Developing Multidisciplinary Laboratories with the Cypress PSoC

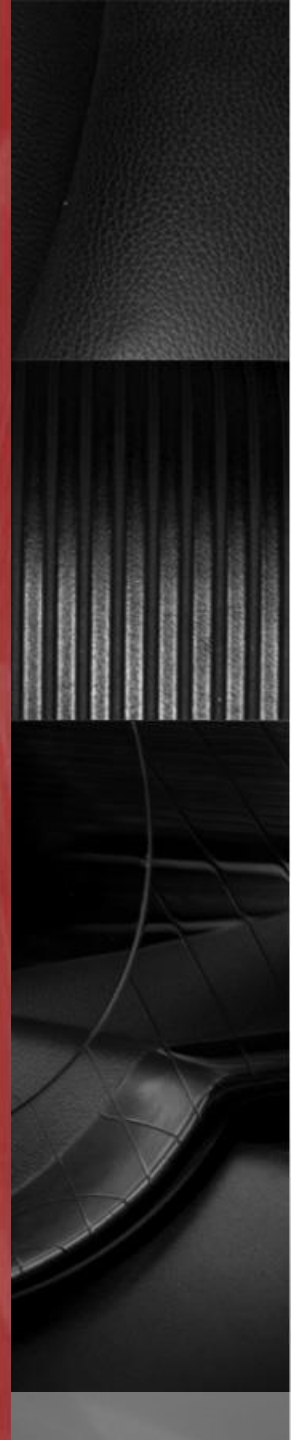
Masters Project by Stephen Hammack

Committee

Jim Vallino

James Heliotis

Zack Butler



Outline

- Purpose
- Deliverables
- PSoC 5LP and PSoC Creator
- Labs
  - Manuals
  - Instructors Guides
- Final Project
  - Example – YM2149 Programmable Sound Generator Emulator
- Lab Evaluations
- Improvements

Purpose

- Few CS courses at RIT introduce concepts of computer hardware
- Solution: create labs that could lead to creating a course focused on teaching computer hardware concepts to CS students
- Best to teach computer hardware concepts with a reconfigurable platform
- FPGA is one solution, but requires learning VHDL; very different from traditional programming languages
- Compromise: use the Cypress PSoC 5LP, which has FPGA-like hardware that can be configured with a schematic
- Bonus: Cypress PSoC includes a CPU for higher level functionality

Deliverables

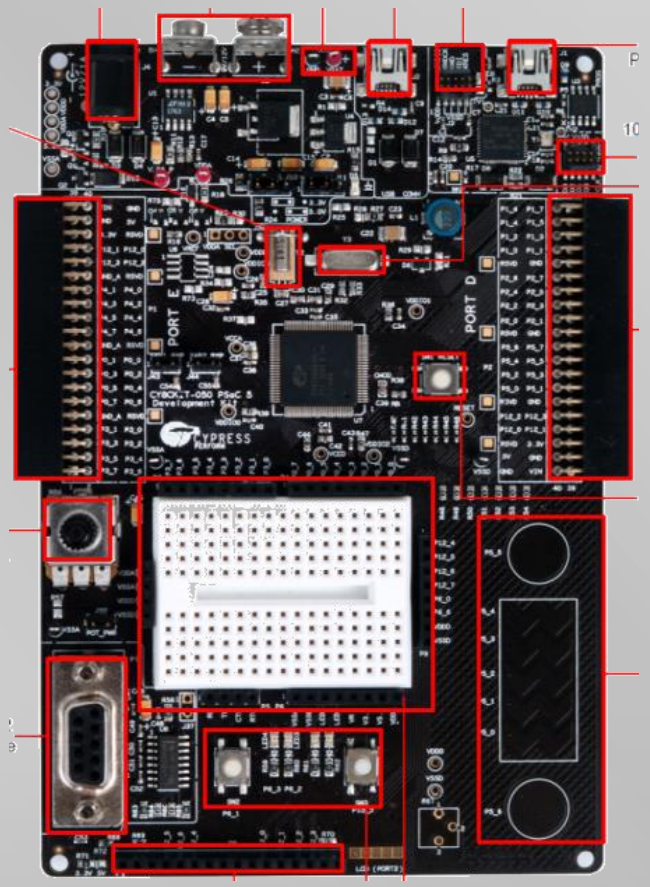
- 6 complete labs
  - Lab Manuals
  - Instructor's Guides
  - Example solution
- 2 incomplete labs; manuals only
  - Contains content removed from labs that were too long
- Final project
  - Project criteria
  - Example project – YM2149 Programmable Sound Generator Emulator
- Instructor introduction
- PSoC handling and safety guide

Cypress PSoC 5LP

- Contains reconfigurable digital and analog hardware
- Digital
  - Fixed function hardware, including four 16-bit timer/counter/PWMs
  - Universal Digital Blocks (UDBs) – 24 highly configurable hardware blocks
- Analog
  - Fixed function hardware, including 3 ADCs, 4 DACs, 4 comparators, and 4 opamps
  - Four programmable analog blocks
  - CapSense – Cypress' capacitive touch solution
- Flexible routing system to connect components within and between digital and analog realms
- Over 60 reconfigurable pins for I/O
- ARM Cortex M3 CPU core with access to digital and analog hardware

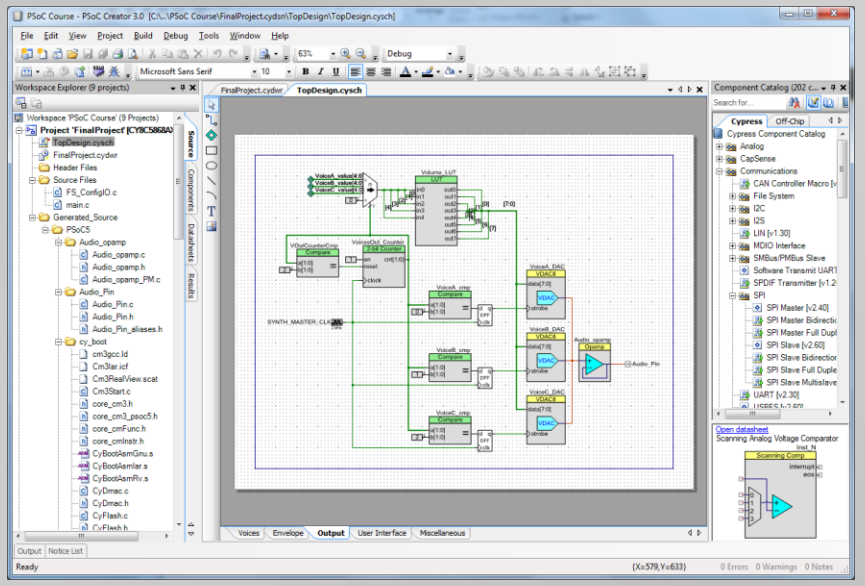
PSoC 5LP Development Kit

- A nearly top of the line PSoC 5LP
- Supporting hardware
- Additional hardware for I/O
  - Pushbuttons
  - LEDs
  - Potentiometer
  - CapSense
  - USB (device)
- Prototyping area (breadboard)
- Other I/O pins accessible around prototyping area and in accessory ports on the sides



PSoC Creator

- The IDE provided to configure and program the PSoC
- For each project:
  - Schematic file for designing the hardware configuration
  - Design-wide resources file
  - Many schematic components generate a software API so the CPU can access them
  - Software programmed in C for the CPU
- Includes programmer to upload project to the PSoC



Labs

- Each lab introduces new hardware components and concepts
- Goals
  - Can be completed in two hours in a laboratory environment
  - Includes sufficient background information to complete without a lecture
  - Is a meaningful learning experience for the students

Labs

- List of labs:
  - Lab 0 – Introduction to PSoC and PSoC Creator
  - Lab 1 – ADC and basic digital I/O
  - Lab 2 – Synchronous circuits, clocks, and timers
  - Lab 3 – Interrupts and debouncers
  - Lab 4 – USBUART and PWM
  - Lab 5 – I^2C
- Incomplete labs made of content removed from labs that were too long
  - Lab 6 – Flip-flops and glitch filters
  - Lab 7 – CapSense

Lab Manuals

- Each lab manual contains three sections:
  - Objectives: the goals of the lab
  - Background: information about the concepts being introduced with focus on details needed to complete the lab
  - Procedure: directions to create an application that uses the concepts introduced
    - Designing the System: details how to configure the hardware
    - Programming the Firmware: details on what to program for the CPU
    - Running the Project: details on how to run the project and how to ensure it works as intended

Instructor’s Guides

- Contains information instructors need to run the lab
- Includes example solutions for each lab
- Includes an introduction for what is necessary for all labs

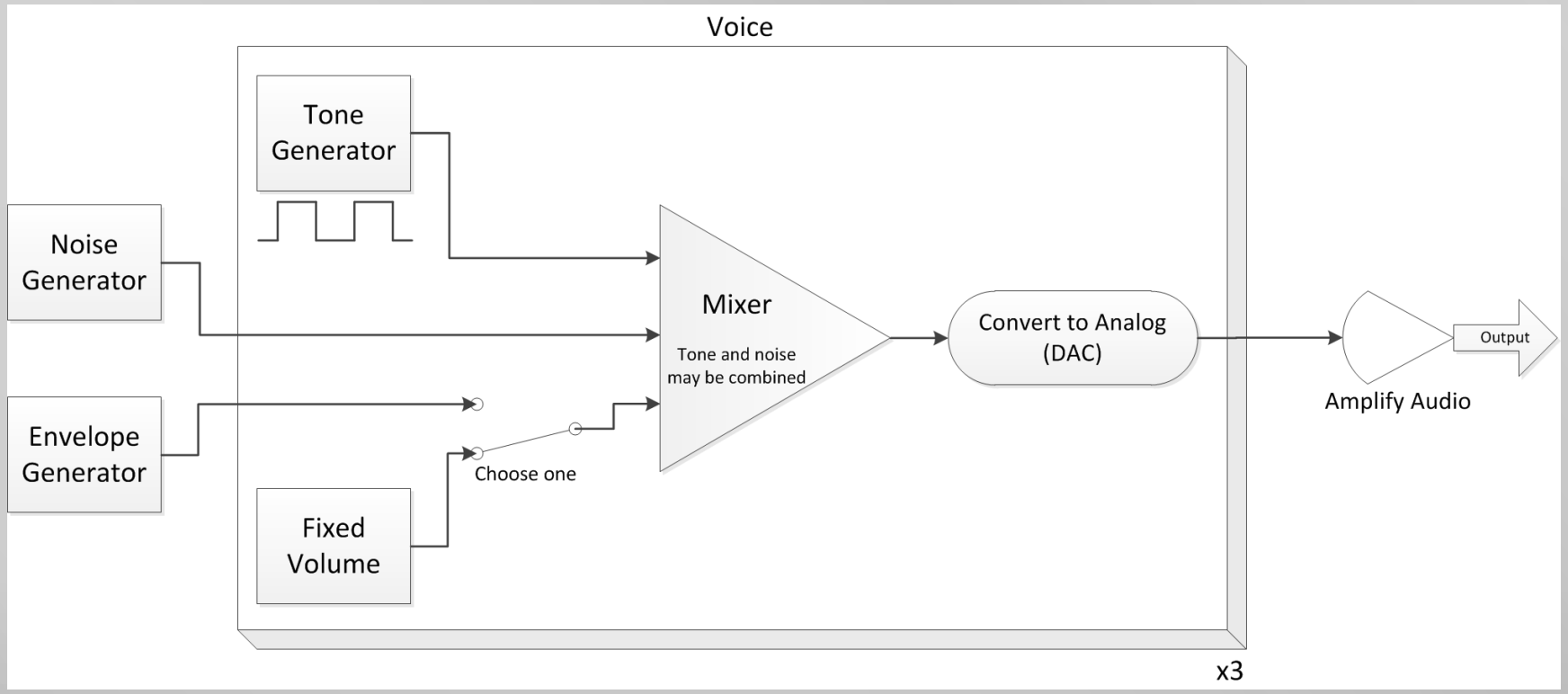
Final Project

- Purpose: demonstrate knowledge of the PSoC 5LP platform by creating a substantial application that uses it
- Criteria for the project involves how many different hardware components are used
  - Each component type is worth a number of points and the project must be worth at least 12 points. See document for more details.
- Using components not taught in the labs is encouraged
- Option to create custom components with schematic macros, UDB editor, or Verilog
- A few general ideas are suggested
- Ultimately up to the student to come up with an idea

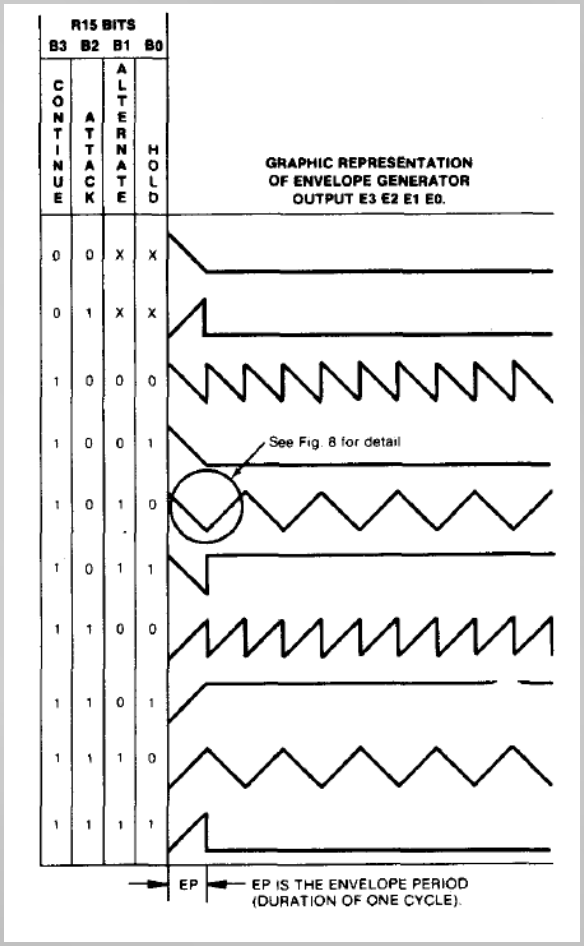
Final Project Example – YM2149 PSG Emulator

- This is an example final project that a student could do
- Worth 20 points based on the criteria
- YM2149 is a programmable sound generator used in microcomputers, such as the Atari ST
- Project plays back Atari ST music by emulating the YM2149 entirely in hardware and using the CPU to update the hardware
- Contains three voices, a noise generator, and a volume envelope generator
- YM2149 uses sixteen 8-bit registers for control
- Music is a series of register dumps that are played back fifty times a second
- Music is buffered off an SD card by the CPU

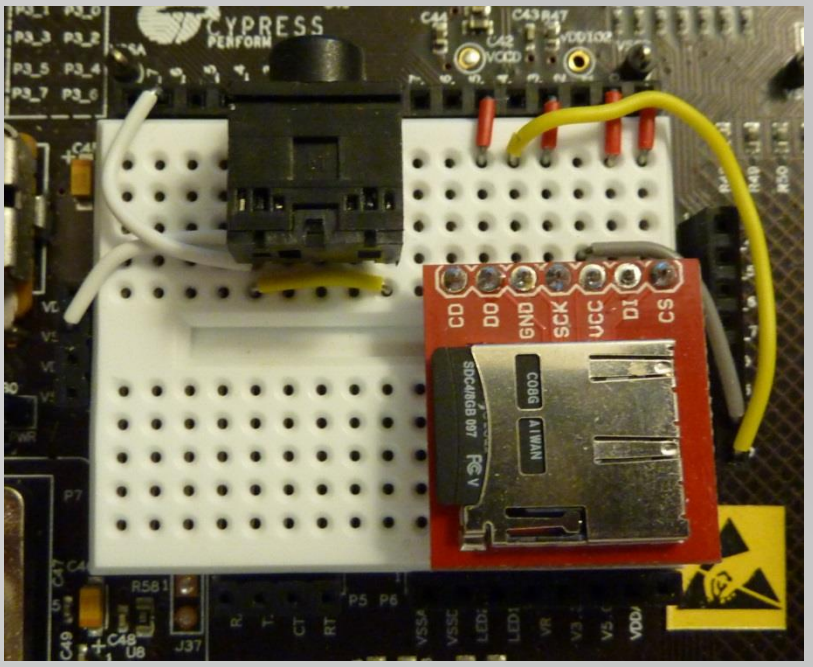
YM2149 PSG Emulator Block Diagram



YM2149 PSG Emulator Envelope Shapes



YM2149 PSG Emulator Demo



Lab Evaluation

- Five students individually evaluated all six completed labs
  - Exception: only three evaluated Lab 5
- Surveys were given
  - Before starting on any lab
  - Before and after each lab
  - After completing all of the labs
- Labs evaluated in weekly two hours sessions, one lab per session
  - Corrected any errors in the manuals as they were found
  - Observed any struggles the students had
  - Exception: Lab 0 was completed, excluding running it, on the students' own time to allow them to set up PSoC Creator
- Two CS, two EE, two SE majors

Lab Evaluation Results

- Labs 0 and 1 well received
  - Possibly add common issues with using the version of C PSoC Creator uses
- Lab 2 took a long time for some students to finish
  - Primary reason: the result of the lab is a stopwatch. Prior labs also provided code to use as a framework
- Lab 3 took too long for most students to finish
  - Primary reason: part of the lab converted the stopwatch in Lab 2 to use interrupts
  - The time consuming portion of the lab was moved to Lab 6
  - Another issue: students were skimming the instructions too much and missing crucial parts

Lab Evaluation Results

- Lab 4 better received, but still had issues
  - CapSense part was moved to Lab 7 to reduce the length of this lab
  - Completed in less than two hours
  - Student suggested making the instructions to make the software less detailed so it is up to the students to figure it out
  - Students seemed to not retain much information from past labs
- Lab 5 was best received
  - Written with feedback from Lab 4 in mind
  - Software section only stated what it should do
  - Students forced to refer to background section to complete the lab; better retention possible
  - Took about two hours to complete; could be faster with lecture beforehand

Improvements

- Modify labs 2-4 to only state how to program the software, rather than giving explicit instructions for how to program it
  - Labs will take longer to complete, but it's more beneficial to learning
- Better to have a lecture before the labs so the students understand what they will be working with
  - Outside of the scope of this Masters project
- Replace the stopwatch application in labs 2 and 3 with something simpler
- Create labs that are mini projects
  - Introduce no new concepts, reinforce concepts already covered
- Create labs that introduce DMA and DACs
- Use more digital logic in an existing or new lab; will be useful for final project
- Expand the total number of labs to 11 to cover an entire semester

Questions