Performance analysis of C/C++ on a FPGA vs Microcontroller

Rainey Anson
d57w762

Skylar Tamke
s96s544

Hongchuan Wang
q14m192

Goals

How efficient is it to run C/C++ code on a FPGA with a soft-core processor compared to using a microcontroller to do the same thing.

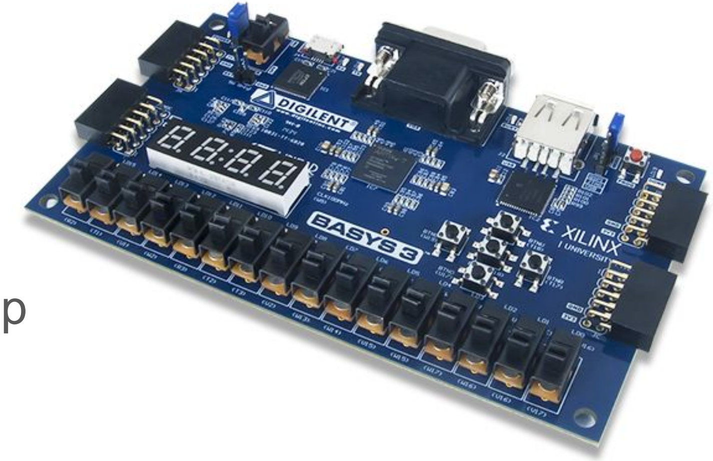
What is an FPGA?

- Field programmable gate array
- Reconfigurable memory
- Can create custom hardware to implement
  - Encryption/decryption
  - Data collection and processing
  - Soft-core processors

Boards

FPGA:

- Digilent Basys 3 - Xilinx Artix-7 35T FPGA chip



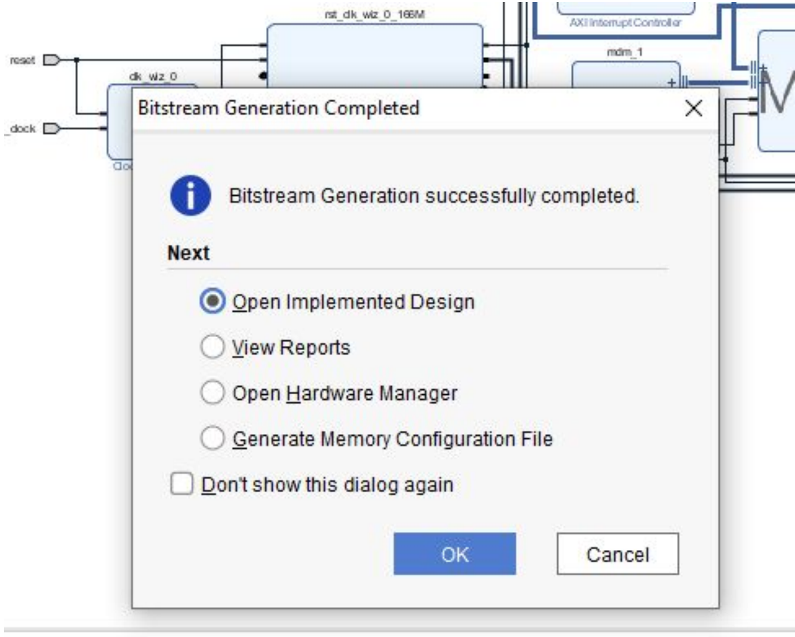
Microcontrollers:

- Atmega328p (uno)
- Atmega2560 (mega)
- Atmega32u4 (feather)

FPGA workflow

XILINX®

- Generate design
  - Drop in micro-blaze IP (Vivado)
  - Add in some kind of memory
  - Generate Bitstream
- Export Hardware to SDK
  - C/C++ code is done in Vitis
  - Need to generate a platform to run the C/C++ code
- Run testbench code and record results
- Compare against microcontroller results



Bitstream Generation Completed

Bitstream Generation successfully completed.

Next

● Open Implemented Design
○ View Reports
○ Open Hardware Manager
○ Generate Memory Configuration File

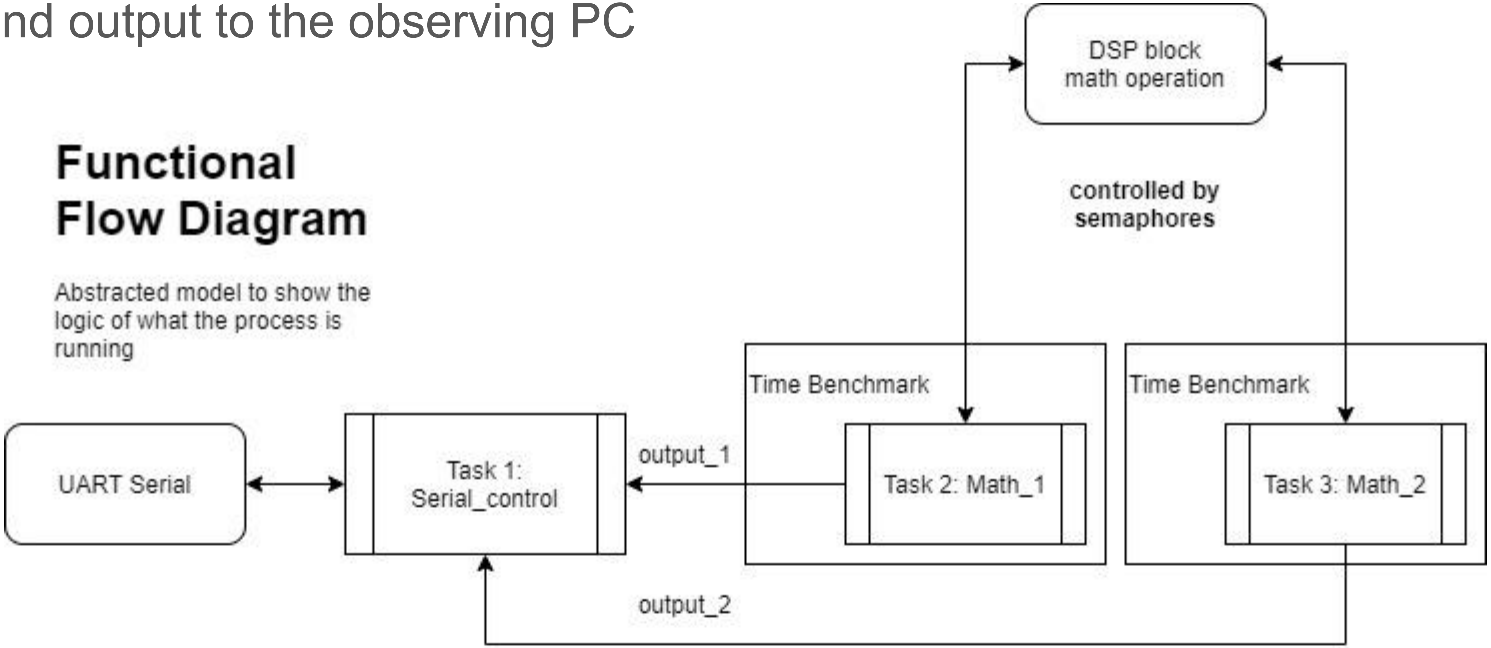
□ Don't show this dialog again

Microcontroller workflow

- Pick a acceptable IDE to write C/C++ code in
  - PlatformIO - VS Code was used to compile and upload code to microcontrollers
- Make sure that this C/C++ code is similar to the FPGA's C/C++ code
- Run code and record results
- Compare against the FPGA results.
  - Need to factor in that the different microcontrollers run at a different clock speed.

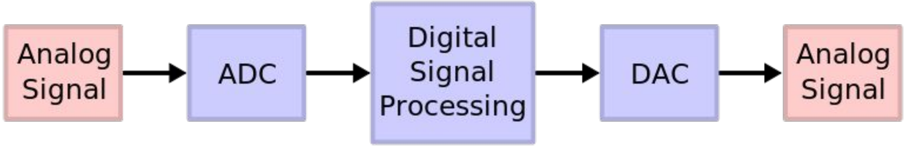
Abstracted design

- Using a seperate Block to do math
- Having the “microcontroller” handle the control of data and output to the observing PC



Functional Flow Diagram

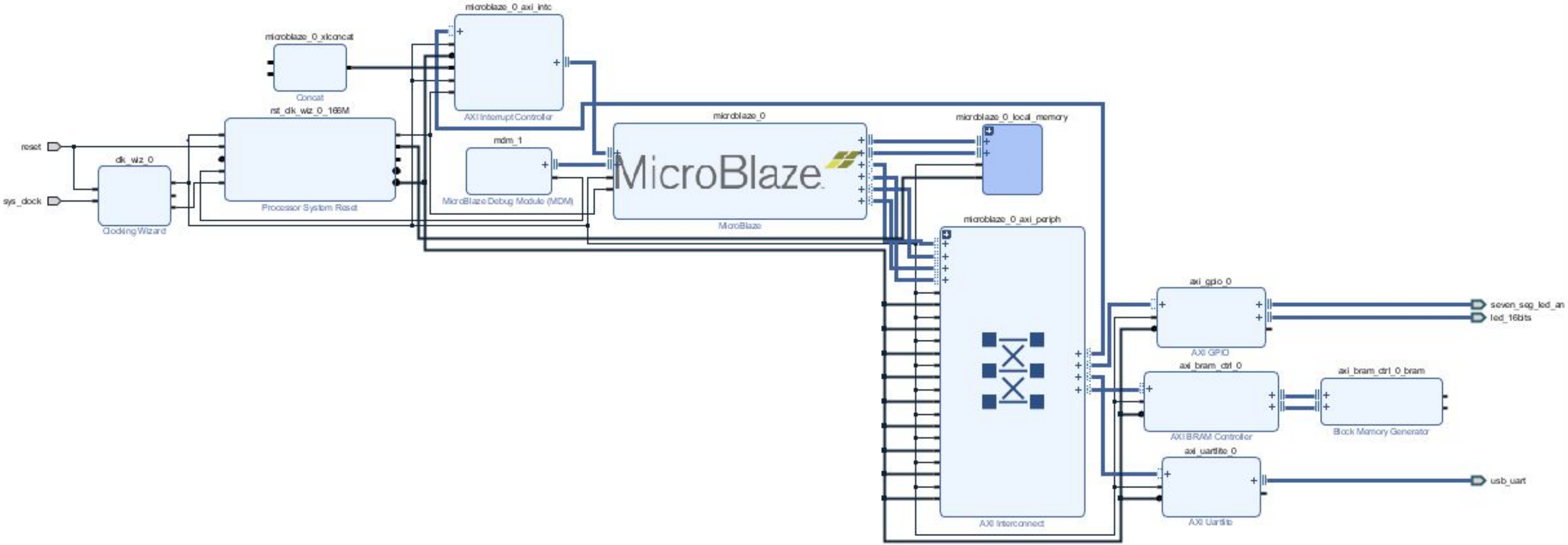
Abstracted model to show the logic of what the process is running



DSP block methodology

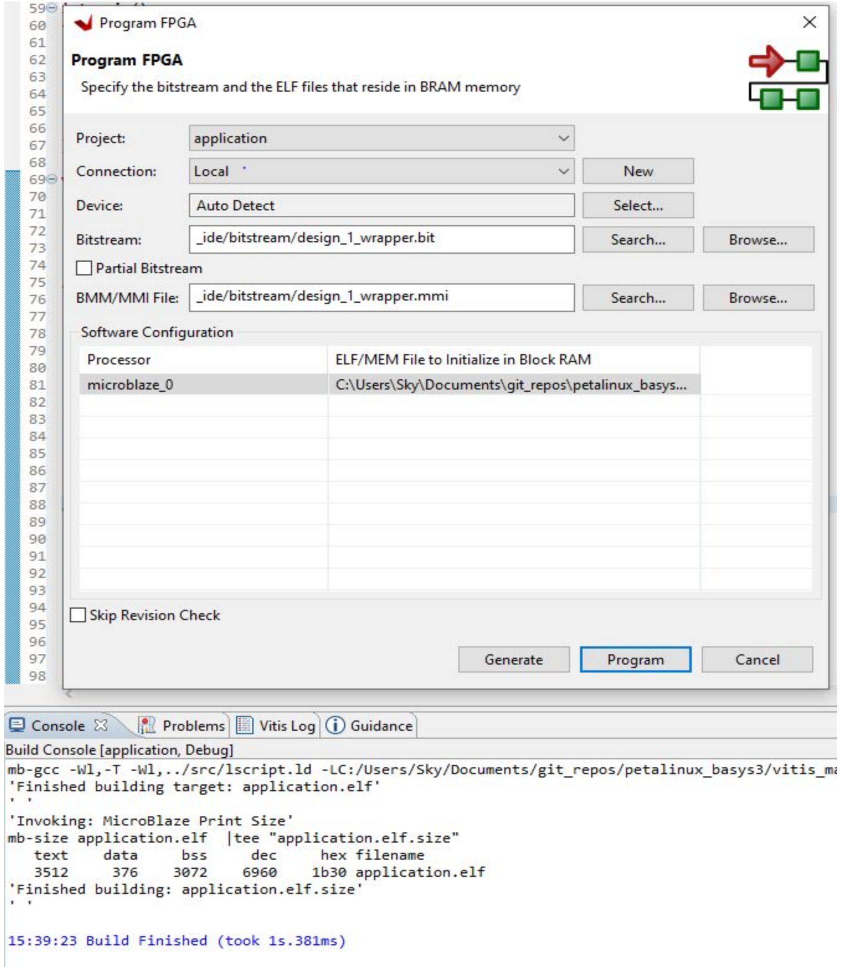
- In FPGA devices and microcontrollers DSP blocks are used to speed up computation time
- Is used in radios to greatly increase speed of signal transmission.
- Used in shaping or identifying waveforms for high speed data

FPGA Block Design



FPGA programming method

- Use Xilinx Vitis SDK to create hardware platform
- Create an application for the hardware platform
- Program FPGA through Vitis with bitstream (hardware config), mmi file (memory location and configuration) and .elf file (software configuration)
- Use SSH terminal to observe data, use python to record and process



Program FPGA

Specify the bitstream and the ELF files that reside in BRAM memory

Project: application

Connection: Local

Device: Auto Detect

Bitstream: _ide/bitstream/design_1_wrapper.bit

□ Partial Bitstream

BMM/MMI File: _ide/bitstream/design_1_wrapper.mmi

Software Configuration

Processor	ELF/MEM File to Initialize in Block RAM
microblaze_0	C:\Users\Sky\Documents\git_repos\petalinux_basys...

□ Skip Revision Check

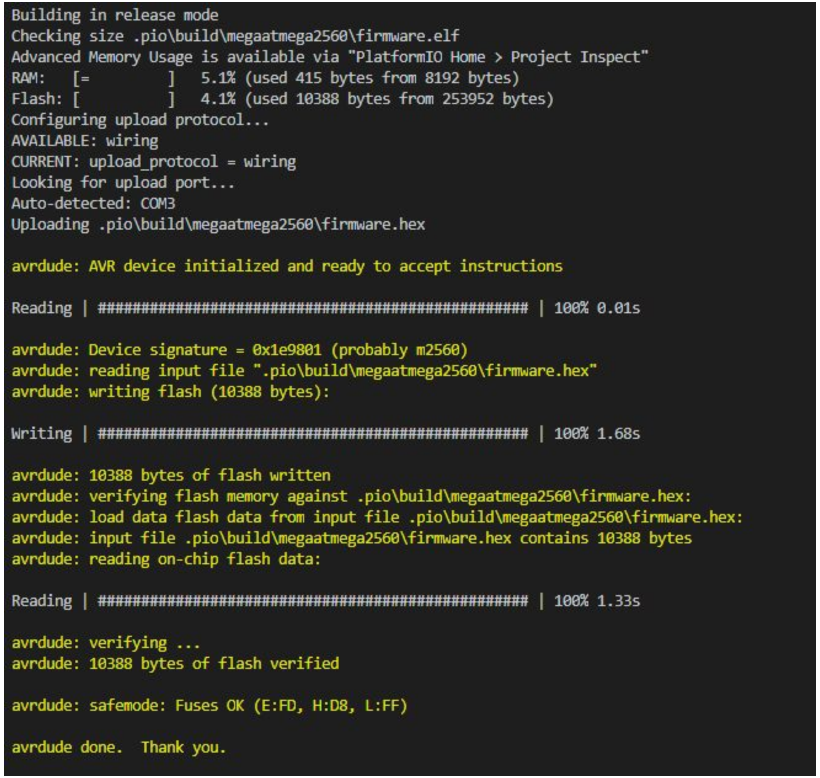
Build Console [application, Debug]

```
mb-gcc -Wl,-T -Wl,../src/lscript.ld -LC:/Users/Sky/Documents/git_repos/petalinux_basys3/vitis_ma
'Finished building target: application.elf'
' '
'Invoking: MicroBlaze Print Size'
mb-size application.elf  |tee "application.elf.size"
   text    data     bss     dec     hex filename
   3512     376    3072    6960    1b30 application.elf
'Finished building: application.elf.size'
' '

15:39:23 Build Finished (took 1s.381ms)
```

Microcontroller programming method

- Using PlatformIO to build and upload program to microcontrollers
    - Ability to link to other libraries easily
- Create a seperate project for each microcontroller to simplify jumping between them.



```
Building in release mode
Checking size .pio\build\megaatmega2560\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [=         ]   5.1% (used 415 bytes from 8192 bytes)
Flash: [          ]   4.1% (used 10388 bytes from 253952 bytes)
Configuring upload protocol...
AVAILABLE: wiring
CURRENT: upload_protocol = wiring
Looking for upload port...
Auto-detected: COM3
Uploading .pio\build\megaatmega2560\firmware.hex

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.01s

avrdude: Device signature = 0x1e9801 (probably m2560)
avrdude: reading input file ".pio\build\megaatmega2560\firmware.hex"
avrdude: writing flash (10388 bytes):

Writing | ################################################## | 100% 1.68s

avrdude: 10388 bytes of flash written
avrdude: verifying flash memory against .pio\build\megaatmega2560\firmware.hex:
avrdude: load data flash data from input file .pio\build\megaatmega2560\firmware.hex:
avrdude: input file .pio\build\megaatmega2560\firmware.hex contains 10388 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 1.33s

avrdude: verifying ...
avrdude: 10388 bytes of flash verified

avrdude: safemode: Fuses OK (E:FD, H:D8, L:FF)

avrdude done.  Thank you.
```

Data collection

- Will collect data from the different boards using the python serial library. This will allow for easy data processing afterwards
- Pycharm IDE was used to program and test code on this end.
- Excel was used for quick processing the data afterwards.

Quick Demo

Wait for a sec….

PlatformIO demo is better for quick demonstration

Results of Computation speed

Board	Completion time of Task 1	Completion time of Task 2	Completion Performance (unitless)
AtMega2560	0.5286404494	1.421825843	0.03304002809
AtMega328p	0.5152314657	1.371265467	0.03220196661
Atmega32u4	0.5310246131	1.438578384	0.03318903832
Micro-blaze(FPGA)	0.6521348421	1.682132457	0.04075842763

- All clocks at 16MHz
- Different Toolchain
- Completion time of Tasks averaged over a decent sample size divided by the clock speed to normalize the output values.

Discussion of Results

- For situations where the project needs be done quickly this microcontrollers might be the best option
- In situations where hardware needs to change out for different processes the FPGA method might be best
- Microcontrollers connected to FPGA fabric seems to be the industries preference. Combining both worlds in an efficient manner.

Fin