



# SoC Microblaze

**AES – IP**  
**PEM**

Center of Electrical Engineering and Informatics  
Federal University of Campina Grande



- Sumário
  - Introdução
  - O processador **Microblaze**
  - A placa de desenvolvimento **ARTY**
  - Arquitetura do SoC
  - Criando SoC no **Vivado**
    - “Empacotando” IP AES no **Vivado**
  - **API**
  - **SDK** / Teste do IP com o SoC desenvolvido
  - **Resultados**

## • Introdução

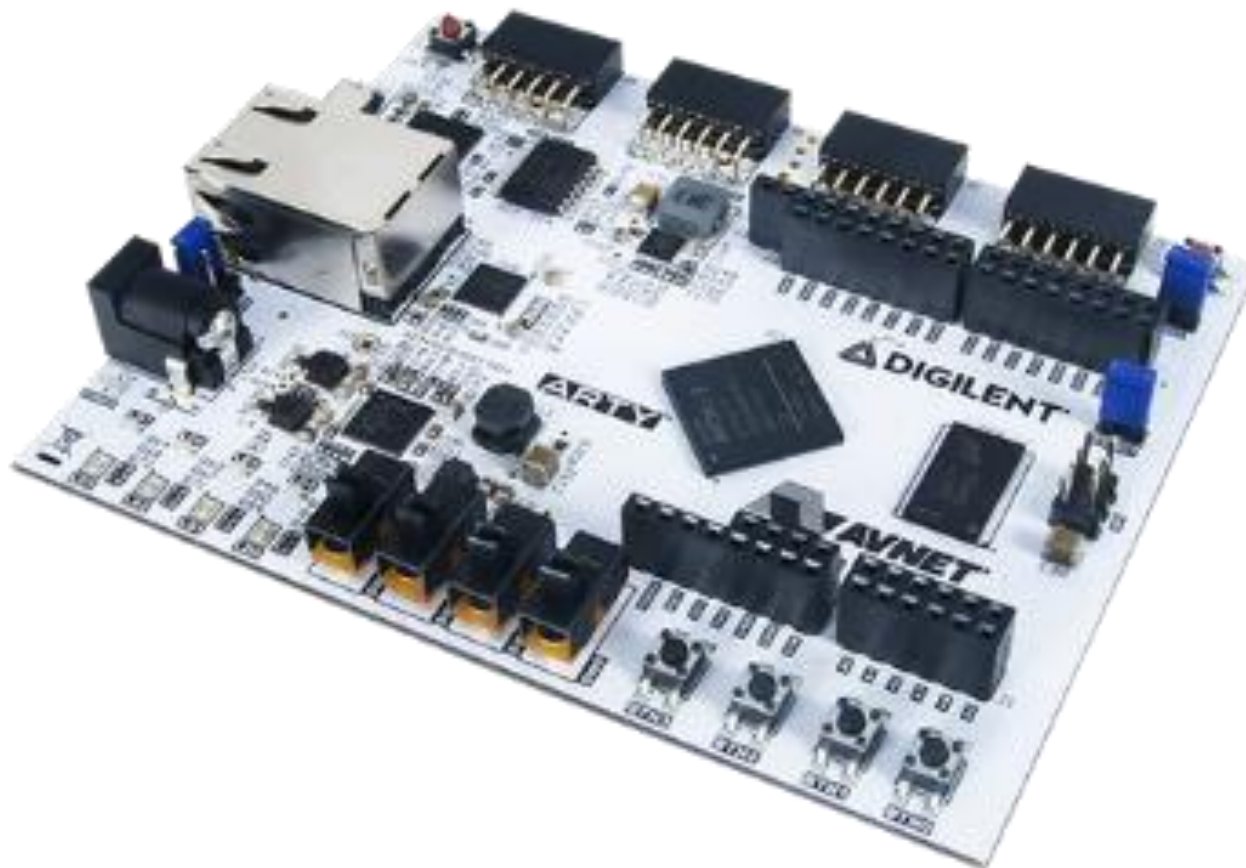
- UVM apresentou bons resultados, isto é: muitos maths e **nenhum mismatch** para a cobertura definida.
- Como uma forma alternativa de teste, foi estabelecido o uso do IP em um SoC

- O processador **Microblaze**

- Microblaze é um Soft Processor Core desenvolvido pela
- Arquitetura **Harvard 32-bits**
- É um **RISC**
- 32 Registradores
- Suporte para Big-endian ou **Little-endian**
- **Interface AXI - AMBA**
- **Não é Open-Source**
- O **Microblaze** apresenta características similares ao **RISC-V**, core escolhido para solução final do projeto
- Dessa forma temos uma metodologia inicial de desenvolvimento do SoC desejado



- A placa de desenvolvimento **ARTY**



- A placa de desenvolvimento **ARTY**

- FPGA embarcada: Artix-35T
  - 33,280 logic cells in 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops);
  - Programmable over JTAG and Quad-SPI Flash
- **Suporte para MicroBlaze**
- System Features:
  - 256MB DDR3L with a 16-bit bus @ 667MHz
  - USB-JTAG Programming circuitry (USB Micro cable required, NOT INCLUDED).

- A placa de desenvolvimento **ARTY**

- System Connectivity:
  - 10/100 Mbps Ethernet
  - **USB-UART Bridge**
- **Interaction and Sensory Devices**
  - 4 Switches
  - 4 Buttons
  - 1 Reset Button
  - 4 LEDs
  - 4 RGB LEDs

## • Arquitetura do SoC

- A Xilinx disponibiliza IPs para os periféricos presentes na ARTY
- Foram escolhidos a UART, para imprimir dados na tela, os push-buttons, switches e LEDs, ou seja, UART + GPIO + AES foram os IPs escolhidos para o SoC com essa FPGA
- Uma memória DDR3 também presente no hardware será usada por ter seu IP pronto
- As próximas etapas mostrará a criação do SoC juntamente com a criação do IP AES e as formas que os testes foram realizados



- Criando **SoC** no **Vivado** (passo-a-passo)
  - Nessa seção será mostrado como criar o SoC com Microblaze, os periféricos presentes na **ARTY** e o **IP AES**.
  - Com o **Vivado** aberto, Create New Project, siga em Next até a escolha da Board (figura seguinte)

New Project x

### Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Select: Parts Boards

Filter/ Preview



Vendor: digilentinc.com

Display Name: Arty

Board Rev: C.0



Reset All Filters

Search: Q

Display Name	Vendor	Board Rev	Part	I/O Pin Count	File Version	Bloc RAM
 Arty	digilentinc.com	C.0	 xc7a35ticsg324-1L	324	1.1	50

?
< Back
Next >
Finish
Cancel

x
New Project

### New Project Summary

i A new RTL project named 'soc\_ublazepem' will be created.

! No source files or directories will be added. Use Add Sources to add them later.

! No Configurable IP files will be added. Use Add Sources to add them later.

! No constraints files will be added. Use Add Sources to add them later.

i The default part and product family for the new project:

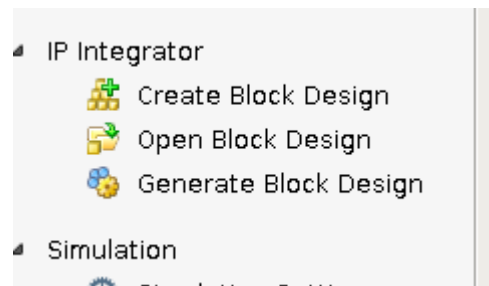
Default Board: Arty  
 Default Part: xc7a35ticsg324-1L  
 Product: Artix-7  
 Family: Artix-7  
 Package: csg324  
 Speed Grade: -1L

To create the project, click Finish

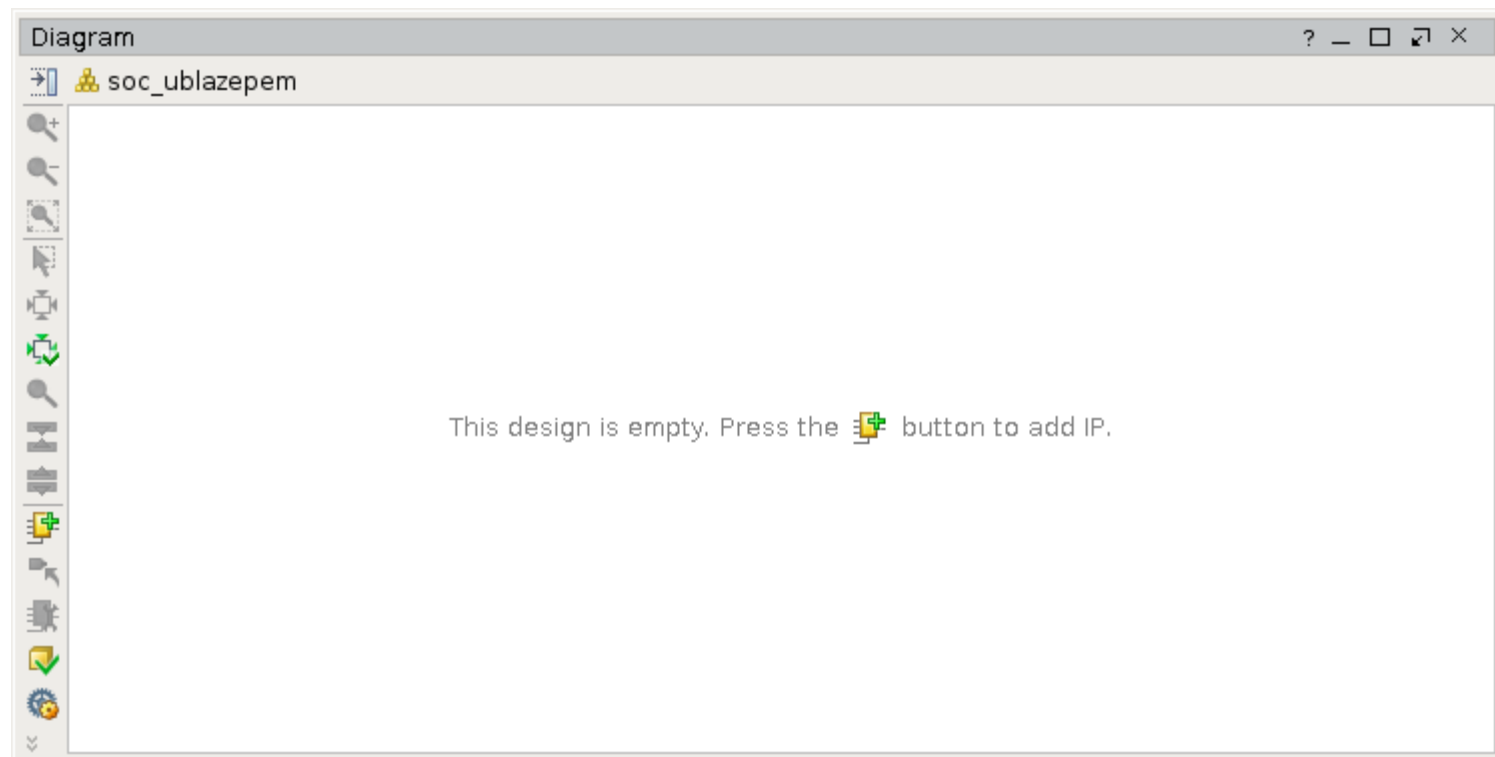
?

< Back
Next >
Finish
Cancel

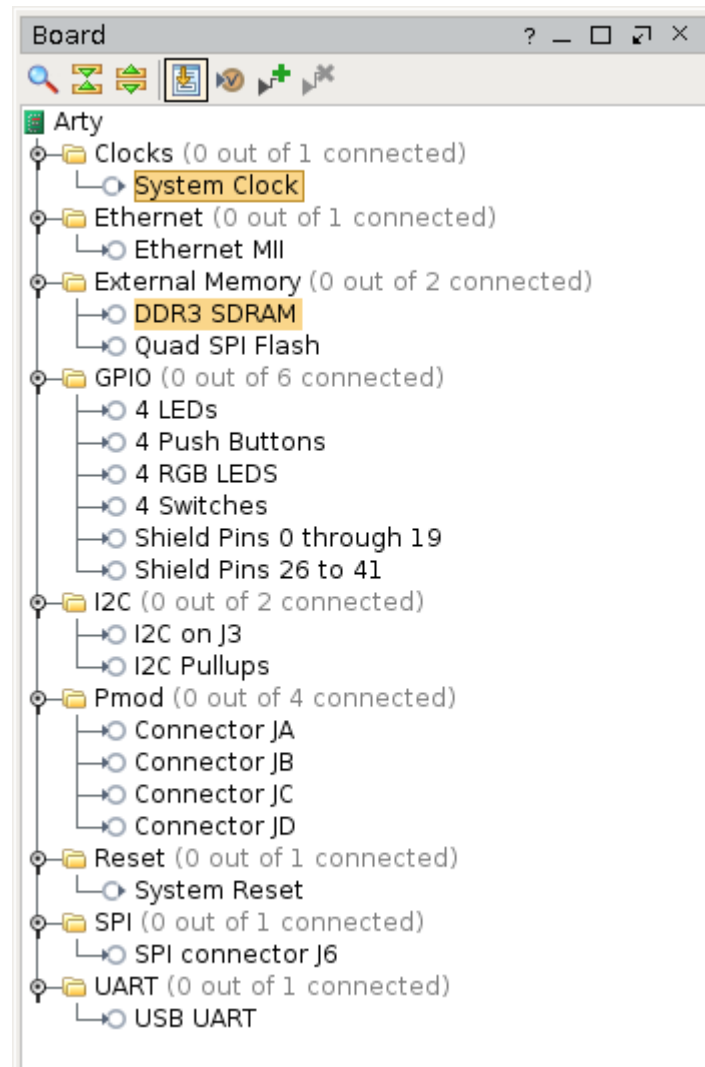
- Na aba lateral esquerda clique em **Create Block Design**



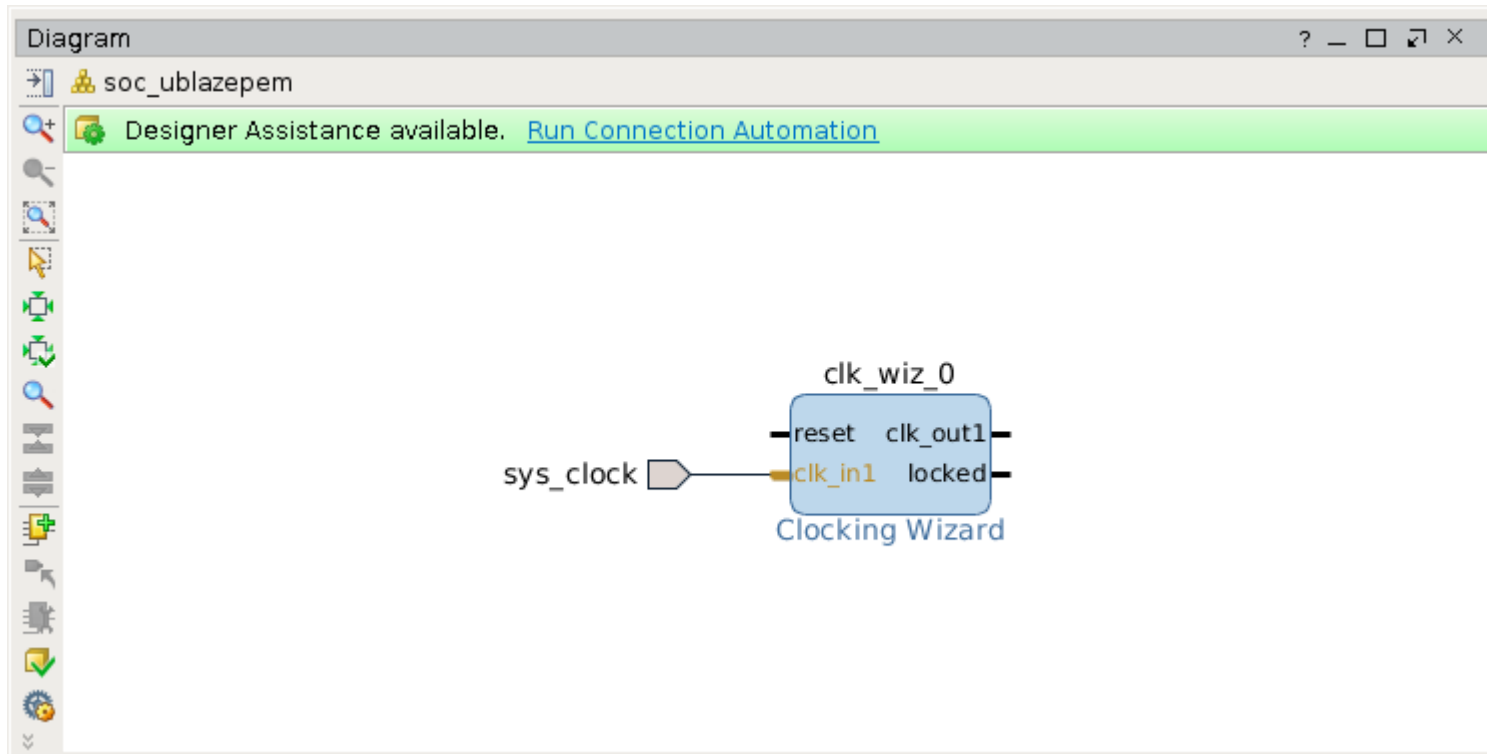
- Na caixa de Diagram vamos começar a “montar” o SoC



- Na **aba** Board vamos começar a colocando o **IP System Clock**

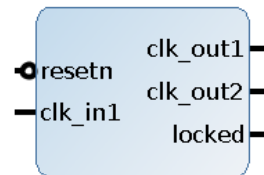


- Clique na caixa Clocking Wizard para configurá-lo



- Configure na aba Output Clock habilite **clk-out1** e **clk-out2** com **166.666** e **200.000** respectivamente

- Em Reset Type seleção **Active Low**



Re-customize IP

Clocking Wizard (5.3)

Documentation IP Location

IP Symbol Resource

☐ Show disabled ports

Component Name: soc\_ubla2epem\_clk\_wiz\_0\_0

Board Clocking Options **Output Clocks** MMCM Settings Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz) Requested	Actual	Phase (degrees) Requested
<input checked="" type="checkbox"/> clk_out1	clk_out1	166.667	166.667	0.000
<input checked="" type="checkbox"/> clk_out2	clk_out2	200.000	200.000	0.000
<input type="checkbox"/> clk_out3	clk_out3	100.000	N/A	0.000
<input type="checkbox"/> clk_out4	clk_out4	100.000	N/A	0.000
<input type="checkbox"/> clk_out5	clk_out5	100.000	N/A	0.000
<input type="checkbox"/> clk_out6	clk_out6	100.000	N/A	0.000
<input type="checkbox"/> clk_out7	clk_out7	100.000	N/A	0.000

☐ USE CLOCK SEQUENCING

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Clocking Feedback

Source

☒ Automatic Control On-Chip

☐ Automatic Control Off-Chip

☐ User-Controlled On-Chip

☐ User-Controlled Off-Chip

Enable Optional Inputs / Outputs for MMCM/PLL

☒ reset ☐ power\_down ☐ input\_clk\_stopped

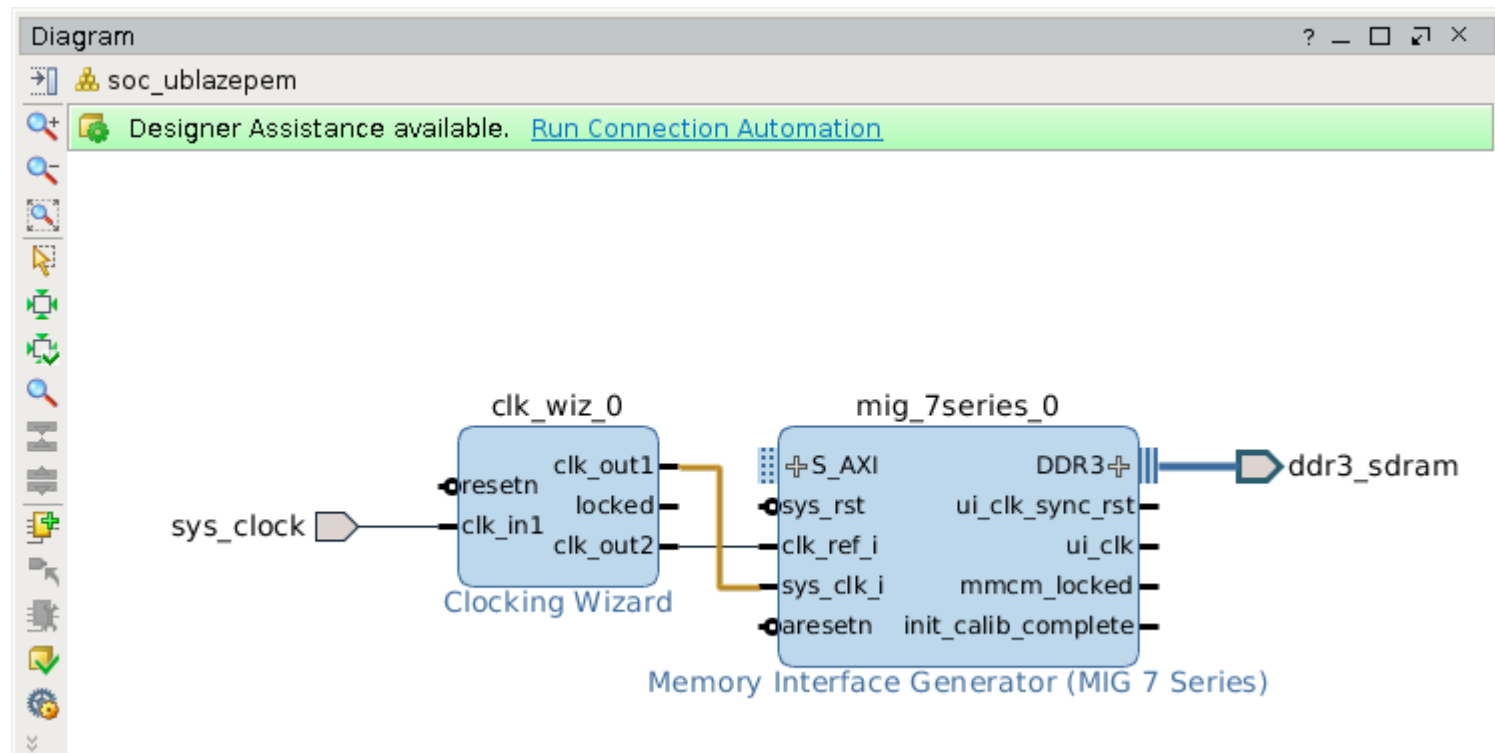
Reset Type

☐ Active High ☒ Active Low

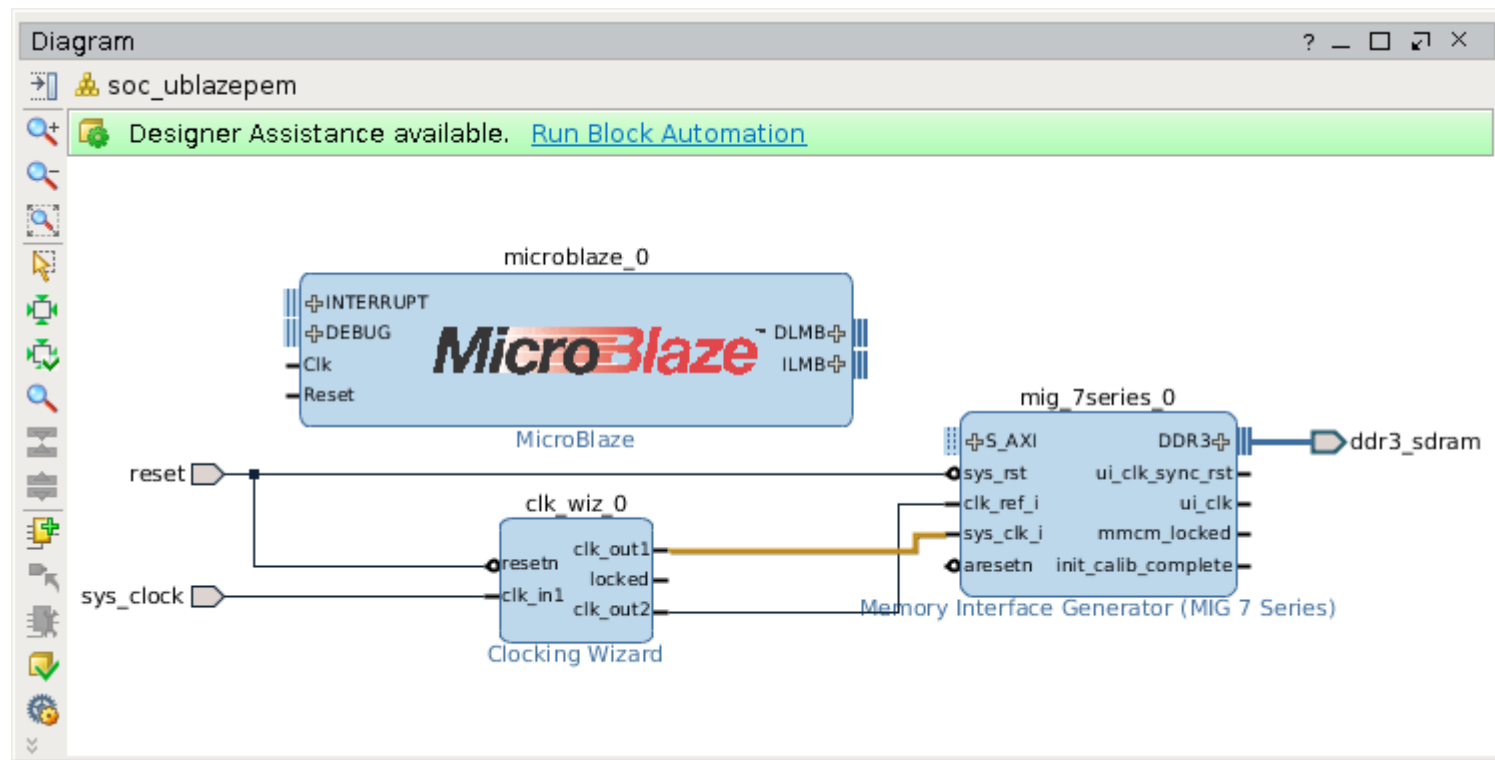
OK Cancel



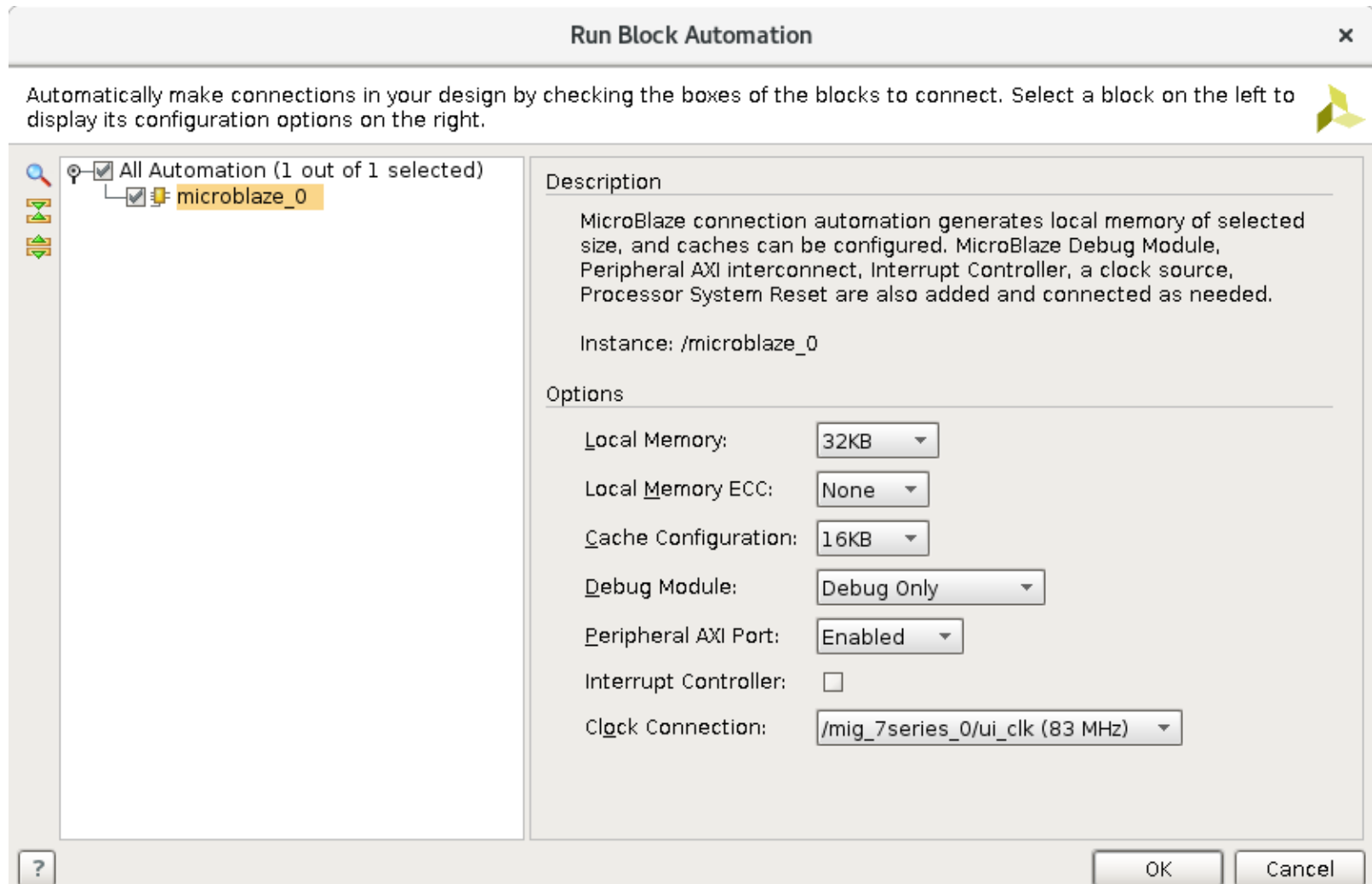
- Adicione o IP DDR3 presente na aba Boards, delete as portas clk-ref-I e sys-clk-I e faça as seguintes ligações
- Clique em Run Connection Automation e habilite All Automation



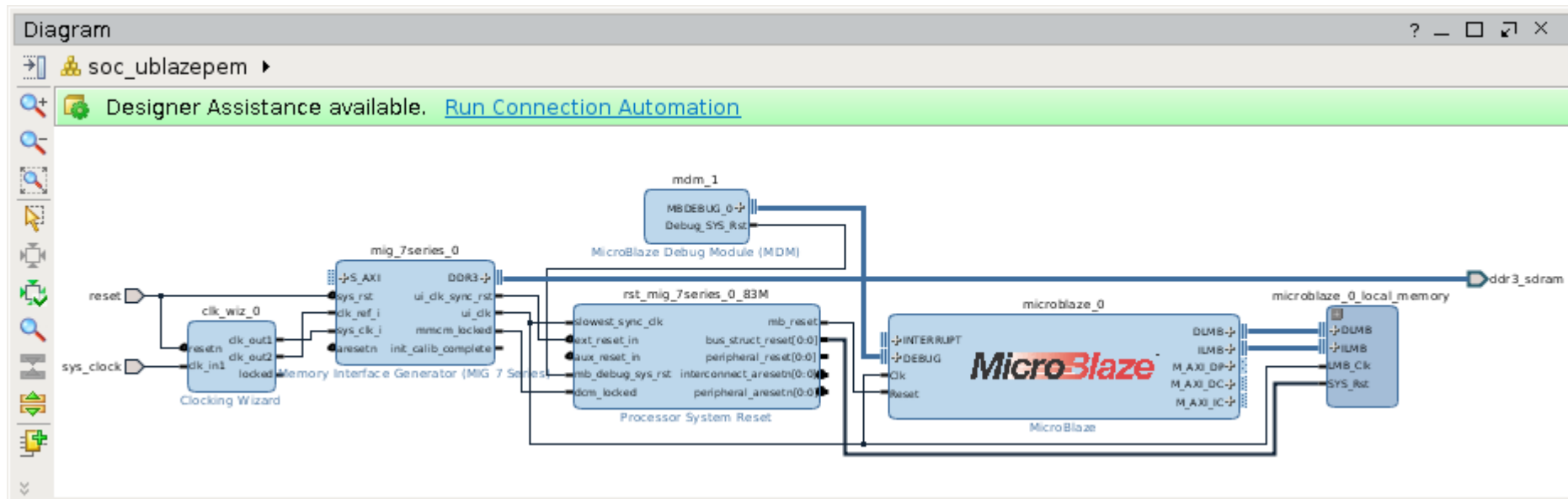
- Clique em **Add IP** (caixinha amarela com um +), procure por **MicroBlaze** e adicione-o



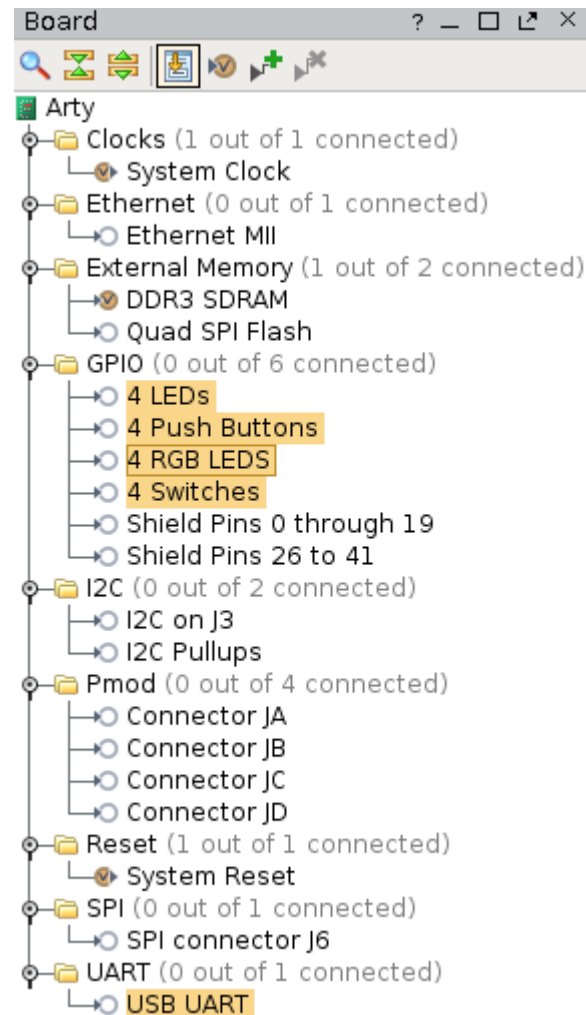
- Clique em no bloco **MicroBlaze** e configure-o como mostrado abaixo



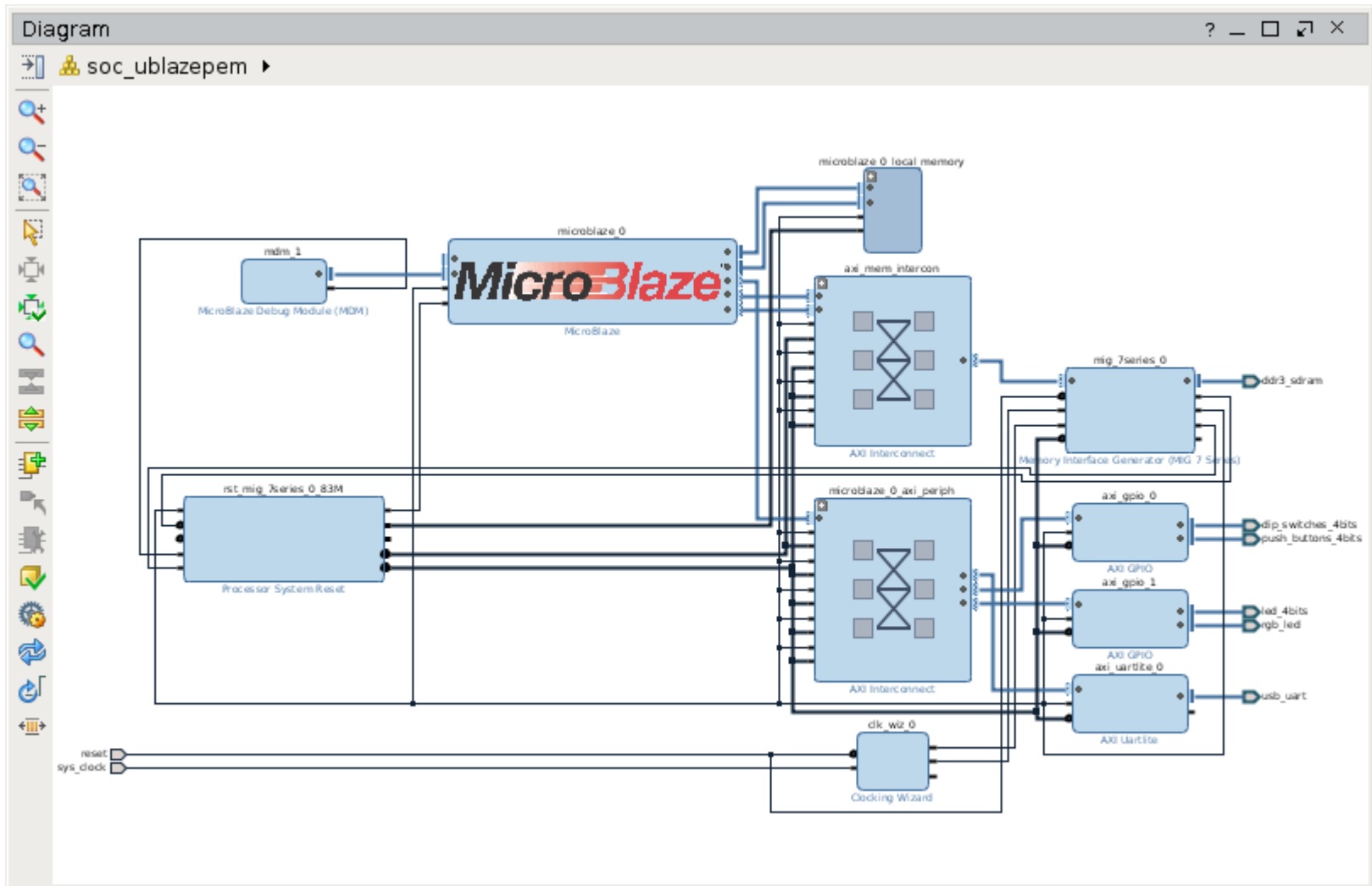
- Clique em **Run Block Automation**
- Temos o sistema do processador MicroBlaze todo pronto até aqui
- Os próximos passos são adicionar os IPs da Xilinx para a ARTY



- Na aba Board selecione todos os IPs mostrados abaixo e arraste-os para o Block Design



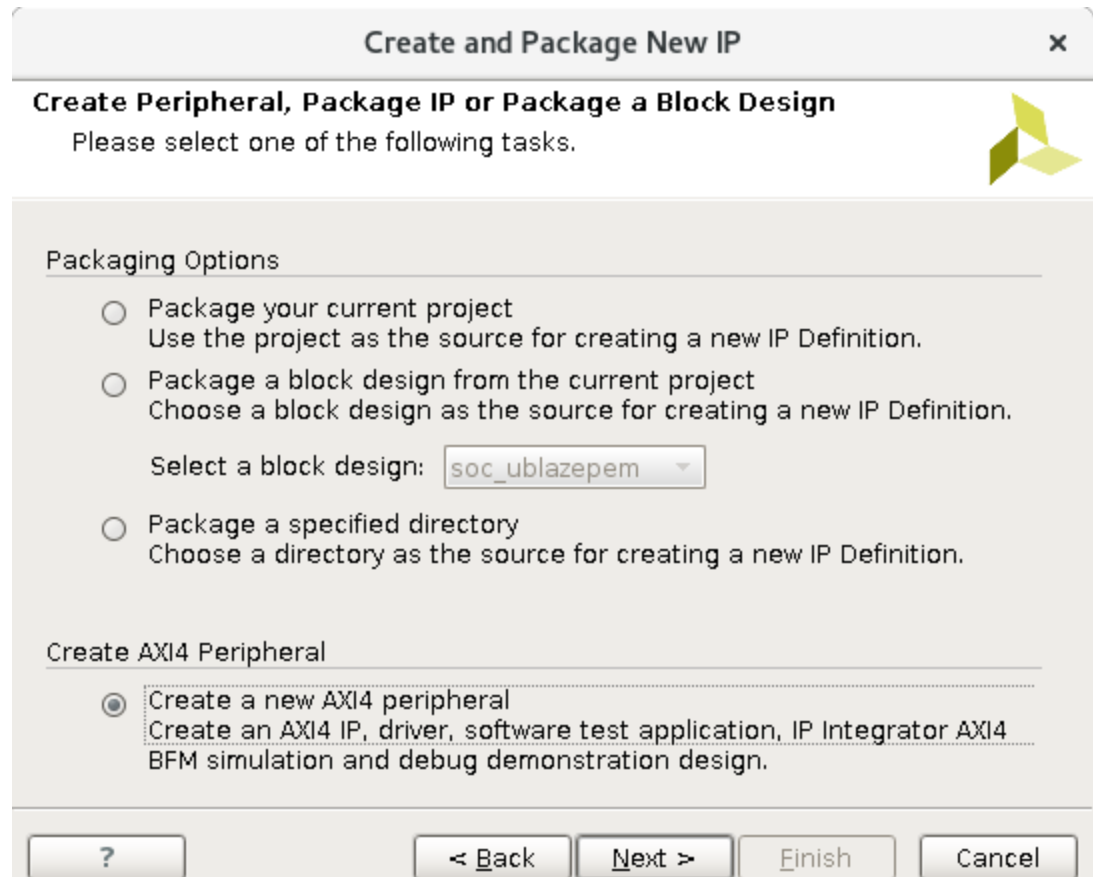
- Clique em Run Connection Automation, selecione All Automation e OK



- Pronto! Até aqui temos o MicroBlaze conectado com a UART, os push-buttons, switches e LEDs da ARTY
- Agora, será colocado o IP AES, a criação desse custom IP dará o template de como portar IPs custom no Vivado

- “Empacotando” IP AES no **Vivado**

- Vá em Tools > Create and Package New IP...
- Como a interface desse SoC é AXI4 e nosso IP também, selecione Create a new AXI4 Peripheral





- Configure o nome do IP

Create and Package New IP

Peripheral Details

Specify name, version and description for the new peripheral

Name:

aes\_ip

Version:

1.0

Display name:

aes\_ip\_v1.0

Description:

AES AXI IP

IP location:

to/ARTY/workspace/soc\_ublazepem/soc\_ublazepem/ip\_repo

☒ Overwrite existing

?

< Back

Next >

Finish

Cancel

- Configure o AMBA, o AES IP é Lite, Slave e contém 20 registradores

**Create and Package New IP** ✕

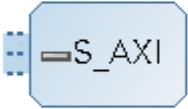
**Add Interfaces**  
 Add AXI4 interfaces supported by your peripheral

☐ Enable Interrupt Support

+
-

Interfaces

S\_AXI

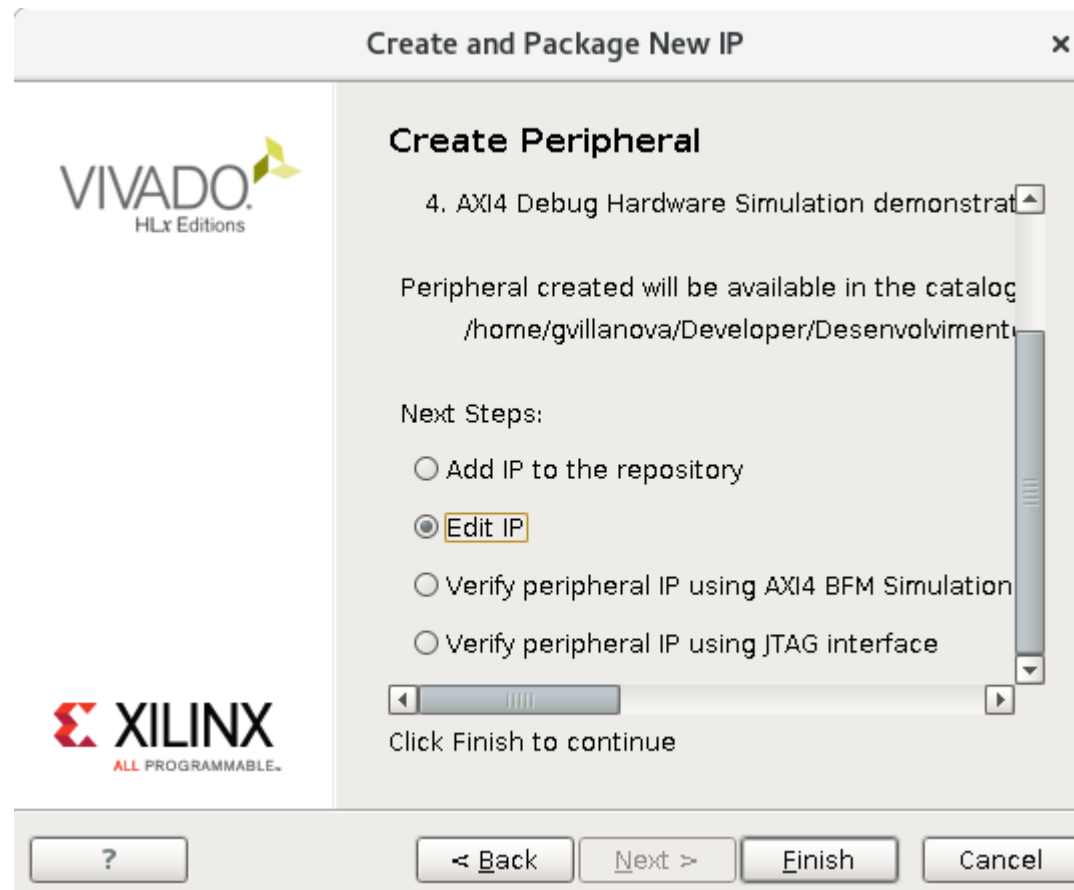


S\_AXI  
aes\_ip\_v1.0

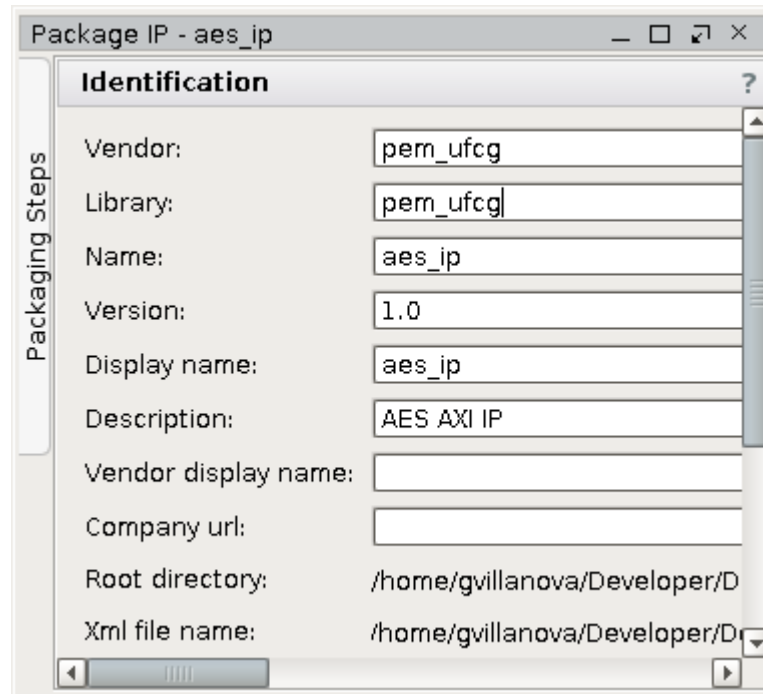
Name	S_AXI
<a href="#">Interface Type</a>	Lite
Interface Mode	Slave
<a href="#">Data Width (Bits)</a>	32
<a href="#">Memory Size (Bytes)</a>	64
<a href="#">Number of Registers</a>	20 [4..512]

?
< Back
Next >
Finish
Cancel

– Seleccione Edit IP



- Preencha a identificação do IP



The screenshot shows a window titled "Package IP - aes\_ip" with a sidebar labeled "Packaging Steps". The main area is titled "Identification" and contains the following fields:

Vendor:	pern_ufcg
Library:	pern_ufcg
Name:	aes_ip
Version:	1.0
Display name:	aes_ip
Description:	AES AXI IP
Vendor display name:	
Company url:	
Root directory:	/home/gvillanova/Developer/D
Xml file name:	/home/gvillanova/Developer/D

- Em **Design Sources**, clique com o botão direito e adicione todas as fontes (.sv) do IP
- O modulo topo terá a interface o AMBA AXI4 Lite como barramento, é **interessante portar o template do AMBA gerado pelo Vivado** para ficar compatível com a nomenclatura usada pelos outros IPs do projeto
- Veja o exemplo seguinte



x

### Add Sources

**Add or Create Design Sources**

Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project.

	Index	Name	Library	Location
+	1	amba_adaptor.sv	xil_defaultlib	/home/gvill...
-	2	ip_aes.sv	xil_defaultlib	/home/gvill...
↑	3	aes.sv	xil_defaultlib	/home/gvill...
↓	4	control.sv	xil_defaultlib	/home/gvill...
	5	ip_aes_noamba.sv	xil_defaultlib	/home/gvill...
	6	key_expansion.sv	xil_defaultlib	/home/gvill...
	7	mux_busA.sv	xil_defaultlib	/home/gvill...
	8	mux_busB.sv	xil_defaultlib	/home/gvill...
	9	mux_busR.sv	xil_defaultlib	/home/gvill...
	10	register_file.sv	xil_defaultlib	/home/gvill...
	11	ula.sv	xil_defaultlib	/home/gvill...

Add Files
Create File

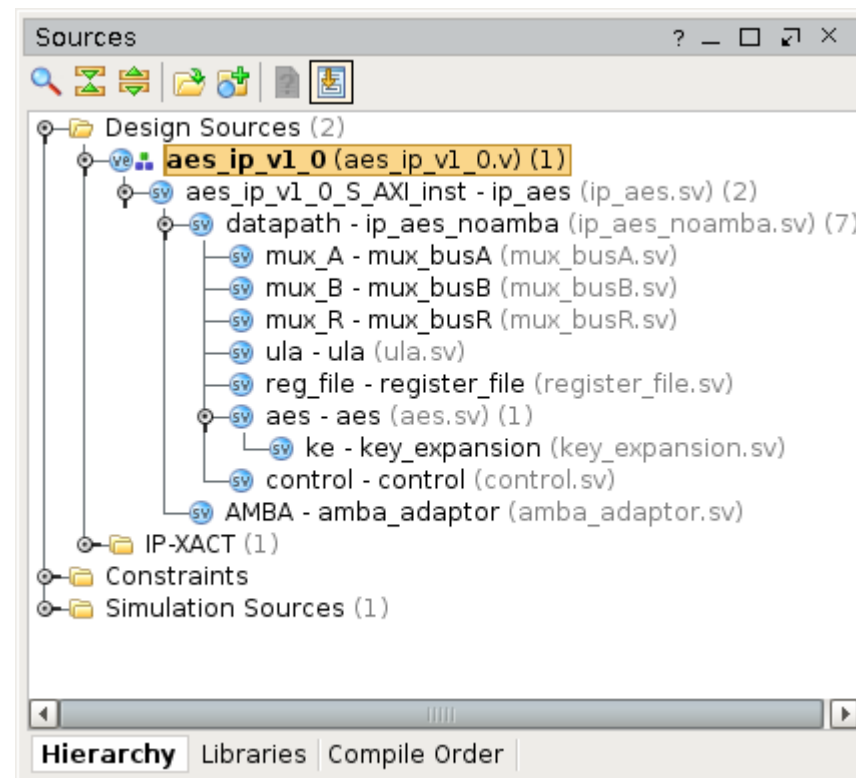
☐ Scan and add RTL include files into project

☒ Copy sources into IP Directory

☒ Add sources from subdirectories

?
< Back
Next >
Finish
Cancel

- Todos os modulos do IP AES





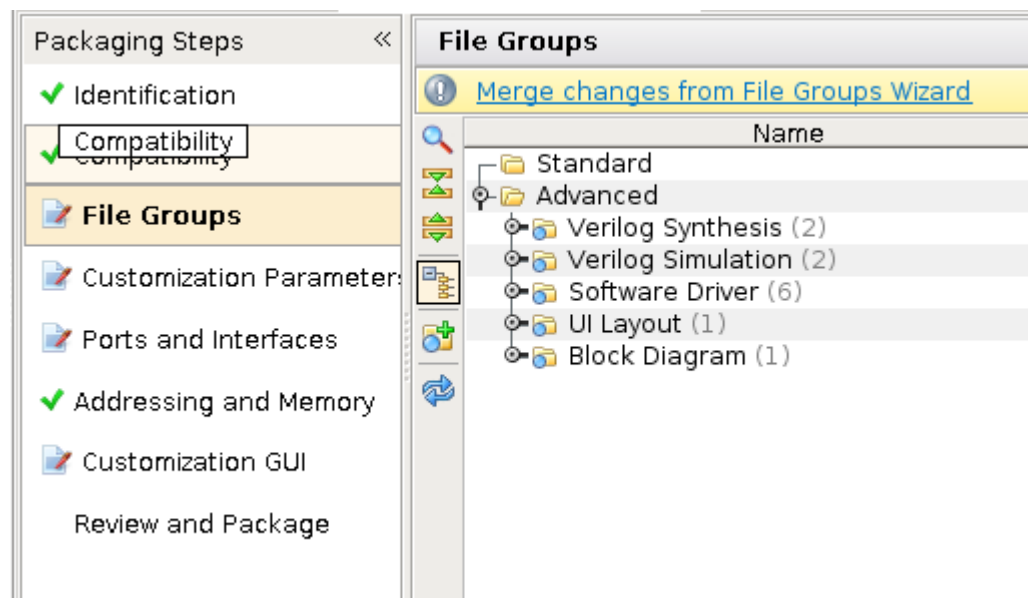
- Veja que o aes-ip-v1-0.v tem o template do AMBA gerado pelo Vivado e nele chamamos o modulo topo do IP AES criado por nós

```

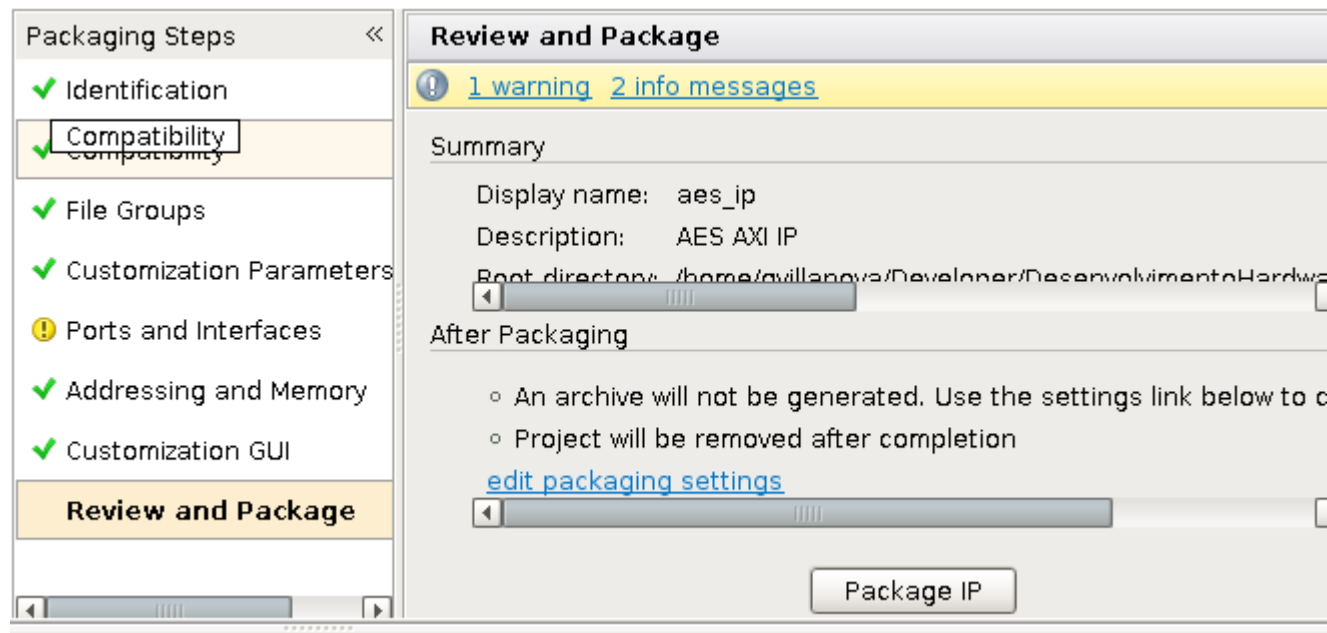
aes_ip_v1_0.v
lvimento/ARTY/workspace/soc_ublazepern/soc_ublazepern
38     input wire [2 : 0] s_axi_arprot,
39     input wire s_axi_arvalid,
40     output wire s_axi_arready,
41     output wire [C_S_AXI_DATA_WIDTH-1 :
42     output wire [1 : 0] s_axi_rresp,
43     output wire s_axi_rvalid,
44     input wire s_axi_rready
45 );
46 // Instantiation of Axi Bus Interface S_AXI
47 //aes_ip_v1_0_S_AXI # (
48     ip_aes # (
49         .C_S_AXI_DATA_WIDTH(C_S_AXI_DATA_WI
50         .C_S_AXI_ADDR_WIDTH(C_S_AXI_ADDR_WI
51     ) aes_ip_v1_0_S_AXI_inst (
52         .S_AXI_ACLK(s_axi_aclk),
53         .S_AXI_ARESETN(s_axi_aresetn),
54         .S_AXI_AWADDR(s_axi_awaddr),
55         .S_AXI_AWPROT(s_axi_awprot),
56         .S_AXI_AWVALID(s_axi_awvalid),
--

```

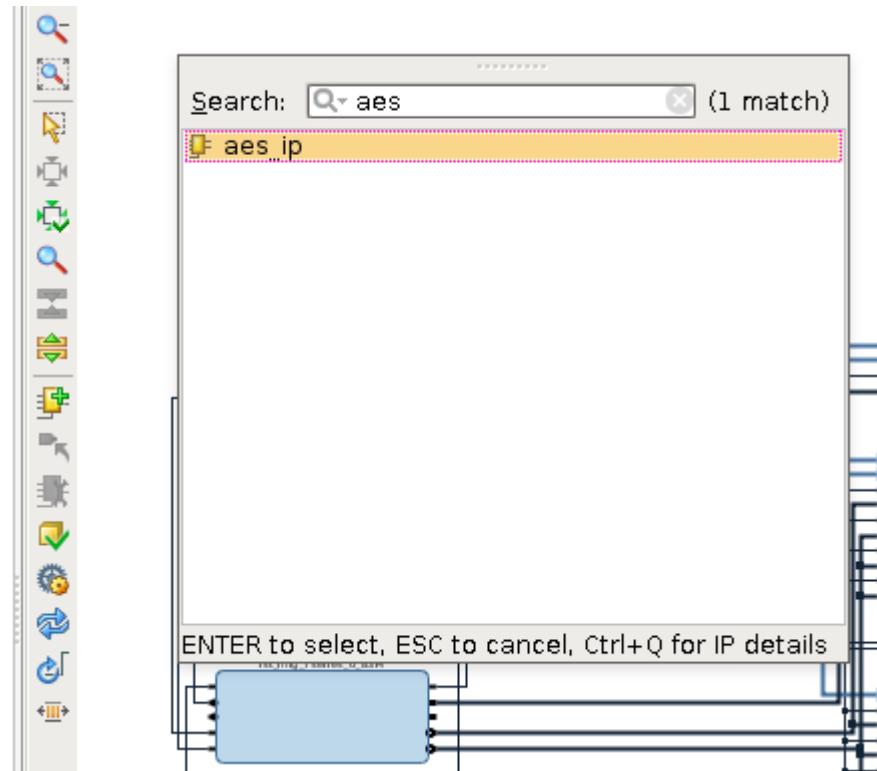
- Na anela Packaging Steps clique em todos as abas sem o tick verde e clique em Merge changes from file Groups Wizard



- Clique em Review and Package e **Package IP**



- De volta ao **Block Design**, clique em **Add IP**, procure por **aes** e adicione-o no projeto

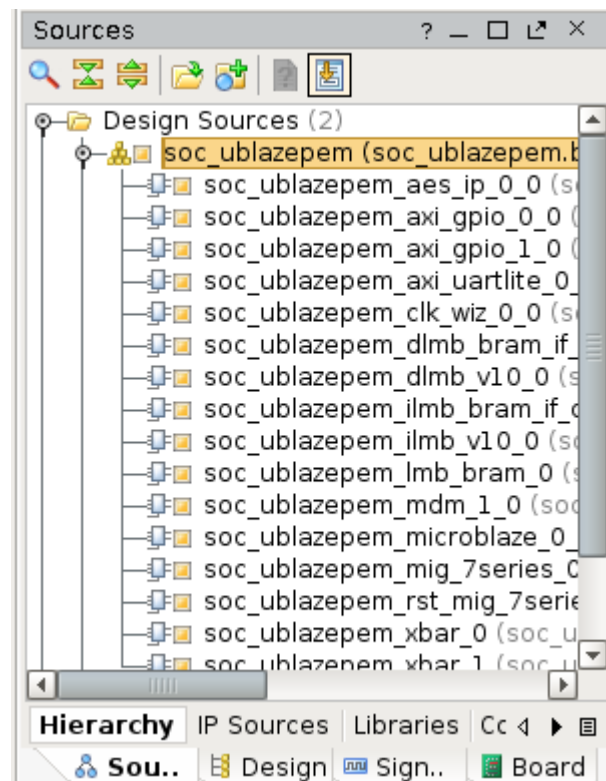




- Clique em Window > Address Editor
- Veja que o Vivado já criou o base-address para o acesso do nosso IP
- Ele também mostra os endereços dos outros periféricos, etc...

Address Editor						
	Cell	Slave Interface	Base Name	Offset Address	Range	High Address
microblaze_0						
Data (32 address bits : 4G)						
microblaze_0_local_memory/dlmb_b...	SLMB	Mem		0x0000_0000	32K	0x0000_7FFF
mig_7series_0	S_AXI	memaddr		0x8000_0000	256M	0x8FFF_FFFF
axi_uartlite_0	S_AXI	Reg		0x4060_0000	64K	0x4060_FFFF
axi_gpio_0	S_AXI	Reg		0x4000_0000	64K	0x4000_FFFF
axi_gpio_1	S_AXI	Reg		0x4001_0000	64K	0x4001_FFFF
aes_ip_0	S_AXI	S_AXI_reg		0x44A0_0000	64K	0x44A0_FFFF
Instruction (32 address bits : 4G)						
microblaze_0_local_memory/ilmb_br...	SLMB	Mem		0x0000_0000	32K	0x0000_7FFF
mig_7series_0	S_AXI	memaddr		0x8000_0000	256M	0x8FFF_FFFF

- Em Sources, clique com o botão direito do mouse em soc\_ublazepem e clique em Create HDL Wrapper...
- Escolha “Let Vivado manage wrapper and auto-update”



Design Runs												
Name	Co...	Status	WNS <sup>1</sup>	TNS	WHS	THS	TPWS	Total Power	Fai...	LUT	FF	BRAM
✓ synth_1 (active)	c...	<b>synth_design Complete!</b>								0	0	0
✓ impl_1	co...	write_bitstream Complete!	1.274	0.000	0.012	0.000	0.000	1.275	0	15025	10413	18
Out-of-Context Module Runs												
✓ soc_ublazepem		Submodule Runs Complete										
✓ soc_ublazepem_axi_u...	so...	synth_design Complete!								109	94	0
✓ soc_ublazepem_mig_...	so...	synth_design Complete!								5362	4183	0
✓ soc_ublazepem_dlmb...	so...	synth_design Complete!								0	1	0
✓ soc_ublazepem_dlmb...	so...	synth_design Complete!								4	2	0
✓ soc_ublazepem_lmb_...	so...	synth_design Complete!								0	0	8
✓ soc_ublazepem_rst_...	so...	synth_design Complete!								19	40	0
✓ soc_ublazepem_clk_w...	so...	synth_design Complete!								1	0	0
✓ soc_ublazepem_micr...	so...	synth_design Complete!								1724	2117	10
✓ soc_ublazepem_ilmb_...	so...	synth_design Complete!										
✓ soc_ublazepem_ilmb_...	so...	synth_design Complete!								4	2	0
✓ soc_ublazepem_mdm...	so...	synth_design Complete!								96	112	0
✓ soc_ublazepem_axi_g...	so...	synth_design Complete!								40	93	0
✓ soc_ublazepem_axi_g...	so...	synth_design Complete!								62	173	0
✓ soc_ublazepem_xbar_...	so...	synth_design Complete!								484	491	0
✓ soc_ublazepem_auto...	so...	synth_design Complete!								821	935	0
✓ soc_ublazepem_aes.i...	so...	synth_design Complete!								6514	2552	0
✓ soc_ublazepem_xbar_...	so...	synth_design Complete!								198	133	0
✓ soc_ublazepem_auto...	so...	synth_design Complete!								331	521	0

Design Runs												
Name	Co...	Status	WNS <sup>1</sup>	TNS	WHS	THS	TPWS	Total Power	Fai...	LUT	FF	BRAM
✓ synth_1 (active)	c...	<b>synth_design Complete!</b>								0	0	0
✓ impl_1	co...	write_bitstream Complete!	1.274	0.000	0.012	0.000	0.000	1.275	0	15025	10413	18
Out-of-Context Module Runs												
✓ soc_ublazepem		Submodule Runs Complete										
✓ soc_ublazepem_axi_u...	so...	synth_design Complete!								109	94	0
✓ soc_ublazepem_mig_...	so...	synth_design Complete!								5362	4183	0
✓ soc_ublazepem_dlmb...	so...	synth_design Complete!								0	1	0
✓ soc_ublazepem_dlmb...	so...	synth_design Complete!								4	2	0
✓ soc_ublazepem_lmb_...	so...	synth_design Complete!								0	0	8
✓ soc_ublazepem_rst_...	so...	synth_design Complete!								19	40	0
✓ soc_ublazepem_clk_w...	so...	synth_design Complete!								1	0	0
✓ soc_ublazepem_micr...	so...	synth_design Complete!								1724	2117	10
✓ soc_ublazepem_ilmb_...	so...	synth_design Complete!										
✓ soc_ublazepem_ilmb_...	so...	synth_design Complete!								4	2	0
✓ soc_ublazepem_mdm...	so...	synth_design Complete!								96	112	0
✓ soc_ublazepem_axi_g...	so...	synth_design Complete!								40	93	0
✓ soc_ublazepem_axi_g...	so...	synth_design Complete!								62	173	0
✓ soc_ublazepem_xbar_...	so...	synth_design Complete!								484	491	0
✓ soc_ublazepem_auto...	so...	synth_design Complete!								821	935	0
✓ soc_ublazepem_aes.i...	so...	synth_design Complete!								6514	2552	0
✓ soc_ublazepem_xbar_...	so...	synth_design Complete!								198	133	0
✓ soc_ublazepem_auto...	so...	synth_design Complete!								331	521	0



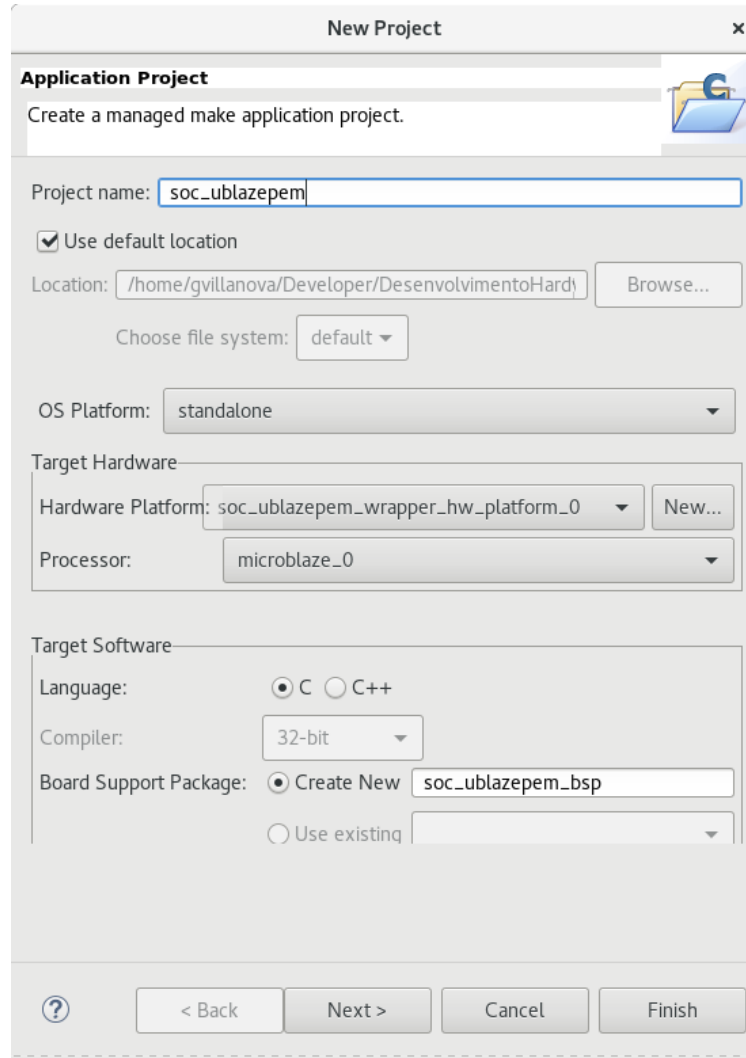
- **API (Application Programming Interface)**

- Com o SoC “montado”, base address definido, foi desenvolvida a seguinte API

```
void AES_init(Xuint32);  
void AES_writeKey(Xuint32,Xuint32,Xuint32,Xuint32);  
void AES_writeData(Xuint32,Xuint32,Xuint32,Xuint32);  
void AES_writeIV(Xuint32,Xuint32,Xuint32,Xuint32);  
void AES_writeCounter(Xuint32,Xuint32);  
void AES_writeResult(Xuint32,Xuint32,Xuint32,Xuint32);  
void AES_setMode(Xuint32);  
void AES_setCounterZero(Xuint32);  
void AES_setEncrypt(Xuint32);  
void AES_writeConfig(Xuint32,Xuint32,Xuint32,Xuint32);  
bool AES_getEncryption(void);  
bool AES_getDecryption(void);  
Xuint32 AES_getRegister(Xuint32);
```

- **SDK (Software Development Kit) e Teste do IP com o SoC desenvolvido**
  - Nessa seção será mostrado o uso do SDK da Xilinx, o porte do SoC para FPGA e por fim o teste
  - Clique em File > Export > Export Hardware...
  - Selecione Include bitstream e OK
  - Clique em File > Launch SDK e OK

- Com o SDK aberto, clique em New > Application Project e siga os passos seguintes



**New Project**

**Application Project**  
Create a managed make application project.

Project name:

☒ Use default location

Location:

Choose file system:

OS Platform:

**Target Hardware**

Hardware Platform:

Processor:

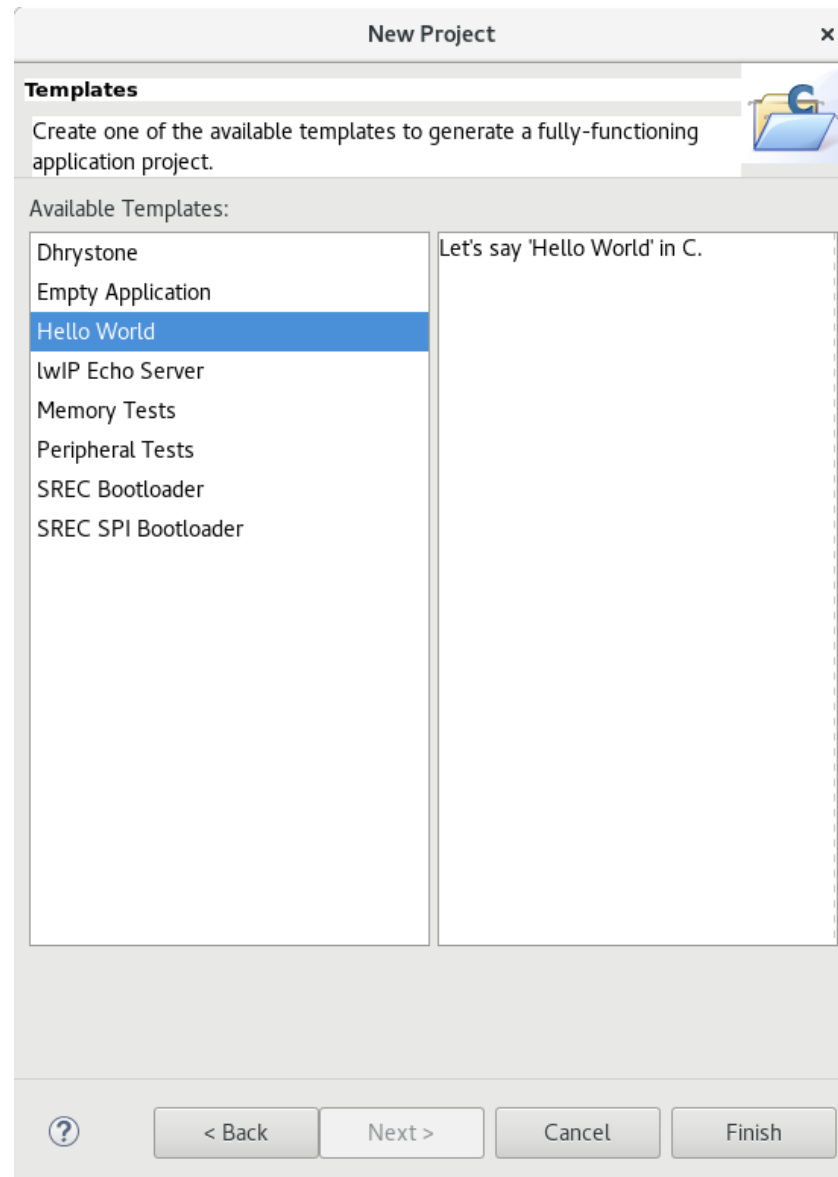
**Target Software**

Language: ☒ C ☐ C++

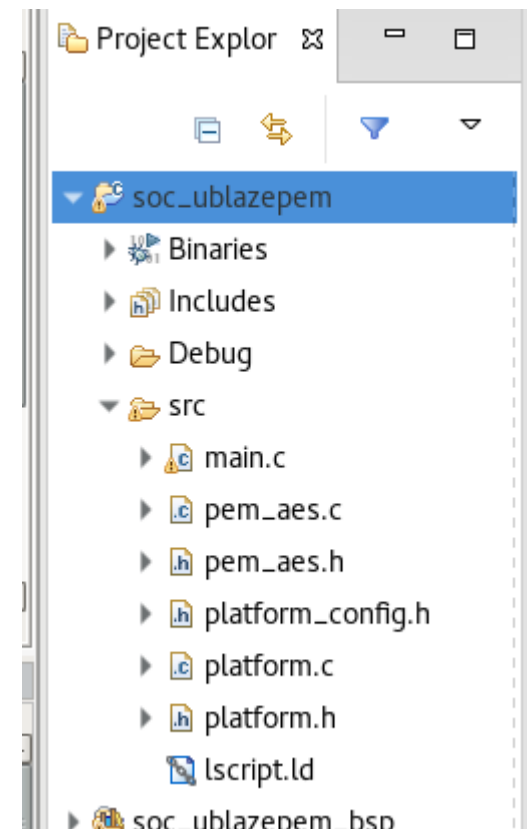
Compiler:

Board Support Package: ☒ Create New

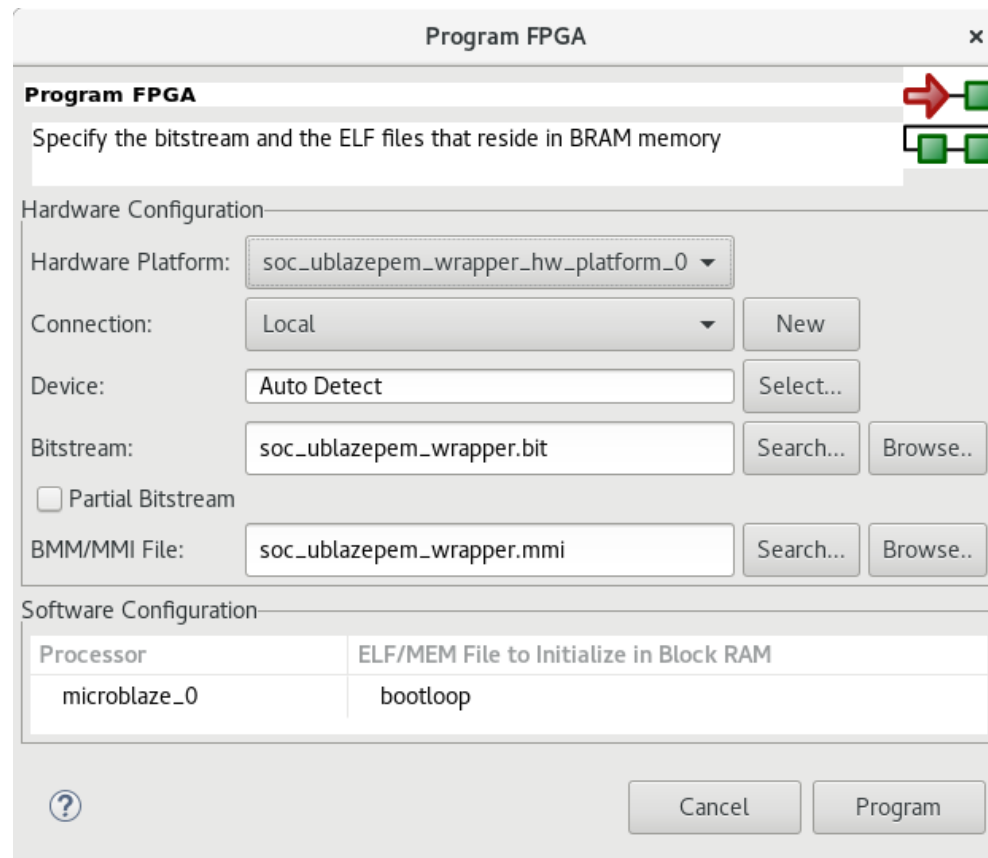
☐ Use existing



- Pode-se observar em Project Explorer toda a estrutura dos arquivos para o projeto (.ld, .c, .h, etc)
- O helloworld.c foi substituído para o main.c que contém o teste
- A API (.c e .h) foi incluída no processo
- Todos esses códigos estão em pem/aes/api



- Clique em **Build Project** para compilar o projeto
- Veja no Console o resultado
- Se tudo certo, conecte a FPGA no PC e clique **Program FPGA**



- Clique em **Run** para passar o binário para a “flash” do SoC (selecione Launch on Hardware (System Debugger))
- **OK, a aplicação está rodando!**

## • Resultado

- Como pode ser visto no main.c o teste replica os resultados mostrados no refmod.c desenvolvido para UVM
- A UART imprime os resultados de cada teste (todos os modos, encriptando e decryptando)



gvillanova@gvillanova:~/Developer/DesenvolvimentoHardware/KitsDesenvolv... x

File Edit View Search Terminal Help

```
*****
AES IP - Embedded Test
*****
```

```
Inputs:
Data1:  0x00112233445566778899AABBCCDDEEFF
Data2:  0xFFEEDDCCBBAA99887766554433221100
Key:    0x000102030405060708090A0B0C0D0E0F
IV:     0xCCCCCCCCAAAAAAFFFFFFFFEEEEEEEE
```

```
ECB MODE
first cipher block:
0xD8E0C469
0x30047B6A
0x80B7CDD8
0x5AC5B470
```

```
second cipher block:
0x7823871B
0xFD4F5F79
0xFC552877
0x4D96CA87
```

```
first decrypted block:
0x33221100
0x77665544
0xBBAA9988
0xFFEEDDCC
```

```
second decrypted block:
0xCCDDEEFF
0x8899AABB
0x44556677
0x00112233
```

```
*****
END TEST
*****
```

gvillanova@gvillanova:~/Developer/DesenvolvimentoHardware/KitsDesenvolv... x

File Edit View Search Terminal Help

```
*****
AES IP - Embedded Test
*****
```

```
Inputs:
Data1:  0x00112233445566778899AABBCCDDEEFF
Data2:  0xFFEEDDCCBBAA99887766554433221100
Key:    0x000102030405060708090A0B0C0D0E0F
IV:     0xCCCCCCCCAAAAAAFFFFFFFFEEEEEEEE
```

```
CBC MODE
first cipher block:
0x617F21F6
0x6A0AD5B2
0x8C1F79E6
0x071E4B38
```

```
second cipher block:
0xC3F7E600
0x3AB389F0
0xBE01D7F8
0x82AF70B1
```

```
first decrypted block:
0x33221100
0x77665544
0xBBAA9988
0xFFEEDDCC
```

```
second decrypted block:
0xCCDDEEFF
0x8899AABB
0x44556677
0x00112233
```

```
*****
END TEST
*****
```

gvillanova@gvillanova:~/Developer/DesenvolvimentoHardware/KitsDesenvolv... x

File Edit View Search Terminal Help

```
*****
AES IP - Embedded Test
*****
```

```
Inputs:
Data1:  0x00112233445566778899AABBCCDDEEFF
Data2:  0xFFEEDDCCBBAA99887766554433221100
Key:    0x000102030405060708090A0B0C0D0E0F
IV:     0xCCCCCCCCAAAAAAFFFFFFFFEEEEEEEE
```

```
PCBC MODE
first cipher block:
0x617F21F6
0x6A0AD5B2
0x8C1F79E6
0x071E4B38
```

```
second cipher block:
0x65EA6617
0xBEE03A88
0xB123E35D
0x54BD1C43
```

```
first decrypted block:
0x33221100
0x77665544
0xBBAA9988
0xFFEEDDCC
```

```
second decrypted block:
0xCCDDEEFF
0x8899AABB
0x44556677
0x00112233
```

```
*****
END TEST
*****
```

gvillanova@gvillanova:~/Developer/DesenvolvimentoHardware/KitsDesenvolv... x

File Edit View Search Terminal Help

```
*****
AES IP - Embedded Test
*****
```

```
Inputs:
Data1:  0x00112233445566778899AABBCCDDEEFF
Data2:  0xFFEEDDCCBBAA99887766554433221100
Key:    0x000102030405060708090A0B0C0D0E0F
IV:     0xCCCCCCCCAAAAAAFFFFFFFFEEEEEEEE
```

```
OFB MODE
first cipher block:
0xF971C579
0x59916E3E
0x98007605
0x8EABA381
```

```
second cipher block:
0x03952103
0x187359DD
0x44C7AD7F
0x47F0E374
```

```
first decrypted block:
0x33221100
0x77665544
0xBBAA9988
0xFFEEDDCC
```

```
second decrypted block:
0xCCDDEEFF
0x8899AABB
0x44556677
0x00112233
```

```
*****
END TEST
*****
```

gvillanova@gvillanova:~/Developer/DesenvolvimentoHardware/KitsDesenvolv... x

File Edit View Search Terminal Help

```
*****
AES IP - Embedded Test
*****
```

```
Inputs:
Data1:  0x00112233445566778899AABBCCDDEEFF
Data2:  0xFFEEDDCCBBAA99887766554433221100
Key:    0x000102030405060708090A0B0C0D0E0F
IV:     0xCCCCCCCCAAAAAAFFFFFFFFEEEEEEEE
```

```
CFB MODE
first cipher block:
0xF971C579
0x59916E3E
0x98007605
0x8EABA381
```

```
second cipher block:
0xF18A773F
0x80E519BA
0x711351C7
0x23FFBC95
```

```
first decrypted block:
0x33221100
0x77665544
0xBBAA9988
0xFFEEDDCC
```

```
second decrypted block:
0xCCDDEEFF
0x8899AABB
0x44556677
0x00112233
```

```
*****
END TEST
*****
```

gvillanova@gvillanova:~/Developer/DesenvolvimentoHardware/KitsDesenvolv... x

File Edit View Search Terminal Help

```
*****
AES IP - Embedded Test
*****
```

```
Inputs:
Data1: 0x00112233445566778899AABBCCDDEEFF
Data2: 0xFFEEDDCCBBAA99887766554433221100
Key:   0x000102030405060708090A0B0C0D0E0F
IV:    0xCCCCCCCCAAAAAAFFFFFFFFEEEEEEEE
```

```
CTR MODE
first cipher block:
0xF971C579
0x59916E3E
0x98007605
0x8EABA381
```

```
second cipher block:
0xD2C73650
0x71E4C614
0xEE8EC419
0xA39A21C5
```

```
first decrypted block:
0x33221100
0x77665544
0xBBAA9988
0xFFEEDDCC
```

```
second decrypted block:
0xCCDDEEFF
0x8899AABB
0x44556677
0x00112233
```

```
*****
END TEST
*****
```



## Contact

### **Angelo Perkusich, D.Sc.**

Professor, CEO

[angelo.perkusich@embedded.ufcg.edu.br](mailto:angelo.perkusich@embedded.ufcg.edu.br)

+55 83 8811.9545

### **Hyggo Almeida, D.Sc.**

Professor, CTO

[hyggo.almeida@embedded.ufcg.edu.br](mailto:hyggo.almeida@embedded.ufcg.edu.br)

+55 83 8875.1894

### **Gabriel Villanova**

Aluno, DEE

[gabriel.magalhaes@embedded.ufcg.edu.br](mailto:gabriel.magalhaes@embedded.ufcg.edu.br)

+55 87 98866.2012

