# ARM® Cortex®M0 DesignStart® System Design

## Example Design Kit

## Quick Start Guide

# END USER LICENCE AGREEMENT



THIS END USER LICENCE AGREEMENT (“LICENCE”) IS A LEGAL AGREEMENT BETWEEN YOU AND ARM LIMITED ("ARM") FOR THE USE OF THE SOFTWARE EXAMPLE ACCOMPANYING THIS LICENCE. ARM IS ONLY WILLING TO LICENSE THE SOFTWARE EXAMPLE TO YOU ON CONDITION THAT YOU ACCEPT ALL OF THE TERMS IN THIS LICENCE. BY INSTALLING OR OTHERWISE USING OR COPYING THE SOFTWARE EXAMPLE YOU INDICATE THAT YOU AGREE TO BE BOUND BY ALL OF THE TERMS OF THIS LICENCE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENCE, ARM IS UNWILLING TO LICENSE THE SOFTWARE EXAMPLE TO YOU AND YOU MAY NOT INSTALL, USE OR COPY THE SOFTWARE EXAMPLE.

ARM hereby grants to you, subject to the terms and conditions of this Licence, a non-exclusive, worldwide, non-transferable, copyright licence only to redistribute and use in source and binary forms, with or without modification, for academic purposes provided the following conditions are met:

a) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

b) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE EXAMPLE IS PROVIDED BY THE COPYRIGHT HOLDER "AS IS" AND ARM EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THIS SOFTWARE EXAMPLE. IN NO EVENT SHALL ARM BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES OF ANY KIND WHATSOEVER WITH RESPECT TO THE SOFTWARE EXAMPLE. ARM SHALL NOT BE LIABLE FOR ANY CLAIMS, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE EXAMPLE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE EXAMPLE. FOR THE AVOIDANCE OF DOUBT, NO PATENT LICENSES ARE BEING LICENSED UNDER THIS LICENSE AGREEMENT.

# 1 Introduction

The ARM Cortex-M0 processor is the fastest licensing ARM processor today. The low power, small gate count and code footprint of the processor makes it ideal for MCU and mixed signal applications, providing 32-bit performance and efficiency in an 8/16-bit footprint. As the entry-level ARM processor, it is also ideal for designers, students and researchers who need low cost access to industry standard processor IP.
The DesignStart version contains obfuscated gate-level Verilog of the processor, and is available to approved educational institutions and companies with no initial payment necessary. Although this special version has some limitations relative to the full ARM Cortex-M0 processor, it is fully software compatible and enables designers to create manufacturable designs.

The example design kit (EDK) helps you start designing complete systems (hardware + software) with CM0DS on an FPGA board. It includes:

1. A selections of simple AHB-Lite peripherals
2. AHB-Lite Bus infrastructure components
3. Example systems which can be used as design templates
4. Software examples with Keil.

# 2 Directory Structure

**Example Design Kit**
||
**AHB_Peripherals** → Contains all the AHB-Lite peripherals with source code
||
AHBL_7SEGDEC
AHBL_APB
AHBL_DUMMY1
AHBL_DUMMY2
AHBL_GPIO
AHBL_KB
AHBL_LED
AHBL_MEM
AHBL_SRAMFLASH
AHBL_TIMER
AHBL_UART
AHBL_VGA
ClockDivider
CM0DS_AHBL
PBDebouncer
UCF_Digilent_Nexys

**Example_SoC** → Contains all the Example Systems with source code
||
ARMSOC_0
ARMSOC_1
ARMSOC_2
ARMSOC_3
ARMSOC_4
ARMSOC_7SEG
ARMSOC_INTERRUPT
ARMSOC_MEMCTRL

**ARMSOC_xx**
||
Software
Xilinx
AHBDCD.v
AHBLITE_SYS.v
AHBMUX.v
ARMSOC_S6.ucf
SystemInfo.pdf

| | |
|---|---|
| **Software:** | This contains KEIL project file with the software source code. |
| **Xilinx:** | This folder is empty. It is intended for FPGA synthesis work directory. |
| **AHBDCD & AHBMUX:** | Simple combination decoder & mux which forms the AHBLite bus fabric. |
| **AHBLITE_SYS.v:** | This is the top level module which integrates CM0DS core with AHBLite peripherals using AHBLite bus matrix. |
| **ARMSOC_S6.ucf:** | Xilinx Spartan6 User constraint file (UCF) for Digilent Nexus3 board. |
| **SystemInfo:** | It gives you an overview of the example system and its memory map. |

# 3 Before you start

1. Download CM0DS Core from www.arm.com/products/designstart

2. Download KEIL MDK for software development http://www.keil.com/arm/mdk.asp
   Note: There is a free version which you can download.

3. Download AMBA3 AHBLite Specification http://infocenter.arm.com

4. Depending on your target FPGA board, please download the relevant FPGA synthesis tool. For examples, Xilinx ISE if you are targeting your design for Xilinx board or Altera Quartus if you are using Altera FPGA board.
   Note: The above example systems were built for Digilent Nexys3 board which comes with Spartan6 FPGA. But the design can be ported to any other FPGA board on similar lines.

# 4 Prerequisite:

1. Digital systems design.
2. VHDL or Verilog
3. C programming or ARM Assembly programming
4. Hands on working with any FPGA board

# 5 FPGA Synthesis Flow

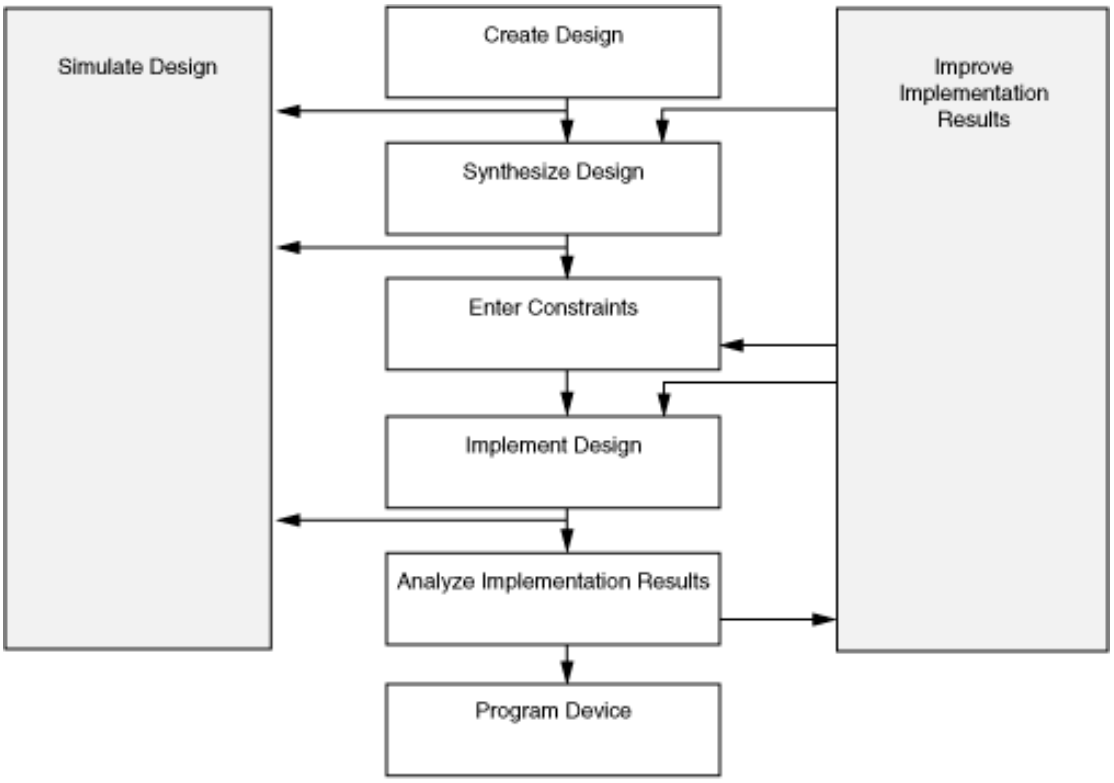
1. Create the project file

2. Add the necessary modules.
   NOTE: Check the systeminfo.pdf to see what modules (source files) you need to include for that particular example. For all the examples the top level is AHBLITE_SYS.v

3. Create your pin constraint file depending on your target FPGA board.
   NOTE: Constraint File for Digilent Nexys3 is already included.

4. Synthesize and Implement the design

5. Generate the FPGA configuration file.
   Example: SOF/POF files for Altera and BIT file for Xilinx



# 6 Software Flow

1. Create the project file
   Note: If you using KEIL MDK as your software development tool, then the project file is already included in the example pack. It resides inside software directory.

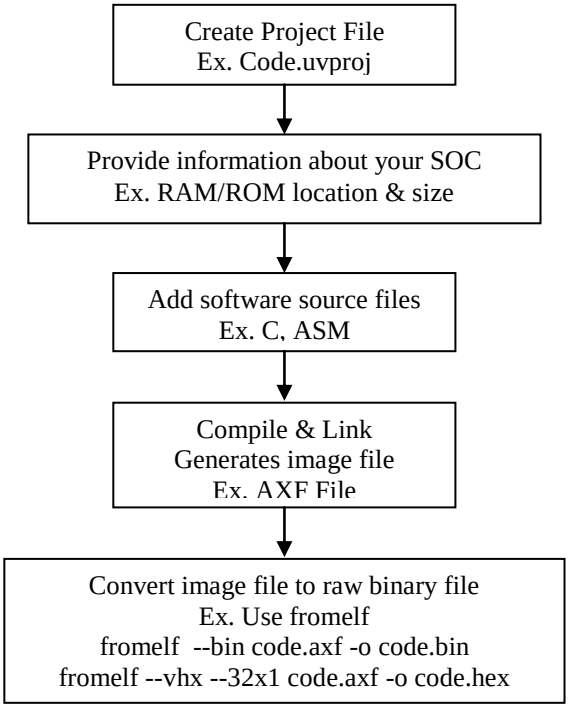
2. Give information regarding your hardware to compiler/linker
   - Device information → Cortex M0
   - Target information → RAM and ROM position
   - R/O and R/W Base address
   - Information to place vector table at 0x0

3. Add the software source files present in the software directory.

4. Compile & link the design to generate the image file.
   - KEIL generate AXF image file.

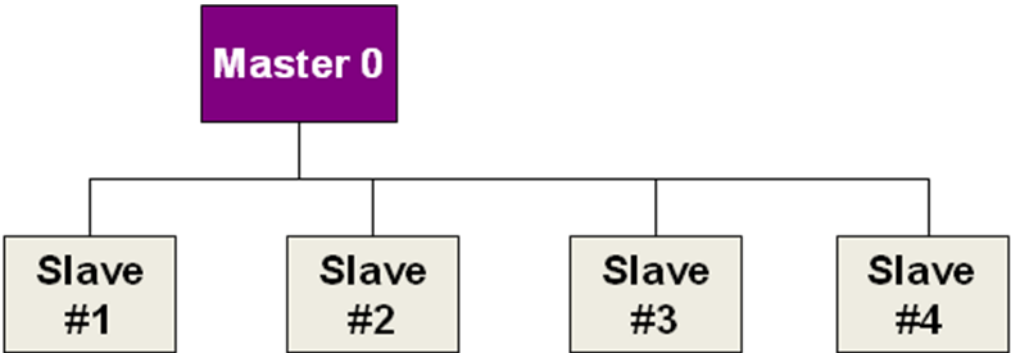
5. Convert the image file to raw binary which can be used to initialise memory
   - Use fromelf to convert image file to raw binary
   - Example: fromelf --bin code.axf -o code.bin
   - The above converts code.axf to raw binary code.bin
   - Example: fromelf --vhx --32x1 code.axf -o code.hex
   - The above convert's code.axf to binary file is ASCII format.

6. Use the raw binary to initialise code memory
   - If your code memory is internal then you need to complete your software flow before you synthesize your design
   - If your code memory is external (for example in Nexys3you can use onboard PSRAM & FLASH as your main memory without any internal memory), then you can keep the software flow independent from your FPGA flow.



# 7 AHBLite System Design Overview

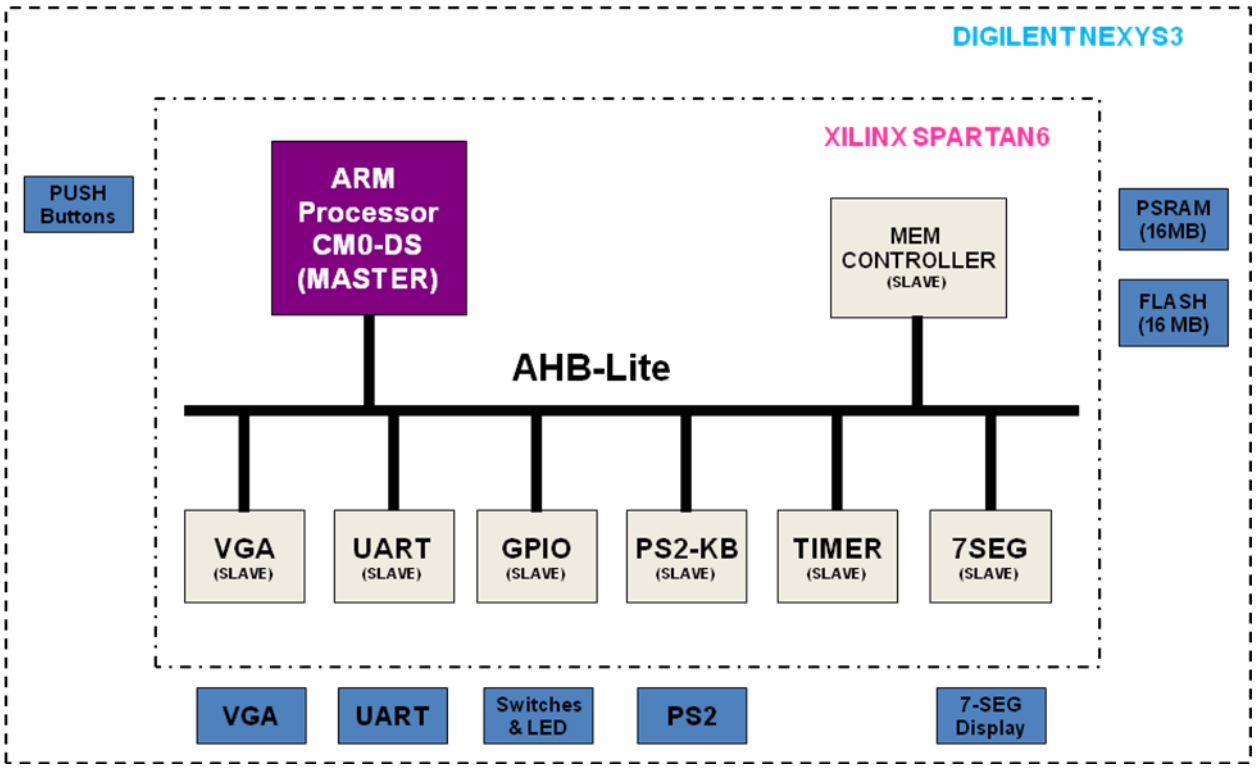
1. Master Slave Architecture




2. AHB-Lite transactions supported by CM0DS
   - Register Read
   - Register Write
   - Burst Read and Burst Write are not supported by CM0DS which makes AHBLite even lighter.

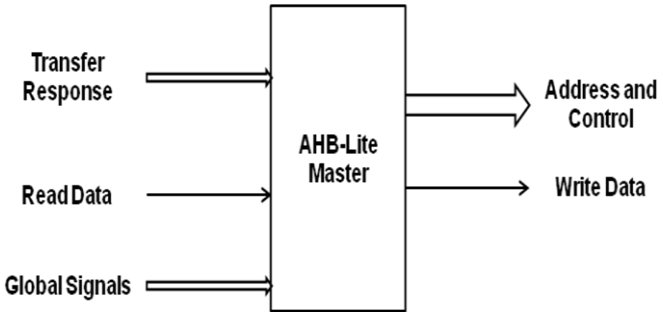
3. AHB-Lite Features
   - Single Clock Edge operation. All transactions happen on posedge of the clock.
   - Uni-directional busses which means no tri-state signals
   - Pipelined operation

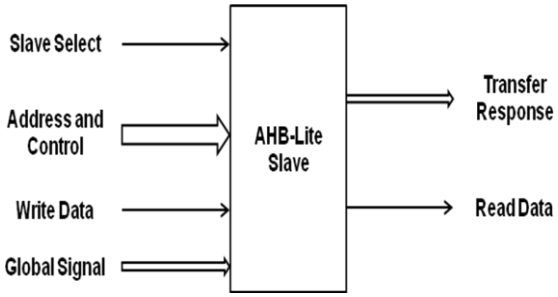
4. Example AHBLite System



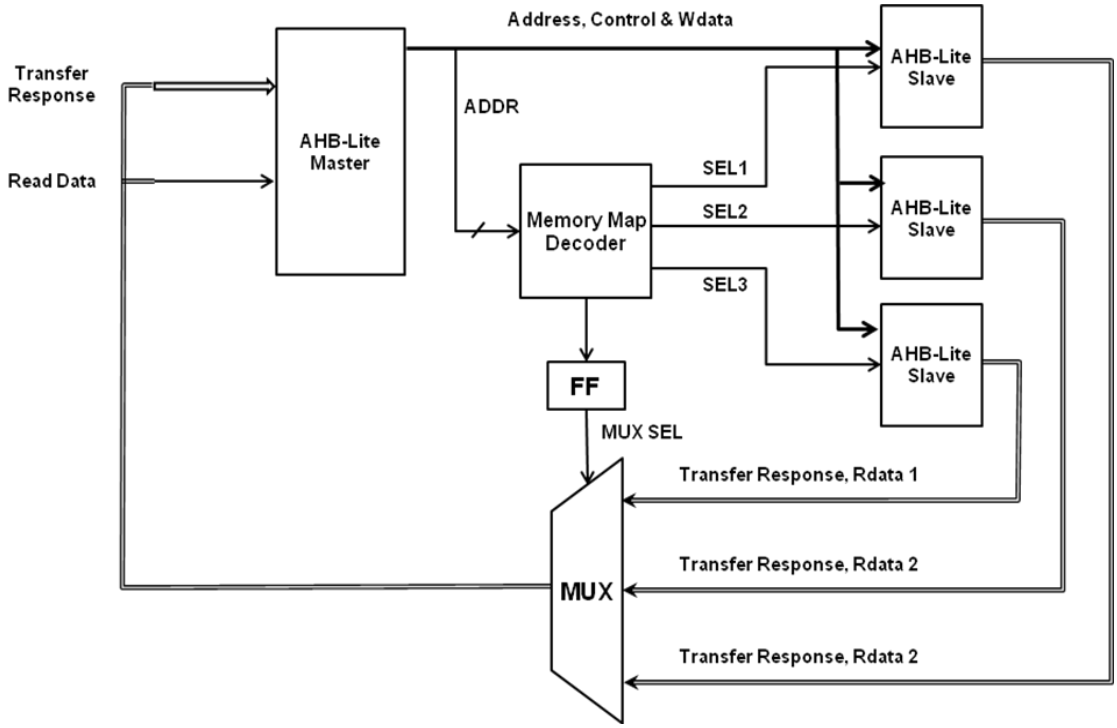
5. Components of AHB-Lite System
   - Master
   - Slaves
   - Address Decoder
   - Slave Multiplexer

6. Master and Slave Block Diagram

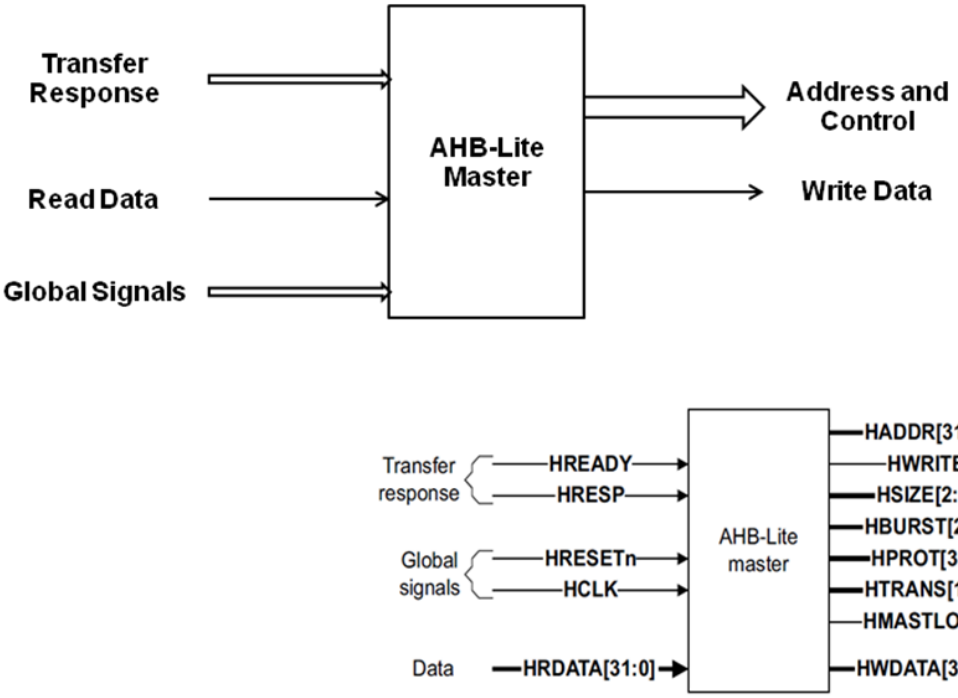


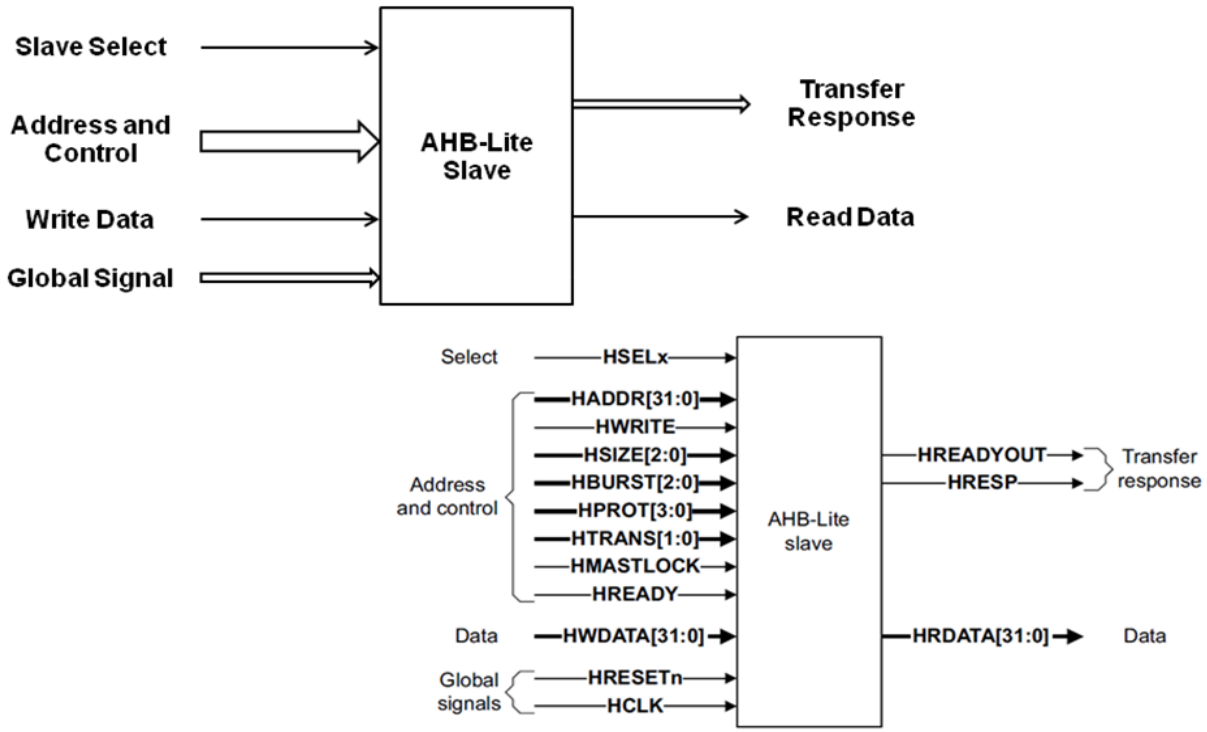

7. AHBLite System Block Diagram



8. AHBLite Master Interface



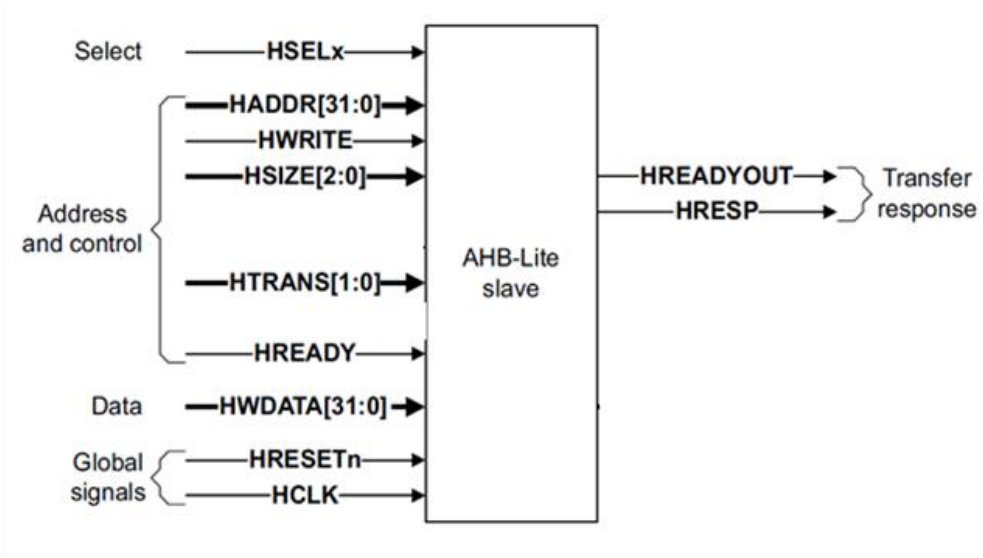

9. AHBLite Slave Interface



10. CortexM0 being the entry level ARM processor uses only a subset of AHBLite. As mentioned before the transactions supported by CM0 are simple Register write and read. This makes AHBLite even lighter. The unused control signals (below) originating from Master can be completely ignored while designing the slave for CM0 system.

    - HBURST
    - HMASTLOCK
    - HPROT

The slave interface now looks like below,




11. Signals Description

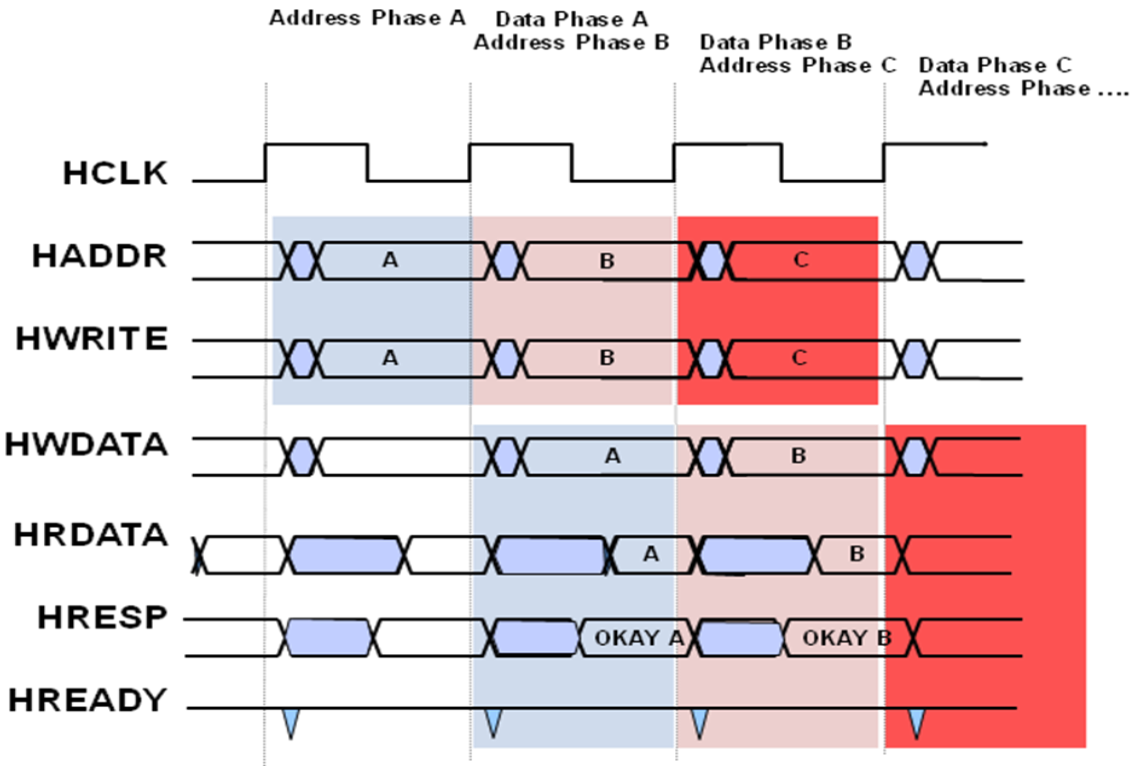
- HSEL → Active high slave select signal comes from address decoder
- HADDR → 32 bit address signal
- HWRITE → Active high for write transaction, active low for read transaction
- HREADY → Transaction is valid only when HREADY is active high
- HWDATA → 32 bit write data from Master to slave
- HRESETn → Active low reset signal
- HCLK → System Clock
- HREADYOUT → Slave can pull this signal low if it needs more than one clock cycle to complete the transaction.
- HRESP → Error response signalling back to master (active high)

- HTRANS →When 2'b00, it's a idle transaction, when 2'b10 then it's a valid non sequential transaction. The other two states 2'b01 and 2'b11 are not used by CM0.

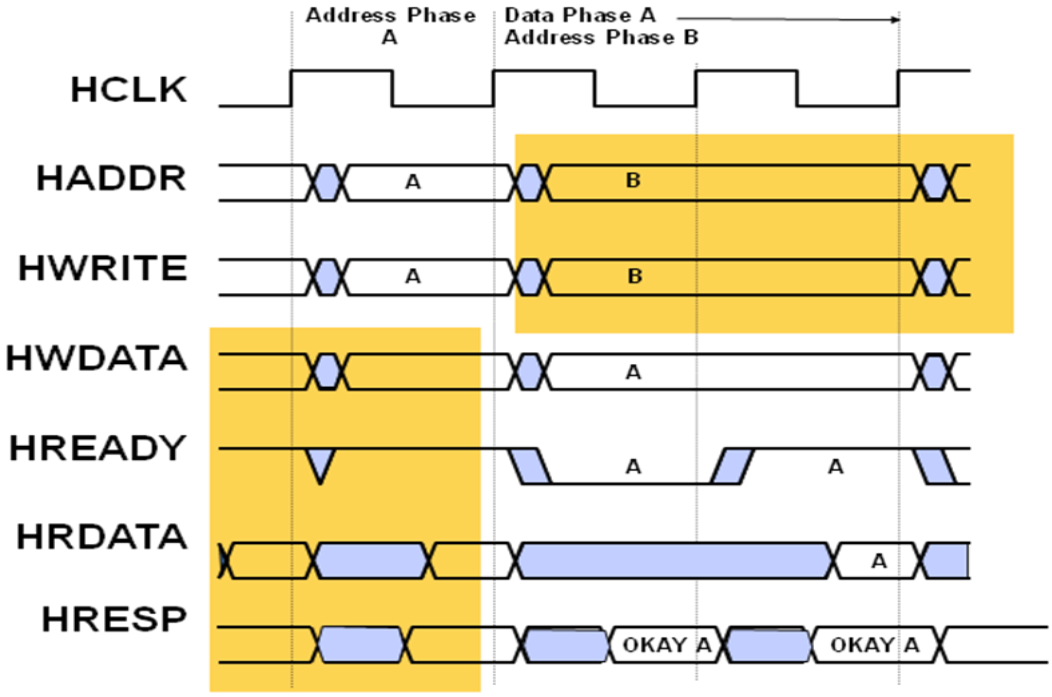
12. Read and Write Transactions can be byte (8bit), halfword (16bit) or word (32'bit). This information is given by HSIZE as below. Depending on the last two bits of the address the corresponding bytes lines are valid.
    - Note: If you are designing a slave which always does 32 bit transaction, then you can ignore HSIZE altogether.

| Address-phase: | | Data-phase: | | | |
|---|---|---|---|---|---|
| HSIZE [1:0] | HADDR [1:0] | HxDATA [31:24] | HxDATA [23:16] | HxDATA [15:8] | HxDATA [7:0] |
| 00 | 00 | - | - | - | Rd[7:0] |
| 00 | 01 | - | - | Rd[7:0] | - |
| 00 | 10 | - | Rd[7:0] | - | - |
| 00 | 11 | Rd[7:0] | - | - | - |
| 01 | 00 | - | - | Rd[15:8] | Rd[7:0] |
| 01 | 10 | Rd[15:8] | Rd[7:0] | - | - |
| 10 | 00 | Rd[31:24] | Rd[23:16] | Rd[15:8] | Rd[7:0] |

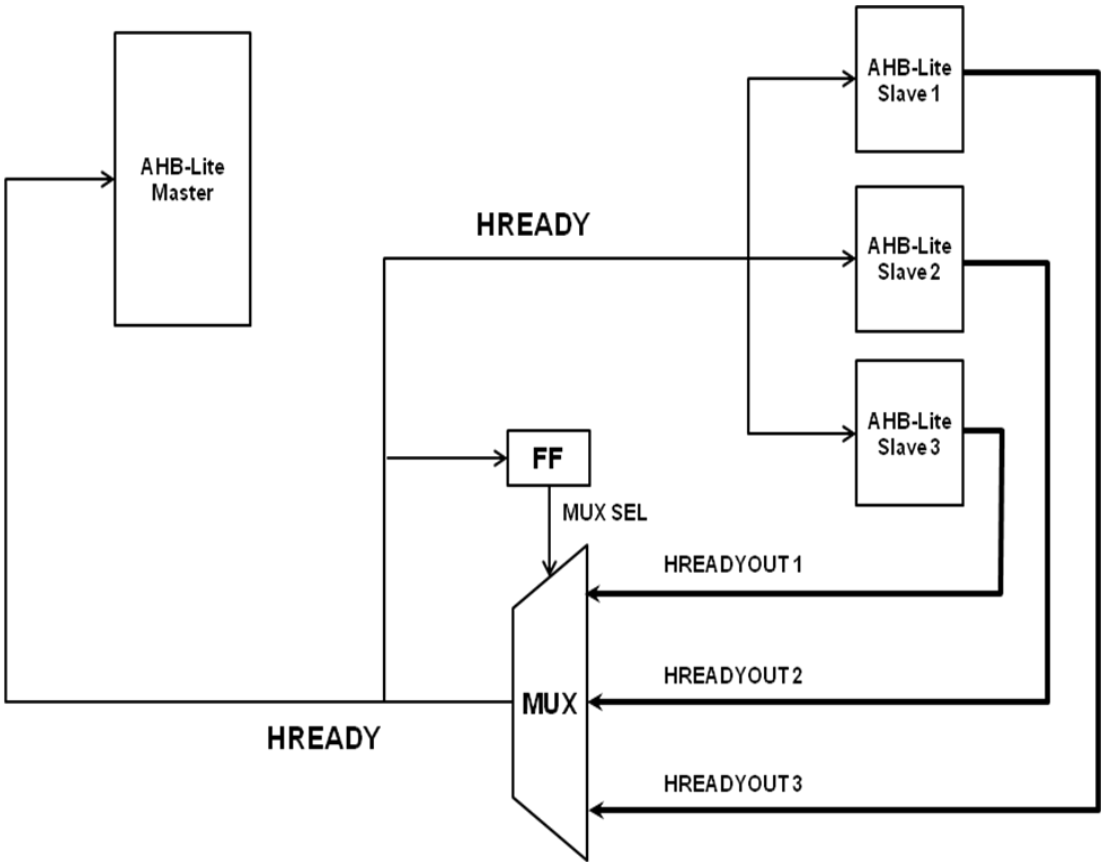
13. Pipelined Transaction (simplified version)



14. Pipelined Transaction with wait states (simplified version)
    - In the below example Slave A adds a single cycle wait state
    - This results in master extending the address phase of B




15. Extending address phase during wait states using HREADY



16. Example AHBLite Slave AHB2LED
    - Does a simple 32 bit register write
    - Does a simple 32 bit register read
    - The transaction is zero wait states

```
//////////////////////////////////////////////////////////////////////////////////////////////////

module AHB2LED(
        //AHBLITE INTERFACE
                //Slave Select Signals
                        input wire HSEL,
                //Global Signal
                        input wire HCLK,
                        input wire HRESETn,
                //Address, Control & Write Data
                        input wire HREADY,
                        input wire [31:0] HADDR,
                        input wire [1:0] HTRANS,
                        input wire HWRITE,
                        input wire [2:0] HSIZE,

                        input wire [31:0] HWDATA,
                // Transfer Response & Read Data
                        output wire HREADYOUT,
                        output wire [31:0] HRDATA,
                //LED Output
                        output wire [7:0] LED
);


//Address Phase Sampling Registers
 reg rHSEL;
 reg [31:0] rHADDR;
 reg [1:0] rHTRANS;
 reg rHWRITE;
 reg [2:0] rHSIZE;

 reg [31:0] rLED;

//Address Phase Sampling
 always @(posedge HCLK or negedge HRESETn)
 begin
        if(!HRESETn)
        begin
                rHSEL           <= 1'b0;
                rHADDR          <= 32'h0;
                rHTRANS         <= 2'b00;
                rHWRITE         <= 1'b0;
                rHSIZE          <= 3'b000;
        end
  else if(HREADY)
  begin
                rHSEL           <= HSEL;
                rHADDR          <= HADDR;
                rHTRANS         <= HTRANS;
                rHWRITE         <= HWRITE;
                rHSIZE          <= HSIZE;
  end
 end

//Data Phase data transfer
```

```
 always @(posedge HCLK or negedge HRESETn)
 begin
  if(!HRESETn)
    rLED <= 32'h0;
  else if(rHSEL & rHWRITE & rHTRANS[1])
    rLED <= HWDATA;
 end

//Transfer Response
 assign HREADYOUT = 1'b1; //Single cycle Write & Read. Zero Wait state operations

//Read Data
 assign HRDATA = rLED;

 assign LED = rLED[7:0];

endmodule
////////////////////////////////////////////////////////////////////////////////////////////////////
```

17. Further Reading
    - AMBA3 AHBLite specification
    - Source code for AHBLite peripherals

# 8 Conclusion

With CortexM0 DesignStart and Example Design Kit you can now design your own digital system starting from SoC Design to software development. You can quickly prototype the design on any suitable FPGA board available in the market. The kit was tested on Digilent Nexys3 board and hence the example SoC can be readily ported on Nexys3 with minimum effort.

The CortexM0DS is a fully synthesizable core. With suitable ASIC synthesis tools and standard cell library you can take your design through the ASIC synthesis flow. This will help you get an overview of the entire SoC design cycle starting from Specification to design to implementation.