20 YEARS of ALTERA INNOVATION

Designing with Nios and SOPC Builder

Nios 3.1 & Quartus 3.0

Agenda

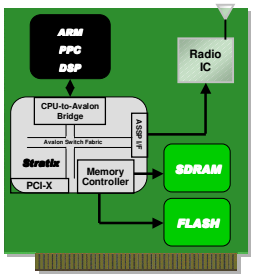
- Applications
- Nios® Hardware Development
- Nios Software Development
- Nios Software Debug
- RTL Simulation
- Avalon Bus
  - User Peripherals
- Custom Instructions
- Multi-Masters and Direct Memory Access (DMA)
- Configuring the Development Board

20 YEARS of ALTERA INNOVATION

SOPC & Nios Applications

Bridging

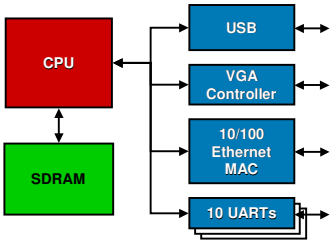
- Peripherals Don't Have to Be Inside the FPGA
- Many Examples Exist
  - Avalon-to-AHB
  - AHB-to-Avalon
  - Tri-State Bridge
- Create Your Own
  - ASSP-to-Avalon Bridge
  - Avalon-to-ASSP Bridge
  - Processor-to-Avalon Bridge
  - Publish as SOPC Builder Components
    - Design Re-Use



Extends the Scope to the Design Beyond the FPGA

Custom Microcontroller

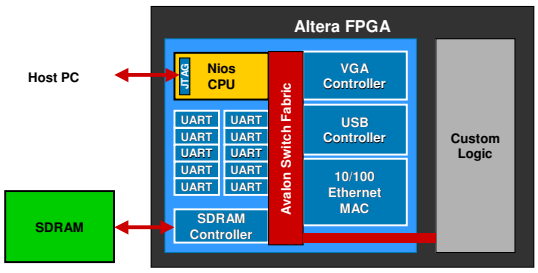
before Nios Processor



Multiple Discrete Off-the-Shelf Devices Vulnerable to Obsolescence

Custom Microcontroller

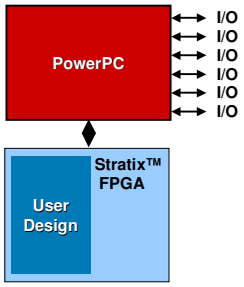
after Nios Processor



System-on-a-Programmable-Chip Solution Architecture Never Goes Obsolete

Offload Existing CPU

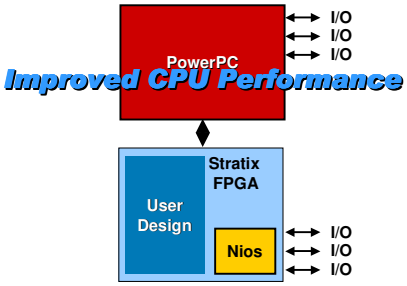
before Nios Processor



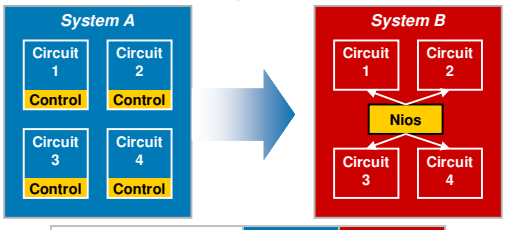
CPU Performance Limited by I/O Processing

Offload Existing CPU

after Nios Processor

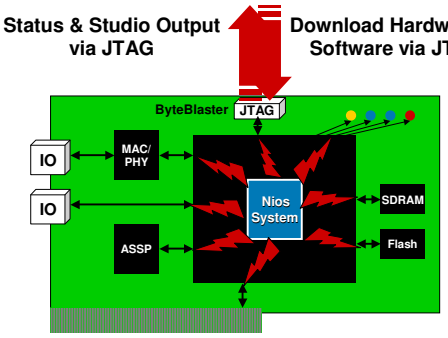


State Machine Replacement

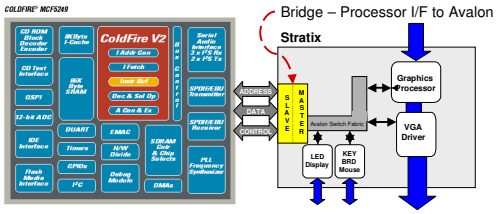


System	A	B
Development Complexity	High	Lower
Development Time	High	Lower
Resource Utilization	High	Lower
Design Flexibility	Low	Higher

Board Bring-Up & System Test



Companion Chip

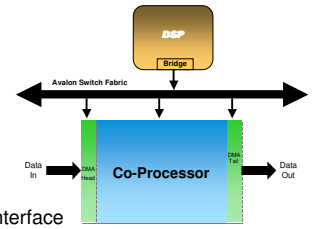


Motorola ColdFire Processor
SDRAM, SRAM, FLASH
12-bit ADC, IDE I/F, Timers, DUART, I2C

Add to a Companion Chip with Custom VGA Controller with Graphics Processor, and Accessory Keyboard / Mouse and LED Display

Co-Processor

- Hardware Accelerator
  - User's Custom Logic
  - Off-The-Shelf IP
    - FIR Filter
    - FFT
    - Image Processing
    - Communications
- DMA Input & Output Interface
  - Facilitates Data Transfers without CPU Intervention
- Easily Designed Using Altera's SOPC Builder Software



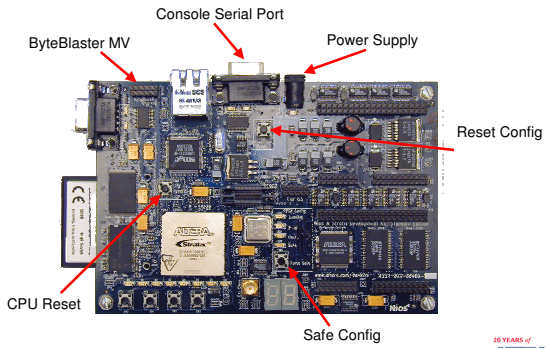
20 YEARS of ALTERA INNOVATION

Nios Hardware Development

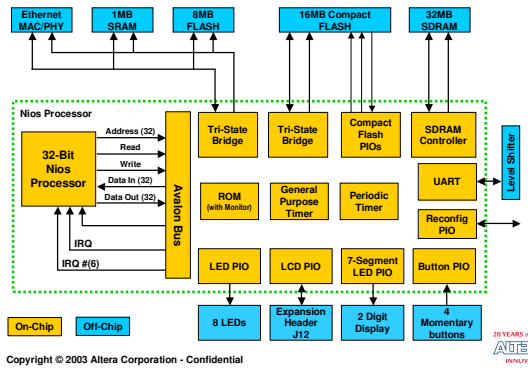
What Is Nios?

- Nios Is A Soft-core 16 or 32 Bit RISC Microprocessor
  - Developed Internally By Altera
  - Harvard Architecture
  - License & Royalty Free
- Nios Plus All Peripherals Written In HDL
  - Can Be Targeted For All Altera FPGAs
  - Synthesis Using Quartus II Integrated Synthesis
- Software Development Using GNUPro Tools
  - Open Source Software Development
- SOPC Builder (System on a Programmable Chip)
  - Graphical Interface That Makes It Easy To Create

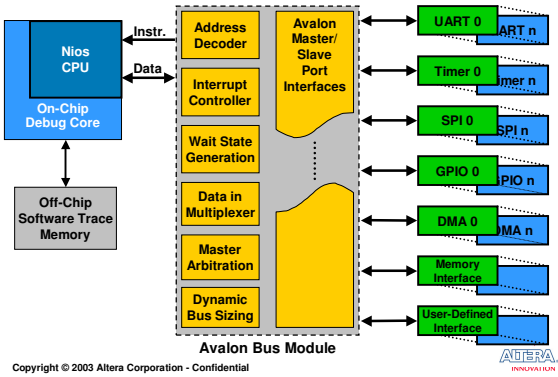
Nios Dev Kits, Stratix & Cyclone



Standard_32 Design Block Diagram

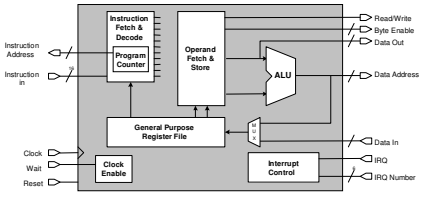


Nios System Architecture



Nios Processor Characteristics

- Pipelined RISC Architecture
- 16-Bit Instructions
- Large Internal Register File
- 32-Bit or 16-Bit Configurable Data Path
- 64 Prioritized Interrupts
- Optional Instruction & Data Cache
- Harvard Architecture
  - Instruction & Data Ports Based on Simultaneous Multi-Master Avalon Bus
- Optional Performance Optimization Features
- Custom Instructions



IRQ: Interrupt Request
ALU: Arithmetic Logic Unit

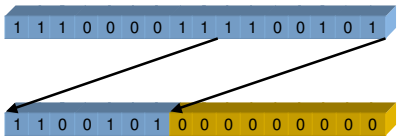
Multiplier Hardware Acceleration

- Software Multiply Option - No Multiply Hardware
  - Uses GNUPro Math Library to Implement Multiplier
- MSTEP Multiply Option - Hardware-Assisted Multiply
  - One-Bit per Clock Cycle Multiply (~4x Improvement over Software)
- MUL Multiply Option - Full Hardware Multiplier
  - 16 x 16 ➔ 32 in 3 Clock Cycles – ***1 Clock with Stratix DSP Block!***

Acceleration Hardware	Additional Logic	Clock Cycles (16 x 16 ➔ 32)	Clock Cycles (32 x 32 ➔ 32)
None	0 LEs	80	250
MSTEP	+20 LEs	18	80
MUL	+450 Les	3	30
MUL in Stratix	+85 LEs +2 9x9 DSP Elements	1	20

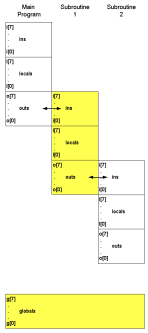
Bit Shift Hardware Acceleration

- Provides Multi-Bit Shift in Single Instruction Cycle
  - Shift Left or Right by 1-to-31 Bits per Instruction Cycle
  - All Nios 2.0 CPUs Include Full 31-Bit Shift Hardware Acceleration
- Example: `i << 9;  /* Shift Left by 9 Bits */`
  - Executes in One Cycle
  - Would Execute in Nine Clock Cycles without Acceleration

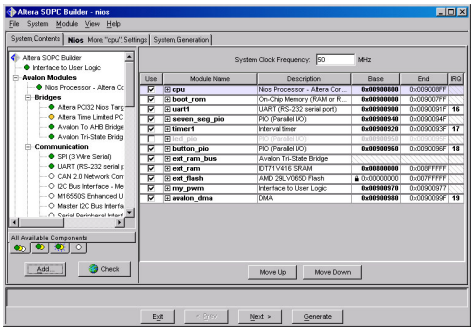


Windowed Register File

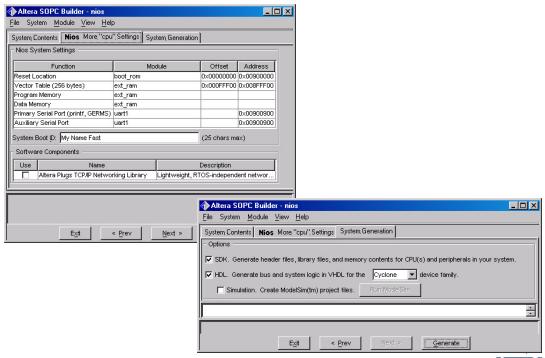
- Common Technique Used by High-Performance CPUs
  - Enables Fast Subroutine Calls
- Up to 512 General-Purpose Registers
- Sliding Window Allows Access to 32 Registers at a Time
  - 24-Register Sliding Window
  - Eight Global Registers
- Used Automatically by C Compiler
- MFLAT Compiler Option
  - Disables Sliding Window
  - Makes Register Space Static, or "Flat"
  - Guarantees Maximum Context Switch Time



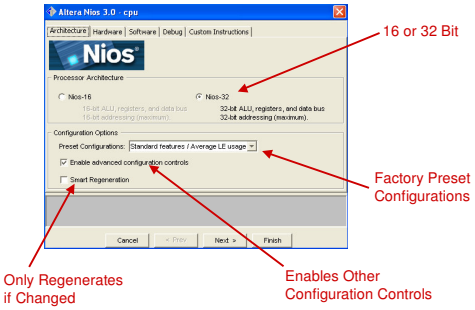
SOPC Builder



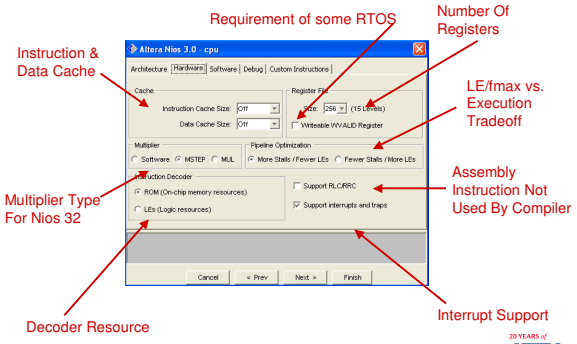
SOPC Builder



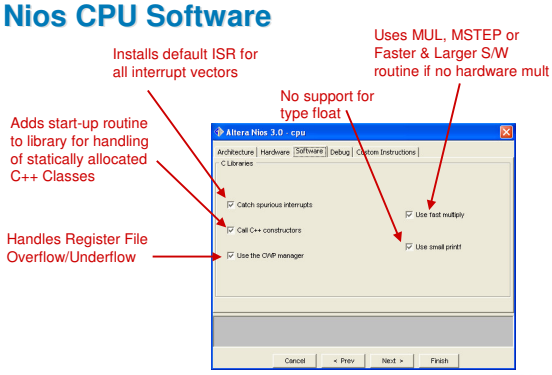
Nios CPU



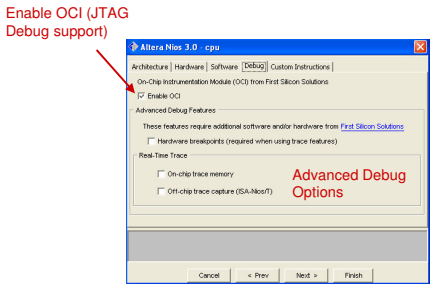
Nios CPU Hardware



Nios CPU Software

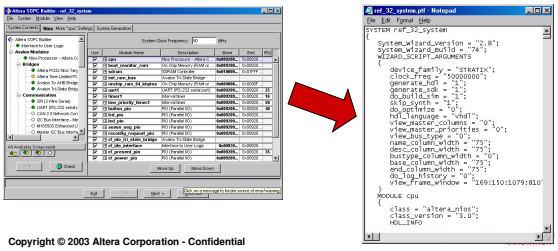


Nios CPU Debug



PTF File

- Text file that records SOPC Builder edits
- Can be used to Archive Nios System
- Do not edit both at the same time



28

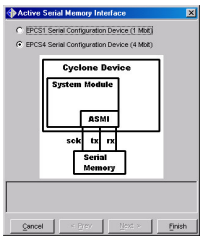
Peripheral Components

- Memory Interface
  - Active Serial Interface
  - On-Chip
    - RAM, ROM
  - Off-Chip
    - SDRAM Controller
    - SRAM
    - Flash
    - ROM
- DMA Controller
  - Memory-Peripheral
  - Memory-Memory
  - Peripheral-Peripheral
- Bridges
  - AHB to Avalon Bus Bridge
- Parallel I/O (PIO) Registers
  - General-Purpose I/O Registers (PIO)
    - Input
    - Output
    - Bidirectional
  - User-Defined Interface
- Serial I/O
  - UART
  - SPI
- Timer
  - Simple Timer
  - Pulse Generator
  - Watchdog Timer

29

ASMI

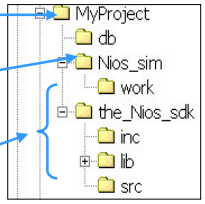
- Interface to Cyclone Config Device
- Unused space can be used for program code or non volatile data storage
- Serial Flash



30

Project Directories

- **Hardware**
  - HDL Source & Netlist
  - db - Quartus project database
- **Simulation**
  - Testbench
  - Automatically generated test memory and vectors
- **Software**
  - **inc** - Header files with address map, interrupts, etc.
  - **lib** - Low-level interface routines for Nios and peripherals
  - **src** - Contains example source code for reference designs



Exercise Files

- Quartus Project files

- Extra design files

- Software files

20 YEARS of ALTERA INNOVATION

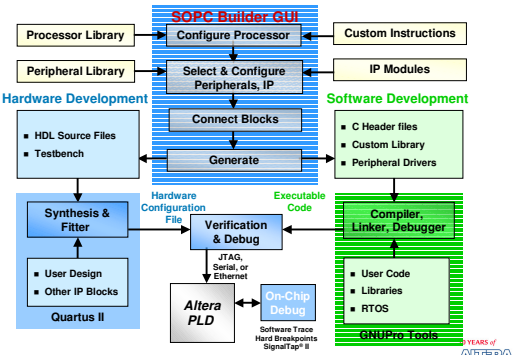
Exercise 1
A Basic Nios Design

30 mins

20 YEARS of ALTERA INNOVATION

Nios Software Development

SOPC Builder Flow



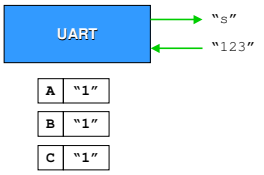
Header Files (excalibur.h)

```
// The Memory Map

#define na_boot_rom             ((void *)      0x00000000)
#define na_boot_rom_end         ((void *)      0x00000400)
#define na_boot_rom_size        ((void *)      0x00000400)
#define na_cpu                  ((np_nios *)   0x00000000)
#define na_uart1                ((np_uart *)   0x00000400)
#define na_uart1_irq                           16
#define na_seven_seg_pio        ((np_pio *)    0x00000420)
#define na_led_pio              ((np_pio *)    0x00000430)
#define na_timer1               ((np_timer *)  0x00000440)
#define na_timer1_irq                          17
#define na_button_pio           ((np_pio *)    0x00000460)
#define na_button_pio_irq                      18

// PIO Peripheral

// PIO Registers
typedef volatile struct
    {
    int np_piodata;          // read/write, up to 32 bits
    int np_piodirection;     // write/readable, up to 32 bits, 1->output bit
    int np_piointerruptmask; // write/readable, up to 32 bits, 1->enable interrupt
    int np_pioedgecapture;   // read, up to 32 bits, cleared by any write
    } np_pio;

// PIO Routines
void nr_pio_showhex(int value); // shows low byte on pio named na_seven_seg_pio
```

Reading/Writing Hardware

- Be careful of compiler optimisation
  - Consecutive reads/writes are optimised away
  - Compiler assumes memory locations are not externally affected

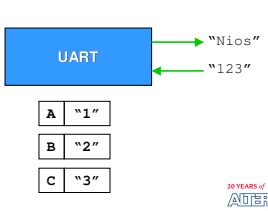
```
char* uart_tx
char* uart_rx

*uart_tx = 'N';
*uart_tx = 'i';
*uart_tx = 'o';
*uart_tx = 's';

A = *uart_rx;
B = *uart_rx;
C = *uart_rx
```



Volatile identifier

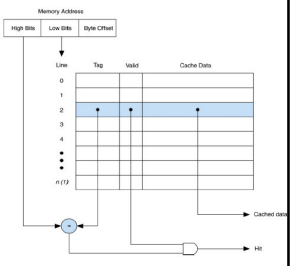
- Declare pointers to hardware as Volatile
  - Compiler will not optimise access to/from these memory locations
  - External hardware can rewrite contents

```
volatile char* uart_tx
volatile char* uart_rx

*uart_tx = 'N';
*uart_tx = 'i';
*uart_tx = 'o';
*uart_tx = 's';

A = *uart_rx;
B = *uart_rx;
C = *uart_rx
```



Instruction and Data Cache

- Available for Nios 32
- Useful for high latency external memories
- Cache is direct mapped
  - Low address bits represent cache line
- Use write-though policy
  - Data is written to external memory as well as cache



D-Cache lines: D-Cache size / 4

I-Cache lines: I-Cache size / 2

Cache Performance

Memory	I-Cache	D-Cache	Normalised Performance
SDRAM	No	No	40.2%
SDRAM	No	Yes	55.2%
SDRAM	Yes	No	64.3%
SDRAM	Yes	Yes	96.4%
OnChip	No	No	100.0%
OnChip	No	Yes	98.0%
OnChip	Yes	No	110.2%
OnChip	Yes	Yes	105.6%

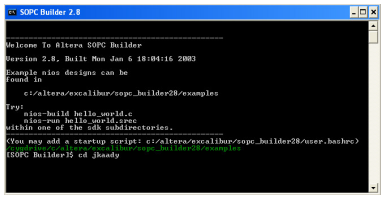
Performance relative to on chip RAM with no Cache running dhry.c modified for unbuffered I/O

Use of Cache

- Cache must be initialised and enabled
  - Routines available for this in `nios_cahce.h`
    - `void nr_icache_init(void);`
    - `void nm_icache_enable(void);`
    - `void nm_icache_disable(void);`
    - Similar for dcache
- To prevent use of Cache
  - ie DMA accessible memory or hardware
  - Use Volatile qualifier for C-code pointers
  - Use PFXIO before LD & LDP instructions in asm
  - PFXIO before LDS gives undefined result

Software Utilities: Nios SDK Shell

- UNIX-like command environment shell
- Start => Programs => Altera => Nios 3.1 => Nios SDK Shell
- Starts in: `c:/altera/kits/nios/examples`

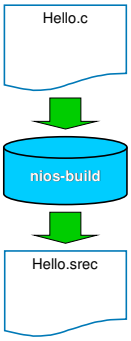


Using Bash

- pwd
  - Shows present working directory
- ls
  - Lists files in current directory
- cd
  - Change directory (using "/" in place of "\")
- cd ..
  - Moves one place up directory structure (try cd ../../..)
- cd //d/
  - Changes to D: drive

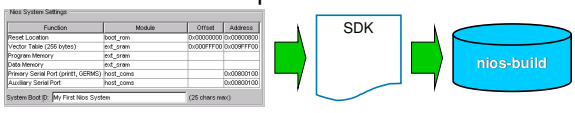
Nios Build

- Simple utility to build software
  - `nios-build hello.c`
  - `nb hello.c`
- Creates .srec file for download
- Supports multiple source files
  - `nb hello.c menu.c mylib.c`
- No need to list all files
  - `nb hello.c`
  - Also links in `hello_*.c`
- Can eliminate need for Make files



Nios Build Detail

- Nios Build takes input from SDK



- Must be used within 5 dirs down of SDK

- If you want to use a Make file
  - see `_sdk/example_makefile` dir

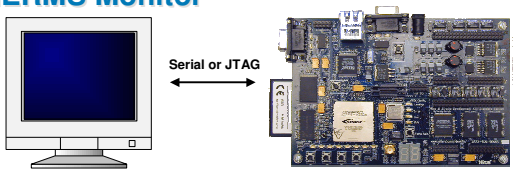
Nios Run

- Utility to Download code and run Terminal
  - `nios-run hello.srec`
  - `nr hello.srec`
- Common switches
  - Start terminal: `nr -t`
  - Download only: `nr -x hello.srec`
  - Force serial: `nr -r hello.srec`
  - Specify baud: `nr -b 9600 hello.srec`
  - Specify port: `nr -p com2 hello.srec`
  - Force JTAG: `nr -j hello.srec`

Advanced Nios Routines

- `nios-elf-objdump`
  - Disassembles object files
- `nios-elf-size`
  - Determines size of srec files
- `nios-elf-gprof`
  - C program execution profiler
- `srec2flash`
  - Prepares srec file to write to dev board flash
- `hexout2flash`
  - Converts Quartus .hexout files to write to dev board flash

GERMS Monitor



- Monitor Program Runs from On-Chip ROM
- Communicates to Host via Serial Port (or JTAG)
- Basic Development Facilities
  - Download Code
  - Burn Flash
  - Examine/Modify Memory
  - Run Programs

GERMS Monitor Commands

- G – Go
  - G40000 Execute code at address 40000
- E – Erase Flash
  - E18000 Erase flash block at 18000
- R – Relocate
  - R18000 Relocate next software download to addr 18000
- M – Memory
  - M14000-14100 Show memory contents for this address range
  - M50000:0001 0002 0003 Write values to consecutive memory locations
  - M50000-50100:AA55 Fill memory range with 16 bit word
- S – Motorola S record
- : – Intel Hex record
- CR – show next 64 Bytes of memory
- Escape – Restart GERMS monitor

Hardware Drivers

- Drivers for Peripherals are produced during generation of the SDK
  - Timer, DMA, LCD, ASMI, seven seg LED etc
  - See data sheet for peripheral for details
- Basic Ethernet Plugs library is also available
  - See mnl_plugs.pdf
  - Select on 2nd page of SOPC Builder

Nios OS / RTOS Support

Provider	Product	Description
Accelerated Technology	Nucleus	Royalty-Free, Source-Available RTOS
Microtronix	µClinux	Open-Source OS
Micriµm	µC/OS-II	Royalty-Free, Preemptive RTOS
MiSPO Co., Ltd.	NORTi	µITRON 4.0-Compatible Real-Time Kernel
Shugyo Design	KROS	Small-Footprint, Royalty-Free, POSIX-Compliant RTOS

20 YEARS of ALTERA INNOVATION

Debug

Software Debug

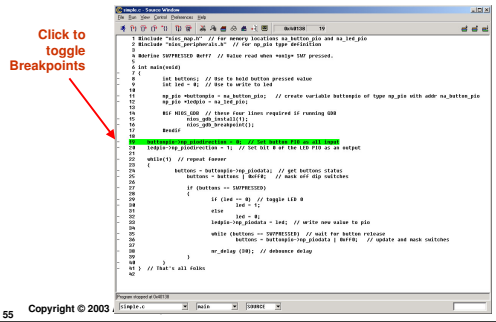
- GDB/Insight over JTAG
- GDB/Insight over Serial
- Nios Console
- Third Party tools
  - Provide extra features such as hardware breakpoints and trace
- To ensure correlation between source and object code switch off compiler optimisation
  - `nb -O0 hello_word.c`

GDB/Insight Over JTAG

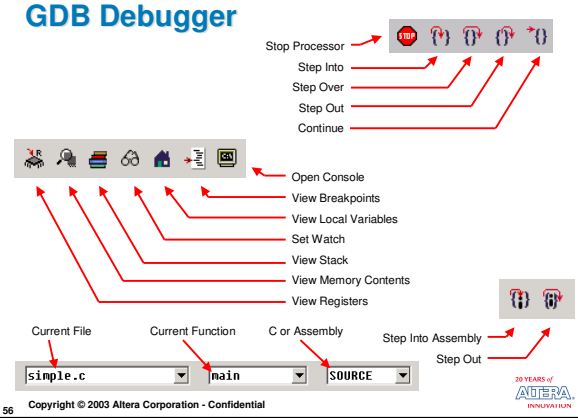
- Build source with nios-build or Make file
  - `nb -O0 hello_world.c`
- Run debug with nios-debug
  - `nios-debug hello_world.srec`
  - `nd hello_world.srec`
- Requirements
  - Must have OCI debug core enabled in CPU
  - Software must reside in RAM

GDB/Insight Debugger

- Program must reside in RAM



GDB Debugger

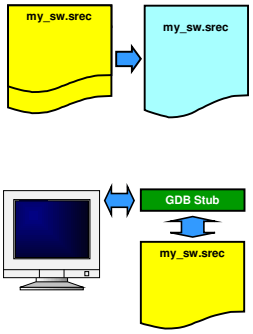


GDB/Insight Over Serial

- Build source with nios-build for debug
  - `nb -d hello_world.c`
- Run debug with nios-debug
  - `nios-debug hello_world.srec`
  - `nd hello_world.srec`
- Requirements
  - Must have UART in System
  - Need another UART for Printf
  - Program must reside in RAM

Serial Debug Detail

- Debug is controlled by extra software linked in to user S/W (GDB Stub)
  - Implemented by nb -d
- GDB stub communicates with host for control
  - step, run, memory, regs etc
- Executes instructions from user code
- Breakpoints implemented by changing opcodes in user S/W



Nios Console

- Build source with nios-build or Make file
  - `nb -O0 hello_world.c`
- Run console with nios-console
  - `nios-console hello_world.srec`
  - `nc hello_world.srec`
- Requirements
  - Must have OCI debug core enabled in CPU
  - Program must reside in RAM

Nios Console

- Command line debugger

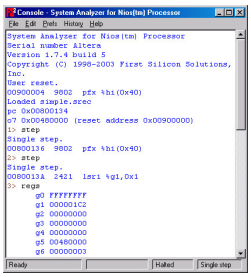


Table 3. Debug Console Commands

Command	Description
go	Starts executing instructions at the current program counter (pc) address.
halt	Stops the processor.
reset	Resets the processor.
pc	Allows you to view the current value of the program counter and change its value.
regs	Allows you to view the registers and change their values.
step	Steps through the code.
bkpt	Sets breakpoints.
dump	Displays the memory contents.
byte, half, word	Writes to memory.

When To Use The Debugger

- Use to Debug Software Issues
- Not Real Time
  - By single stepping code execution cause and effect can be viewed on a line by line basis
- Provides Level of Software Detail Not Available with Other Debug Methods
  - Viewing and editing of registers, memory and variables
- Software Engineers Expect Access to this tool

- Disadvantages
  - Can be difficult to debug real time events
  - Requires that source code is not optimised

Nios Debug Solutions

Provider	Product	Description
Accelerated Technology	code|lab Debug	Full-Featured, RTOS-Aware Debug Environment, Integrated with code|lab EDE
Microtronix	Debugger Module	External Memory Daughter Card for Use as Software Trace Capture Buffer
Microtronix	OCD Solutions Kit	Insight (gdb) Debugger Enhanced to Support Hard Breakpoints & Software Trace
Viosoft	Arriba!	IDE with Integrated Support for Hardware Breakpoints & Processor Trace
Sophia Systems	WatchPoint	Full-Featured Debug Environment with Advanced On-Chip Debug Support
Red Hat	GDB / Insight*	Software Debugger

* Included in Nios Development Kit

Upgrades from FS2

- <http://store.yahoo.com/fs2/>

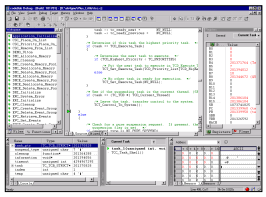
Product Codes (1)

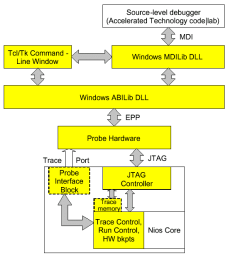
Features	BB-UPG1	BB-UPG2	BB-UPG3	BB-UPG4	Stratix Pro	ISA-NIOS	ISA-NIOS/T
Unlimited Software Breakpoints	X	X	X	X	X	X	X
4 HW Execution Breakpoints	X	X	X	X	X	X	X
Complex Triggers (2)		X	X	X	X	X	X
Instruction Trace			X	X		X	X
Instruction & Bus Cycle Trace				X		X	X
Use ByteBlaster™ cable	X	X	X	X	X		
Black Box--Faster downloading (3)						X	X
Trig in/out						X	X
Connector Used	JTAG	JTAG	JTAG	JTAG	JTAG	JTAG	Mictor 38
On-chip Trace Collection				X		X	X
Off-chip Trace Collection (4)							X
List Price	$495	$995	$1,695	$2,295		$2,995	$4,995

*Prices correct at time of writing

JTAG-Based Debug Interface

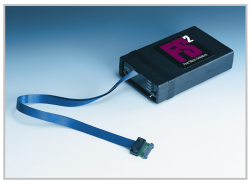
- Nios OCI Debug Module
  - On-Chip Instrumentation (OCI™) Core
  - Integrated Within Nios CPU
  - Created by First Silicon Solutions (FS2) Inc.
- Software Developer Support
  - ATI code|lab EDE & Debug





FS2 System Analyzer

- Nios-ISA System Analyzer
  - 10-pin JTAG Target Connection
  - Supports On-Chip Trace
- Nios-ISA/T System Analyzer
  - 38-pin Mictor Connection
  - JTAG & External Trace
- Single Source
  - Full Hardware & Software Package Available from Mentor Graphics



20 YEARS of ALTERA INNOVATION

Lab 2
Software Flow

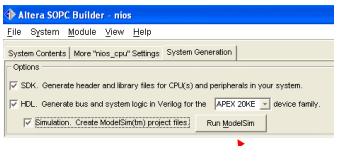
30 mins

20 YEARS of ALTERA INNOVATION

RTL Simulation

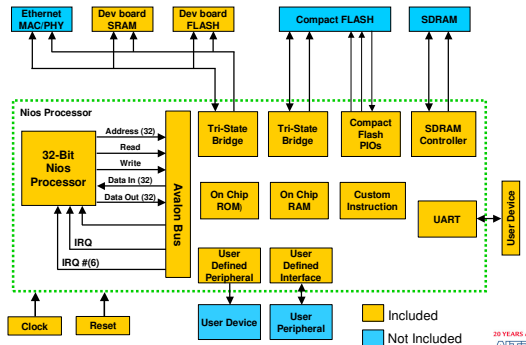
RTL Simulation

- Nios SOPC Builder Automatically creates simulation model plus:-
  - ModelSim Project
  - Testbench
  - Simulation Scripts
  - Formatted Wave Window



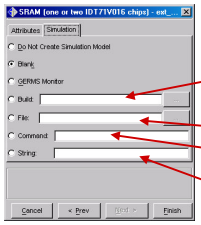
Start ModelSim

Simulation TestBench



Memory Device Simulation Models

- Applies to the following Nios Memories
  - On Chip Memory (ROM or RAM)
  - SRAM
  - Flash Memory



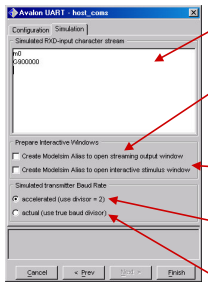
Source files (.c & .s) files are compiled. Data files (.mif & .srec) are converted

File is used in raw format

Useful for non-standard build eg nb –cc –O0 hello.c

String is used in raw format

UART Simulation

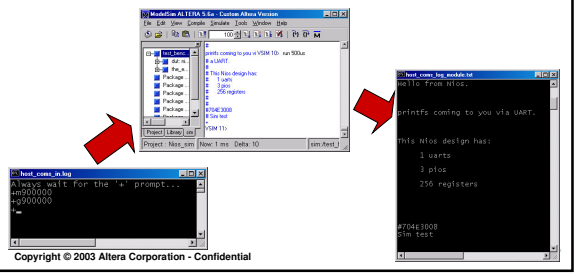


- Text is transmitted to UART during simulation
- Creates and saves txt file containing UART tx stream
- Creates window to input text at simulation run time
- Increase simulation speed
- Simulate true baud rate

RXD-input character stream and interactive stimulus window are mutually exclusive

UART Simulation

- Input is interactive or predefined
- Output is shown and saved independently for multiple UARTs



User Additions to Nios TestBench

```
// <ALTERA_NOTE> CODE INSERTED BETWEEN HERE
//include additional files here
// AND HERE WILL BE PRESERVED </ALTERA_NOTE>


module test_bench ;

  wire    [ 11: 0] in_port_to_the_button_pio;
  reg              clk;
  wire    [  1: 0] bidir_port_to_and_from_the_led_pio;
  wire    [  3: 0] be_n;
  wire             write_n_to_the_ext_flash;
  wire    [ 15: 0] address;
  wire             select0_n_to_the_ext_ram;
  wire             uart1_s1_readyfordata_from_sa;
  wire    [ 19: 0] off_chip_bus_address;
  wire             write_n;
  wire    [  3: 0] off_chip_bus_byteenablen;
  reg              reset_n;
  wire    [ 31: 0] off_chip_bus_data;
  wire             select1_n_to_the_ext_ram;
  wire    [ 15: 0] out_port_from_the_seven_seg_pio;
  wire             select0_n;
  wire             off_chip_bus_readn;
  wire             read_n;
  wire             txd_from_the_uart1;
  wire             rxd_to_the_uart1;
  wire             select_n_to_the_ext_flash;
  wire             uart1_s1_datasavailable_from_sa;
  wire    [ 31: 0] data;
  wire             select1_n;
  wire             write_n_to_the_ext_ram;


// <ALTERA_NOTE> CODE INSERTED BETWEEN HERE
//  add your signals and additional architecture here
// AND HERE WILL BE PRESERVED </ALTERA_NOTE>
```

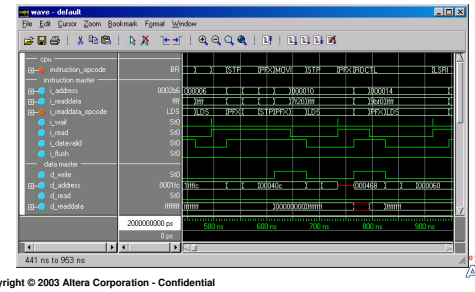
- SOPC Builder creates testbench embedded in top level file eg Nios.v
- Sections within this file are reserved to add user files and code
- These sections are preserved if the SOPC builder is used to re-generate the Nios system

Simulation Scripts

- When ModelSim is started from the SOPC Builder a set-up script is run automatically which creates aliases for simulation scripts
- The set up script can be run independently as follows:
  - do setup_sim.do
- Simulation Scripts
  - s↵ Compiles HDL source code and loads design
  - c↵ Rebuilds memory contents based on software code
    - Includes changes since Nios generation
  - w↵ Opens Wave window with “useful” signals
  - l↵ Opens List window with “useful” signals
  - h↵ Displays help message describing scripts

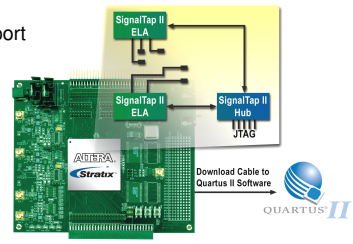
Wave Window

- Adds UART and CPU signals by default
  - CPU Opcodes are decoded and displayed to help trace software execution



SignalTap™ II Logic Analyzer

- Up to 200 MHz
- Multi-Analyzer Support
- 1,024 Channels
- 128K Samples
- 10 Trigger Levels
- No Probes!



Capture the state of internal nodes In-system, at full system speeds

20 YEARS of ALTERA INNOVATION

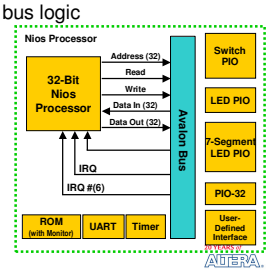
Avalon Bus

User Peripherals

- What if I need to add a peripheral not included with the Nios system?
  - This is a very common practice where a user wants to add their own peripheral to perform some kind of proprietary function or perhaps a standard function that is not yet included as part of the Nios kit.
- We are now going learn how to connect our own design directly to the Nios system via use of the Avalon bus
  - As many peripherals contain registers we could also have chosen to connect to a pio rather than directly to the bus

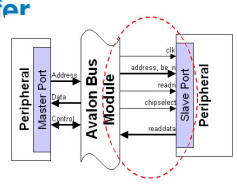
Avalon Bus

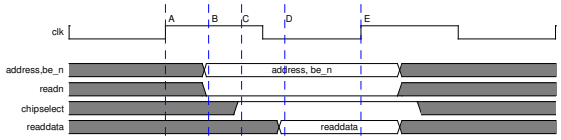
- Proprietary bus specification used with Nios
- Principal design goals of the Avalon Bus
  - Low resource utilization for bus logic
  - Simplicity
  - Synchronous operation
- Transfer Types
  - Slave Transfers
  - Master Transfers
  - Streaming Transfers
  - Latency-Aware Transfers



Slave Read Transfer

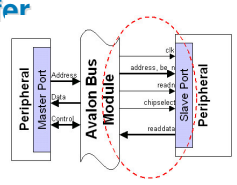
- 0 Setup Cycles
- 0 Wait Cycles

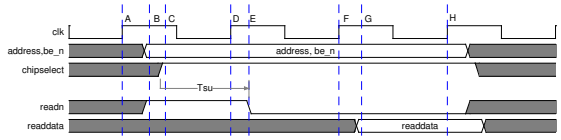




Slave Read Transfer

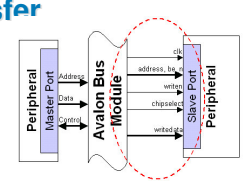
- 1 Setup Cycle
- 1 Wait Cycle

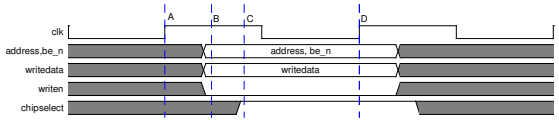




Slave Write Transfer

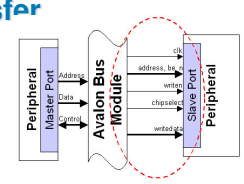
- 0 Setup Cycles
- 0 Wait Cycles
- 0 Hold Cycles

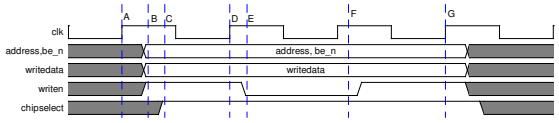




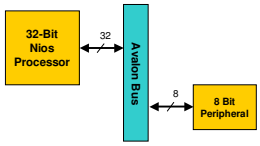
Slave Write Transfer

- 1 Setup Cycle
- 0 Wait Cycles
- 1 Hold Cycle





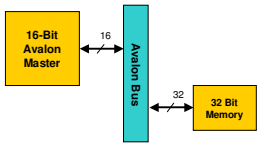
Address Alignment – Narrow Slave



Peripheral Registers	
Base	aa
Base + 0x1	bb
Base + 0x2	cc
Base + 0x3	dd
Base + 0x4	ee

- Dynamic Address Alignment
  - LD from Base + 0x0: dd cc bb aa
  - LD from Base + 0x4: uu uu uu ee
- Native Address Alignment
  - LD from Base + 0x0: uu uu uu aa
  - LD from Base + 0x4: uu uu uu bb
  - LD from Base + 0x8: uu uu uu cc

Address Alignment – Narrow Master

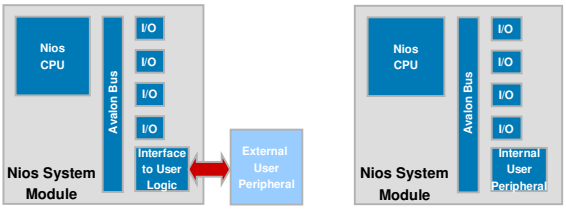


Memory Contents	
Base	33 22 11 00
Base + 0x4	77 66 55 44
Base + 0x8	bb aa 99 88

- Dynamic Address Alignment
  - LD from Base + 0x0: 11 00
  - LD from Base + 0x2: 33 22
  - LD from Base + 0x4: 55 44
- Native Address Alignment
  - LD from Base + 0x0: 11 00
  - LD from Base + 0x2: 55 44
  - LD from Base + 0x4: 99 88
  - High bytes are unobtainable – warning issued

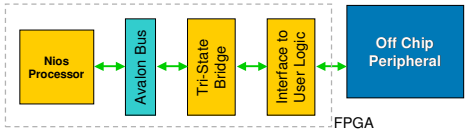
Interface To User Logic

- Used to Connect to Existing HDL
- Mapped into Nios Memory Space
- Can be “Inside” or “Outside” Nios System



Tri-State Peripherals

- Must use Tri-State Bridge



- Tri-State peripheral is defined by the presence of a bi-direction data port
- Off chip peripherals do not have to be Tri-State

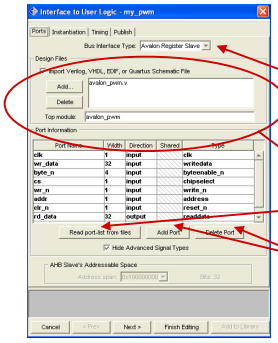
Addr Connections for Tri-State Slaves

- Assume byte addressing when connecting up slaves

Table 15. Connecting the Avalon Bus Module to External Devices

Alignment	Master Width	Slave Width	A[0] on Slave Is Connected to Byte Address Bit Number
native	32	32	2
native	32	16	2
native	32	8	2
native	16	32	-- Not Applicable --
native	16	16	1
native	16	8	1
dynamic	32	32	2
dynamic	32	16	1
dynamic	32	8	0
dynamic	16	32	2
dynamic	16	16	1
dynamic	16	8	0

Interface To User Logic



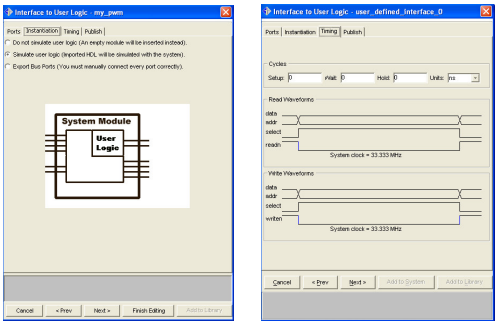
Choose interface type
Register slave uses native alignment, memory Slave uses dynamic alignment

Add design files that describe the user logic

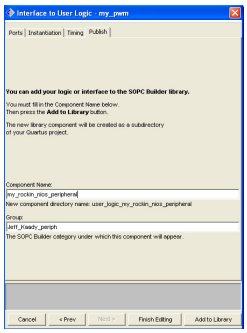
Automatically define port table from design files

Make port changes or enter ports manually

Interface to User Logic

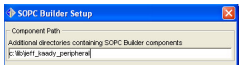


Publishing User Logic



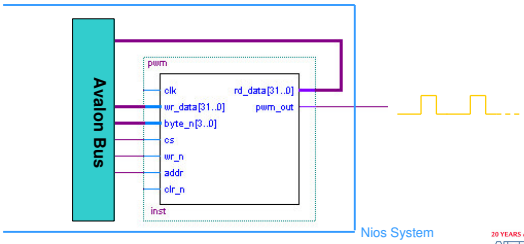
- Fill in the details and click Add to Library
- This adds component to current Quartus project
- To add to library
  - copy the dir created to a library dir
  - Edit SOPC Builder Setup

File -> SOPC Builder Setup



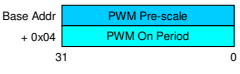
Add a User PWM to the Nios System

- HDL for PWM already exists with standard micro-processor type interface
- This will be added to our Nios system in the next Lab

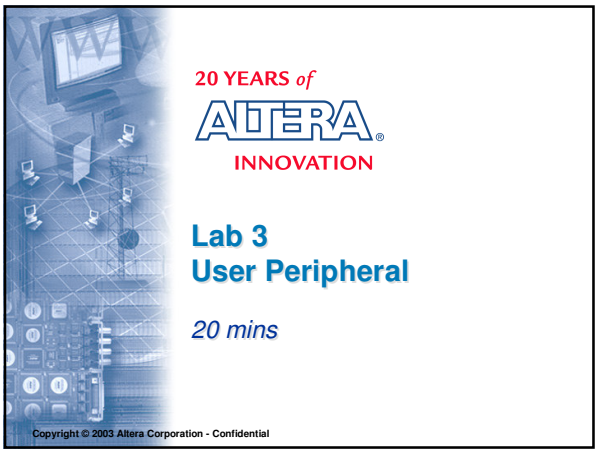


PWM Memory Map

- Pre-scale factor divides Nios clock to produce PWM operating frequency
- On Period should be less than or equal to Prescale



$$\text{PWM Duty Cycle} = \frac{\text{On Period}}{\text{Pre-scale}}$$

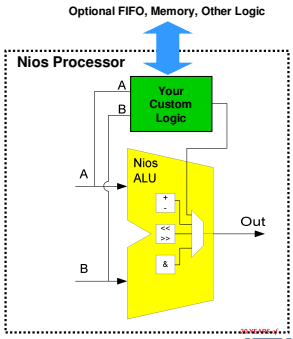


Custom Instructions

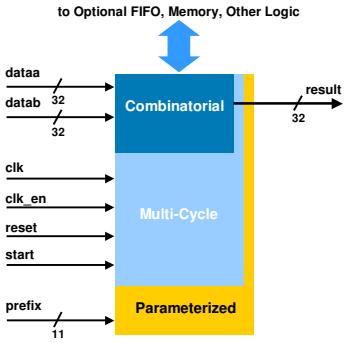
- Custom Instructions Augment Nios Instruction Set
- Integrated Into Nios Development Tools
  - System Builder Design Tool Handles Op-code Assignment
  - Generates C and Assembly-language Macros
- Application Examples
  - Data Stream Processing (E.G. Network Applications)
  - Application Specific Processing (E.G. MP3 Audio Decode)
  - Software Inner Loop Optimization

Custom Instruction

- Dramatic Boost in Processing Performance
  - ***No Increase in f_{MAX}***
- Extends Nios Instruction Set
  - Up to Five Instructions
- SOPC Builder Development Tool
  - Automatically Adds User Logic to Nios ALU
  - Assigns Op-Code
  - Generates C & Assembly Macros



Three Levels of Customization



Why Custom Instruction?

- Dramatically Accelerate Software Algorithms
  - Reduce Complex Sequence of Instructions to One Instruction
- Example: Floating Point Multiply

```
float a, b, result_slow, result_fast;

result_slow = a * b;         /* Takes 2,874 clock cycles  */
result_fast = nm_fpmult(a, b);  /* Takes 19 clock cycles */
```

Faster than DSP Processor!

- Typical Flow
  - Profile Code (gprof)
  - Identify Critical Inner Loop
  - Create Custom Instruction Logic
    - Replace One or All Instructions in Inner Loop
  - Import Custom Instruction Logic into Design
  - Call Custom Instruction from C or Assembly

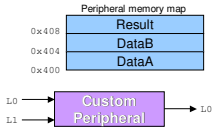
Custom Instruction vs Peripheral

- Custom Instruction can execute in a single cycle
  - No overhead for call to custom Hardware

```
USR0  %L0, %L1
```

- Access to same hardware as peripheral takes an extra 4 cycles
  - Add 2 more cycles if base address is not 16 bit

```
PFX   %hi(0x400)
MOVI  %L3, %lo(0x400)
ST    [%L3], %L0
STP   [%L3, 1], %L1
LDP   %L0, [%L3, 2]
```



Hardware Design

- Custom Instruction module can be of following formats:
  - VHDL
  - Verilog HDL
  - EDIF
  - Quartus Block Diagram (.bdf)
- Port list
  - All Custom Instruction Modules need these ports
    - Port names must match exactly

Table 1. Ports for Combinatorial Custom Logic

Port Name	Width (Bits)	Direction	Description
dataa	CPU width	Input	Operand.
datab	CPU width	Input	Operand (optional).
result	CPU width	Output	Result.

Hardware Design

- Port list for Multi-Cycle Custom Instructions
  - Must have all of these ports with exact names

Table 2. Additional Ports for Sequential Custom Logic

Port Name	Width (Bits)	Direction	Description
clk	1	Input	CPU master input clock, which is fed by the Nios system clock.
reset	1	Input	CPU master asynchronous reset, which is fed by the Nios master reset. reset is only asserted when the Nios system is reset.
clk_en	1	Input	Clock qualifier. The custom logic block should use the clk_en signal as a conventional clock qualifier signal and should ignore all rising clock edges when clk_en is not asserted.
start	1	Input	Instructs the block to latch data and begin operation.

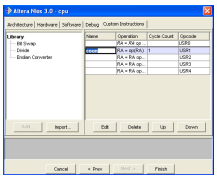
- Port list for Prefixed Custom Instructions
  - Must have exact port names

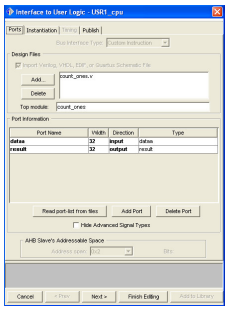
Table 3. Prefix Port

Port Name	Width (Bits)	Direction	Description
prefix	11	Input	Payload of the K register.

Define Custom Instruction

- Select from Library or Import





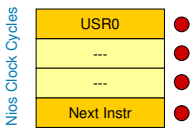
Software Design

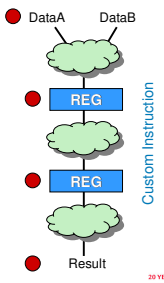
- C Code Macros (include excalibur.h)
  - nm_<macro_name> (dataa, datab)
  - nm_<macro_name>_pfx (prefix, dataa, datab)
- Assembly Code
  - Use Opcodes or Assembly Macro

```
LD   %r1,[%L6]          ; Load word at [%L6] into %r1
LD   %r0,[%L2]          ; Load word at [%L2] into %r0
PFX 1                   ; Only needed if using prefix
nm_my_cust_inst %r1     ; Macro calling a RW opcode, r1 <= r1 "OP" r0
ST   [%L4],%r1          ; %L4 is the pointer, %r1 is stored
```


Multi-Cycle Custom Instructions

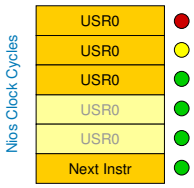
- Processor stalls whilst awaiting result
  - Clock cycles = 3

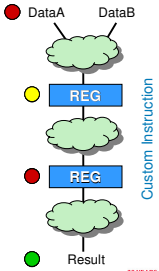




Pipelined Custom Instructions

- Result not always needed for each input
  - Clock Cycles = 1
  - Route start sig to reg clk_en





Design Example

- Requirement
  - Count the number of bits that are logic '1' within a block of data
- Pseudo code

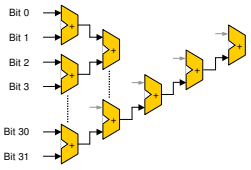
```
For pointer = start to finish Loop
    temp = @pointer
    For index = 0 to 31 Loop
        accumulator = accumulator + temp(index)
    End Loop
End Loop
```


Design Example

- Assembly Code
  - Use assembly code for speed
  - Inner loop takes 64 cycles
  - Loop control = 6 cycles
- CPU time for each data word
  - 70 Cycles

```
IF1 %r0, 0      ; if bit 0 is 1
ADDI  %l2, 1  ; inc accumulator
IF1 %r0, 1      ; if bit 1 is 1
ADDI  %l2, 1  ; inc accumulator
etc ....
```

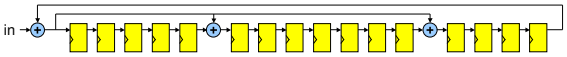
- Custom Instruction
  - 31 adders (56 LEs)
  - Inner loop takes 1 cycle
  - Loop control = 7
- CPU time for each data word
  - 8 Cycles



Custom Instruction improves performance by 875%

Cyclic Redundancy Check

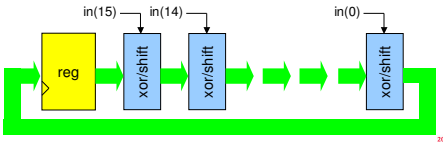
- CRCs are used to check data integrity
- Are best understood in serial format
- The following implements CRC16-CCITT
  - A common CRC used worldwide
  - Used mainly on 8 bit data ie ASCII



- CRCs often appear in embedded software applications

Accelerating CRC

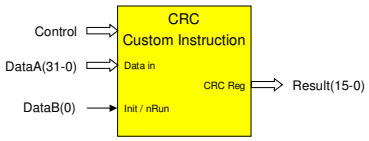
- Implementing the shift and XOR for each bit takes many clock cycles ~50
- Software algorithms tend to use look up tables to pre-compute each byte
- Parallel Hardware is fastest



CRC Custom Instruction

- CRC16-CCITT needs to be preset to 0xFFFF at the start of each computation
- Can use the Data B input to select between run and load
  - Use of prefix would waste a clock cycle

```
// reset crc
nm_crc(0xFFFF, 1);
// run crc
nm_crc(word, 0);
```



20 YEARS of ALTERA INNOVATION

Lab 4
Custom Instruction

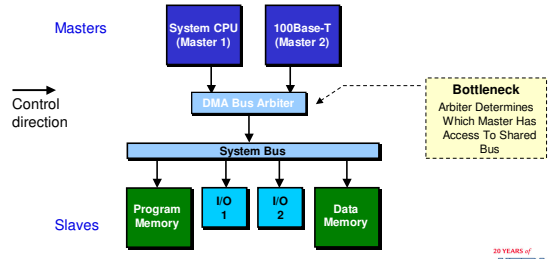
20 mins

20 YEARS of ALTERA INNOVATION

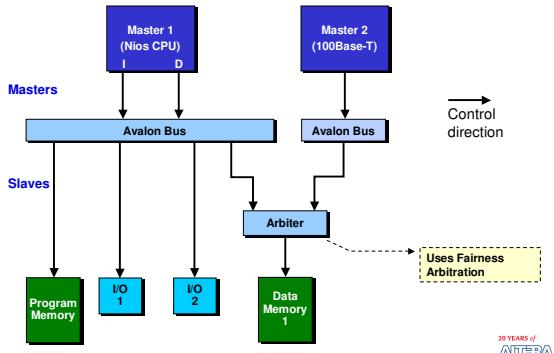
Multi-Masters and Direct Memory Access (DMA)

Traditional Multi-Masters

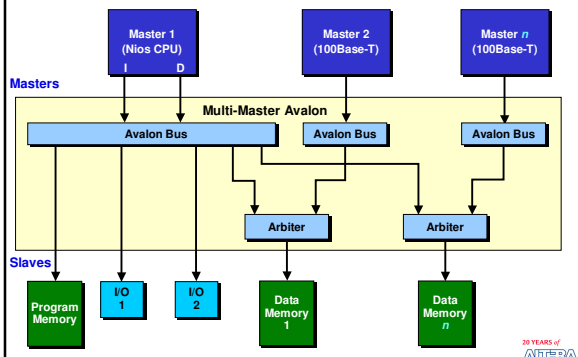
- Direct Memory Access (DMA)
  - Processor Waits For Bus During DMA



Simultaneous Multi-Master Bus

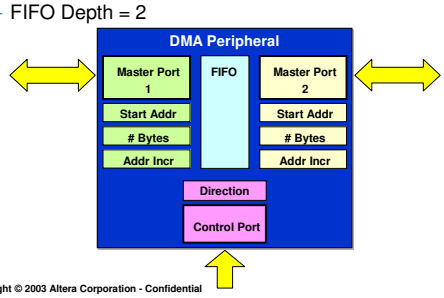


Simultaneous Multi-Master Bus

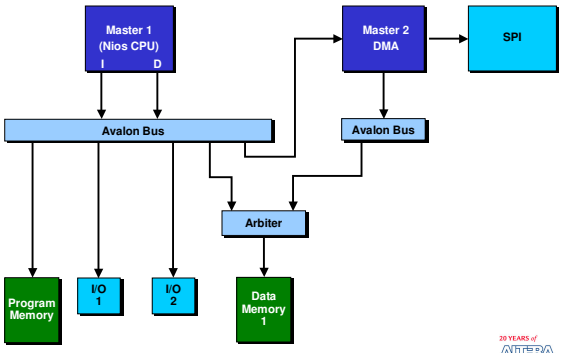


DMA Peripheral

- Provides Bus Master Capability to Any Nios Peripheral
  - FIFO Depth = 2

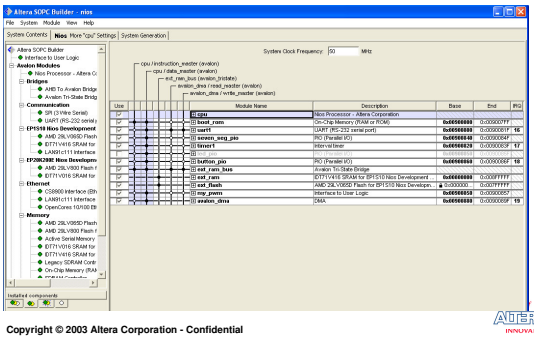


DMA Peripheral



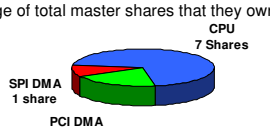
Master Connections

- View => Show Master Connections



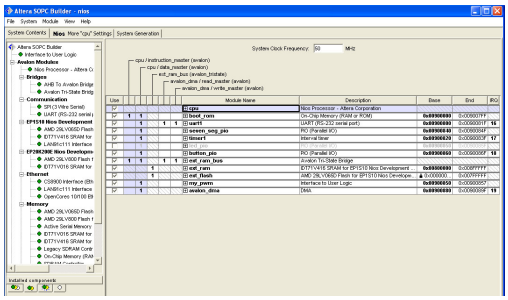
Master Arbitration Scheme

- Nios Multi-Master Avalon Bus utilises Fairness arbitration scheme
  - Each Master/Slave pair is assign an integer “shares”
  - Upon conflict Master with most shares takes bus until all shares are used
  - Master with least shares then takes bus until all shares are used
  - Assuming all Masters continuously request the bus, they will each be granted the bus for a percentage of time equal to the percentage of total master shares that they own



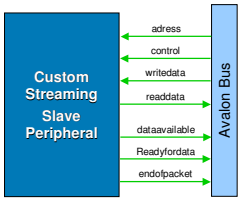
Set Arbitration Priority

- View => Show Arbitration Priorities



Custom Streaming Slave Peripherals

- For use of DMA with slow peripherals
  - Example: UART
- Adds up to three outputs to Avalon Slave
  - dataavailable
  - readyfordata
  - endofpacket

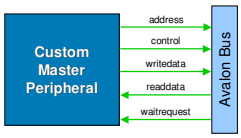


Streaming Peripheral Signals

- dataavailable
  - Indicates that the peripheral has data available to be read by DMA or other Master
  - ie, there is data in the rx buffer or register
- readyfordata
  - Indicates that the peripheral is able to receive data written by DMA or other Master
  - Ie, the tx buffer or register is not full
- Endofpacket
  - Usage not defined
  - DMA can be optionally set to end transfer

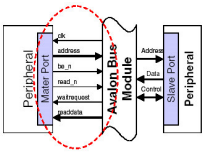
Custom Master Peripherals

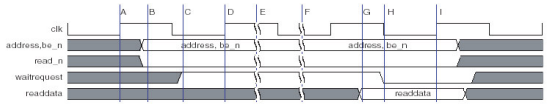
- Integrates DMA function
  - Eg VGA that takes data from memory directly
- Simpler than Slave peripherals
  - Assert outputs until waitrequest is low
- Transaction are between Master and Avalon Bus, not Slave



Master Read Transfer

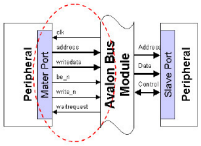
- Assert addr, be, read
- Wait for waitrequest = '0'
- Read in Data
- End of transfer

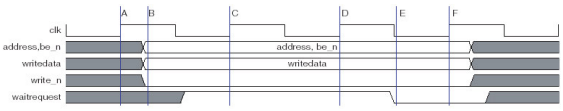




Master Write Transfer

- Assert addr, be, read
- Assert Write Data
- Wait for waitrequest = '0'
- End of transfer





20 YEARS *of*

ALTERA®

INNOVATION

Lab 5
Run Custom Design

20 mins

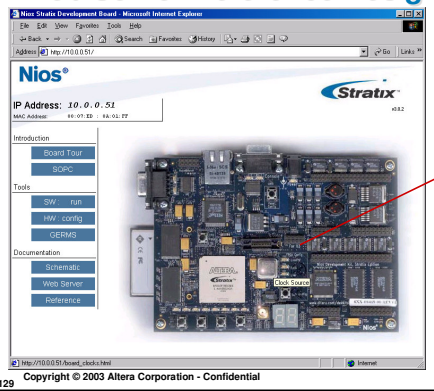
20 YEARS *of*

ALTERA®

INNOVATION

Working with the Development Board

Web Server Reference Design



Clock Networks Board Schematic Board Feature List

Original Clock Source

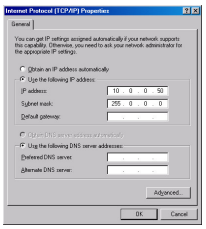
The clock-networks on the Nios Development board are derived from one of two possible original sources:

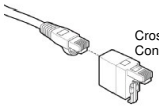
- Y2 is a 50MHz free-running crystal oscillator. It is socketed so you can easily replace it with another frequency if required.
- J4 (Ext Clock) is an external-clock input connector. You can attach an external clock source (e.g. a signal-generator).

Left click mouse to get description of board components

Connecting to Web Sever

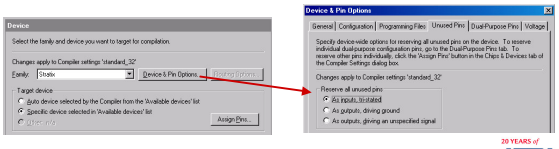
- Configure PC IP Address
  - Open LAN Settings and click Properties
  - Select Internet Protocol (TCP/IP) and click Properties
  - Make the settings shown
  - Don't forget to restore these settings later
- Connect PC to Dev board via the Crossover connector
- Open Internet Browser and enter "10.0.0.51" in address bar



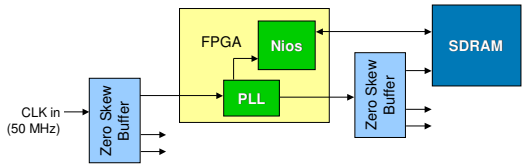


Ensure Unused I/O are Tri-State

- The FPGA may connect to components on the board not used by your design
- There is a connection between FPGA and MAX device to force reconfiguration
  - Active low, pulled high
- Assignments -> Device …



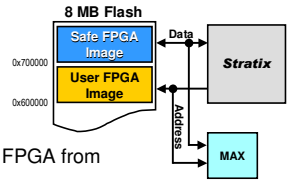
Clock Distribution

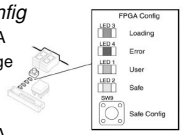


- PLL required to meet SDRAM I/O timing
  - Introduces -60° phase shift relative to Nios
- CLK in is socket crystal or external input
  - Resistor changes required for external
- See board schematic and ref design

Hardware Configuration Process

- Flash Configuration
  - Two FPGA images
    - Safe Image
    - User Image
- MAX® EPM7128 Configures FPGA from Flash
  - Upon power up or press of *Reset Config*
    - MAX Device Loads User Image into FPGA
    - If This Fails MAX Device Loads Safe Image
      - Failure includes no user image present
  - Upon press of Safe Config
    - MAX Device Loads User Image into FPGA





Downloading Hardware to Flash

- hexout2flash Utility
  - Converts Quartus Hardware Configuration File (.hexout) to GERMS Script for Download to Flash
  - Default Target: User Image Portion of Flash
    - Assumes Flash located at address 0x00000
- Example:

```
hexout2flash my_nios_design.hexout
```

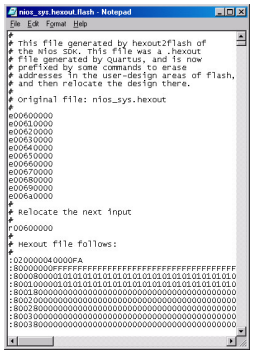
- Generates:

```
my_nios_design.hexout.flash
```

- To Run Download Script:

```
nios-run my_nios_design.hexout.flash
```


Result of hexout2flash

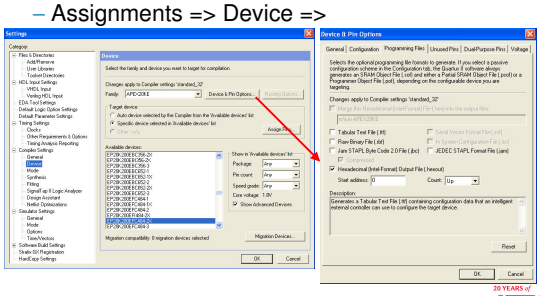


- Based on Flash base address of 0x0 and 1S25 config file
- Use switches for other scenarios

- `Hexout2flash -b 0x400000`
  - Target addr 0x400000
- `Hexout2flash -s 200000`
  - 200000 byte config file

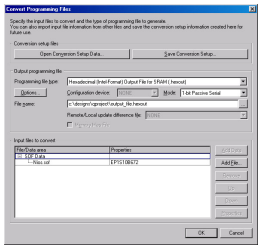
Creating the Hexout File

- This is not created by default
  - Assignments => Device =>



What If I Forget the Hexout?

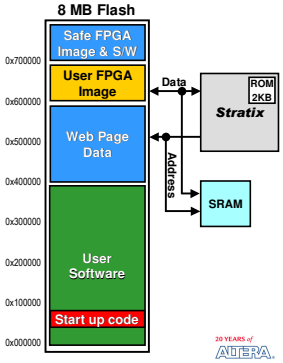
- Convert SOF File
  - Select Convert Programming Files from the File Menu
  - Change Output programming file to type:
    - Hexadecimal (Intel-Format) Output File for SRAM (.hexout)
  - Click Add File
    - browse to project SOF file
    - Click Open
  - Click OK



Processor Boot Process (User S/W)

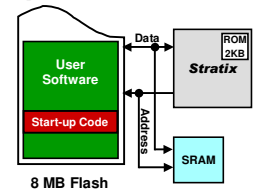
- Boots from "Reset" Address
  - Default is GERMS Monitor
    - On-Chip ROM
- During GERMS Boot
  - If SW0 is not pressed
    - Checks if user software is present at address 0x40000
    - Runs user code
    - Code is identified by the characters "Nios" at 0x4000C
  - If SW0 is pressed
    - User code is not run
    - GERMS runs until go command is received

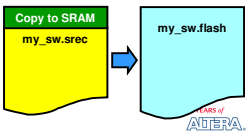
SW0 is defined as bit 0 of button_pio



Application Software Operation

- Boot Program from Flash
  - Copy Executable File (srec) to Flash
  - Code Stored at Address base +0x40000 Is Executed Automatically
- Use Flash for Program Storage
  - Flash Is Slow
  - Difficult Writing to Flash
- Execute Code from SRAM
  - *srec2flash* Utility
  - Adds "Copy" Code to srec File
    - Transfers Program to SRAM
    - Executes Program from SRAM





Downloading Software to Flash

- srec2flash Utility
  - Relocates Executable (srec) file from Memory to Flash
    - Default Target: Software Image Portion of Flash
  - Adds "Copy" Header Program to Flash Image
- Example:

```
srec2flash my_nios_program.srec
```

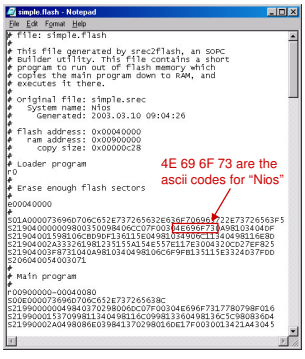
- Generates:

```
my_nios_program.flash
```

- To Run Download Script:

```
nios-run my_nios_program.flash
```

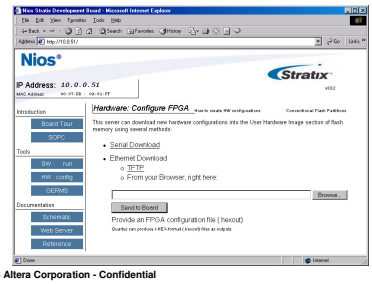

Result of srec2flash



- Targets user S/W portion of Flash
- Base address of Flash and SRAM can be user defined
- See White Paper to port to other systems

Downloading to Board

- OCI core only supports .srec files
- Can also use Web Server ref design



Reset the Board to Factory Settings

- Use ByteBlaster MV to reprogram FPGA with recovery.sof file
  - `recovery_configuration_stratix_1s10.sof`
- Press CPU Reset whilst holding down SW0
- Download Nios 3.0 flash image
  - `nr -x -r default_board_image_stratix_1s10.flash`
- Files located in ….\Excalibur\sopc_builder\examples

1S10ES Device on Stratix Board

- Early versions of the Stratix board feature ES silicon
- Two Errata points:
  - Needs configuration data length of 1S25
    - Plenty of on board flash to accommodate this
  - High power consumed during configuration
    - Power supply is designed for this
- Choose the EP1S10F780C6ES within Quartus II

Partner Development Kits

- Linux Development Kit
  - IDE Interface
  - Compact Flash
  - SDRAM Controller
- VGA/LCD Touch Screen
- Secure Wireless Development Kit
- ADC/DAC Analog Module
- ACEX EP1K100 Development Kit

The List Keeps Growing

20 YEARS *of*

ALTERA®

INNOVATION

Thank You