Reconfigurable Computing

PLDCon'97

April 23, 1997

Ray Andraka, Chairman
the Andraka Consulting Group

"Relax, it's a dessert topping AND a floor wax"

-Chevy Chase alluding to reconfigurable computing, circa 1978

Agenda

- Overview
- Intro to Reconfigurable Computing
- Using Reconfigurability
- Design Examples

Overview

- Custom hardware performance
  - 10-1000x advantage over microprocessor
- Microprocessor Flexibility
  - Function changed by reconfiguration
- Cost Competitive
  - On par with DSP microprocessor system

Introduction to Reconfigurable Computing

John Watson,
Xilinx

Reconfigurable Logic

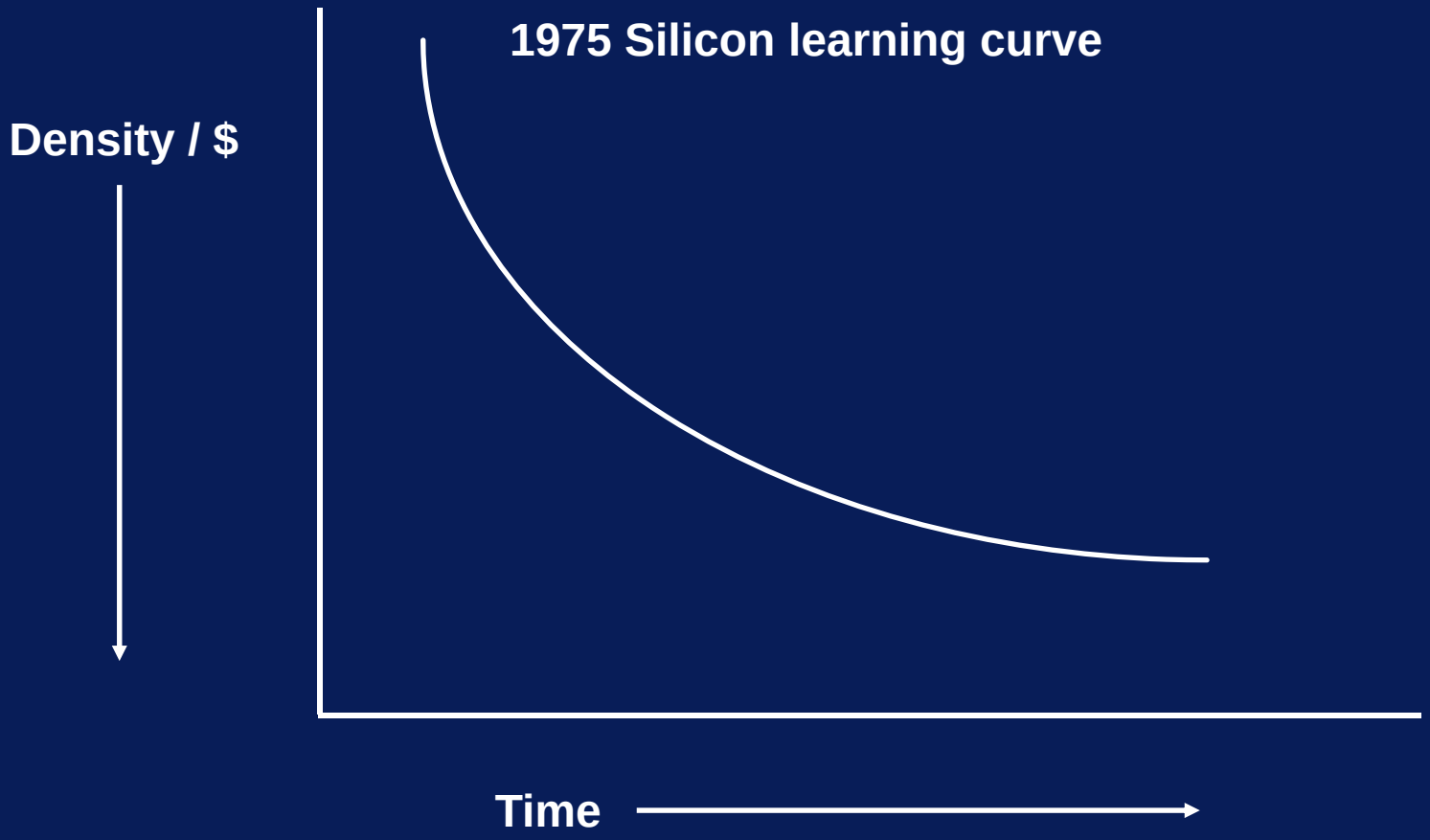
Reconfigurable Processing

April 1997

Electronics's Golden Rule

- **Get more done faster... cheaper...**
  in less space...
  and with less power.

Living Under Moore's Law



Tornado Forces in the Electronic Market

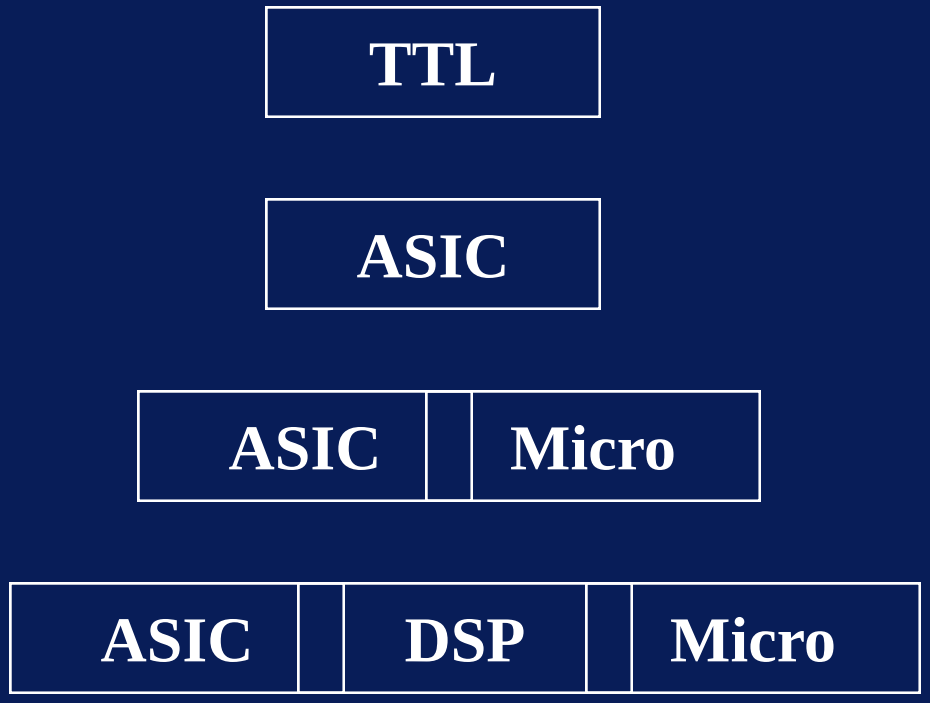
Market change is accelerating
- Shrinking product life cycles / TTM / TTV

Products are becoming global
- Worldwide competition
- Increased complexities / cost of errors
- Standards evolving & are replaced

Electronic information in vogue - Internet
- Homogeneous customer
- Lower cost of sales

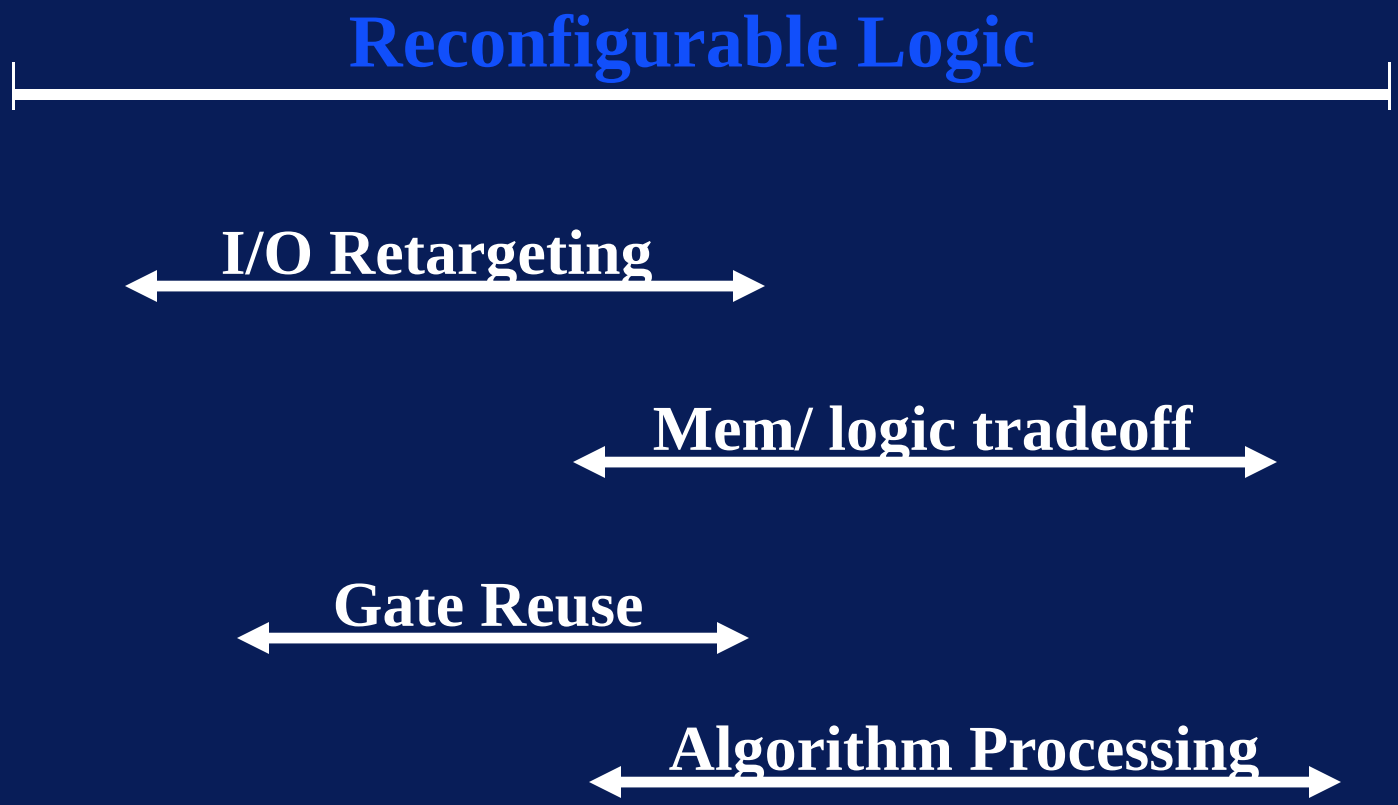
Markets becoming consumer oriented
- Entertainment value / Mobility / Price driven perceptions
- Hide technology
- Provide choices

Silicon Evolution



Silicon Evolution

Reconfigurable Logic



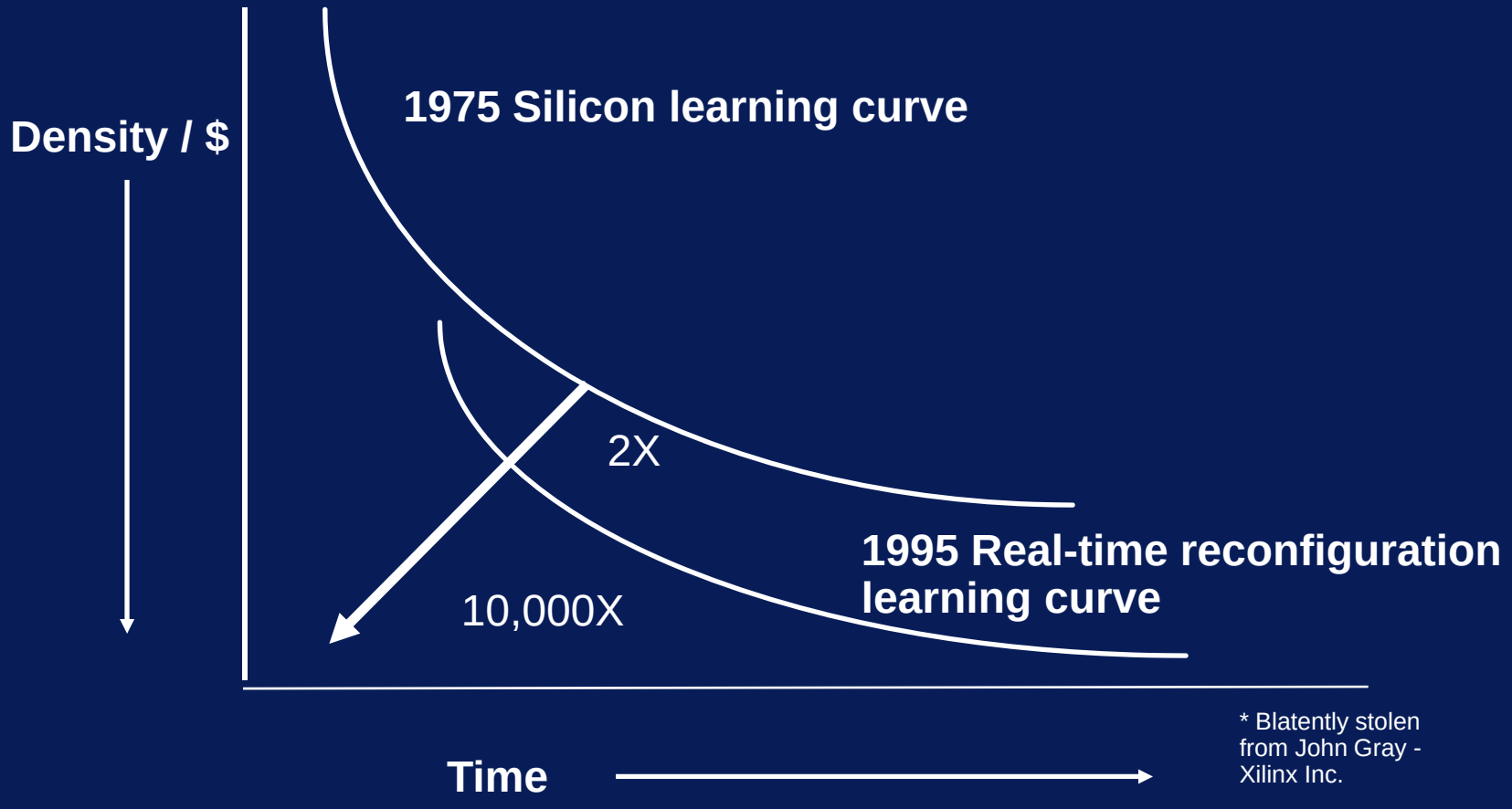
Gate Reuse

- Lowers silicon & product costs
  - Opens opportunity to sell future h/w upgrades
  - Reduces end-customer fear, “I made wrong choice”
- Provides a bridge between software algorithms and hardware implementations
  - Reprogrammable Co-processor
- Allows a “custom-fit” of specific logic for a specific application
  - Increased performance

Evolution of Moore's Law

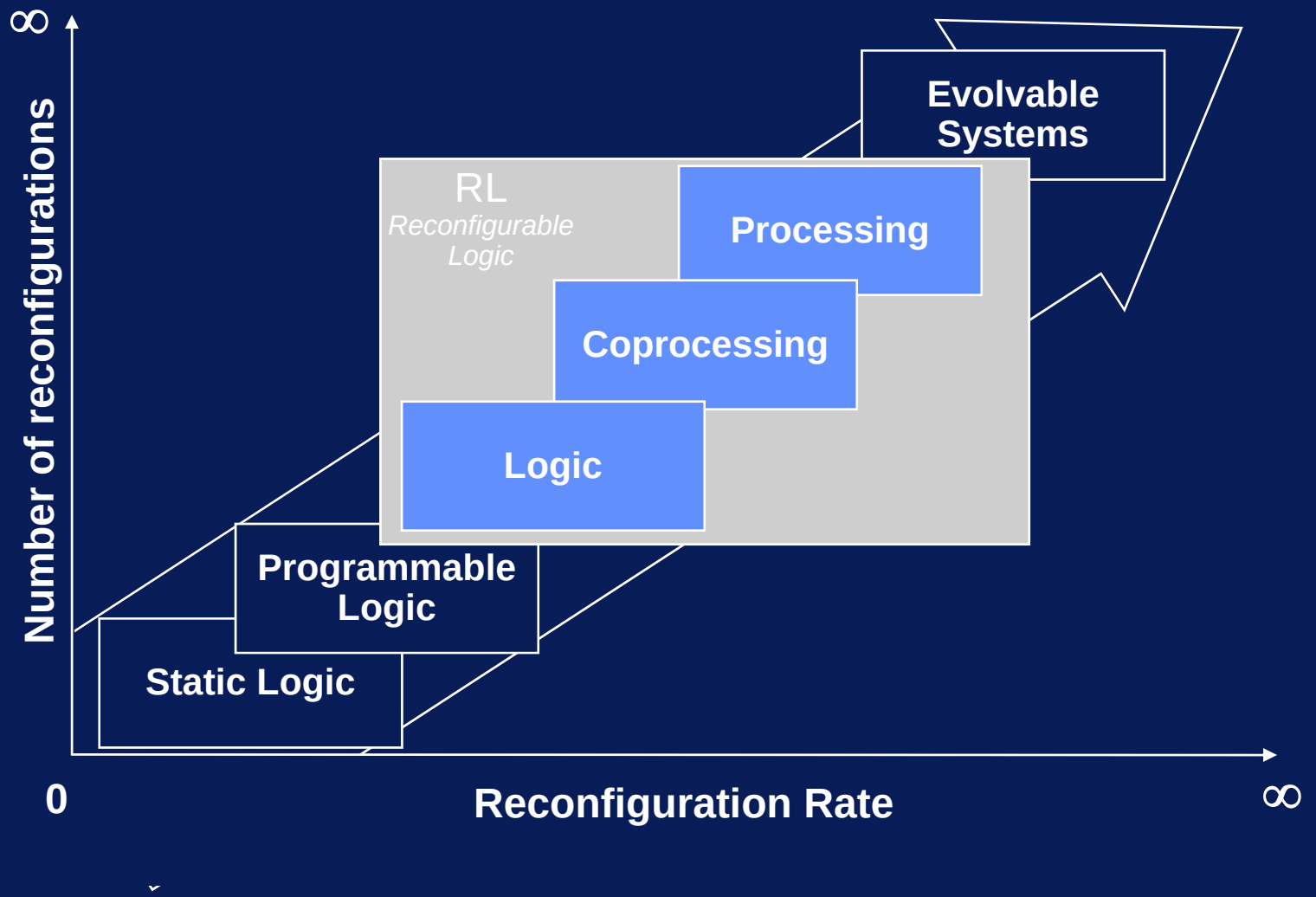
- Moore's Law does not work for cars, drinking glasses, houses...
- Moore's Law works for silicon because it moves 3 dimensions into "micro" dimensions.
  - Height, width, and depth move away from human terms into "micro" terms.
- But, we live in 4 dimensions...
  - What happens when time moves from human terms to micro terms?

Silicon Reuse in Time - More's Law*



* Blatently stolen from John Gray - Xilinx Inc.

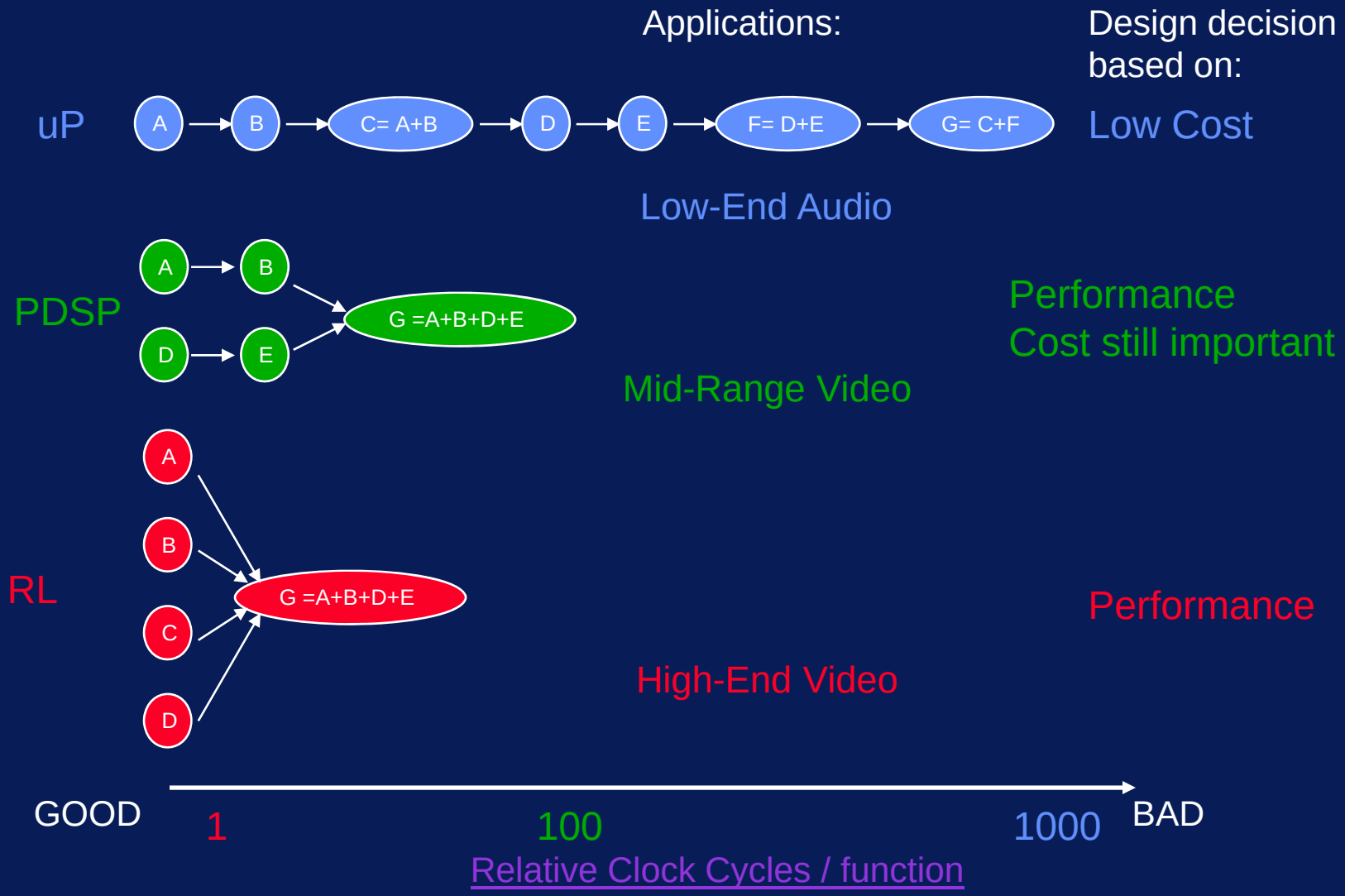
Gate Reuse


∞
Number of reconfigurations
Evolvable Systems
RL
Reconfigurable Logic
Processing
Coprocessing
Logic
Programmable Logic
Static Logic
0
Reconfiguration Rate
∞

Algorithm Performance

- Allows a “custom-fit” of specific logic for a specific application
  - Logic conforms to s/w instead of visa versa
  - Increased performance
- Provides a bridge between software algorithms and hardware implementations
  - Reprogrammable Co-processor
- Data stays resident
  - Logic moves throught the device, not the data

Algorithm Processing



Reconfigurable Hardware

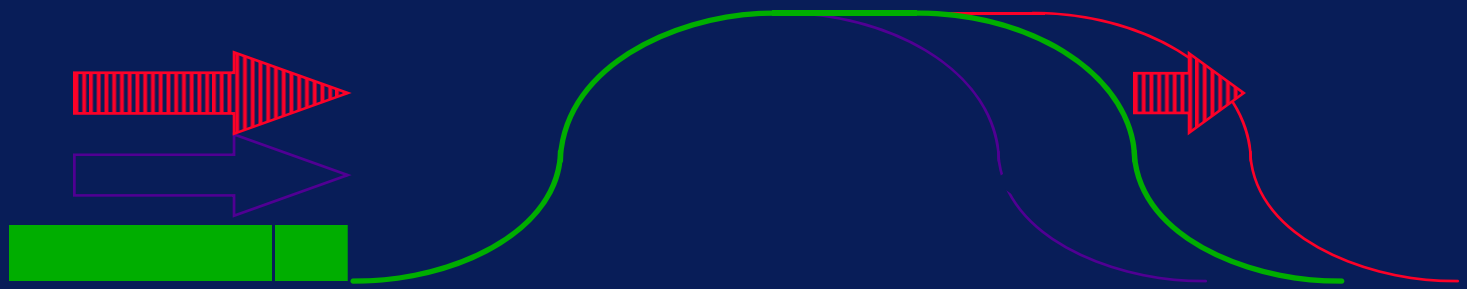
Reconfigurable Logic is the underlying fabric of the future

- Industry talks about physical space (3 dimensions)
- Reconfigurable logic adds the 4th dimension - TIME

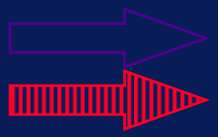
Word Processor	**PC**	**Future**
Fixed h/w	Fixed h/w	Reconf. h/w
Fixed s/w	Reconf. s/w	Reconf. s/w
GOOD →	**BETTER** →	**BEST**

Extending Life Cycles

Gate Reuse can extend product life-cycles - thereby keeping customers.

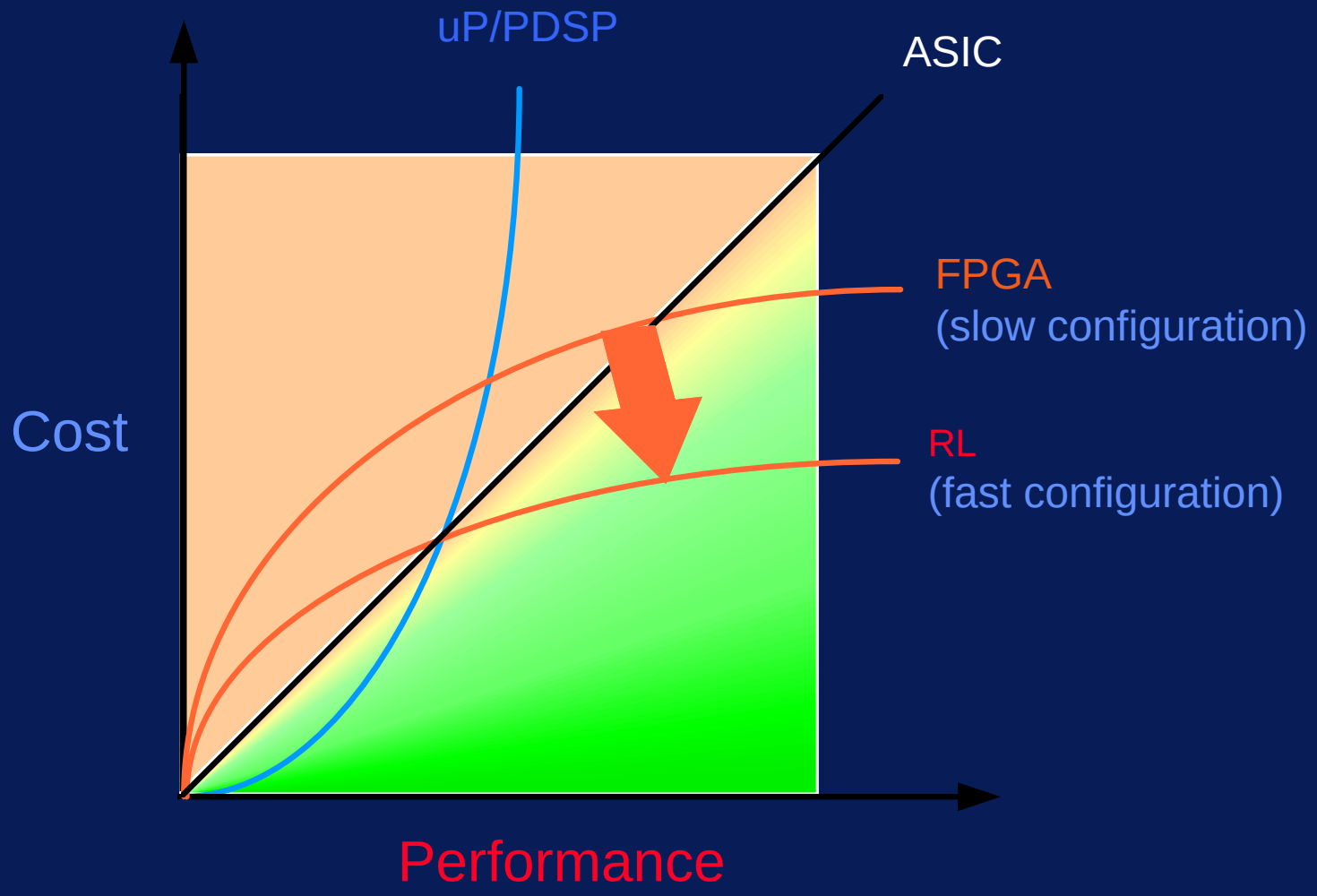


Product Life-Cycle



Reducing TTM is a traditional use of FPGAs
Extending product life cycles is a Non-traditional use of RL

Lowering The Cost Of High-Performance



Tornado Forces in the Electronic Market

Market change is accelerating
- Shrinking product life cycles / TTM / TTV

Products are becoming global
- Worldwide competition
- Increased complexities / cost of errors
- Standards evolving & are replaced

Electronic information in vogue - Internet
- Homogeneous customer
- Lower cost of sales

Markets becoming consumer oriented
- Entertainment value / Mobility / Price driven perceptions
- Hide technology
- Provide choices

= DESIGN FOR CHANGE

= DESIGN FOR USER CUSTOMIZATION

= SELL OVER THE NET

= DESIGN FOR INSTANT CHOICES

So, Why isn't everyone doing this?

- Devices have just become large and fast enough
  - 4010/4013's
  - 20MHz
- Design tools are at the embryo stage
- This takes a thinking "paradigm" shift

The 3 Stages in a Paradigm Shift

1 - Total rejection of a new idea by entrenched leaders

- If this was a good idea, then we'd have thought of it
- No market research data needed

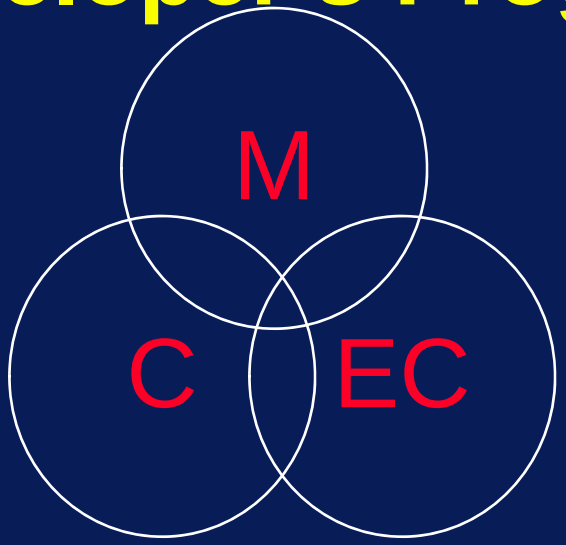
2 - When the entrenched leaders can't ignore it, they'll do it their way

- New idea shoved into old box
- Focus stays on improving stage 1 solution, not on the problem
- Justifies stage 1 - Market research data/researchers FUD used

3 - New paradigm used to solve the problem in a new way

- The new paradigm severely damages the entrenched leaders
- Usually a new company that didn't know better

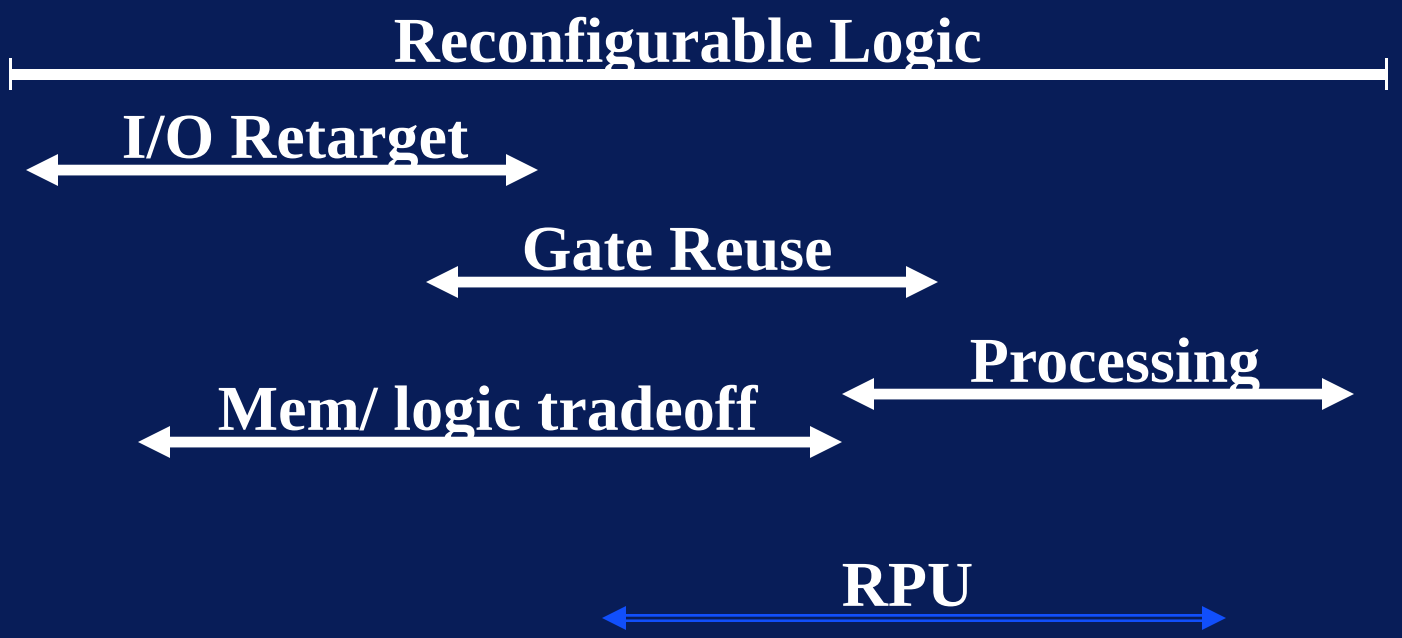
Developer's Program



Segment	Applications	Benefit
Multimedia	Video editing / effects / machine imaging	Speed / cost / extensibility
	FFT/convolutions/etc	
	128 track MIDI recording studio	Speed / cost / extensibility
Communications	ATM / multi-service / protocols	Flexibility / extensibility
	Encryption / compression	
Embedded Control	Industrial, DSP, reconfig computing.	Flexibility / cost

Reconfigurable Processing Unit

RPU's focus on processing, and gate reuse.



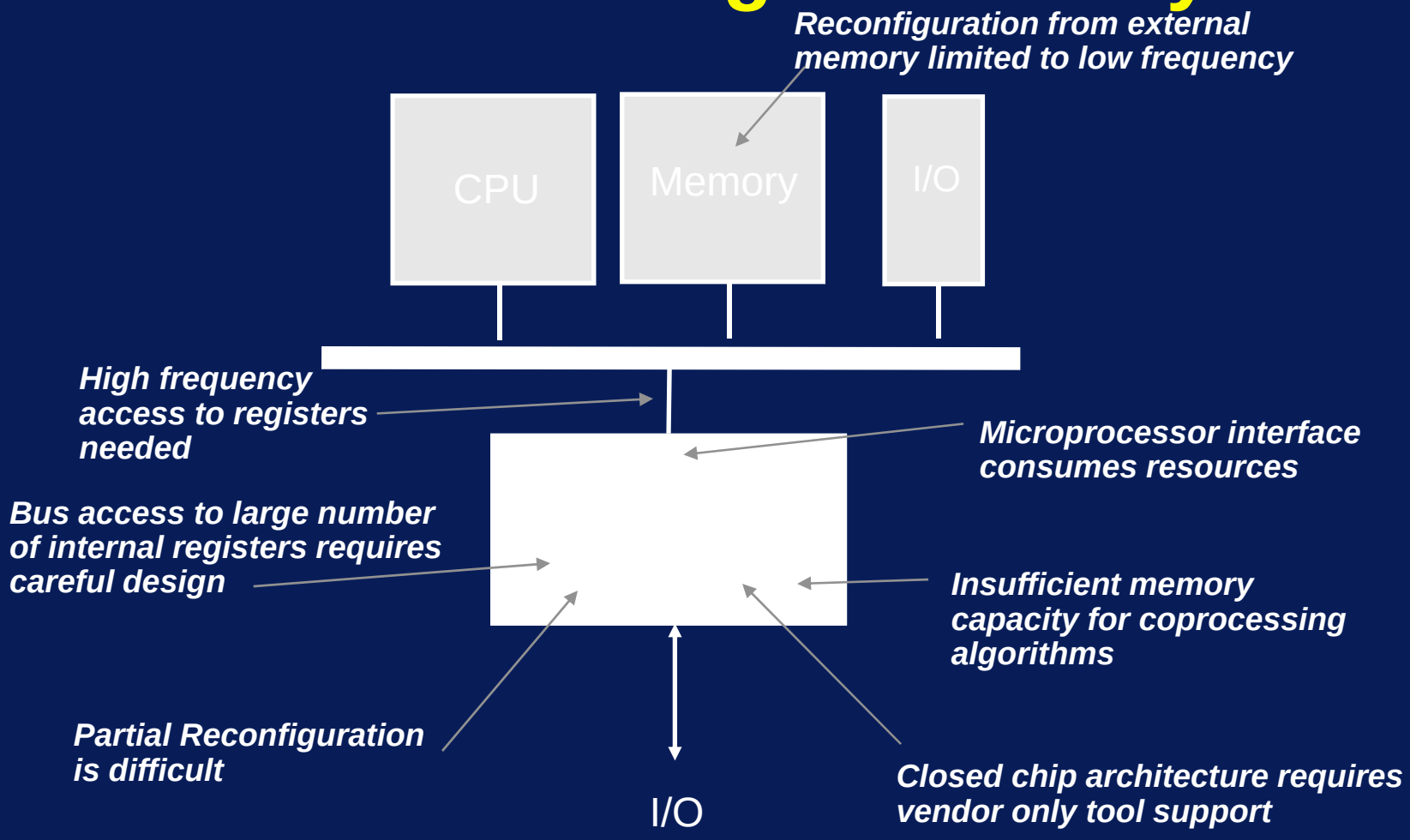
The XC6200 is

the first FPGA architecture

optimized for RPU

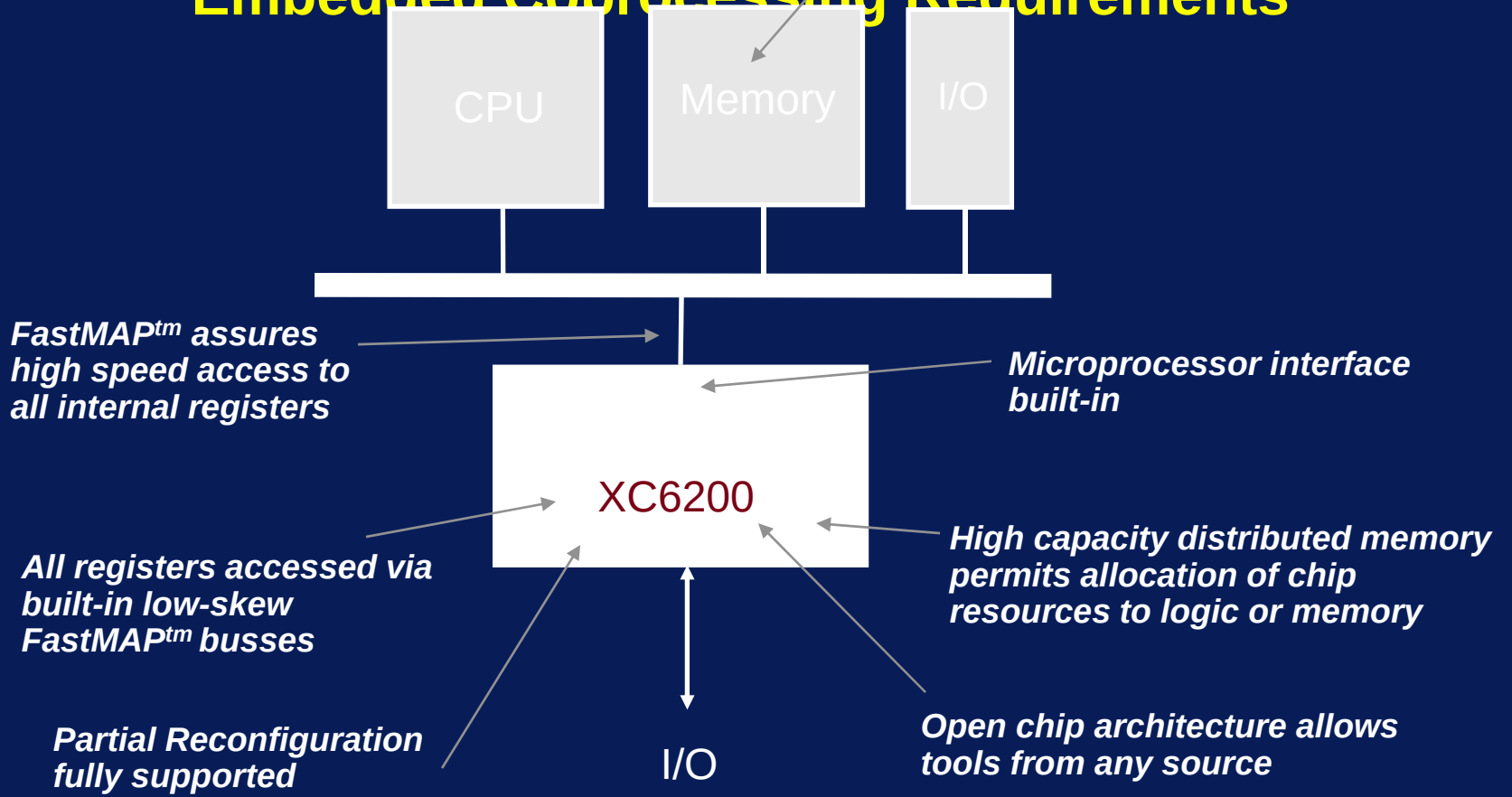
applications.

Problems Confronting Embedded Control Designers Today

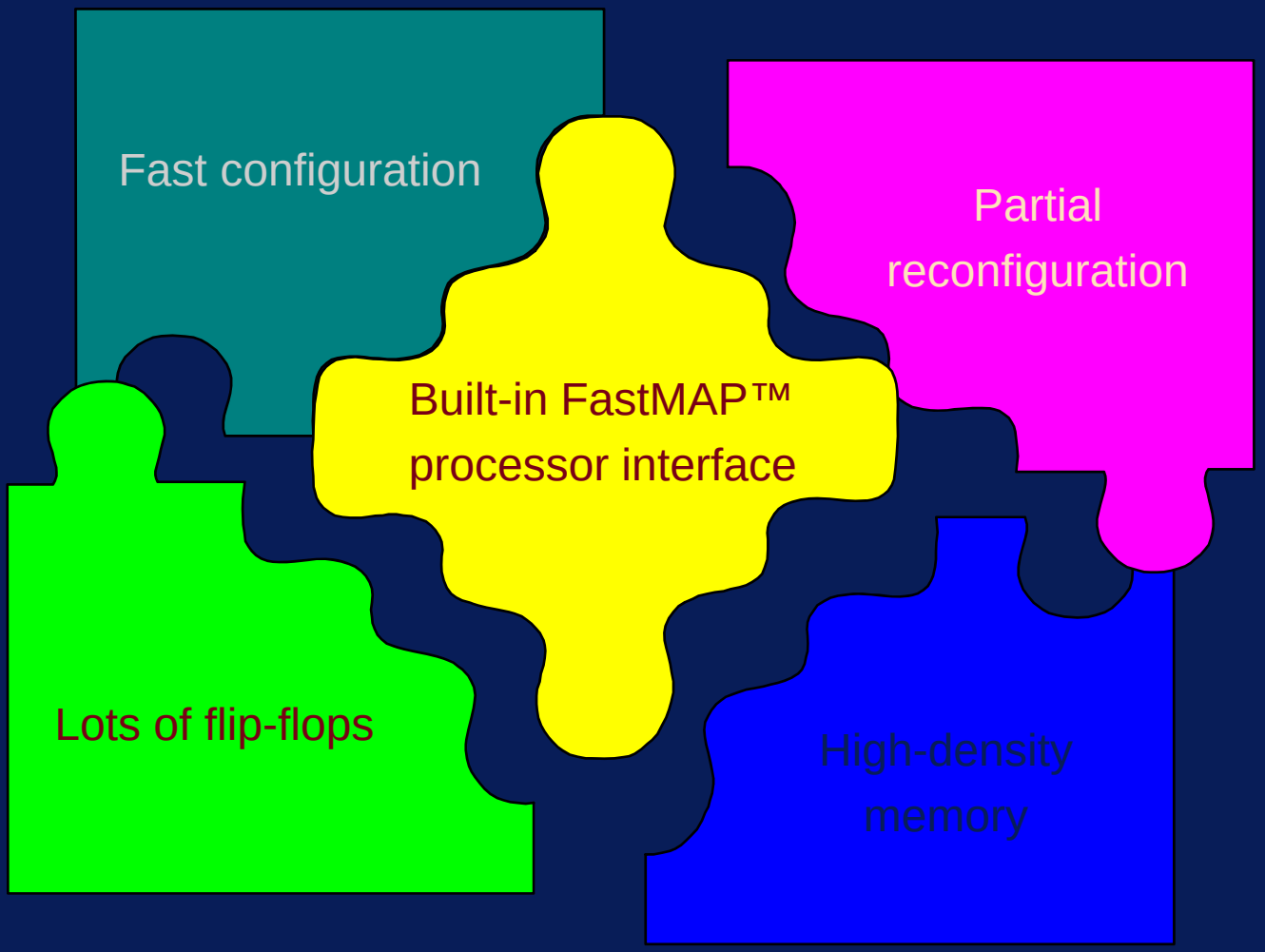


6200 Solutions for Embedded Control Designers

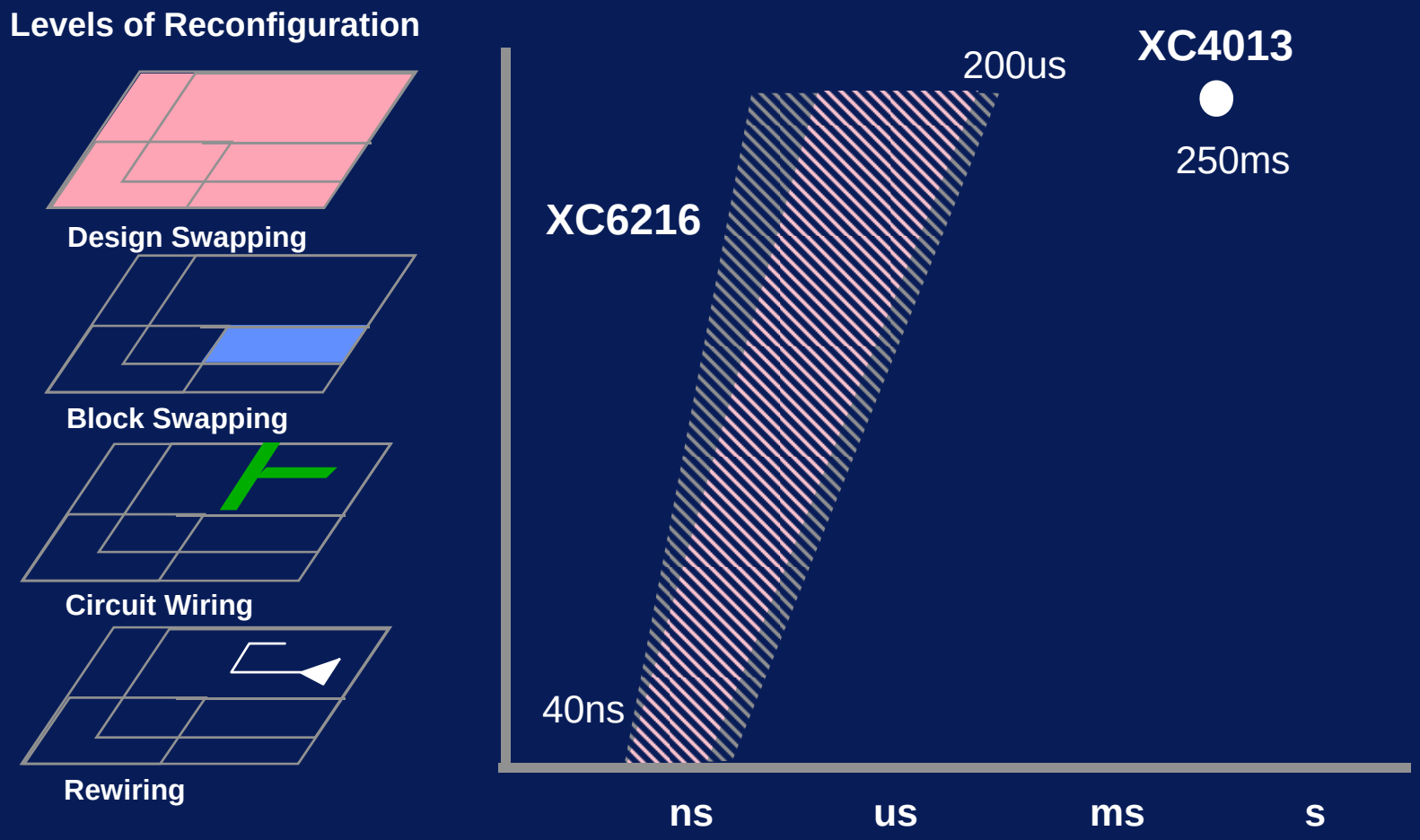
XC6200 System Features Meet Embedded Coprocessing Requirements



XC6200 Key Features



Reconfiguration Speed of Current Technologies



XC6200 Features & Benefits

Xilinx XC6200 Family

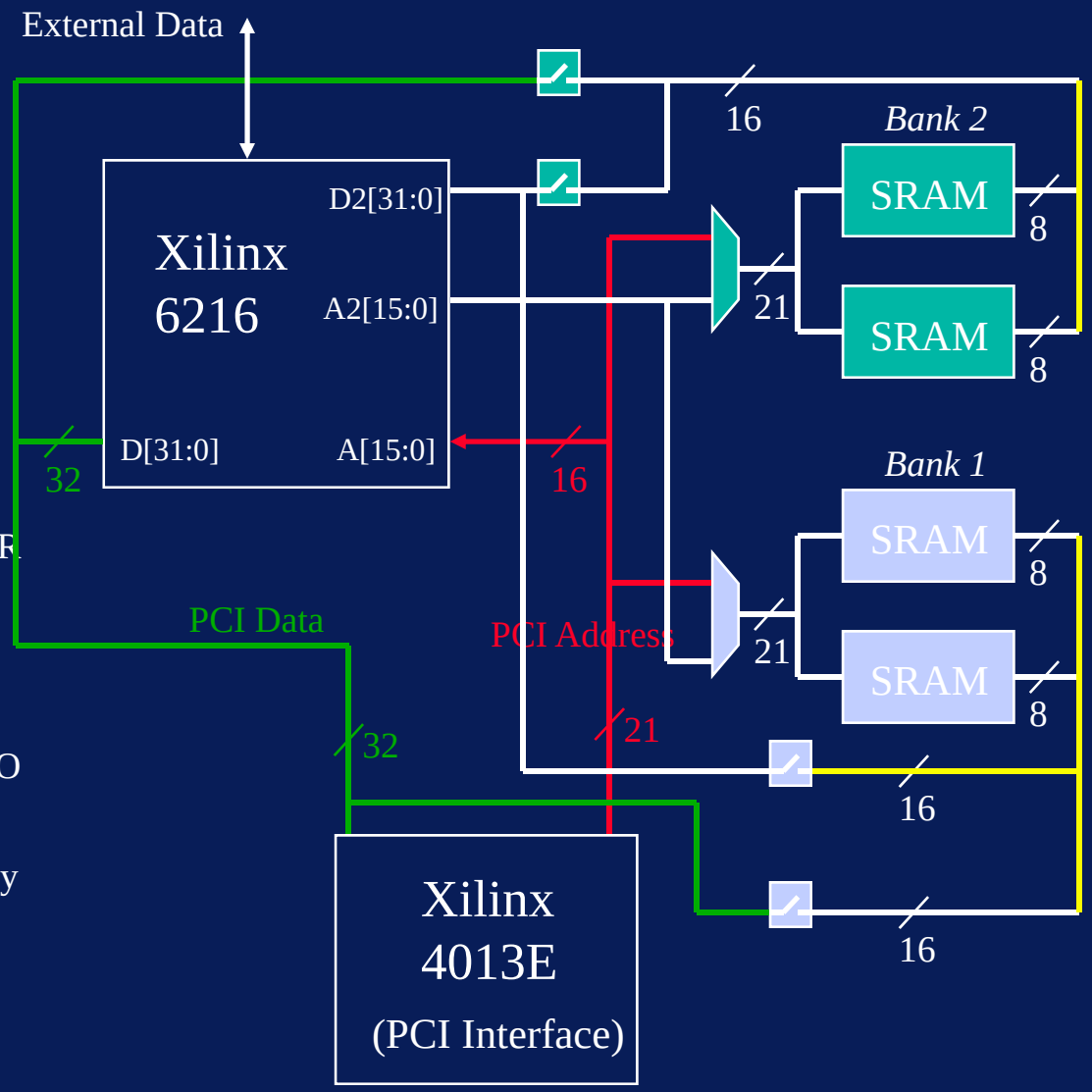
Changing the Rules of System Design

Features	*Benefits*
• Optimized processo (8, 16, or 32 bit)	• Simple hardware design, faster data transfer
• Ultra fast reconfigu	• 1000X faster context switching (logic and memory reconfiguration)
• Dynamic, partial ch	• Modify part of the design without processing interruption
• Efficient, symmetric	• Position independent design mapping
• open Architecture	• 3rd party tools, own tool development

PCI/6200 Board Architecture

Features

- Rapid prototyping system
- Automatically handles all architecture details (hides chip level interfacesl)
- Enables RL designs
- Sold through 3rd parties
  - Will include XC6200 P&R software
  - Can include other s/w modules
  - Headers for mezzanine I/O cards
- Xilinx will test for compatibility

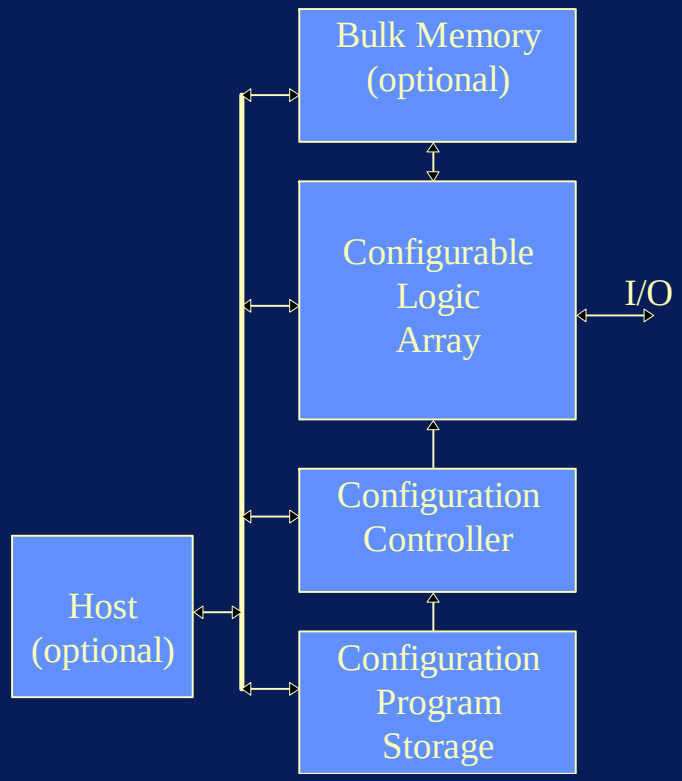


Using Reconfigurability

Ray Andraka,
the Andraka Consulting Group

Reconfigurable Systems

- SRAM FPGA for configurable logic
- Mechanism for loading configuration programs
- Storage for configuration programs
- I/O
- Optionally may include
  - Extra memory
  - Host processor

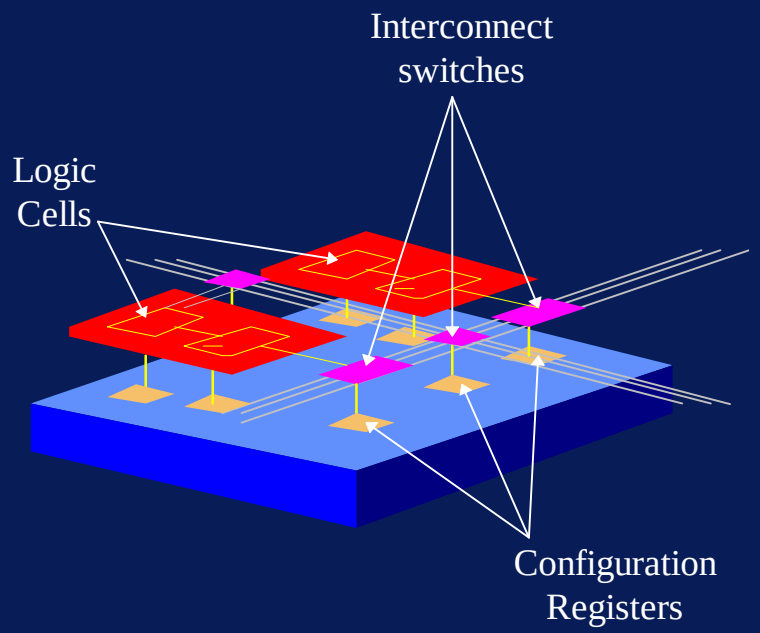


Configurable Logic

- Consists of one or more SRAM based FPGAs
- May also have programmable interconnect
  - Bus switches
  - Tri-state logic
  - Multiplexers
  - Crossbar switches
  - Field Programmable InterConnect (FPIC)

SRAM Based FPGA

- Infinitely reprogrammable
- Logic & connections set by underlying registers
- Rewriting configuration registers changes function
- Desirable features:
  - partial reconfiguration
  - reduced configuration time
  - state preservation
  - random config register access



Reconfiguration Mechanism

- Transfers bitstream from storage to configuration registers
- Sequences configuration controls on FPGA
- Needs to select among multiple configurations
- Should detect and recover from errors
- Triggered by FPGA, host, or external stimulus

Reconfiguration Controller

- Follow published sequence and timing exactly
- Host microprocessor may serve as controller
  - Local controller eases burden on host
- Simple controller built-in on most FPGAs
- Multiple configurations require external logic
- External controller often faster
  - Not part of reconfigured logic

Configuration Storage

- Bulk memory holds configurations
- May be stored on host processor
- Local storage for faster reconfiguration
  - Byte wide for easy paging
    - Multiple configurations require pointer
  - SRAM for security or frequent modification
    - must be initialized
  - EEPROM for non-volatility

Bulk Memory

- Often needed in DSP systems
- Data buffering
- Look-up tables
- Function Generation
- Size, type & speed application dependent

Configuration Classification

- Fixed function
  - product update
  - one-time customization
- Static configuration
  - Select configuration at system initialization
  - Multi-mode systems
- Dynamic configuration
  - Configuration changed as part of process

Dynamic Configuration Types

- Full or partial
  - Active logic during reconfiguration
- Compile-time or run-time
  - Configuration sequencing
- Rigid, adaptive or evolutionary
  - Configuration program alterations

Partial Reconfiguration

- Some of logic remains active
- Transients caused by reconfiguration
  - Ignore inputs from reconfiguring logic
  - Force newly reconfigured logic to known state
- Special consideration to preserve state of flops
- Signal contention a possibility
- Full reconfiguration avoids issues
- Partial can be chip or system level

Run-Time Reconfiguration

- Sequence of configurations affected by process
  - Compile-time uses fixed sequence
- Pointer to next configuration program
- Controller must have hooks to obtain pointer

Adaptive Configuration

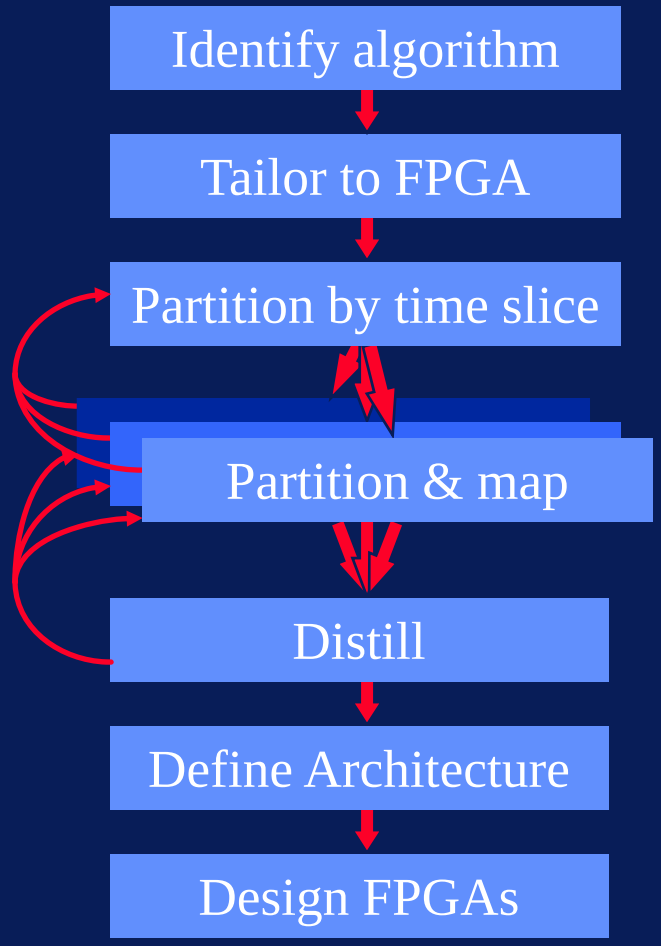
- Modifies explicit parameters in next configuration
  - Normally to change coefficients
  - Logic alterations if no routing changes
- Requires knowledge of bitstream format
  - Find by differences in bitstreams
  - Formats from vendor in some cases
- May require recomputing error check codes
- Potential for bitstream corruption

Evolutionary Configuration

- Next bitstream generated by genetic evolution
- Requires intimate knowledge of bitstream format
- Rules required for mutation
- Extensive bounds checking required
  - Illegal configurations
  - Signal contention
  - Floating inputs
- Not all devices suitable

Design Process

- Partition algorithm into time slices and devices
- Identify resource for each configuration
  - FPGA
  - Memory
  - Interconnect
- Derive minimum common architecture
- Design configurations to architecture

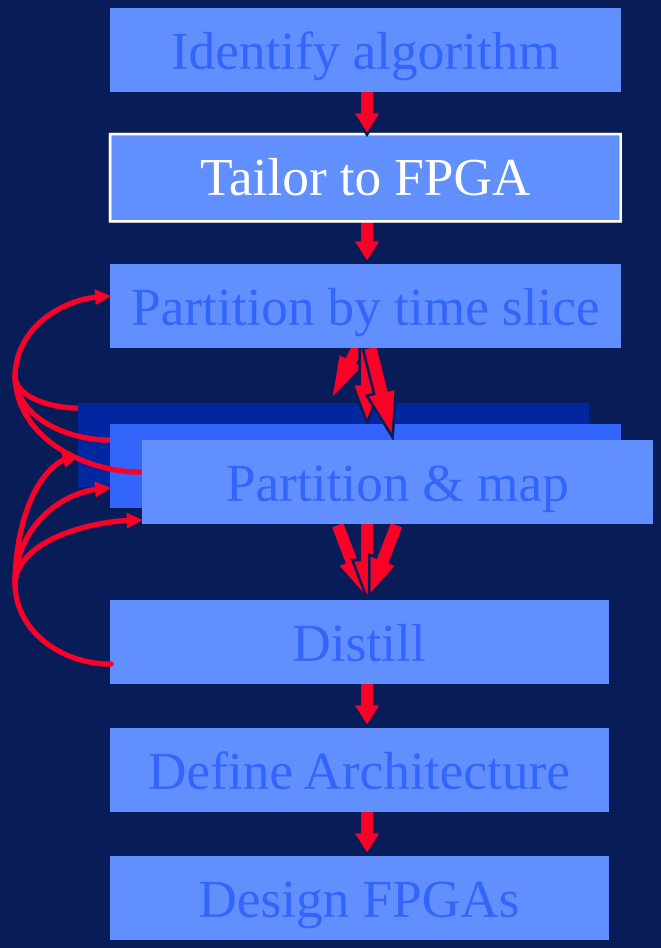


Identify Algorithm

- Develop block diagrams
- Identify parameters
  - data rates
  - precision
  - data flow
  - functions
  - storage

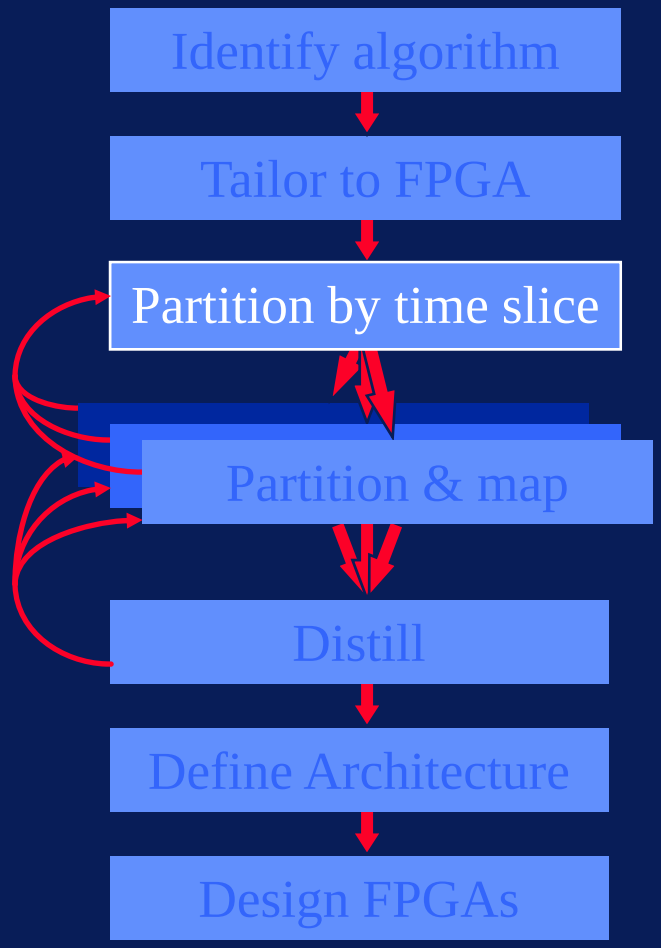
Tailor to FPGA

- Eliminate multipliers where possible
- Distributed arithmetic
- Bit / radix serial arithmetic
- Alternate approaches or approximations
- One-hot state machines
- High performance design



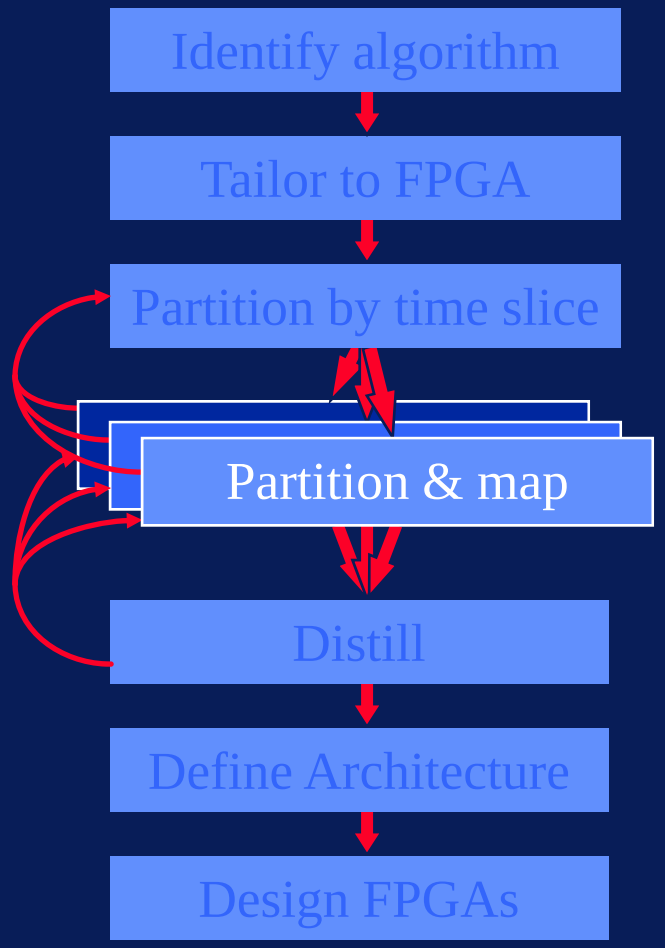
Partition by Time Slice

- Account for reconfiguration time
- Preserve data during reconfiguration
- Examine all configuration combinations for partial
- May not be practical



Partition & Mapping

- FPGA family, size, quantity, and speed
- Memory size, type, speed and location
- Interconnect
- Group common functions
- Partial configuration requires greater detail

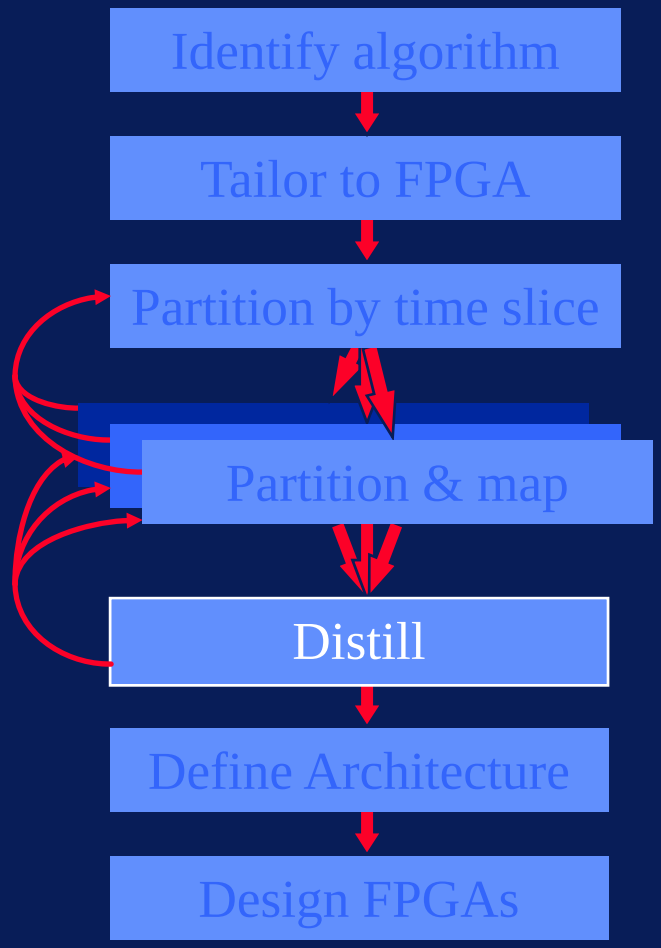


Partitioning & Mapping Tips

- DSP functions found in app-notes
- Don’t overpack FPGA
  - Leave room for floorplanning, routing and changes
- Consider pin counts
- Group logic common to several configurations

Distill to Common Architecture

- Architecture
  - FPGA type and quantity
  - Interconnect pattern
  - Memory
- Re-map (or modify) configurations
- Exploit similarities
  - function
  - memory location, size
  - interconnect
- Iterative process

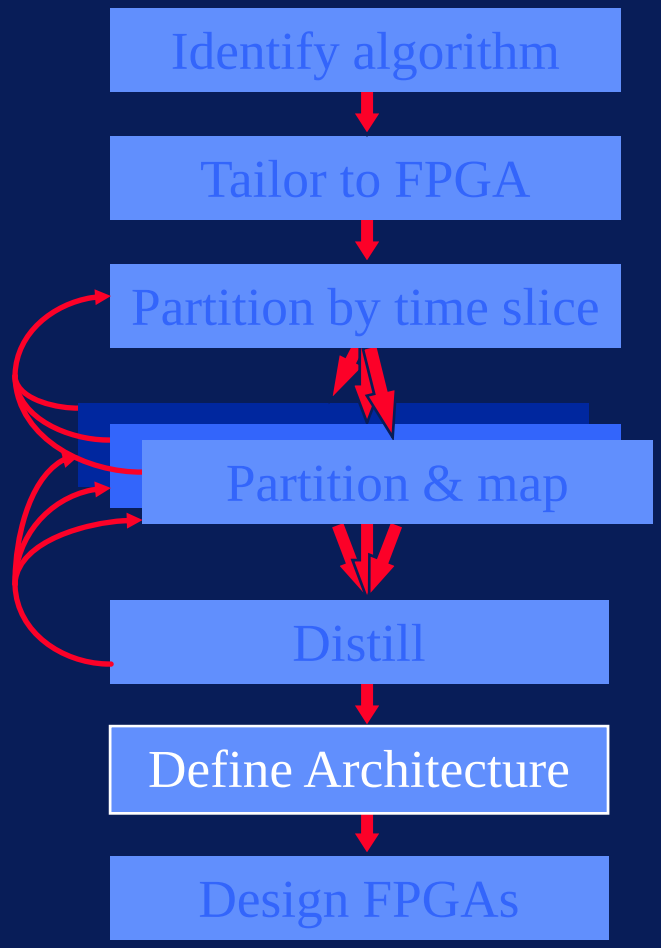


Distillation Tips

- Bus switches or FPIC for interconnect conflicts
- Seek symmetry in the design
  - helps for new applications
  - same memory connections for each FPGA
  - symmetric interconnect
- Use extra bulk memory for function generation
- Daughtercard for special I/O circuits
- Set parameters through reconfiguration

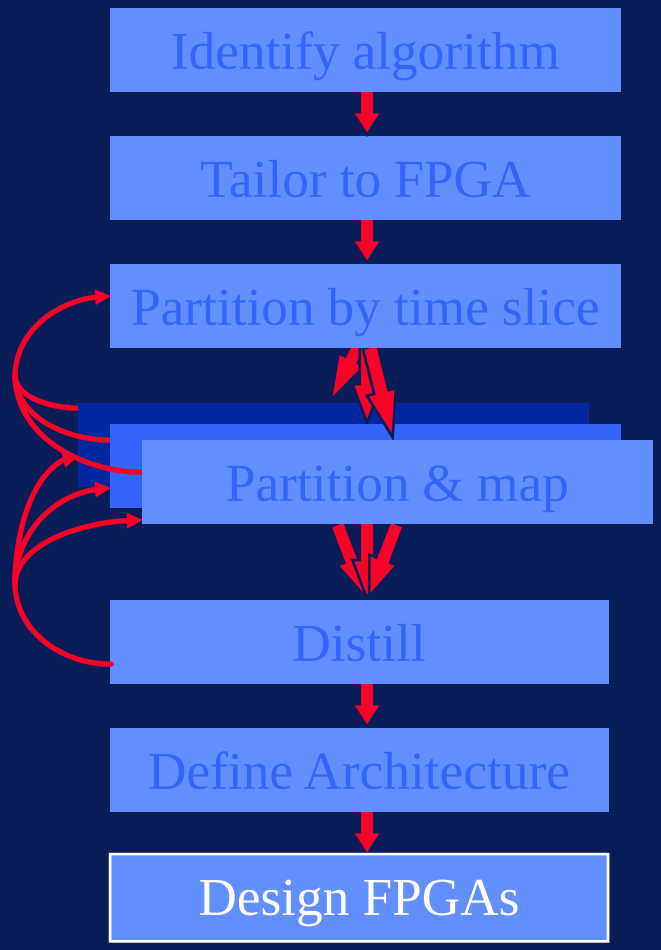
Define Architecture

- FPGA type, size, quantity
- Interconnect scheme
- Memory size & distribution
- I/O and host interfaces
- Support circuitry
  - config controller
  - clocks and resets
- Generalize architecture



Design FPGA Logic

- High performance techniques
- Pin-locked
- Partial reconfiguration requires floorplanning



High Performance Design Techniques

- Tailor design to device
- Handcraft where necessary
  - performance critical logic
  - density critical areas
- Floorplan design
- Set timing constraints
- Avoid HDLs

High Performance Design Techniques

- Use Level Compression
- Pipeline design
- Duplicate logic
  - reduce fan outs
  - eliminate extra interconnect
- Minimize internal tri-state use

Configuration Controller

- Use published configuration sequence & timing
- Need to select among multiple programs
- Detect & recover from errors
- Allow for remote configuration
- Daisy-chain not always practical
- Dedicated controller & storage preferred
- Isolate configuration path for partial

Configuration Storage

- Size depends on number of configurations
- Byte wide memory easier for multiple programs
  - uses more FPGA pins
- SRAM or EEPROM
  - adaptive and evolutionary need SRAM
  - Design security may demand SRAM
  - SRAM needs to be initialized

Pin Assignment

- Must be consistent across all configurations
- Be aware of dual function pins
- Follow data path flows
  - busses on same sides of chip
- Correlate pins to row and column on FPGA

Debug

- Hooks to monitor configuration (via host)
- Design in host/debug port access to all FPGAs
- Access to clock, special signals
- Use reconfiguration in debug
- Include remote configuration capability

Clocking

- Number and frequencies of clocks
- 2x data rate clock handy
- Bit serial clocks at 100MHz and beyond
- Consider PLL
  - Frequency synthesis
  - External clock duplication
- Clock skew between FPGAs
- Signal quality

Power Dissipation

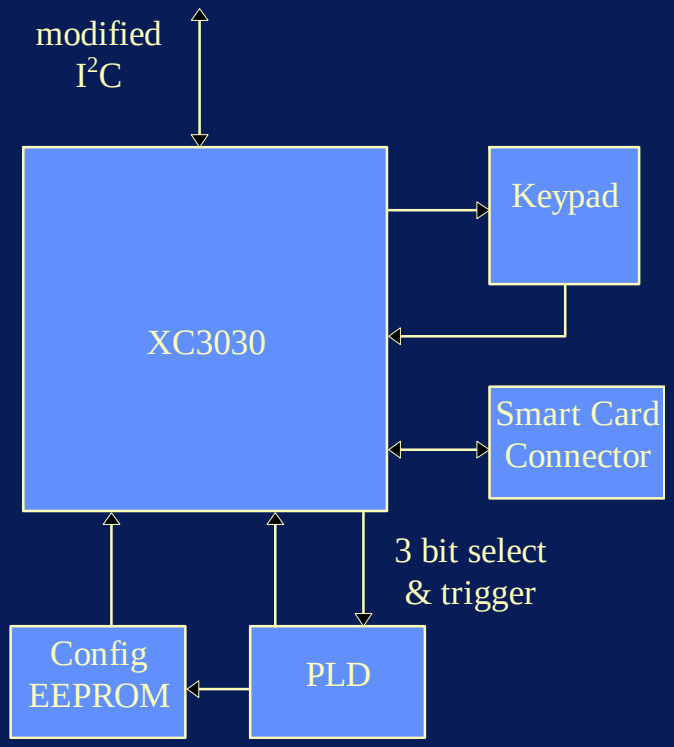
- > 50 % of nodes switching in pipelined DSP
- High clock frequencies
- Device dissipation can easily exceed 5W
- Problem worse in bit serial designs
- Thermal management is a must

Reconfiguration for DSP

- High data rates limit reconfiguration opportunity
  - Relatively slow configuration expensive
  - Improved by partial reconfiguration
  - improved by reduced configuration times
- Typically a reconfigurable logic coprocessor
  - Load custom process unique to application
  - Reload with new coprocessor when done
- Semi-static or full reconfiguration
- Reconfiguration requires breaks in datastream

Example:
Universal Smartcard Controller

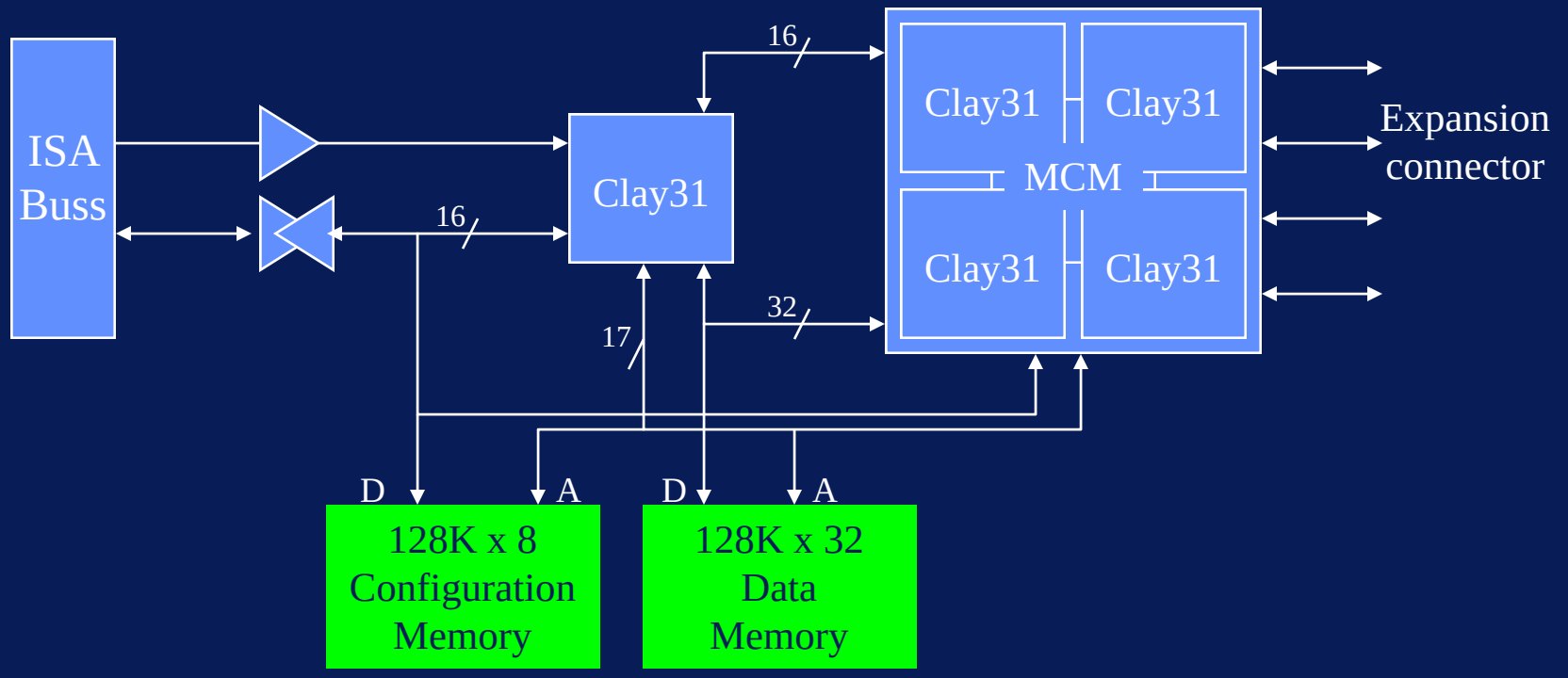
- Run-time, full, rigid configuration
- Start with detect logic
- Detect Smartcard
- Reconfigure with optimal interface
- Access card
- Reconfigure with detect logic upon withdrawal
- Keypad scan in all configurations



Example:
Video Processing Platform

- Dynamic compile-time partial configuration
  - partial configuration reduces overhead
- Use an existing platform plus daughtercard
  - Minimal hardware on daughtercard
- Reconfigure during frame retrace time
- Common functions remain
- Overlays replaced to change process

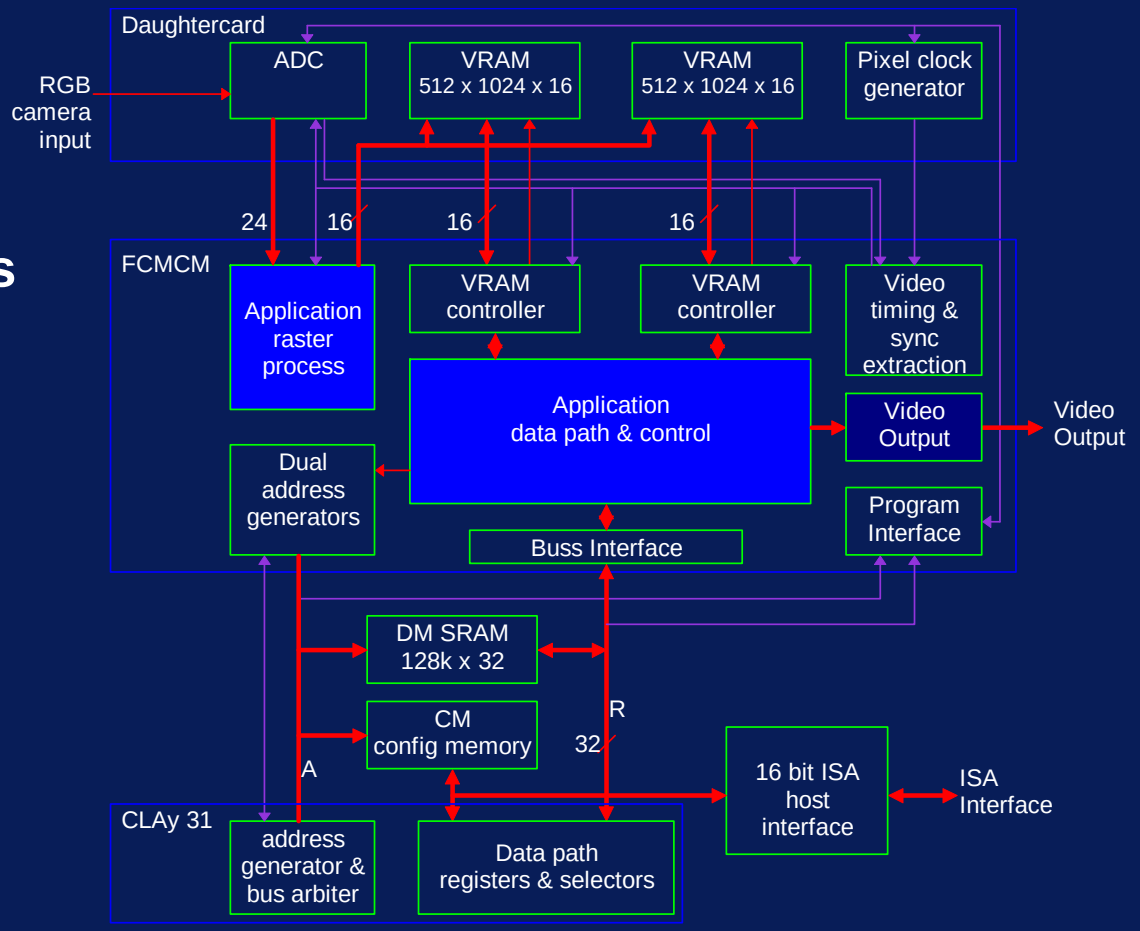
FCMfun Board Architecture



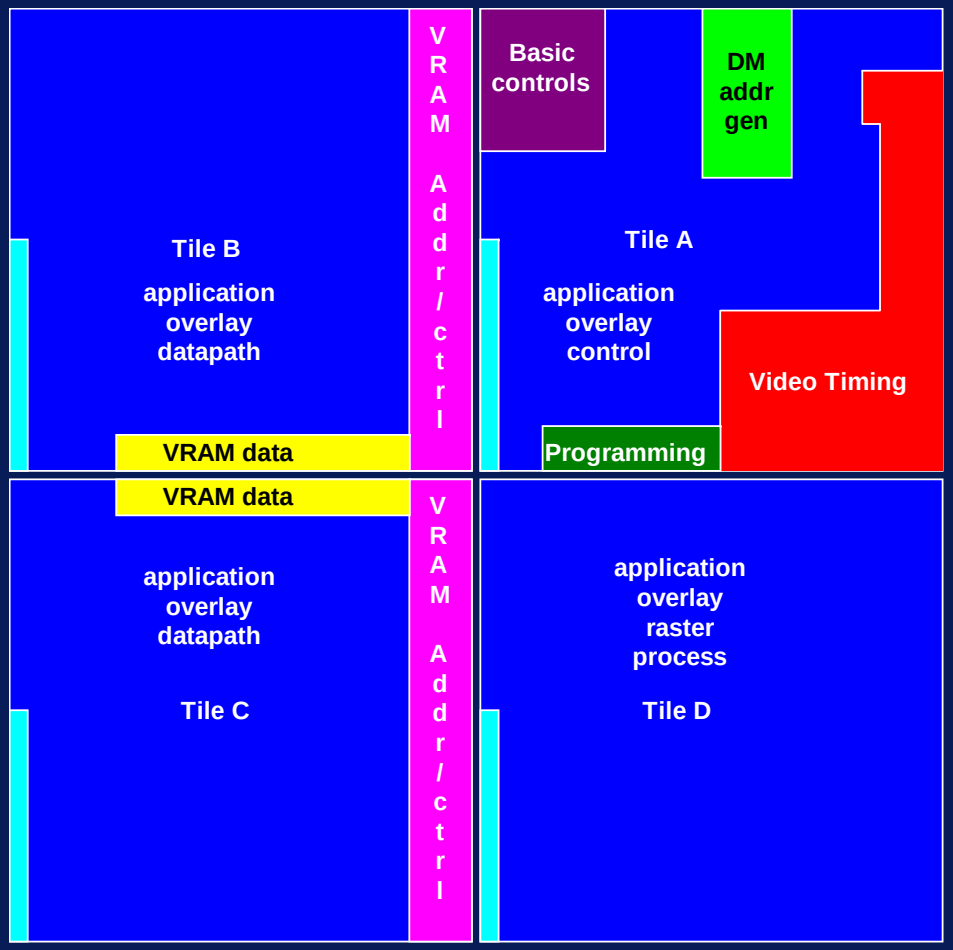
Courtesy of National Semiconductor Corp.

Dynamic Hardware Video System

- FCMCM dynamic
  - Video timing
  - Memory controllers
  - Process overlay
- CLAy31 static
  - ISA interface
  - Config controller
- Reconfigure FCMCM during frame retrace



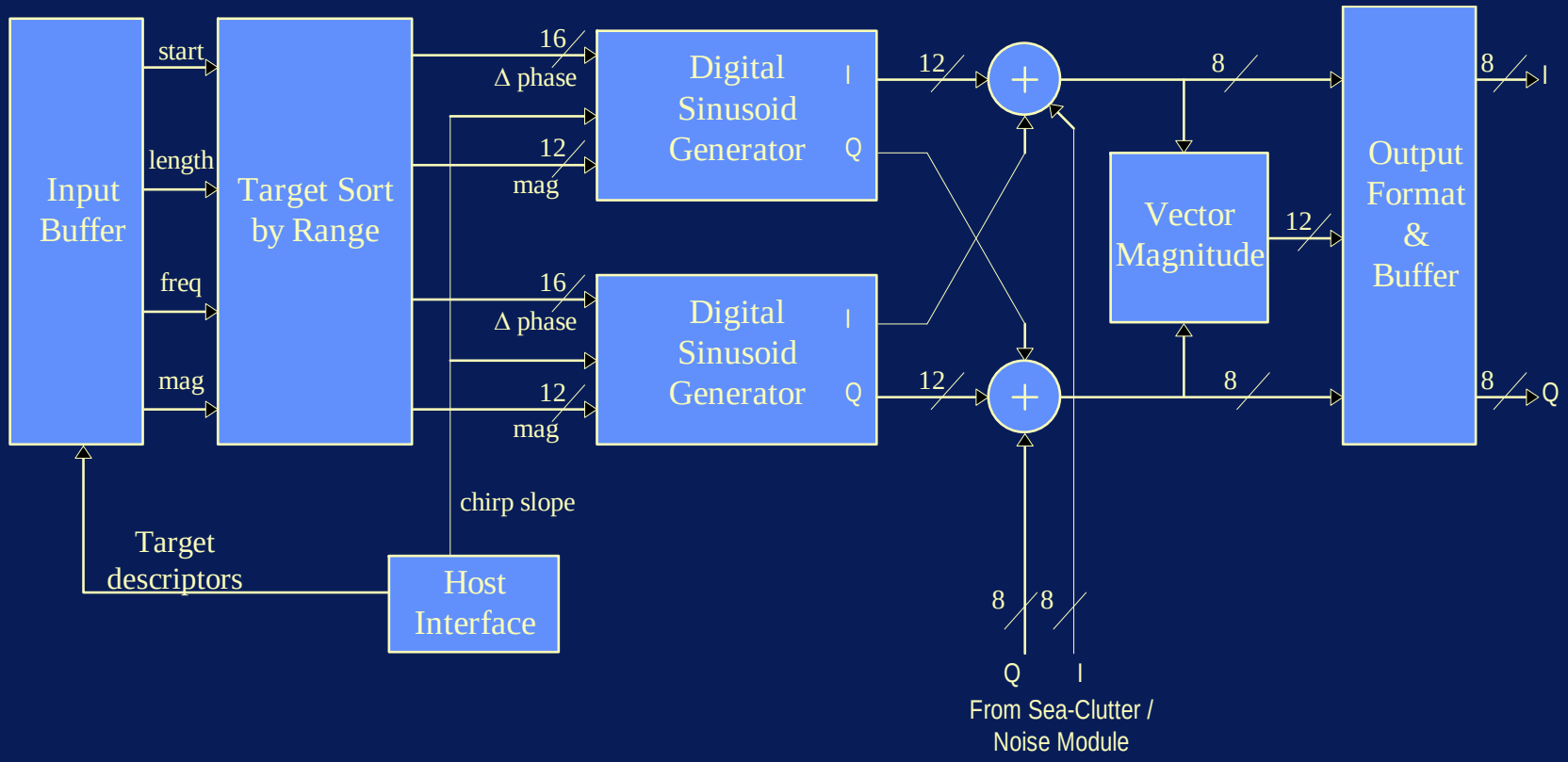
FCM Floorplan



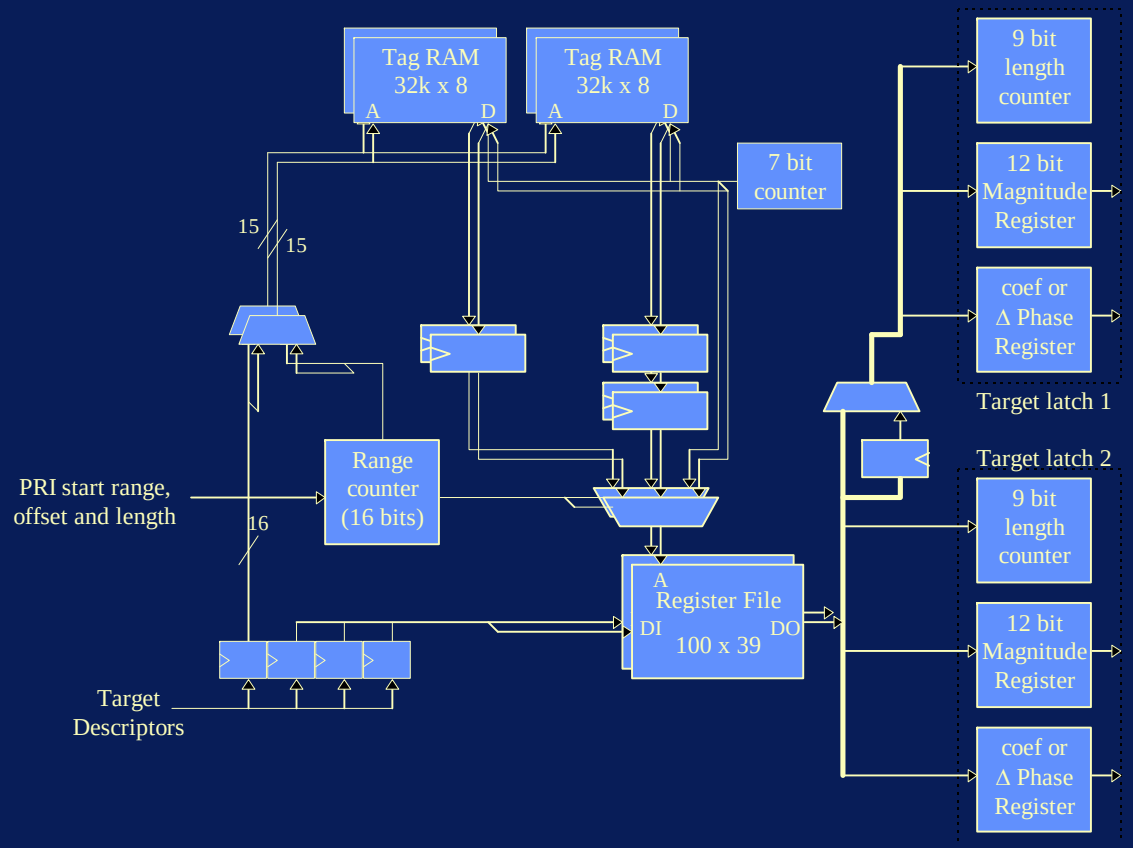
Example:
Radar Target Generator

- Full compile-time rigid configuration
- Platform has 4 required configurations:
  - Chirp/CW target generator
  - Impulse target generator
  - Sea Clutter generator
  - Transponder (IFF) response generator
  - Tables loaded by extra configurations
- Future applications expected
- 40 MHz data

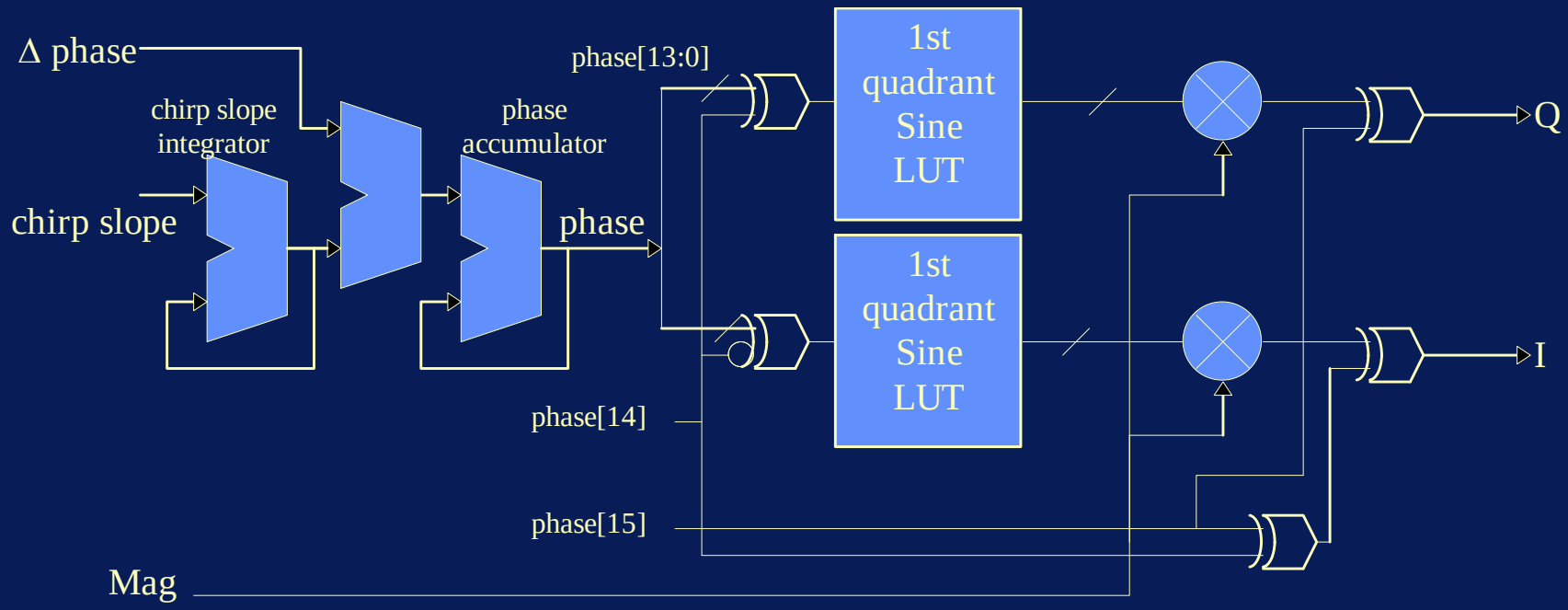
Chirp Target Generator



Tailored Target Sort

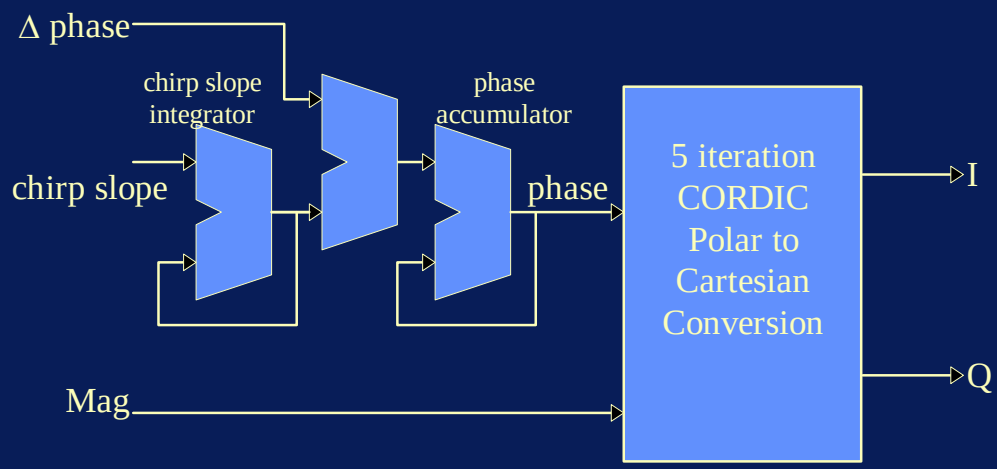


Sinusoid Generator

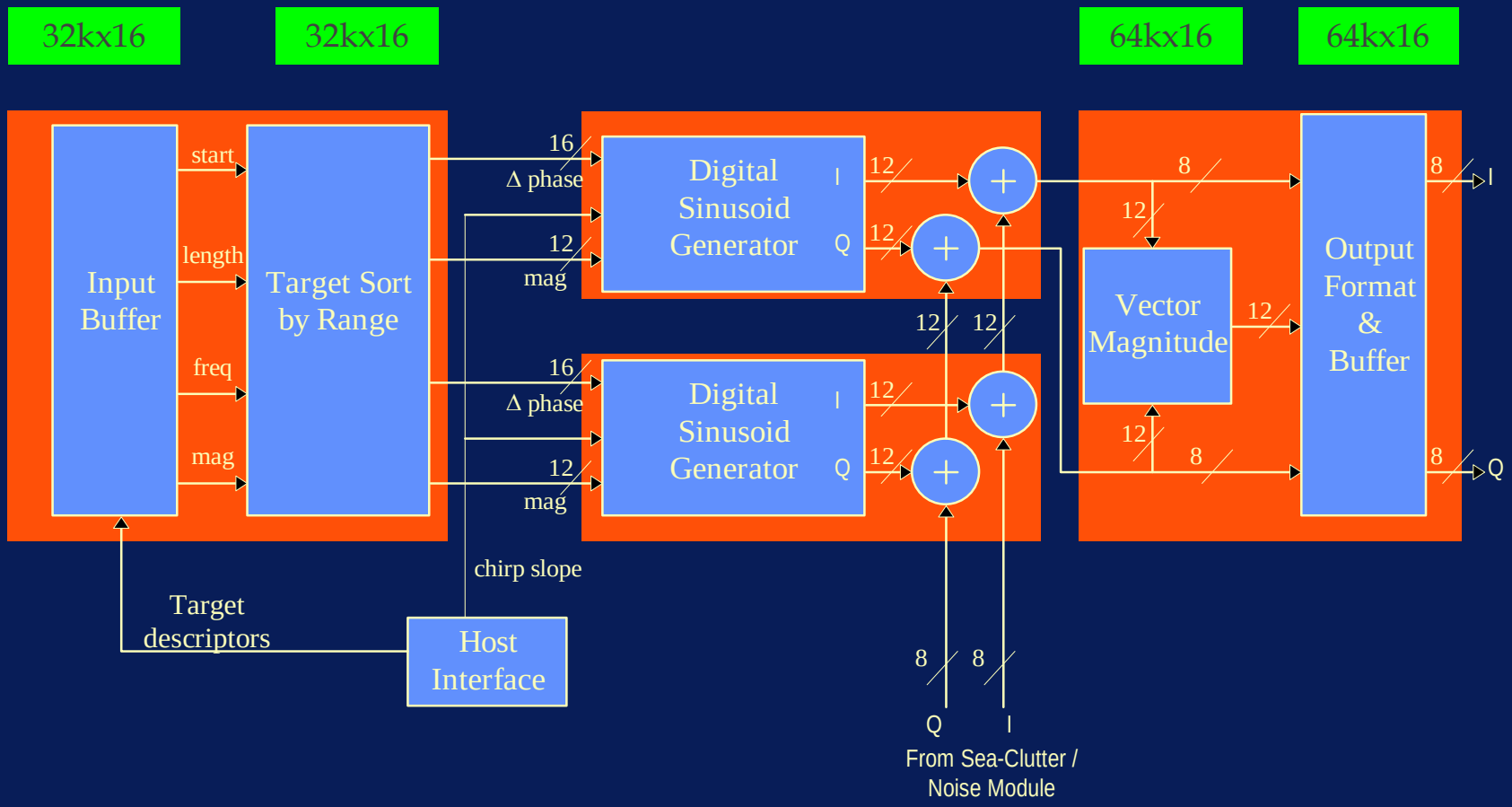


Tailored Sinusoid Generator

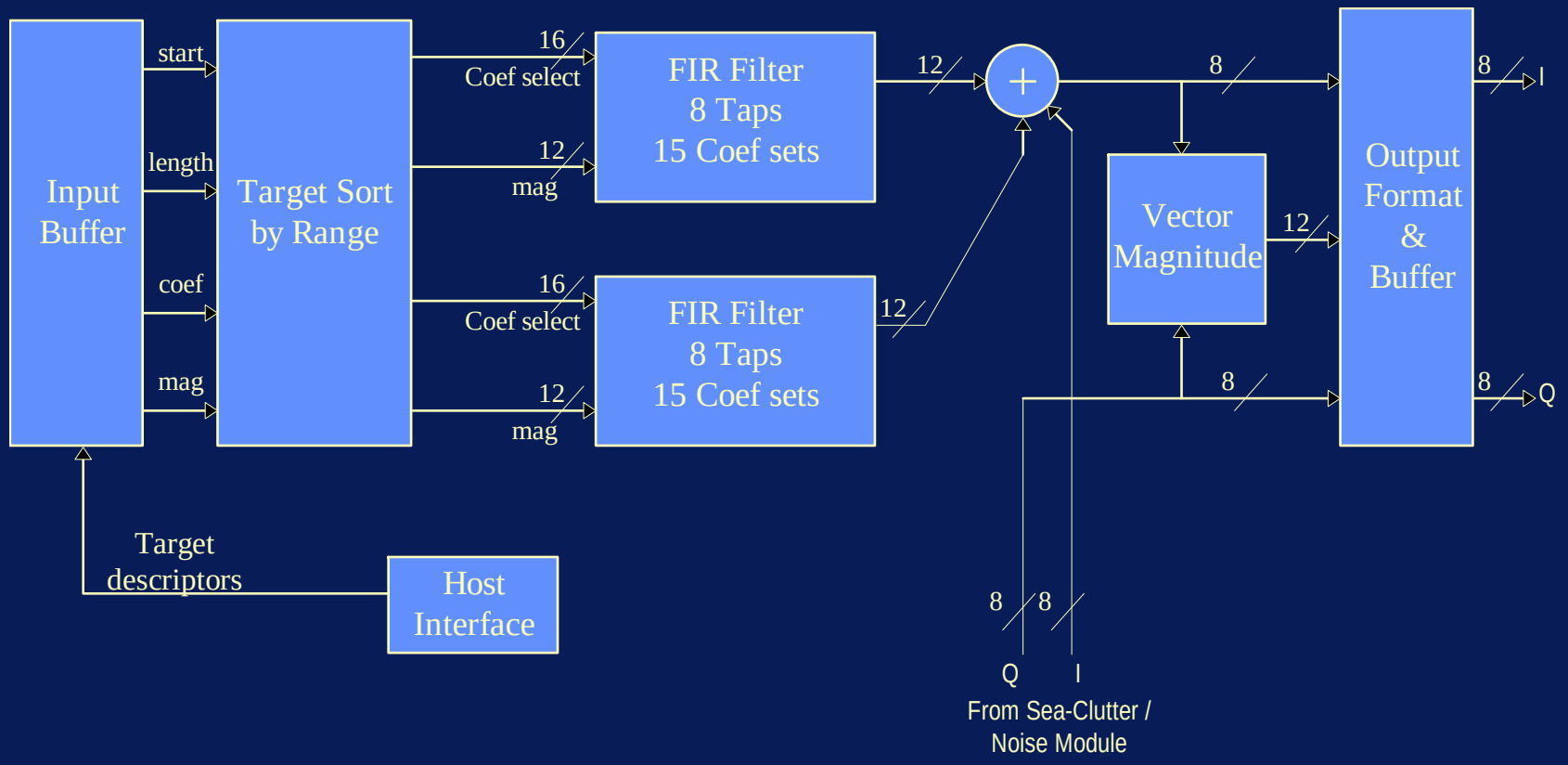
- **Smaller than multiplier**
- **20% of XC4013E**
- **50+ MHz data**



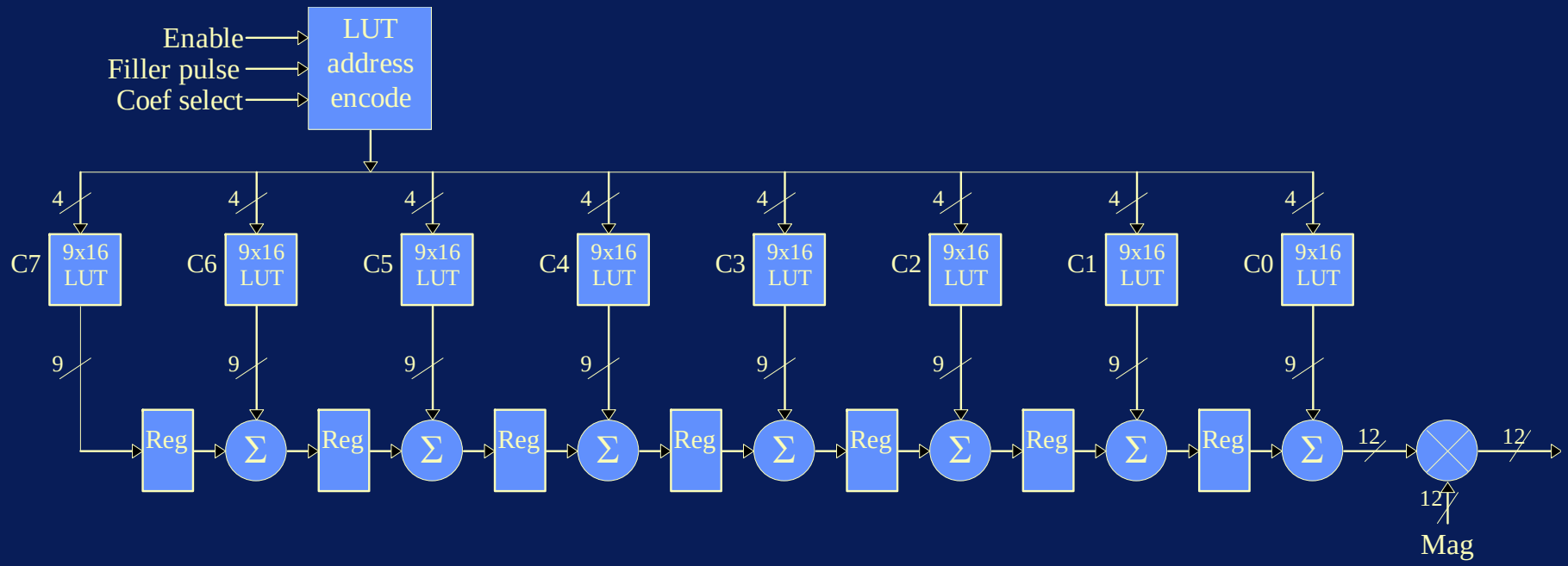
Chirp Target Partitioning



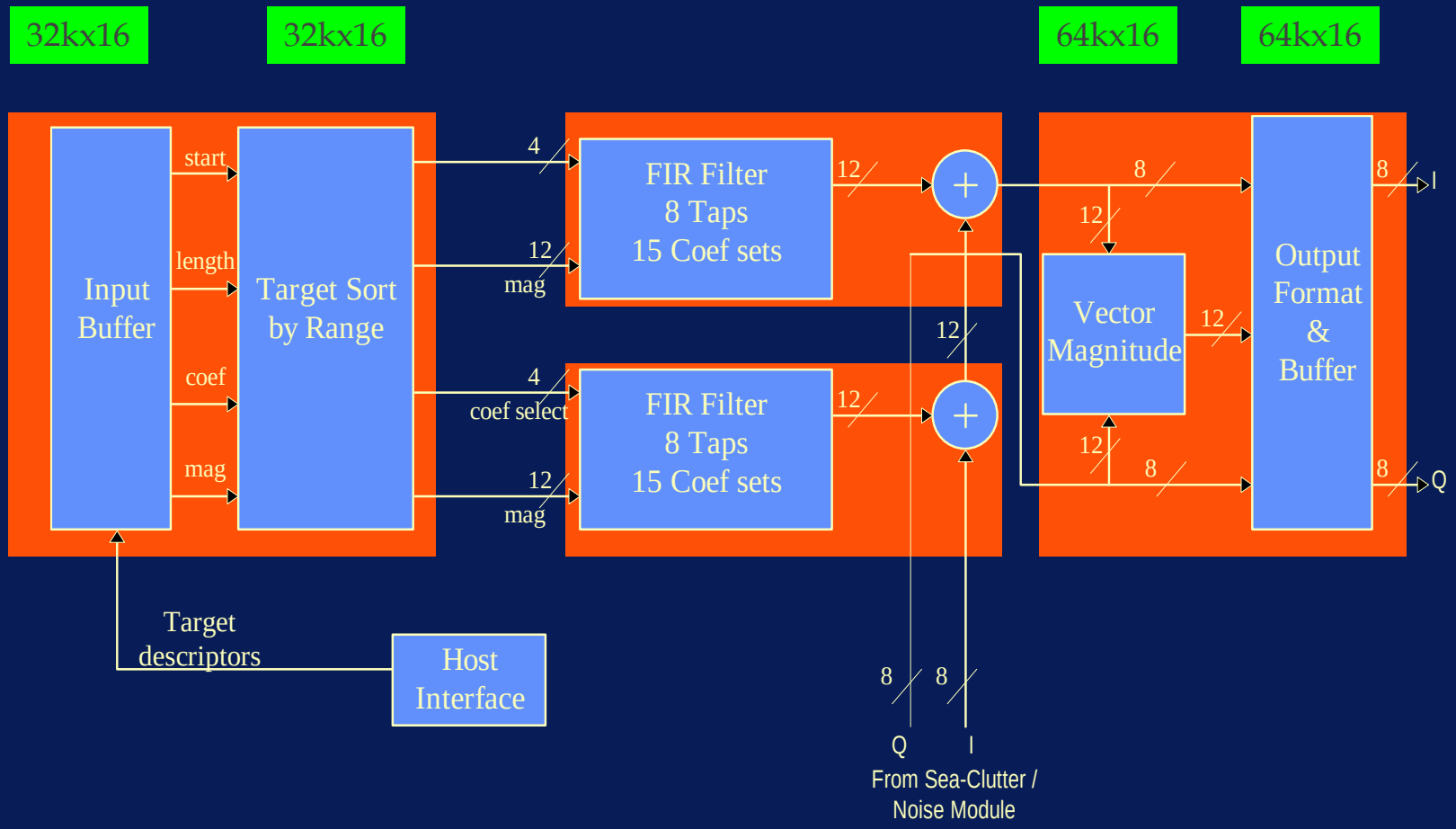
Impulse Target Generator



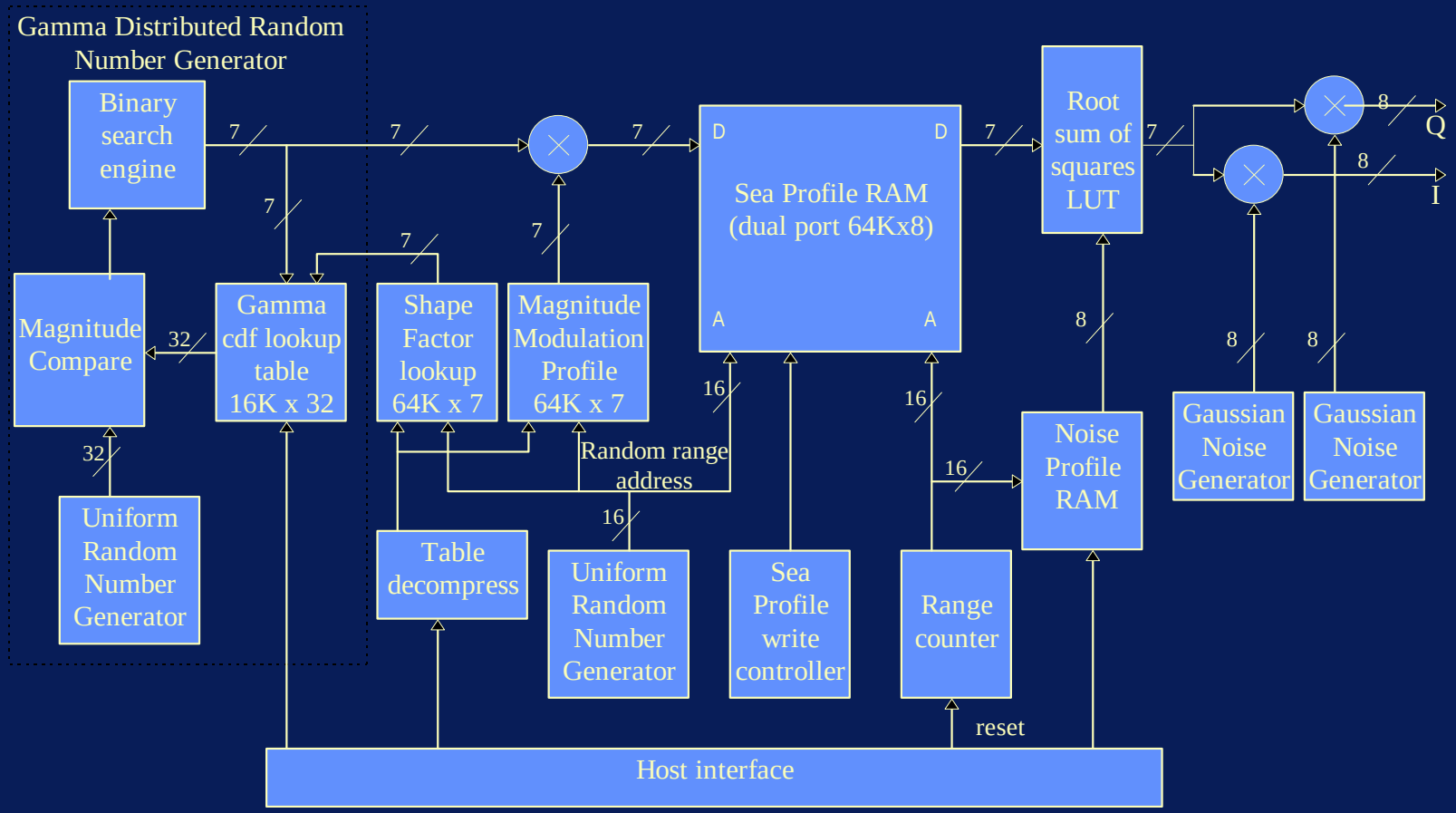
Tailored FIR Filter



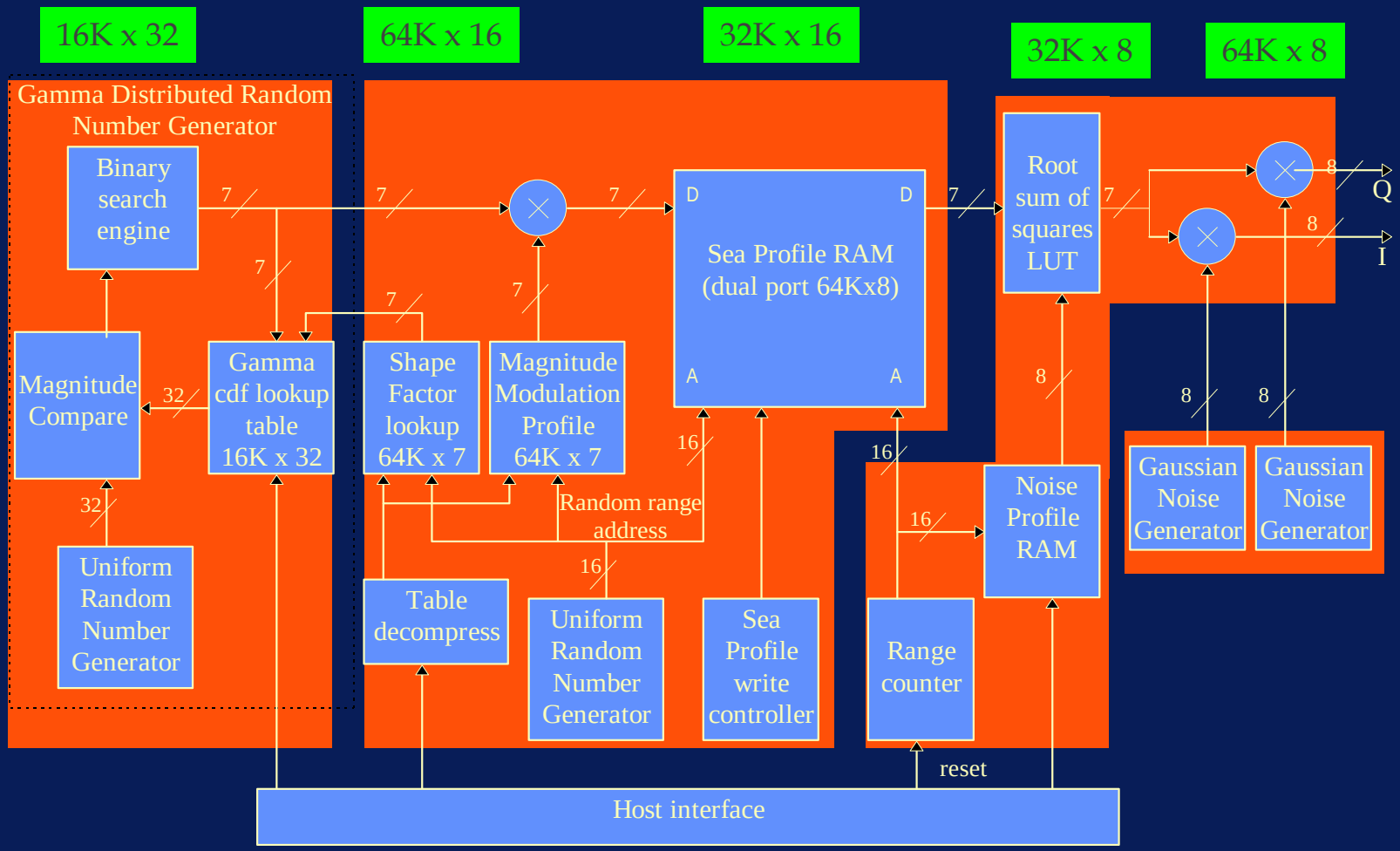
Impulse Target Partitioning



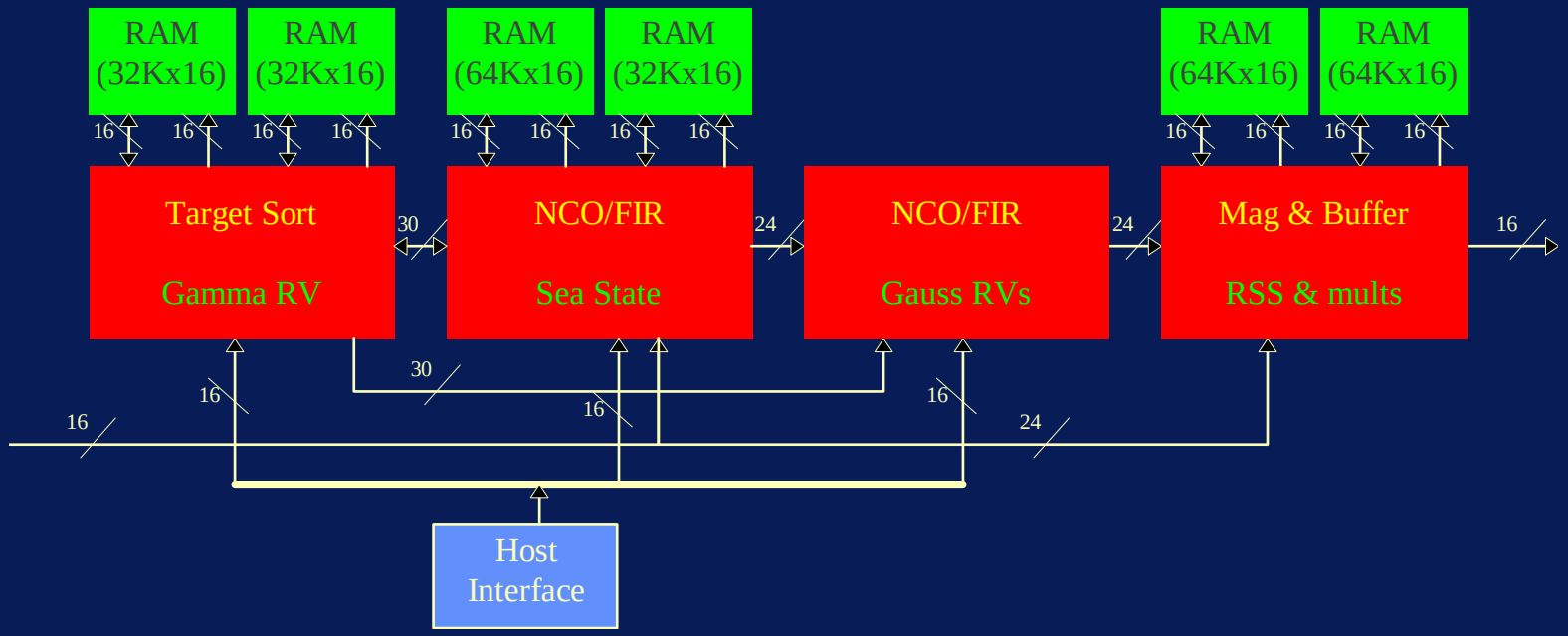
Sea Clutter Generator



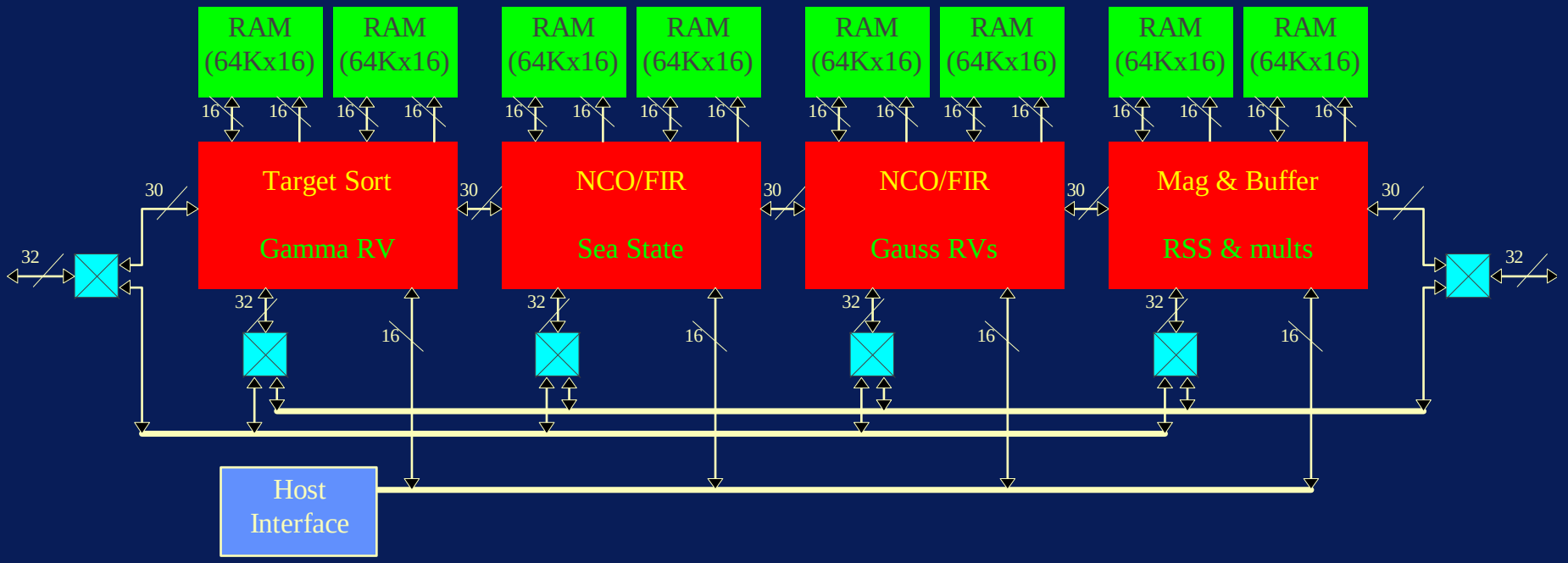
Sea Clutter Partitioning



Minimum Common Architecture



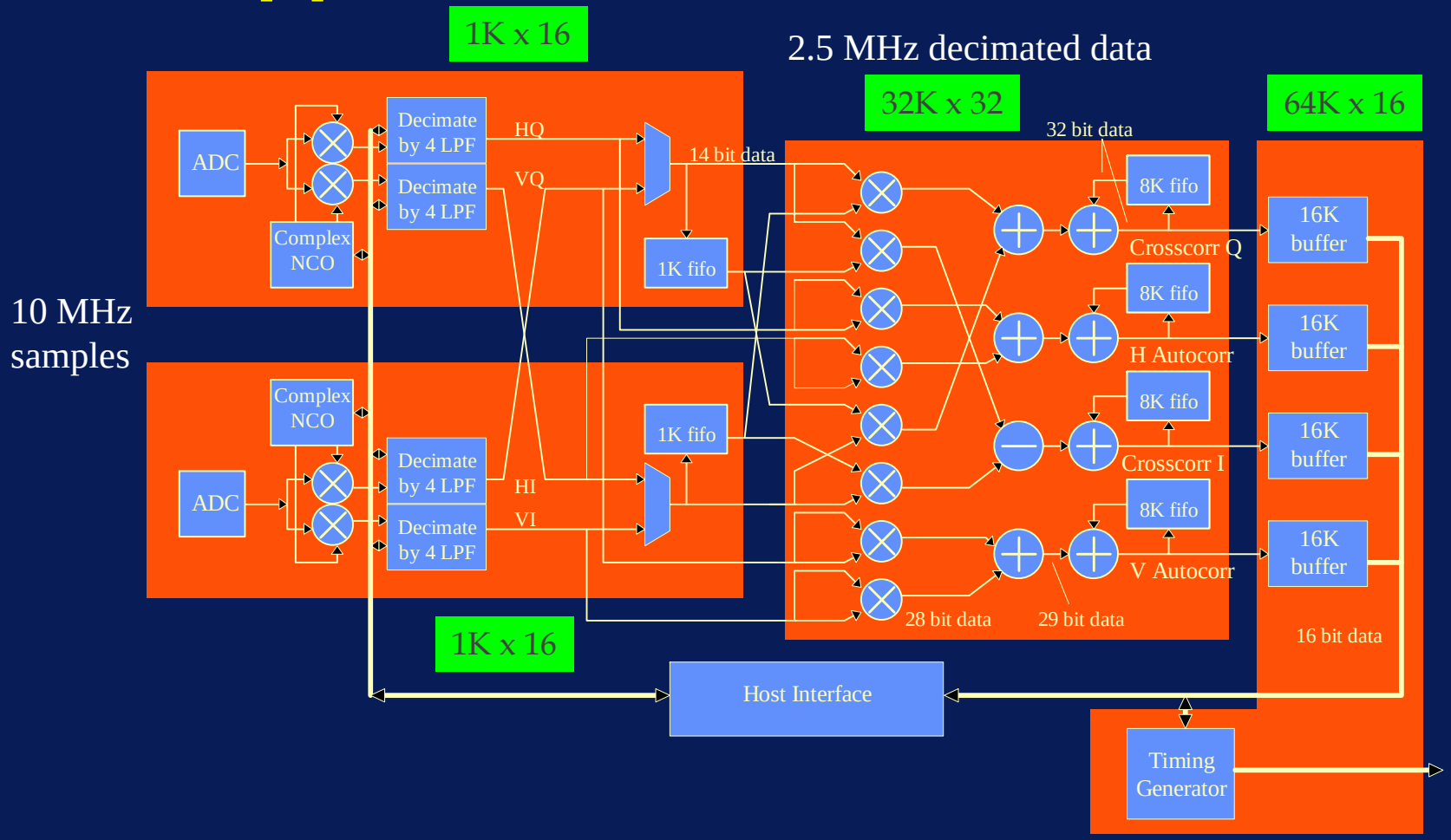
Enhanced Architecture



Example:
Doppler Radar Processor

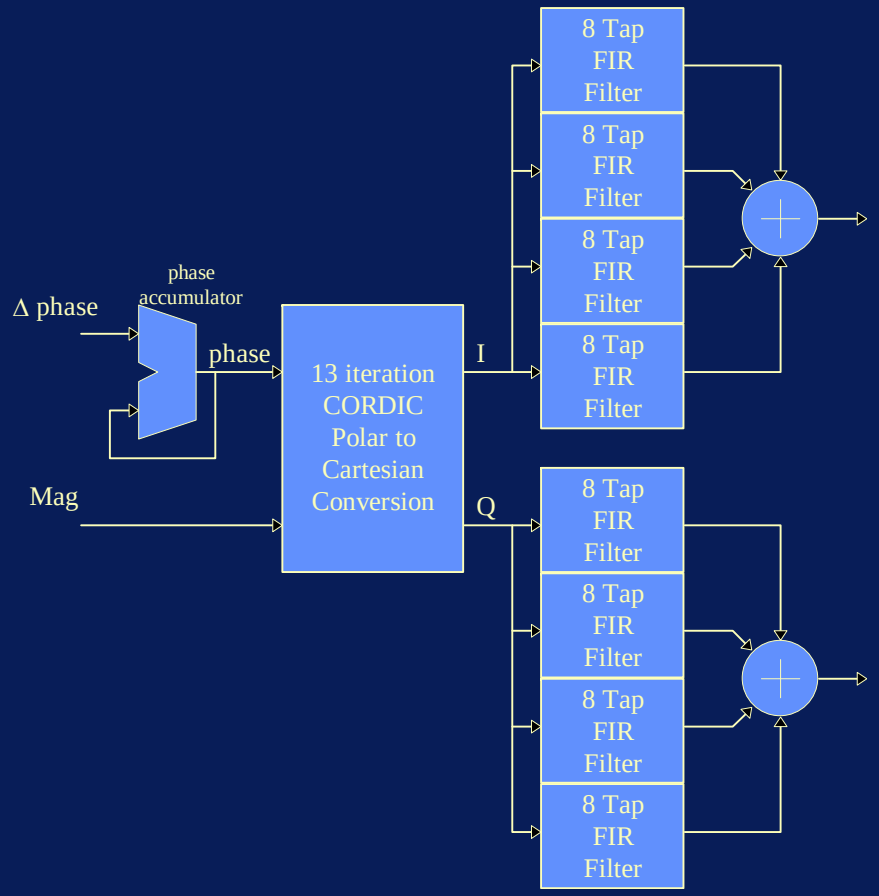
- Use existing platform architecture
- 2.5 MHz decimated data rate
- Daughtercard for special I/O (ADC's)
- Reconfigure for:
  - Receiver response (filter) changes
  - Mode changes (pulse position, pairing, calibration)

Doppler Pulse Pair Processor

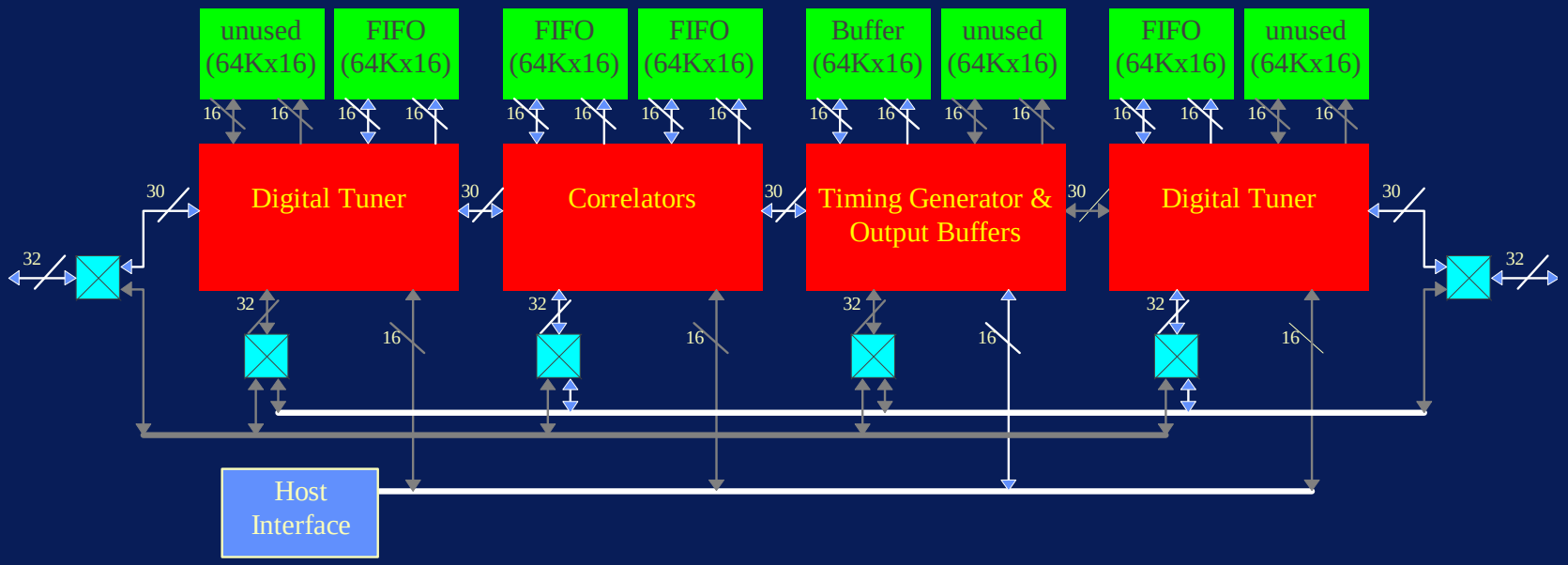


Digital Tuner in FPGA

- 20 MHz input
- 5 MHz output
- Fits XC4020E
- 14 bits out
- 70 MHz bit clock
- Serial Distributed Arithmetic filters
- Bit parallel CORDIC



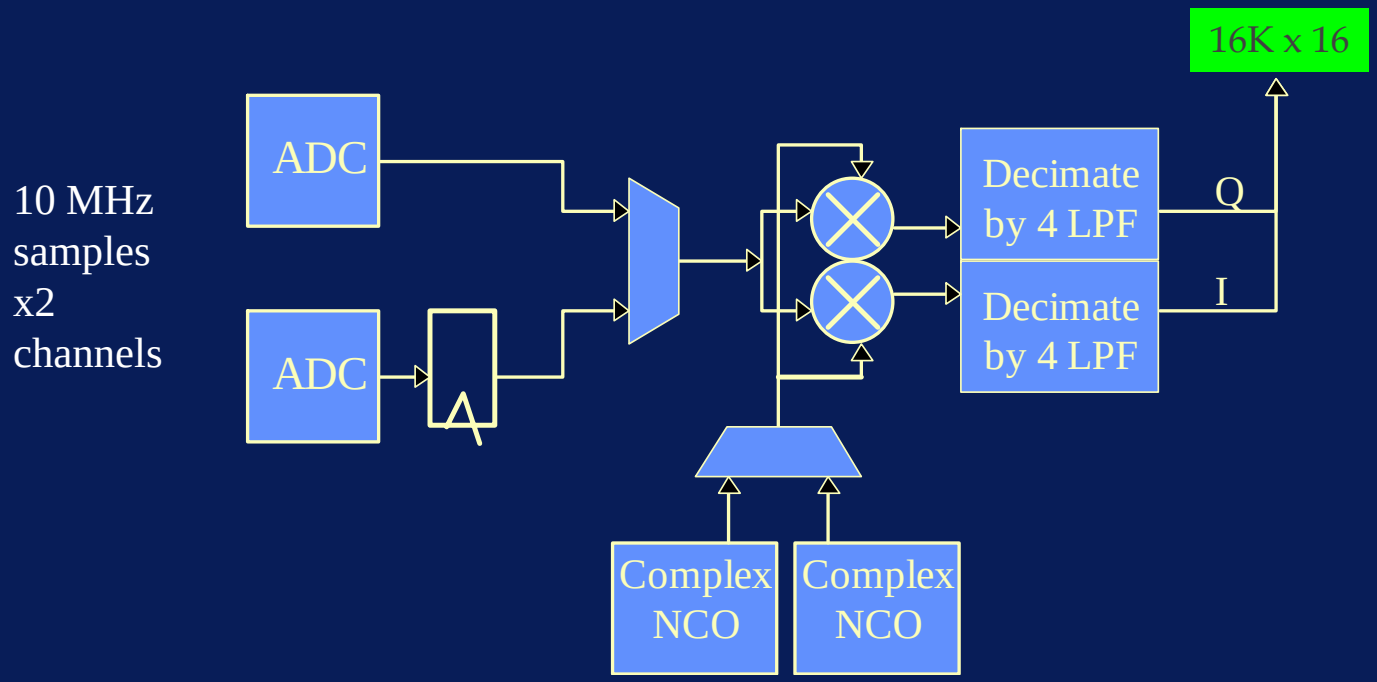
Doppler Pulse Pair Mapping



Another Solution: Temporal multiplexing

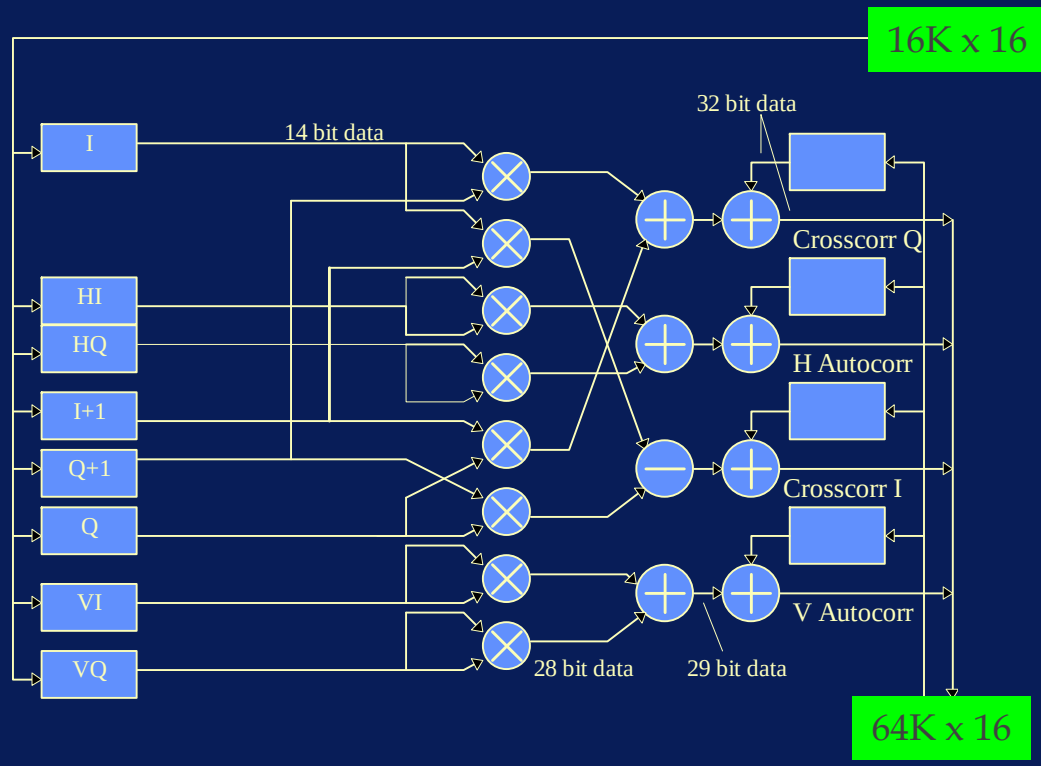
- Single chip time multiplexed
- 100 ms between pulse groups
- 42 ms reconfiguration time
- 1st configuration is demodulator
  - collect 5 pulses (1K samples each)
- 2nd configuration is correlator & averaging
  - process 5 pulses & repeat for next group
- 3rd configuration for data read

Digital Demodulator



Correlator Time Slice

Parallel to Serial Shift registers



Summary

- Performance of dedicated hardware
  - 10x to 1000x faster then Microprocessor
- Flexibility of Microprocessor
  - Process changed by reconfiguring
- Cost on par with DSP microprocessor systems
- Design flow similar to typical FPGA
  - Partition for time slices
  - Common architecture distillation or fitting
  - pin locked across all configurations

FPGAs are more than just PALs