Introduction to MicroBlaze

LECTURE 5

IOULIIA SKLIAROVA

MicroBlaze Processor

Soft processor
- ~1 900-7 000 logic cells (~1 200-4 500 LUTs) - estimates
- 63 400 LUTs available in Artix-7 XC7A100T

RISC architecture
- utilizes a small, highly-optimized set of instructions, rather than a more specialized set of instructions often found in other types of architectures

32/64-bit architecture
- Thirty-two 32-bit or 64-bit general purpose registers

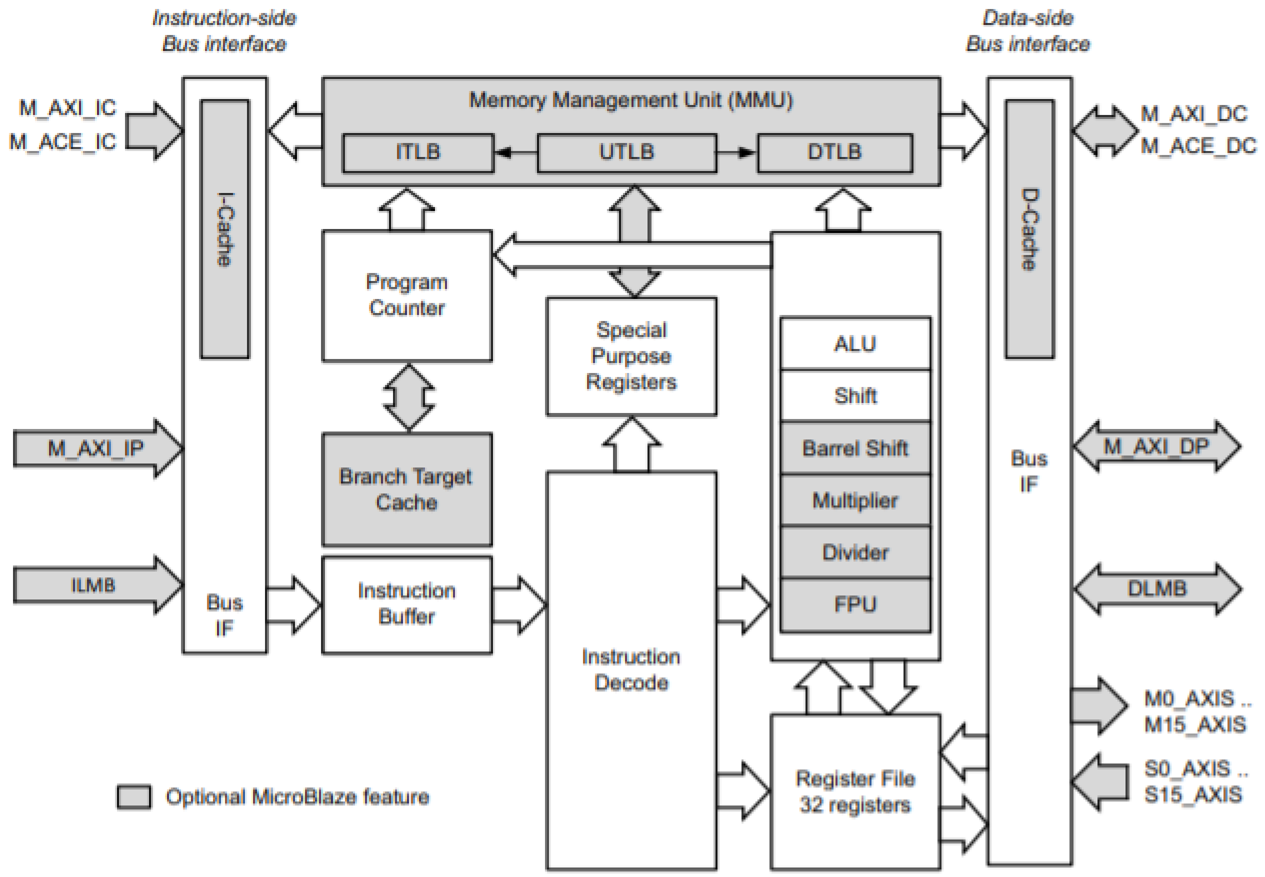
In production since 2002

Supported in
- 7-series/UltraScale/UltraScale+ devices

Three preset configurations:
- a simple microcontroller running bare-metal applications (~200 MHz in Artix-7);
- a real-time processor running FreeRTOS (~170 MHz);
- an application processor with a memory management unit running Linux (~140 MHz)

MicroBlaze Overview

MicroBlaze Processor Reference Guide - UG984



MicroBlaze Features

The fixed feature set of the processor includes:

- Thirty-two 32-bit or 64-bit general purpose registers
- 32-bit instruction word with three operands and two addressing modes
- Default 32-bit address bus, extensible to 64 bits
- Single issue pipeline

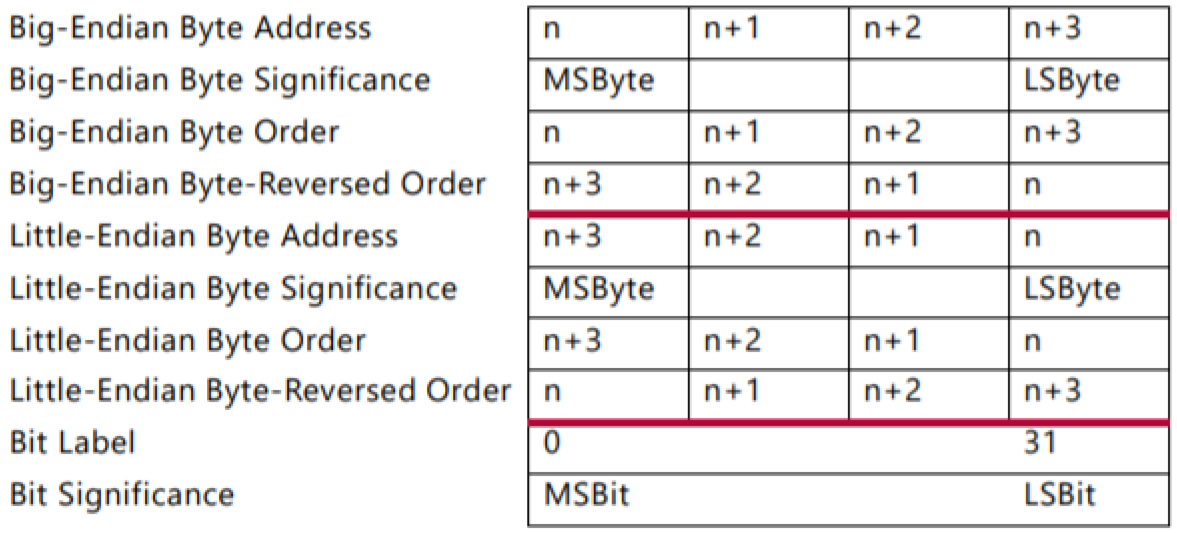
Configurable features

- Processor pipeline depth
- Floating-point unit (FPU)
- Hardware divider
- Area or speed optimized
- 64-bit mode
- …

MicroBlaze Data Types

The MicroBlaze processor uses Big-Endian or Little-Endian (default) format to represent data, depending on the selected endianness.

The hardware supported data types for 32-bit MicroBlaze are word, half word, and byte. With 64-bit MicroBlaze the data types long and double are also available in hardware.



Big-Endian Byte Address	n	n+1	n+2	n+3
Big-Endian Byte Significance	MSByte			LSByte
Big-Endian Byte Order	n	n+1	n+2	n+3
Big-Endian Byte-Reversed Order	n+3	n+2	n+1	n
Little-Endian Byte Address	n+3	n+2	n+1	n
Little-Endian Byte Significance	MSByte			LSByte
Little-Endian Byte Order	n+3	n+2	n+1	n
Little-Endian Byte-Reversed Order	n	n+1	n+2	n+3
Bit Label	0			31
Bit Significance	MSBit			LSBit

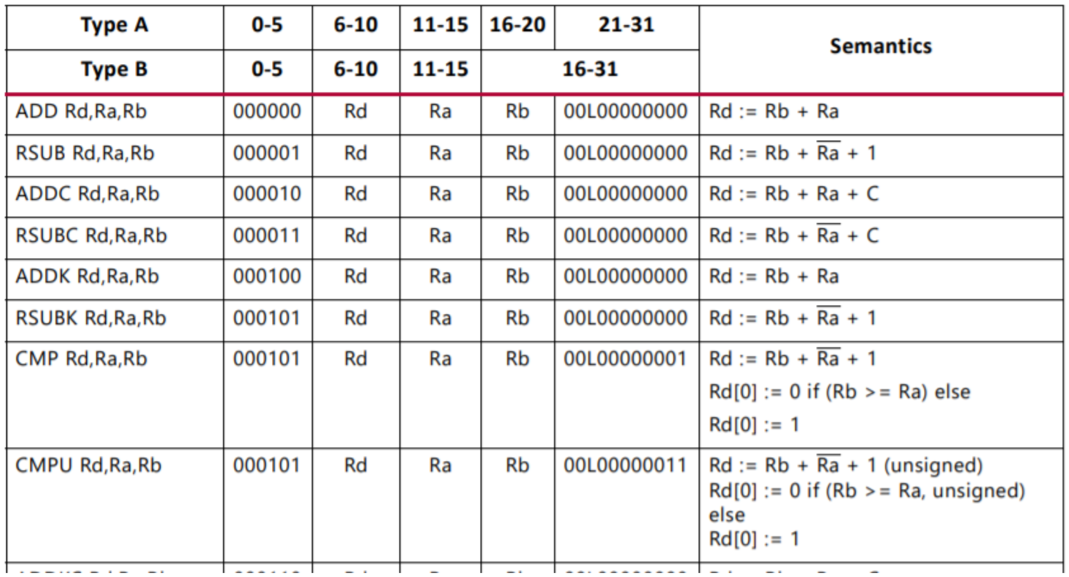
MicroBlaze Instruction Summary

All MicroBlaze instructions are 32 bits and are defined as either Type A or Type B.

Type A instructions have up to two source register operands and one destination register operand.

Type B instructions have one source register and a 16-bit immediate operand. Type B instructions have a single destination register operand.

Table 2-7: **MicroBlaze Instruction Set Summary**



Type A	**0-5**	**6-10**	**11-15**	**16-20**	**21-31**	**Semantics**
Type B	**0-5**	**6-10**	**11-15**	**16-31**		
ADD Rd,Ra,Rb	000000	Rd	Ra	Rb	00L00000000	Rd := Rb + Ra
RSUB Rd,Ra,Rb	000001	Rd	Ra	Rb	00L00000000	Rd := Rb + $\overline{Ra}$ + 1
ADDC Rd,Ra,Rb	000010	Rd	Ra	Rb	00L00000000	Rd := Rb + Ra + C
RSUBC Rd,Ra,Rb	000011	Rd	Ra	Rb	00L00000000	Rd := Rb + $\overline{Ra}$ + C
ADDK Rd,Ra,Rb	000100	Rd	Ra	Rb	00L00000000	Rd := Rb + Ra
RSUBK Rd,Ra,Rb	000101	Rd	Ra	Rb	00L00000000	Rd := Rb + $\overline{Ra}$ + 1
CMP Rd,Ra,Rb	000101	Rd	Ra	Rb	00L00000001	Rd := Rb + $\overline{Ra}$ + 1 Rd[0] := 0 if (Rb >= Ra) else Rd[0] := 1
CMPU Rd,Ra,Rb	000101	Rd	Ra	Rb	00L00000011	Rd := Rb + $\overline{Ra}$ + 1 (unsigned) Rd[0] := 0 if (Rb >= Ra, unsigned) else Rd[0] := 1

MicroBlaze Pipeline

MicroBlaze instruction execution is pipelined.

For most instructions, each stage takes one clock cycle to complete.

Consequently, the number of clock cycles necessary for a specific instruction to complete is equal to the number of pipeline stages, and one instruction is completed on every cycle in the absence of data, control or structural hazards.

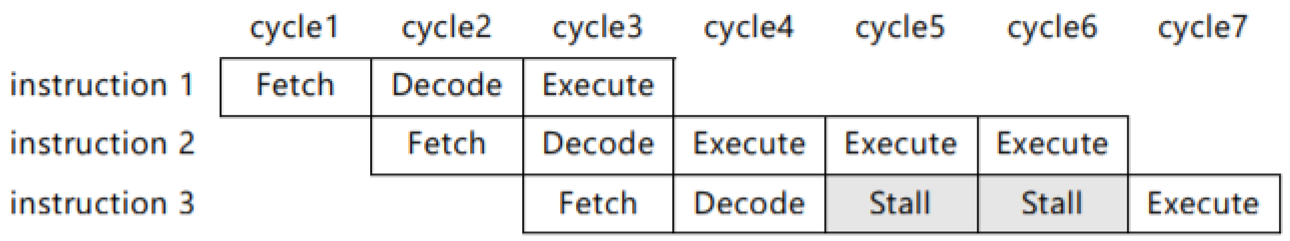
- A data hazard occurs when the result of an instruction is needed by a subsequent instruction. This can result in stalling the pipeline, unless the result can be forwarded to the subsequent instruction. The MicroBlaze GNU Compiler attempts to avoid data hazards by reordering instructions during optimization.
- A control hazard occurs when a branch is taken, and the next instruction is not immediately available. This results in stalling the pipeline. MicroBlaze provides delay slot branches and the optional branch target cache to reduce the number of stall cycles.
- A structural hazard occurs for a few instructions that require multiple clock cycles in the execute stage or a later stage to complete. This is achieved by stalling the pipeline.

Three Stage Pipeline

With the MicroBlaze is optimized for area, the pipeline is divided into three stages to minimize hardware cost: Fetch, Decode, and Execute.

The three stage pipeline does not have any data hazards.

Pipeline stalls are caused by control hazards, structural hazards due to multi-cycle instructions, memory accesses using slower memory, instruction fetch from slower memory, or stream accesses.



	cycle1	cycle2	cycle3	cycle4	cycle5	cycle6	cycle7
instruction 1	Fetch	Decode	Execute				
instruction 2		Fetch	Decode	Execute	Execute	Execute	
instruction 3			Fetch	Decode	Stall	Stall	Execute

Five Stage Pipeline

With the MicroBlaze is optimized for performance, the pipeline is divided into five stages to maximize performance: Fetch (IF), Decode (OF), Execute (EX), Access Memory (MEM), and Writeback (WB).

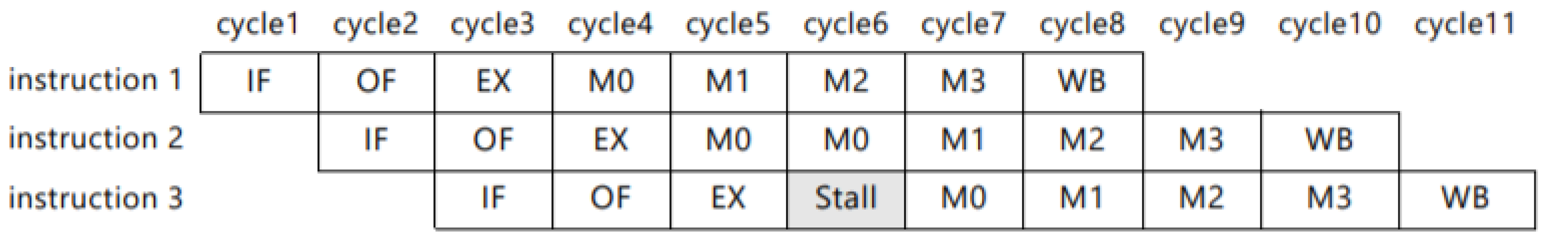
Pipeline stalls are caused by data hazards, control hazards, structural hazards due to multicycle instructions, memory accesses using slower memory, instruction fetch from slower memory, or stream accesses.

	cycle1	cycle2	cycle3	cycle4	cycle5	cycle6	cycle7	cycle8	cycle9
instruction 1	IF	OF	EX	MEM	WB				
instruction 2		IF	OF	EX	MEM	MEM	MEM	WB	
instruction 3			IF	OF	EX	Stall	Stall	MEM	WB

Eight Stage Pipeline

With the MicroBlaze is optimized for frequency, the pipeline is divided into eight stages to maximize possible frequency: Fetch (IF), Decode (OF), Execute (EX), Access Memory 0 (M0), Access Memory 1 (M1), Access Memory 2 (M2), Access Memory 3 (M3) and Writeback (WB).

Pipeline stalls are caused by data hazards, control hazards, structural hazards, memory accesses using slower memory, instruction fetch from slower memory, or stream accesses.



	cycle1	cycle2	cycle3	cycle4	cycle5	cycle6	cycle7	cycle8	cycle9	cycle10	cycle11
instruction 1	IF	OF	EX	M0	M1	M2	M3	WB			
instruction 2		IF	OF	EX	M0	M0	M1	M2	M3	WB	
instruction 3			IF	OF	EX	Stall	M0	M1	M2	M3	WB

Memory Architecture

MicroBlaze is implemented with a Harvard memory architecture; instruction and data accesses are done in separate address spaces.

- The instruction address space has a 32-bit virtual address range with 32-bit MicroBlaze (that is, handles up to 4GB of instructions), and can be extended up to a 64-bit physical address range.
- The data address space has a default 32-bit range, and can be extended up to a 64-bit range (that is, handles from 4GB to 16EB of data).

The instruction and data memory ranges can be made to overlap by mapping them both to the same physical memory. The latter is necessary for software debugging.

Both instruction and data interfaces of MicroBlaze are default 32 bits wide and use big endian or little endian, bit-reversed format, depending on the selected endianness. MicroBlaze supports word, halfword, and byte accesses to data memory.

Data accesses must be aligned (word accesses must be on word boundaries, halfword on halfword boundaries), unless the processor is configured to support unaligned exceptions. All instruction accesses must be word aligned.

MicroBlaze Interfaces

MicroBlaze does not separate data accesses to I/O and memory (it uses memory-mapped I/O).

The processor has up to three interfaces for memory accesses:

- Local Memory Bus (LMB) providing single-cycle access to on-chip dual-port block RAM.
- Advanced eXtensible Interface (AXI4) for connection to both on-chip and off-chip peripherals and memory.
- Advanced eXtensible Interface (AXI4) or AXI Coherency Extension (ACE) for cache coherent connections to memory.

MicroBlaze also supports up to 16 AXI4-Stream interface ports, each with one master and one slave interface.

The interfaces on MicroBlaze are 32 bits wide.

MicroBlaze Core Interfaces

M_AXI_DP: Peripheral Data Interface, AXI4-Lite or AXI4 interface

DLMB: Data interface, Local Memory Bus (BRAM only)

M_AXI_IP: Peripheral Instruction interface, AXI4-Lite interface

ILMB: Instruction interface, Local Memory Bus (BRAM only)

M0_AXIS..M15_AXIS: AXI4-Stream interface master direct connection interfaces

S0_AXIS..S15_AXIS: AXI4-Stream interface slave direct connection interfaces

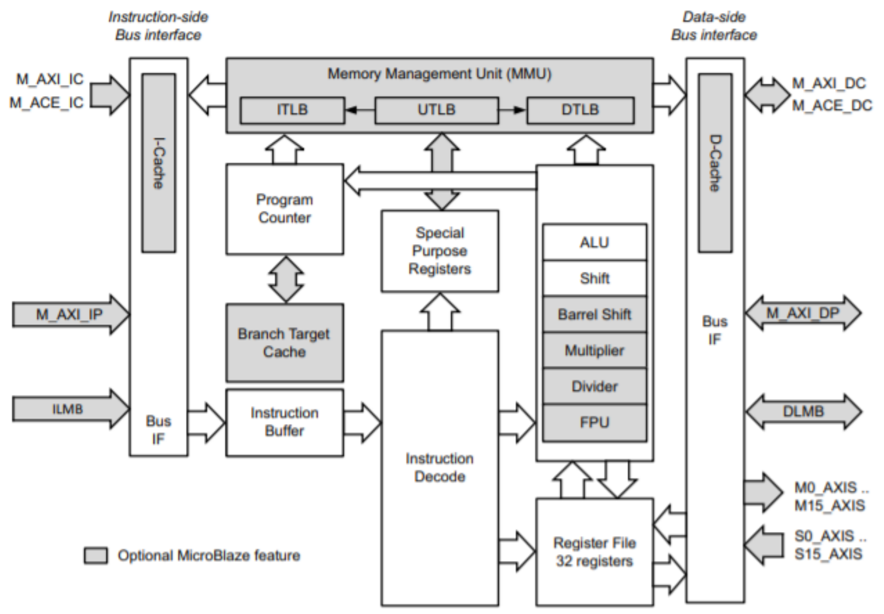
M_AXI_DC: Data-side cache AXI4 interface

M_ACE_DC: Data-side cache AXI Coherency Extension (ACE) interface

M_AXI_IC: Instruction-side cache AXI4 interface

M_ACE_IC: Instruction-side cache AXI Coherency Extension (ACE) interface

Core: Miscellaneous signals for: clock, reset, interrupt, debug, trace



AXI

Advanced eXtensible Interface is a point to point interconnect that is designed for high performance, high speed microcontroller systems.

The **AXI** protocol is based on a point to point interconnect to avoid bus sharing and therefore allow higher bandwidth and lower latency.

There are three types of AXI4 interfaces:

- AXI4-Lite—for simple, low-throughput memory-mapped communication, providing a register-like structure with reduced features and complexity (1 transfer per transaction)
- AXI4—for high-performance memory-mapped requirements (up to 256 data transfers)
- AXI4-Stream—for high-speed streaming data (unlimited amount of data)

The AXI specifications describe an interface between a single AXI master and a single AXI slave.

AXI interconnects allow multiple masters and/or multiple slaves to interface with each other.

In reality, interconnects contain slave interfaces that connect to AXI masters and master interfaces that connect to AXI slaves. What goes on in an interconnect—i.e., how different masters communicate to different slaves—depends on the implementation. Interconnects can allow a shared address bus, shared data bus, both shared, or neither shared.

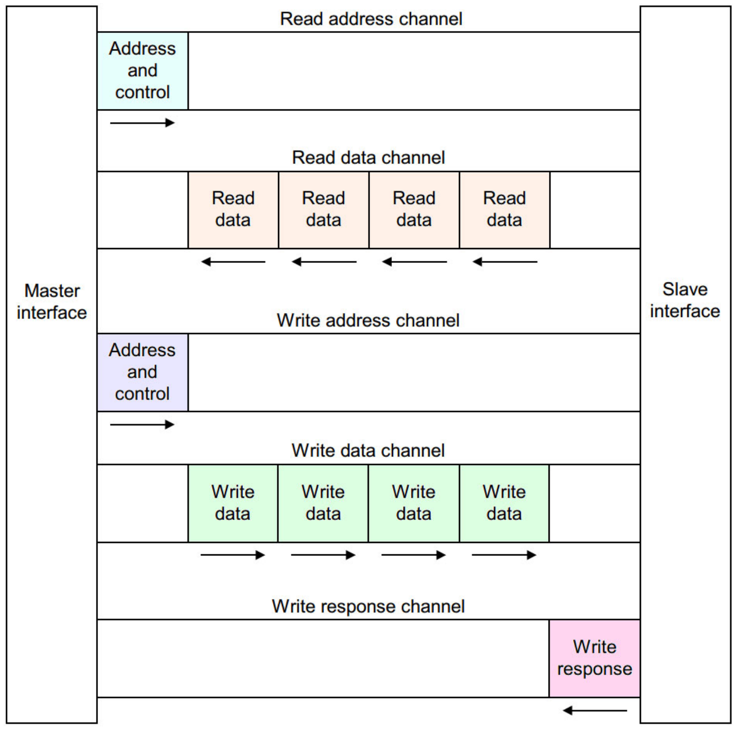
AXI4 and AXI4-Lite Channels

There are five independent channels between an AXI master and slave.

The address channels are used to send address and control information while performing a basic handshake between master and slave.

A master reads data from and writes data to a slave. Read response information is placed on the read data channel, while write response information has a dedicated channel. This way the master can verify a write transaction has been completed.

Every exchange of data is called a transaction. A transaction includes the address and control information, the data sent, as well as any response information. The actual data is sent in bursts which contain multiple transfers.



Debug Overview

MicroBlaze features a debug interface to support JTAG based software debugging tools.

The debug interface is designed to be connected to the Xilinx Microprocessor Debug Module (MDM) core, which interfaces with the JTAG port of Xilinx FPGAs.

To be able to download programs, set software breakpoints and disassemble code, the instruction and data memory ranges must overlap, and use the same physical memory.

MicroBlaze-based Project Example

MicroBlaze processor

Local instruction and data memory

Clock and reset units

General purpose I/O ports (connected to Nexys-4 LEDs, switches, buttons and 7-segment displays)

RS232 UART

AXI interconnect (crossbar for microprocessor and peripherals interconnect)

Interrupt controller

Board File Installation

Configure the board:

- Do not use Install/Update feature when creating a new project
- Download the board files from **Ficheiro "Nexys4.rar" (board support files da Nexys-4 para Xilinx Vivado)** on eLearning
- Copy nexys4 folder to Vivado installation folder (C:\Xilinx\Vivado\2020.2\data\boards\board_files)
  - By default this folder contains XML files for different FPGA boards manufactured by Xilinx
  - XML files define various interfaces on the board, such as slide switches, push buttons, LEDs, USB-UART, memory, Ethernet etc.
- Restart Vivado
- You are now ready to start a new IP Integrator based Vivado project for the Nexys-4 board

Final Remarks

At the end of this lecture you should be able to:

- have a generic idea of MicroBlaze processor

To do:

- Complete the lab. 4