FPGA & SoC-based Qubit Control

Quantum Journal Club

Sara Sussman

Houck Lab at Princeton

February 4, 2020
How are FPGA designs made?

- FPGA designs are either written in a hardware description language (HDL) or as a schematic design. **Verilog** and **VHDL** are examples of HDLs.

- We write RTL (register transfer level) code—which describes circuits—in a HDL. Later, the synthesis tool converts the RTL code into a gate level description or even a macro (this is device-dependent).

- Example: this line of RTL (written in Verilog) describes a multiplexer:

  ```
  assign mux_out = (sel) ? din_1 : din_0;
  ```

  The synthesis tool converts this RTL into a macro with the same inputs and outputs:

  ```
  mux u3 (mux_out, din_1, din_0);
  ```
Programming FPGAs

```verilog
module Mixing {

// ///////// ADC /////////
inout ADC_CS_N ,
output ADC_DIN ,
input ADC_DOUT ,
output ADC_SCLK ,

// ///////// ADC /////////
inout AUD_ADCDAT ,
inout AUD_ADCLRCK ,
inout AUD_BCLK ,
output AUD_DACDAT ,
inout AUD_DACLRCK ,
output AUD_XCK ,

}
```
Why doesn’t everyone program FPGAs?

Well, there is no debugger. There is almost zero visibility in an FPGA design. Documentation is scarce.

Princeton undergrads take ELE 206 which involves some of this.

Maybe you yourself remember taking a course like this!

Figure courtesy of ZipCPU.
Almost zero visibility: exhibit A

- Documentation tends to look like this.
- Figure courtesy of “Exploring Zynq MPSoC”.

Figure 1.2: Simplified Design Flows for Working with Zynq MPSoC
(left: conventional ‘hardware/software’ design flow; right: ‘software defined’ design tool, using SDs)
Experts always build a simulator for the device their FPGA will interface with. This allows their simulated logic to think it is talking to the actual device.

Then, they build a test bench that exercises their logic against the simulator.

Co-simulation is a must and they use formal methods to find bugs.

Verilog is not a programming language!

Figure courtesy of ZipCPU.
FPGA workflow- reality...

- Experts have a powerful workflow.
- They have each built their own scaffolding of how to communicate with the FPGA and how to get diagnostics from the FPGA.
- The scaffolding involves simulators and formal verification.
- It’s a process!
- Figure courtesy of ZipCPU.
What is a System on Chip?

- A SoC is a Processing System (PS), coupled with FPGA Programmable Logic (PL). The two sections are connected via a number of Advanced eXtensible Interface (AXI) interfaces.
- You own ASIC-based SoCs. They are inside your PCs, tablets, and smartphones. Each has at least two processor cores, memory, graphics, interfacing, and other functions...
- Figure courtesy of “Exploring Zynq MPSoC”.

Figure 1.1: A simplified diagram of the Zynq MPSoC architecture
What can SoCs do?

- When we say SoC we actually mean “System on Programmable Chip”, a SoC implemented on a programmable, reconfigurable device (the FPGA). System upgrades are easily managed since reprogramming a SoC is (almost) as easy as rewriting an SD card.

- Applications include
  - Advanced Driver Assistance Systems (ADAS)
  - Computer vision
  - Big data analytics
  - Software Defined Radio (SDR)
  - AI (e.g. Baidu’s AI EdgeBoard, 2019)
  - 5G massive MIMO antenna systems
  - Qubit control!
SoCs on the market

- American SoC companies are Xilinx (60% of market) and Intel (40% of market, recently bought Altera). Xilinx clients include Alibaba, Amazon, Microsoft. Global semiconductor market worth over $400 billion in 2017, its highest ever level, and an increase of over 20% compared to the previous year.

- 1985: Xilinx invents the FPGA.

- 2011: Xilinx releases Zynq-7000 SoC, which integrates a complete ARM Cortex-A9 MPCore processor-based system on a 28 nm FPGA. Eval board $300.

- 2014: Xilinx releases Zynq UltraScale MPSoC (multi-core system-on-chip), which integrates a multi-processor system on a 20 nm FPGA. Up to 5x system level performance per watt over Zynq-7000 SoCs. Eval board $300.

Slow Integrated RF SoC board: the Red Pitaya

- **Fast analog outputs**: 125 MS/s, 14 bits, ±1 V
- **Fast analog inputs**: 125 MS/s, 14 bits, ±1 V
- **Digital extension**: 16 I/O ch., 125 MS/s, 3.3 V
- **Analog extension**: ~100 kS/s, ~12 bits, 0-1.8 V
  - 4 SAR ADC ch., 4 PWM DAC ch.
- **Microprocessor + FPGA**
  - Dual core ARM-Cortex A9 Zynq SoC
- **DDR3 RAM**: 512 MB
- **Daisy chain**
- **microSD**: OS drive & FPGA design
- **Ethernet**: 1 Gb/s
- **USB OTG**
- **micro USB**: Power in 5 V, 2 A

FPGA & SoC-based Qubit Control

Sara Sussman

FPGAs and SoCs

RF SoCs

...As a platform for qubit control!

FPGA-based DACs

FPGA-based ADCs

Why not DAC and ADC!

SoC-based CNNs
RF IO specs

- **Red Pitaya (slow DAC/ADC and Zynq-7000)**
  - DAC
    - Sample rate: 125 MS/s, Resolution: 14 bits
  - ADC
    - Sample rate: 125 MS/s, Resolution: 14 bits
- **Typical qubit control setup**
  - DAC (M9330A)
    - Sample rate: 1.25 GS/s, Resolution: 15 bits
  - ADC (U1084A)
    - Sample rate: 2 GS/s, Resolution: 8 bits
- **RFSoC (fast DAC/ADC and Zynq Ultrascale+ MPSoC)**
  - DAC
    - Sample rate: 10 GS/s, Resolution: 14 bits
  - ADC
    - Sample rate: 5 GS/s, Resolution: 14 bits
FPGA & SoC-based Qubit Control

Sara Sussman

FPGAs and SoCs

RF SoCs

...As a platform for qubit control!

FPGA-based DACs
FPGA-based ADCs
Why not DAC and ADC!
SoC-based CNNs

Red Pitaya DAC schematic
Harmonics: -50 dBc for frequencies 1-45 MHz at 8 dBm.
FPGA & SoC-based Qubit Control

Sara Sussman

FPGAs and SoCs

RF SoCs

...As a platform for qubit control!

FPGA-based DACs
FPGA-based ADCs
Why not DAC and ADC!

SoC-based CNNs

Red Pitaya ADC schematic
Red Pitaya ADC bandwidth (50 MHz)

From Red Pitaya documentation.
Red Pitaya ADC: used interpolation to get more accurate results of Vpp (above 10 MHz)

From Red Pitaya documentation.
FPGA & SoC-based Qubit Control

Sara Sussman

FPGAs and SoCs

RF SoCs

As a platform for qubit control!

FPGA-based DACs
FPGA-based ADCs
Why not DAC and ADC?
SoC-based CNNs

Agilent ESA vs. Red Pitaya ADC: digitizing a 30 MHz, 0 dBm signal

From Red Pitaya documentation.
Red Pitaya in action: “Active cancellation of acoustical resonances with an FPGA FIR filter”

- Used Red Pitaya to create a digital FIR filter which cancels numerous high-Q mechanical resonances, with a built-in network analyzer for characterizing its performance. “The last 25600 values of a sampled input signal from an analog to digital converter (ADC) are multiplied by 25600 user-defined filter coefficients, and the results are summed and sent to a digital to analog converter (DAC).”

We employ a filter with \( J_{\text{max}} = 25600 \) coefficients (17-bit). The essential building block is the Multiply-ACcumulator (MAC), a physical (non-reconfigurable) component of an FPGA which can, within a single clock cycle, perform a signed, fixed-point multiplication, along with both pre- and post-additions. The filter is implemented using \( N_{\text{MAC}} = 50 \) MACS, operating in parallel, each realized in a single DSP48 slice in the FPGA, and each capable of carrying out \( n_{\text{op}} = 512 \) serial multiplications and additions within one sample period. This implementation yields a sampling rate \( f_s = 243 \ \text{kHz} \approx \frac{f_{\text{clock}}}{n_{\text{op}}} \) and a delay of \( \tau = 2.6 \ \mu\text{s} \approx \frac{1}{2f_s} + \frac{N_{\text{MAC}}}{f_{\text{clock}}} \) where the FPGA’s clock rate is \( f_{\text{clock}} = 125 \ \text{MHz} \). See Figure 2 for a diagram of the FIR implementation.

The FPGA is a commercial Red Pitaya board (redpitaya.com, ~$240 at the date of publication)
Red Pitaya in action: “Active cancellation of acoustical resonances with an FPGA FIR filter”

- "The analog feedback controller $K$ (typically proportional plus integral gain) takes the difference between the filtered signal and the reference $r$ and feeds it back into the system to stabilize it."
- "Magnitude of KGF before (blue) and after (red) implementing the FIR filter to cancel the resonances and increasing the total gain."
“Scalable and customizable AWG for superconducting quantum computing”

- Jin Lin et al., AIP Advances 9, 115309 (2019)
- Used a Xilinx Kintex UltraScale FPGA with a MicroBlaze soft processor core to create a customizable AWG.
  - Context: Before Zynq came along with an ARM processor, users used a soft core processor such as Microblaze, (released in 2009). The main advantage of using Microblaze was (and still is) the flexibility of the processor instances within a design. On the other hand, Zynq’s hard processor delivers significant performance improvements.
- DAC sample rate: 2 GS/s, Resolution: 16 bits
- “This customizable AWG has been used in several superconducting quantum processors, and the result of multiple qubits’ measurement verifies that the AWG is qualified for controlling tens of superconducting qubits.”
“Scalable and customizable AWG for superconducting quantum computing”
“Scalable and customizable AWG for superconducting quantum computing”

- This AWG was used in several impressive qubit experiments coming out of USTC in 2019.
- “Propagation and Localization of Collective Excitations on a 24-Qubit Superconducting Processor” Phys. Rev. Lett. 123, 050502 – Published 30 July 2019. “On a 24-qubit superconducting processor, 80 AWG channels are used to control this processor, and the average single-qubit gate fidelity is 0.995.”
- “Genuine 12-Qubit Entanglement on a Superconducting Quantum Processor” Phys. Rev. Lett. 122, 110501 – Published 20 March 2019. “In the genuine 12-qubit entanglement experiment where 40 AWG channels are used, the average single-qubit gate fidelity of 12 qubits is 0.998.”
“Realizing Rapid, High-Fidelity, Single-Shot Dispersive Readout of Superconducting Qubits”

- They implemented a FPGA-based 8 ns ADC with hardware averaging such that they could discriminate between qubit ground and excited states within 50 ns with over 98% fidelity.
- “The amplified quadrature of the JPD is chosen to maximize the contrast between the mean ground- and excited-state responses...the ground- and excited-state responses can be clearly distinguished in a single shot of a measurement.”
“Stabilizing Rabi oscillations in a superconducting qubit using quantum feedback”

- They implemented some kind of feedback controller which has DAC and ADC functionality.
- Ran out of time but it’s a great paper. Wish they gave more details on their hardware.
“Comparing and Combining Measurement-Based and Driven-Dissipative Entanglement Stabilization”

- Xilinx Virtex 6-based system capable of controlling multiple qubits that integrated digitizer, demodulator, state estimator and AWG functionality onto a single chip.
- Yehan Liu’s thesis has the details!
“Fast and High-Fidelity Readout of Single Trapped-Ion Qubit via Machine Learning Methods”

- The only paper (so far) on the arXiv that uses a programmable SoC for qubit control!
- Zynq-7000 SoC-based neural network mediated active reset, obtaining 99.5% readout fidelity within 171 us.