECTORY ZIPPED FOR EACH OF 4 MODULES

Starting with the top level of each module, zip up the entire directory including subfolders. This will allow us to replicate your work and help provide feedback on any errors you encounter. Label them appropriately so we can find the information for each module.

SOFTWARE FILES FOR ALL THE MODULES IN A ZIP FILE

Zip up all source files used to run the SmartFusion2 ARM Cortex-M3 and Pynq ARM Cortex-A9 software.

III. EVALUATION

Any award to be made pursuant to this project will be based upon deliverables. The following elements will be the primary considerations in evaluating all submitted projects:

30% Technical Report

20% TA Demos (Deliverables)

15% Deliverables for Project Module 1

20% Deliverables for Project Module 2

15% Deliverables for Project Module 3

10% Deliverables for Project Bonus Module