DAT096: Getting Started with GRLIB

By: Bhavi

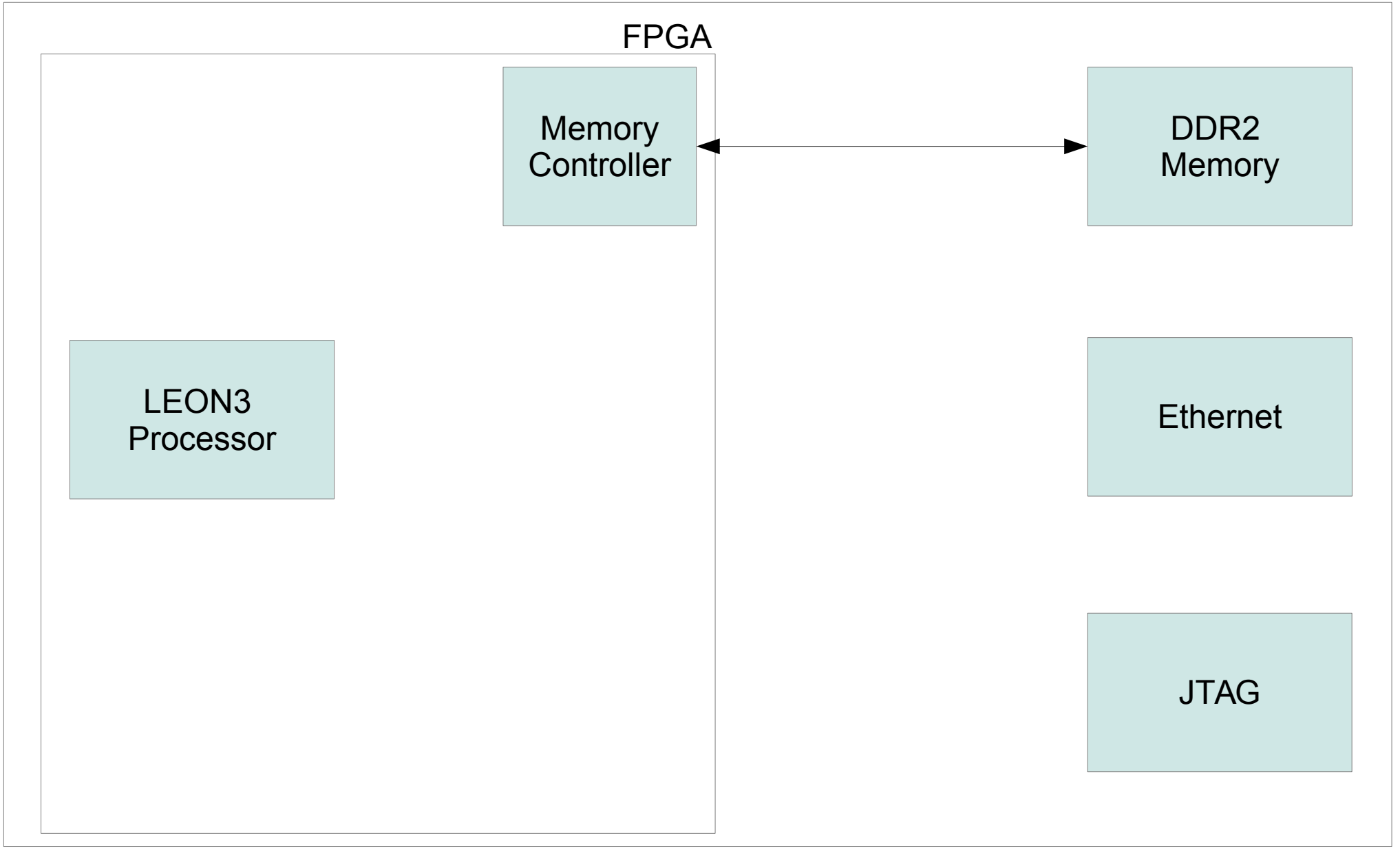
GRLIB - Introduction

- Gaisler Research Library
- Set of reusable IP cores for system-on-chip development using Gaisler's LEON3 processor core
- Includes
  - VHDL code for for processor core, peripherals, bus and memory controllers, debug support unit,
  - Design and user constraint files for specific boards
  - Design templates for specific boards
  - Makefiles and scripts for simulation and synthesis

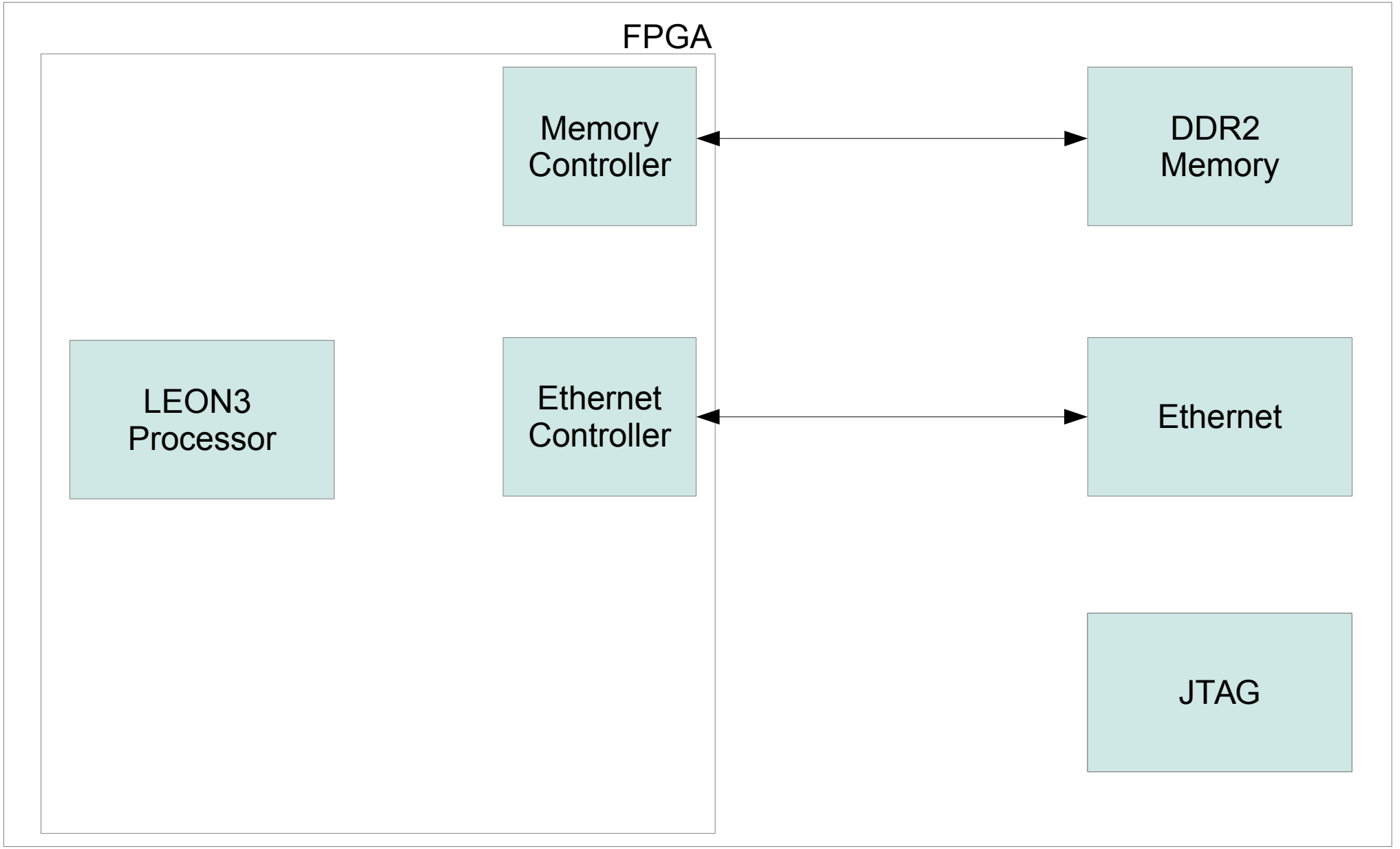
LEON3 Based System

LEON3 Based System

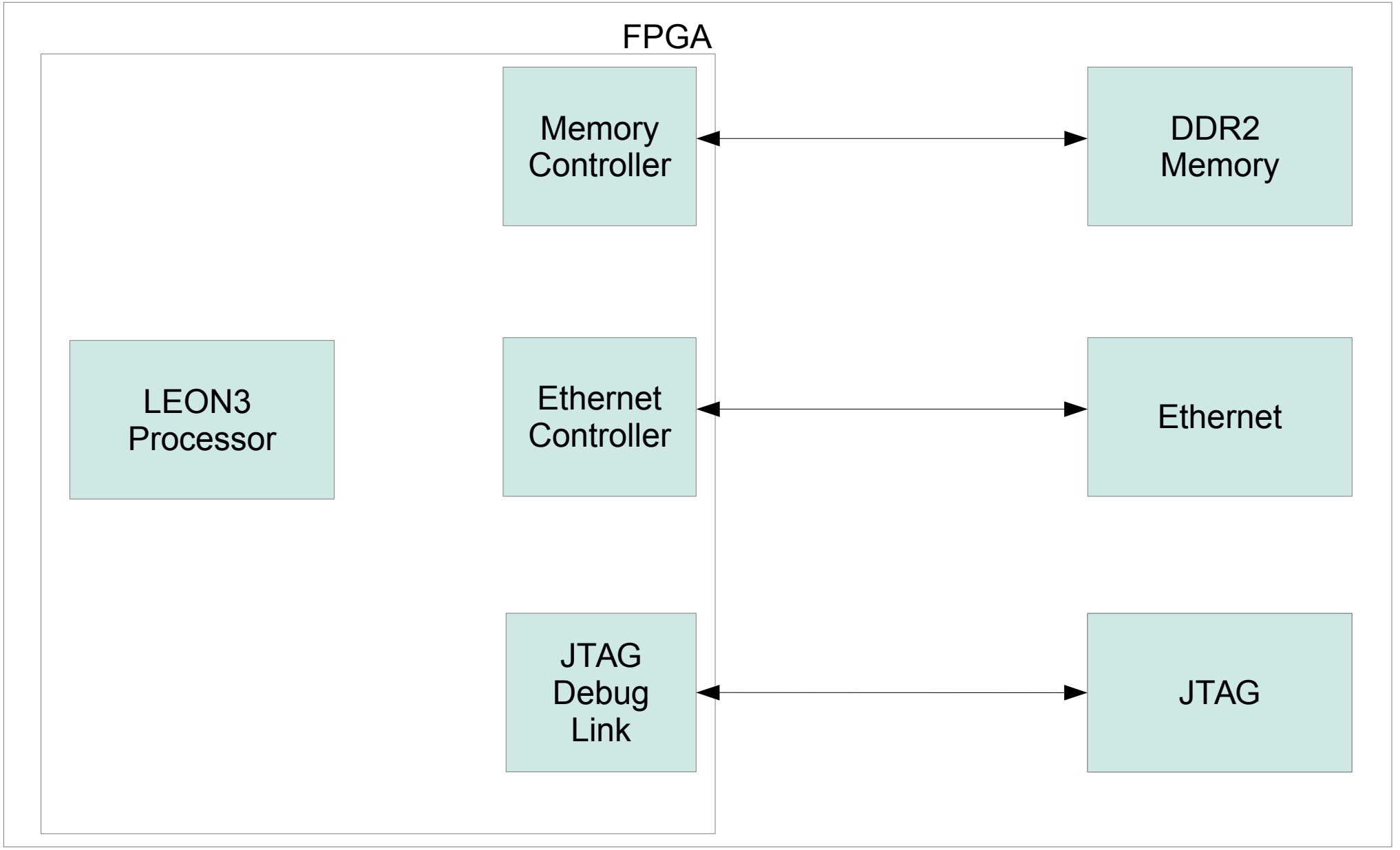
LEON3 Based System



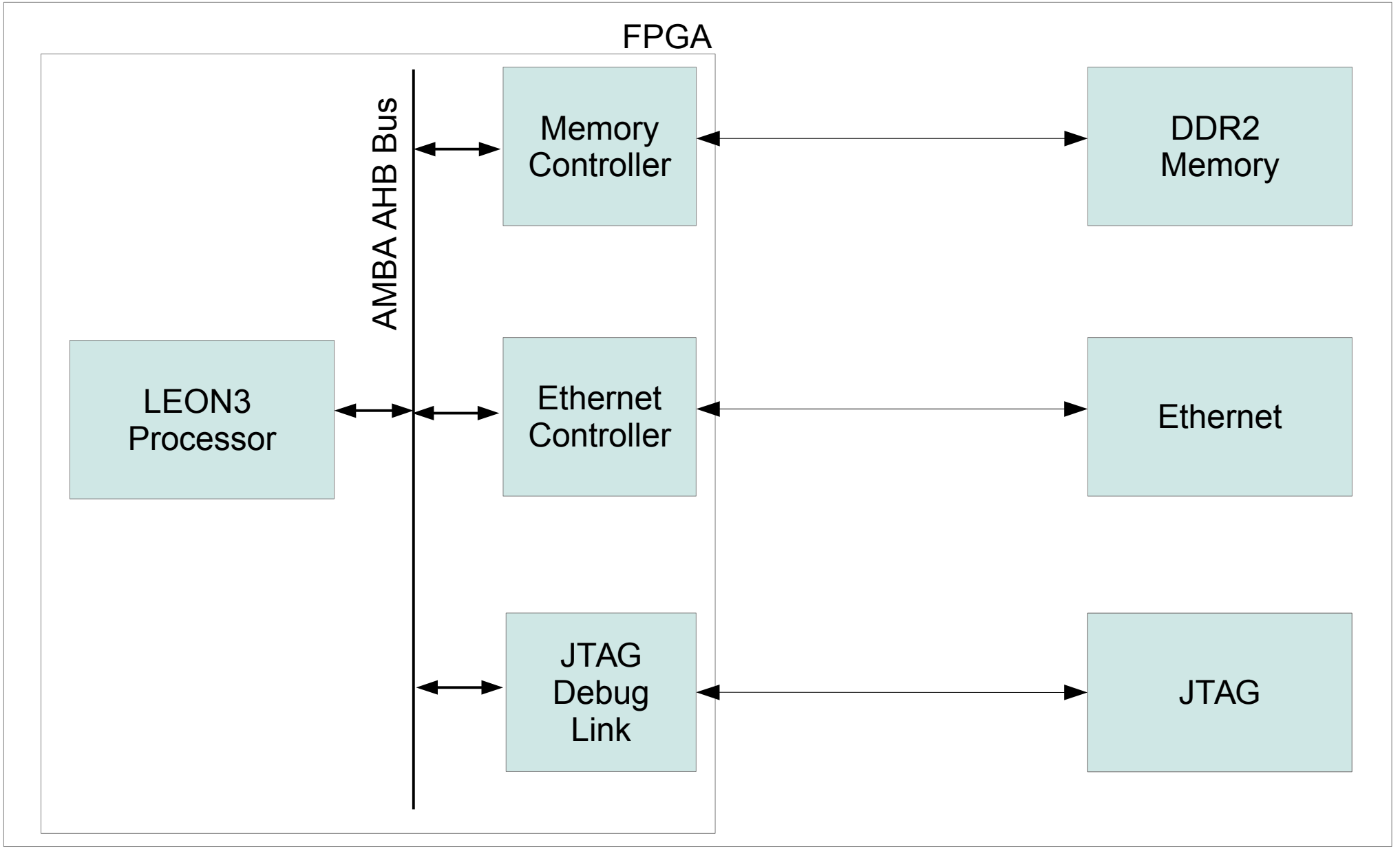
LEON3 Based System



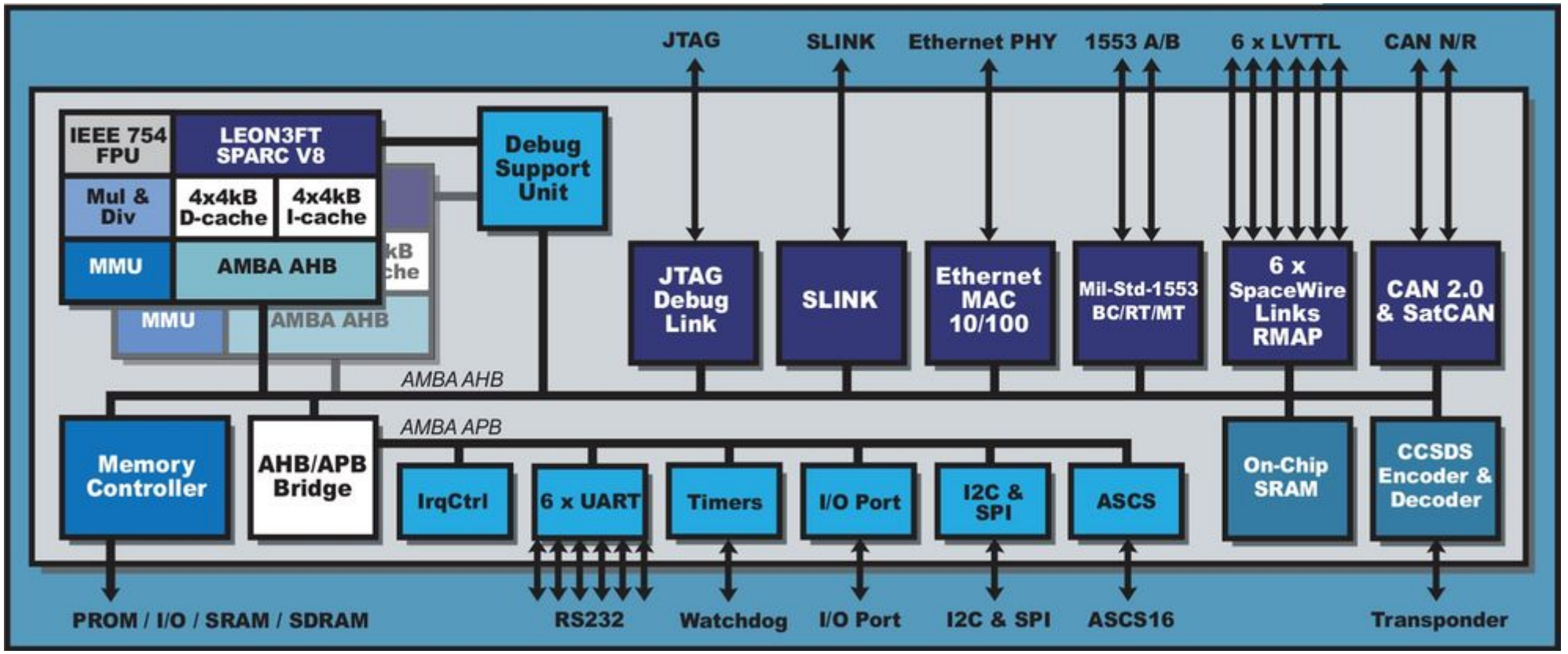
LEON3 Based System



LEON3 Based System

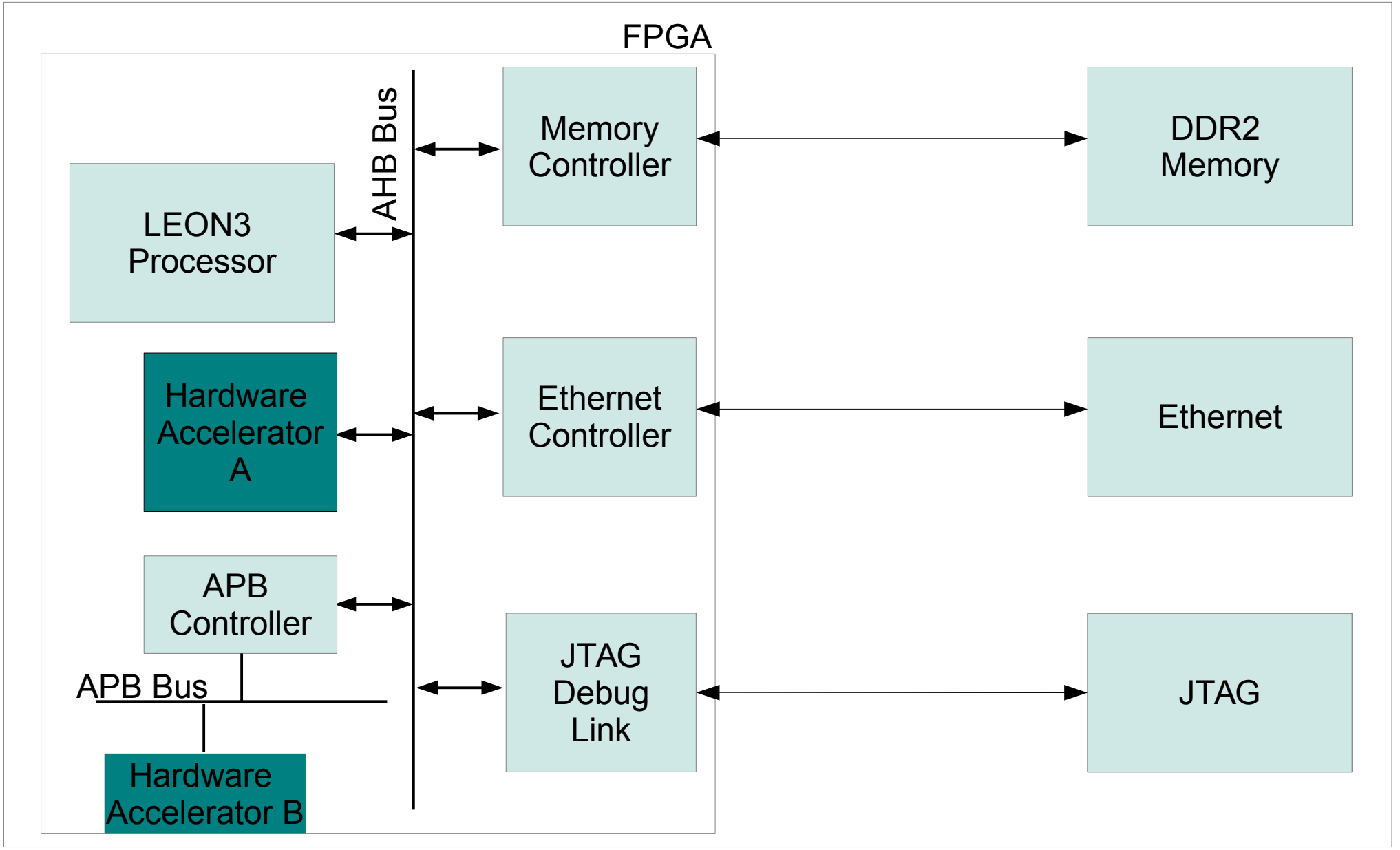


Example Block Diagram



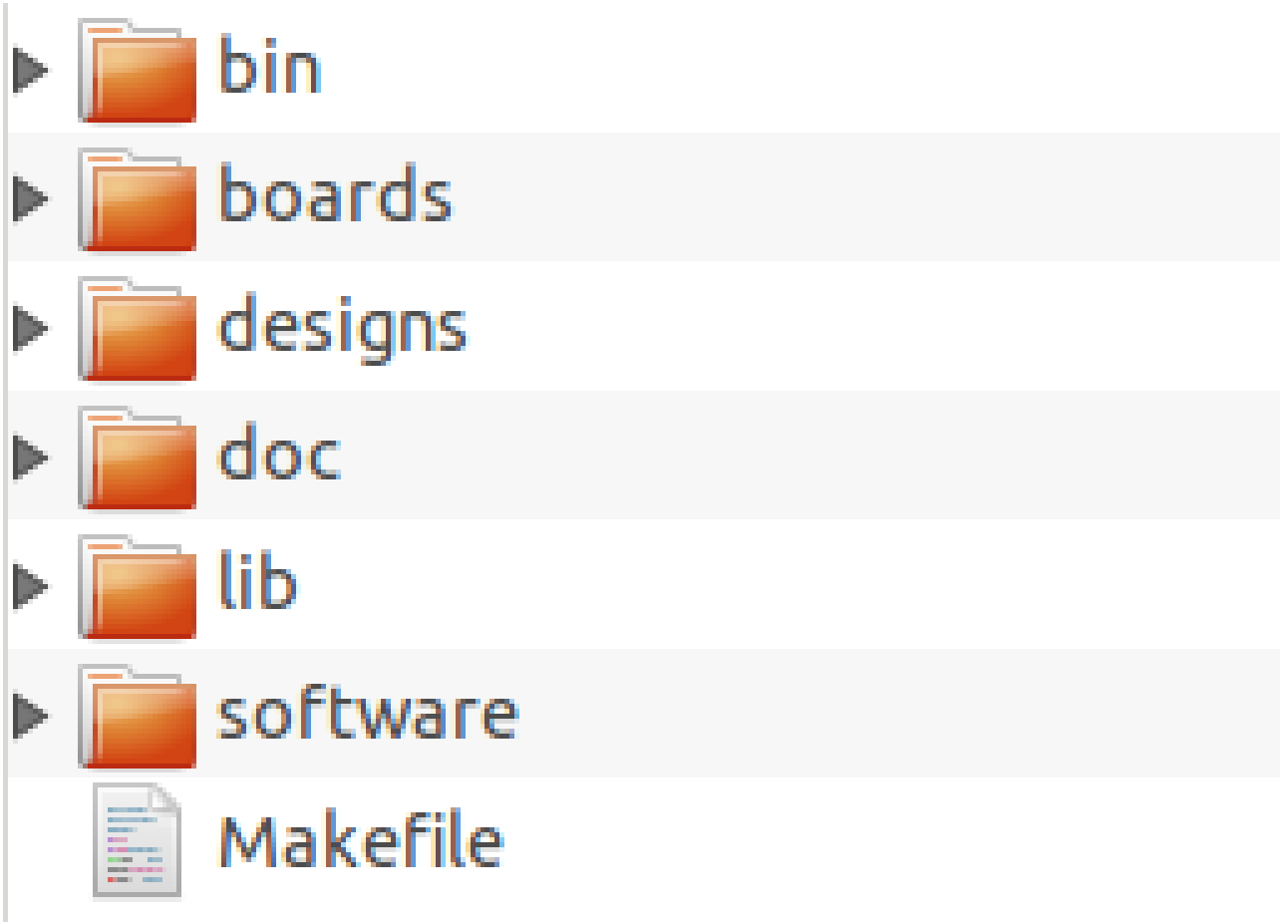
Reference: http://www.gaisler.com/images/gr712rc-large.jpg

LEON3 Based System

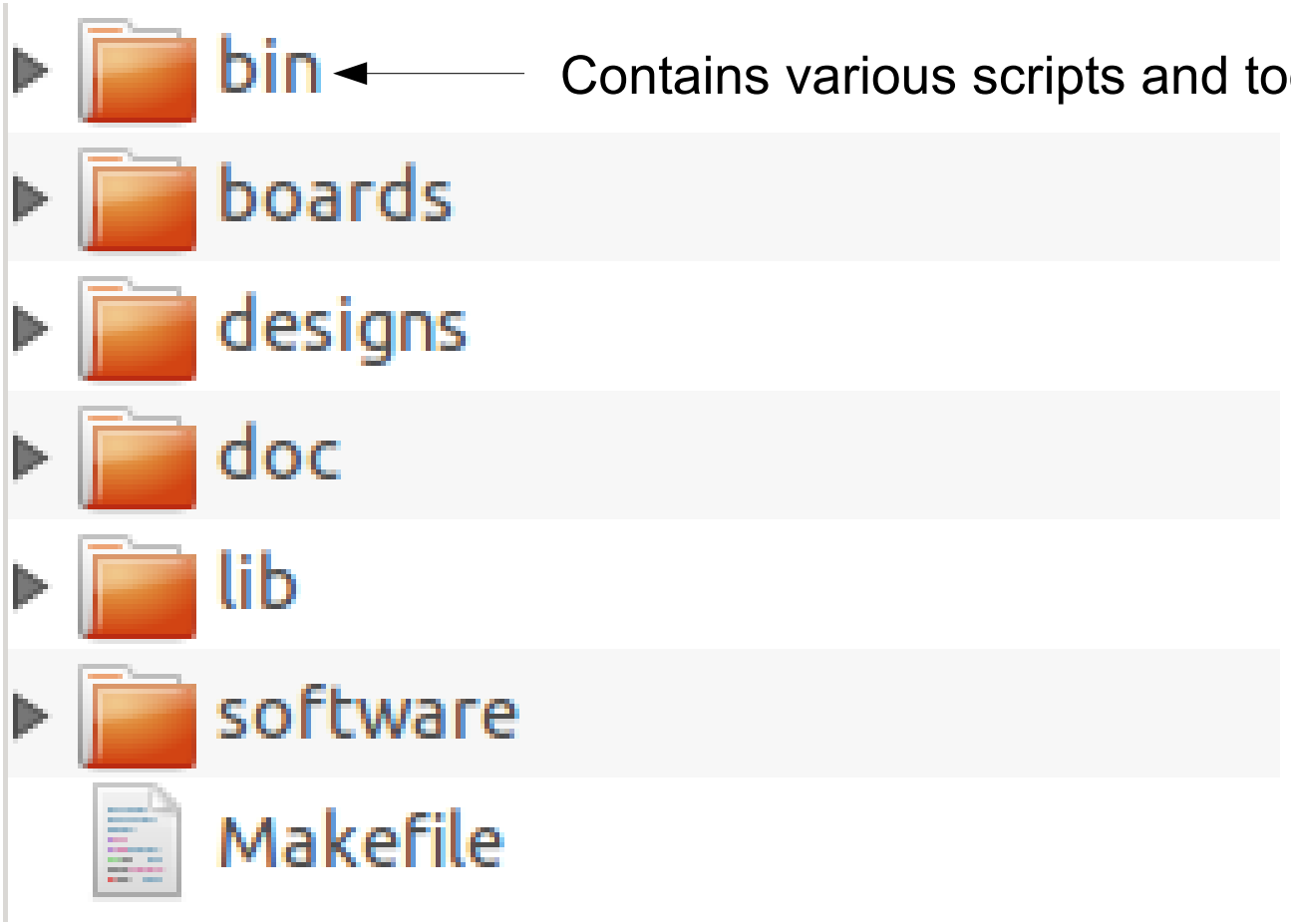


GRLIB Folder Structure ...

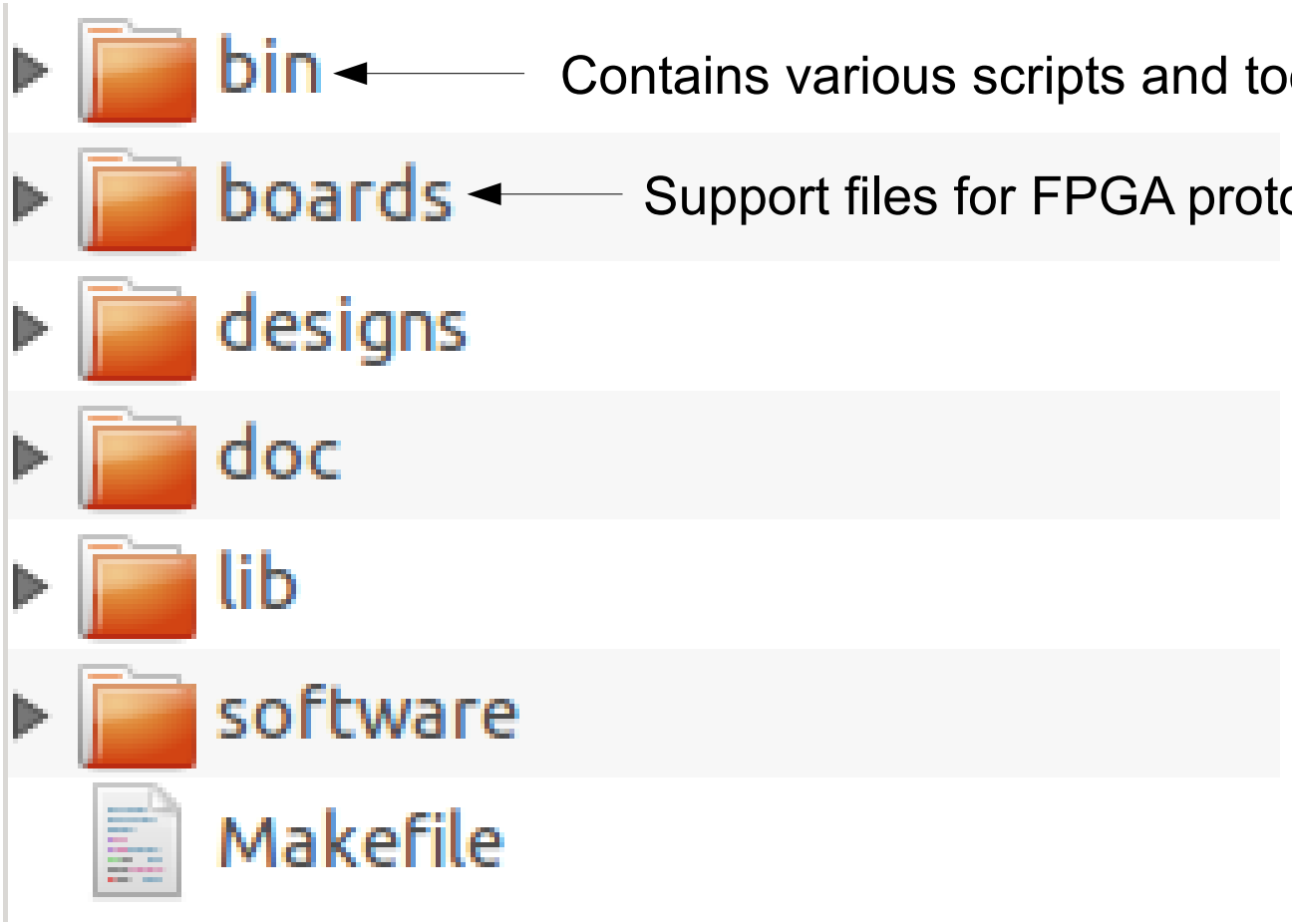
GRLIB Top Folders...



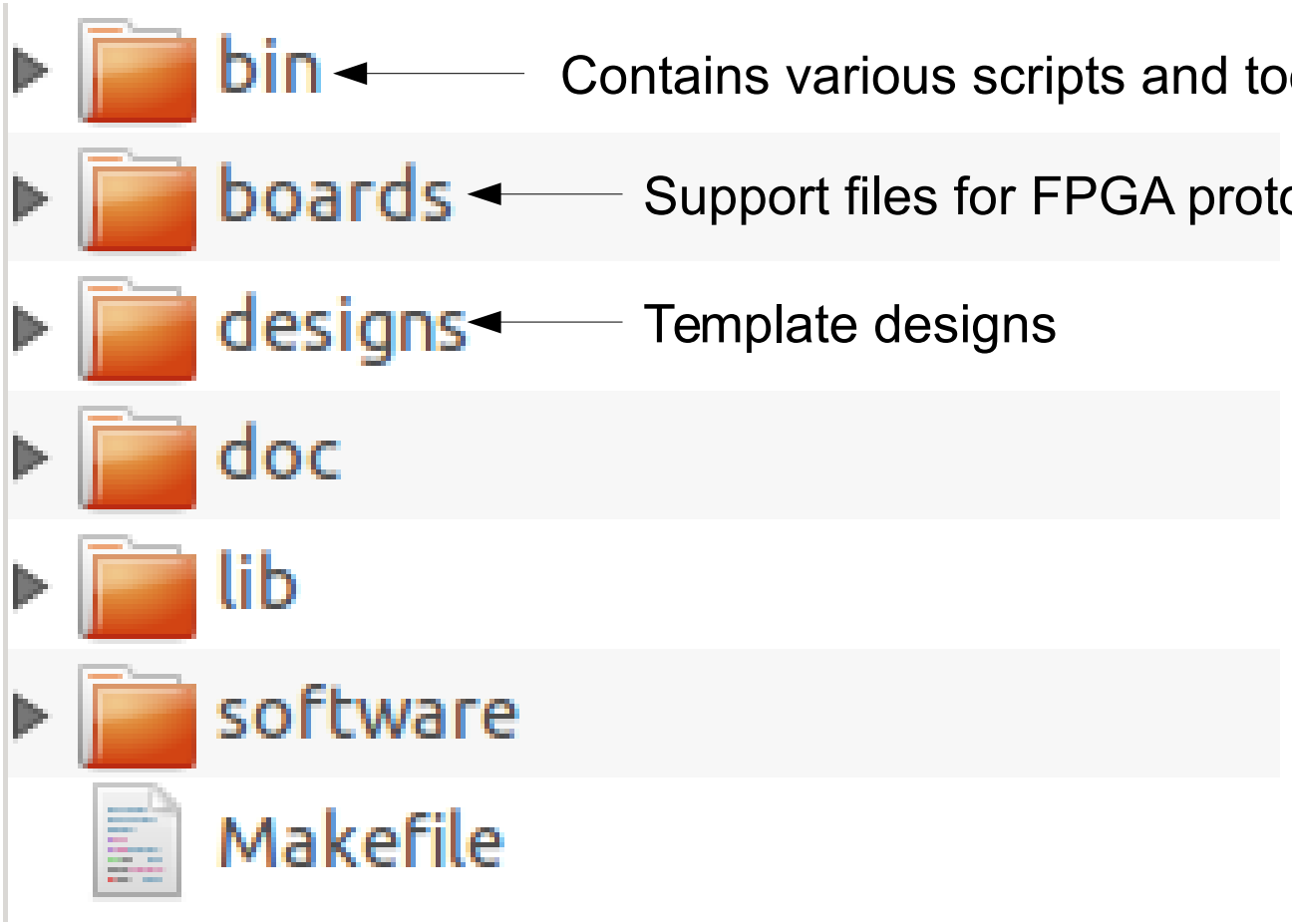
GRLIB Top Folders...



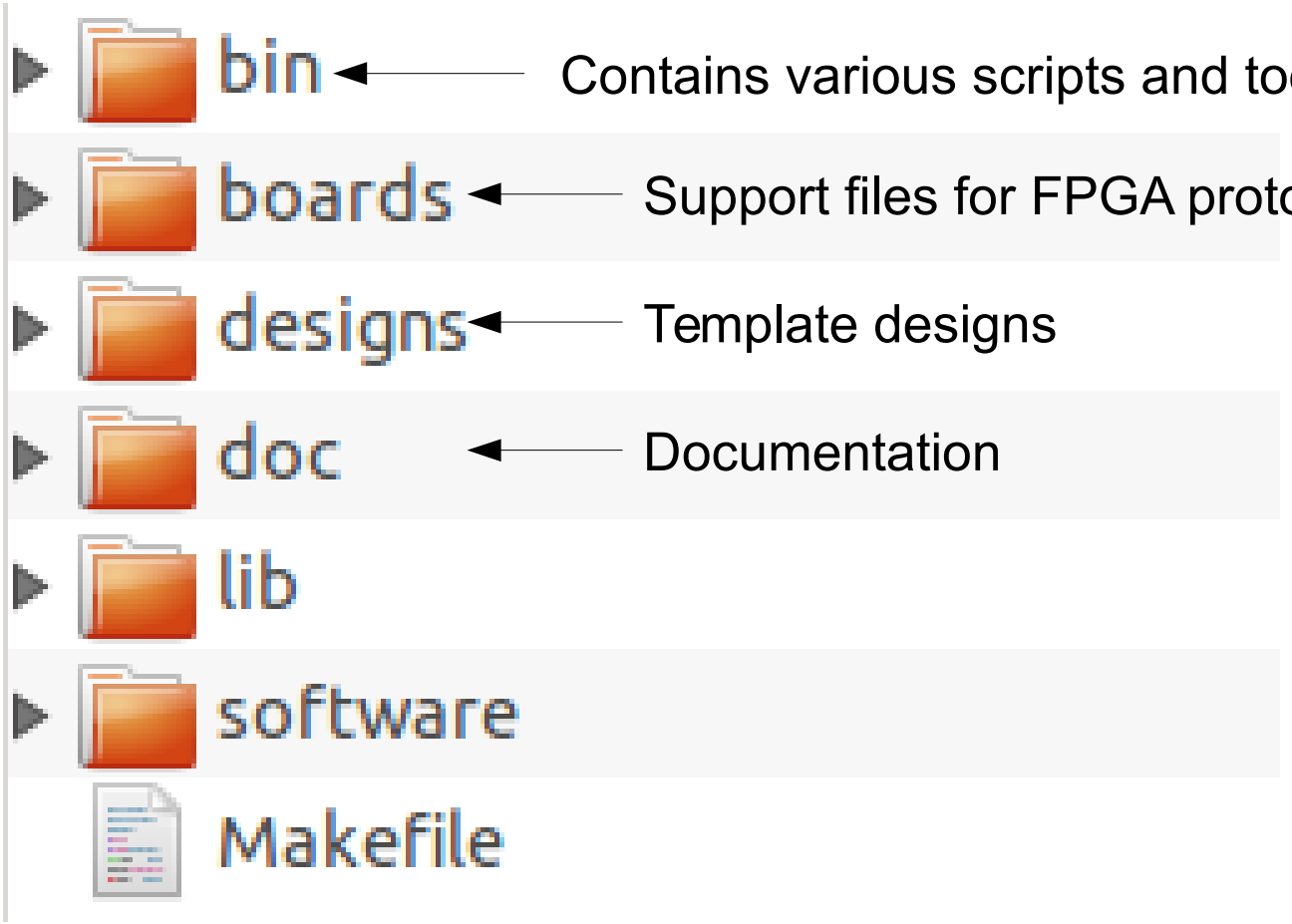
GRLIB Top Folders...



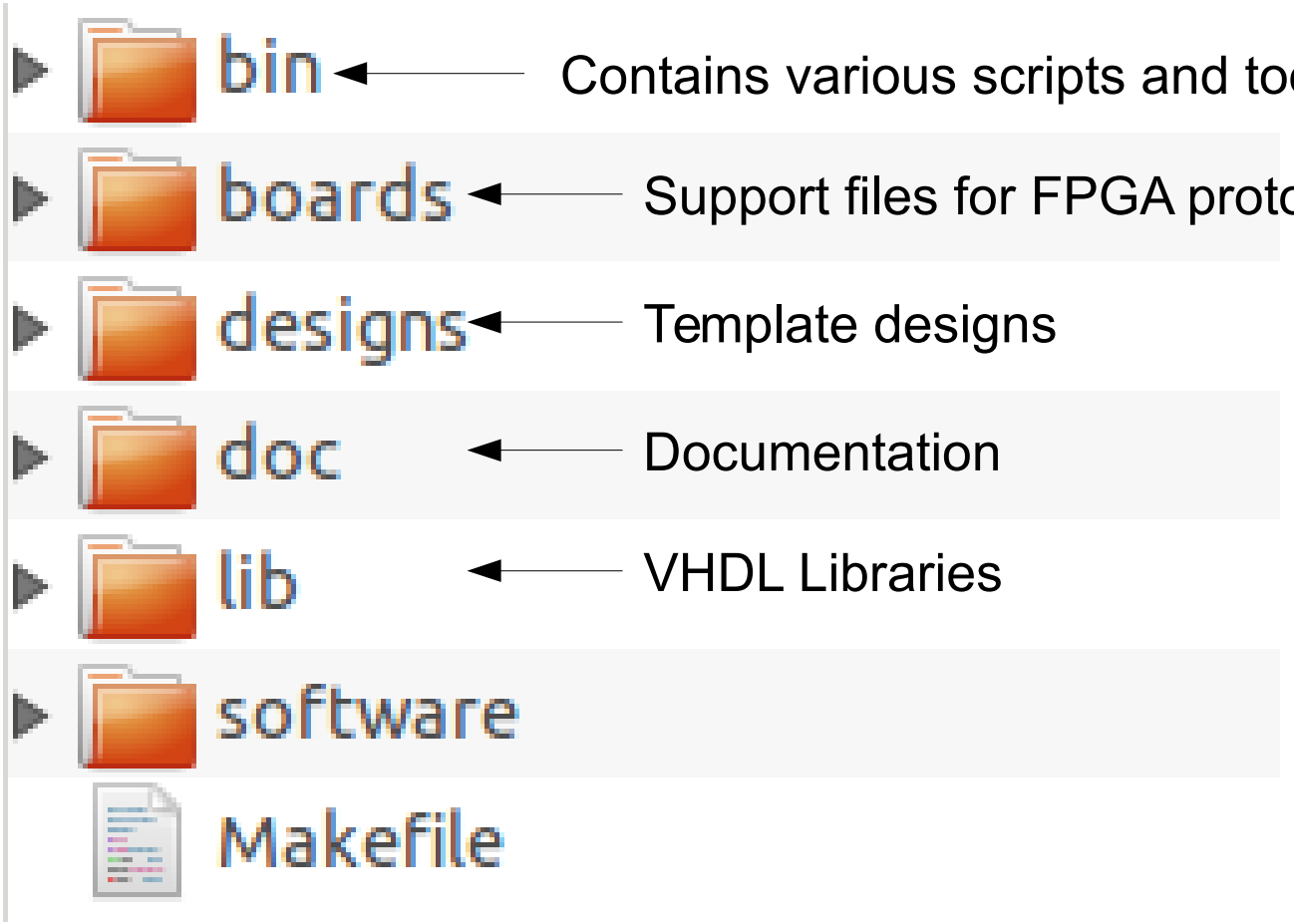
GRLIB Top Folders...



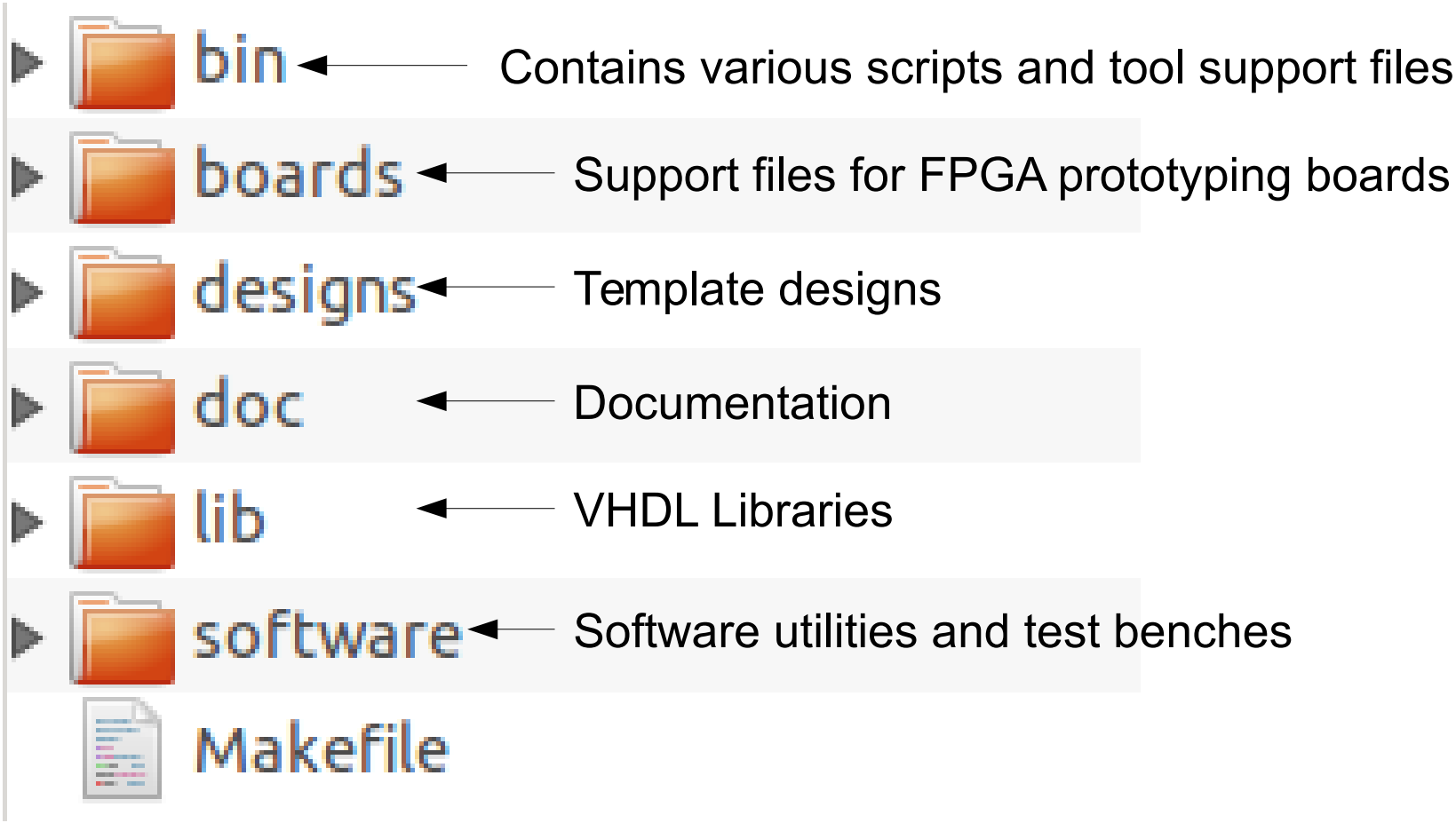
GRLIB Top Folders...



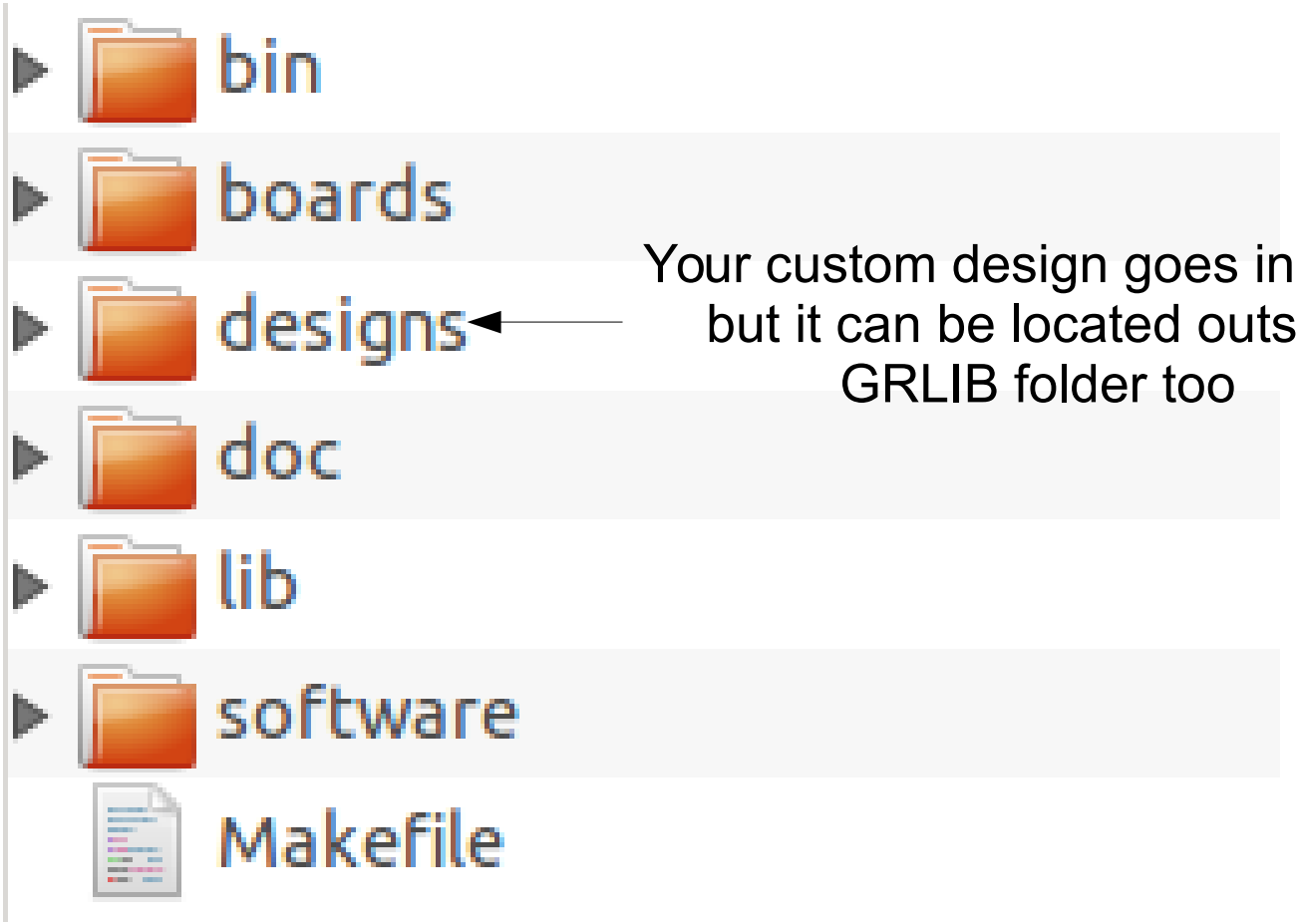
GRLIB Top Folders...



GRLIB Top Folders...



GRLIB Top Folders...



Our design: **leon3-*digilent-nexys4***

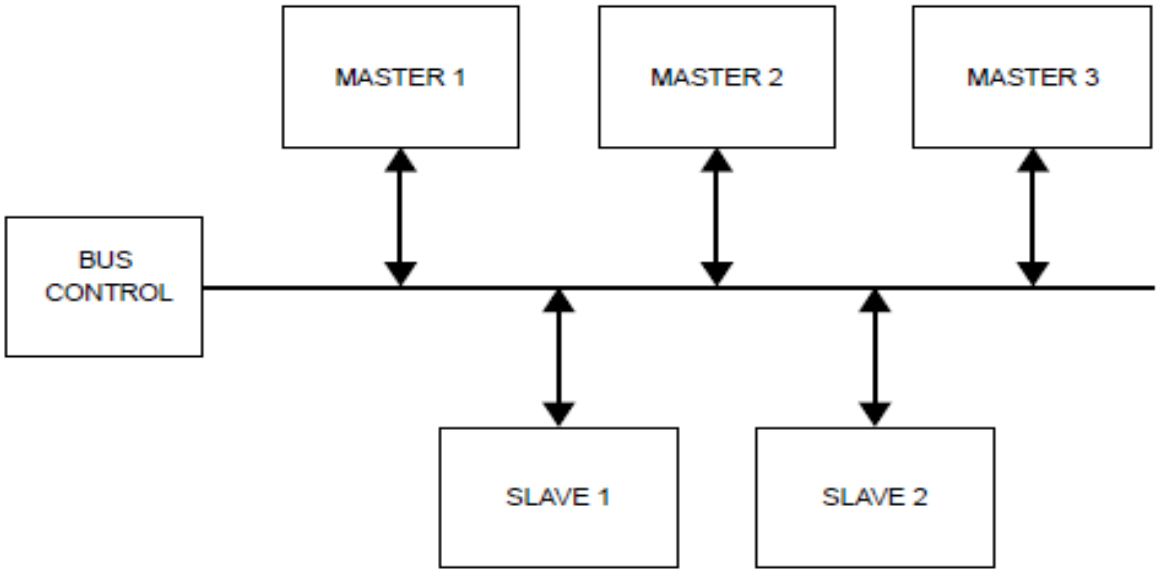
- Files of concern:
  - **leon3mp.vhd**: Top VHDL design file
  - **leon3mp.ucf**: FPGA user-constrained file
  - **Makefile**: Set GRLIB folder path, files to include for synthesis
  - Additional VHDL files created by you

Adding peripherals

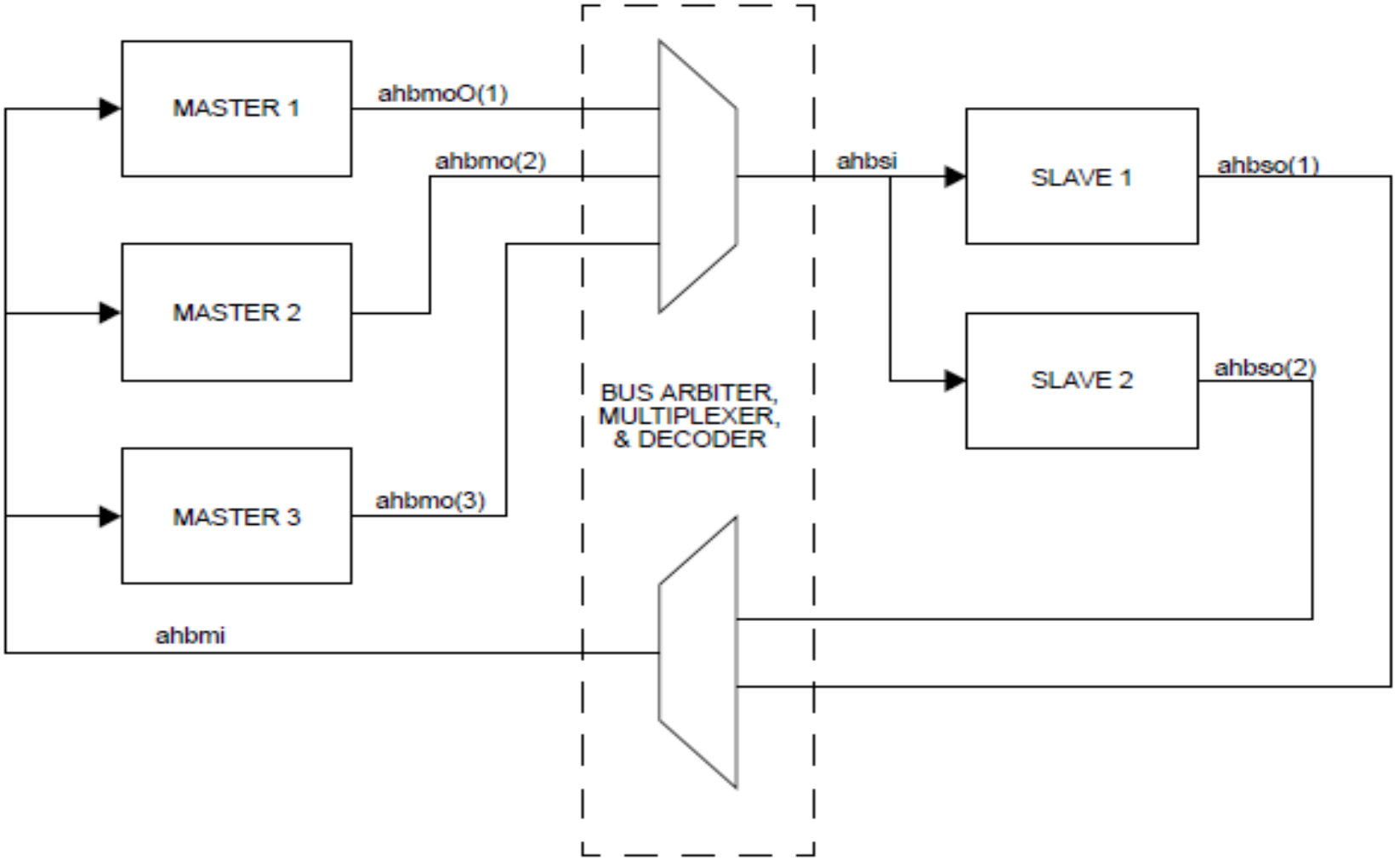
- Peripherals are instantiated from the top vhdl entity *leon3mp* in file *leon3mp.vhd*
- Peripherals can be added as AHB master/slave or APB slave
- Two types of banks defined for AHB Bus: AHB memory bank and AHB I/O Bank

AHB Bus

- AMBA (Advanced Microcontroller Bus Architecture) High-performance Bus
- Multi-master bus for connecting peripherals with high data rates and variable latency



AHB Bus


MASTER 1
MASTER 2
MASTER 3
ahbmoO(1)
ahbmo(2)
ahbmo(3)
ahbmi
BUS ARBITER,
MULTIPLEXER,
& DECODER
ahbsi
SLAVE 1
SLAVE 2
ahbso(1)
ahbso(2)

AHB Slave Instantation Example

```
----------------------------------------------------------------------
----------  Sklansky Adder on AHB ------------------------------------
----------------------------------------------------------------------
adderahb_if : adderahb
    generic map (hindex => 4, haddr => 16#500#, hmask => 16#FFF#)
    port map (rstn => rstn, clk => clkm, ahbsi => ahbsi, ahbso => ahbso(4));
```

leon3mp.vhd

AHB Slave Instantation Example

```
----------------------------------------------------------------------
----------  Sklansky Adder on AHB ------------------------------------
----------------------------------------------------------------------
adderahb_if : adderahb
    generic map (hindex => 4, haddr => 16#500#, hmask => 16#FFF#)
    port map (rstn => rstn, clk => clkm, ahbsi => ahbsi, ahbso => ahbso(4));
```

leon3mp.vhd

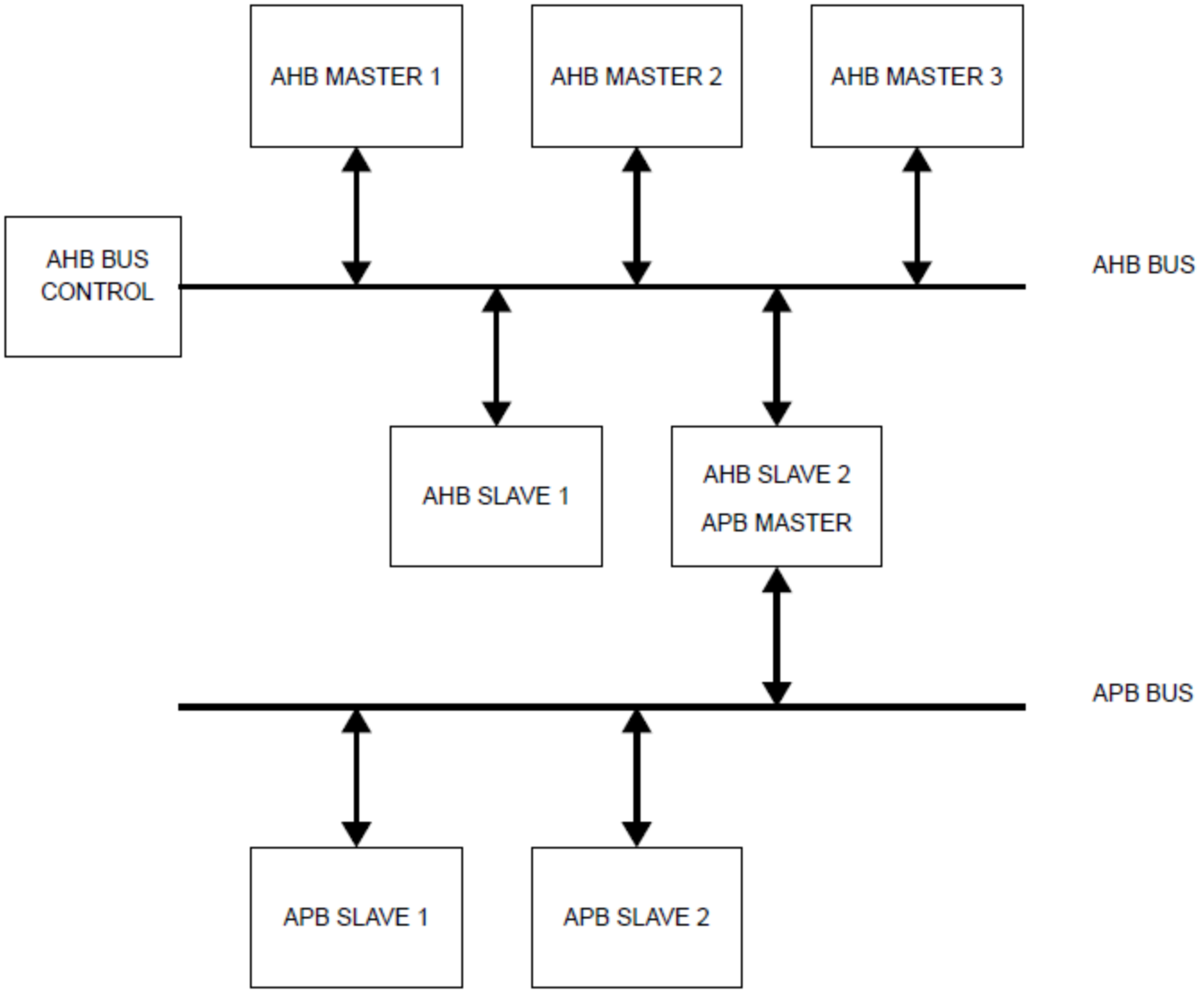
```
entity adderahb is
  generic(
    hindex      : integer := 0;
    haddr       : integer := 0;
    hmask       : integer := 16#fff#
    );
  port (
    rstn  : in  std_ulogic;
    clk   : in  std_ulogic;
    ahbsi : in ahb_slv_in_type;
    ahbso : out ahb_slv_out_type
  );
end entity adderahb;
```

adderahb.vhd

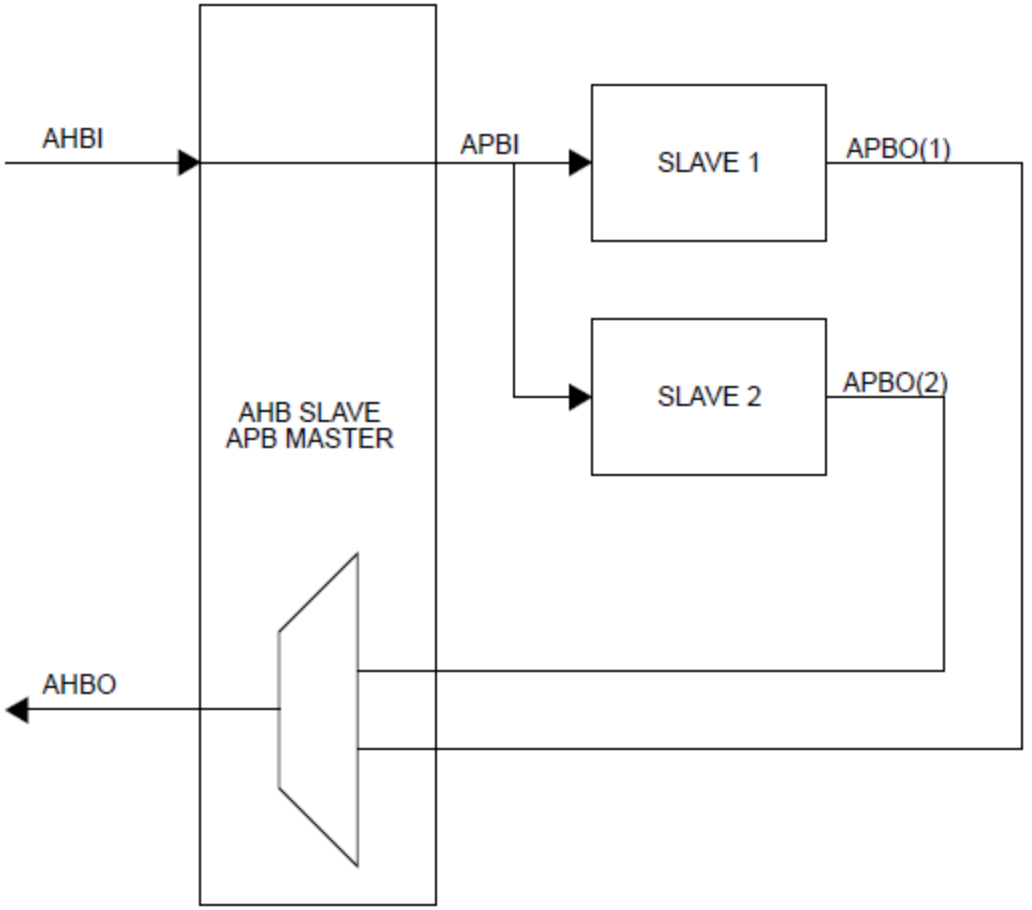
AHB Slave Addressing

- Most significant 12 bits used to generate *hsel*
- Minimum 1 Mbyte for each peripheral
- For increased address range, use *hmask*
- *Example:* For *hindex=5, haddr=0x250, hmask=0xFFF, address range = ?*
- *Example:* For *hindex=5, haddr=0x250, hmask=0xF00, address range = ?*

APB Bus


AHB MASTER 1
AHB MASTER 2
AHB MASTER 3
AHB BUS CONTROL
AHB BUS
AHB SLAVE 1
AHB SLAVE 2
APB MASTER
APB BUS
APB SLAVE 1
APB SLAVE 2

APB Bus


AHBI
APBI
SLAVE 1
APBO(1)
SLAVE 2
APBO(2)
AHB SLAVE
APB MASTER
AHBO

APB Slave Instantation Example

```
----------------------------------------------------------------------
----------  Sklansky Adder on APB ------------------------------------
----------------------------------------------------------------------
adder_if : adder
    generic map (pindex => 8, paddr => 8, pmask => 16#FFF#)
    port map (rstn => rstn, clk => clkm, apbi => apbi, apbo => apbo(8));
```

leon3mp.vhd

APB Slave Instantation Example

```
----------------------------------------------------------------------
----------  Sklansky Adder on APB ------------------------------------
----------------------------------------------------------------------
adder_if : adder
    generic map (pindex => 8, paddr => 8, pmask => 16#FFF#)
    port map (rstn => rstn, clk => clkm, apbi => apbi, apbo => apbo(8));
```

leon3mp.vhd

```
entity adder is
  generic(
    pindex      : integer := 0;
    paddr       : integer := 0;
    pmask       : integer := 16#fff#
    );
  port (
    rstn  : in  std_ulogic;
    clk   : in  std_ulogic;
    apbi   : in  apb_slv_in_type;
    apbo   : out apb_slv_out_type
  );

end entity adder;

architecture rtl of adder is
```

adderapb.vhd

APB Addressing

- Most significant 12 bits for selecting AHB/APB bridge
- Address 19:8 for selecting APB slave
- Minimum 256 bytes for each slave
- Use *pmask* for increased range

Changing CPU Frequency

- CPU Clock Generation in leon3mp.vhd:

```
-- main clock generator
clkgen0 : clkgen
  generic map (fabtech, CFG_CLKMUL, CFG_CLKDIV, 0, 0, 0, 0, 0, BOARD_FREQ, 0)
  port map (lclk, gnd, clkm, open, open, open, open, cgi, cgo, open, open, open);
```

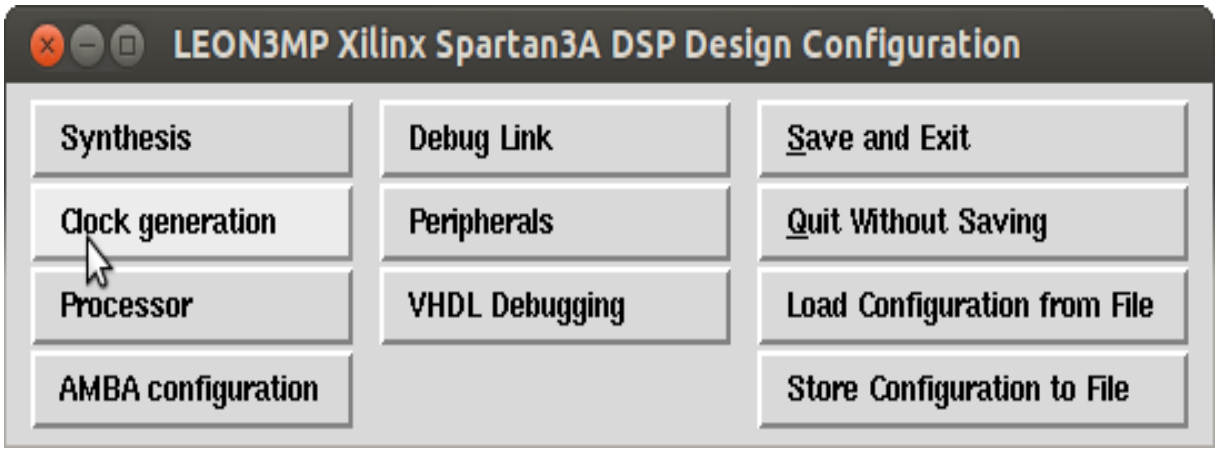
- clkm = lclk*(CFG_CLKMUL)/CFG_CLKDIV

Changing CPU Frequency

- CPU Clock Generation in leon3mp.vhd:

```
-- main clock generator
clkgen0 : clkgen
  generic map (fabtech, CFG_CLKMUL, CFG_CLKDIV, 0, 0, 0, 0, 0, BOARD_FREQ, 0)
  port map (lclk, gnd, clkm, open, open, open, open, cgi, cgo, open, open, open);
```

- clkm = lclk*(CFG_CLKMUL)/CFG_CLKDIV

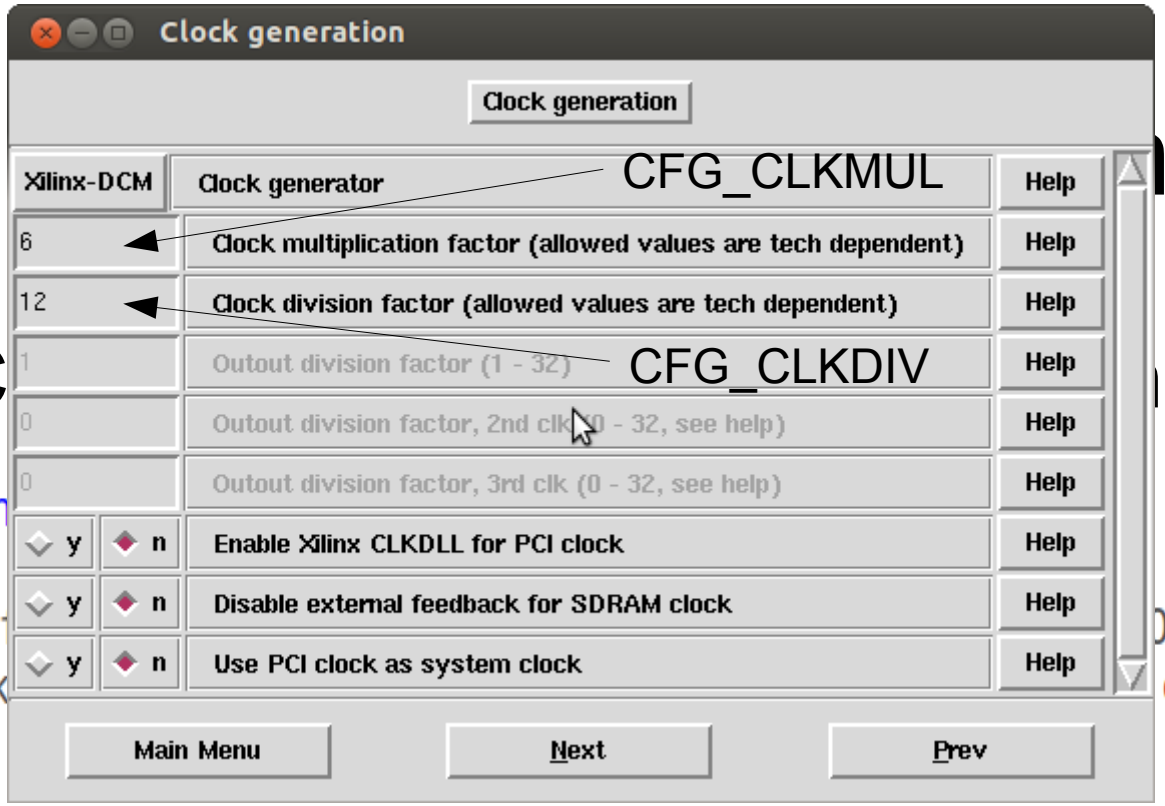


C ncy

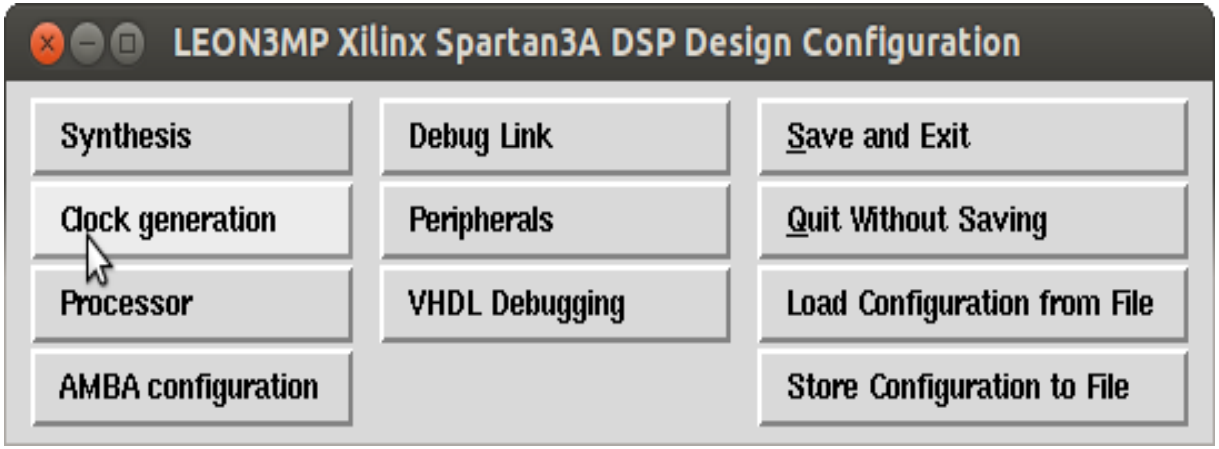
- CPU C d:

```
-- main clock gen
clkgen0 : clkgen
  generic map (
  port map (lclk
```

```
ARD_FREQ, 0)
open, open, open);
```



- clkm = lclk*(CFG_CLKMUL)/CFG_CLKDIV



GRLIB Design Practices

- Records for grouping input and output signals
- Two-process method
- Packages for grouping all component declarations

Further Reading

- GRLIB User Manual
- AMBA Specifications
- VHDL 2 Process method (from Gaisler)

The End!