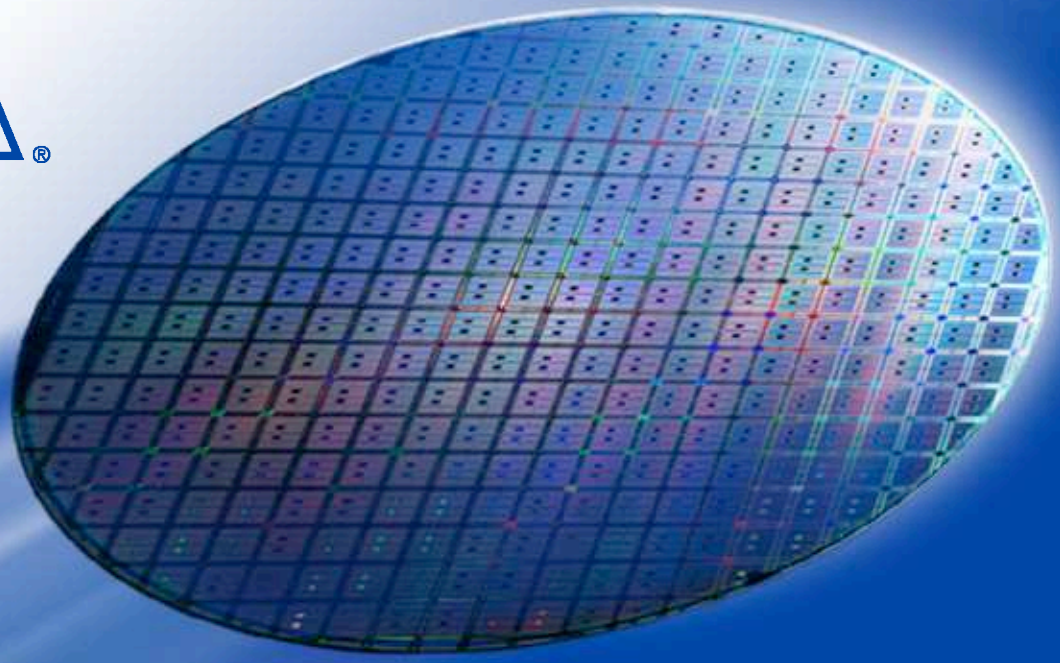


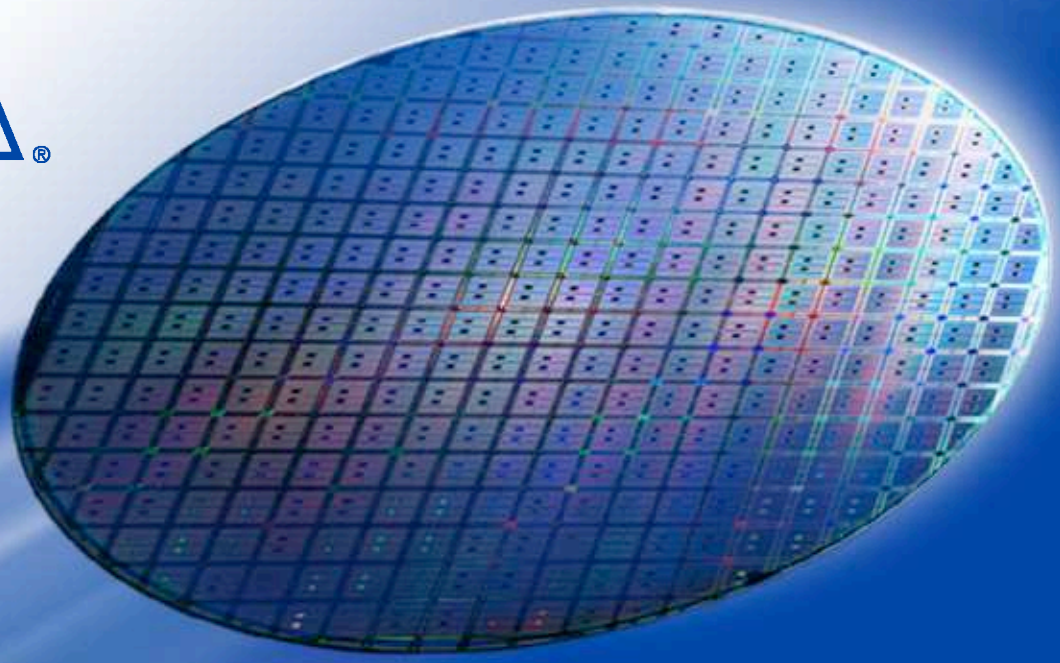
The Nios II Family of Configurable Soft-core Processors

James Ball

August 16, 2005

Agenda

- Nios II Introduction
  - Configuring your CPU
- FPGA vs. ASIC CPU Design
  - Instruction Set Architecture
  - CPU Micro-architecture
- Nios II/f CPU Description
  - Pipeline details
- Nios II Embedded Systems
  - Taking advantage of FPGA configurability



Nios II Introduction

Nios II Overview

- Nios II is Altera's soft-core configurable CPU
  - Introduced summer/2004
  - New 32-bit RISC Instruction Set Architecture (ISA)
  - Replaces original 16-bit Nios
- Over 4500 active licenses
  - Most licensed embedded CPU in the world
- Designed for embedded FPGA-based systems
  - Strong performance (up to 225 Dhrystone MIPS)
  - Support for many operating systems
  - Available in all current Altera FPGAs

Why a New Instruction Set?

- Primary Issue
  - Existing instruction sets optimized for ASIC
  - Inefficient in FPGA
- Secondary Issue
  - Existing instruction sets have licensing restrictions

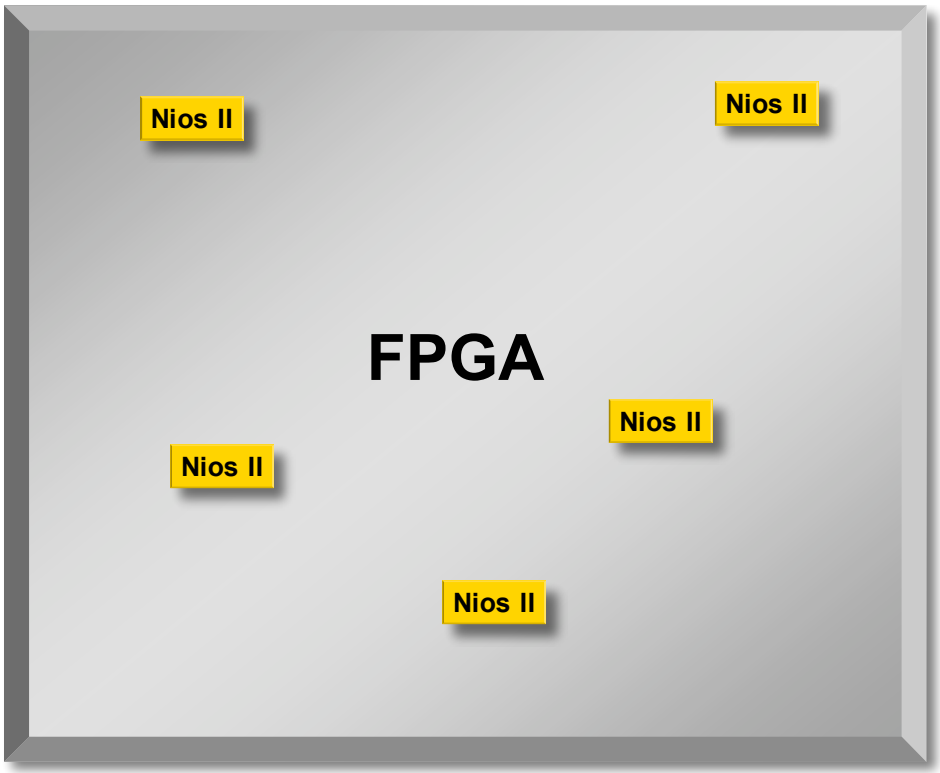
Nios II Size

Smallest 90nm FPGA
4600 LUTs

13% of FPGA
Nios II/e "economy"

35¢ in lowest cost FPGA

Largest 90nm FPGA
180,000 LUTs

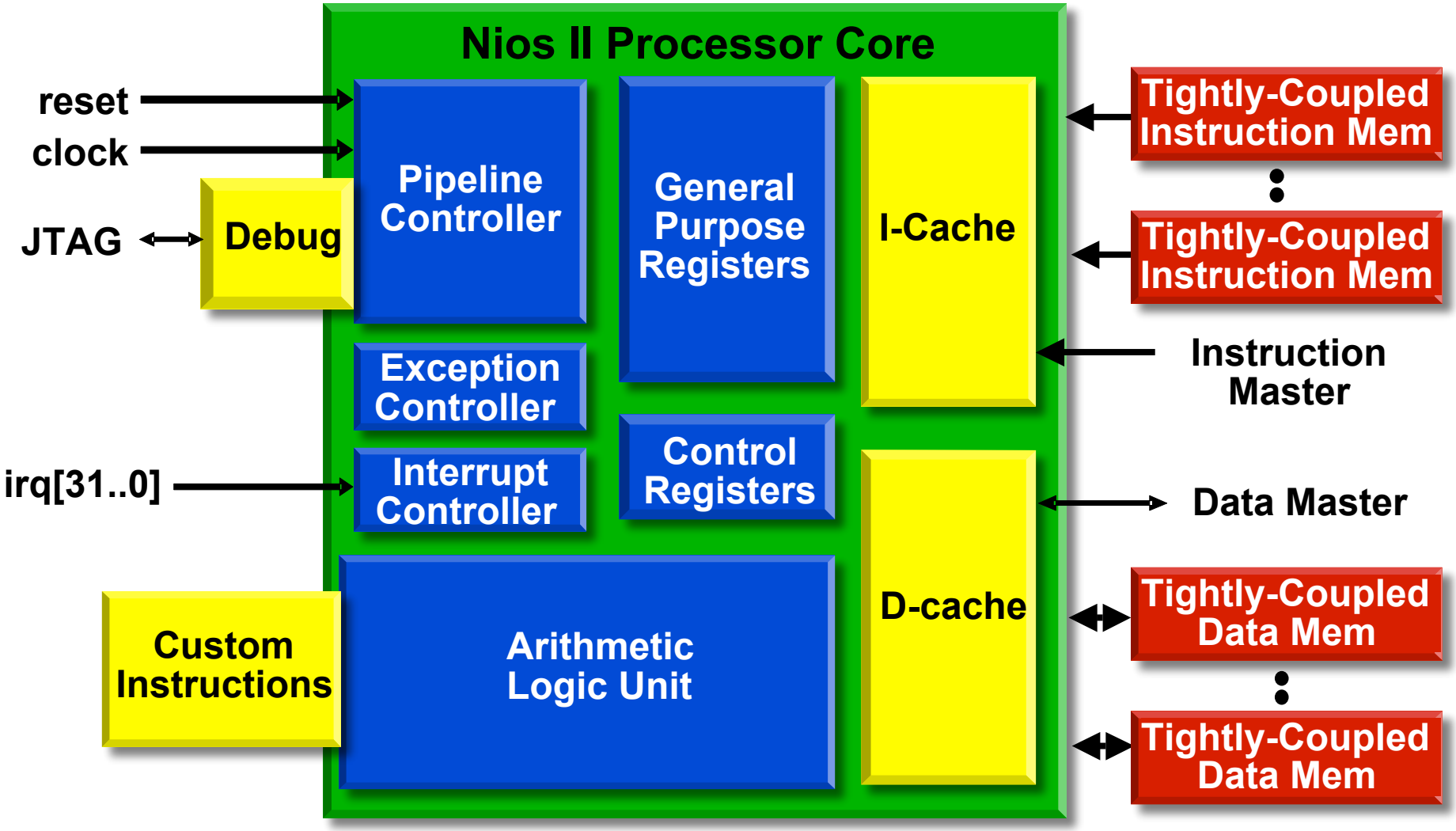


1% of FPGA
Nios II/f "fast"

Nios II is Classic RISC

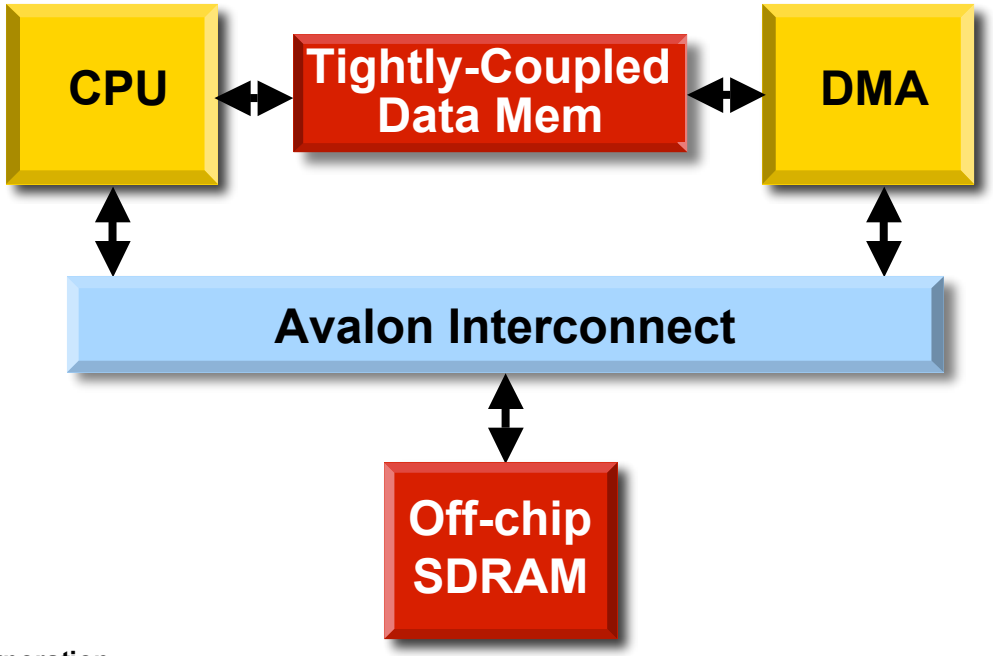
- 32-Bit Instruction Set
- 32-Bit Data path
- 32 General-Purpose Registers
- 3 Instruction Formats
- 82 Instructions
  - Instruction set is not configurable
  - Provides code compatibility for all implementations
- Up to 256 Custom Instructions
- 3 Operand Instructions (2 source, 1 destination)
- Optional Multiply and Divide

Nios II Processor Block Diagram



Configurable Tightly Coupled Memories

- Map on-chip RAMs into CPU address space
  - Behave like caches that never miss
  - One access every cycle without stalling
- FPGA RAMs are already dual-ported
  - One port for Nios II connection
  - Second port available for other uses



Configurable CPU Implementation

- Choose your pipeline

Nios® II	*Nios II/f* *"Fast"*	*Nios II/s* *"Standard"*	*Nios II/e* *"Economy"*
Pipeline	**6-stage**	**5-stage**	**none**
Max Frequency[1]	**200 MHz**	**180 MHz**	**210 MHz**
Max D-MIPS[1]	**225**	**130**	**30**
Size (4-input LUTs)	**1800**	**1200**	**600**
Branch Prediction	**Dynamic**	**Static**	**no**
I-Cache	**Up to 64K**	**Up to 64K**	**no**
D-Cache	**Up to 64K**	**no**	**no**

1. Characteristics in Stratix II 90nm FPGA

Configurable Pipeline Options

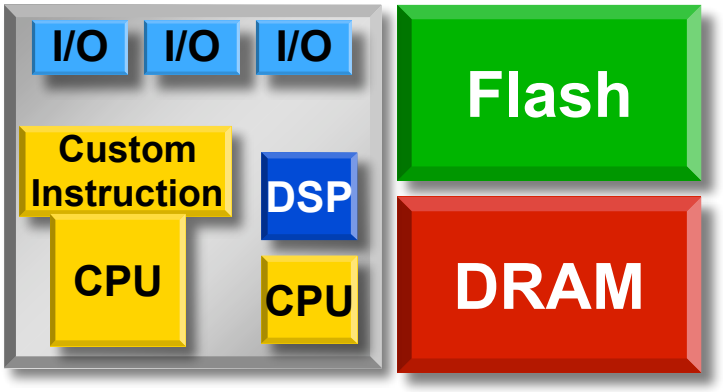
- Cache options
  - Size
  - Line size
- Multiply instruction options
  - Fully pipelined using built-in FPGA multipliers
  - Un-pipelined using normal LUT logic
  - Trap (software emulated)
- Divide instruction options
  - Un-pipelined using normal LUT logic
  - Trap (software emulated)

Configurable Custom Instructions

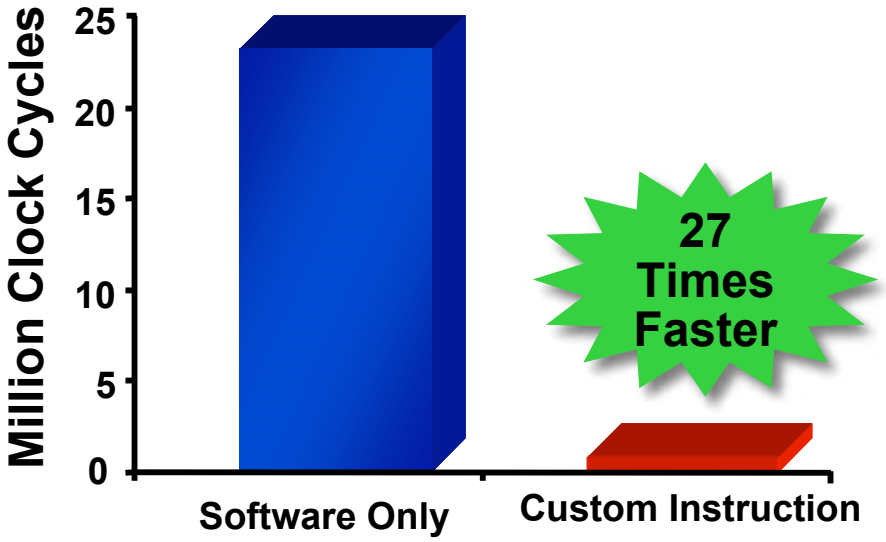
- Users write Verilog/VHDL for custom instructions
  - Added to CPU with automatic configuration tool
  - Callable from C-code or assembly language
- Pipeline independent
- 2 source operands and 1 destination operand
  - Access CPU register file
  - Access custom instruction register file
- Combinatorial custom instructions
  - Execute in parallel with ALU
- Multi-cycle custom instructions
  - Stall CPU pipeline until complete

Configuring for Higher Performance

Add Custom Instructions

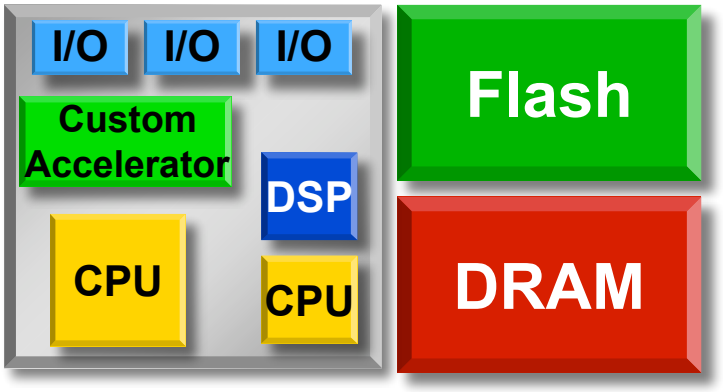


Example:
64 Kbyte CRC

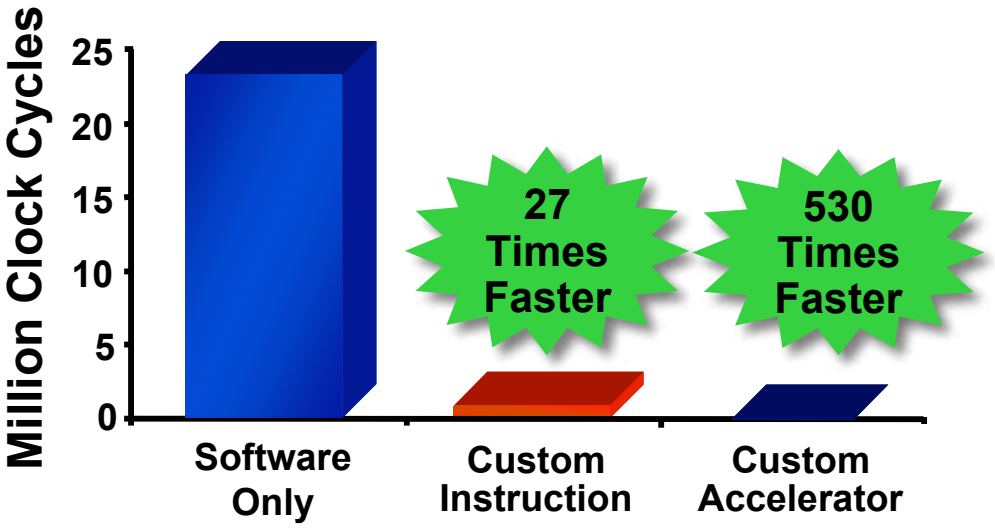


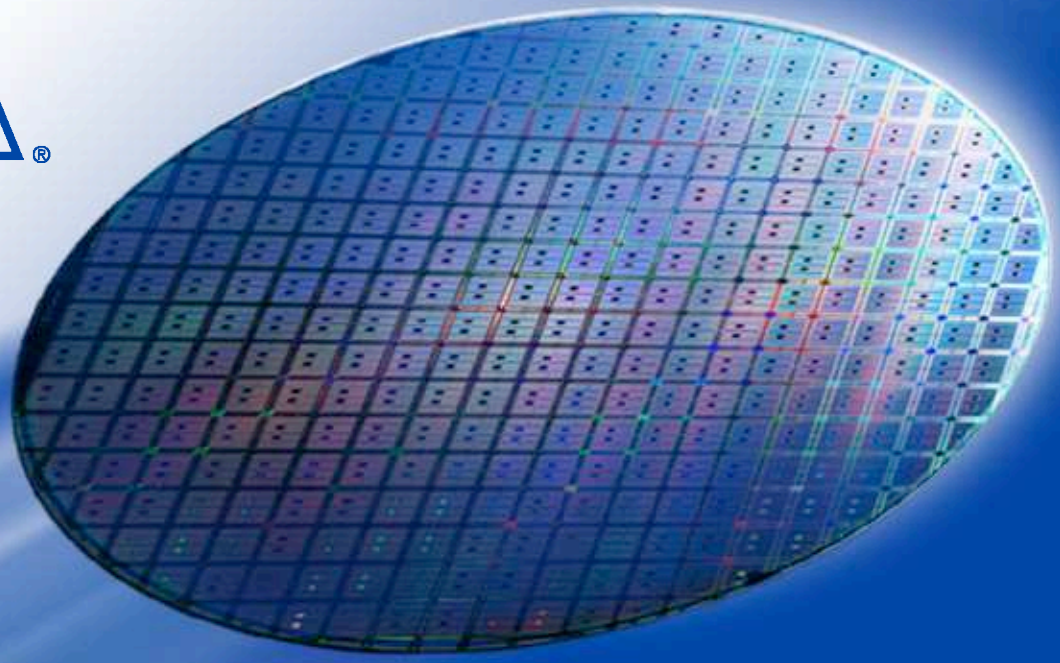
Configuring for Higher Performance

Add Custom Accelerator



Example:
64 Kbyte CRC





FPGA vs. ASIC CPU Design

Efficient FPGA Design Guidelines

- RAMs, adders, registers, and multipliers
  - Relatively fast and plentiful
  - RAMs are already dual-ported
- Muxing and control logic
  - Relatively slow and expensive
- Wire delays
  - Relatively long
- Take advantage of FPGA configurability
  - Minimize run-time control registers
  - Rely on configuration-time options

Existing ISAs are Inefficient in FPGAs

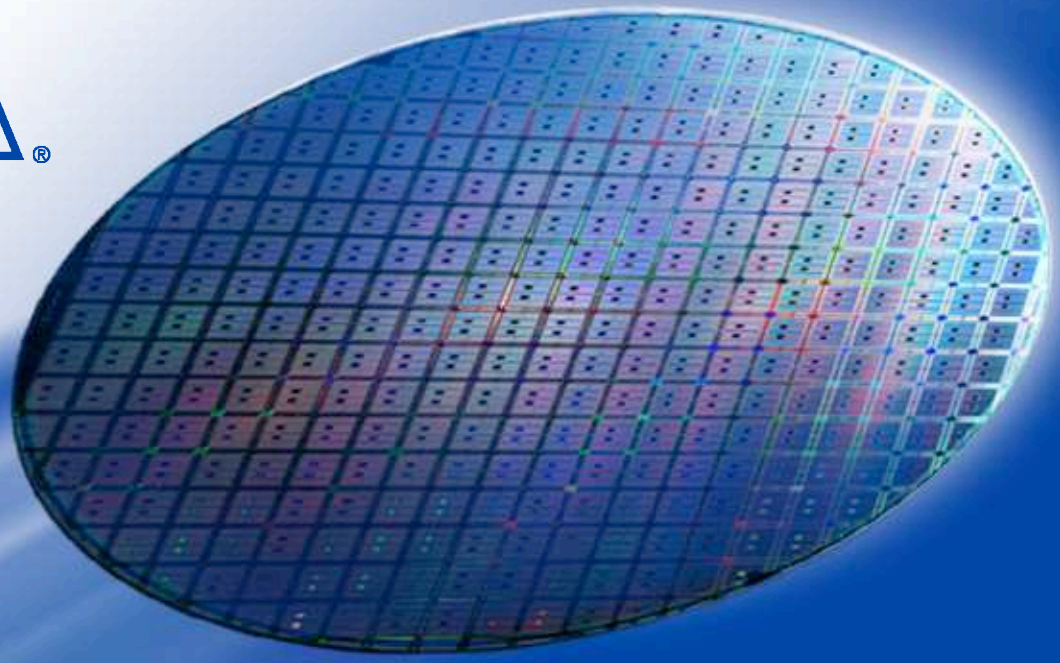
- Variable-length instructions or 16-bit instructions
  - Higher code density not worth extra control logic
- Register windows
  - Lower memory bandwidth not worth extra control logic
  - Can create difficult real-time requirements
- Barrel shifts combined with other arithmetic operations
  - Barrel shifts are relatively slow on FPGAs due to muxing
- Delay slots
  - Decreased branch penalty not worth extra control logic
  - Unnatural for some pipelines

Existing ISAs are Inefficient in FPGAs

- Condition code register
  - Complicates pipeline control and increases muxing
- Multiply/divide 64-bit operand registers
  - All 64-bits rarely used in C language and increases muxing
- Many run-time control registers
  - Extra logic not required in a configurable FPGA CPU
- Complex cache management
  - State machines to initialize on reset not worth extra logic
  - Many instruction options for flushing not worth extra logic
- Vectored interrupts
  - Not required for most designs
  - Use custom instruction to reduce interrupt latency

Getting Back to RISC Roots

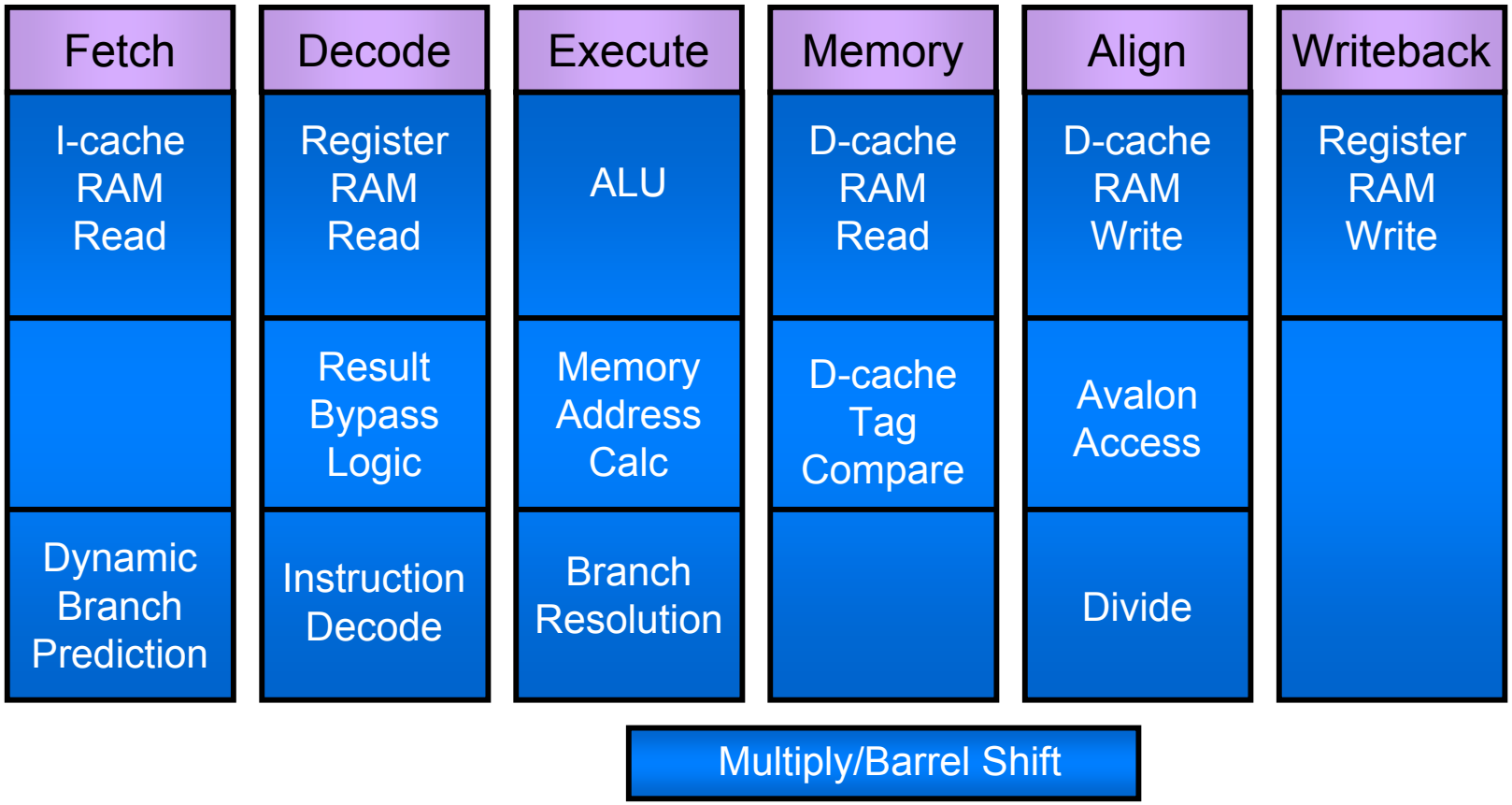
- CPU is an engine to run C code
  - Benchmarking shows Nios II has comparable performance to established embedded CPUs
- To increase CPU performance in an FPGA
  - Increase the Nios II cache size
  - Add Nios II custom instructions
  - Add custom accelerators
  - Add multiple Nios II CPUs
  - Add tightly-coupled memories



Nios II/f CPU Description

"Fast"

Nios II/f Pipeline



Caches

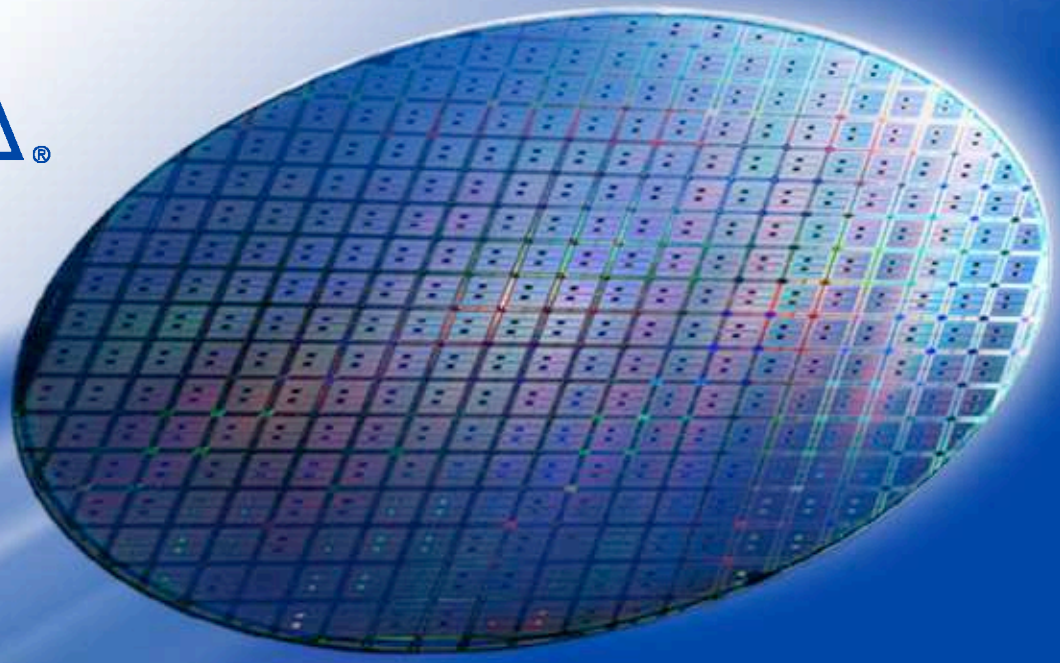
- Direct-mapped
  - Set-associative caches inefficient in FPGA
- I-cache
  - 32-byte line
  - Critical word first
- D-cache
  - 4/16/32-byte line
  - Writeback with write allocate
  - One entry writeback buffer

Dynamic Branch Prediction

- 2-bit branch prediction (g-Share algorithm)
  - Branch History Table RAM (256x2 bits)
- No Branch Target Buffer
  - Simple ISA allows fast branch target calculation
- Performance
  - Taken branch is 2 cycles
  - Not taken branch is 1 cycle
  - Mispredicted branch penalty is 4 cycles

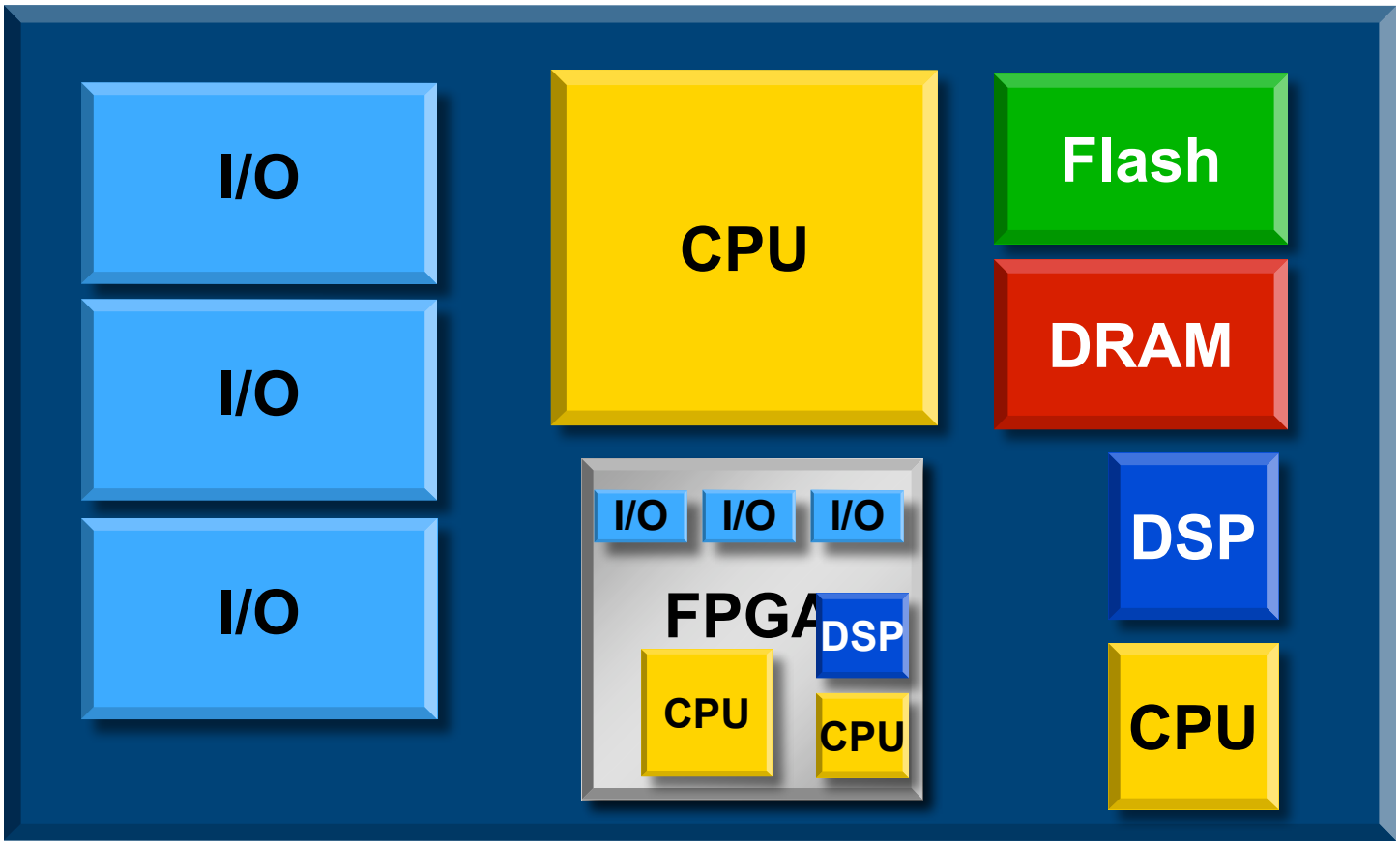
Arithmetic Instructions

- 32-bit Multiply
  - 1 cycle throughput (fully pipelined)
- 32-bit Divide
  - 4-67 cycle throughput (not pipelined)
- Barrel shift/rotate
  - Uses multiplier with 2^n calculation
  - Better performance and lower cost than using LUTs



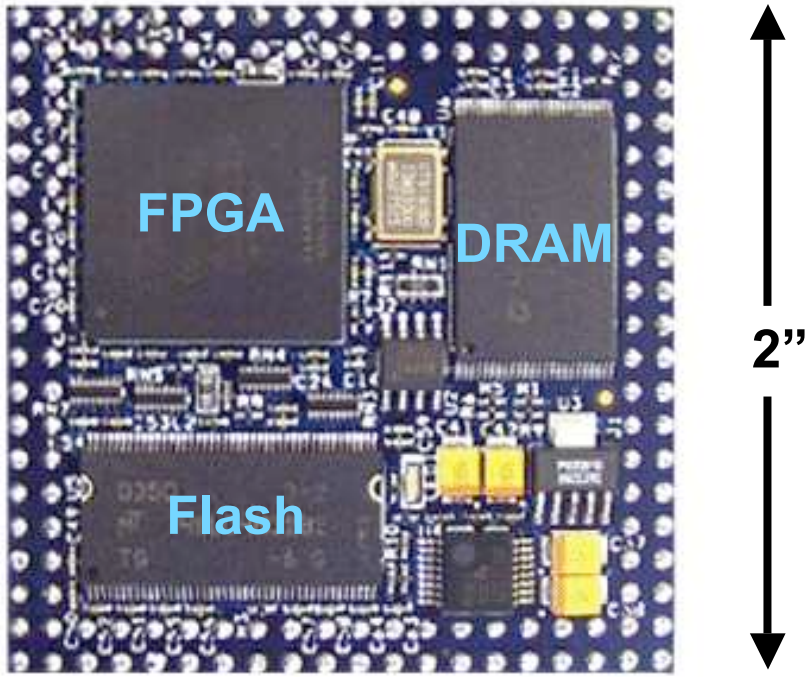
Nios II Embedded Systems

Board-based Embedded System
FPGA-based Embedded System



Move board components into FPGA

Nios II Evaluation Board

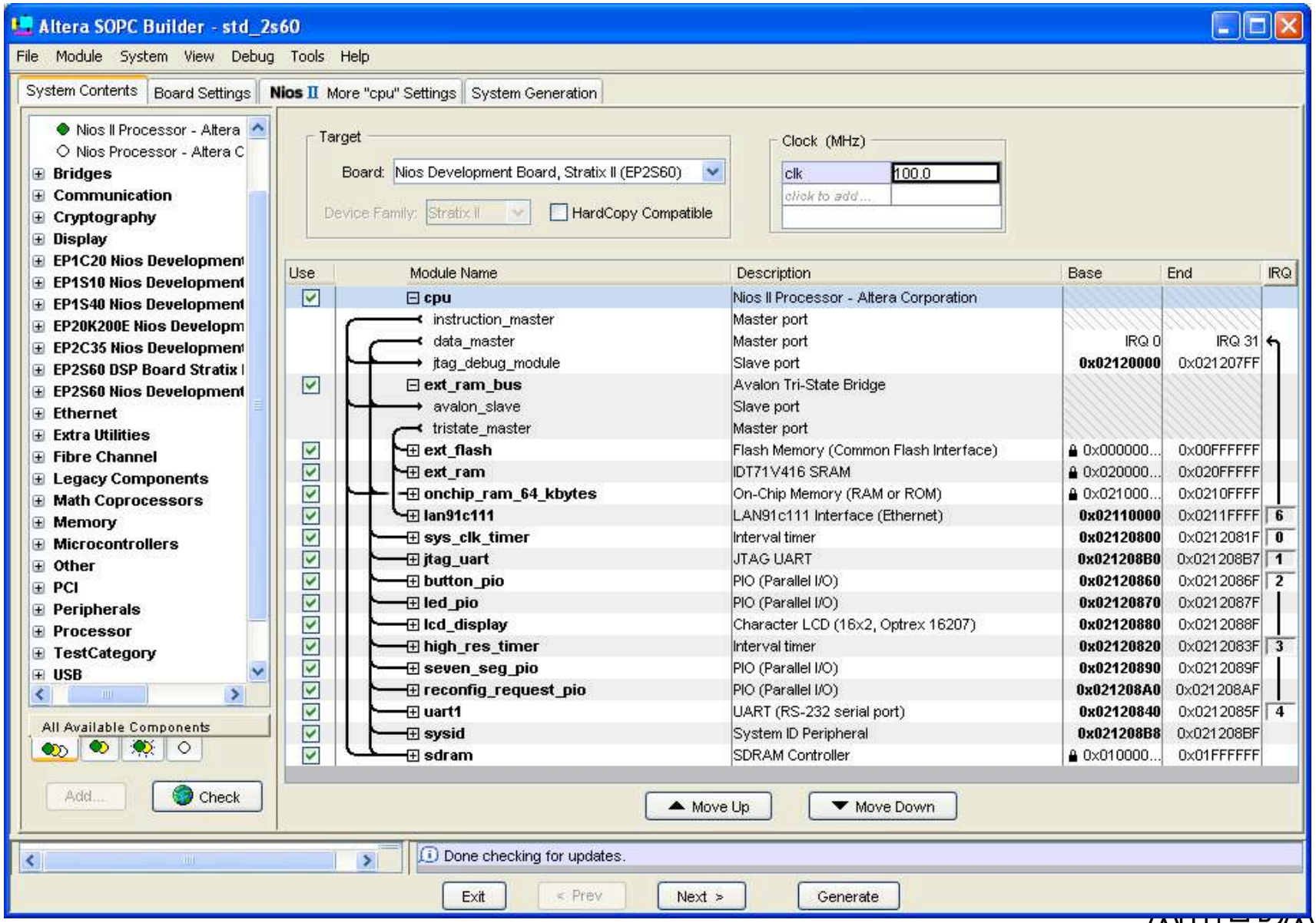


Preconfigured with a web server running under µClinux

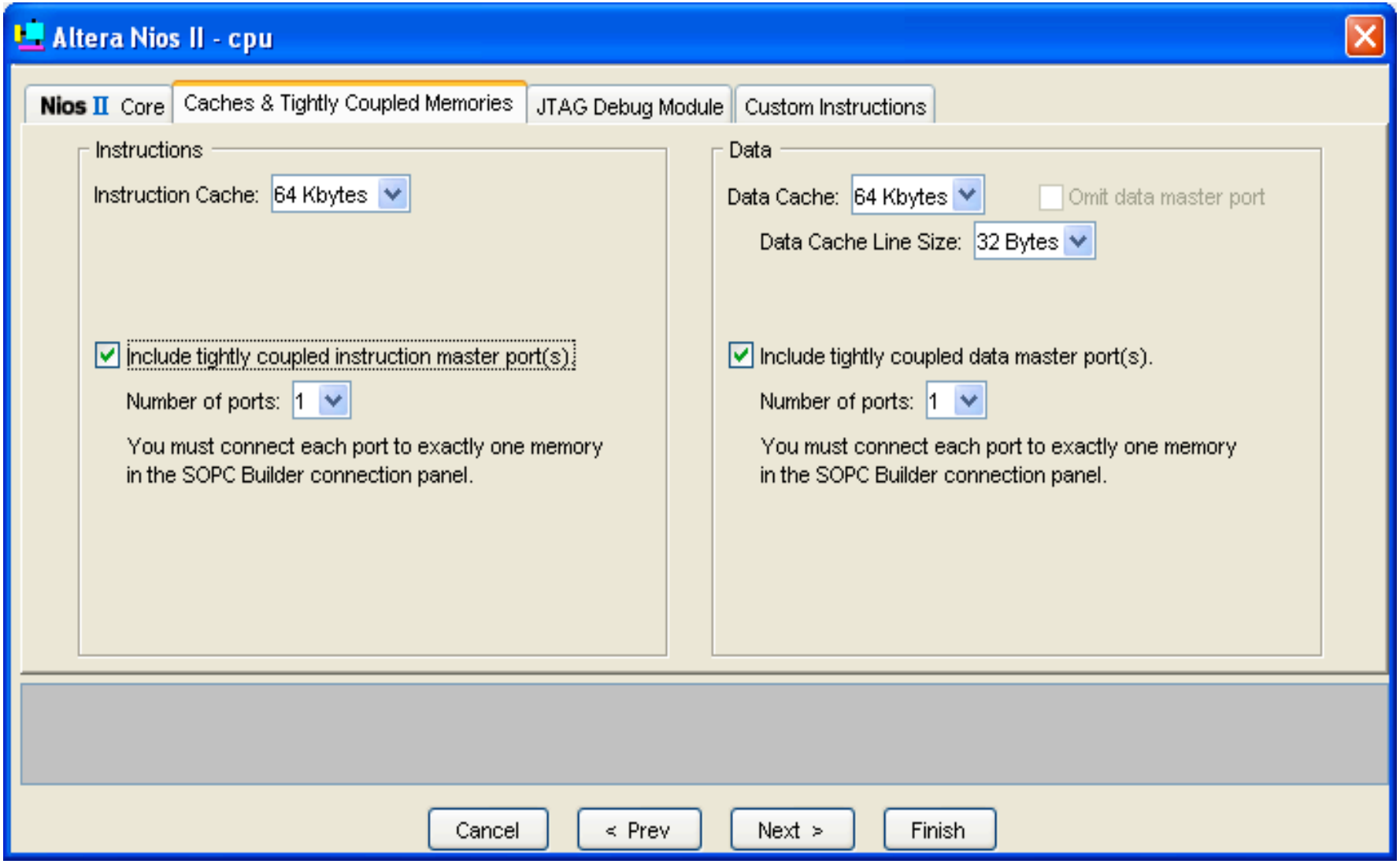
FPGA-based Systems

- It's all configurable
  - Configurable CPUs
  - Configurable Memories (on-chip and off-chip)
  - Configurable Peripherals
  - Configurable I/O
  - Configurable System Interconnect
  - Custom Accelerators
- and we provide the tools to make it easy …

System Configuration Tool



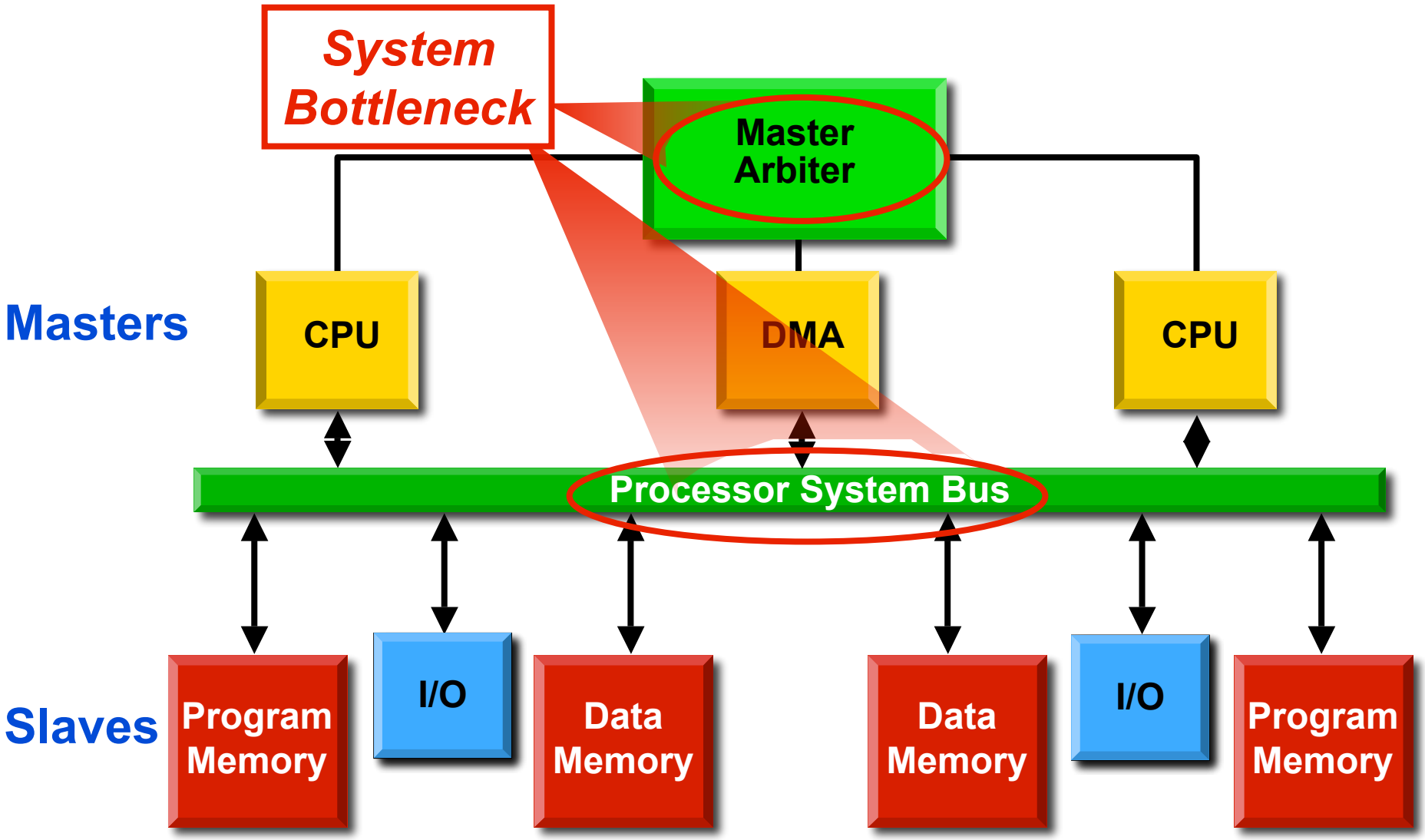
CPU Configuration Tool



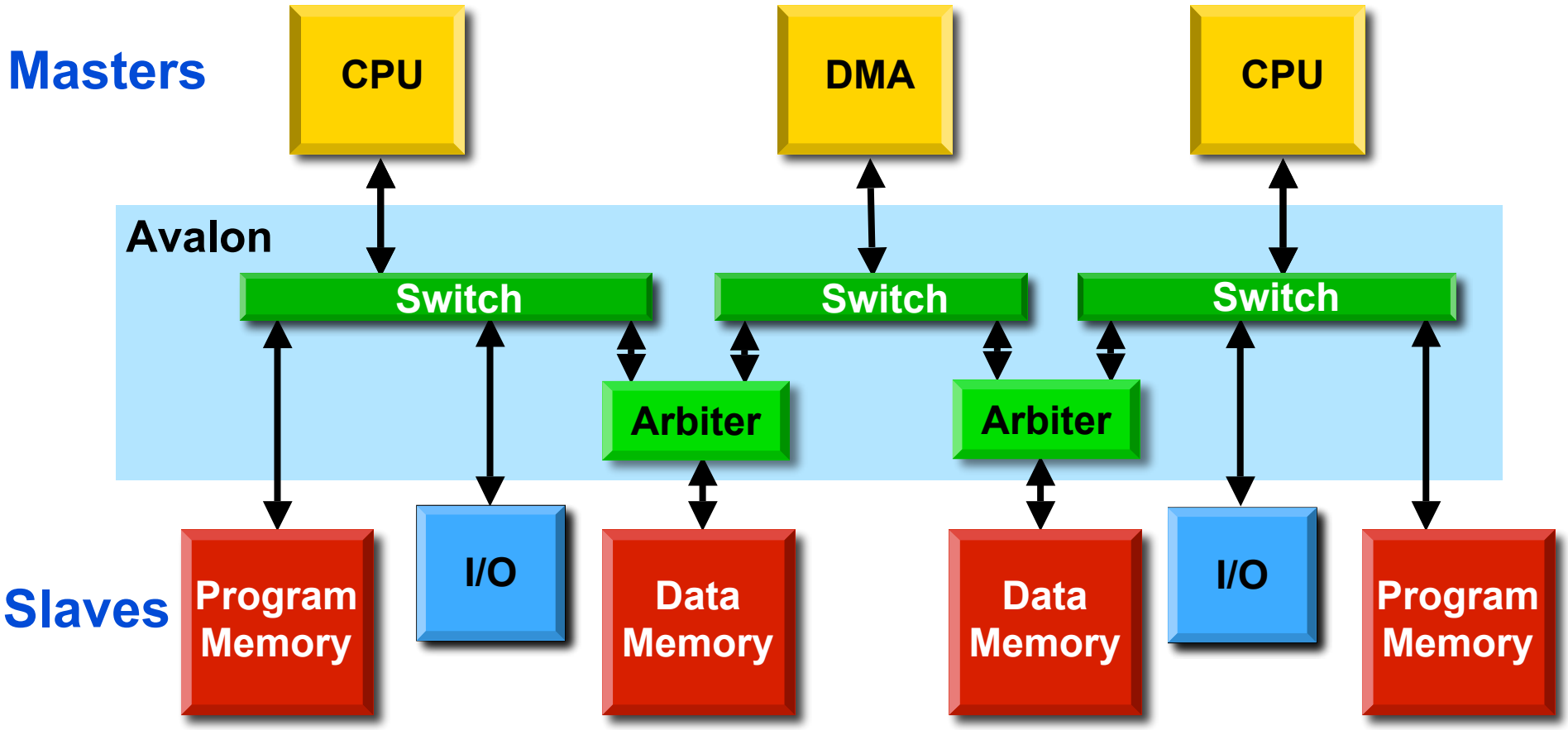
Avalon System Interconnect

- Automatically generated for your system
- Switches connect components – not a bus
- Slave side arbitration
  - Enables concurrent accesses
- Avalon Functions
  - Arbitration
  - Multiplexing
  - Address Decoding
  - Wait-State Generation
  - Dynamic Bus Sizing

Traditional Bus Interconnect

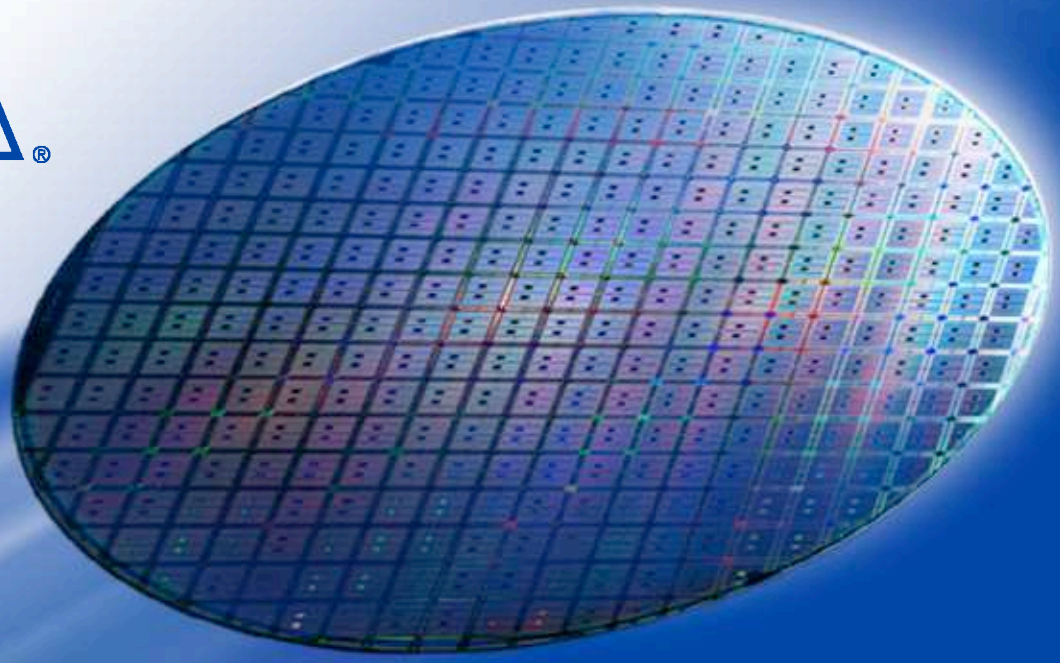


Avalon Switch Interconnect



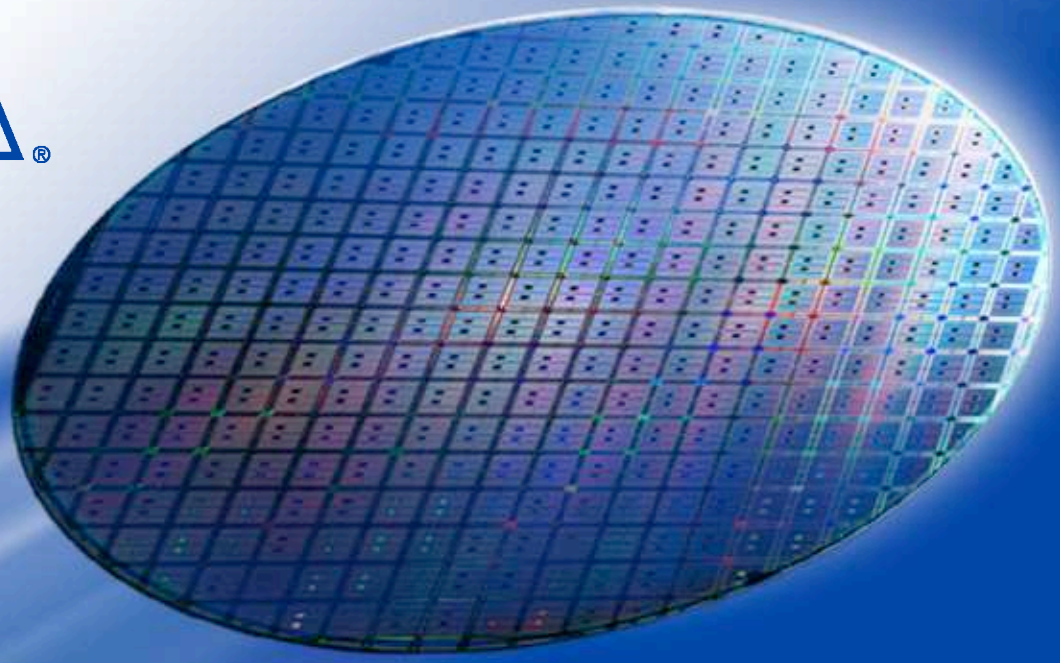
Conclusions

- Efficient FPGA design takes advantage of configurable CPUs and systems
- Nios II is optimized for FPGA-based systems
- Established CPUs based on ISAs optimized for ASICs are less efficient in FPGAs

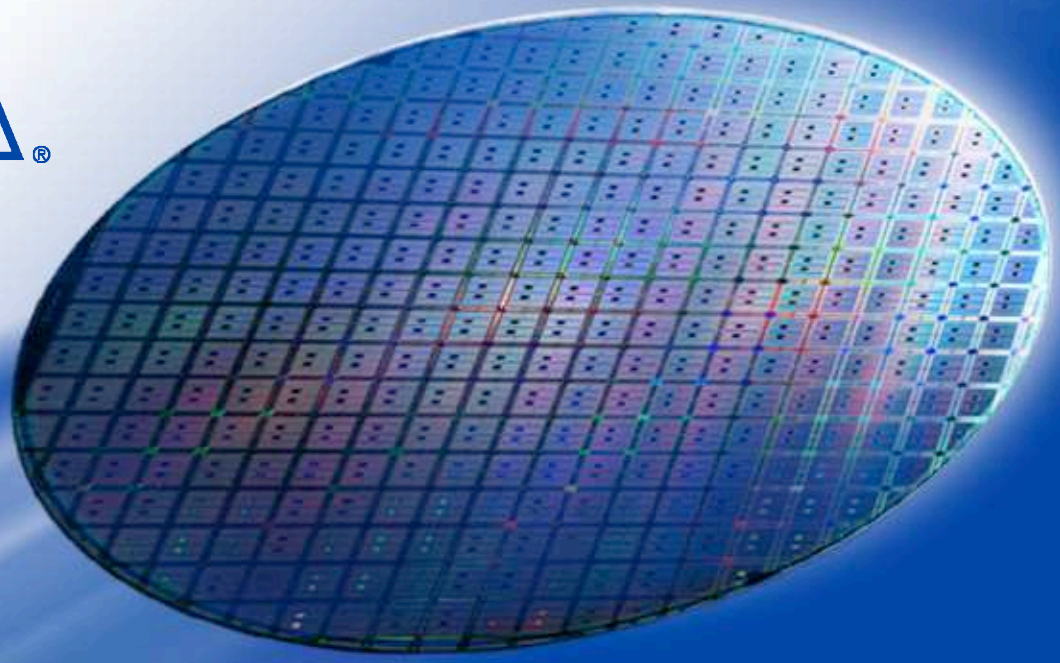


The End

Questions?



Backup Slides



Why a Soft-Core FPGA CPU?

FPGA Soft-Core CPU Advantages

- Flexibility
  - Utilize existing silicon resources
- Scalability
  - Number of CPUs, CPU types, cache sizes, etc.
- Configurability
  - Generation-time configuration instead of run-time
  - Eliminates logic required to control CPU options
- Ubiquity
  - Available in all FPGA families

FPGA Soft-Core CPU Advantages

- Relatively small compared to FPGA capacities
  - Largest Altera FPGA fits 300 Nios II/e CPUs
  - May have spare capacity so CPU is free
- Lifecycle
  - No obsolescence
  - New releases of CPU improve your design
  - Improved efficiency with latest silicon technologies

Altera's Latest FPGA Devices

	Stratix II	*Cyclone II*
Technology	90 nm	90 nm
4-input LUTs	180,000	70,000
18-bit Multipliers	384	180
On-chip RAM	1.2 Mbytes	144 Kbytes