FPGA Design

Dr. Paul D. Franzon

Outline
1. Architectural Features of FPGAs
2. Design techniques specific to FPGAs

References
1. Xilinx Virtex5 documentation
2. S. Kilts, “Advanced FPGA Design”
FPGA Architecture

Using Xilinx Virtex5 to focus discussion

- But many other FPGAs have similar features

Architectural Resources:

- Logic Blocks
  - Logic Slices
  - Two logic slices per Configurable Logic Block

- Programmable Switch matrix
  - Programmable interconnect to connect CLBs and other resources
Logic (M) Slice

Look UP Tables /RAM
- Up to 256 bits RAM
- Shift Register

Steering Logic
- Incl. Carry Chain

Storage
Logic (D) slice
LUTs only
Other Hardware Logic Resources

Varies by FPGA
- Virtex5 here:

- 36-Kbit block RAM/FIFOs
  - True dual-port RAM blocks
  - Enhanced optional programmable FIFO logic
  - Programmable
    - True dual-port widths up to x36
    - Simple dual-port widths up to x72
  -Built-in optional error-correction circuitry
  - Optionally program each block as two independent 18-Kbit blocks

- Advanced DSP48E slices
  - 25 x 18, two’s complement, multiplication
  - Optional adder, subtracter, and accumulator
  - Optional pipelining
  - Optional bitwise logical functionality
  - Dedicated cascade connections

- PowerPC 440 Microprocessors
  - FXT Platform only
  - RISC architecture
  - 7-stage pipeline
  - 32-Kbyte instruction and data caches included
  - Optimized processor interface structure (crossbar)
Logic Level Design

Tune design to make best use of available resource (Virtex 5 used to illustrate)

- Use 18-bit integer Multipliers instead of wider of FP multipliers
  - Use design techniques to turn these into emulators of other multipliers if needed
    - E.g. Use multiple 18-bit multiplies to make larger multiplies
    - Track exponent and mantissa separately to emulate Floating Point
- Use available RAM options by constraining code size
  - Small RAMs in CLBs
  - Large RAM blocks
  - Make sure your RAMs fit to the available options
- Embed SW in PowerPC CPUs where appropriate
- Shift registers, carry ripple arithmetic
- Not a lot of embedded SRAM – be efficient with it

Use vendor provided IP where appropriate

- DRAM interfaces, Ethernet interfaces, etc.
Clock and IO Resources

Clock

- Resources for multiple clock domains and clock gating
- Use Xilinx provided Verilog templates to enable

IO

- Rocket IO Serializer/Deserializer provides high bandwidth inter FPGA communications channel
- Flip-flops in IO should be used at interfaces
- IO very flexible
Programmable Interconnect

Logic Resources interconnected via a programmable interconnect structure

- Interconnect resources are
  - Limited in capacity
  - Slow, especially over long distances
  - Consumes a lot of power (high capacitance due to switches)

Figure 3: Switch box design for conventional SRAM-based FPGAs. The left option provides slightly less area and power consumption compared to the right one but signal propagation delay increases.
Making Effective Use of Interconnect

Logic Design
- Favor designs and algorithms that favor logic over wiring
  - E.g. Build Fast Fourier Transform (FFT) as a pipeline rather than as a Butterfly network

Partitioning and Floorplanning
- Try to create partitions that favor nearest neighbor interconnect
  - E.g. A high level pipeline that transfers data block to block
  - I.e. Create a dataflow structure and try to layout floorplan to match dataflow
    - Keep interblock transfers to parallel blocks
      - Place pipeline stages next too each other in floorplan
- Put logic that uses fixed resources (RAM, DSP) next to that resource
- Keep critical paths within one block AND close together when floorplanning
  - High fanout nets are more likely to be critical than in a standard cell asic
Tailor your design to the FPGA

Xilinx Coolrunner

- Includes double edge triggered flip-flops so that clock distribution can be done at half normal rate
  
  \[
  \text{always}@(\text{posedge clock or negedge clock})
  \]

  - Halves power in clock distribution

Consider Synchronous Reset

- Cores in Xilinx parts (e.g. DSP core) use synchronous reset, so specifying asynchronous reset requires adding FPGA resources external to core

Use a pair of tri-states for some muxes

- Certain Xilinx FPGAs have lots of tri-state buffers

Use resources provided for clock gating

- E.g. Xilinx global clock mux, BUFGMAX

As resources utilization approaches 100% relax speed targets

- Otherwise you will run out of hardware resources as you add functionality
- A relaxation of about 20% of clock speed might free up substantial resources
Summary

What is the main guideline to make efficient use of logic resources available on FPGAs?

What is the main guideline to minimize interconnect delay and power in an FPGA?