Hands on with FPGA's: Module 8

Venkat Rangan

Questions on Module 7

- Example of an end-end vision project using an FPGA: autonomous drone landing
  - Could this be modified for your use cases?

Topics

- Pre-class: Open floor for questions
  - Module 7
- Guest Lecture: Dave Shah
- APIO, LiteX: River, HW Intern @ tinyVision.ai
- Beyond Verilog:
  - High Level Synthesis
  - CHISEL
- When to use FPGA's?
- Open discussion

Architecture Exploration: A FIR filter

- FIR filter is a common design in signal processing

- Design variables:
  - How many multipliers?
  - What clock speed do we want?
  - Design reuse:
    - Extendability for # of taps
    - Bitwidths
    - Signed vs. unsigned

```
int fir_filter(int xn) {
static int h[5] = {1,2,3,2,1};
// delay line of time samples
static int xd[5] = {0,};
// filter output
int yn = 0;
// implementation of delay line
xd[4] = xd[3];
xd[3] = xd[2];
xd[2] = xd[1];
xd[1] = xd[0];
xd[0] = xn;
// convolve delay line by // filter coefficients
for(int i=0;i<5;i++)
          yn += h[i]*xv[i];
return yn;
}
```

HLS Tools

- What is HLS
- Why HLS?
  - Use C/Python as a HDL
  - Interface synthesis: map I/O to stream/register/SPI etc. automatically
  - Architecture exploration
  - Fast verification

- Example: Streamlogic.io
  - HLS based video processing example
  - Hardware-as-code series

Other HDL's…

- CHISEL: Commercial RISCv processors written using this HDL
  - See Chipyard for a detailed project
  - Concept of generators vs. instances
    - Also somewhat applicable to plain RTL designs!!!

  - FIR Filter generator example using Chisel
    - Generate Verilog using:

```
import chisel3.stage.{ChiselStage, ChiselGeneratorAnnotation}
val verilogString = chisel3.stage.ChiselStage.emitVerilog(new FirFilter(4, Seq(0.U, -1.S, 2.U, 5.U)) )
println(verilogString)
```

    - Moving average 13 bit filter, 4 taps:
      - `new FirFilter(13, Seq(1.U, 1.U, 1.U, 1.U))`
    - Triangular, 8 bit filter, 5 taps:
      - `new FirFilter(8, Seq(1.U, 2.U, 3.U, 2.U, 1.U))`

- Many others out there!

FPGA's: when/where?

	FPGA	Microcontroller	Comments
Ease of programming	Difficult: you're starting here!	Easy for simple applications	Some problems lend themselves well to FPGA's vs. micro and vice-versa
Cost	Low to very high	Usually low	Wide range here for FPGA's!
Toolchains/IDE's	Lagging…	Well developed	
Flexibility/Parallelism	Very high	Limited	Micros are good enough until they aren't ☺
Energy Efficiency	Depends	Depends	FPGA's work well for compute intensive tasks, micros are good for going to sleep and doing work occasionally

Many other criteria possible, very application specific!

Be very aware of pros/cons of each when choosing solutions

Module 8:

- Challenge:
  - Install Chisel and the Chipyard on your computer (Beware, chipyard can take a loong time!!!) and play with it.
  - Install Amaranth and apply some of the concepts you learnt in the class. You may want to try something simple like a counter at first and then start to stitch up more complex things like a CPU.
  - Get CFU Play installed on your machine and go through the accelerator design. You should now have enough background to start working with this complex project.

Open Discussion