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

Largest 90nm FPGA
180,000 LUTs

Smallest 90nm FPGA
4600 LUTs

13% of FPGA
Nios II/e “economy”

35¢ in lowest
cost FPGA

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
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

<table>
<thead>
<tr>
<th></th>
<th>Nios II/f “Fast”</th>
<th>Nios II/s “Standard”</th>
<th>Nios II/e “Economy”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline</td>
<td>6-stage</td>
<td>5-stage</td>
<td>none</td>
</tr>
<tr>
<td>Max Frequency</td>
<td>200 MHz</td>
<td>180 MHz</td>
<td>210 MHz</td>
</tr>
<tr>
<td>Max D-MIPS</td>
<td>225</td>
<td>130</td>
<td>30</td>
</tr>
<tr>
<td>Size (4-input LUTs)</td>
<td>1800</td>
<td>1200</td>
<td>600</td>
</tr>
<tr>
<td>Branch Prediction</td>
<td>Dynamic</td>
<td>Static</td>
<td>no</td>
</tr>
<tr>
<td>I-Cache</td>
<td>Up to 64K</td>
<td>Up to 64K</td>
<td>no</td>
</tr>
<tr>
<td>D-Cache</td>
<td>Up to 64K</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

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

27 Times Faster
Configuring for Higher Performance

Add Custom Accelerator

Example:
64 Kbyte CRC

Software Only

Custom Instruction

Custom Accelerator

27 Times Faster

530 Times Faster
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 Pipeline

Fetch
- I-cache RAM Read
- Dynamic Branch Prediction

Decode
- Register RAM Read
- Instruction Decode

Execute
- ALU
- Branch Resolution

Memory
- D-cache RAM Read
- D-cache Tag Compare

Align
- D-cache RAM Write
- Avalon Access

Writeback
- Register RAM Write

Multiply/Barrel Shift
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
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

In the CPU Configuration Tool, you can configure the instruction cache size and data cache size. The instruction cache size is set to 64 Kbytes, and the data cache size is also set to 64 Kbytes. Additionally, you can choose to include tightly coupled instruction and data master port(s). The number of ports for both instruction and data ports is set to 1. You must connect each port to exactly one memory in the SOPC Builder connection panel.
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

System Bottleneck

Masters
- CPU
- DMA
- CPU

Slaves
- Program Memory
- I/O
- Data Memory
- Data Memory
- I/O
- Program Memory

Processor System Bus

Master Arbiter
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?
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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Stratix II</th>
<th>Cyclone II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>90 nm</td>
<td>90 nm</td>
</tr>
<tr>
<td>4-input LUTs</td>
<td>180,000</td>
<td>70,000</td>
</tr>
<tr>
<td>18-bit Multipliers</td>
<td>384</td>
<td>180</td>
</tr>
<tr>
<td>On-chip RAM</td>
<td>1.2 Mbytes</td>
<td>144 Kbytes</td>
</tr>
</tbody>
</table>