Using FPGAs in the real world

- Performance boosting techniques.
- Floating point designs.
- Powering FPGAs.
- Interfacing to the outside world.
- Clock domains and metastability.
- Safe design and radiation hardness.
Using FPGAs in the real world

- Performance boosting techniques.
- Floating point designs.
- Powering FPGAs.
- Interfacing to the outside world.
- Clock domains and metastability.
- Safe design and radiation hardness.
Reminder: basic digital design

High clock rate: 144.9 MHz on a Xilinx Spartan IIE.

Higher clock rate: 151.5 MHz on the same chip.
Buffering

- Delay in modern designs can be as much as 90% routing, 10% logic. Routing delay is due to long nets + capacitive input loading.
- Buffering is done automatically by most synthesis tools and reduces the fan out on affected nets:

Before buffering

After buffering
Replicating registers (and associated logic if necessary)
Retiming (a.k.a. register balancing)

Before

Large combinational logic delay

Small Delay

After

Balanced delay

Balanced delay
Pipelining

Before

Large combinational logic delay

After

Small delay

Small delay

Small delay
Time multiplexing

De-multiplexer

50 MHz logic

50 MHz logic

50 MHz logic

50 MHz logic

50 MHz logic

50 MHz logic

50 MHz logic

Multiplexer

Data In

100 MHz

50 MHz

Data Out
An example: boosting the performance of an IIR filter (1/2)

Simple first order IIR: $y[n+1] = ay[n] + b x[n]$

Problem found in the phase filter of a PLL used to track bunch frequency in CERN’s PS

Performance bottleneck in the feedback path
An example: boosting the performance of an IIR filter (2/2)

Look ahead scheme:
From $y[n+1] = ay[n] + b x[n]$ we get
Another example: being smart about what you need exactly.

- \( u \times v = u_x v_y - u_y v_x \)
- \(|u \times v| = |u| \times |v| \sin \theta = \varepsilon \text{ } IcFwd\)
- \( u = \text{Vacc}, v = \text{IcFwd}\)

Cross Product for 2D vectors \(U(x,y) & V(x,y)\)

Cross product used as phase discriminator by John Molendijk in the LHC LLRF.
Using FPGAs in the real world

- Performance boosting techniques.
- Floating point designs.
- Powering FPGAs.
- Interfacing to the outside world.
- Clock domains and metastability.
- Safe design and radiation hardness.
Floating point designs

To work in floating point you (potentially) need blocks to:
- Convert from fixed point to floating point and back.
- Convert between different floating point types.
- Multiply.
- Add/subtract (involves an intermediate representation with same exponent for both operands).
- Divide.
- Square root.
- Compare 2 numbers.

The main FPGA companies provide these in the form of IP cores. You can also roll your own.
s: sign.
e: exponent.
f: fractional part \( b_0 . b_1 b_2 b_3 b_4 \ldots b_{w_f-1} \)
Convention: normalized numbers have \( b_0 = 1 \)

Exponent value:
\[
E = e - (2^{w_e-1} - 1)
\]

Total value:
\[
v = (-1)^s 2^E b_0 b_1 b_2 \ldots b_{w_f-1}
\]

IEEE-754 standard single format: 24-bit fraction and 8-bit exponent (\( w=32 \) and \( w_f=24 \) in the figure).
IEEE-754 standard double format: 53-bit fraction and 11-bit exponent.
Some performance figures (single precision)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Resources</th>
<th>Maximum Frequency (MHz)¹</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Embedded</td>
<td>Fabric</td>
</tr>
<tr>
<td></td>
<td>Type</td>
<td>Number</td>
</tr>
<tr>
<td>Multiply</td>
<td>DSP48E (max usage)</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>DSP48E (full usage)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>DSP48E (medium usage)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>0</td>
</tr>
<tr>
<td>Add/Subtract</td>
<td>DSP48E (speed optimized, full usage)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Logic (speed optimized, no usage)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Logic (low latency)</td>
<td>0</td>
</tr>
<tr>
<td>Fixed to float</td>
<td>Int32 input</td>
<td></td>
</tr>
<tr>
<td>Float to fixed</td>
<td>Int32 result</td>
<td></td>
</tr>
<tr>
<td>Float to float</td>
<td>Single to double</td>
<td></td>
</tr>
<tr>
<td>Compare</td>
<td>Programmable</td>
<td></td>
</tr>
<tr>
<td>Divide</td>
<td>C_RATE=1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C_RATE=26</td>
<td></td>
</tr>
<tr>
<td>Sqrt</td>
<td>C_RATE=1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C_RATE=25</td>
<td></td>
</tr>
</tbody>
</table>

¹. Maximum frequency obtained with map switches -ol high and -cm speed, and par switches -pl high and -rl high.
Some performance figures (double precision)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Resources</th>
<th>Embedded</th>
<th>Fabric</th>
<th>Virtex-5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Type</td>
<td>Number</td>
<td>LUTs</td>
</tr>
<tr>
<td>Multiply</td>
<td>DSP48E (max usage)</td>
<td>13</td>
<td>416</td>
<td>654</td>
</tr>
<tr>
<td></td>
<td>DSP48E (full usage)</td>
<td>12</td>
<td>424</td>
<td>669</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>0</td>
<td>2,309</td>
<td>2,457</td>
</tr>
<tr>
<td>Add/Subtract</td>
<td>DSP48E (speed optimized, full usage)</td>
<td>3</td>
<td>821</td>
<td>976</td>
</tr>
<tr>
<td></td>
<td>Logic (speed optimized, no usage)</td>
<td>0</td>
<td>804</td>
<td>1,060</td>
</tr>
<tr>
<td></td>
<td>Logic (low latency, no usage)</td>
<td>0</td>
<td>1,045</td>
<td>1,185</td>
</tr>
<tr>
<td>Fixed to float</td>
<td>Int64 input</td>
<td></td>
<td>402</td>
<td>504</td>
</tr>
<tr>
<td>Float to fixed</td>
<td>Int64 result</td>
<td></td>
<td>396</td>
<td>446</td>
</tr>
<tr>
<td>Float to Float</td>
<td>Double to single</td>
<td></td>
<td>107</td>
<td>131</td>
</tr>
<tr>
<td>Compare</td>
<td>Programmable</td>
<td></td>
<td>142</td>
<td>24</td>
</tr>
<tr>
<td>Divide</td>
<td>C_RATE=1</td>
<td></td>
<td>3,228</td>
<td>6,002</td>
</tr>
<tr>
<td></td>
<td>C_RATE=26</td>
<td></td>
<td>354</td>
<td>399</td>
</tr>
<tr>
<td>Sqrt</td>
<td>C_RATE=1</td>
<td></td>
<td>1,940</td>
<td>3,234</td>
</tr>
<tr>
<td></td>
<td>C_RATE=25</td>
<td></td>
<td>355</td>
<td>391</td>
</tr>
</tbody>
</table>

1. Maximum frequency obtained with map switches -ol high and -cm speed, and par switches -pl high and -rl high.
Rolling your own. Example:

Ray Andraka, “Hybrid Floating Point Technique Yields 1.2 Gigasample Per Second 32 to 2048 point Floating Point FFT in a single FPGA.”
Put three of these together and triplicate throughput!

Limited by DSP48 max. clock rate in Virtex 4 XCV4SX55-10: 400 MHz.
Total throughput: 1.2 Gs/s
Using FPGAs in the real world

- Performance boosting techniques.
- Floating point designs.
- Powering FPGAs.
- Interfacing to the outside world.
- Clock domains and metastability.
- Safe design and radiation hardness.
FPGA power requirements (1/2)

- Voltage: different voltage rails: core, I/Os, AUX, SERDES, PLL...

- Tolerance: typically +/- 5%.

- Monotonicity: Vcc must rise steadily from GND to desired value (could work otherwise but FPGAs are not tested that way).
FPGA power requirements (2/2)

- Power-on current. Watch out for PCB capacitor in-rush current: \( I_c = C \frac{\Delta V}{\Delta T} \). Slow down voltage ramp if needed.

- Sequencing: required for old technologies and recommended for new ones. Read datasheet. Example for Virtex-4/5: VCCINT → VCCAUX → VCCO. Use Supply Voltage Supervisor (SVS) to control sequencing.

- Power-on ramp time. Devices specify a minimum and a maximum ramp time. Again, this is how they are tested.
Power solutions

- Switching solutions (some have external clk pins that you can drive at a frequency you can easily filter afterwards)
  - Controller (external FET)
  - Converter (built-in FET)
  - Module
- Multi-rail solutions
LDO: Be aware - Under-voltage lockout

- Problem: LDO with non-monotonic voltage output.
- Cause:
  - 5V primary supply was powering on at the same time.
  - Caps and 3 LDOs caused the 5V to droop.
- Result:
  - Primary 5V current-limiter shut it down.
  - LDO's under-voltage lockout tripped, shutting down the LDO.
- How can we fix this?
LDO: under-voltage lockout solution

- Use SVS to sequence regulators after caps are charged.
LDO: be aware – in-rush and current limit

- A fast-starting LDO induces a huge in-rush current from charging capacitors (remember $I_c = C \frac{\Delta V}{\Delta T}$)
- LDO enters current-limit mode due to capacitor in-rush.
- The transition to current-limit mode causes a glitch.
- What to do?
LDO: in-rush and current-limit solution

- Slow down the ramp time using a soft-start circuit.
  - Reduces $\Delta V/\Delta t$ which reduces capacitor in-rush current.
  - Regulator never hits current-limit and stays in voltage mode.
  - Good for meeting FPGA minimum ramp time specs.
  - External or built-in.

Note: in-rush FPGA current during configuration is a thing of the past thanks to the introduction of proper housekeeping circuitry.
How much current is our design consuming?

- Insert a small high-precision resistor in series with primary voltage source before the regulator, and measure the voltage drop with a differential amplifier. Below an example from a LLRF board designed by Larry Doolittle (LBNL).

- Then compare with the predicted power consumption from your vendor’s software tool ;)

![Diagram of voltage regulators](image-url)
Decoupling capacitors

- Capacitors are not ideal! They have parasitic resistance and inductance:
Decoupling capacitors

- Knee frequency in the spectrum of a digital data stream is related by the rise and fall times ($T_r$) by: $F_{knee} = 0.5/T_r$ (1).
- We want our Power Distribution System to have low impedance at all frequencies of interest → low voltage variations for arbitrary current demands.
- Solution: parallel combination of different capacitor values. For more info: Xilinx XAPP623.

Outline

- Using FPGAs in the real world
  - Performance boosting techniques.
  - Floating point designs.
  - Powering FPGAs.
  - Interfacing to the outside world.
  - Clock domains and metastability.
  - Safe design and radiation hardness.
FPGAs have very versatile connectivity. Example: Xilinx Spartan 3 family.

- Single ended and differential.
  - 784 single-ended, 344 differential pairs.
  - 622 Mb/sec LVDS.
  - 24 I/O standards, 8 flexible I/O banks.
  - PCI 32/33 and 64/33 support.
  - Eliminate costly bus transceivers.
- 3.3V, 2.5V, 1.8V, 1.5V, 1.2V

**Chip-to-Chip Interfacing:**
- LVDS
- LVCMOS
- LVTTL

**Backplane Interfacing:**
- GTL
- GTL+
- PCI
- BLVDS

**High-speed Memory Interfacing:**
- HSTL
- SSTL
Interfacing with ADCs and DACs

- Large parallel busses working at high clock rates → potential for timing and noise problems.

- Possible solutions:
  - ADCs nowadays have analog bandwidths well above twice their maximum sampling rate → sample band pass signals at slower rates (in other Nyquist zones).
  - Use high speed differential serial links for ADCs and DACs (so far, no embedded clock: clk + data on two separate LVDS links).
  - Run digital supply in parallel ADCs as low as possible: 2.0-2.5V feasible.
Interfacing with busses using 5V signaling (e.g. VME)

- Dual supply level translators are the most flexible solution.

- Alternatives:
  - 5V compliant 3.3V buffers exist, such as the LVTH family. They also provide more current than standard FPGA I/Os.
  - Open-drain devices (uni-directional, can do wired-or).
  - FET switches (very fast, no active drive).
Using FPGAs in the real world

- Performance boosting techniques.
- Floating point designs.
- Powering FPGAs.
- Interfacing to the outside world.
- Clock domains and metastability.
- Safe design and radiation hardness.
Characterizing metastability

Use measurements with this setup to find $K_1$ and $K_2$, assuming an MTBF of the form:

$$MTBF = \frac{e^{K_2 \cdot \tau}}{F_1 \cdot F_2 \cdot K_1}$$
Virtex II Pro Metastability results

From Xilinx XAPP094
Synchronizer circuit

- Place the two flip-flops close together to minimize net delay.

- When a signal comes on-chip, synchronize it first then fan-out (don’t fan-out then synchronize at multiple places).

- Make sure clk period is OK for desired MTBF. E.g. for Virtex II Pro, giving the flip-flop 3 ns to resolve will give you an MTBF higher than 1 Million years!
Crossing clock domains

- For single-bit signals, use the double flip-flop synchronizer.
- For multi-bit signals, using a synchronizer for each bit is *wrong*.
  - Different synchronizers can resolve at different times.
  - No way to know when data is valid, other than waiting a long time.
  - For slow transfers, you can use 4-phase or 2-phase handshake (a single point of synchronization). Otherwise, give up acknowledgement and make sure system works “by design”. FIFOs are also useful.
The VALID signal is synchronous to the source clock and gets synchronized at the receiving end by a double flip-flop synchronizer. The same happens in the opposite sense with the ACK signal.
Two phase handshake

[Adapted from VLSI Architectures Spring 2004 www.ee.technion.ac.il/courses/048878 by Ran Ginosar]
Michael Crews and Yong Yuenyongsgool, Practical design for transferring signals between clock domains. EDN magazine, February 20, 2003.
Using FPGAs in the real world
- Performance boosting techniques.
- Floating point designs.
- Powering FPGAs.
- Interfacing to the outside world.
- Clock domains and metastability.
- Safe design and radiation hardness.
Reset strategies

Different flip-flops see reset deasserted in different clock cycles!

It matters in a circuit like this.

You can fix this problem with a proper reset generator.

Even better if you can use this as a synchronous reset
Safe state machines

One-hot encoding:
s0 => 0001
s1 => 0010
s2 => 0100
s3 => 1000

12 “illegal states” not covered, or covered with a “when others” in VHDL or equivalent.
→ Use option in synthesis tool to prevent optimization of illegal states.
Single Event Effects (SEE) created by neutrons

Neutrons

Cosmic rays

Space

Atmosphere

Earth

Neutron

Source

Gate

Drain

Alpha particle

Silicon nucleus

Sensitive region

Memory Cell: CMOS
Configuration Latch (CCL)

Upset: $0 \rightarrow 1$, $1 \rightarrow 0$
Classification of SEEs

Single Event Transient (SET)
A signal briefly fluctuates somewhere in design

Single Event Upset (SEU)
Bit-Flip Somewhere

Single Event Latch-Up (SEL)
Parasitic transistors activated in a device, causing internal short

Single Event Functional Interrupt (SEFI)
Bit-Flip specifically in a control register – POWER ON RESET/JTAG etc.
SEL description

- Has virtually disappeared in new technologies (low $V_{ccint}$ not enough to forward bias transistors).
- Only cure used to be epitaxial substrate (very expensive).

Activation of either of these transistors causes a short from $V+$ to $V-$.
SEU Failures in Time (FIT)

- Defined as the number of failures expected in $10^9$ hours.
- In practice, configuration RAM dominates. Example:

  Virtex XCV1000 memory Utilization

<table>
<thead>
<tr>
<th>Memory Type</th>
<th># of bits</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>5,810,048</td>
<td>97.4</td>
</tr>
<tr>
<td>Block RAM</td>
<td>131,072</td>
<td>2.2</td>
</tr>
<tr>
<td>CLB flip-flops</td>
<td>26,112</td>
<td>0.4</td>
</tr>
</tbody>
</table>

- Average of only 10% of FPGA configuration bits are used in typical designs
  - Even in a 99% full design, only up to 30% are used
  - Most bits control interconnect muxes
  - Most mux control values are “don’t-care”
- Must include this ratio for accurate SEU FIT rate calculations.
Not all parts of the design are critical

- Average of only 40% of circuits in FPGA designs are critical
  - Substantial circuit overhead for startup logic, diagnostics, debug, monitoring, fault-handling, control path, etc.
- Must also include this ratio in SEU FIT rate calculations
Actual FIT

<table>
<thead>
<tr>
<th>Name</th>
<th>SEUPI Ratio</th>
<th>CC Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>“SEU Probability Impact” Ratio</td>
<td>“Critical Circuit” Ratio</td>
<td></td>
</tr>
<tr>
<td>Definition</td>
<td>% of total configuration bits that impact a given customer design</td>
<td>% of total design that is critical for standard system operation</td>
</tr>
<tr>
<td>Typical Range(^1)</td>
<td>1% - 30%</td>
<td>20% - 80%</td>
</tr>
<tr>
<td>Average(^1)</td>
<td>10%</td>
<td>40%</td>
</tr>
</tbody>
</table>

Note 1: From analysis of real FPGA designs

Actual FIT = Base FIT * SEUPI Ratio * CC Ratio
Half-latches (weak keepers) in Virtex devices

- Provide constants
- Save logic resources
- Used throughout device
- Subject to SEU upset
  - Can reset over time
- Not observable
  - Not defined by configuration bits
- Reinitialized as part of device initialization
  - Full reconfiguration required
Mitigation techniques: scrubbing

- Readback and verification of configuration.
  - Most internal logic can be verified during normal operation.
  - Sets limits on duration of upsets.
- Partial configuration
  - Not supported by all FPGA vendors/families.
  - Allows fine grained reconfiguration.
  - Does not reset entire device.
    - Allows user logic to continue to function.
- Complete reconfiguration
  - Required after SEFI.
  - No user functionality for the duration of reconfiguration.
- Verification by dedicated device
  - Usually radiation tolerant antifuse FPGA
  - Secure storage of checksums and configuration an issue
    -FLASH is radiation sensitive
- Self verification
  - Often the only option for existing designs
  - Not possible in all device families
    - Utilizes logic intended for dynamic reconfiguration
    - Verification logic has small footprint
      - Usually a few dozen CLBs and 1 block RAM (for checksums).
Triple Module Redundancy (TMR)

Feedback TMR

- Three copies of user logic
- State feedback from voter
  - Counter example
- Handles faults
- Resynchronizes
  - Operational through repair
- Speed penalty due to feedback
- Desirable for state based logic
Alternatives

- **Antifuse**
  - Configuration based on physical shorts
    - Invulnerable to upset
    - Cannot be altered
  - Over 90% smaller upset cross section for comparable geometry
  - Signal routing more efficient
    - Much lower power dissipation for similar device geometry
  - Lags SRAM in fabrication technology
    - Usually one generation behind
    - Latch up more of a problem than in SRAM devices

- **Rad-hard Antifuse**
  - All flip-flops TMRed in silicon
    - Unmatched reliability
    - High (extreme) cost
    - Unimpressive performance
      - Feedback TMR built in
      - Usually larger geometry
      - Not available in highest densities offered by antifuse

- **FLASH FPGAs**
  - Middle ground in base susceptibility
  - Readback/Verification problematic
    - Usually only JTAG (slow) supported
  - Maximum number of write cycles an issue
Many thanks to Jeff Weintraub (Xilinx University Program), Eric Crabill (Xilinx), John Molendijk (CERN), Ben Todd (CERN), Matt Stettler (LANL), Larry Doolittle (LBNL) and Silica for some of these slides.