

NIOS II Development with the Altera DE2-115 FPGA and the CPUIlator Simulator

Presented by:

Alae Bouchiba
Elyes Khechine



01

FPGA

Filed-Programmable Gate Array

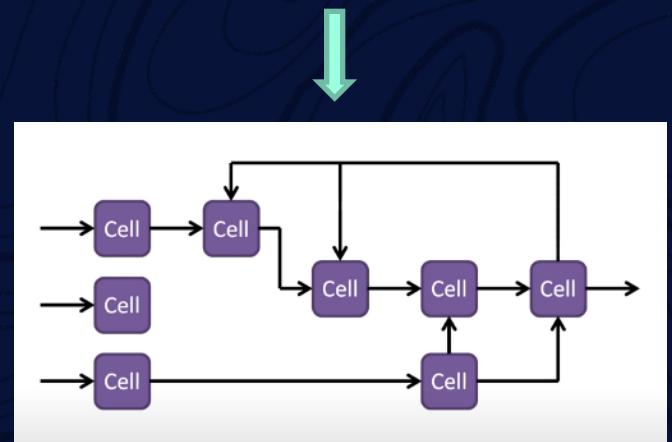
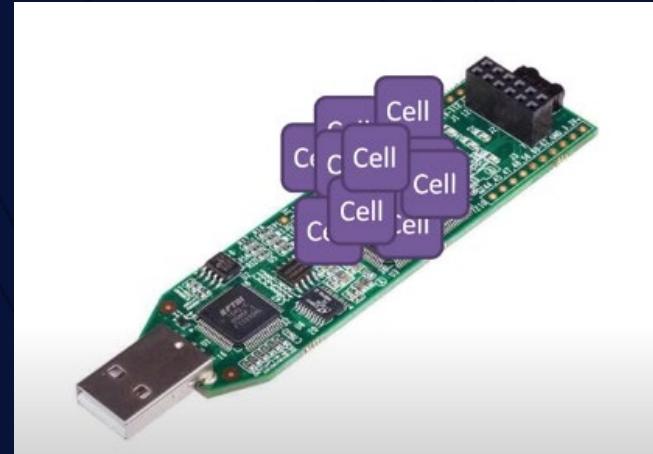
What is an FPGA?

FPGA stands for field programmable gate array. It's an integrated circuit or chip that allows you to design completely custom digital logic- hence the term field-programmable.

The FPGA configuration is generally specified using a hardware description language (HDL).

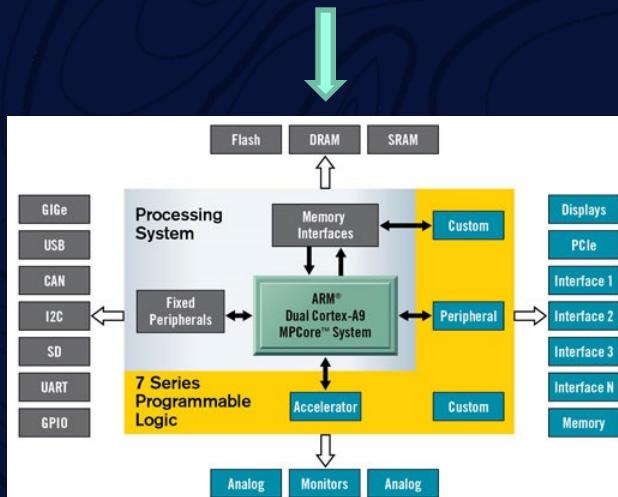
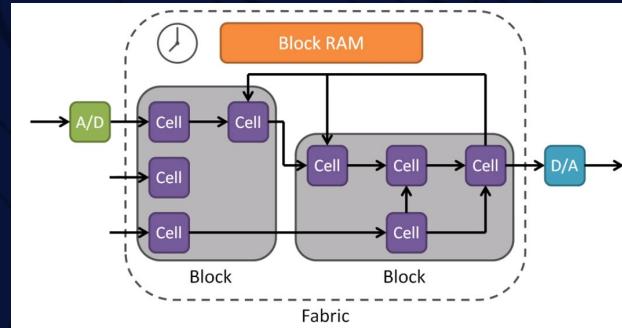
FPGAs are made up of a bunch of logic cells that act as fundamental building blocks for creating digital circuits.

You can configure the individual cells to operate in certain ways and you can connect the cells together to form the basis of any number of digital circuits.



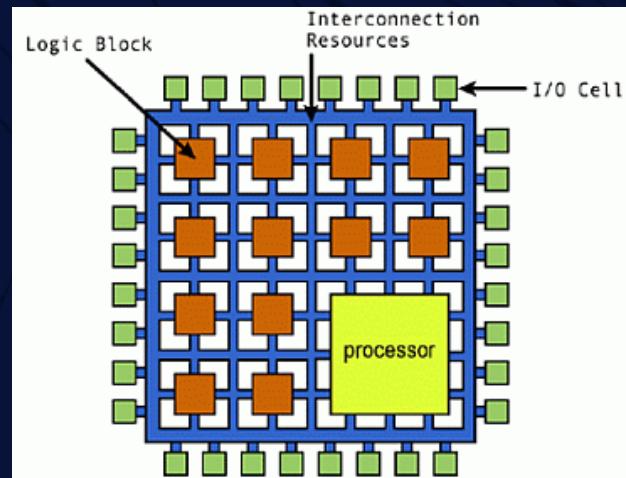
What is the advantage of an FPGA?

- Some FPGAs can have other peripherals like analog to digital converters or analog outputs cells which are often grouped into **programmable logic blocks** and this reconfigurable group of interconnected hardware is often referred to as the FPGA fabric.
- Logic blocks can be configured to perform complex combinational functions, or act as simple logic gates like AND and XOR.
- Many FPGAs can be reprogrammed to implement different logic functions, allowing flexible reconfigurable computing as performed in computer software.
- FPGAs have a remarkable role in embedded system development due to their capability to start system software development simultaneously with hardware, enable system performance simulations at a very early phase of the development before final freezing of the system architecture.



What is a Softcore Processor?

- Unlike a purpose-built microcontroller/processor which has static peripherals, in an FPGA you can use these logic blocks to make your own processor. This is known as a **softcore processor**, and it allows you to run code like you would on a microcontroller or microprocessor.
- Soft CPUs are microprocessors whose architecture and behavior are fully described using a subset of a hardware description language (HDL). If you have the space, you might even be able to implement more than one processor on the FPGA.
- In contrast, you may or may not have access to some of the peripherals you find on a microcontroller like those analog to digital converters.
- Soft processors can be synthesized for any application-specific integrated circuit (ASIC) or field programmable gate array (FPGA) technology.



Examples of Soft CPUs

Nios II, MicroBlaze, PicoBlaze and Xtensa are the leading soft-core processors provided by Altera, Xilinx and Tensilica respectively.

COMPARISON OF SOFT-CORE PROCESSORS

Category	Nios II (Fast Core)	MicroBlaze	Xtensa XL	OpenRISC 1200	LEON3
Maximum MHz	200 (FPGA)	200 (FPGA)	350 (ASIC)	300 (ASIC)	400/125 (ASIC/FPGA)
ASIC/FPGA Technology	– /Stratix and Stratix II	– /Virtex-4	0.13 μ m/ –	0.18 μ m/ –	0.13 μ m/Not given
Reported DMIPS	150 DMIPS	166 DMIPS	–	250 DMIPS	85 DMIPS
ISA	32-bit RISC	32-Bit RISC	32-Bit RISC	32-bit RISC	32 or 64-bit RISC
Cache Memory (I/D)	Up to 64 KB	Up to 64 KB	Up to 32 KB (1)	Up to 64 KB	Up to 256 KB
Floating Point Unit (optional)	IEEE-754	IEEE-754	IEEE-754	As peripheral	IEEE-754
Pipeline	6 Stages	3 Stages	5 Stages	5 Stages	7 Stages
Custom Instructions	Up to 256 Instructions	None	Unlimited	Unspecified limit	None
Register File Size	32	32	32 or 64	32	2 to 32
Implementation	FPGA	FPGA	FPGA, ASIC	FPGA, ASIC	FPGA, ASIC
Area	700-1800 LEs	1269 LUTs	0.26 mm ²	N/A	N/A

As shown in the table, LEON3 has the highest operating frequency for ASIC implementation but has the lowest for FPGA implementation. The Nios II and Microblaze soft-core processors have the highest operating frequency for FPGA implementation.

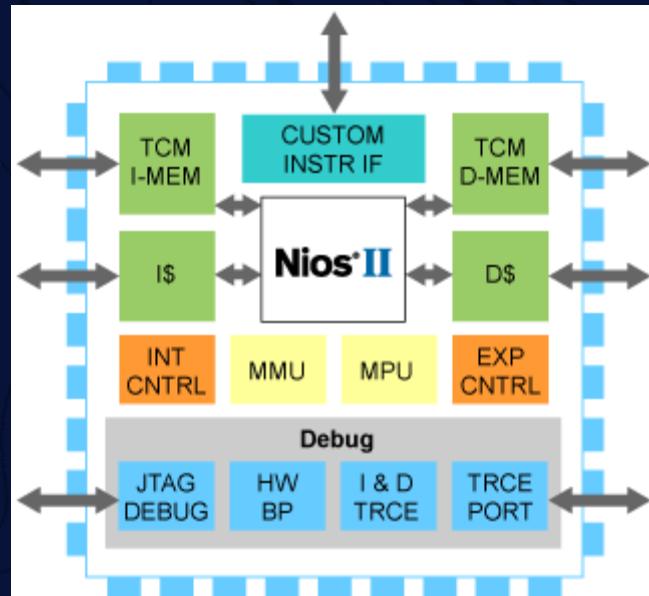
02

Nios II Processor

The World's Most Versatile Embedded Processor

Nios II Processor

- The **Nios II** Soft-Core Processor is a general purpose 32-bit Reduced Instruction Set Computer(RISC) embedded processor core that features a Harvard memory architecture.
- It is a successor to Altera's first configurable 16-bit embedded processor Nios, introduced in 2000.
- It features major improvements focused on the reduction of logic element(LE) consumption on an FPGA and improved performance, making it more suitable for a wider range of embedded computing applications, from digital signal processing(DSP) to system-control.



Nios II Processor Features



Single-Cycle Hardware

Multiply and barrel shifter.



Custom Instructions

Up to 256 custom instructions and unlimited hardware accelerators.



External Vector Interrupt Controller

Up to 32 interrupts per controller.



Advanced Exception Support



Separate Instruction and Data Caches

Configurable from 512 bytes to 64 KB.



Achieve Maximum DMIPS

Up to six-stage pipeline to achieve maximum DMIPS per MHz.



Access up to 4 GB of external address space



Optional Tightly-Coupled Memory for Instructions and Data

Benefits of Nios II Embedded Processors

01

Flexibility

With the perfect fit of CPUs, peripherals, memory interfaces, and custom hardware peripherals to meet the unique demands of every new design cycle.

02

Long Life Cycle

Nios II processors can help product developers to maximize their return on a product by providing life cycle benefits at every stage of a product's life.

03

Low Cost

Provides phenomenal cost flexibility, allowing to choose the exact set of processors, peripherals, memory, and interfaces needed for an application, without paying for features that aren't needed.

04

High Performance

Giving the ultimate flexibility to achieve the exact performance required for an embedded design, without overpaying for high clock frequency, power-hungry off-the-shelf processors.

Benefits of Nios II Embedded Processors

05

JTAG Debug Module

The architecture supports a Joint Test Action Group (JTAG) debug module that provides on-chip emulation features to control the processor remotely from a host PC.

06

Hardware Acceleration

Similar to hardware accelerators, custom instructions allow Nios II processor designers to increase system performance by offloading portions of the software code to hardware functions.

07

Nios® II Processor Cores

Used by more designers than any other soft processor in the world, Nios II embedded processors remain the industry-standard processor for FPGA design.

Nios II Processor Family

The NIOS II family consists of three members – fast(II/f), economy(II/e) and standard(II/s) – each optimized for a specific price and performance range. All three cores use the same instruction set architecture (ISA) and are 100% binary code compatible.



Nios II/f (fast)

Optimized for highest performance, optional memory management unit (MMU), or memory protection unit (MPU)



Nios II/e (economy)

Designed for smallest possible logic utilization of FPGAs. This is especially efficient for low-cost FPGA applications.



Nios II/s (standard)

Designed to maintain a balance between performance, size and cost

Nios II Family Specifications

Nios II classic is offered in 3 different configurations: Nios II/f (fast), Nios II/s (standard), and Nios II/e (economy).

Nios II gen2 is offered in 2 different configurations: Nios II/f (fast), and Nios II/e (economy).

Nios II/f	Nios II/s	Nios II/e
<ul style="list-style-type: none">• Separate instruction and data caches (512 B to 64 KB)• Optional tightly coupled memory for instructions and data• Six-stage pipeline to achieve maximum DMIPS/MHz• Single-cycle hardware multiply and barrel shifter• Optional hardware divide option• Dynamic branch prediction• Up to 256 custom instructions and unlimited hardware accelerators• Optional JTAG debug module enhancements, including hardware breakpoints, data triggers, and real-time trace	<ul style="list-style-type: none">• Instruction cache• Optional tightly coupled memory for instructions• Five-stage pipeline• Static branch prediction• Hardware multiply, divide, and shift options• Up to 256 custom instructions• Optional JTAG debug module enhancements, including hardware breakpoints, data triggers, and real-time trace	<ul style="list-style-type: none">• Complete systems in fewer than 700 LEs• Optional debug enhancements• Up to 256 custom instructions• Free, no license required

Nios II Processors Performance Comparison

The Nios II family of processors can be used in Altera's Stratix and Cyclone series of FPGAs.

	NIOS II - fast	NIOS II - standard	NIOS II - economy
Stratix II	225 DMIPS @205MHz 1319 ALUTS Stratix 2S60-C3	133DMIPS @180MHz 1029 ALUTS Stratix 2S60-C3	31 DMIPS @209MHz 483 ALUTS Stratix 2S60-C3
Stratix	157 DMIPS @143MHz 1808 LEs Stratix 1S80-C5	99 DMIPS @134MHz 1170 LEs Stratix 1S80-C5	23 DMIPS @160MHz 529 LEs Stratix 1S80-C5
Cyclone II	105 DMIPS @126MHz 1595 LEs Cyclone EP2C20-C6	57 DMIPS @110MHz 1033 LEs Cyclone EP2C20-C6	22 DMIPS @159MHz 542 LEs Cyclone EP2C20-C6
Cyclone	101 DMIPS @128MHz 1676 LEs Cyclone EP1C20-C6	105 DMIPS @124MHz 1145 LEs Cyclone EP1C20-C6	16 DMIPS @126MHz 522 LEs Cyclone EP1C20-C6
Cyclone IV (GX)	190 DMIPS @165MHz		30 DMIPS @175MHz

Nios II Industry Benefits and Applications

Application	Nios II Processor Core	Vendor	Description
Power and cost sensitive	Nios II economy core	Intel	<p>With as low as 600 logic elements, the Nios II economy processor core is ideal for microcontroller applications. The Nios II economy processor core, software tools, and device drivers are offered free of charge.</p>
Real time	Nios II fast core	Intel	<p>Absolutely deterministic, jitter free real-time performance with unique hardware real-time feature options</p> <ul style="list-style-type: none">• Vector Interrupt Controller• Tightly Coupled Memory• Custom instructions (ability to use FPGA hardware to accelerate a function)• Supported by industry-leading real-time operating systems (RTOS)• Nios II processor is the ideal real-time processor to use with DSP Builder-based hardware accelerators to provide deterministic, high performance real-time results
Applications processing	Nios II fast core	Intel	<p>With a simple configuration option, the Nios II fast processor core can use a memory management unit (MMU) to run embedded Linux* operating system. Both open source and commercially supported versions of Linux for Nios II processors are available.</p>

Nios II Development Processes

Development is hosted inside an Altera application called the Embedded Design Suite (EDS). The EDS contains a complete integrated development environment to manage both hardware and software in two separate steps:

01. Hardware Generation

Nios II hardware designers use the Qsys system integration tool, a component of the Quartus-II package, to configure and generate a Nios system.

The configuration graphical user interface (GUI) allows users to choose the Nios-II's feature-set, and to add peripheral and I/O-blocks (timers, memory-controllers, serial interface, etc.) to the embedded system.

Qsys is replacing the older SOPC (System-on-a-Programmable-Chip) Builder, which could also be used to build a Nios II system, and is being recommended for new projects.

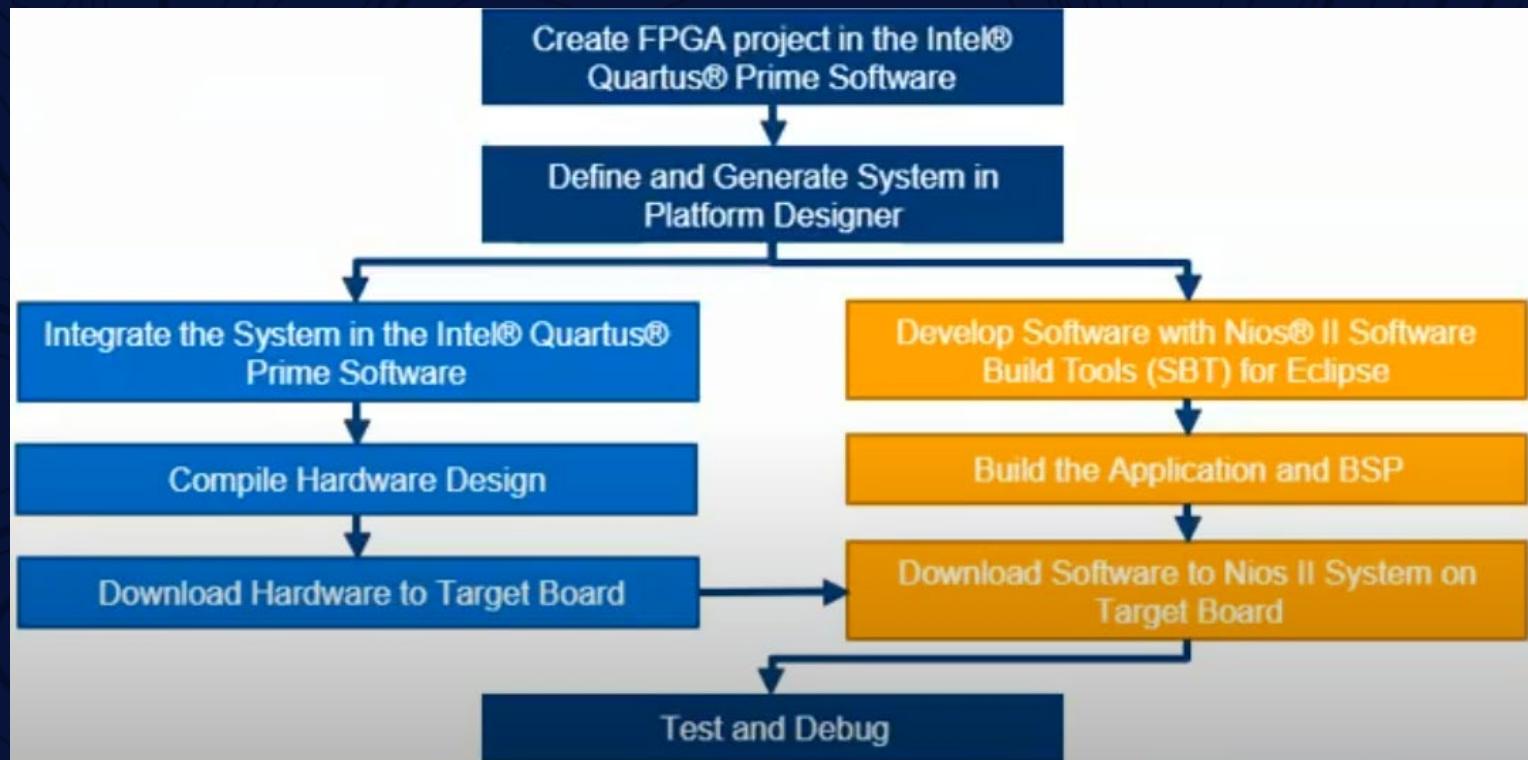
02. Software Creation

EDS is based on the Eclipse IDE, the EDS includes a C/C++ compiler, debugger, and an instruction-set simulator.

EDS allows programmers to test their application in simulation, or download and run their compiled application on the actual FPGA host.

Because the C/C++ development-chain is based on GCC, the vast majority of open source software for Linux compiles and runs with minimal or no modification.

Nios II System Development Design Flow



03

Altera DE2-115

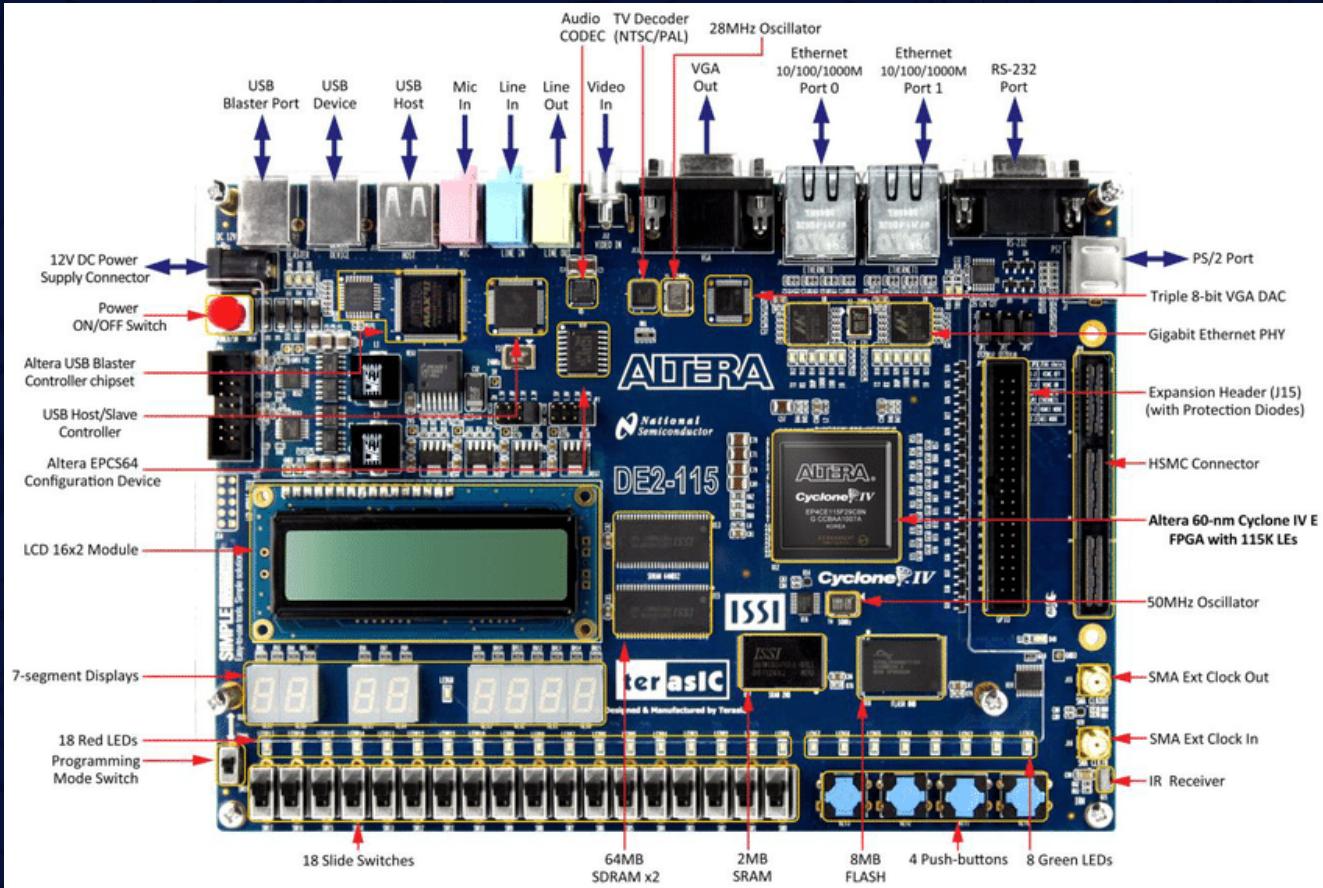
Development and Education Board

Altera DE2-115

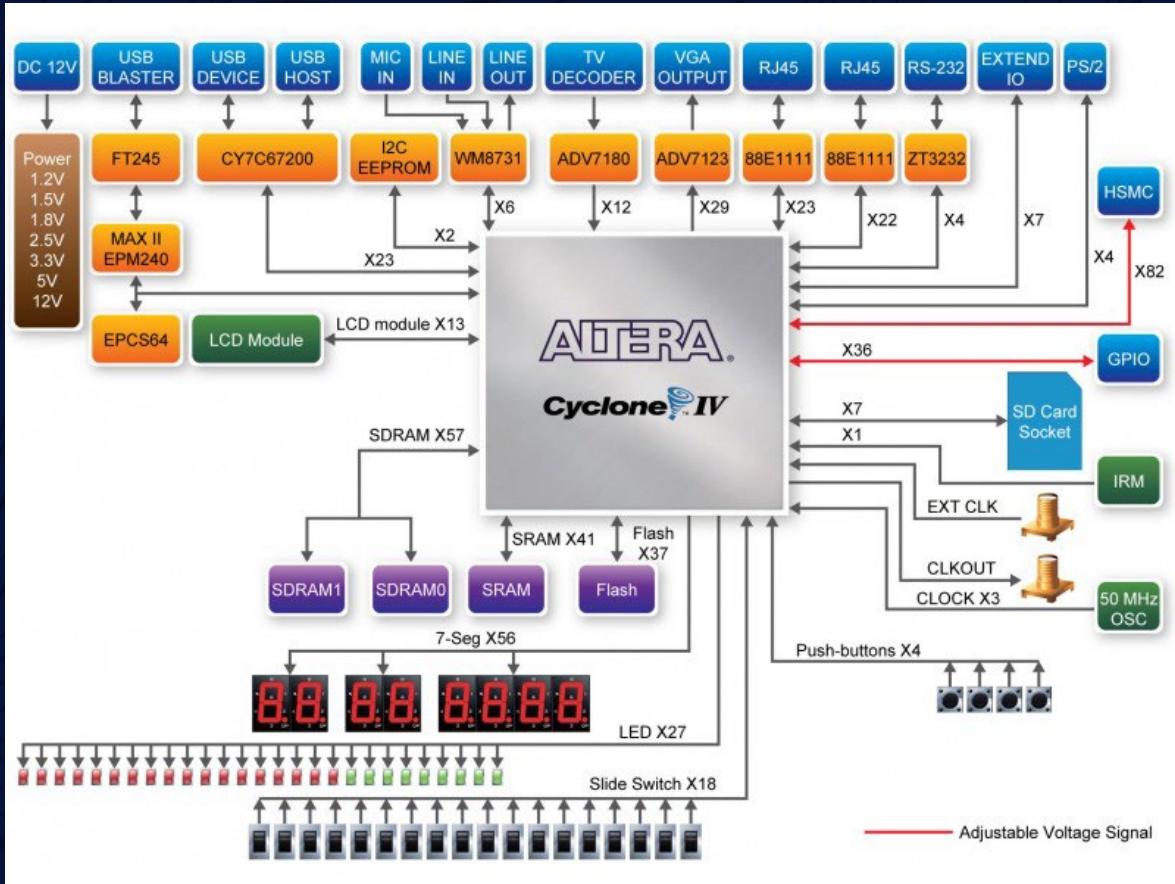
The latest Altera DE2-115 features the Cyclone IV E device. Responding to increased versatile low-cost spectrum needs driven by the demand for mobile video, voice, data access, and the hunger for high-quality images, it offers an optimal balance of low cost, low power and a rich supply of logic, memory and DSP capabilities.

The Cyclone EP4CE115 device equipped on the DE2-115 features 114,480 logic elements (LEs), the largest offered in the Cyclone IV E series, up to 3.9-Mbits of RAM, and 266 multipliers. In addition, it delivers an unprecedented combination of low cost and functionality, and lower power compared to previous generation Cyclone devices.

Altera DE2-115 Board Architecture



Altera DE2-115 Block Diagram



04

Cpulator

Altera DE2-115 Web-based Simulation

What is a CPU Simulator?

Simulation allows running and debugging programs without needing to use a hardware board. When loading a program onto the simulator, the program will behave (almost) identically to the same program running on real hardware.

A simulator has several advantages over hardware. It is more convenient and easily accessible than a hardware board. Also, compared to hardware, a simulator has greater visibility into the system state and can warn about suspicious behaviors, such as self-modifying code or making debugging easier.

Popular CPU Simulators

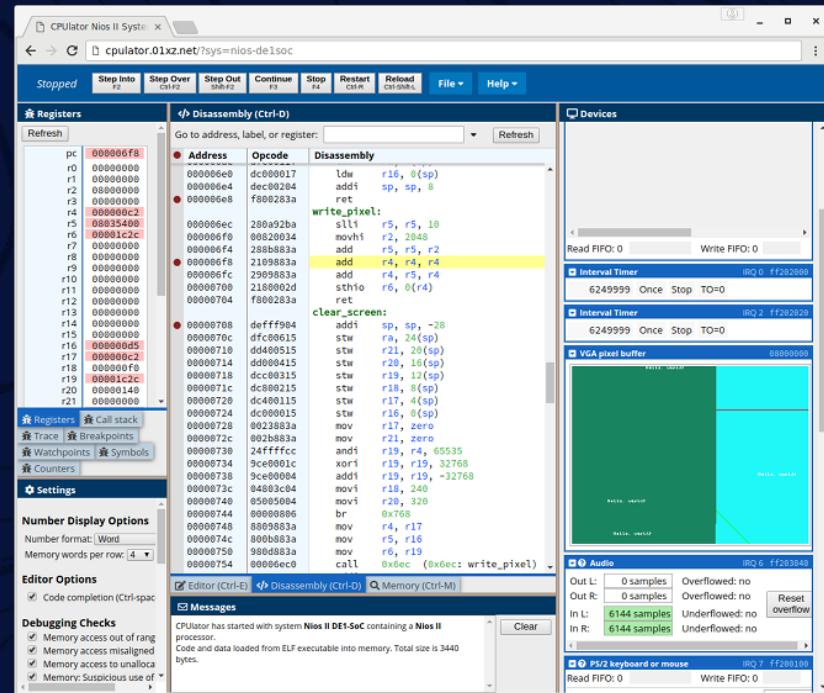
There are many CPU simulators in the industry, including MARS 4.5, QtSPIM 9.1.20, ARMSim 1.91, but each of them have specific instruction set support, and all them require a Desktop installation, except for CPULator which is web-based.

Example of a Simulator: CPULATOR

CPULATOR is a simulator and debugger of a computer system(CPU, memory, and I/O devices) that runs inside a web browser.

The simulated systems are based on Nios II, ARMv7 or SPIM (MIPS).

CPULATOR's design was based on the computer systems from the Altera Monitor Program (Nios II and ARMv7) or SPIM (MIPS).



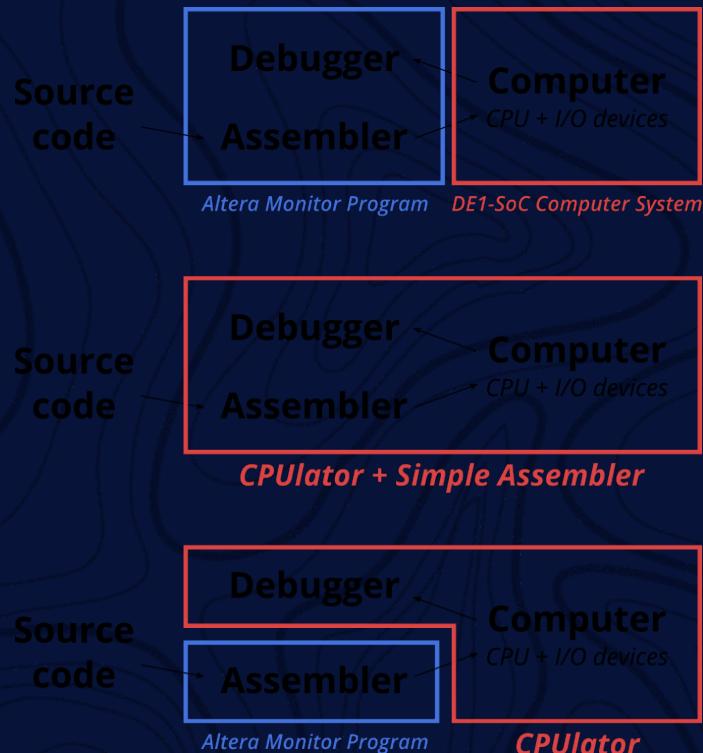
CPULATOR'S DESIGN

There are four main components in a typical development flow for simple assembly programs:

- Source code written by the user.
- An assembler (or compiler for a higher-level language) to transform the source code into executable machine code
- A computer system to run the machine code
- A debugger to observe the behavior of the program running on the computer.

Internally, the CPULATOR simulator simulates executable machine code (it does not directly simulate assembly source code).

The built-in editor and assembler is limited to working with a single assembly-language source file. If your program requires more than one file or is written in another language (e.g. C), you must compile the program yourself and simulate the compiled executable.



Compilation with the Built-in Assembler

The built-in assembler is an interface to the GNU assembler. When Compile and Load is clicked, the contents of the Editor is uploaded to the server, saved as a single file, then run through the GNU assembler. The server then sends back the ELF executable and any assembler output messages, and the executable is loaded into the simulator.

Because the ELF file is generated at the server, there are limits on the size of the compiled executable (currently 12 MB). If your program is bigger than a megabyte (which sometimes happens if it includes images or audio as initialized data), it is usually faster to compile your program locally than to download megabytes at every compile.

If your program is substantially bigger than this, be aware that the executable size causes the simulator itself to consume memory to hold the executable. Don't expect the simulator to ever work with gigabyte-sized executables.

Compilation using Altera Monitor Program

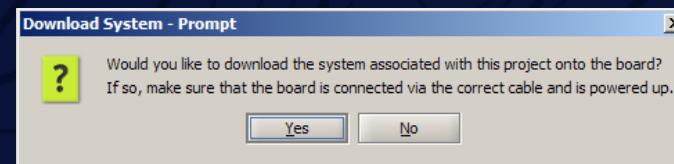
The Altera Monitor Program can be used to compile projects without being connected to an FPGA board, as long as actions that interact with the board are avoided.

When creating a project, specify the "DE2-115 Computer" system. This is important because the compiler makes certain assumptions(e.g., system memory size)that depend on the computer system, and should match what's being simulated.

When prompted, do not download the computer system to the board.

You can now "Compile" your project, but do not use "Load" nor "Compile & Load". Compiling the program produces a filename.elf executable file in your project directory.

To load the executable into the CPUlator simulator, use File → Load ELF Executable and choose this executable file.



Simulation 1: Included Sample VGA test

The purpose of this program is to test a VGA display by cycling through a color spectrum.

The screenshot shows a software development environment for a Nios II processor. The top menu bar includes options like Running, Step Into, Step Over, Continue, Stop, Restart, Reload, File, and Help. The main window is divided into several panes:

- Registers:** Shows the current values of various processor registers (pc, r0-r12) in hexadecimal format.
- Editor (Ctrl-E):** Displays the assembly code for the VGA test program. The code defines constants for screen dimensions (WIDTH, HEIGHT), bytes per row (LOG2_BYTES_PER_ROW, LOG2_BYTES_PER_PIXEL), and buffer addresses (PIXBUF, CHARBUF). It also sets up the stack pointer (sp) at address 0x800000.
- Devices:** Monitors the VGA pixel buffer, which displays a green screen with a small white logo in the center. Other visible devices include an Interval Timer and a Parallel port.
- Messages:** Logs the build process, indicating a successful compilation of the assembly code into an ELF file.

Simulation 1: Included Sample VGA test

Memory
Registers

Editor

VGA output

Refresh

pc	000000d4
r0	00000000
r1	00000000
r2	0000000a
r3	00000001
r4	000000aa
r5	000000d1
r6	7c0cc884
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000

Go to address, label, or register: 000000e4 Refresh

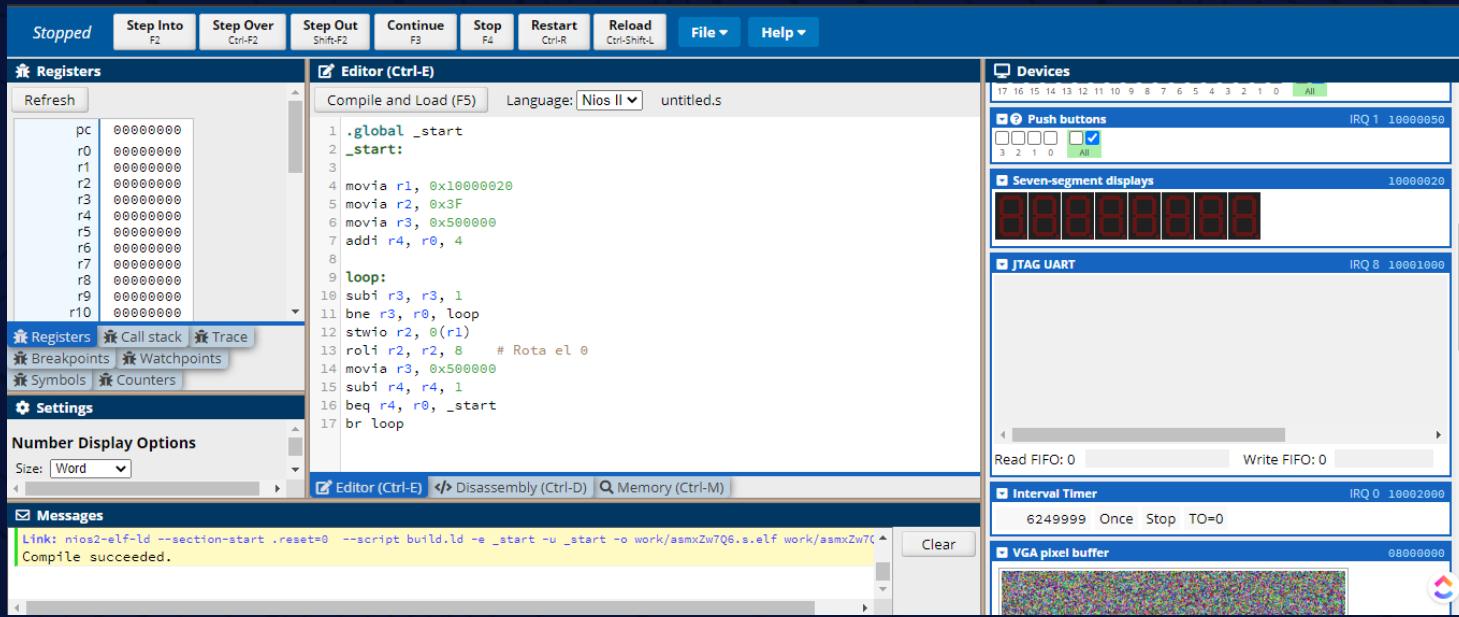
Address	Opcode	Disassembly
000000d4	00c00044	movi r3, 1
000000d8	288a983a	sll r5, r5, r2
000000dc	20c8983a	sll r4, r4, r3
000000e0	290b883a	add r5, r5, r4
000000e4	01020034	movia r4, PIXBUF
000000e8	21000004	movhi r4, 2048
000000ec	290b883a	addi r4, r4, 0 # movia r4, 0x08000000
000000f0	1800021e	bne r3, zero, @xfc
000000f4	29800025	stbio r6, 0(r5) # Write 8-bit pixel
000000f8	f800283a	stbio r6, 0(r5)
000000fc	2980002d	ret
00000100	f800283a	1: stbio r6, 0(r5) # Write 16-bit pixel
		stbio r6, 0(r5)
		ret
		_data:
		end:

VGA pixel buffer 00000000

The screenshot shows a debugger interface with three main panes. The left pane, labeled 'Registers', lists memory registers (pc, r0-r12) with their current values. The middle pane, labeled 'Editor', displays assembly code with addresses, opcodes, and disassembly. A specific instruction at address 000000f0 is highlighted with a yellow background. The right pane, labeled 'VGA output', shows a green square representing the rendered image.

Simulation 2: Zero Rotation

The goal is to design an assembly program that shows on the DE2 board a zero on the first "seven-segment" display and rotate it through the first four displays "seven-segments". Note: Observe the address assignments of the parallel interfaces of seven-segment displays.



Simulation 2: Zero Rotation

Memory
Registers

Refresh	
pc	00000001c
r0	00000000
r1	10000020
r2	00003f00
r3	00008553
r4	00000003
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000

Output



Disassembly

Go to address, label, or register:			Refresh
Address	Opcode	Disassembly	
00000008	00800034	movia r2, 0x3f	
0000000c	10800fc4	movhi r2, 0	
00000010	00c01434	addi r2, r2, 63	# movia r2, 0x0000003f
00000014	18c00004	movia r3, 0x500000	
00000018	01000104	movhi r3, 80	
		addi r3, r3, 0	# movia r3, 0x00500000
0000001c	18ffffc4	addi r4, r0, 4	
		movi r4, 4	
		loop:	
		subi r3, r3, 1	
		loop:	
		addi r3, r3, -1	
00000020	183fffe1	bne r3, r0, loop	
		bne r3, zero, 0x1c (0x1c: loop)	
00000024	08800035	stwio r2, 0(r1)	
		stwio r2, 0(at)	
00000028	1004123a	roli r2, r2, 8	# Rota el 0
		roli r2, r2, 8	

Simulation 3: Green LED Train

Design an assembly program that alternately turns on each of the the green leds of the DE2 board. Note: Observe address assignment of the parallel interface NIOS II processor address space of green leds.

The screenshot shows the Nios II IDE interface with the following components:

- Toolbar:** Stopped, Step Into (F2), Step Over (Ctrl-F2), Step Out (Shift-F2), Continue (F3), Stop (F4), Restart (Ctrl-R), Reload (Ctrl-Shift-L), File ▾, Help ▾.
- Registers:** Shows the state of registers pc, r0 through r10. pc is at 00000000, and r1 is also at 00000000.
- Editor (Ctrl-E):** Language: Nios II, untitled.s [changed since save]. The assembly code is as follows:

```
.global _start
_start:
    movia r1, 0x10000010    # Principio leds verdes
    addi r2, r0, 1
    movia r3, 0x500000    # Un segundo de tiempo
    addi r4, r0, 9

loopP:
    subi r3, r3, 1
    bne r3, r0, loopP
    stwio r2, @r1
    roli r2, r2, 1
    subi r4, r4, 1
    movia r3, 0x500000
    beq r4, r0, _start
    br loopP
```

- Devices:** Shows the state of various I/O components:
- LEDs:** Address 10000000, all 16 LEDs are off.
- Push buttons:** Address 10000050, IRQ 1, button 0 is pressed (indicated by a checked box).
- Seven-segment displays:** Address 10000020, displays show "88888888".
- JTAG UART:** Address 10001000, IRQ 8, no data displayed.
- Messages:** Link: nios2-elf-ld --section-start .reset=0 --script build.ld -e _start -u _start -o work/asmVcyp0.s.elf work/asmVcyp. Compile succeeded.

Simulation 3: Green LED Train

Memory
Registers

Registers	
pc	000000018
r0	00000000
r1	100000010
r2	00000001
r3	0006e176
r4	00000009
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000

LED output

LEDs 10000010

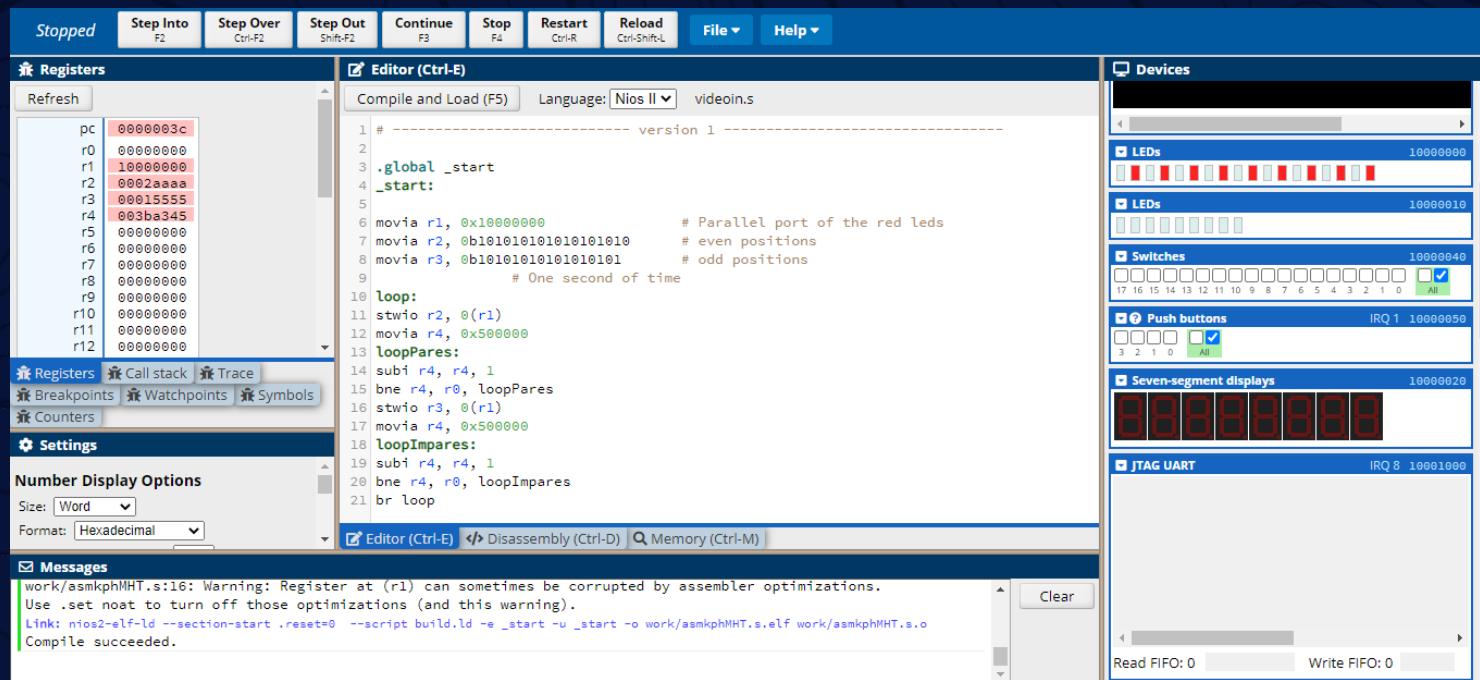


Disassembly

Go to address, label, or register:			Refresh
Address	Opcode	Disassembly	
00000004	084000404	addi at, at, 10 # movia at, 0x100000010	
00000008	00800044	addi r2, r0, 1	
0000000c	00c01434	movi r2, 1	
00000010	18c00004	movia r3, 0x500000 # Un segundo de tiempo	
00000014	01000244	movhi r3, 80	
00000018	18ffffc4	addi r3, r3, 0 # movia r3, 0x00500000	
0000001c	183ffe1e	addi r4, r0, 9	
00000020	08800035	movi r4, 9	
00000024	1004107a	loopP: subi r3, r3, 1	
		bne r3, r0, loopP	
		bne r3, zero, 0x18 (0x18: loopP)	
		stwio r2, 0(r1)	
		stwio r2, 0(at)	
		roli r2, r2, 1	
		subi r4, r4, 1	

Simulation 4 : Alternating Red LEDs

We designed an assembly program that alternately turns on all the red LEDs in even positions and all red LEDs in odd positions.



Simulation 4 : Alternating Red LEDs

Memory
Registers

Refresh	
pc	000000024
r0	000000000
r1	100000000
r2	0002aaaa
r3	00015555
r4	004c7d97
r5	000000000
r6	000000000
r7	000000000
r8	000000000
r9	000000000
r10	000000000

LED output



Disassembly

Go to address, label, or register: 00000000 Refresh		
Address	Opcode	Disassembly
00000020	21000004	addi r4, r4, 0 # movia r4, 0x00500000
13	loopPares:	
14	subi r4, r4, 1	
loopPares:		
00000024	213ffffc4	addi r4, r4, -1
15	bne r4, r0, loopPares	
00000028	203fffele	bne r4, zero, 0x24 (0x24: loopPares)
16	stwio r3, 0(r1)	
17	stwio r3, 0(at)	
0000002c	08c00035	movia r4, 0x500000
18	movhi r4, 80	
00000030	01001434	addi r4, r4, 0 # movia r4, 0x00500000
19	loopImpares:	
20	subi r4, r4, 1	
loopImpares:		
00000034	21000004	addi r4, r4, -1
21	bne r4, r0, loopImpares	
00000038	213ffffc4	bne r4, zero, 0x38 (0x38: loopImpares)
22	br 0x18 (0x18: loop)	
0000003c	203fffele	
00000040	003ff506	

References

https://en.wikipedia.org/wiki/Nios_II

<https://web.archive.org/web/20101225092752/http://www.altera.com/products/ip/processors/nios2/ni2-index.html>

<https://www.intel.com/content/www/us/en/products/details/fpga/nios-processor/ii.html>

https://www.youtube.com/watch?v=IG2I56NKLQk&ab_channel=IntelFPGA

<https://cpulator.01xz.net/doc/#implementation>