

# FPGA Architecture

**Mark McDermott**

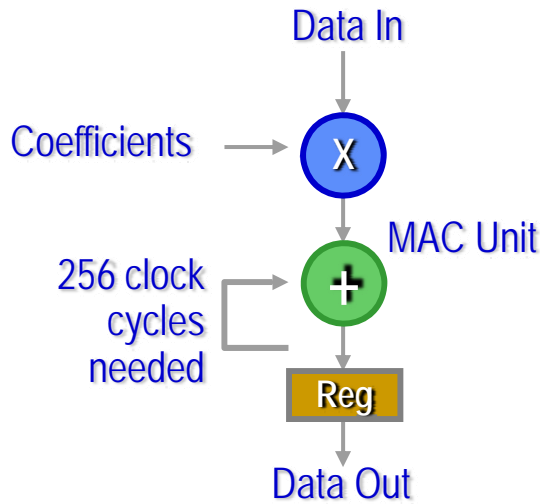
# Agenda

- **Applications of FPGAs**
- **Overview**
- **Basic Architecture of an FPGA**
  - Slice Resources
  - Memory
  - Clocking
  - I/O Resources
  - Programming Configuration
- **Self Reconfiguring FPGAs**

# FPGAs Enable Massively Parallel DSP

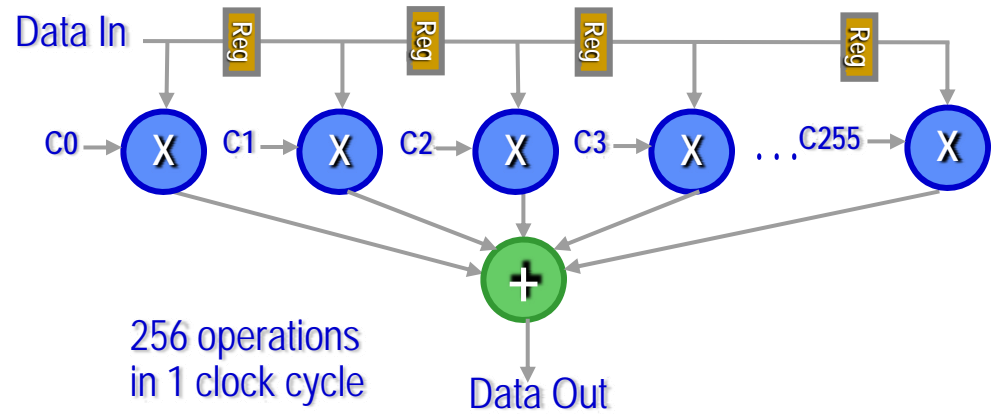
## Example 256 TAP Filter Implementation

Programmable DSP - Sequential



$$\frac{1 \text{ GHz}}{256 \text{ clock cycles}} = 4 \text{ MSPS}$$

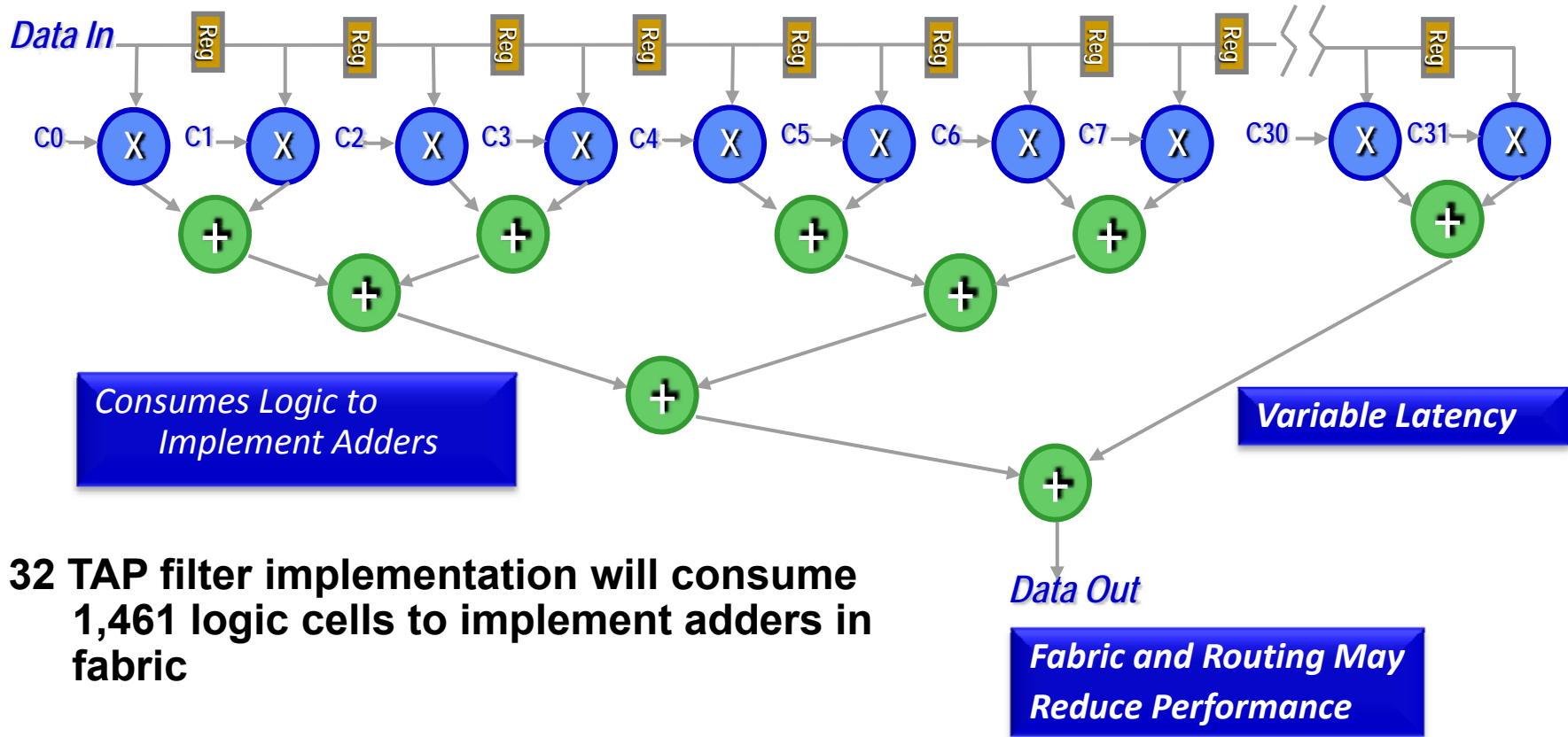
FPGA - Fully Parallel Implementation



$$\frac{500 \text{ MHz}}{1 \text{ clock cycle}} = 500 \text{ MSPS}$$

# Usual Parallel Adder Tree Implementation

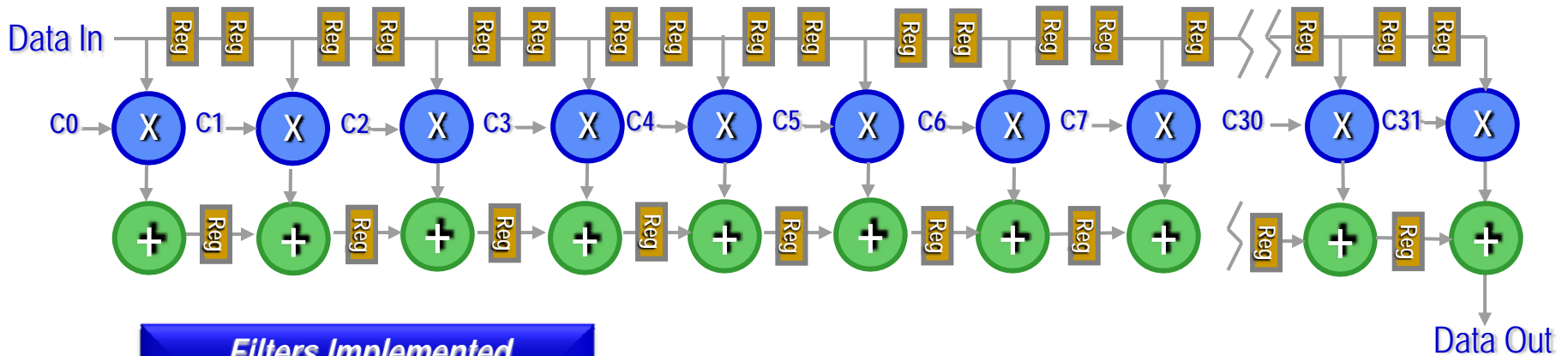
Parallel Adder Tree Implementation



**32 TAP filter implementation will consume 1,461 logic cells to implement adders in fabric**

# Pipelined Parallel Adder Implementation

Parallel Adder Cascade Implementation



*Filters Implemented  
Entirely Within the  
XtremeDSP Slice*

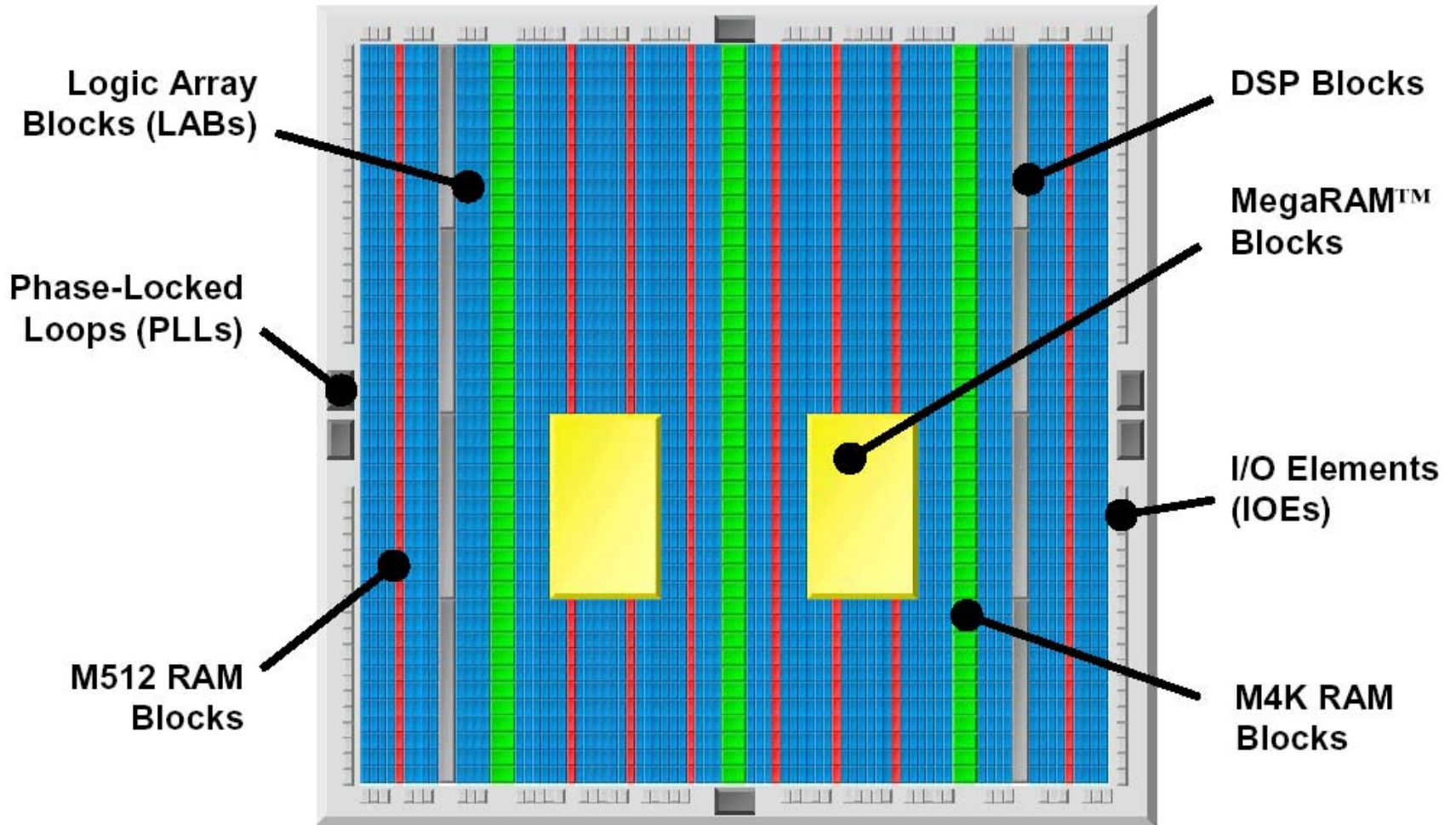
*Guaranteed 500MHz Performance  
Regardless of Filter Size*

- 32 TAP filter implementation using 32 XtremeDSP Slices

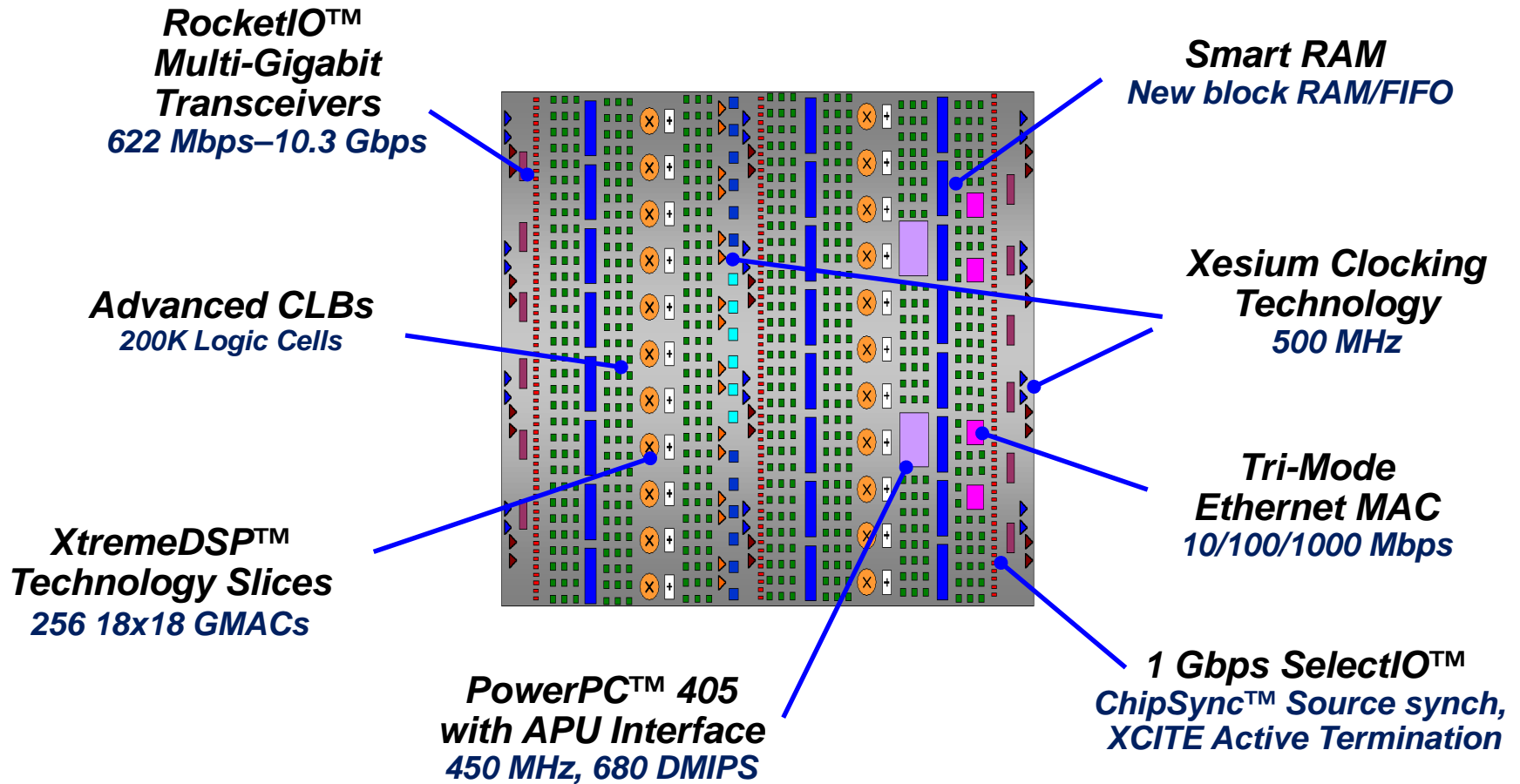
# Overview

- **Most FPGAs contain the same basic resources**
  - Slices grouped into combinational logic blocks (CLBs)
    - Contain combinatorial logic and register resources
  - Input/Output Blocks (IOBs)
    - Interface between the FPGA and the outside world
  - Programmable interconnect
  - Other resources
    - Memory
    - Multipliers
    - Global clock buffers
    - Boundary scan logic

# Altera Stratix Resource Block Diagram



# Virtex-4 Resource Block Diagram



# Basic Architecture of an FPGA

# Logic Fabric

## Logic Cell

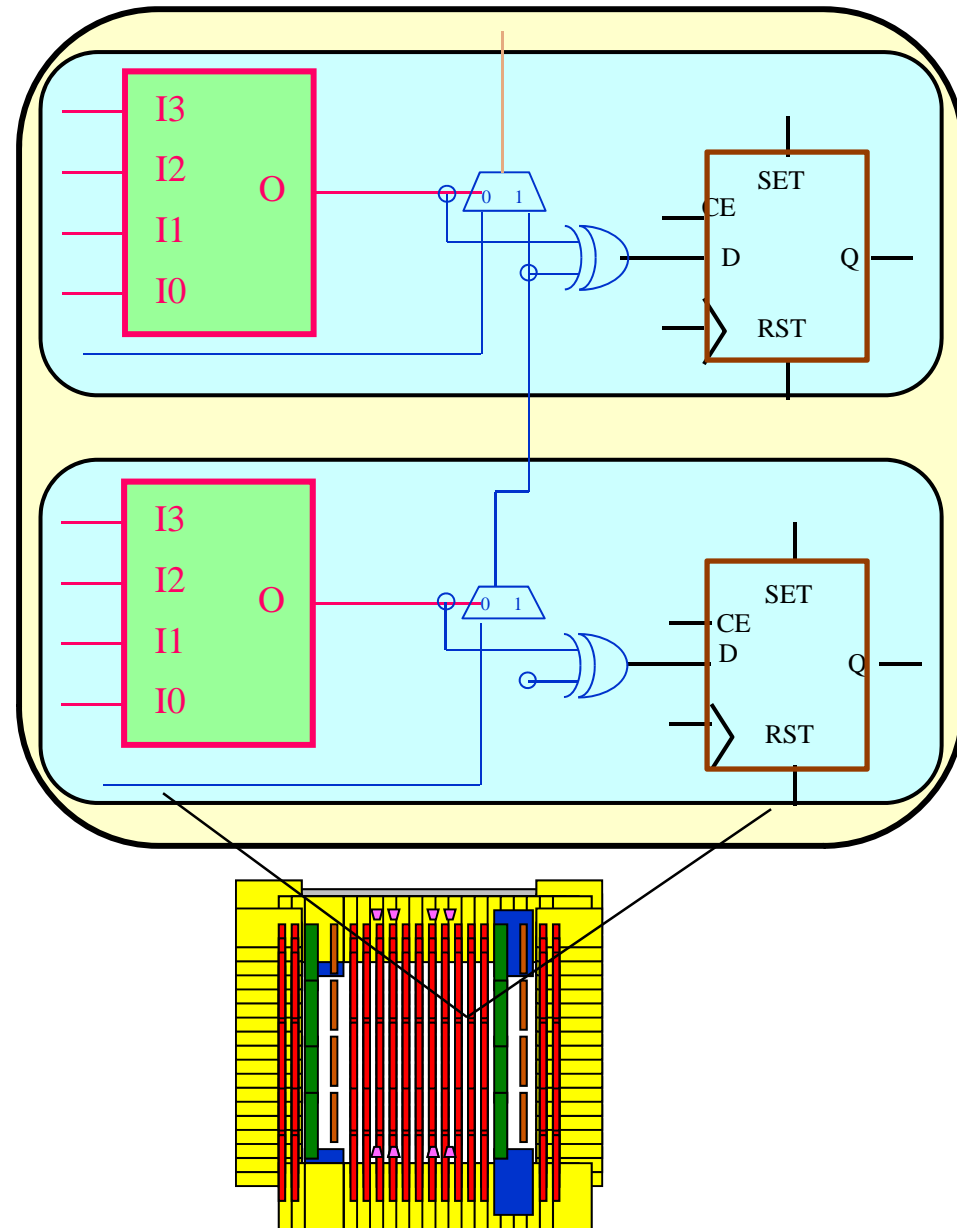
- Lookup table (LUT)
- Flip-Flop
- Carry logic
- Muxes (not shown)

## Slice

- Two Logic Cells

## Spartan-3E FPGAs

- 2K to 33K logic cells

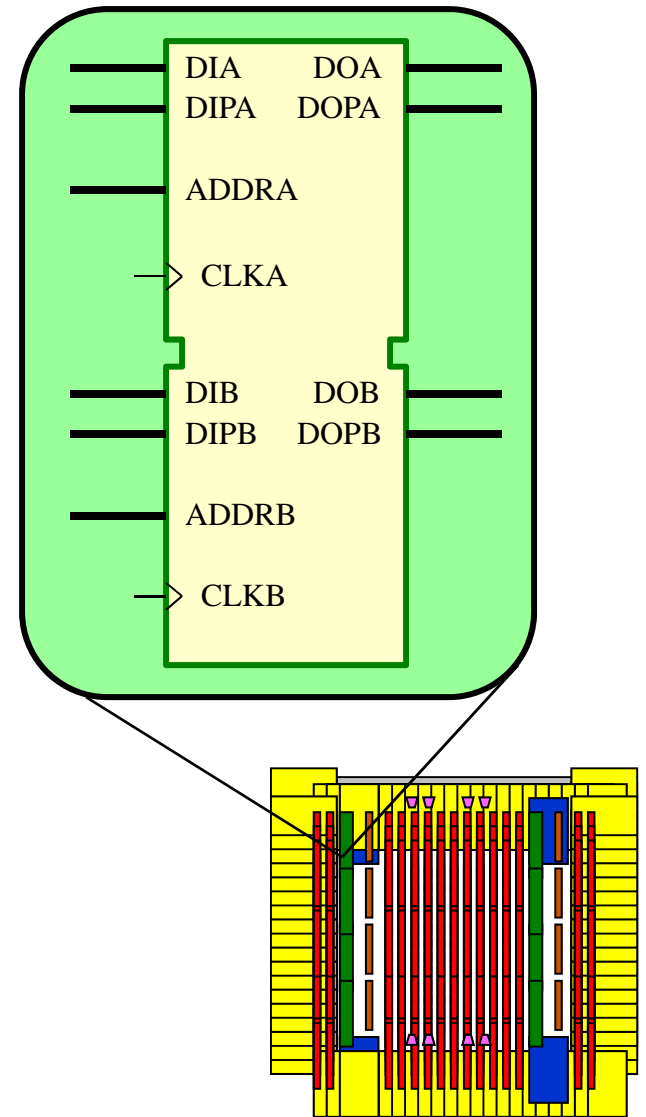


Slide courtesy of: XILINX®

# Memory

- **Block RAM**
  - RAM or ROM
  - True dual port
    - Separate read and write ports
  - Independent port size
    - Data width translation
  - Excellent for FIFOs

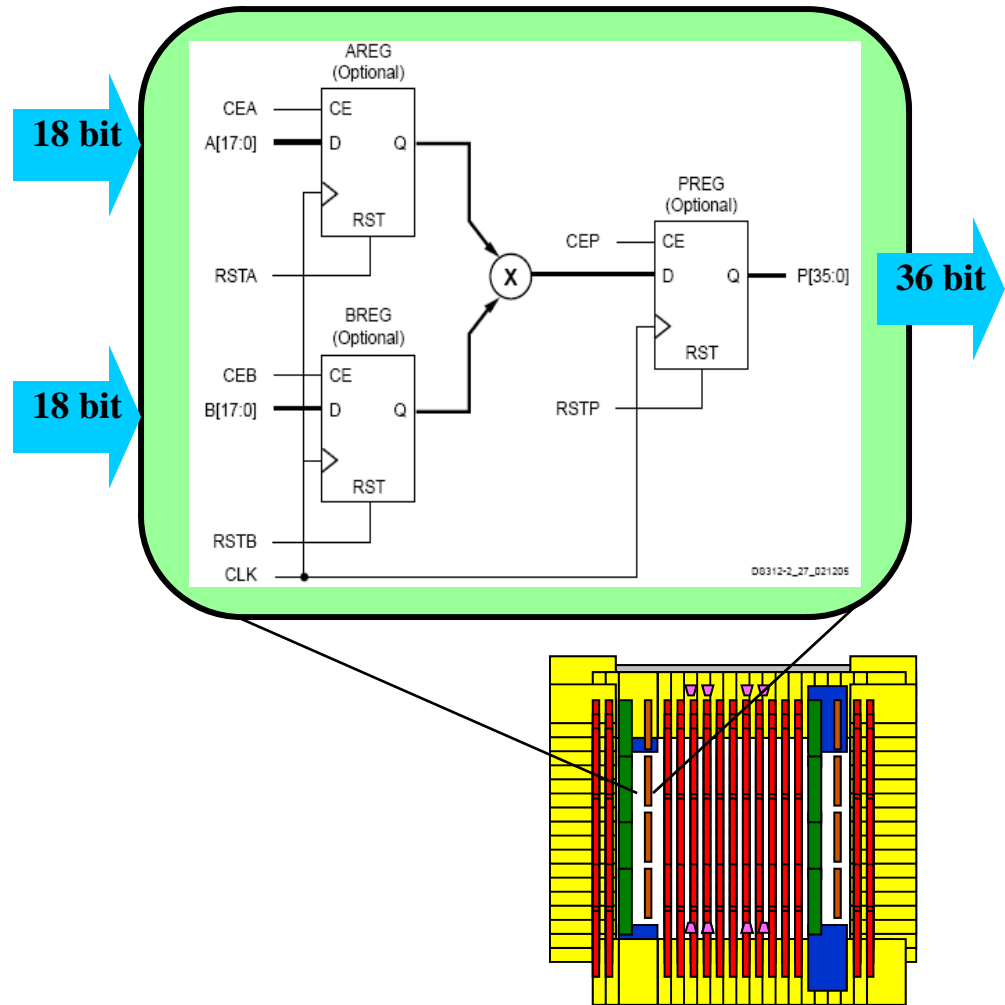
Block RAM Configurations			
Configuration	Depth	Data bits	Parity bits
16K x 1	16Kb	1	0
8K x 2	8Kb	2	0
4K x 4	4Kb	4	0
2K x 9	2Kb	8	1
1K x 18	1Kb	16	2
512 x 36	512	32	4



Slide courtesy of:  XILINX®

# Multipliers

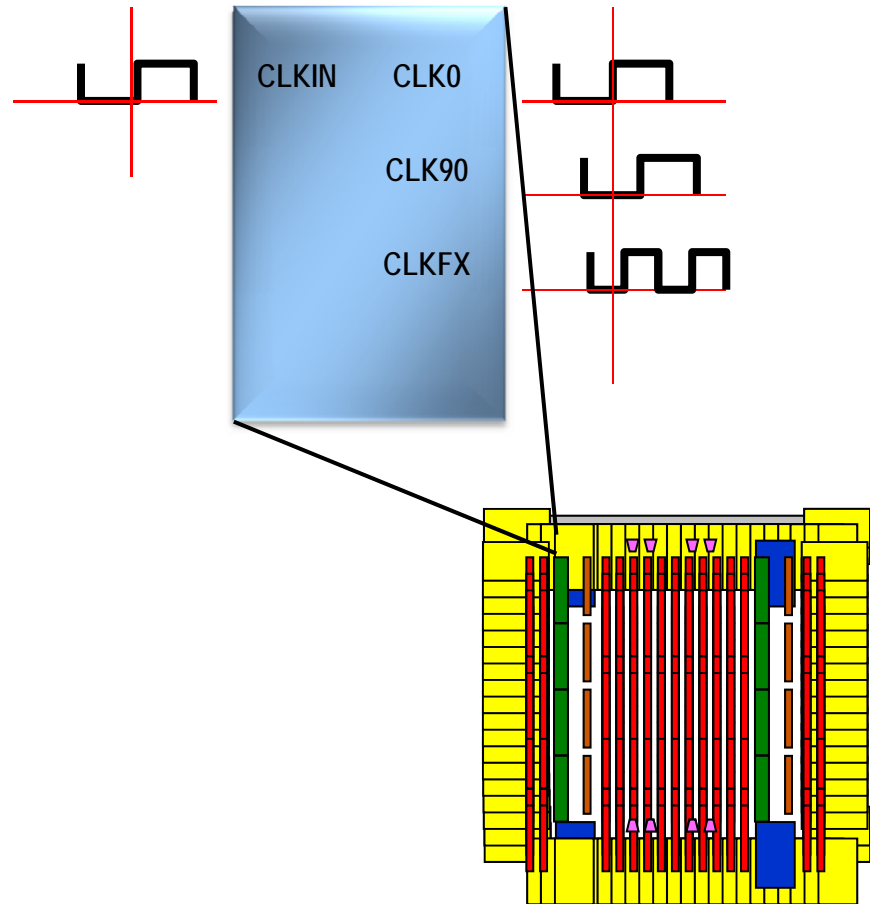
- **18 x 18 Multipliers**
  - Signed or unsigned
  - Optional pipeline stage
  - Cascadable



# Clock Management

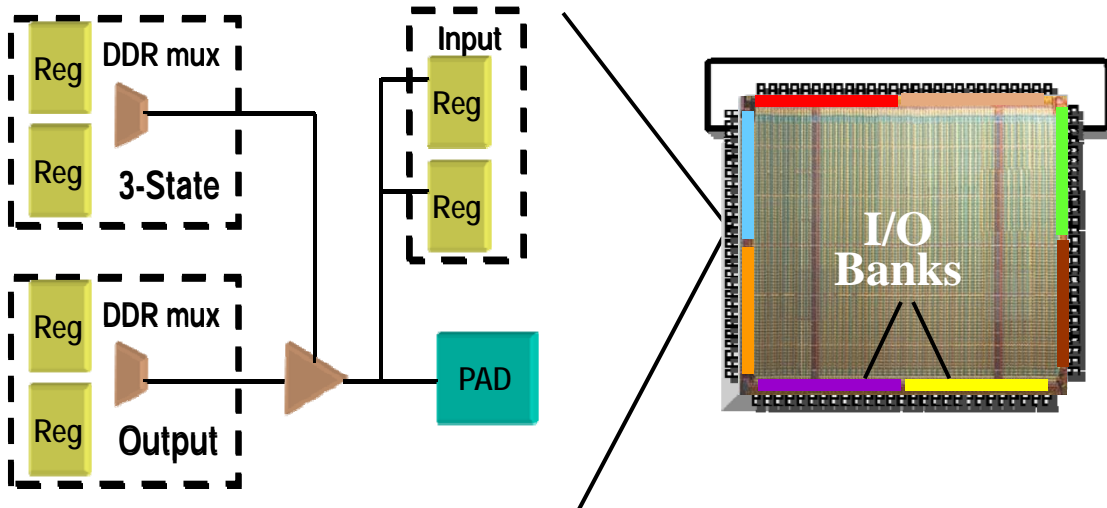
## ■ Digital Clock Managers (DCMs)

- Clock de-skew
- Phase shifting
- Clock multiplication
- Clock division
- Frequency synthesis



# Programmable I/Os

- **Single-ended**
- **Differential / LVDS**
- **Programmable I/O standards**
  - Multiple I/O banks
- **DDR I/O registers**
- **On-chip termination**

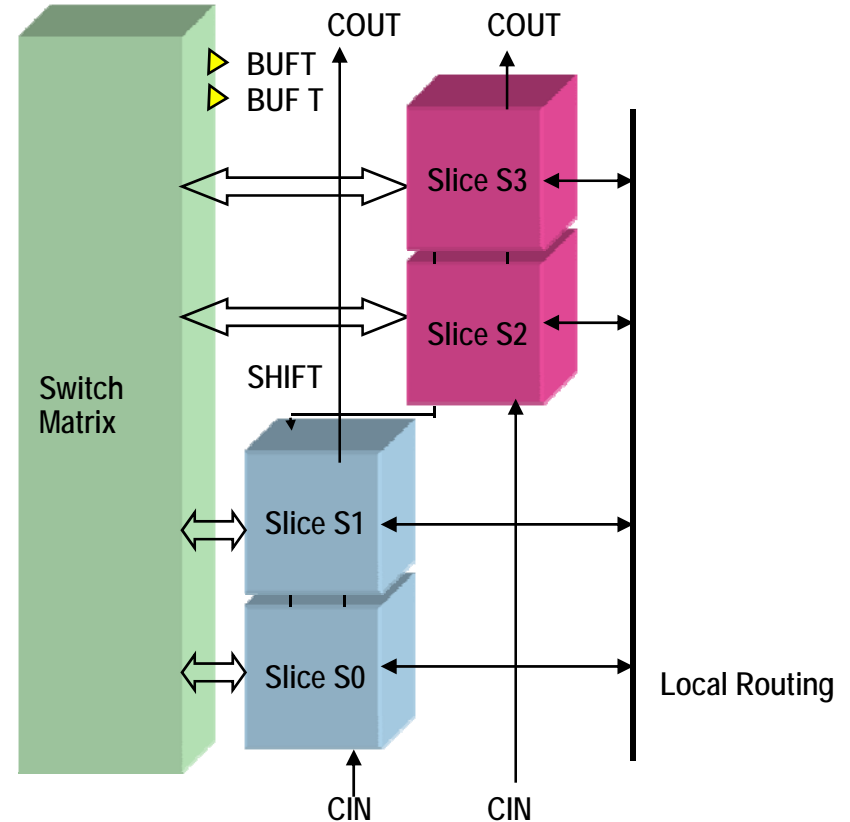


	Standard	Output $V_{CCO}$	Input $V_{REF}$
Single ended	LVTTTL	3.3V	--
	LVC MOS33	3.3V	--
	LVC MOS25	2.5V	--
	LVC MOS18	1.8V	--
	LVC MOS15	1.5V	--
	LVC MOS12	1.2V	--
	PCI 32/64 bit 33MHz	3.3V	--
	SSTL2 Class I	2.5V	1.25V
	SSTL2 Class II	2.5V	1.25V
	SSTL18 Class I	1.8V	0.9V
	HSTL Class I	1.5V	0.75V
	HSTL Class III	1.5V	0.9V
	HSTL18 Class I	1.8V	0.9V
	HSTL18 Class II	1.8V	0.9V
HSTL18 Class III	1.8V	1.1V	
Differential	GTL	--	0.8V
	GTL+	--	1.0V
	LVDS2.5	2.5V	--
	Bus LVDS2.5	2.5V	--
	Ultra LVDS2.5	2.5V	--
	LVDS_ext2.5	2.5V	--
	RSDS	2.5V	--
LDT2.5	2.5V	--	

# Logic Fabric

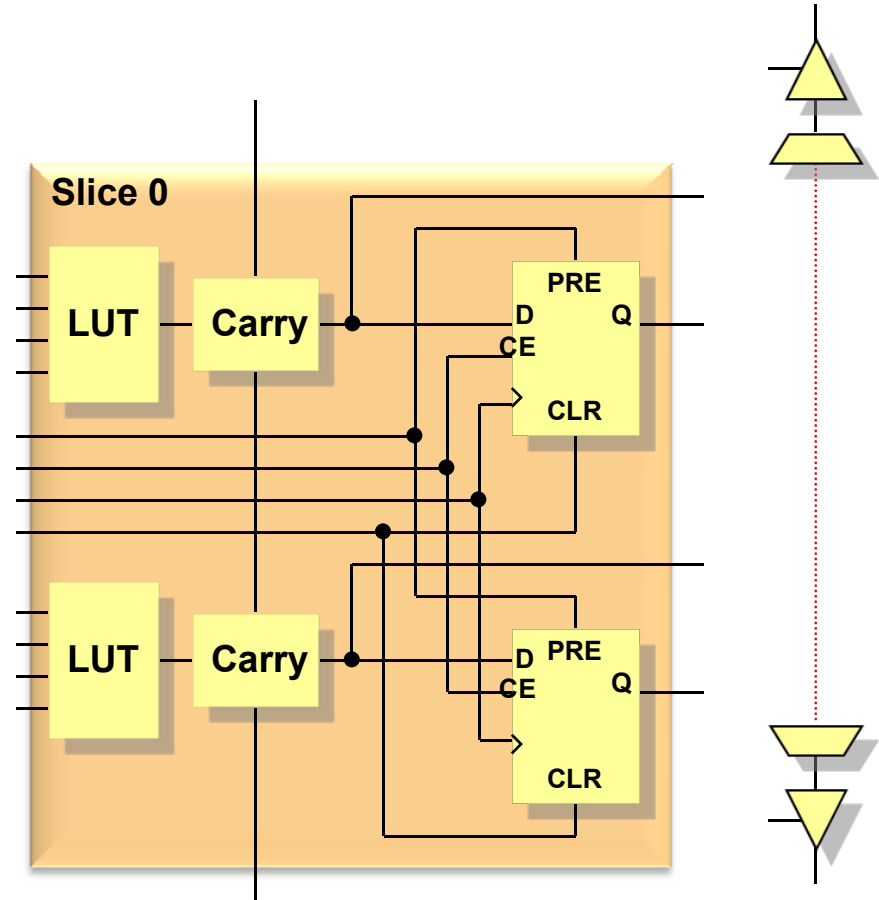
# Slices and CLB

- Each Virtex™-II CLB contains four slices
  - Local routing provides feedback between slices in the same CLB, and it provides routing to neighboring CLB
  - A switch matrix provides access to general routing resources



# Simplified Slice Structure

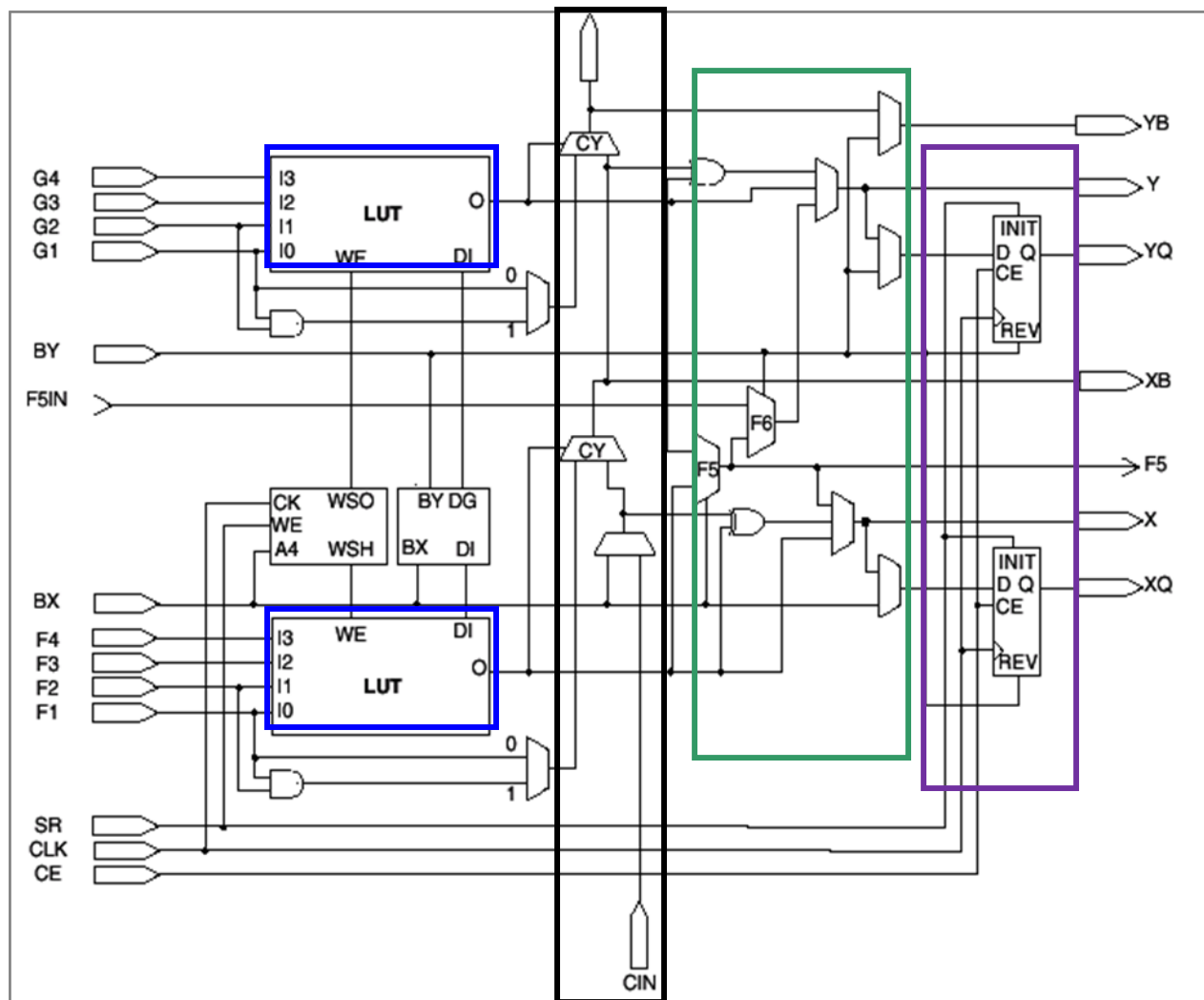
- **Each slice has four outputs**
  - Two registered outputs, two non-registered outputs
  - Two BUFTs associated with each CLB, accessible by all 16 CLB outputs
- **Carry logic runs vertically, up only**
  - Two independent carry chains per CLB



# Detailed Slice Structure

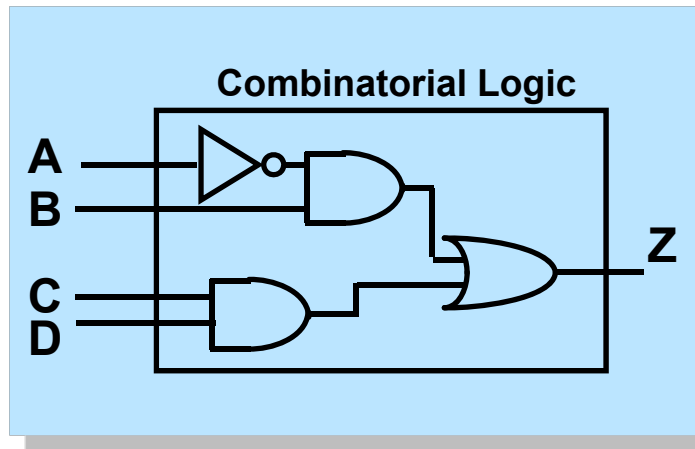
## The next few slides discuss the slice features

- LUTs
- MUXF5, MUXF6, MUXF7, MUXF8 (only the F5 and F6 MUX are shown in this diagram)
- Carry Logic
- MULT\_ANDs
- Sequential Elements



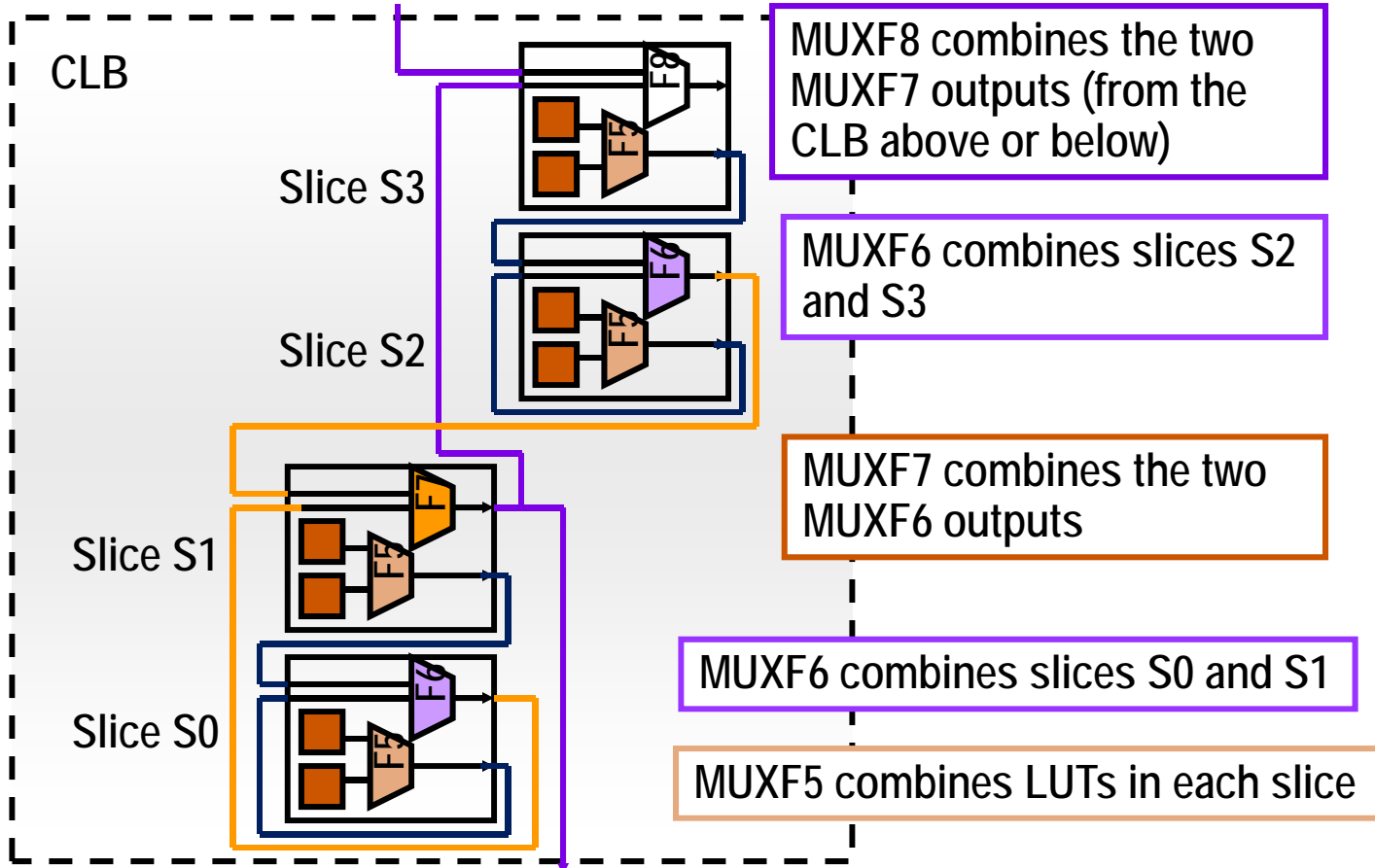
# Look-Up Tables

- **Combinatorial logic is stored in Look-Up Tables (LUTs)**
  - Also called Function Generators (FGs)
  - Capacity is limited by the number of inputs, not by the complexity
- **Delay through the LUT is constant**



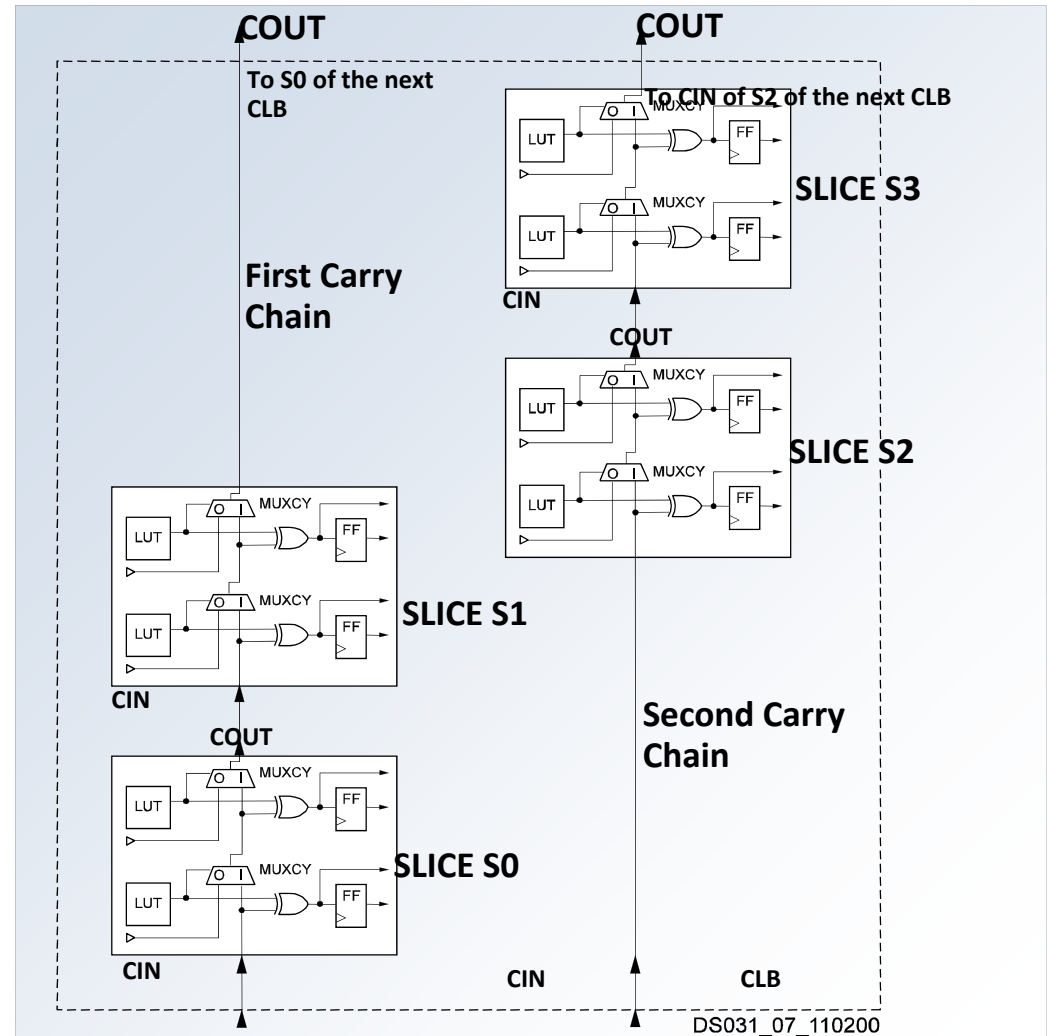
A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
	.	.	.	
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

# Connecting Look-Up Tables



# Fast Carry Logic

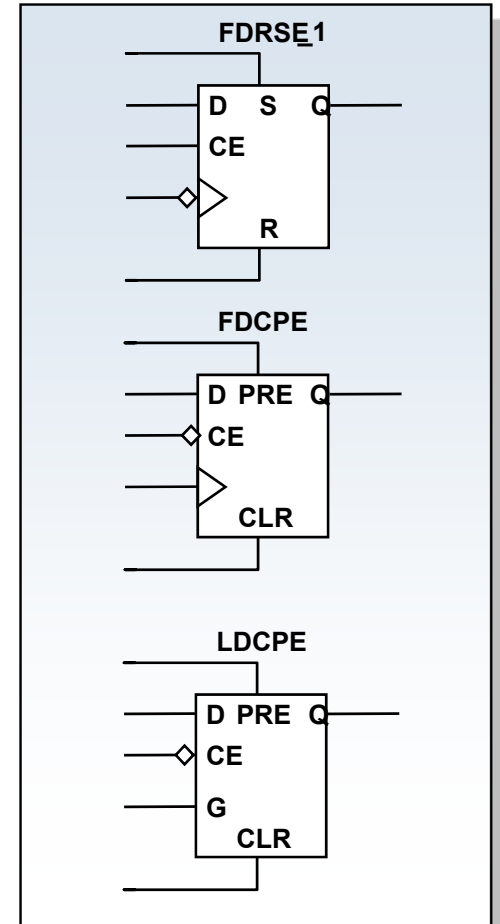
- **Simple, fast, and complete arithmetic Logic**
  - Dedicated XOR gate for single-level sum completion
  - Uses dedicated routing resources
  - All synthesis tools can infer carry logic





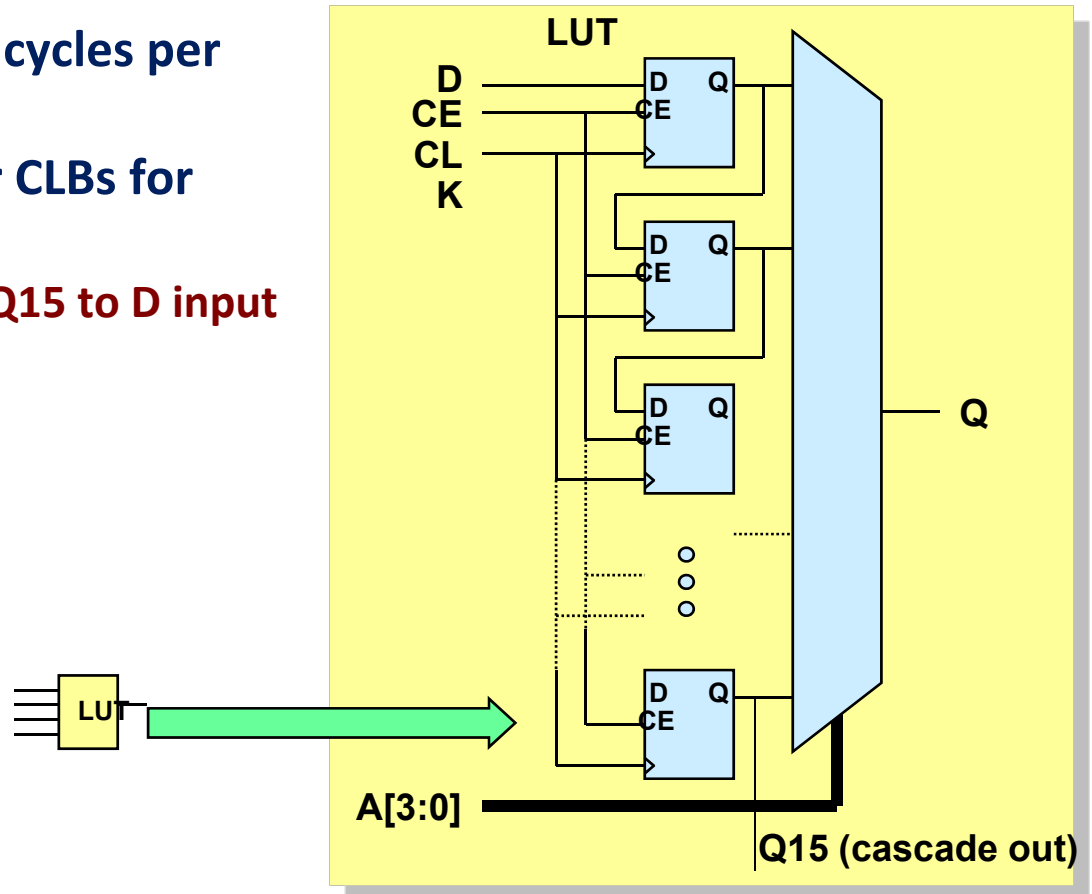
# Flexible Sequential Elements

- **Either flip-flops or latches**
- **Two in each slice; eight in each CLB**
- **Inputs come from LUTs or from an independent CLB input**
- **Separate set and reset controls**
  - Can be synchronous or asynchronous
- **All controls are shared within a slice**
  - Control signals can be inverted locally within a slice



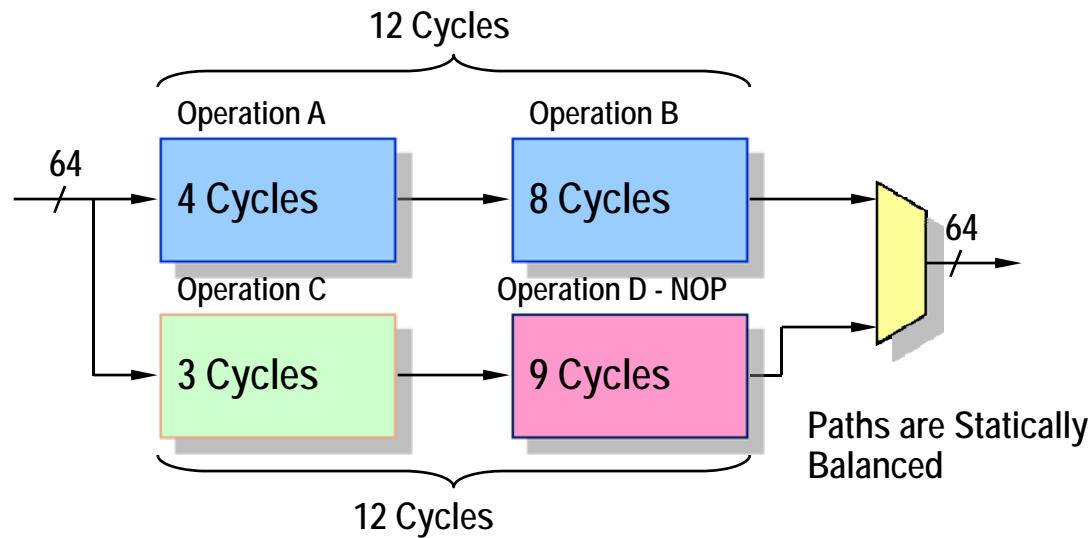
# Shift Register LUT (SRL16CE)

- **Dynamically addressable serial shift registers**
  - Maximum delay of 16 clock cycles per LUT (128 per CLB)
  - Cascadable to other LUTs or CLBs for longer shift registers
    - **Dedicated connection from Q15 to D input of the next SRL16CE**
  - Shift register length can be changed asynchronously by toggling address A



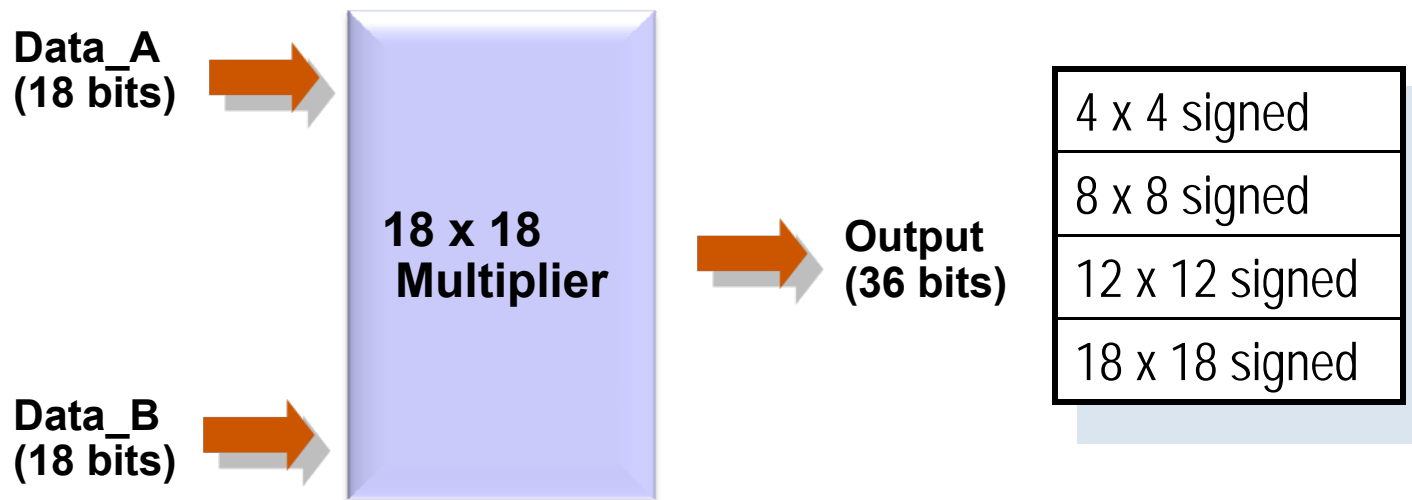
# Shift Register LUT Example

- **The SRL can be used to create a No Operation (NOP)**
  - This example uses 64 LUTs (8 CLBs) to replace 576 flip-flops (72 CLBs) and associated routing and delays



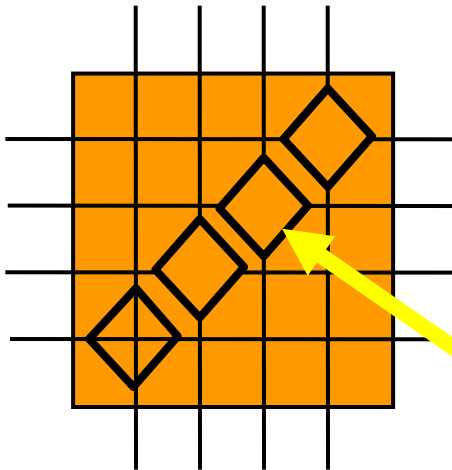
# Dedicated Multiplier Blocks

- 18-bit two's complement signed operation
- Optimized to implement Multiply and Accumulate functions
- Multipliers are physically located next to block SelectRAM™ memory

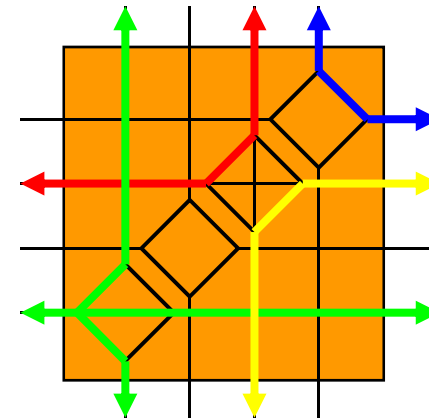


# Switchbox Operation

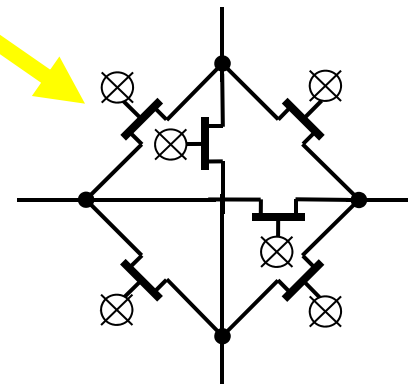
## Before Programming



## After Programming

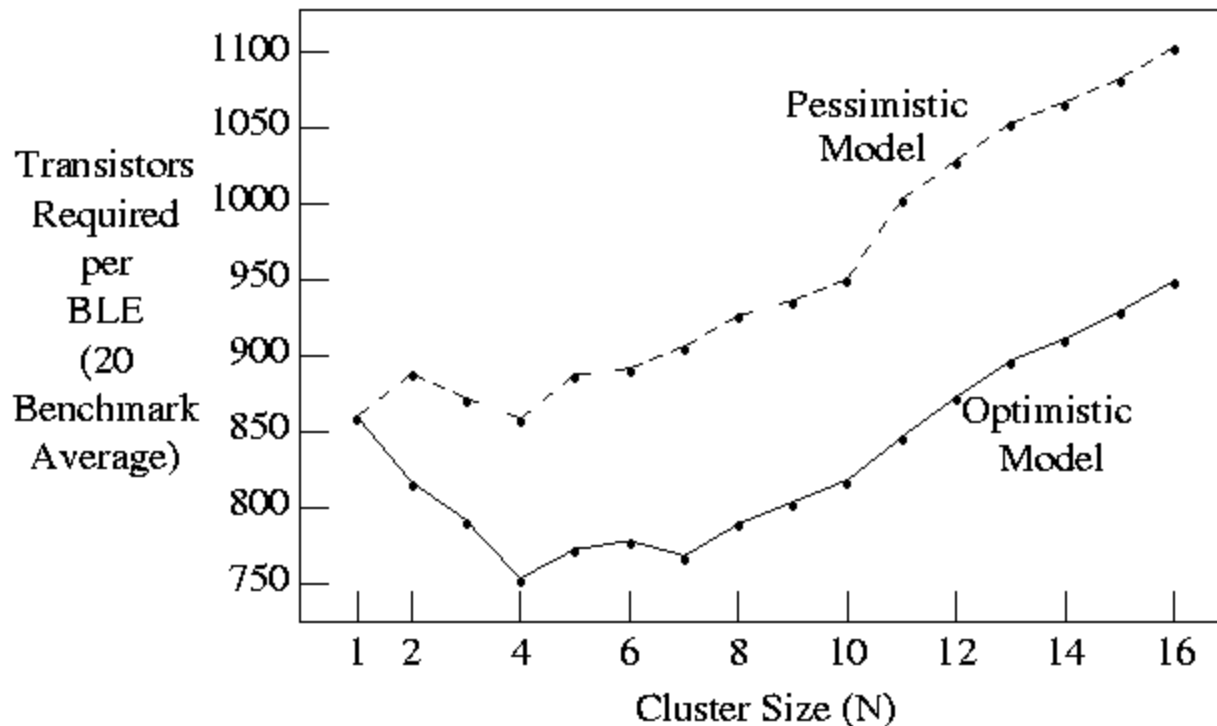


- 6 pass transistors per switchbox interconnect point
- Pass transistors act as programmable switches
- Pass transistor gates are driven by configuration memory cells



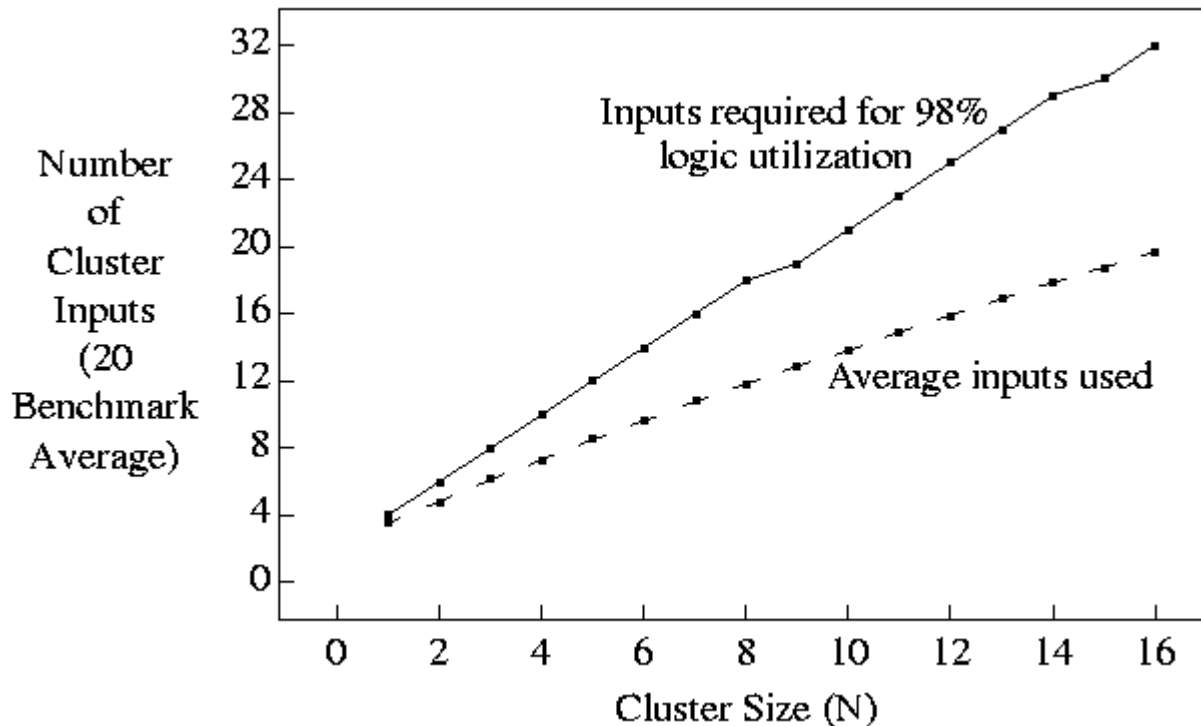
# Logic Cluster Size

- Interestingly, small block cluster more efficient (Betz – CICC’99)
- Includes area needed for routing.
- Small clusters (e.g. one BLE (Basic Logic Element) per cluster) are not “CAD friendly).
- Most commercial devices have 4-8 BLEs per cluster



# Number of Inputs per Cluster

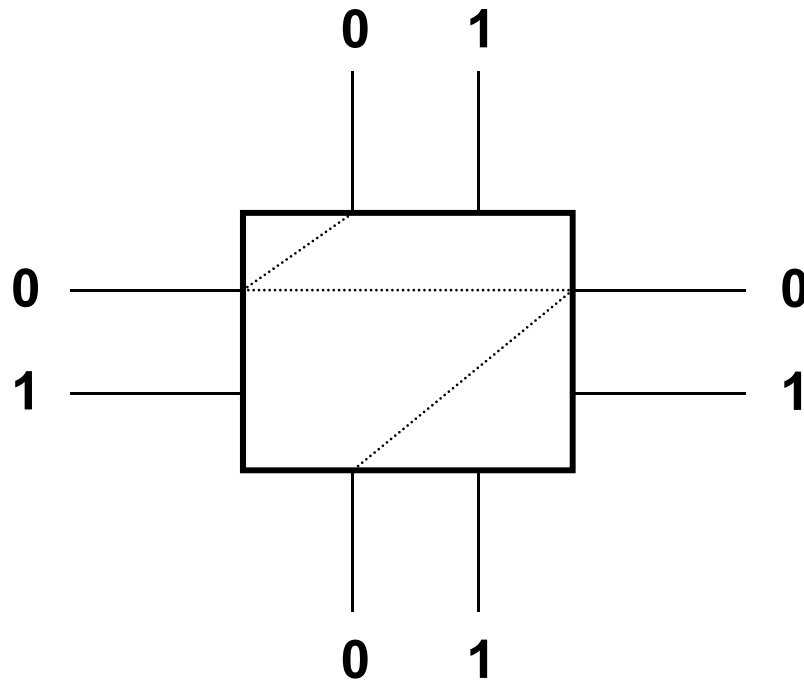
- Lots of opportunities for input sharing in large clusters (Betz – CICC'99)
- Reducing inputs reduces the size of the device and makes it faster.
- Most FPGA devices (Xilinx, Lucent) have 4 BLE per cluster with more inputs than actually needed.





# Switchbox Flexibility

- Switch box provides optimized interconnection area.
- Flexibility found to be not as important as  $F_C$
- Six transistors needed for  $F_S = 3$



# Putting it all together

## ■ Xilinx XC4000EX family

- $F_S = 3$
- $F_C = 0.2$
- $I = 8$
- Num BLE – about 2.5

## ■ Altera Flex10K family

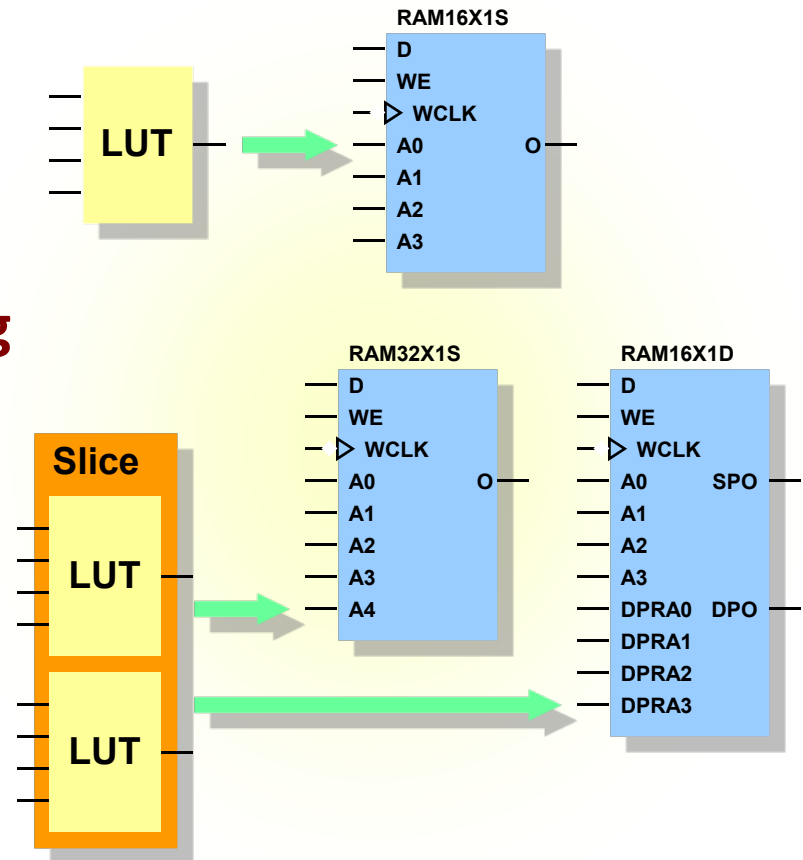
- $F_S = 3$
- $F_C = 0.25$
- $I = 22$
- Num BLE – about 28

- **NOTE: Altera doesn't use segmentation. All lines span large fraction of device.**

# Memory

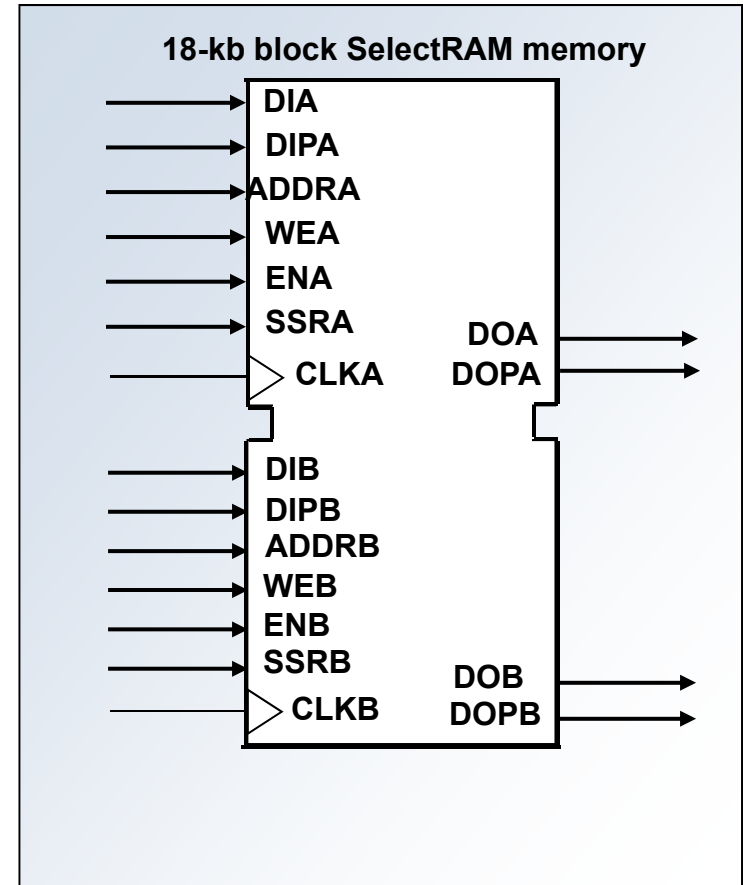
# Distributed SelectRAM Resources

- **Uses a LUT in a slice as memory**
- **Synchronous write**
- **Asynchronous read**
  - Accompanying flip-flops can be used to create synchronous read
- **RAM and ROM are initialized during configuration**
  - Data can be written to RAM after configuration
- **Emulated dual-port RAM**
  - One read/write port
  - One read-only port



# Block SelectRAM Resources

- **Up to 3.5 Mb of RAM in 18-kb blocks**
  - Synchronous read and write
- **True dual-port memory**
  - Each port has synchronous read and write capability
  - Different clocks for each port
- **Supports initial values**
- **Synchronous reset on output latches**
- **Supports parity bits**
  - One parity bit per eight data bits



# Clocking

# Global Clock Routing Resources

- **Sixteen dedicated global clock multiplexers**
  - Eight on the top-center of the die, eight on the bottom-center
  - Driven by a clock input pad, a DCM, or local routing
- **Global clock multiplexers provide the following:**
  - Traditional clock buffer (BUFG) function
  - Global clock enable capability (BUFGCE)
  - Glitch-free switching between clock signals (BUFGMUX)
- **Up to eight clock nets can be used in each clock region of the device**
  - Each device contains four or more clock regions

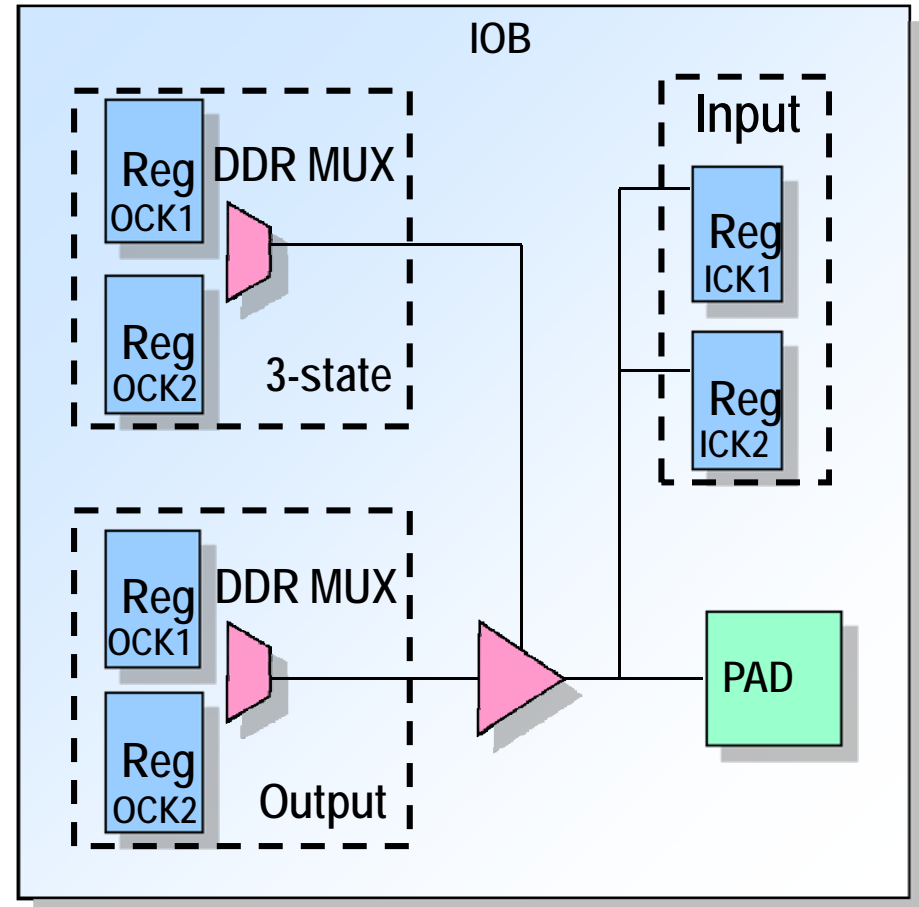
# Digital Clock Manager (DCM)

- **Up to twelve DCMs per device**
  - Located on the top and bottom edges of the die
  - Driven by clock input pads
- **DCMs provide the following:**
  - Delay-Locked Loop (DLL)
  - Digital Frequency Synthesizer (DFS)
  - Digital Phase Shifter (DPS)
- **Up to four outputs of each DCM can drive onto global clock buffers**
  - All DCM outputs can drive general routing

# Input Output Resources

# IOB Element

- **Input path**
  - Two DDR registers
- **Output path**
  - Two DDR registers
  - Two 3-state enable DDR registers
- **Separate clocks and clock enables for I and O**
- **Set and reset signals are shared**



# SelectIO Standard

- **Allows direct connections to external signals of varied voltages and thresholds**
  - Optimizes the speed/noise tradeoff
  - Saves having to place interface components onto your board
- **Differential signaling standards**
  - LVDS, BLVDS, ULVDS
  - LDT
  - LVPECL
- **Single-ended I/O standards**
  - LVTTTL, LVCMOS (3.3V, 2.5V, 1.8V, and 1.5V)
  - PCI-X at 133 MHz, PCI (3.3V at 33 MHz and 66 MHz)
  - GTL, GTLP
  - and more!

# Digital Controlled Impedance (DCI)

## ■ DCI provides

- Output drivers that match the impedance of the traces
- On-chip termination for receivers and transmitters

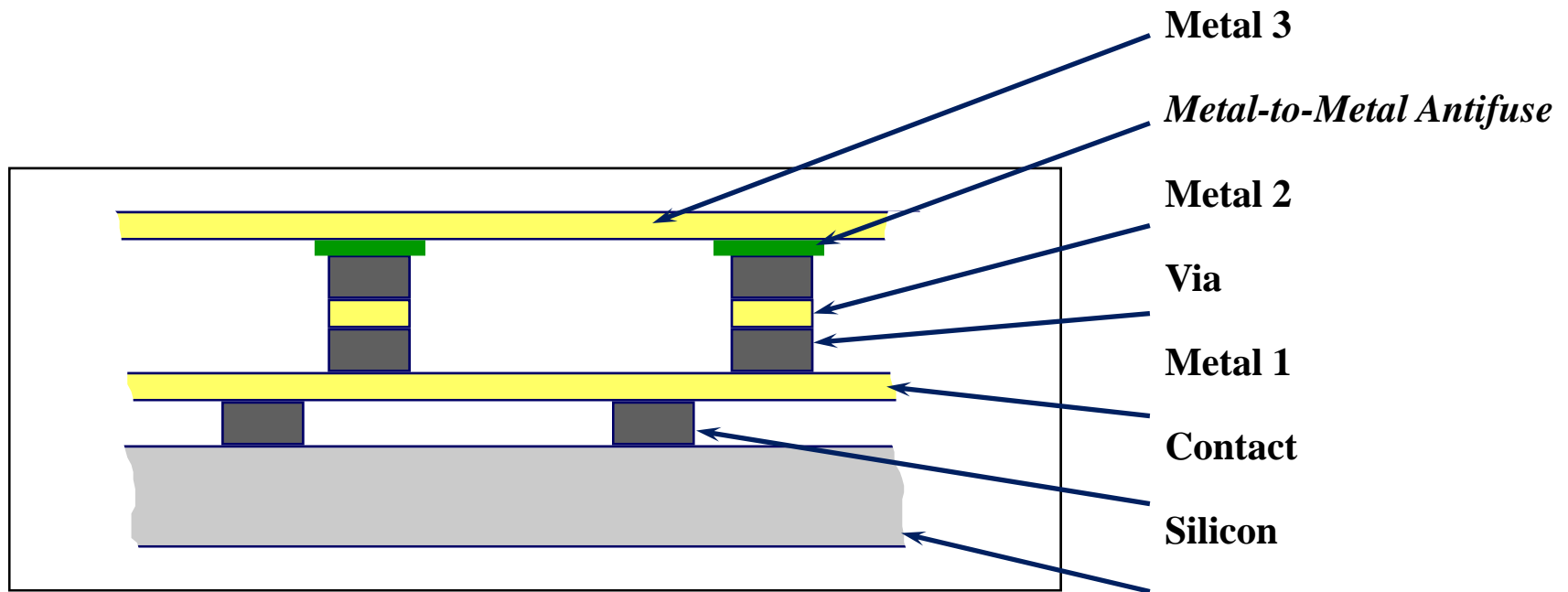
## ■ DCI advantages

- Improves signal integrity by eliminating stub reflections
- Reduces board routing complexity and component count by eliminating external resistors
- Eliminates the effects of temperature, voltage, and process variations by using an internal feedback circuit

# Programming Configurations

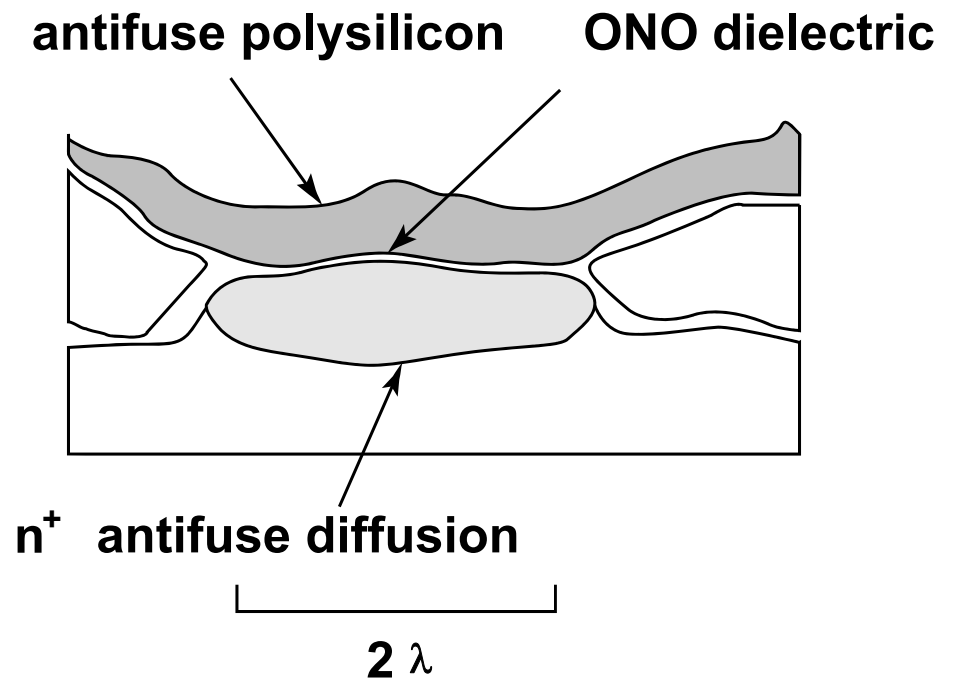
# Anti-fuse Switch

- **Anti-fuses are one-time programmable.**
  - Pulse eliminates dielectric
  - Only need to program once.

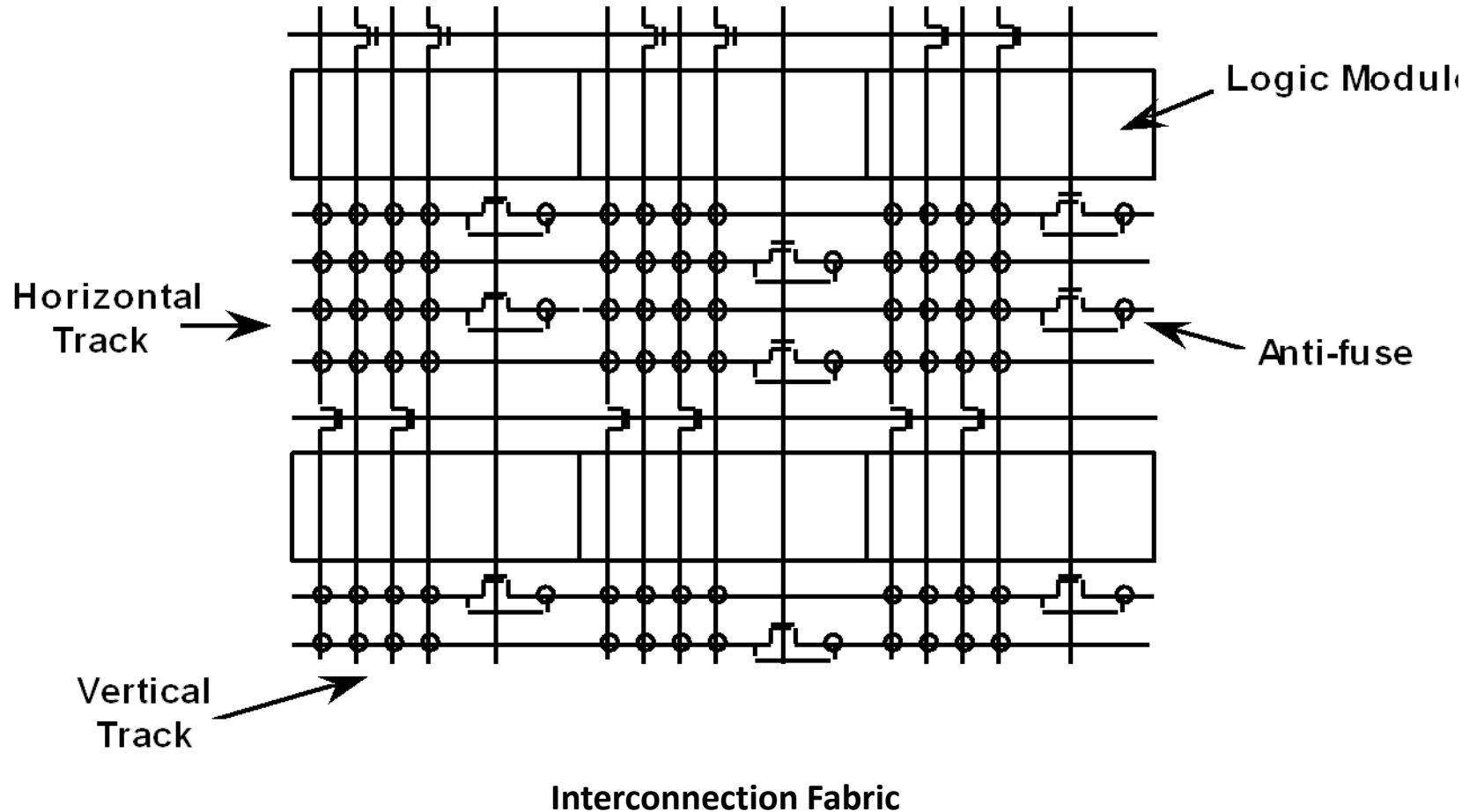


# Anti-Fuse FPGA

- Negligible programming overhead
- Low capacitance routing (fast)
- Security
- Tolerant of firm errors
- Resistance of about  $100 \Omega$

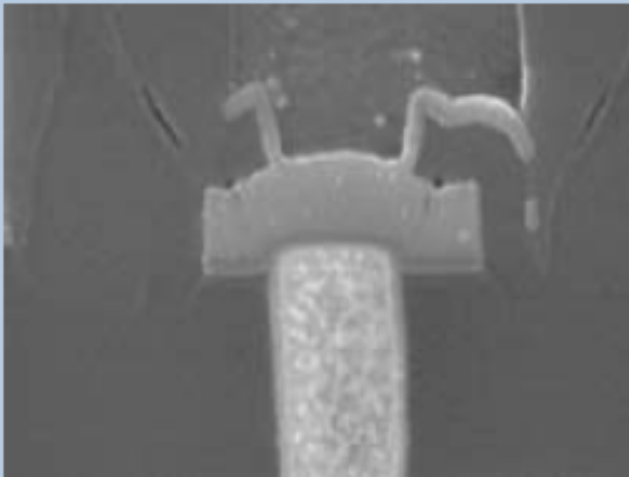


# Typical Actel Anti-fuse Interconnect

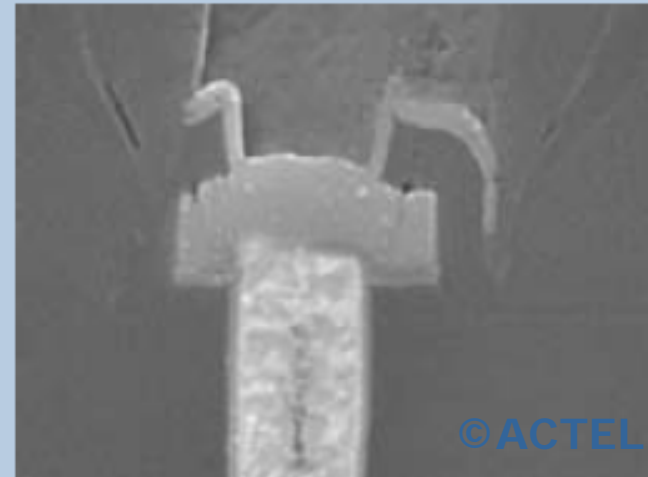


# Anti-fuse Security

- **Very good for design security**
  - No bitstream can be intercepted in the field (no bitstream transfer, no external configuration device)
  - Need a Scanning Electron Microscope (SEM) to try to know the antifuse states (an Actel AX2000 antifuse FPGA contains 53 million antifuses with only 2-5% programmed in an average design)



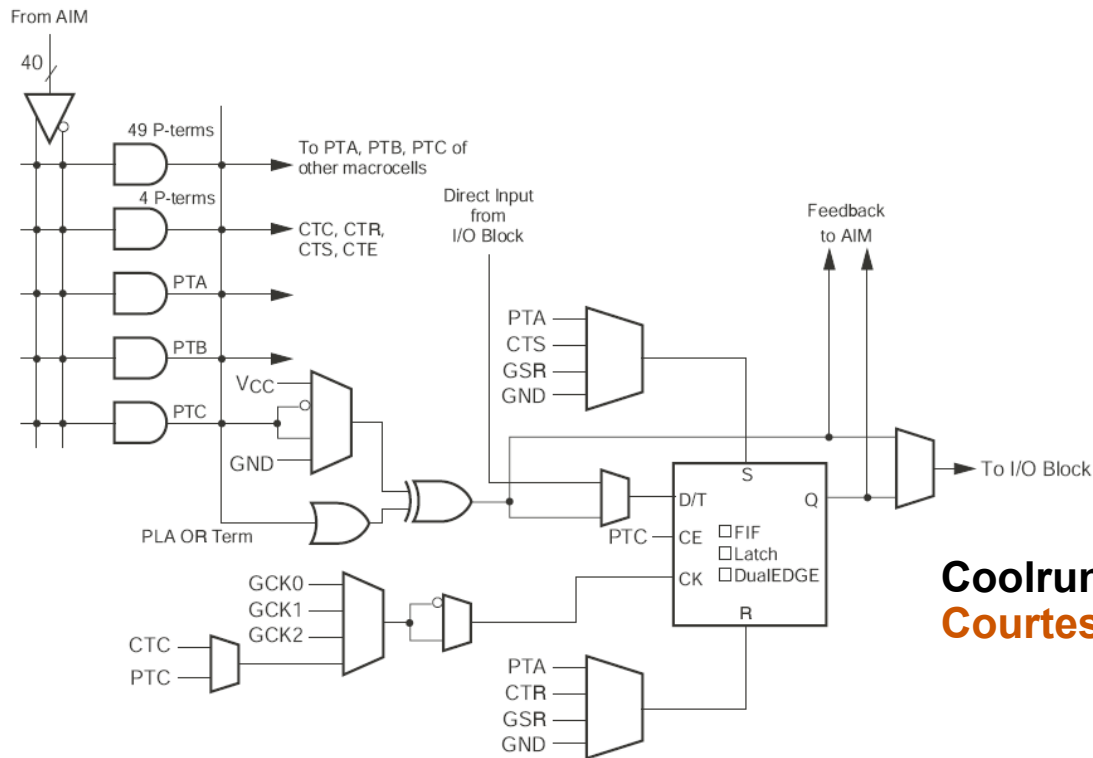
Unprogrammed Antifuse Element



Programmed Antifuse Element

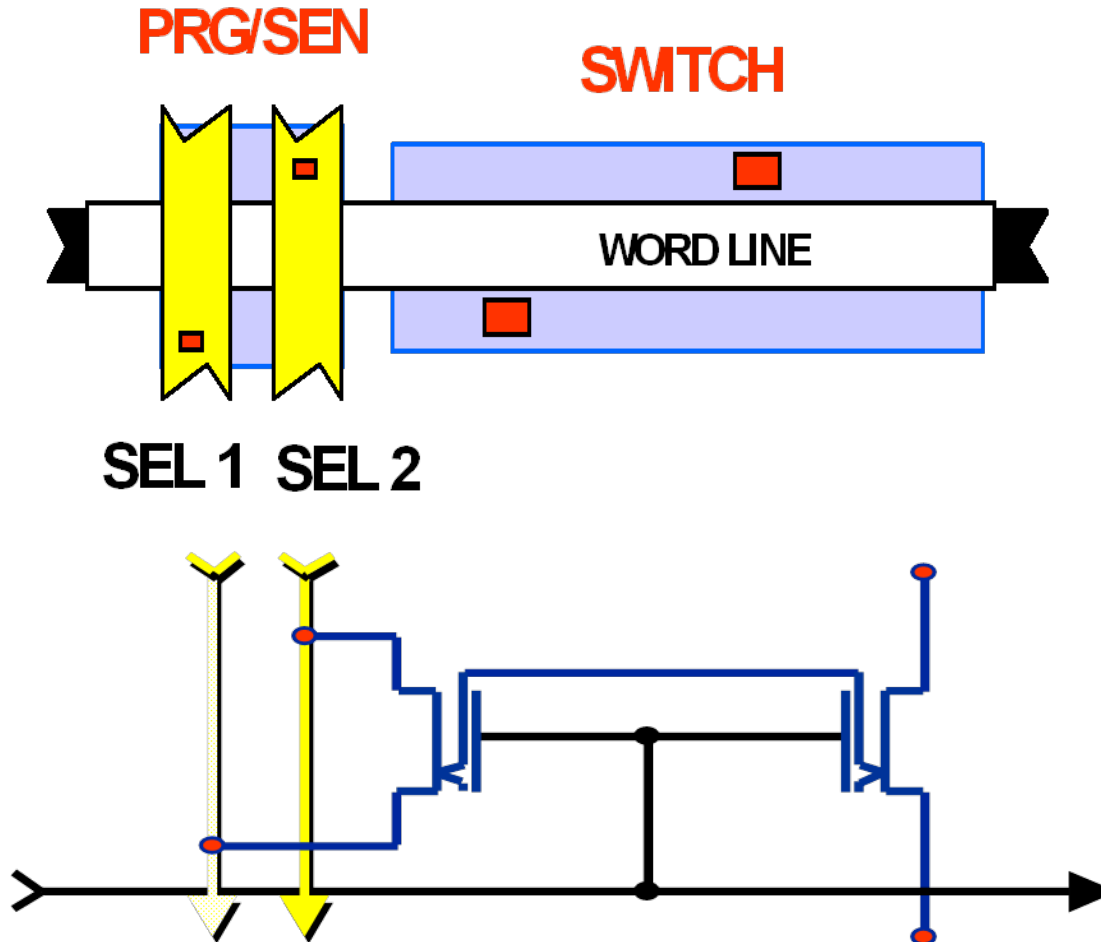
# Flash / EEPROM Devices

- Migrated from early PLD technology
- Traditionally based on AND/OR architecture
- Most often used for glue logic, state machines, etc
- High ratio of logic-to-registers

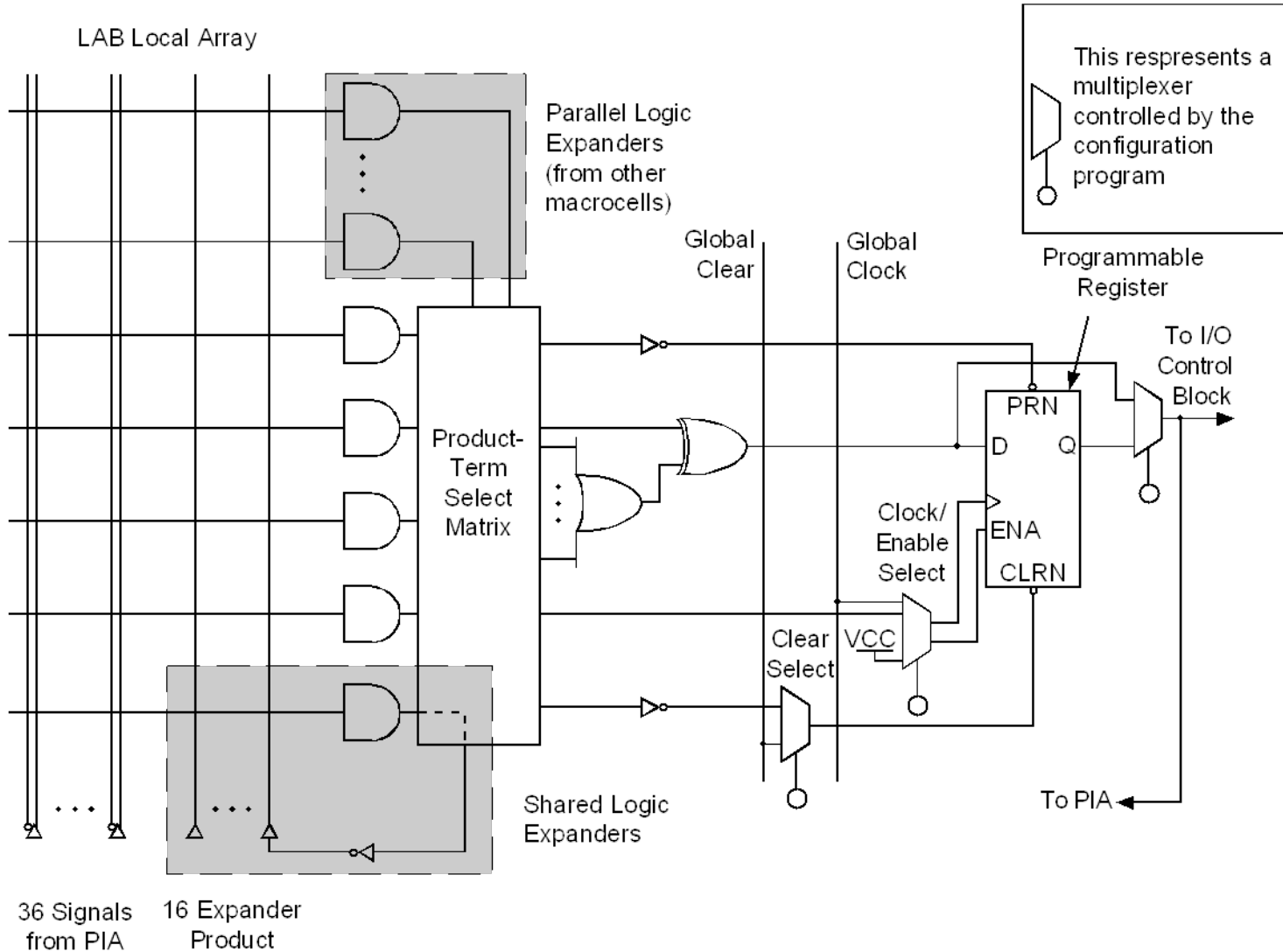


**Coolrunner II**  
 Courtesy: Xilinx

# FLASH-Memory Switch

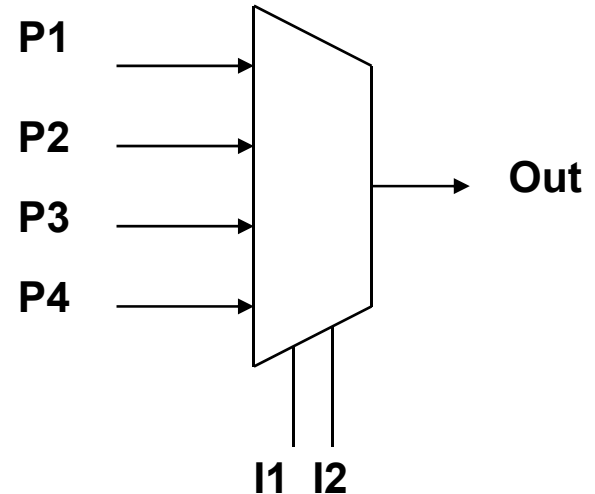
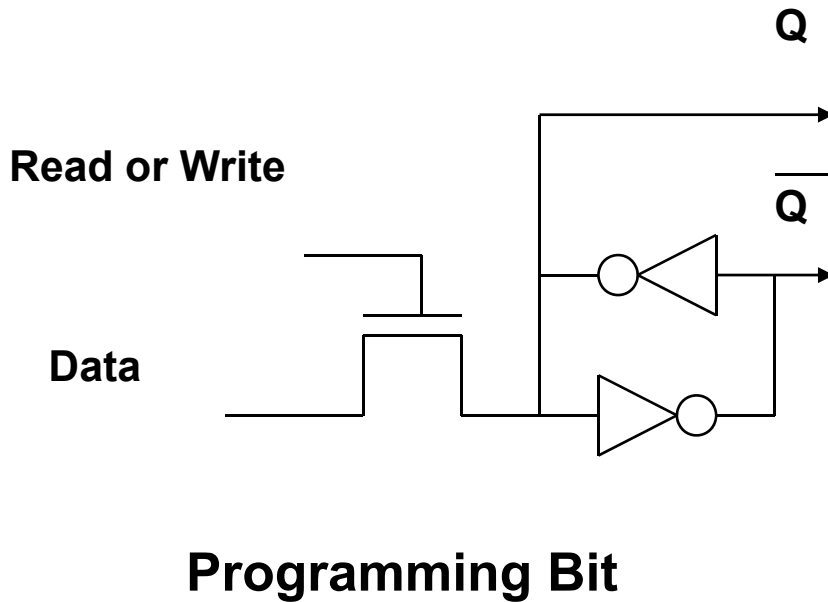


# Altera Max 7000 Macrocell



# SRAM-based FPGA

- SRAM bits can be programmed many times
- Each programming bit takes up five transistors
- Larger device area reduces speed versus EPROM and anti-fuse.



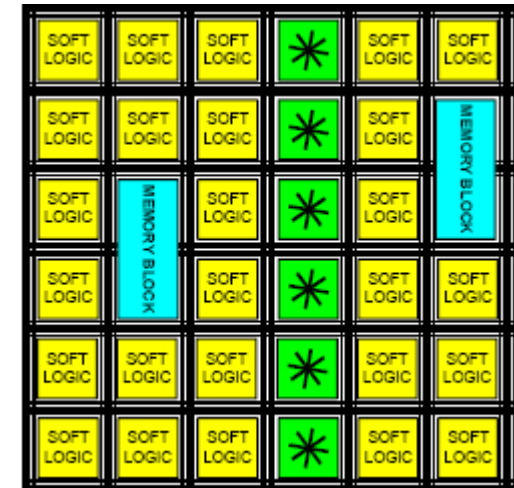
2-Input LUT

# Heterogeneous FPGA Architectures

# Heterogeneity in FPGA Architectures

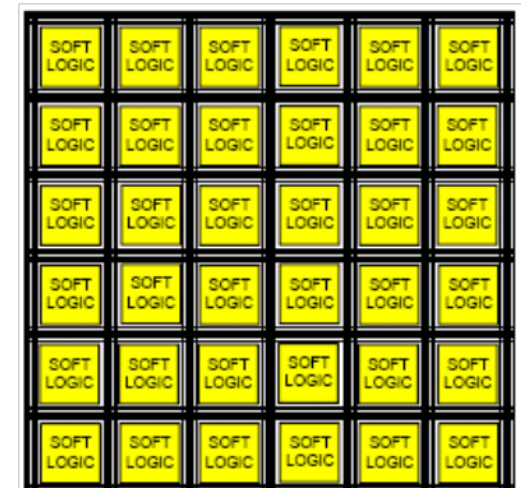
## ■ Heterogeneity among SLICES

- Programmable logic and routing
- Tiles are not identical
  - soft logic fabric [Kaviani, FPGA'96]]
  - hard structures [Jamieson, FPL'05]
- Dedicated hard structures
  - e.g. DSP
  - e.g. memory block



## ■ Heterogeneity within a SLICE

- Programmable logic and routing
- Tiles (SLICES) are identical
- Different logics exist within a SLICE
  - e.g. LUTs with different size [Cong, FPGA'99]
  - e.g. mixed PLAs and LUTs [Cong, TODAES'05]
  - e.g. mixed macro-gates and LUT



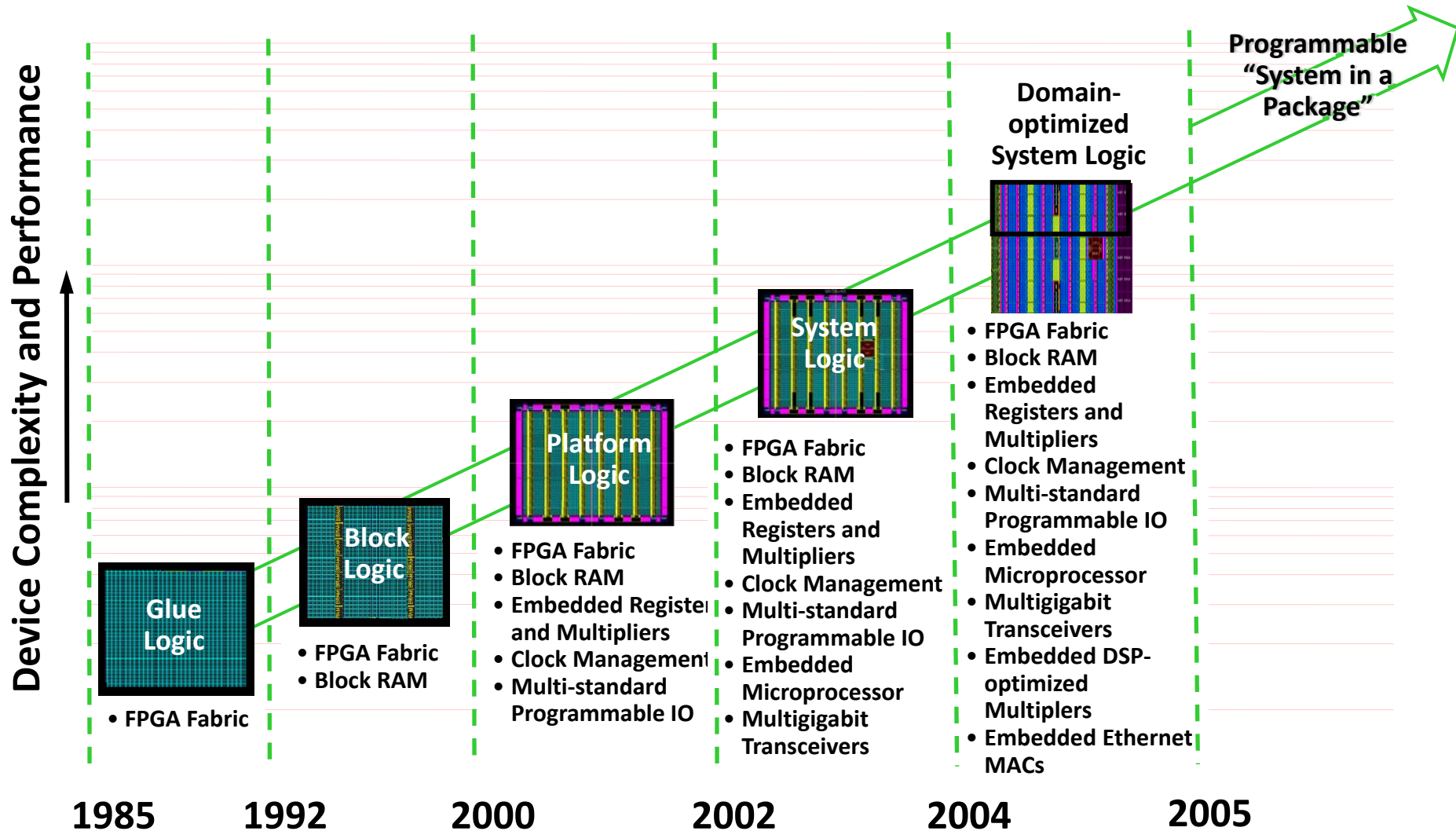
(source: Jamieson@FPL'05)

# Heterogeneous FPGA with Macro-Gates

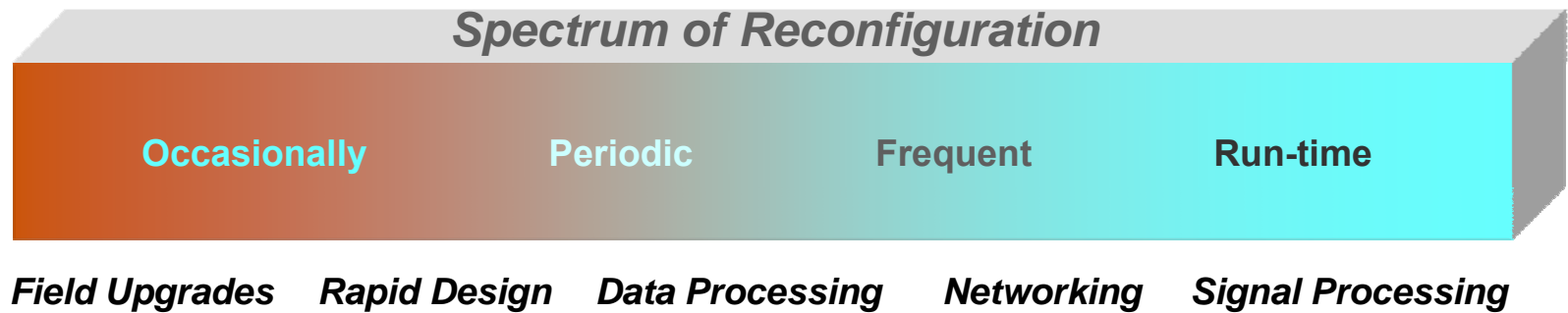
- **There exists programmability and cost trade-off between LUTs and macrogates**
  - Xilinx V4 benefits from small gates (MUX2, XOR2) built in SLICES.
- **The benefit of wider macro-gates**
  - Effectiveness of the incorporation of wider logic functions (macro gates) is not clear.
- **Our contributions**
  - Design a new FPGA architecture with mixed LUTs and macro-gates
  - Propose a new automatic synthesis flow for mapping a circuit to the proposed FPGA architecture
  - Evaluate the architecture and show that the proposed architecture reduces delay and area by 16.5% and 30%, respective, compared to the LUT-only architecture.

# Reconfigurable FPGAs

# Architectural Evolution Reconfigurable FPGAs



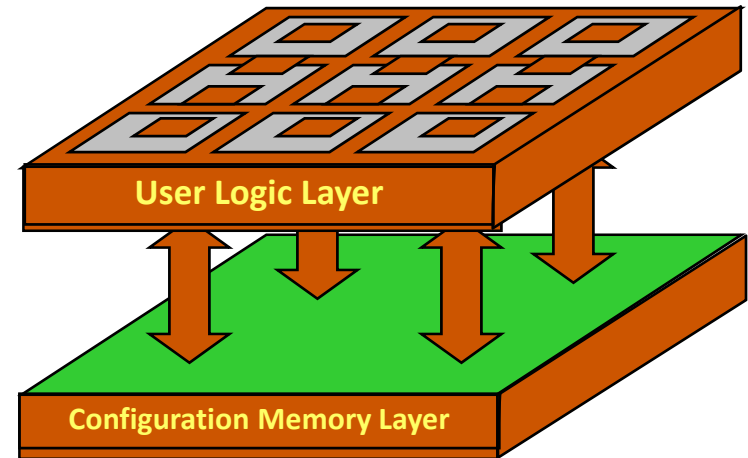
# New use models enabled with Reconfigurable FPGAs



- **More efficient use of hardware**
  - Adaptive hardware algorithms
  - Design modules that time share device resources
    - Reduced device count and lower power consumption

# FPGA Partial Reconfiguration

- **Think of an FPGA as Two Layers**
  - Configuration Memory Layer
  - User Logic Layer
- **Configuration memory controls functions on user logic layer**
- **Partial Reconfiguration allows a portion of device to be changed while the rest is still running**
- **Documented in XAPP 290**

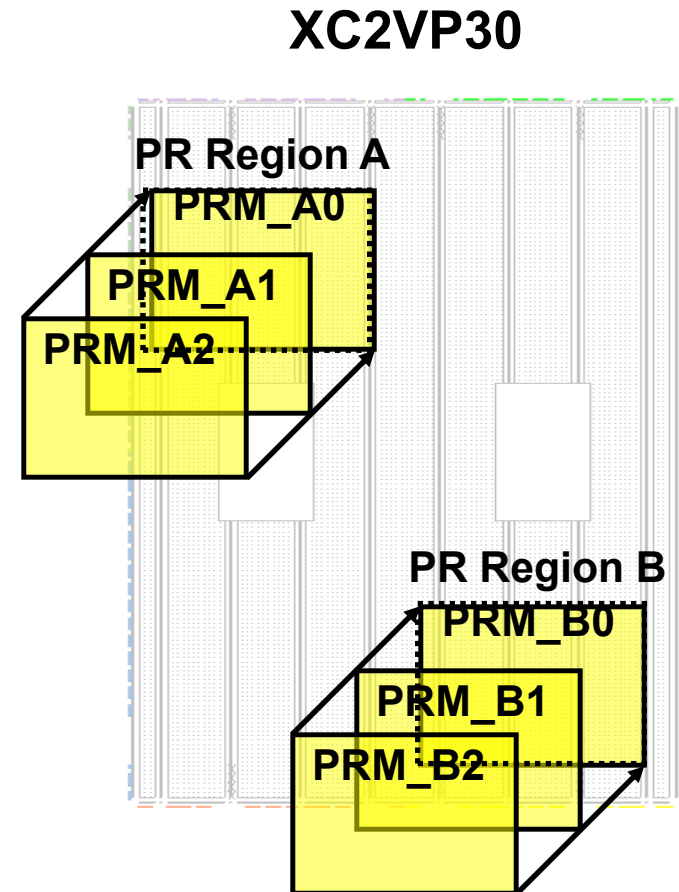


### *What FPGA Configuration Memory Controls*

- All interconnection (wiring)
- Logic Definition (Look-up Tables or "LUTs")
- Multiply by, divide by, etc.
- Inversion
- Feature selection
- Interface to hardwired blocks, e.g. PPC
- Pipeline on/off
- ECC enable/disable
- BRAM width
- I/O Modes

# Partial Reconfiguration Modules (PRMs)

- One or more PR regions can be defined
- Multiple PRMs can be defined for each region



# References

- [http://toolbox.xilinx.com/docsan/xilinx8/books/data/docs/dev/dev0001\\_1.html](http://toolbox.xilinx.com/docsan/xilinx8/books/data/docs/dev/dev0001_1.html)
- [http://www.xilinx.com/support/documentation/application\\_notes/xapp466.pdf](http://www.xilinx.com/support/documentation/application_notes/xapp466.pdf)
- [http://www.xilinx.com/support/documentation/application\\_notes/xapp290.pdf](http://www.xilinx.com/support/documentation/application_notes/xapp290.pdf)

# Backup

# Timing

# Timing Constraints

- **Timing Constraints give the tools a performance goal**
- **Place & Route uses timing constraints**
  - PAR runs the timing analysis tool in the background
  - Real-time analysis of current results against performance goals
- **Without constraints, PAR tries to reduce run-time**
  - Finishes quickly
  - Modest effort to optimize performance
- **With constraints, PAR tries to meet performance goals**
  - Run-time may be longer
  - Aggressive time constraints and higher effort levels can significantly increase run-time

## Basic Timing Constraints

- **PERIOD** – Target clock period for internal sequential paths
- **OFFSET IN BEFORE** – Target “input setup” time ( $T_{su}$ )
  - Reference between *external INPUT pin* and *CLK pin*
- **OFFSET OUT AFTER** – Target “clock to out” time ( $T_{co}$ )
  - Reference between *external CLK pin* and *OUTPUT pin*

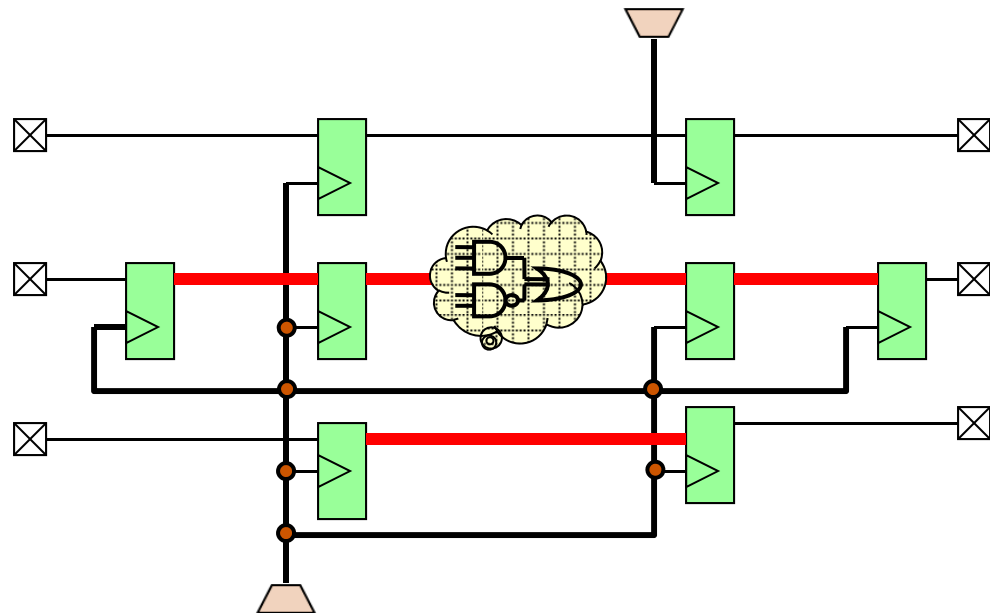
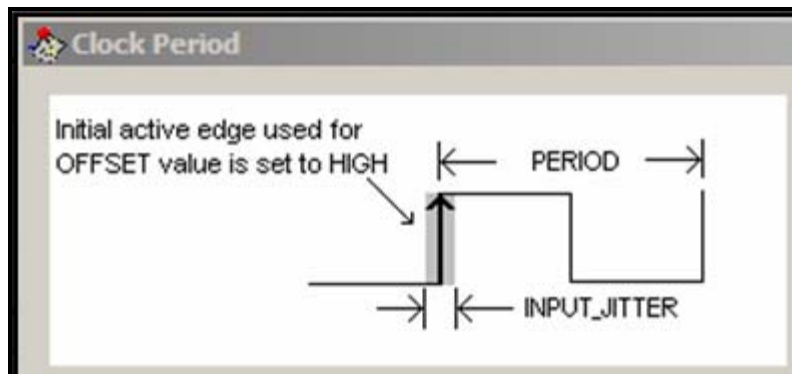
# PERIOD Constraint

## ■ CLOCK PERIOD

*NET "MYCLK" TNM\_NET = " MYCLK ";*

*TIMESPEC "TS\_MYCLK" = PERIOD " MYCLK " 10 ns HIGH 50 %;*

- Data paths between synchronous elements only
- Does not cover
  - Cross clock domains between unrelated clocks

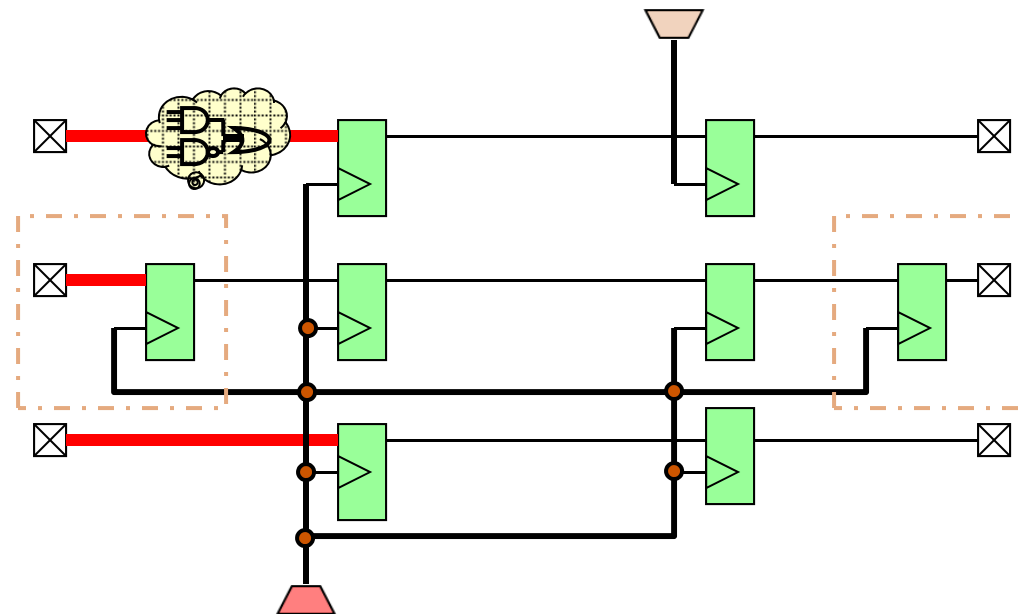
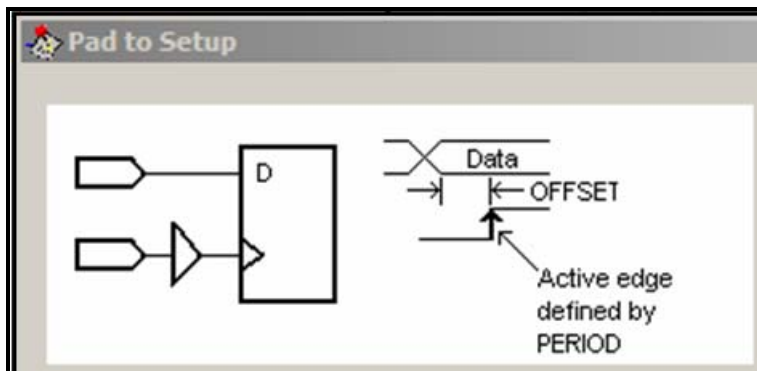


# OFFSET IN BEFORE Constraint

## ■ OFFSET IN BEFORE

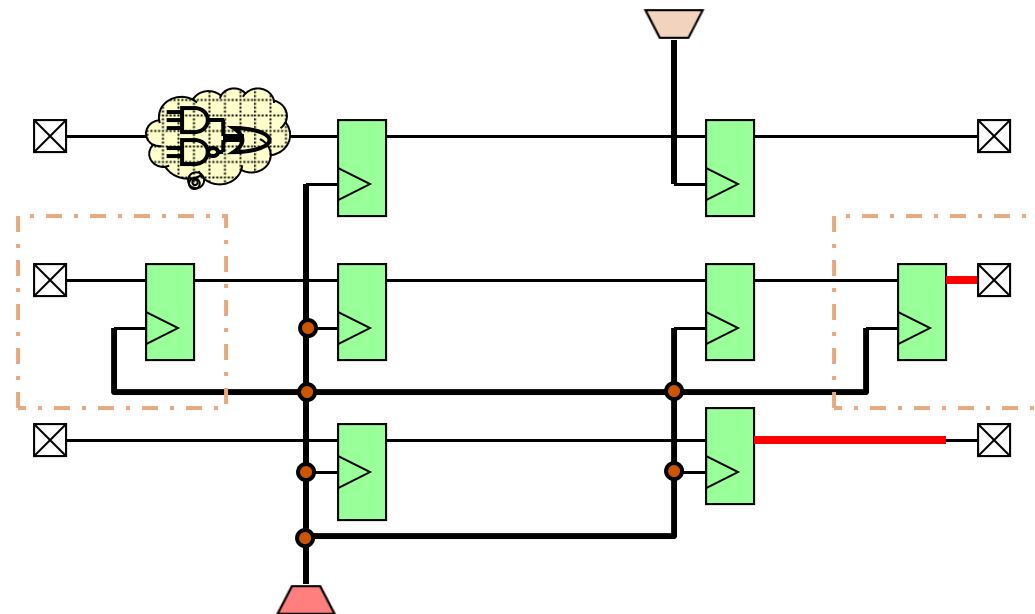
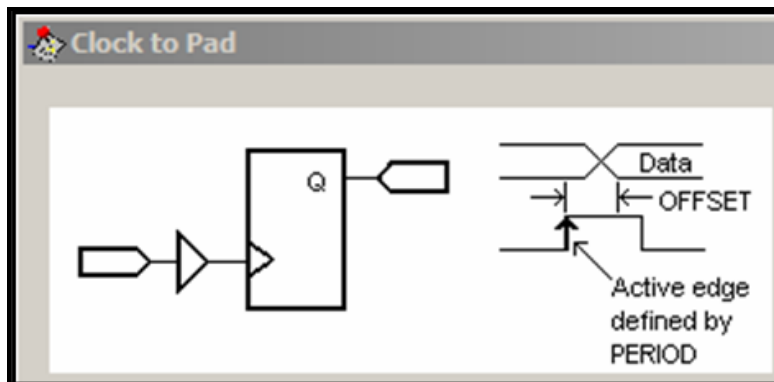
*OFFSET = IN 5 ns BEFORE "MYCLK" ;*

- “The signal will be valid at the pad X nanoseconds before the clock appears at the clock pad...”
- Covers the first path to a synchronous element
- Recommendation for high performance:
  - Use the IOB registers



# OFFSET OUT AFTER Constraint

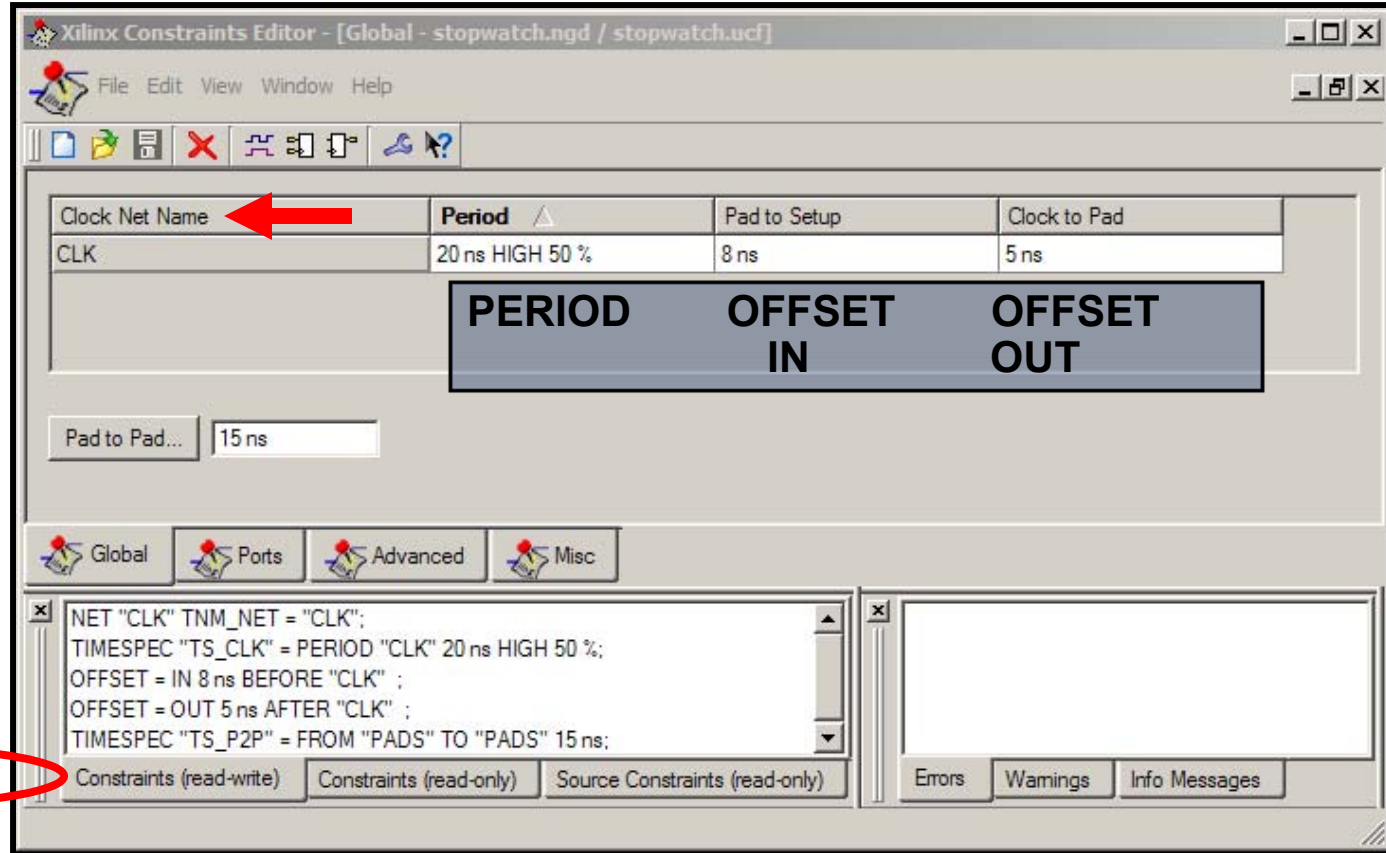
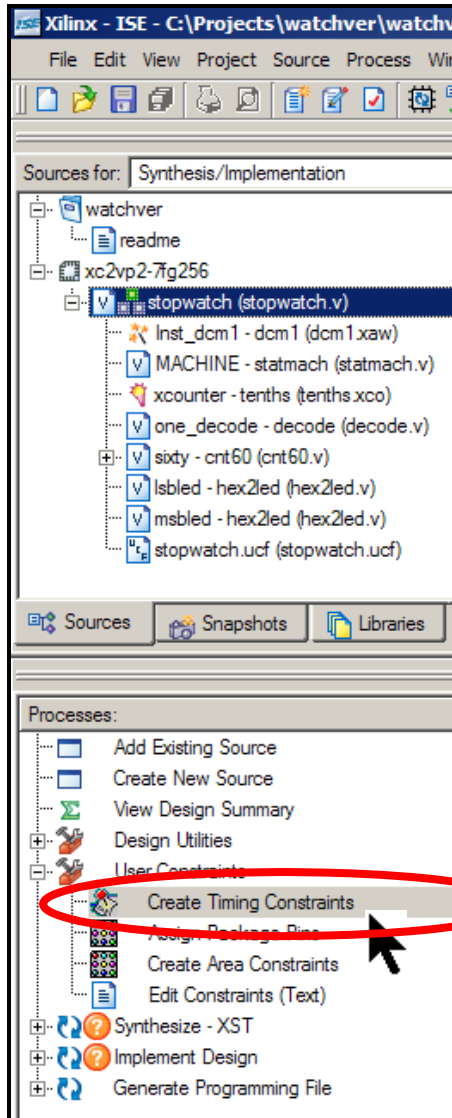
- **OFFSET OUT AFTER**
  - OFFSET = OUT 7 ns AFTER " MYCLK " ;*
  - “The signal will be valid at the pad X nanoseconds after the clock appears at the clock pad...”
  - Covers the last path from a synchronous element
  - **Recommendation for high performance:**
    - Use the IOB registers



# UCF – User Constraints File

```
#  
#Global Clock Constraint  
# constrain net on external pin  
NET "CLK" TNM_NET = "CLK":  
TIMESPEC "TS_CLK" = PERIOD "CLK" 20 ns HIGH 50%  
#  
#Input Timing  
OFFSET = IN 8 ns BEFORE "CLK" ;  
#  
#Output Timing  
OFFSET = OUT 5 ns AFTER "CLK" ;  
#  
#Pad-pad combinatorial timing  
TIMESPEC "TS_P2P" = FROM "PADS" TO "PADS" 15 ns;  
#  
# Input timing exception from global input constraint  
NET "STRTSTOP" OFFSET = IN 3 ns BEFORE "CLK" ;
```

# Timing Constraints Editor



# Timing Analysis

