FPGA Workshop

Physical structure, programming principals and their applications in Nuclear Physics

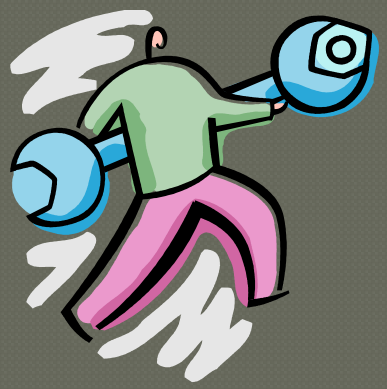
Marc-André Tétrault
Université de Sherbrooke

Presentation Overview

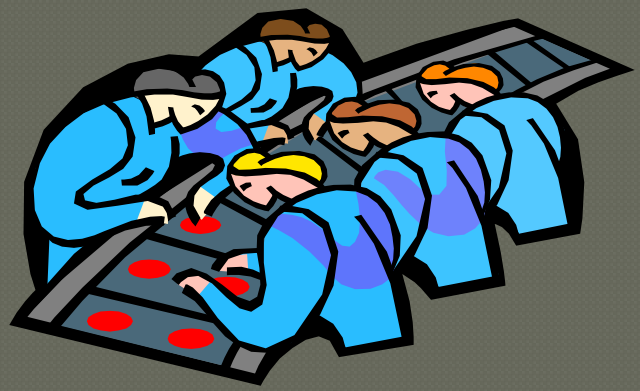
- FPGA building blocks
- Basic design considerations
- Performance and resource optimization techniques
- Optimization techniques applied in a PET subsystem

Field Programmable Gate Array

Processors

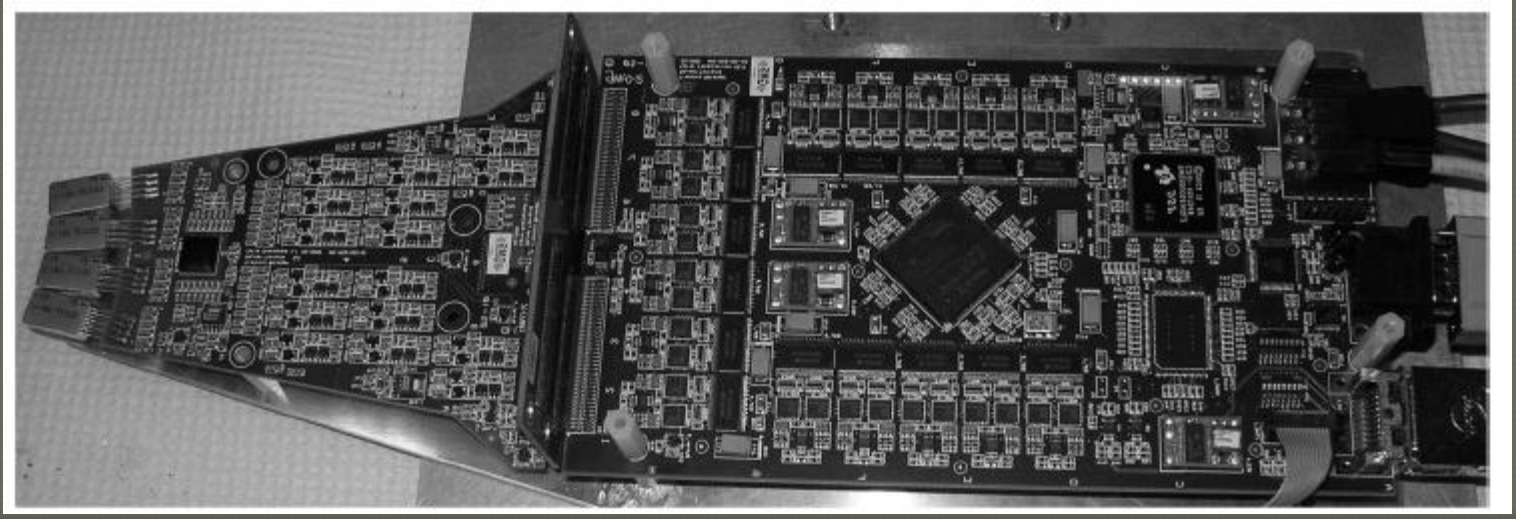


FPGA



Applications

- Basic interfacing
  - Glue logic, interconnect components
- Complex system
  - System on Chip solutions (SoC)
  - Real time signal sampling and/or processing



FPGA Types

- High end devices (Virtex, Stratix)
- Low cost devices (Spartan, Cyclone)
- Mixed signal devices (Actel Fusion)
  - Multiplexed ADC and clock oscillator included
- Low power devices (Actel Igloo)

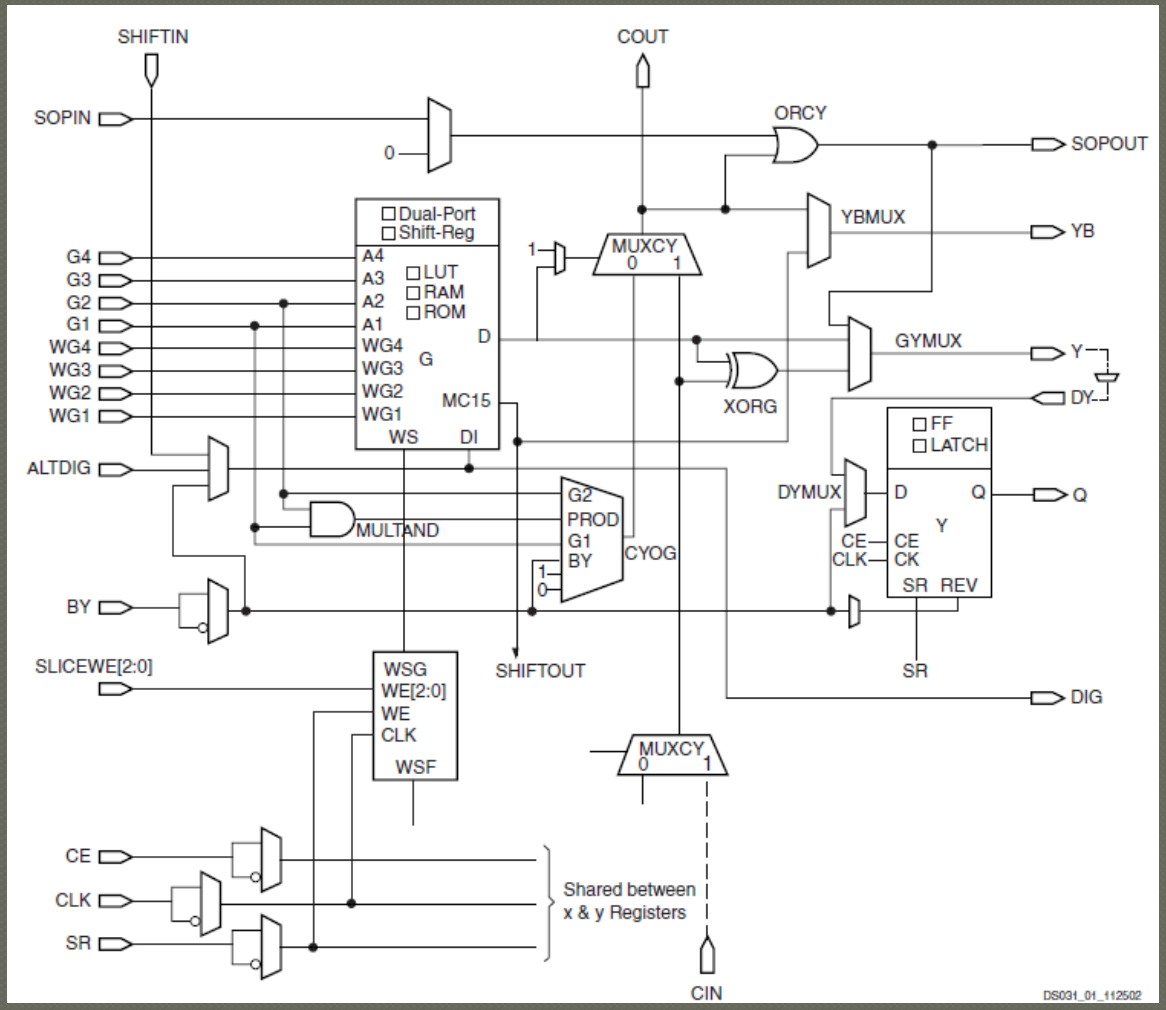
Building Blocks

- Logic
- Memory
- I/O
- Clock management
- Specialized units
- Pre-designed cores

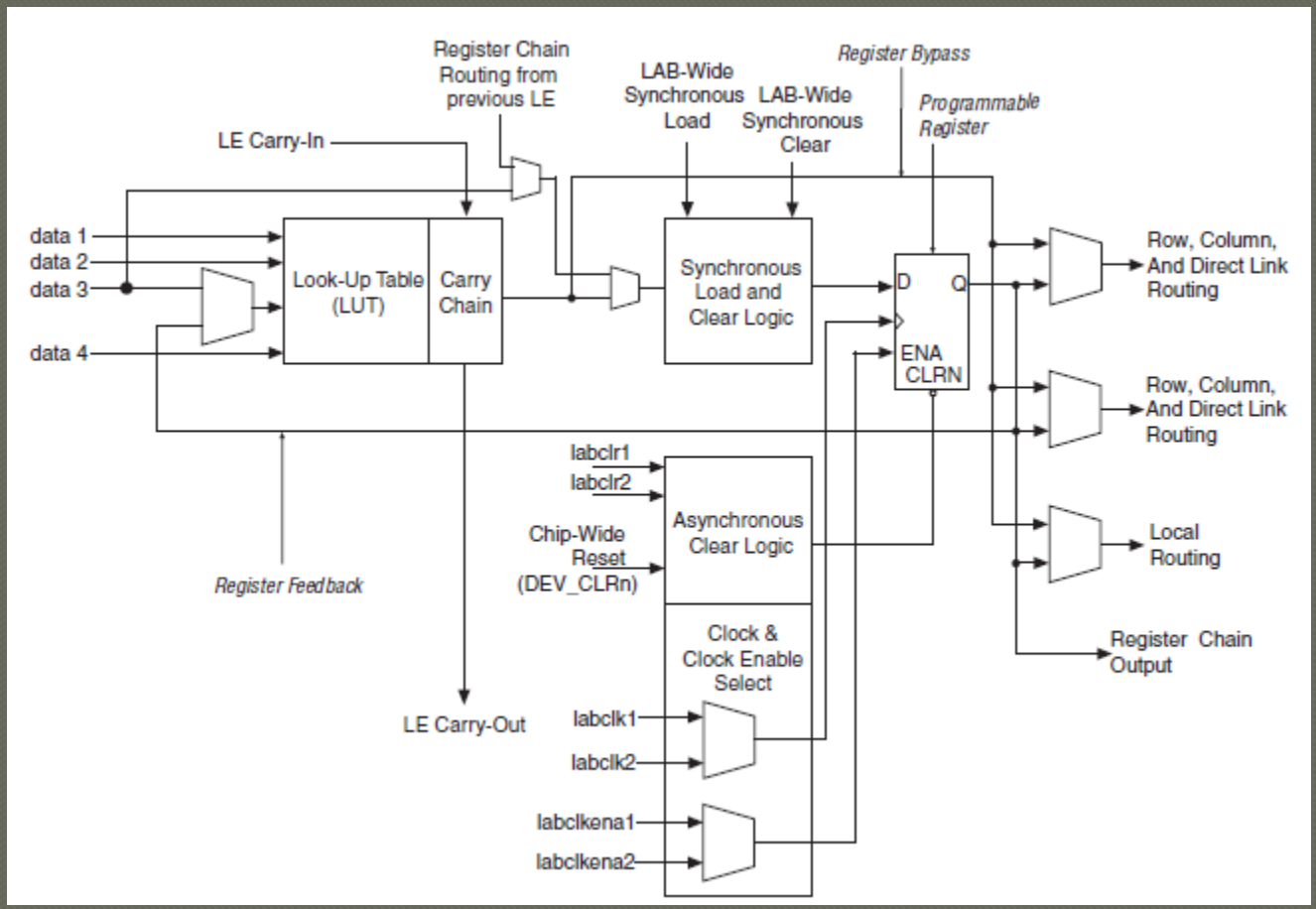
FPGA Internals : Primitive Cells

- Basic logic generator
  - Look-up table
- Storage unit
  - Clock synchronous flip flop
  - Latch
- Signal routing

Local Routing Example

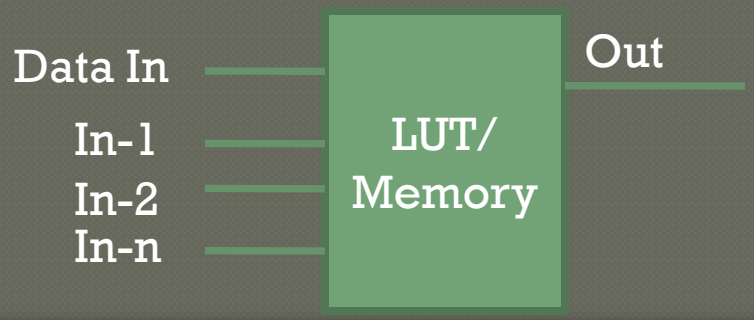


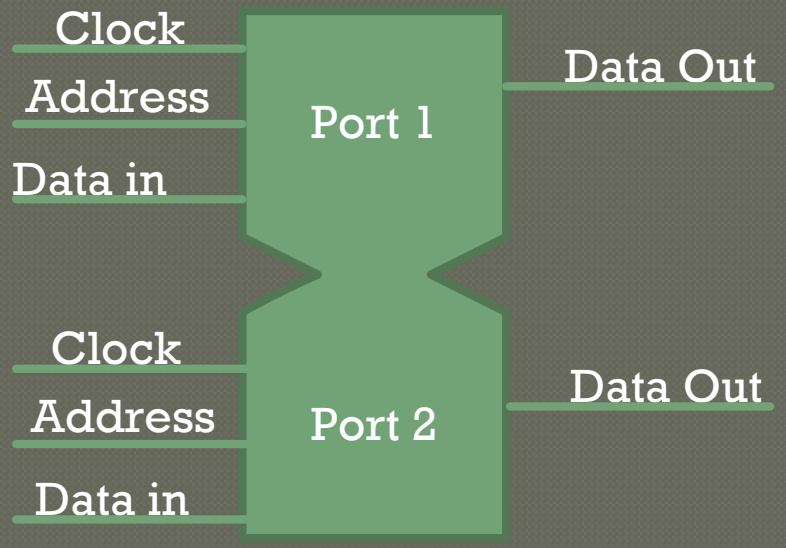
Local Routing Example



Embedded Memory

- Two types
  - Small local memory – uses LUT resources
  - Large, dedicated SRAM
- Pre-charged with user-defined values at configuration
  - Read Only Memory
    - Filter coefficients



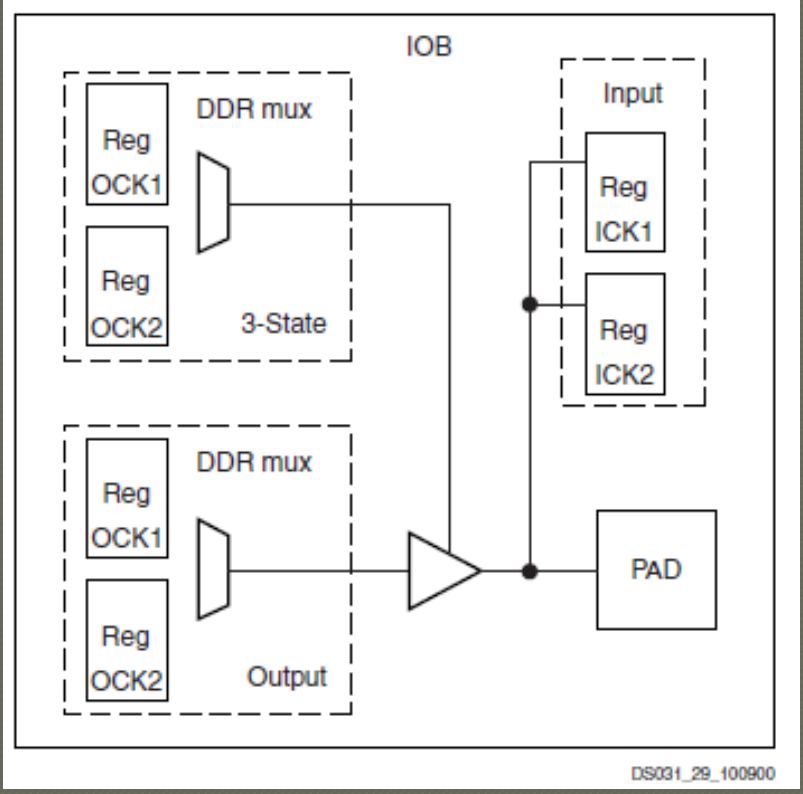


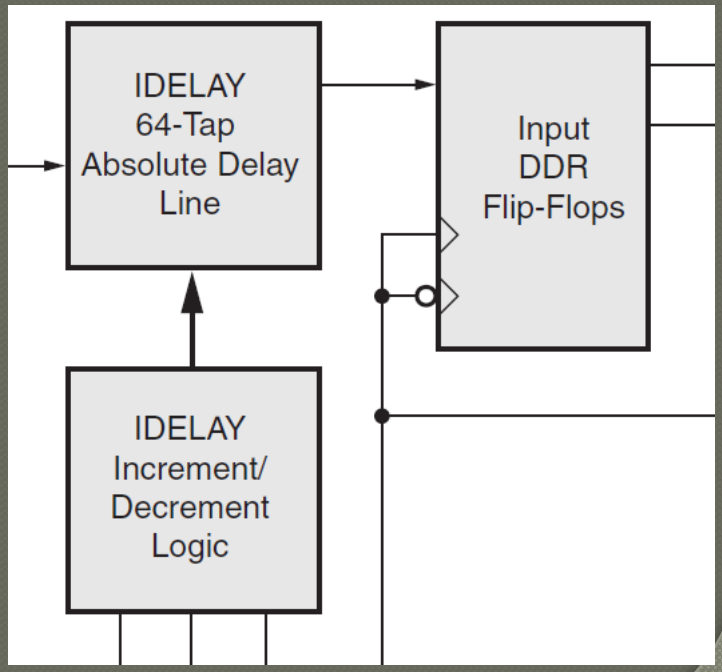
Input/Output Pads

- Electrical Standard support
  - LVTTL
  - LVCMOS
  - SSTL
  - LVDS
  - LVPECL

Input/Output Pads

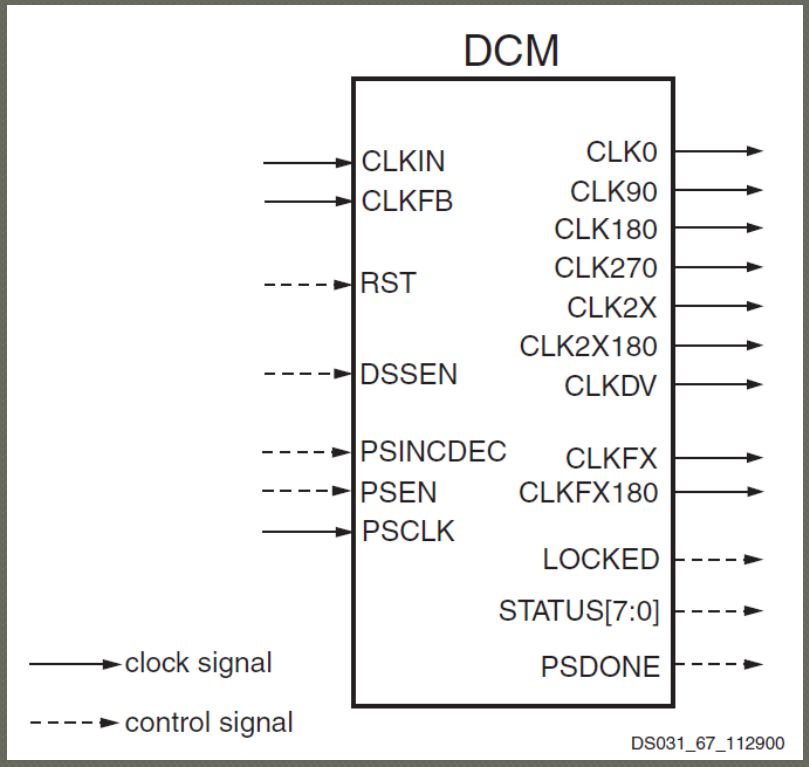
- Dedicated logic resources
- High-end device extras





Clock Resources

- In general, external clock source
- Clock distribution uses dedicated resources
- Clock manipulation
  - Phase lock
  - Jitter control



Specialized Units

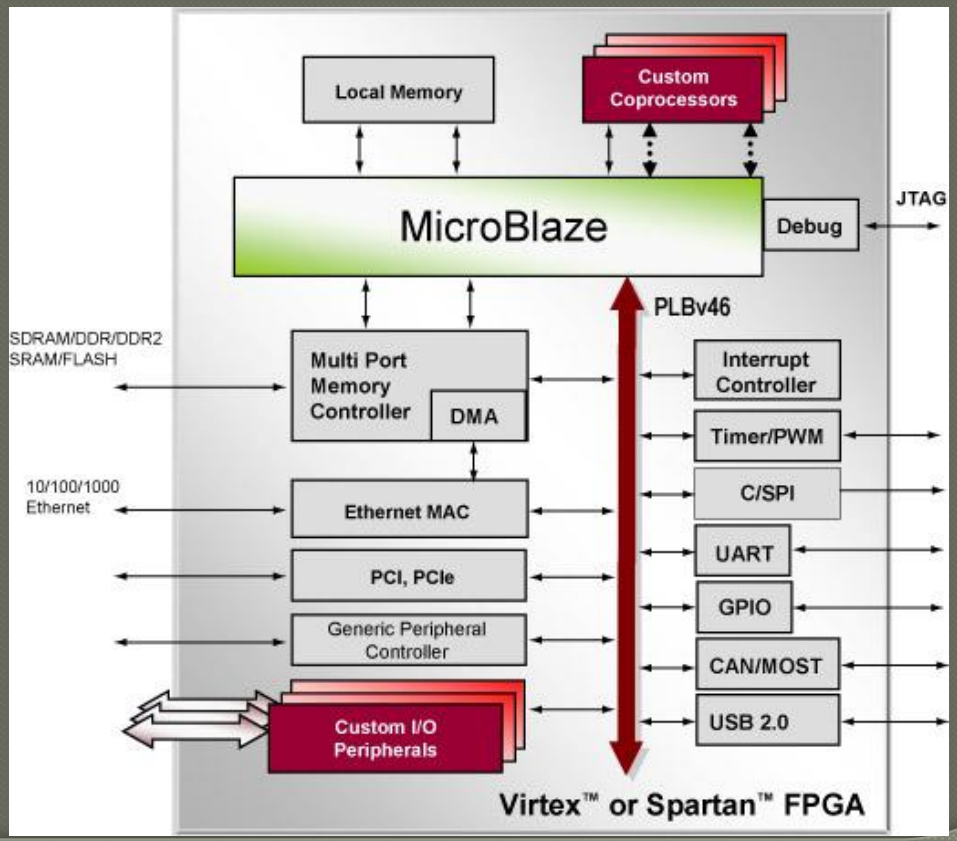
- Multiplier/DSP units
- Embedded high speed differential transceivers
- PCI Express Core
- Ethernet MAC (Media Access Controller)

Intellectual Property (IP) Cores

- Free provided cores
  - Memory controllers
  - Basic standard protocols
  - www.opencores.org
- Licensed cores
  - Full featured protocols
- Third party cores

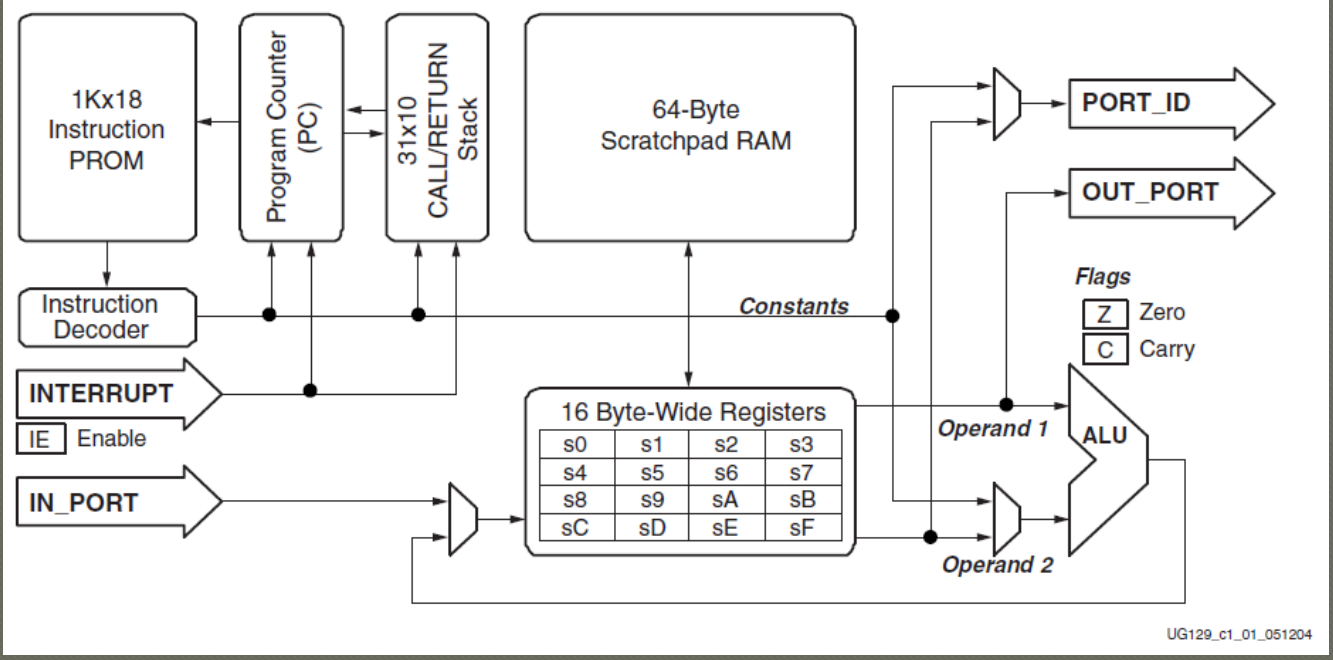
Embedded Processors

- 32-bit processors
- Supports embedded OSs
  - Linux
- Large programs require external memory



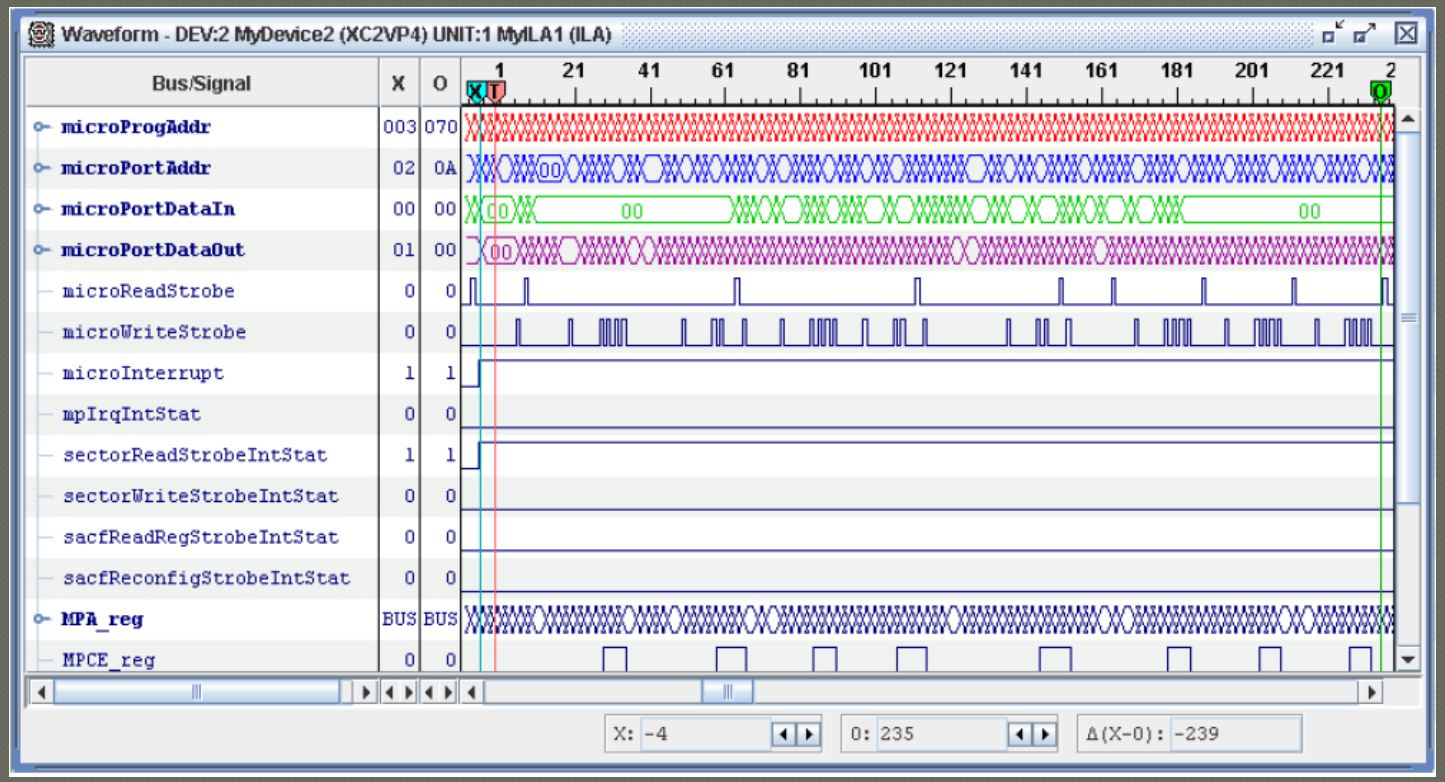
Embedded Processors

- Minimal footprint 8-bit processors
  - 68HC11
  - PicoBlaze



Logic Analyser Tool

- For in-system debugging
- Allows complex triggering



Basic Design Considerations

- Timing closure
- Coding style
- Metastability
- Reset signal considerations
- Multiple clock domains

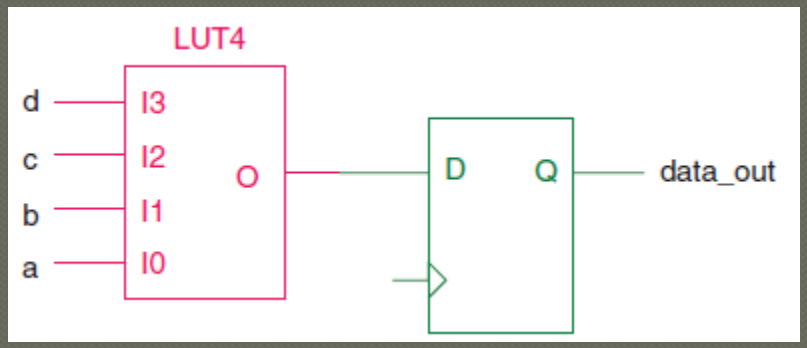
Timing Closure

- Logic performance depends on the longest logic/routing path
- Final result depends on
  - Worst case logic levels
  - Fan-out
- Performance can be improved with coding style and pipelining

Coding Style

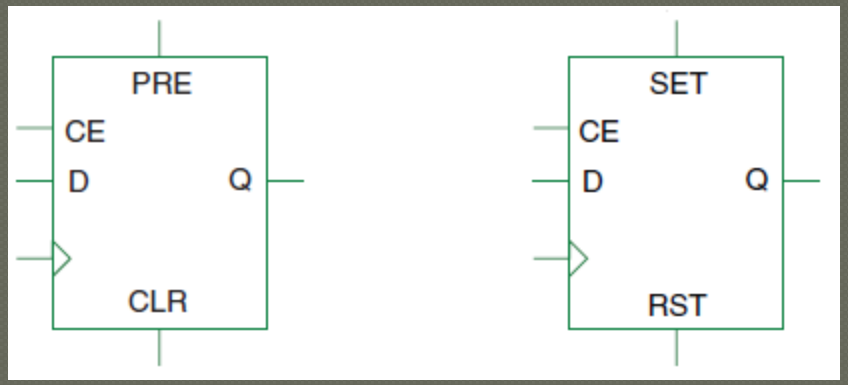
- Code while considering the size of your Look-Up Tables
  - Case of 4-input LUTs

```
1
2 process(clk)
3 begin
4     if(clk'event and clk = '1') then
5         ChannelTrigger <= MasterEnable and ChannelEnable
6                           and OverThreshold and not ChannelBusy;
7     end if;
8 end process;
```



Coding Style

- Flip-Flop optional controls
  - Reset/Clear
  - Set/Preset
  - Chip Enable

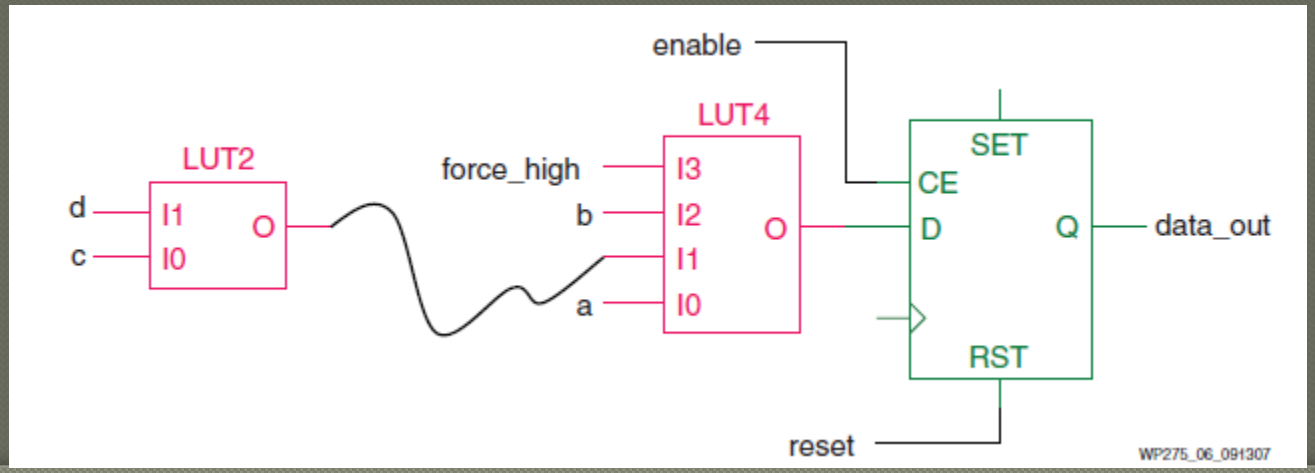


Coding Style

```
1
2  process (clk, reset)
3  begin
4      if(reset = '1')then
5          data_out <= '0';
6      elsif(clk'event and clk = '1') then
7          if(enable = '1')then
8              if(force_high = '1')then
9                  data_out <= '1';
10             else
11                 data_out <= a and b and c and d;
12             end if;
13         end if;
14     end if;
15 end process;
```

← Reset (asynchronous)
← Clock
← Enable
← Set (synchronous)

← Logic Operation



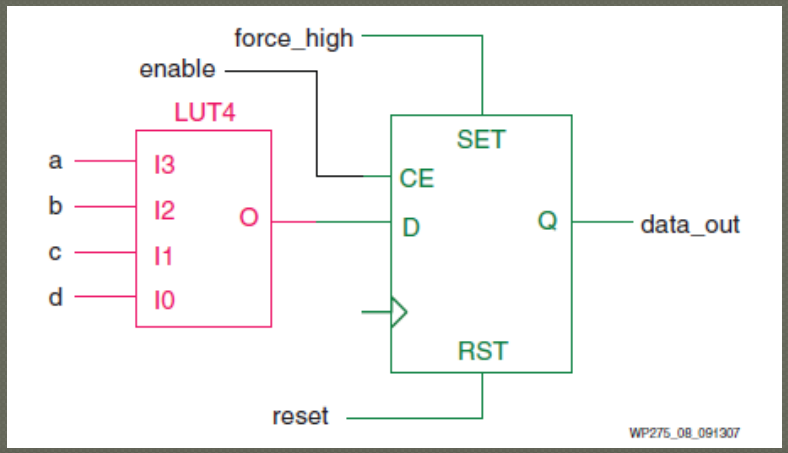
Coding Style

```
1
2  process (clk, reset)
3  begin
4      if(clk'event and clk = '1') then
5          if(reset = '1')then
6              data_out <= '0';
7          elsif(force_high = '1')then
8              data_out <= '1';
9          elsif(enable = '1')then
10             data_out <= a and b and c and d;
11         end if;
12     end if;
13 end process;
14
15
```

← Clock
← Reset

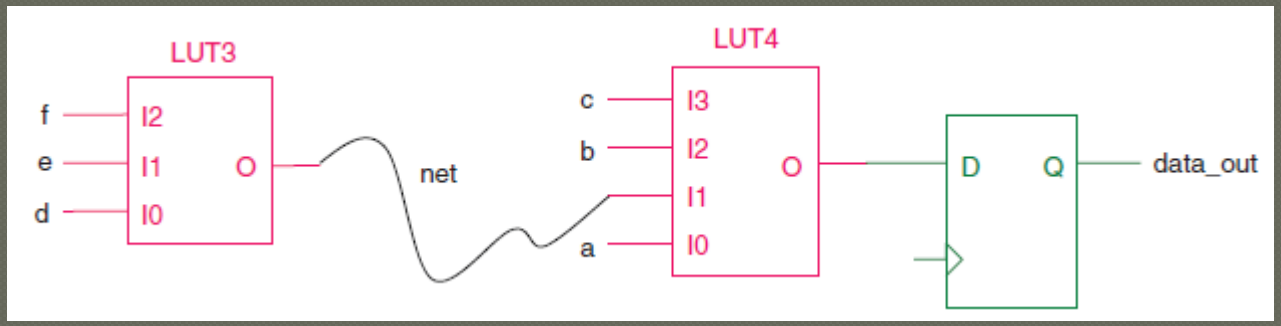
← Set

← Enable
← Logic Operation



Coding Style - Pipelining

- Break down logic in n-input functions
  - Adds latency, increases achievable speed
  - Case by case decisions
  - ie : 4 value adder

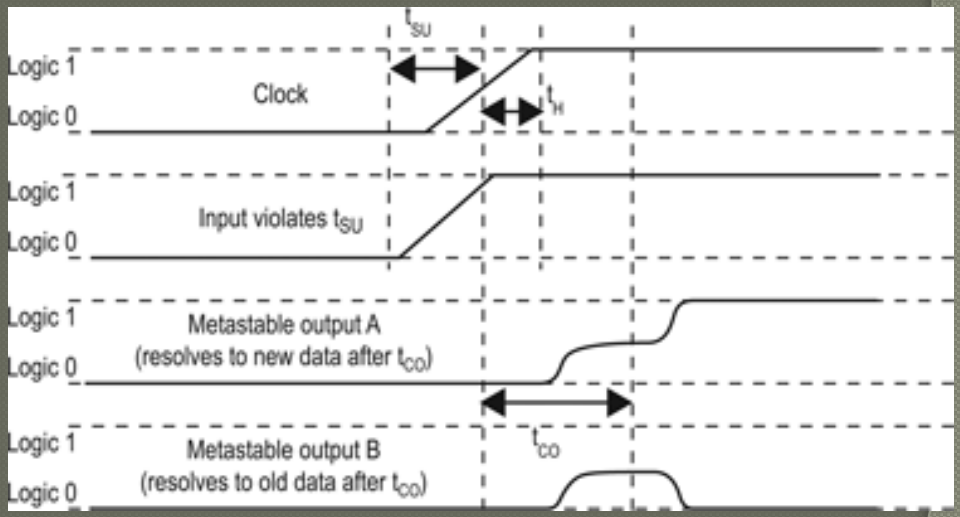


Coding Style

- Pay attention to your logic priorities
- Pipeline larger operations
  - Split the arithmetic/logic operation with registers to reduce propagation delays
- Use provided modules when possible
  - Optimized for speed and/or resource usage

Metastability

- One source of intermittent bugs, disassociated from logic behaviour
  - When setup/hold times not respected
- Cases when it appears
  - Asynchronous signals
    - Push buttons
  - Unrelated clock domain crossing



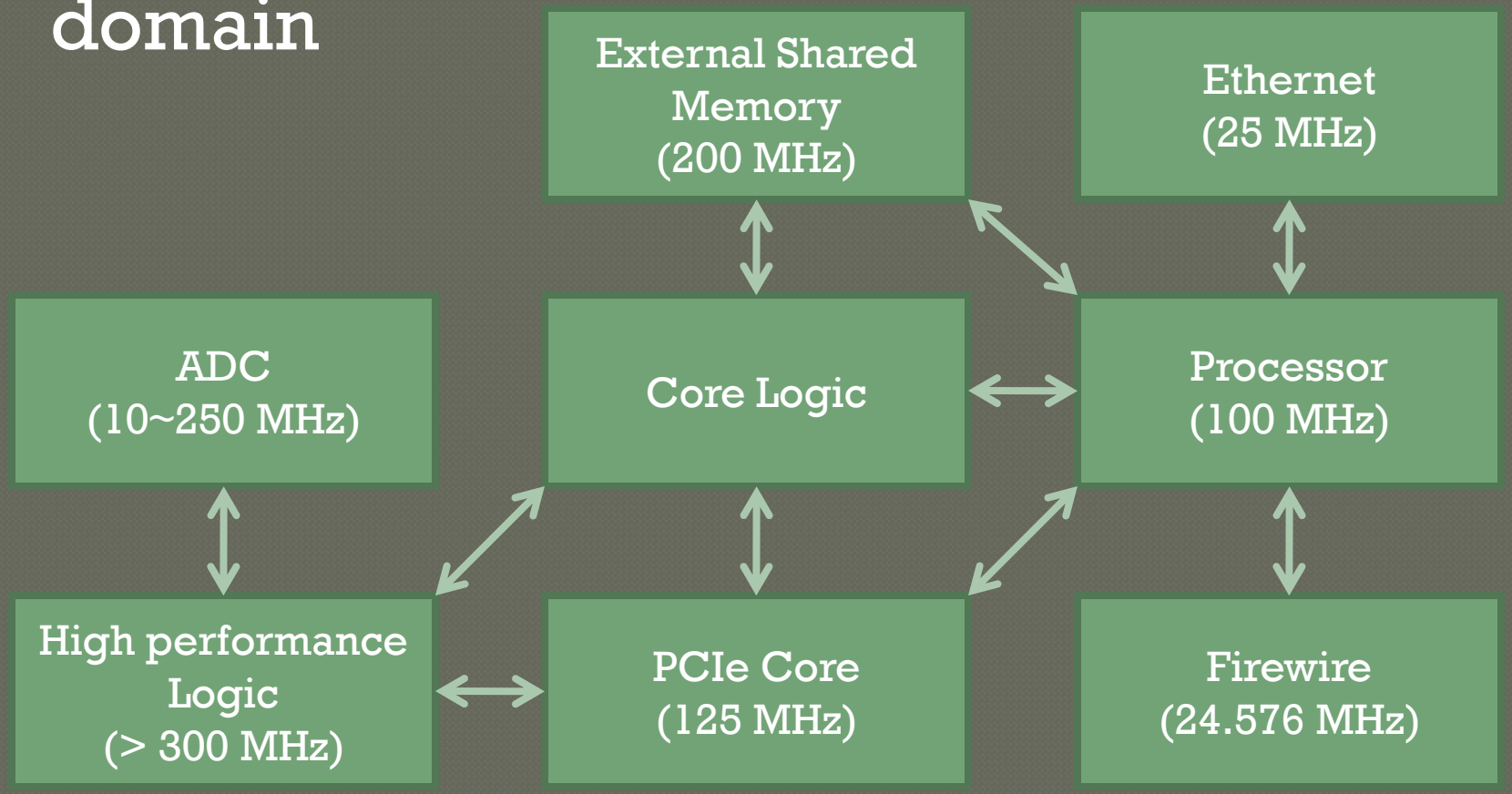
Source :
http://dataweek.co.za/news.aspx?pklNewsId=32117&pklCategoryID=35

Reset Signal Considerations

- Goal : Bring design in a known state
- Is the reset really required?
  - FPGAs are in known state after programming
- Local vs Global reset
  - Xilinx White Paper #272 "Get Smart About Reset: Think Local, not Global", by K. Chapman
- Timing considerations
  - External reset synchronous or asynchronous?
  - No problems in 99% cases

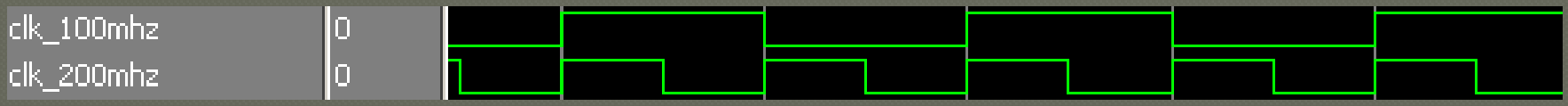
Clock Domain Crossing

- Very few designs today have only 1 clock domain



Clock Domain Crossing

- Related, phased-locked clocks

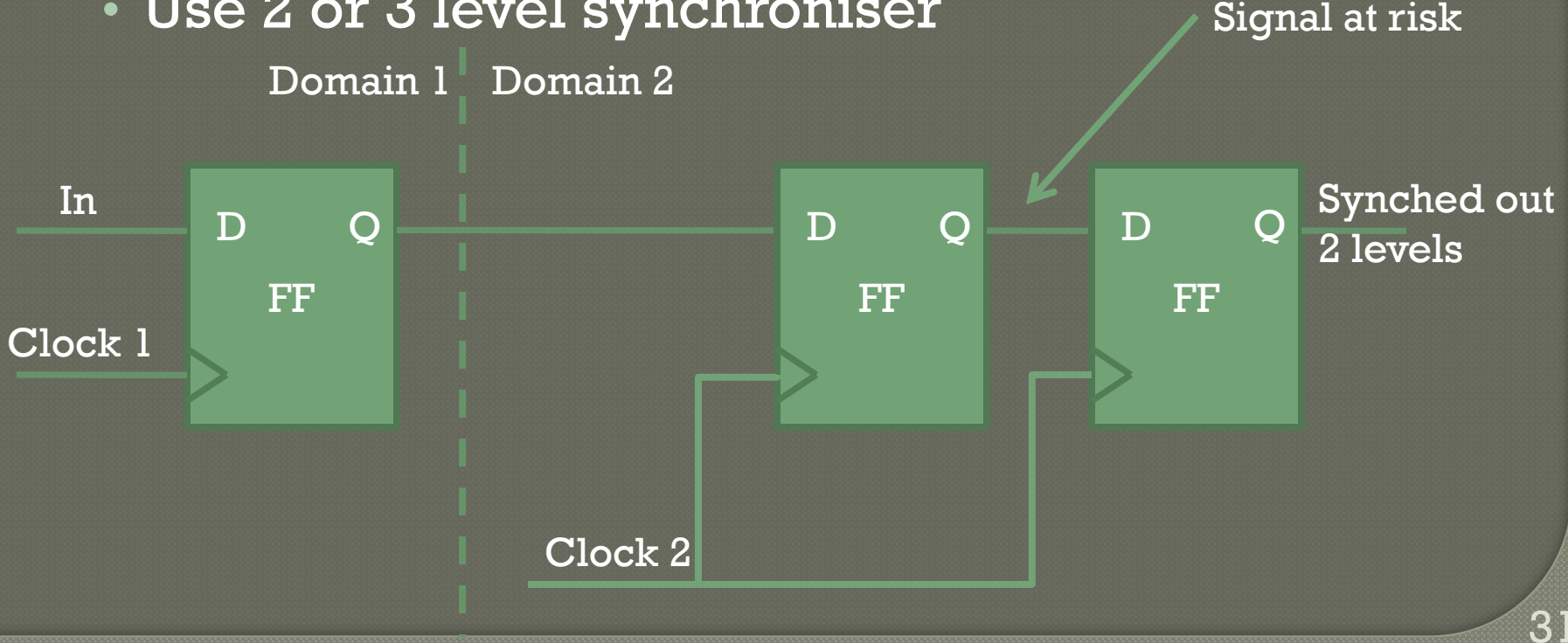


- Unrelated clocks

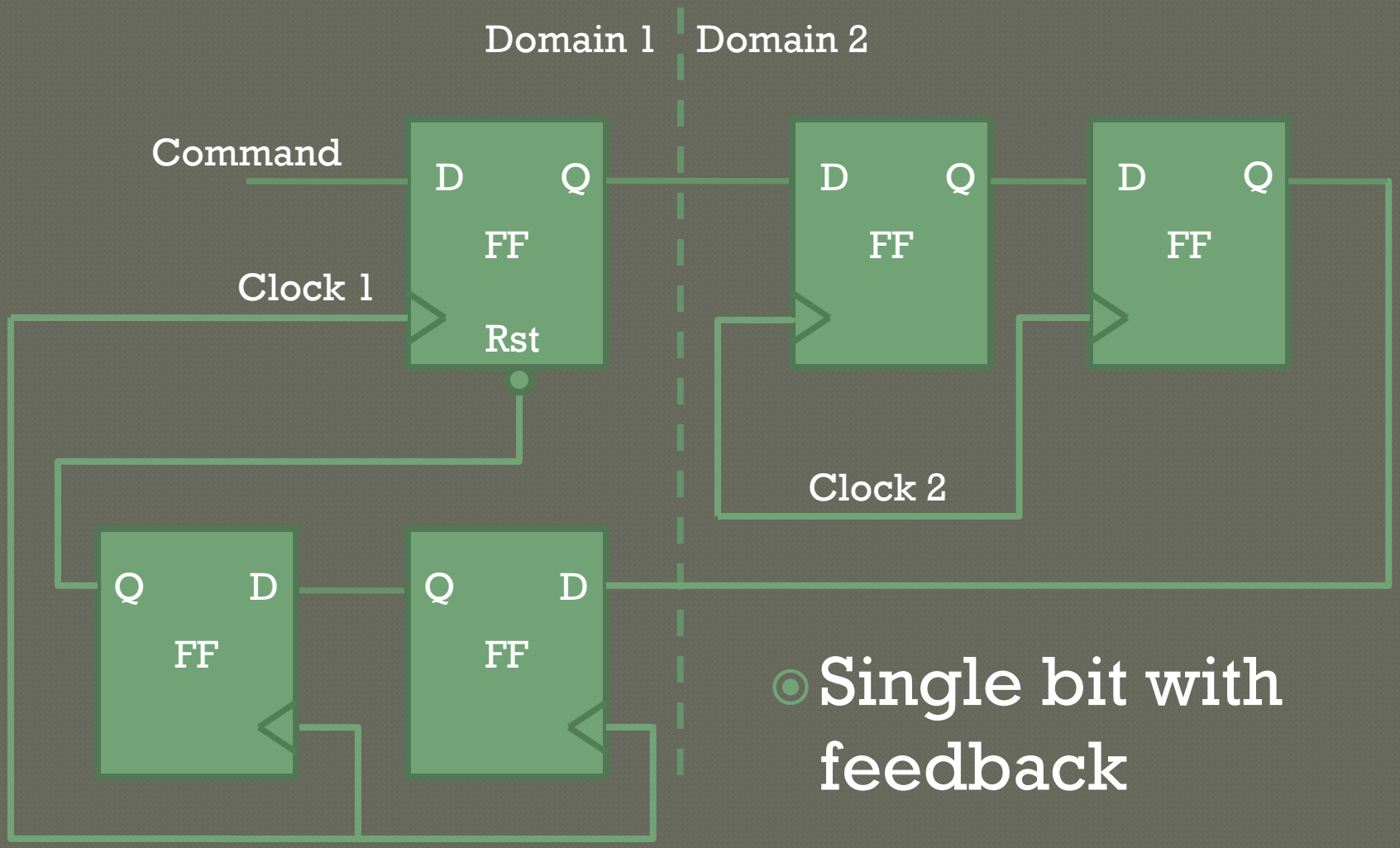
~2 ns difference if related clocks

Clock Domain Crossing

- Passing single-bit signals
  - Prevent metastability from reaching combinational logic
  - Use 2 or 3 level synchroniser



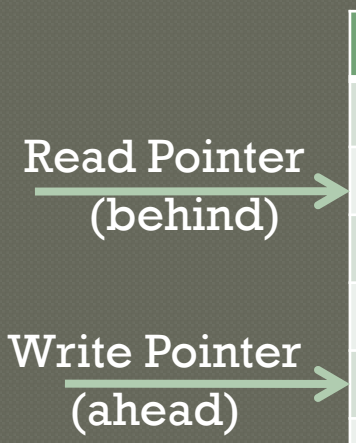
Clock Domain Crossing



- Single bit with feedback

Clock Domain Crossing

- Data Exchange
  - Asynchronous FIFO



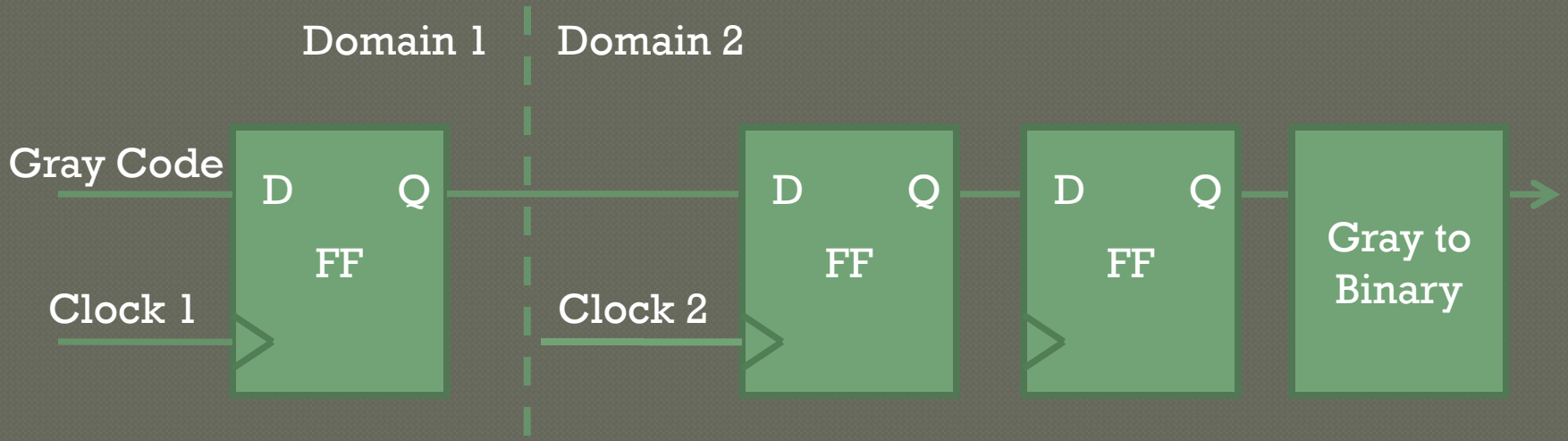
Memory Index
0
1
2
3
4
5
6
7

Binary Counter
000
001
010
011
100
101
110
111
000

Gray Counter
000
001
011
010
110
111
101
100
000

Clock Domain Crossing

- Asynchronous FIFO macros provided by vendors
- Custom Asynchronous FIFO
  - Don't forget the synchronizer stage



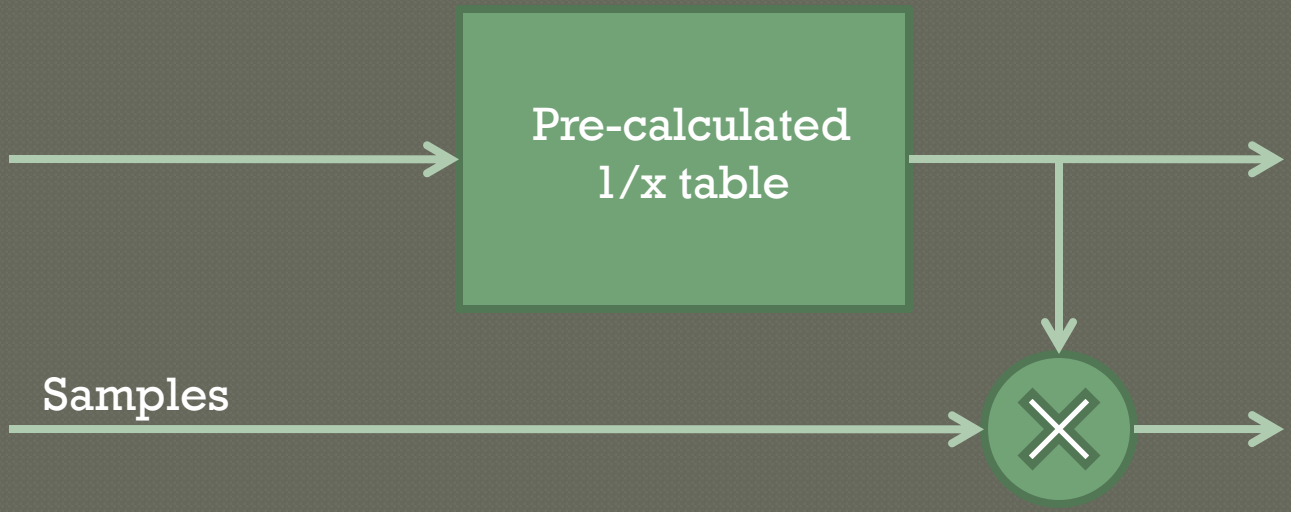
Clock Domain Crossing

- Multiple-bit command sequence crossing
  - Common "ready" feedback synchronizer
  - Asynchronous FIFO as a command buffer
- For any clock domain crossing, think on how to avoid the effects of metastability

Programming Techniques

- Alternate memory usage
- Single-cycle vs Multi-cycle
- Upclocking
- Resource time sharing
- Shift registers for compact control
- Double Buffering
- Deserializer as 1-bit ADC
- Leveraging latency

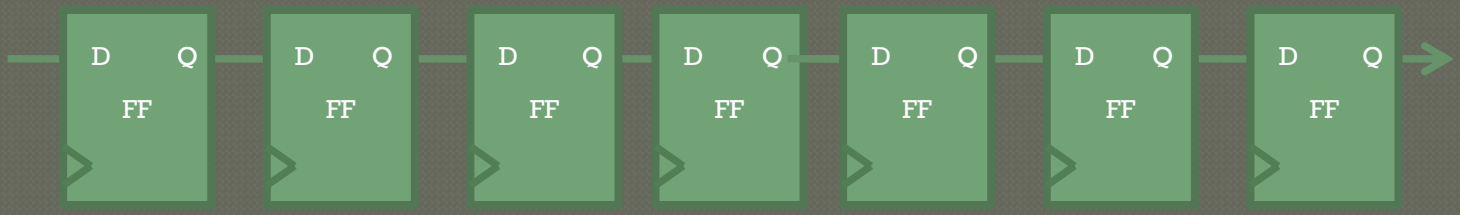
Alternate Memory Usage



- Division
- Trigonometry
- Precision = memory size

Alternate Memory Usage

- Shift registers

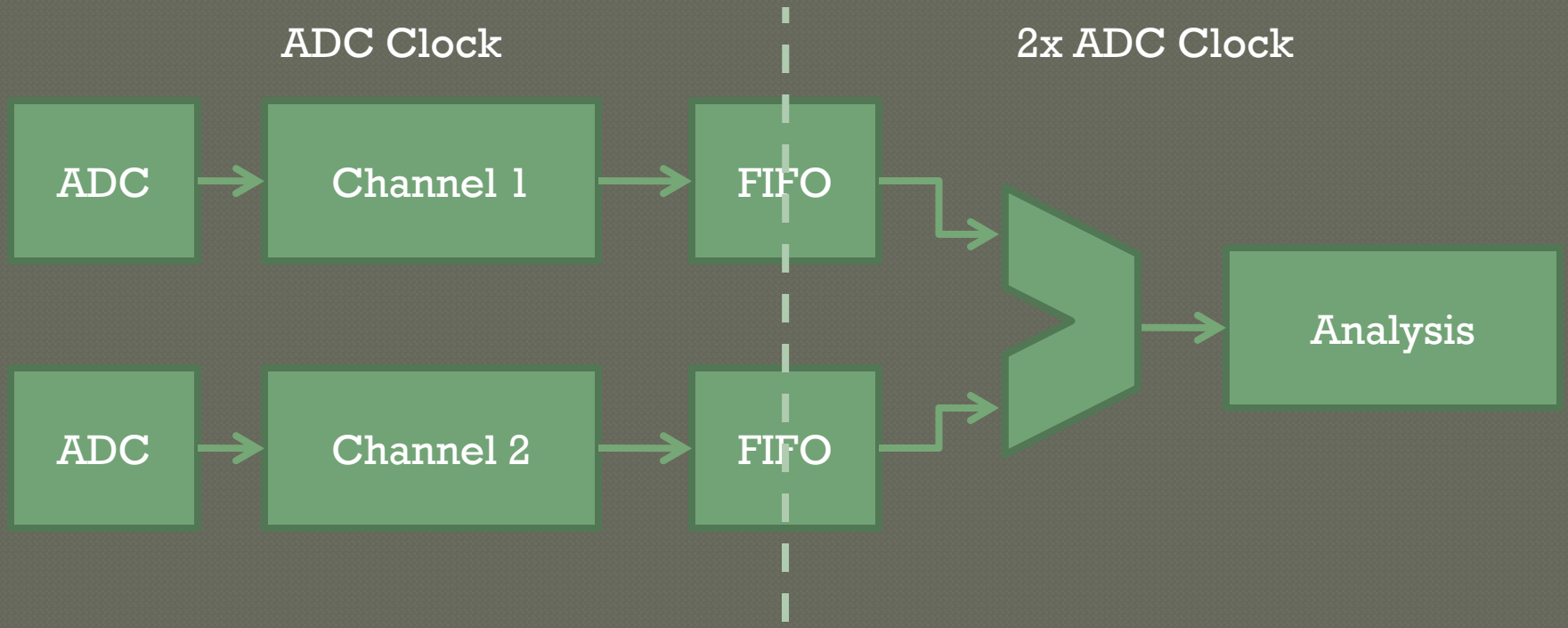


Space/Time Optimization

- Similar to real estate
  - Using its full speed potential
  - Activity ratio

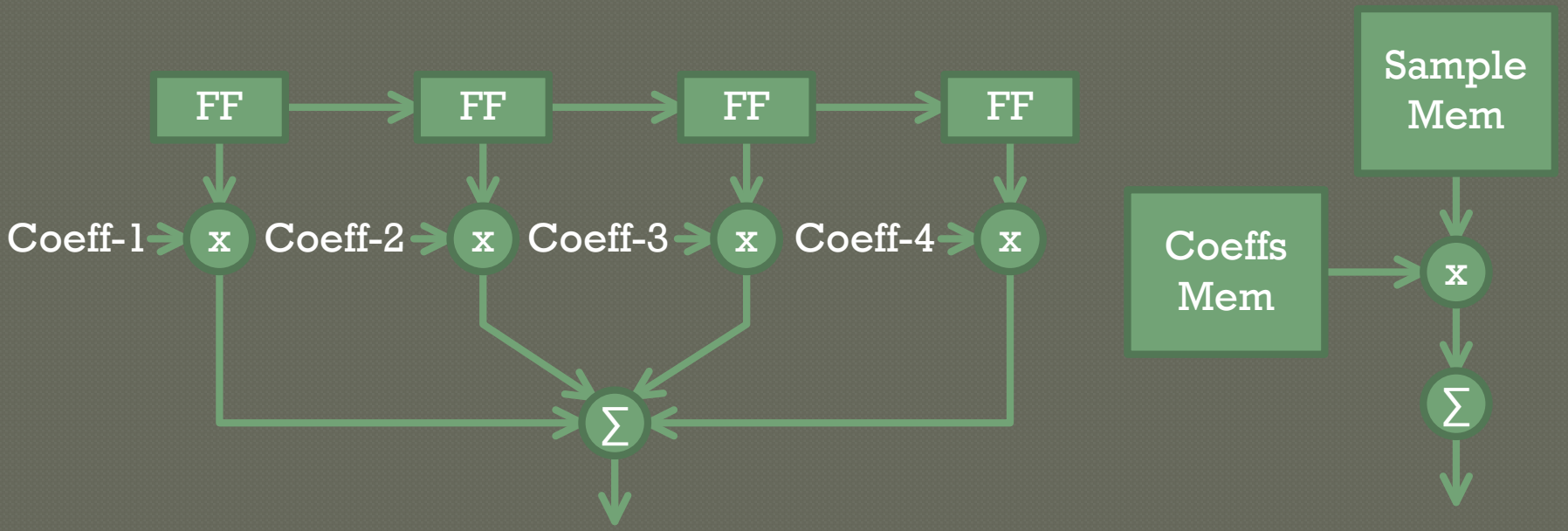
Upclocking

- When dataflow is slow compared to potential FPGA speed



Single-Cycle VS Multi-Cycle

- Resource optimization
  - ie: Finite Impulse Response (FIR) filter

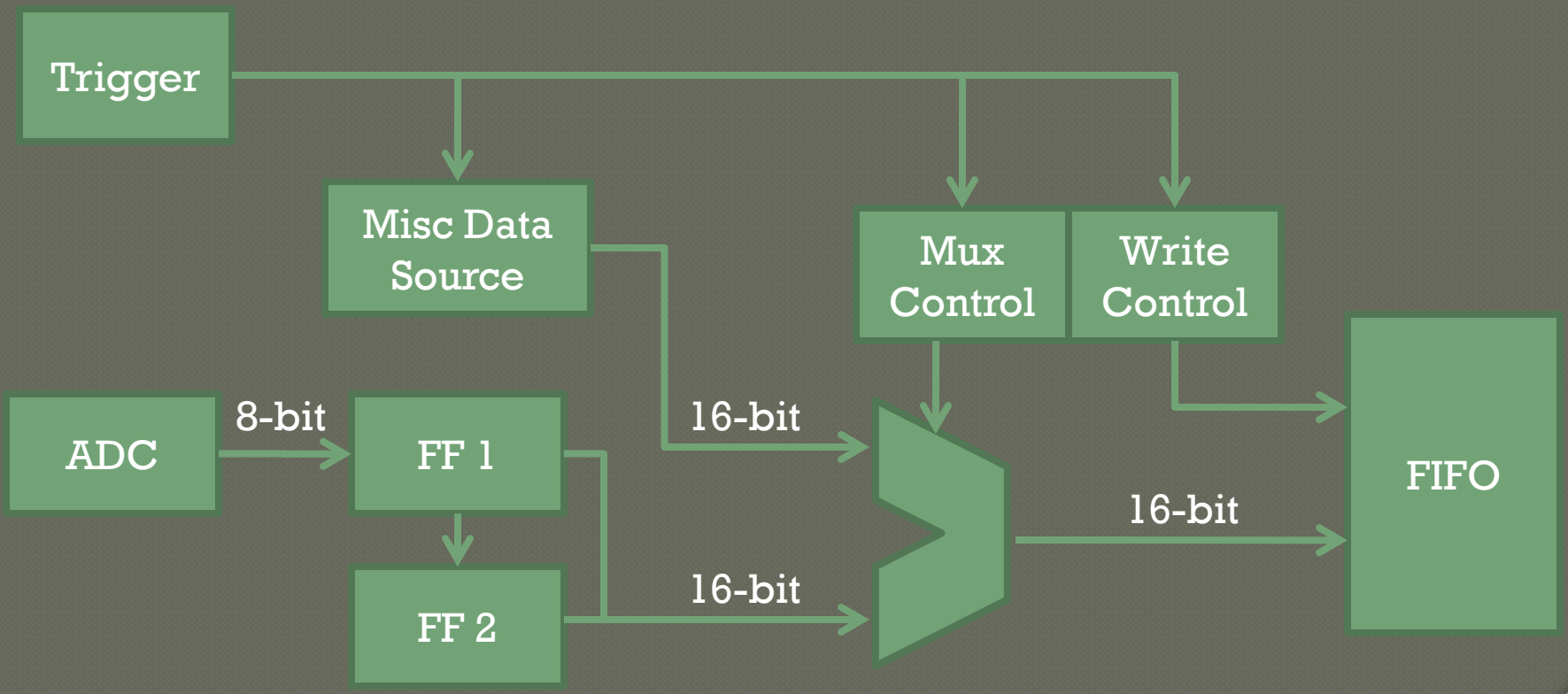


Resource Time Sharing

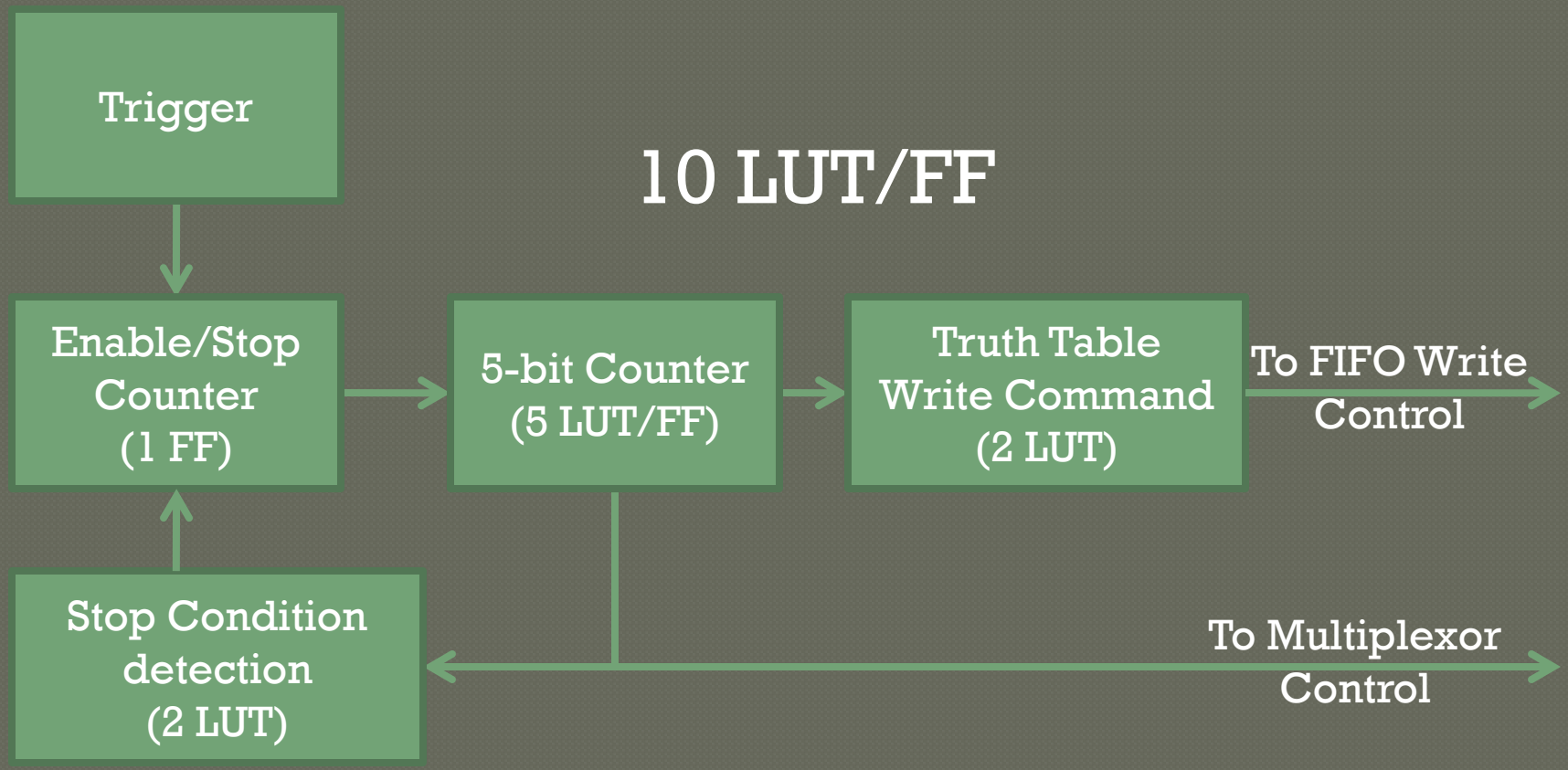
- Resource optimization
  - If there is "time" available, share resources for similar operations
  - Project-specific optimization
- Two filters with different coefficient sets

Compact Control

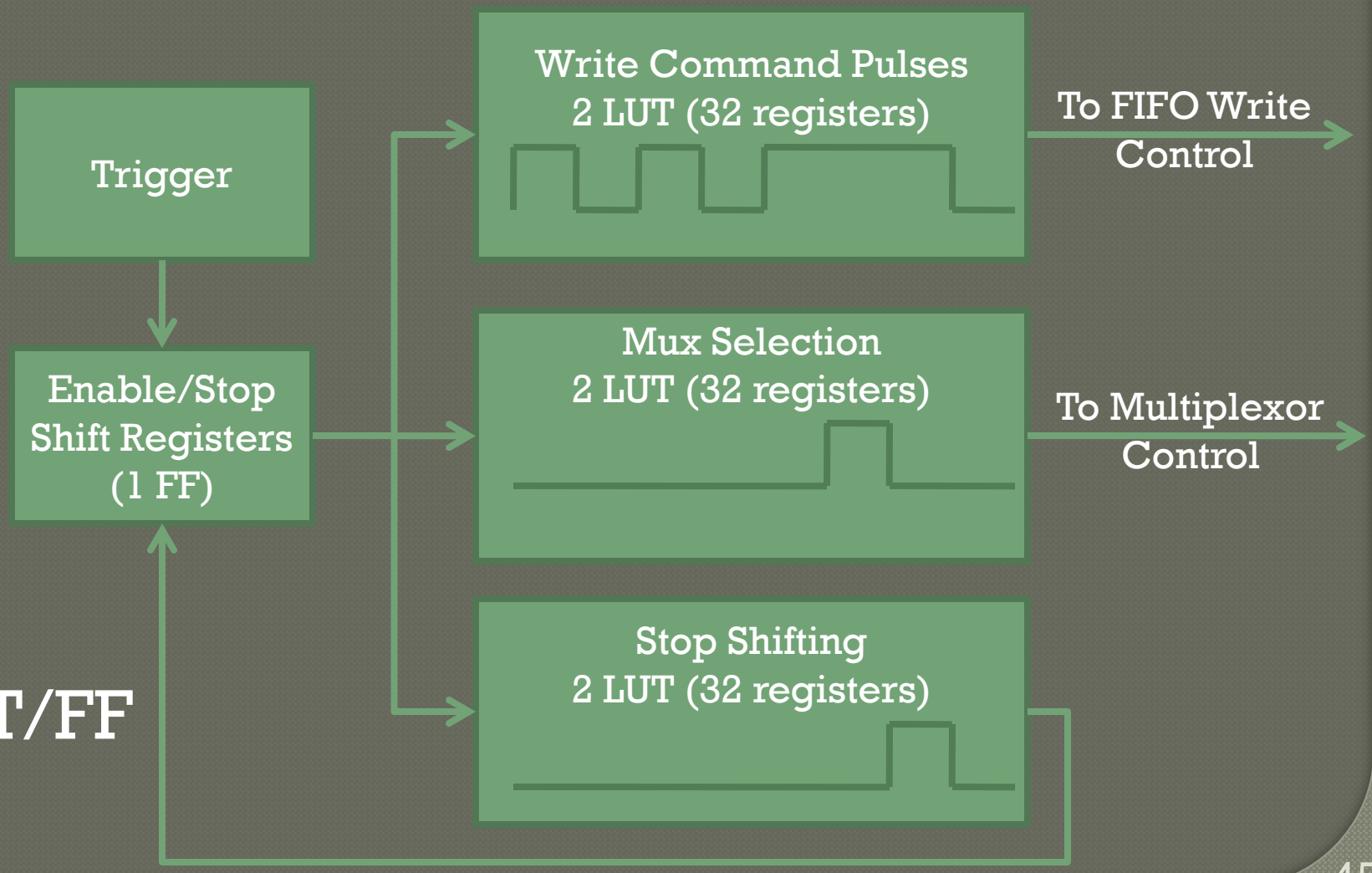
- Example : Signal sampler without digital dead time including miscellaneous data



Counter Control Model



Shift Register Control Model

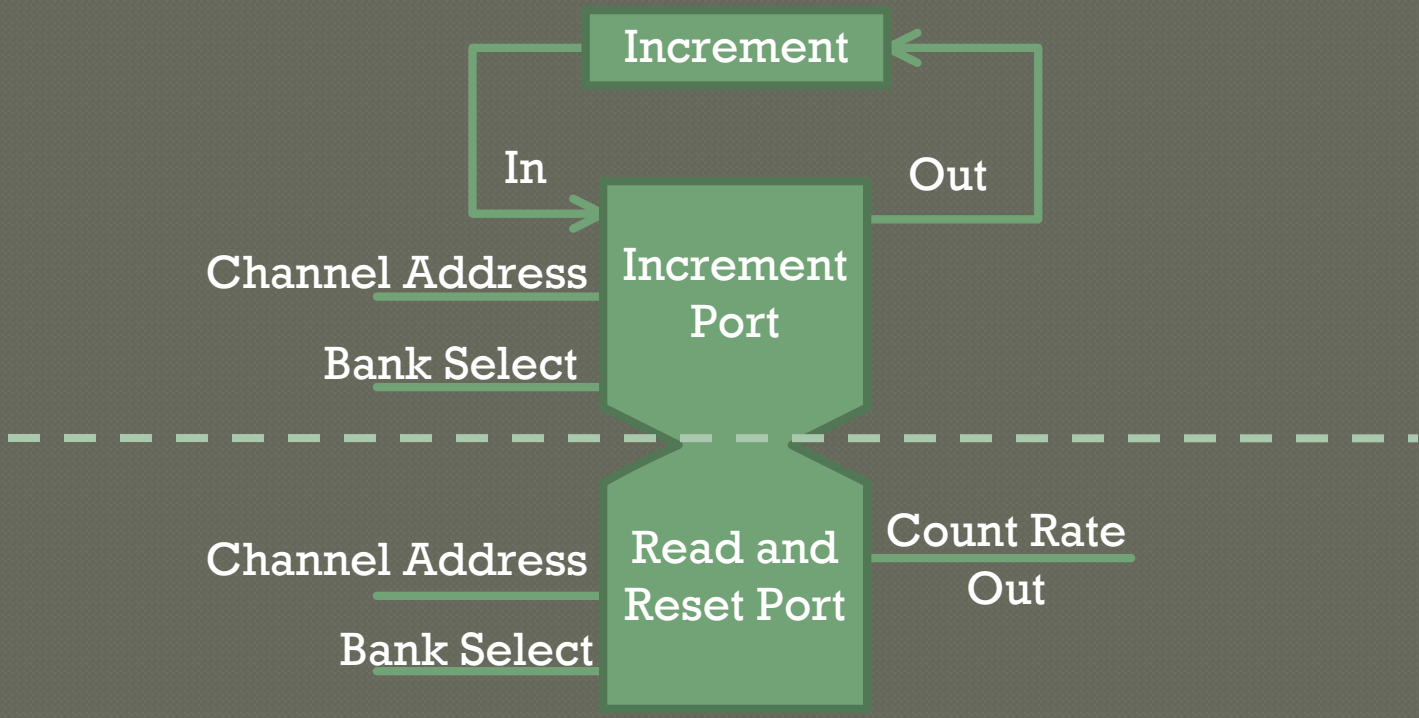


Bill Of Resources

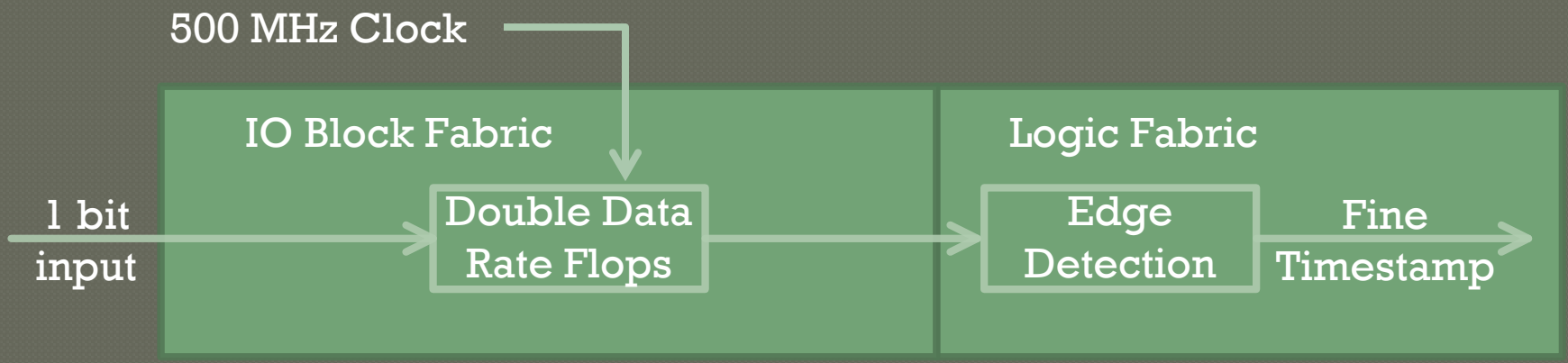
- Multiplexer Control
  - Counter Approach → 10 FF/LUT
  - Shift Register Approach → 7 FF/LUT
- 30% smaller
- Reset-less design

Double Buffering

- Uninterrupted index incrementation
  - Multi-channel count rate harvester
  - Real-time histogram builder



Edge Detection With I/O Pin

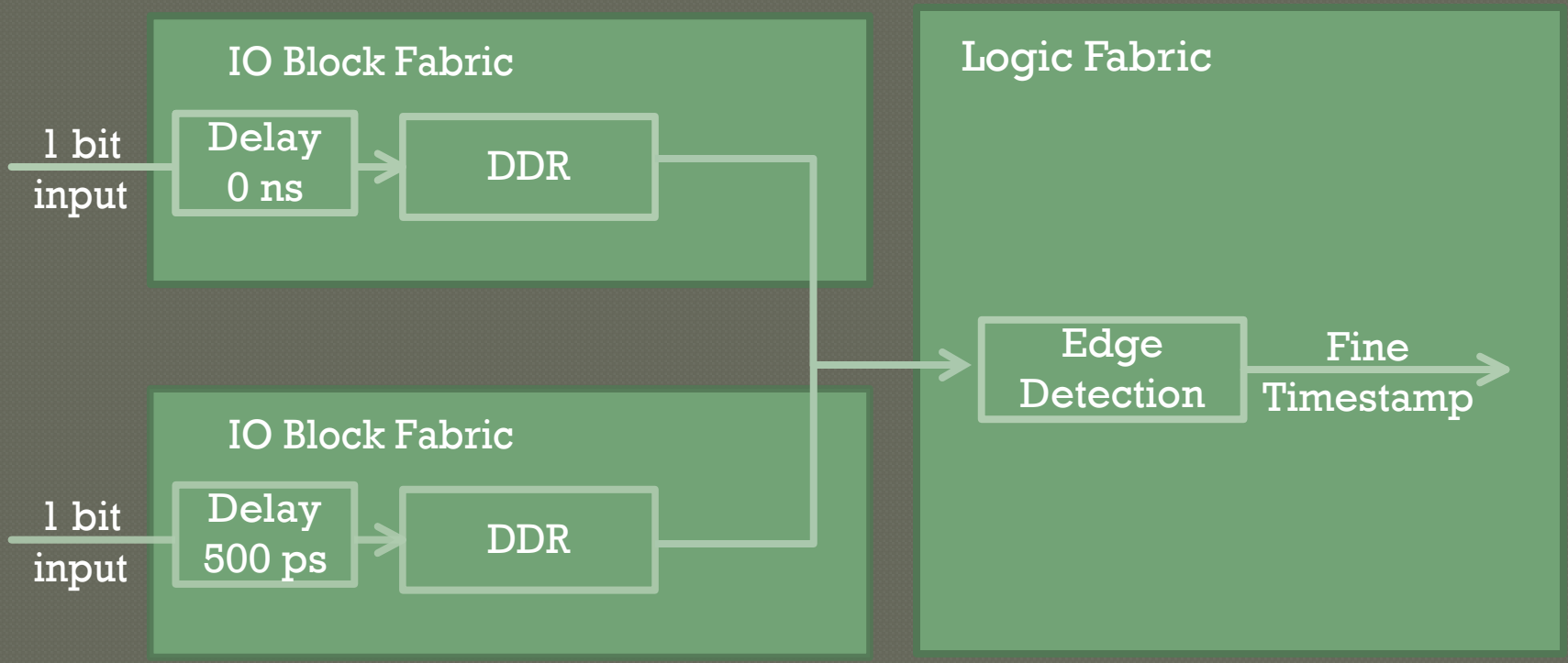


500 MHz x DDR = 1 GHz Sampling
→ 1 ns resolution

Edge Detection With I/O Pin

- Simple, efficient method of edge detection
  - 1 ns resolution out-of-the-box
- Phase control/delay units can increase resolution at the cost of IO pins
  - Xilinx calibrated delay lines
    - 80 ps/step, 64 steps

Edge Detection With I/O Pin



500 MHz x DDR x 2 offset units =
2 GHz Sampling → 500 ps resolution

Leveraging Latency

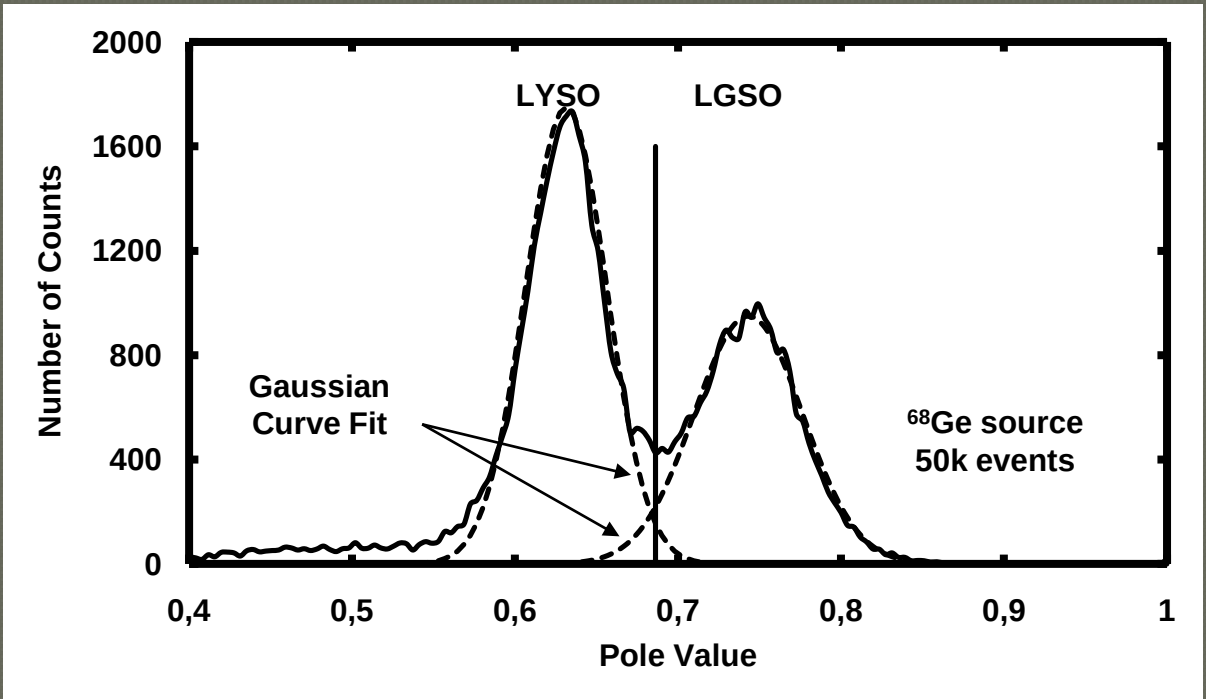
- Splitting complex processes

Full design example

Full design example

- Crystal Identification for PET
  - LYSO-LGSO dual crystal array coupled to APD
  - 40 ns and 65 ns decay times
  - Custom charge sensitive preamplifier
  - 22 ns sampling (45 MHz), 8-bits
  - 64 independent channels
  - Target : >1 Million events/s, before energy window
  - Minimize logic usage for future expansion modules
  - FPGA with 4-Input LUT

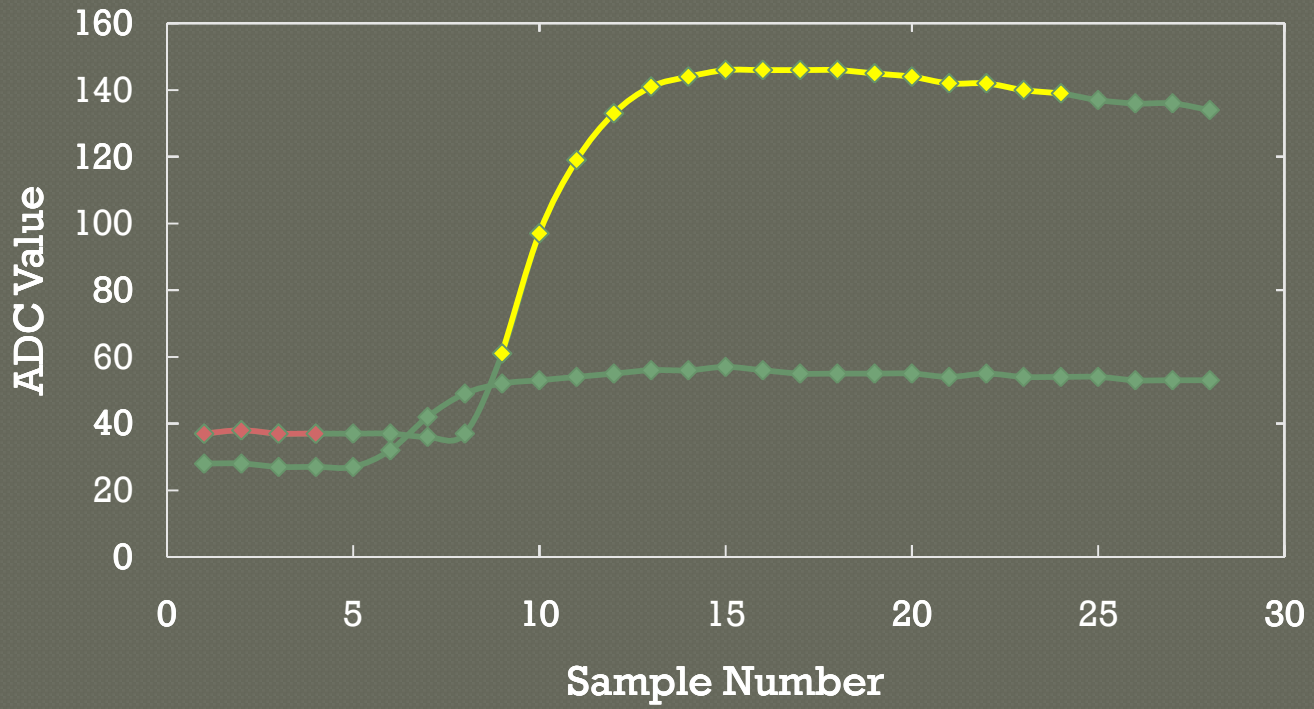
Z-Transform Histogram



- Fixed-point, Wiener Filter approach
  - N. Viscogliosi et al, "Real time implementation of a Wiener filter based crystal identification algorithm", *IEEE Trans. Nucl. Sci*, vol 55, no. 3, pp. 925-929.

Sampling Length

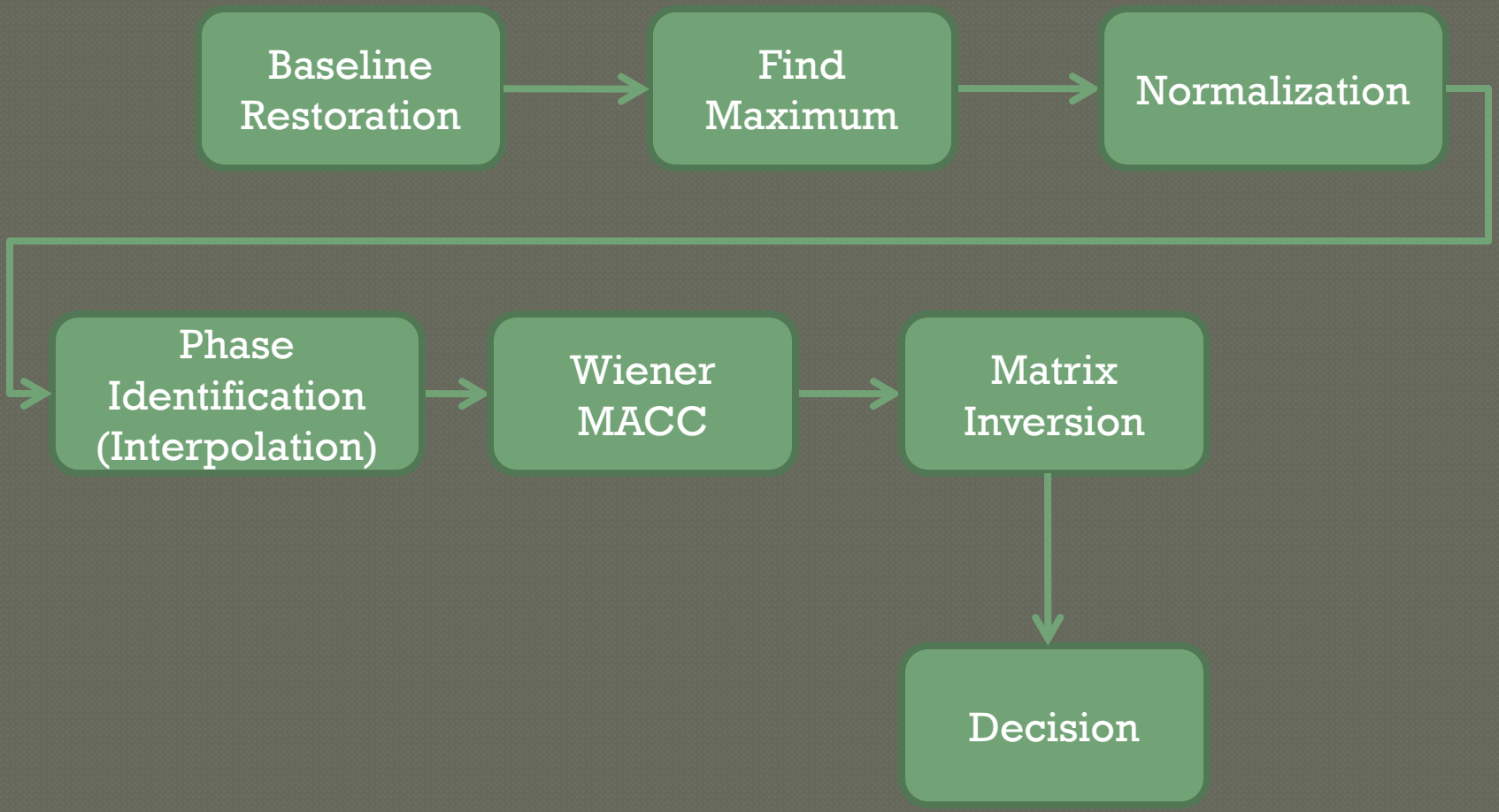
- How many samples are used?
  - 28 samples



Minimal Latency

- 10 intermediate values needed
  - Time stamp (3 bytes)
  - Energy value (1 byte)
  - Wiener phase alignment (1 byte)
  - Detector address (1 byte)
  - Wiener results (4 bytes)
- Requirement : minimal logic footprint
  - Minimum of 38 values per event

Process Overview

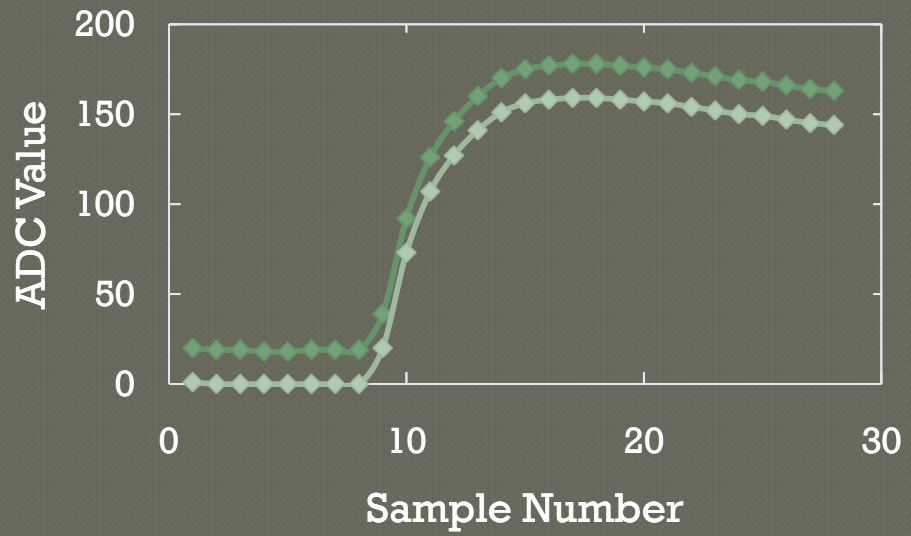


Techniques Used

- Shift registers as compact control
  - Seen earlier
  - 64 parallel channels
- Up-clocking
  - Processing 3x ADC clock (135 MHz)
- Pipelining
- Multi-Cycle
- Resource time sharing
- Process Splitting/Streaming

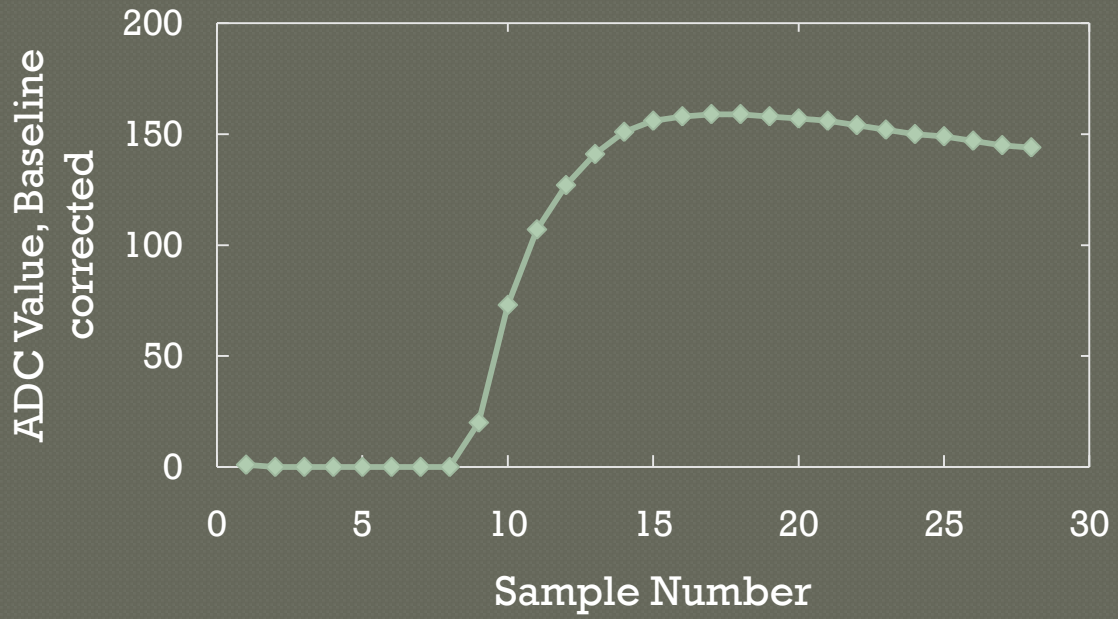
Preprocessing

- Step 1: Remove baseline
  - Mean 4 first samples : 4 clocks
  - Subtract with saturation : 1 clock per sample



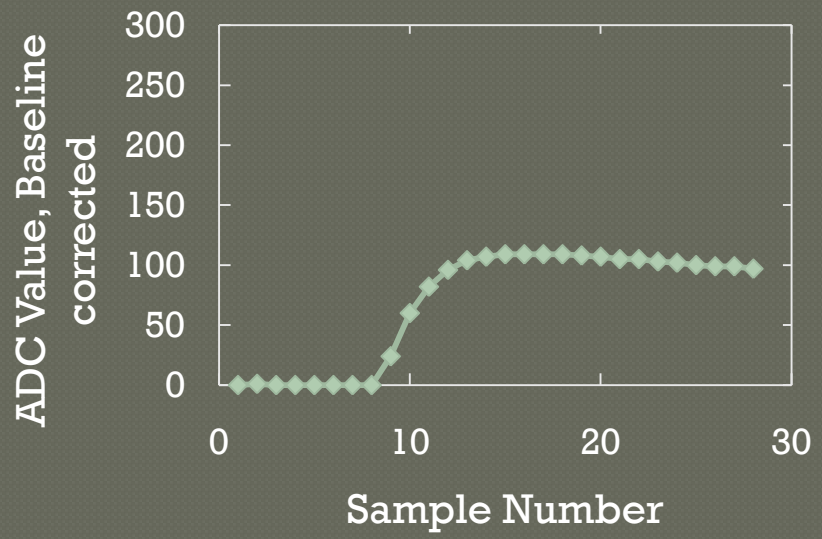
Preprocessing

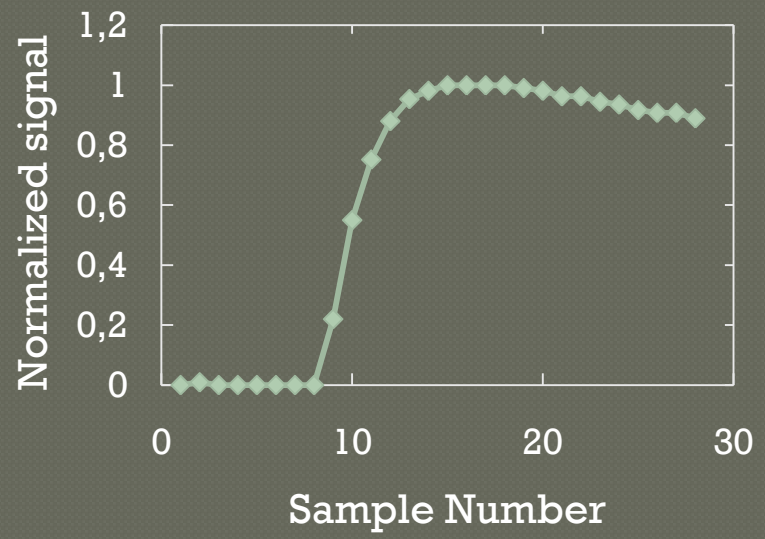
- Step 2: find maximum value in signal to determine normalization factor
  - Travel samples until first local max is encountered



Normalization

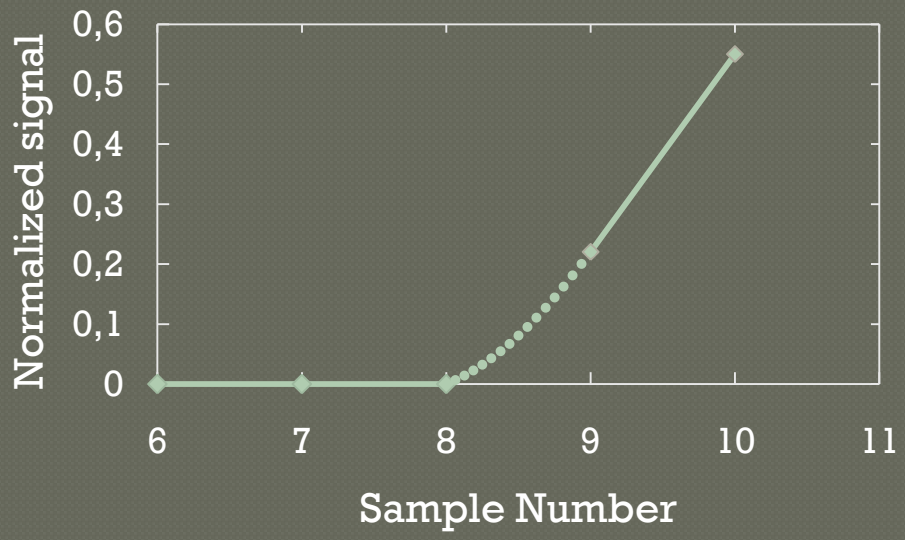
- Samples * 1 / Max
  - Hardware multiplier, fixed point
  - Look-up table for 1 / Max, 8-bit resolution
- 1 clock per sample



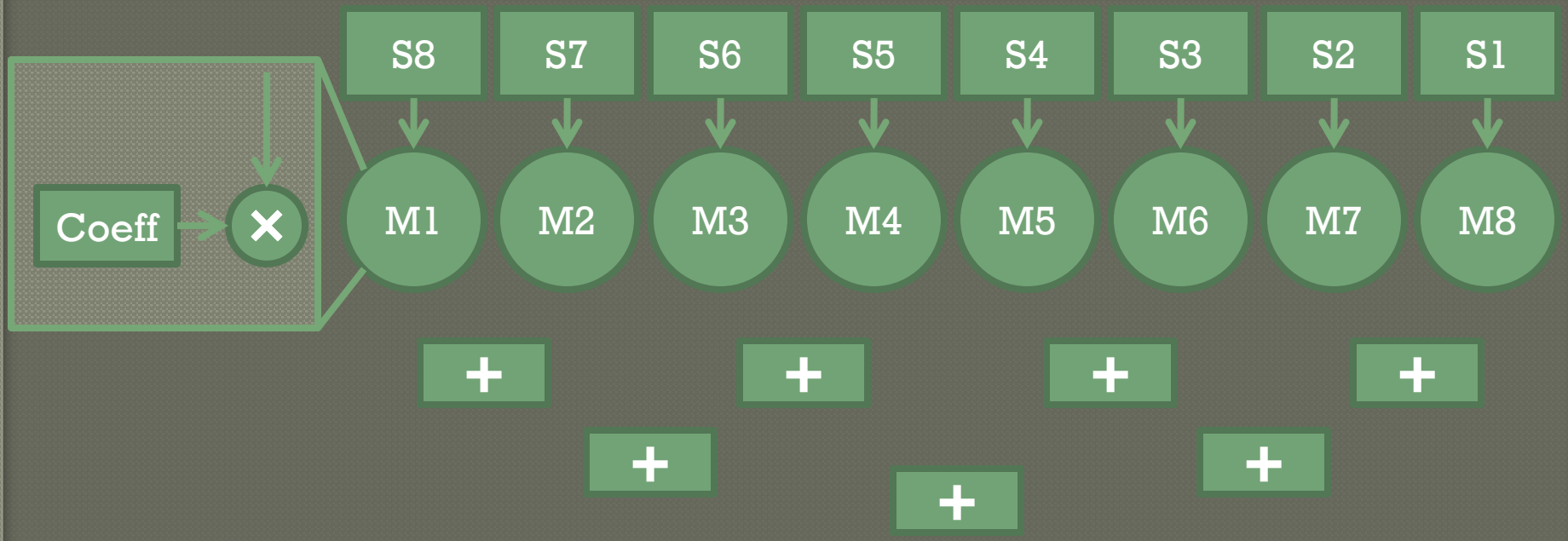


Wiener Phase Identification

- Find the signal's phase offset from a 20% threshold
  - Use interpolation around the 20% value
  - Filter-based interpolation



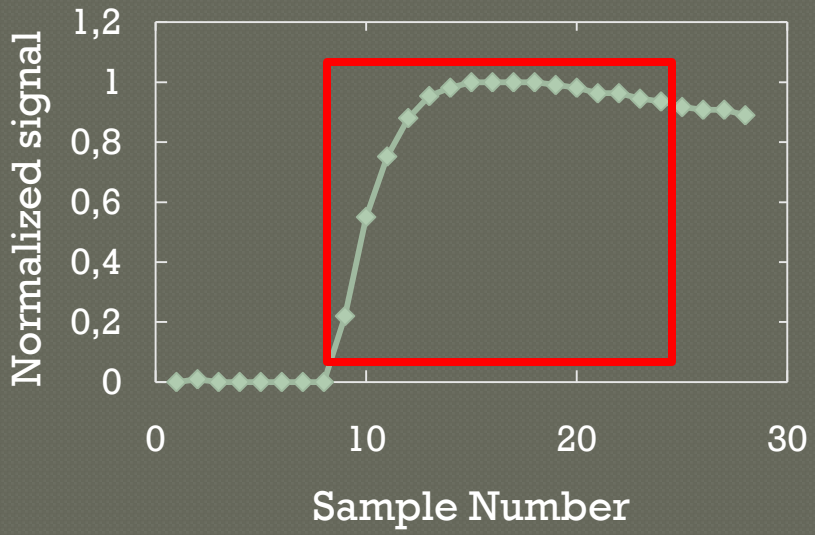
Interpolation



Interpolation

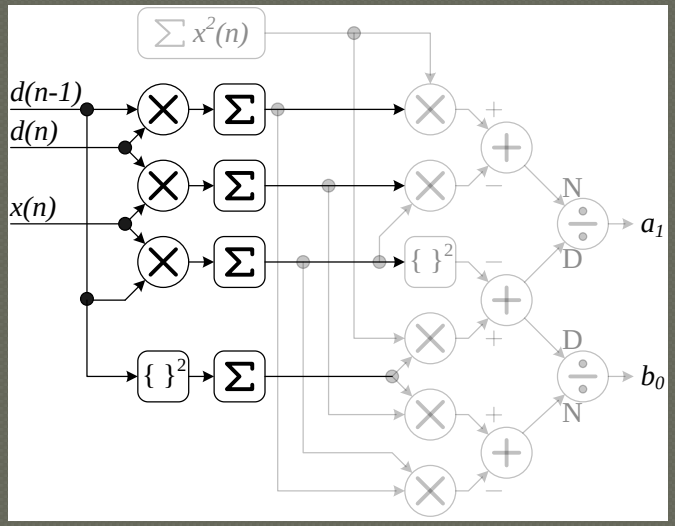
- Processing Time
  - Search for threshold interval
    - The 10^{th} sample on average
  - Load 8 samples
  - 4 clock latency (adder)
  - 1 clock per generated sample (16 samples)
  - Threshold verification (1 clock)
- ~28 to 30 clocks

Wiener MACC



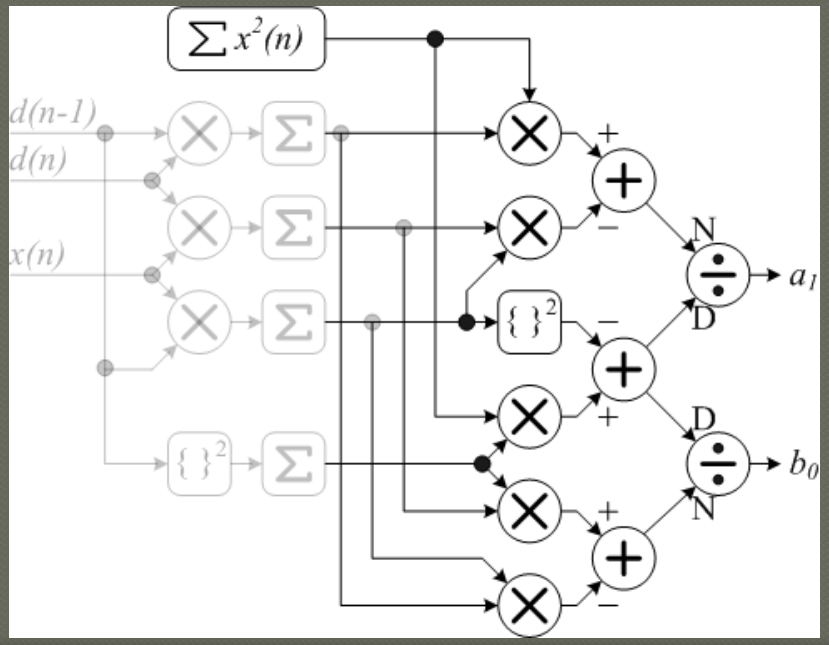
Samples

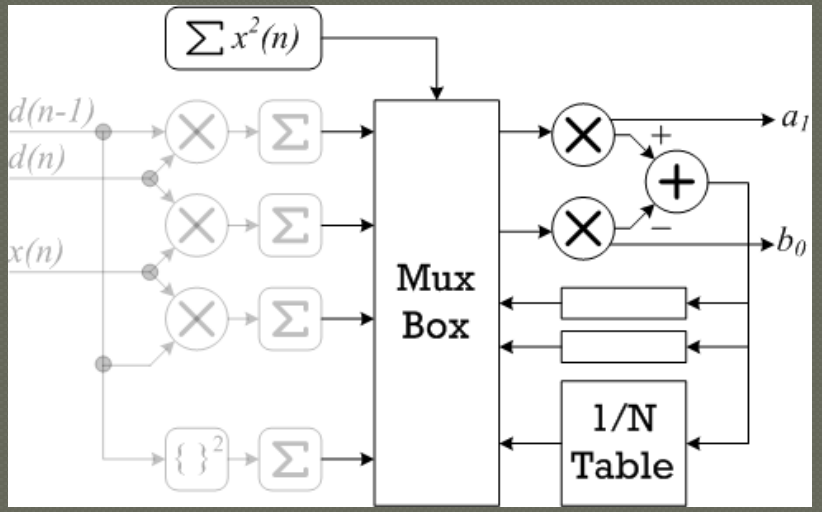
System Model



Wiener Matrix Inversion

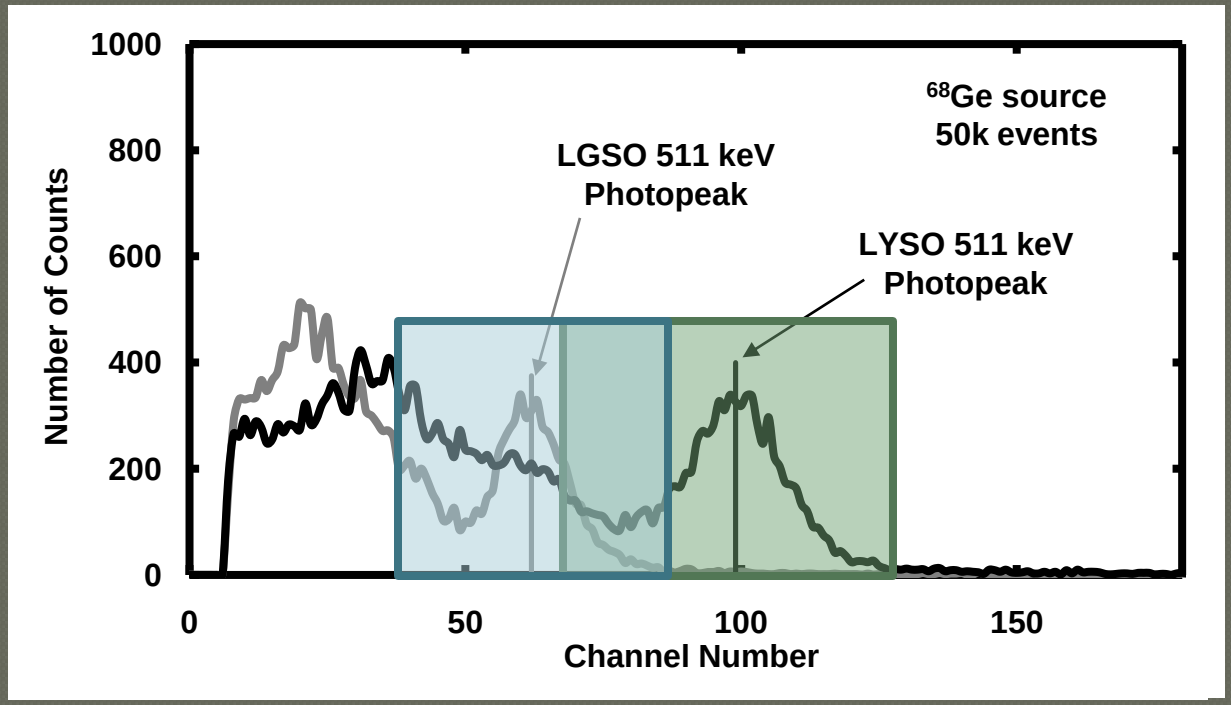
- 8 Multiplication, 3 Sums
  - Direct approach, 3 clock cycle latency
  - Shared resource approach : 2 multipliers, 1 subtractor and 2 input multiplexers, 6 clocks





Identification And Energy Windows

- Simple Look-up table in RAM memory



Latency Estimate

- Baseline evaluation: 4 clocks
- Baseline subtraction: 1 clock
- Maximum search: 10 clocks, minimum
- Normalization : 1 clock
- Phase search : 6 clocks, minimum
- Phase Interpolation : 28 clocks
- Wiener Macc : 16 clocks
- Wiener Matrix Inversion : 3 clocks
- Identi. and energy windows : 3 clocks

  - Total 72 clocks

Complex Process Splitting

- Adjust latency to strengthen modularity
  - Eases module insertion/update
  - Avoids event overlapping if module is modified
  - No need to realign/retime modules relative to each other

Final Performance

- Uniform latency for all process segments
  - Fixed to event length for maximum flexibility
- Event size : 38 bytes
  - Event analysis rate > 3 M events/s @ 135 MHz

Conclusion, 1st half

- FPGA building blocks
  - Know your hardware
- Basic design considerations
  - Know about the low level pitfalls
- Performance and resource optimization techniques
  - Think about what kind of performance you seek

References

- Cummings, C. E., "Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs", Synopsys Users Group Conference, San Jose, 2001, www.sunburst-design.com/papers
- Xilinx White papers, www.xilinx.com
  - wp272.pdf (Reset discussion)
  - wp275.pdf (HDL priority coding style)
- L. Arpin et al, "A Nanosecond Edge System With Embedded FPGA Fabrics", IEEE NPSS Real Time Conference Record, Beijing, 2009
- N. Viscogliosi et al, "Real time implementation of a Wiener filter based crystal identification algorithm", *IEEE Trans. Nucl. Sci.*, vol. 55, no. 3, pp. 925-929.
- www.opencores.org