FPGA technology in beam instrumentation and related tools

Javier Serrano
CERN, Geneva, Switzerland
Plan of the presentation

- FPGA architecture basics
- FPGA design flow
- Performance boosting techniques
- Doing arithmetic with FPGAs
- Example: RF cavity control in CERN’s Linac 3.
A preamble: basic digital design

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

Higher clock rate: 151.5 MHz on the same chip.

FPGA internal architecture 1/3

Example: Xilinx Spartan-IIE family architecture

CLB: Configurable Logic Block
DLL: Delay Locked Loop
Simplified view of Spartan-IIIE CLB Slice (two identical slices inside each CLB)

Members of the Spartan-IIIE family range from the XC2S50E (16*24=384 CLBs) to the XC2S600E (48*72=3456 CLBs).
FPGA internal architecture 3/3

Other design resources in modern FPGAs:

- Clock control blocks (DLL or PLL).
- Fast differential signaling support (LVDS, LVPECL,…).
- Fast hard-wired DSP blocks made of multipliers and accumulators.
- High speed external RAM interfacing, plus lots of internal RAM.
- Multi gigabit transceivers (useful for global orbit feedback).
- Embedded CPU cores (PowerPC, ARM,…).
- Digitally Controlled Impedance active I/O termination.
FPGA vs. DSP chips

Virtex-4SX55: 512 MAC units @ 500 MHz = 256 GMAC/s!

Loop 256 times per Data In sample for a 256 tap FIR filter.

DSP

FPGA

Data In

Register

Data In

Reg0

Reg1

Reg255

Data Out

Data Out

X

X

X

C0

C1

C255

+

+
FPGA design flow

- Design Entry
- Behavioral simulation
- Synthesis
- Place and Route
- Post P&R simulation

 RTL View

```
DummyOut <= DummyInA when Selector='1' else DummyInB;
```

Technology view
FPGA flow: P&R results

FPGA flow: floorplanning

myCounter0: process(Reset(0), Clk)
begin
if Reset(0)='1' then
  counter0 <= (others=>'0');
elsif Clk'event and Clk='1' then
  counter0 <= counter0 + 1;
end if;
end process myCounter0;

Increasing performance 1/5

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
Increasing performance 2/5
Replicating registers (and associated logic if necessary)
Increasing performance 3/5
Retiming (a.k.a. register balancing)

Before

Large combinatorial logic delay

Small delay

After

Balanced delay

Balanced delay

Increasing performance 4/5

Pipelining

Before

Large combinatorial logic delay

After

Small delay

Increasing performance 5/5
Time multiplexing

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 2/2
Boosting performance of an IIR filter

Look ahead scheme:
From $y[n+1] = ay[n] + b \times x[n]$ we get
Performing arithmetic in FPGAs 1/2

- Binary adders: made of \(N\) full adders, each implementing:
  - \(s_k = x_k \text{ XOR } y_k \text{ XOR } c_k\)
  - \(c_{k+1} = (x_k \text{ AND } y_k) \text{ OR } (x_k \text{ AND } c_k) \text{ OR } (y_k \text{ AND } c_k)\)
  - Easy to pipeline.

- Multipliers: hardwired (if your chip has them) or "pencil and paper":

\[
P = A \cdot X = \sum_{k=0}^{N-1} a_k 2^k X
\]

\(X\) is successively shifted by \(k\) positions. Then, whenever \(a_k = 1\), \(X2^k\) is accumulated. These multipliers can be pipelined, as opposed to the hardwired variety.
Performing arithmetic in FPGAs 2/2

- **Dividers:** pencil and paper method.
  
  
  \[ Q = \frac{A}{X} = \frac{\sum_{k=0}^{N-1} a_k 2^k}{X} \]

  Start with an empty auxiliary register B and start shifting bits from A into it (right to left). Whenever B-X is positive, replace B with B-X. After every shift we get a bit of the quotient: 0 if B-X is negative, 1 otherwise.

  - Keep in mind that these are good solutions when both operands are variable. Example with one fixed operand: 0.5625a = 9a/16 = a/2 + a/16. Used at CERN to get baseline from BPM signal through lossy integrator.

  - Sin, cos, sinh, cosh, atan, atanh, square root and vector rotation: CORDIC.
Digital Signal Processing is about sums of products:

\[ y = \sum_{n=0}^{N-1} c[n] \cdot x[n] \]

Let’s assume:

- \( c[n] \) constant (prerequisite to use DA)
- \( x[n] \) input signal B bits wide

Then:

\[ y = \sum_{n=0}^{N-1} \left( c[n] \cdot \sum_{b=0}^{B-1} x_b[n] \cdot 2^b \right) \]

\( x_b[n] \) is bit number \( b \) of \( x[n] \) (either 0 or 1)

And after some rearrangement of terms:

\[ y = \sum_{b=0}^{B-1} 2^b \cdot \left( \sum_{n=0}^{N-1} c[n] \cdot x_b[n] \right) \]

This can be implemented with an N-input LUT
Distributed Arithmetic (DA) 2/2

\[ y = \sum_{b=0}^{B-1} 2^b \cdot \left( \sum_{n=0}^{N-1} c[n] \cdot x_b[n] \right) \]

General vector rotation:
\[ x' = x \cdot \cos \phi - y \cdot \sin \phi \]
\[ y' = y \cdot \cos \phi + x \cdot \sin \phi \]

Rearranging:
\[ x' = \cos \phi [x - y \cdot \tan \phi] \]
\[ y' = \cos \phi [y + x \cdot \tan \phi] \]

We restrict rotation angles to be:
\[ \tan \phi = \pm 2^{-i} \]

The cosine can be treated as a constant since:
\[ \cos(\delta_i) = \cos(-\delta_i) \]

Giving the CORDIC equations:
\[ x_{i+1} = K_i [x_i - y_i \cdot d_i \cdot 2^{-i}] \]
\[ y_{i+1} = K_i [y_i + x_i \cdot d_i \cdot 2^{-i}] \]

With:
\[ K_i = \cos(\arctan 2^{-i}) \]
\[ d_i = \pm 1 \]
CORDIC 2/2

- Two working modes:
  - Rotation mode: rotates the input vector by a specified angle given as an argument.
  - Vectoring mode: rotates the vector until it aligns with the x axis while recording the angle required to make that rotation.

- Usage examples:
  - To compute \((\rho, \varphi)\) from \((x,y)\) (polar to cartesian transformation) feed \((x,y)\) to the CORDIC rotator in vectoring mode, then find the results in \(x\) and the phase accumulator.
  - To compute \(\sin \varphi\), feed \((x=1,y=0)\) to the CORDIC in rotation mode, then find the result in \(y\).
Case study: low level RF cavity control in CERN’s Linac 3 1/4
Case study: low level RF cavity control in CERN’s Linac 3 2/4

80 MHz LO

Mixer

20 MHz LPF

100 MHz from cavity

Sampling the 20 MHz at exactly 4 times its frequency produces I, Q, -I, -Q, I, Q…
Case study: low level RF cavity control in CERN’s Linac 3 3/4

Case study: low level RF cavity control in CERN’s Linac 3 4/4
Thanks!

- Many thanks to Uli Raich and Tony Rohlev for help in preparing this talk.
- Snapshots in some slides courtesy of Xilinx and Synplicity.
- Some references for further study:
  - “Digital Signal Processing With Field Programmable Gate Arrays” 2nd edition by U. Meyer-Baese
  - Andraka Consulting Group: http://www.andraka.com/
  - comp.arch.fpga newsgroup
Case study: low level RF cavity control in CERN’s Linac 3