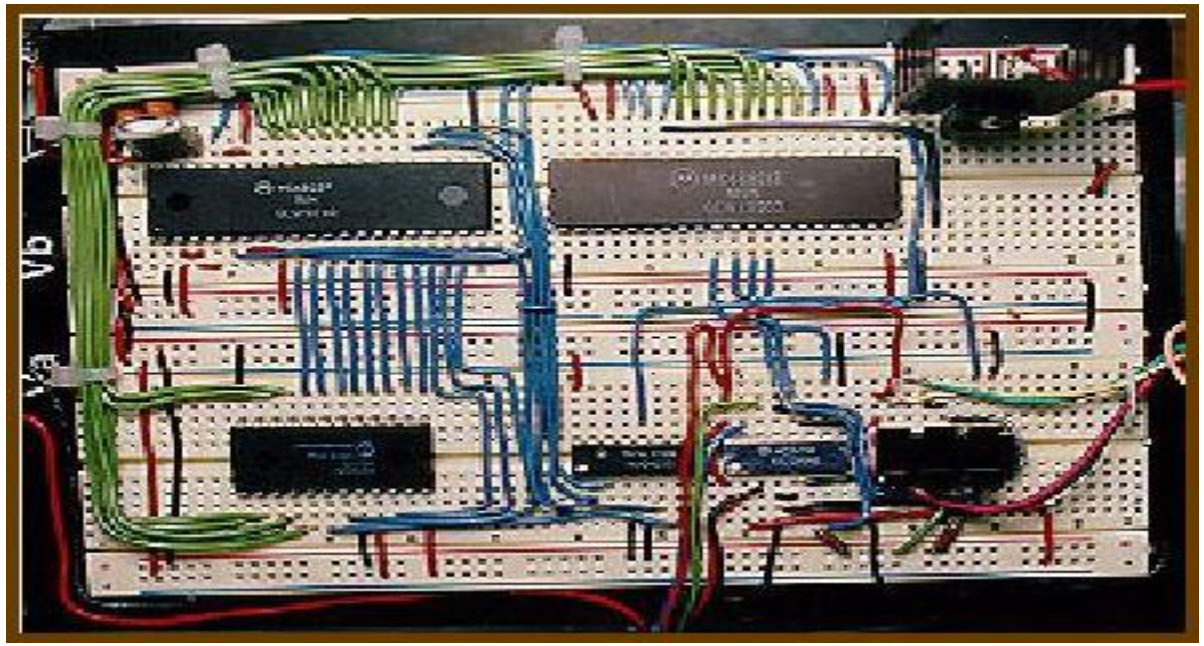
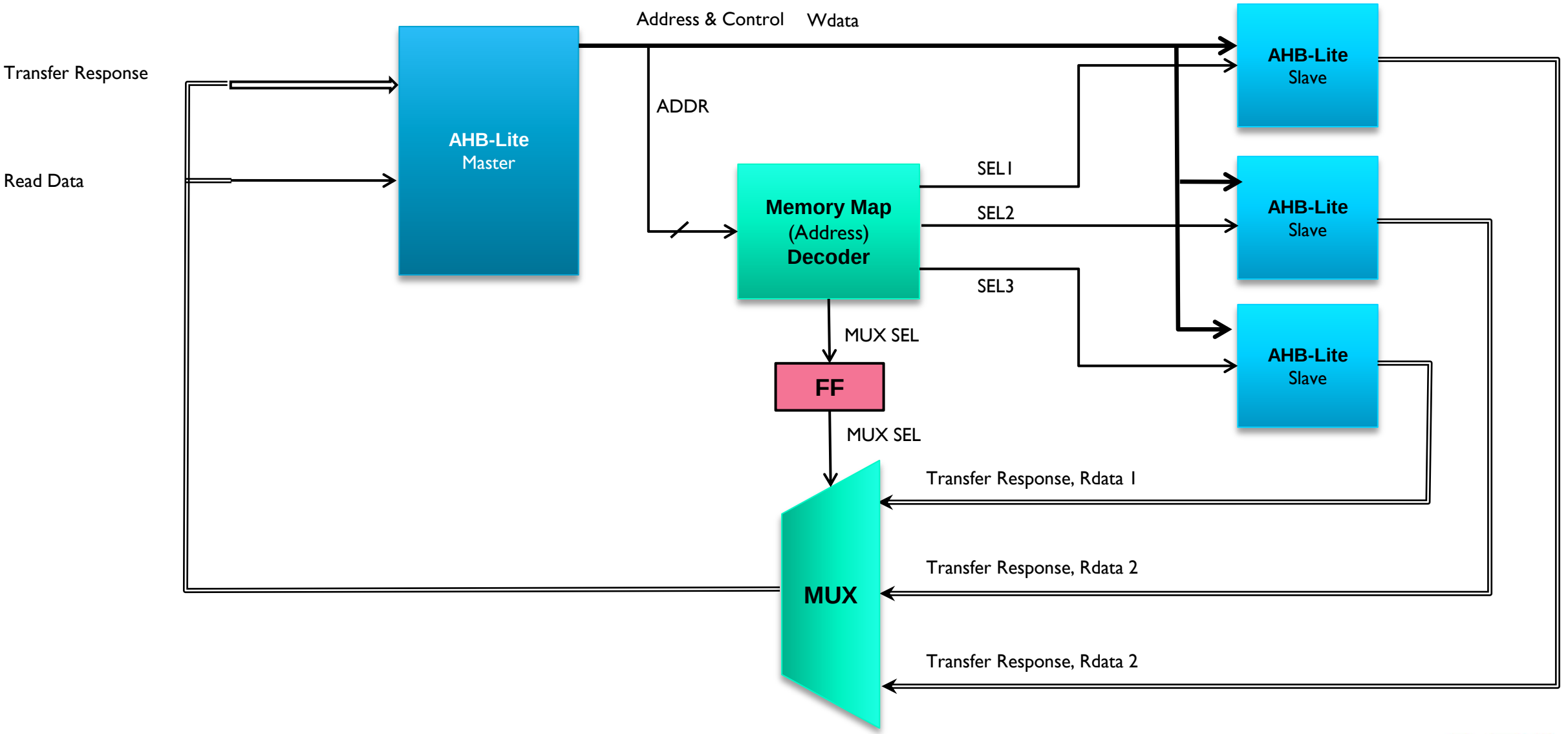
Integrating AHB-Lite UART and Memory Controller



AHB-Lite System (Recap)



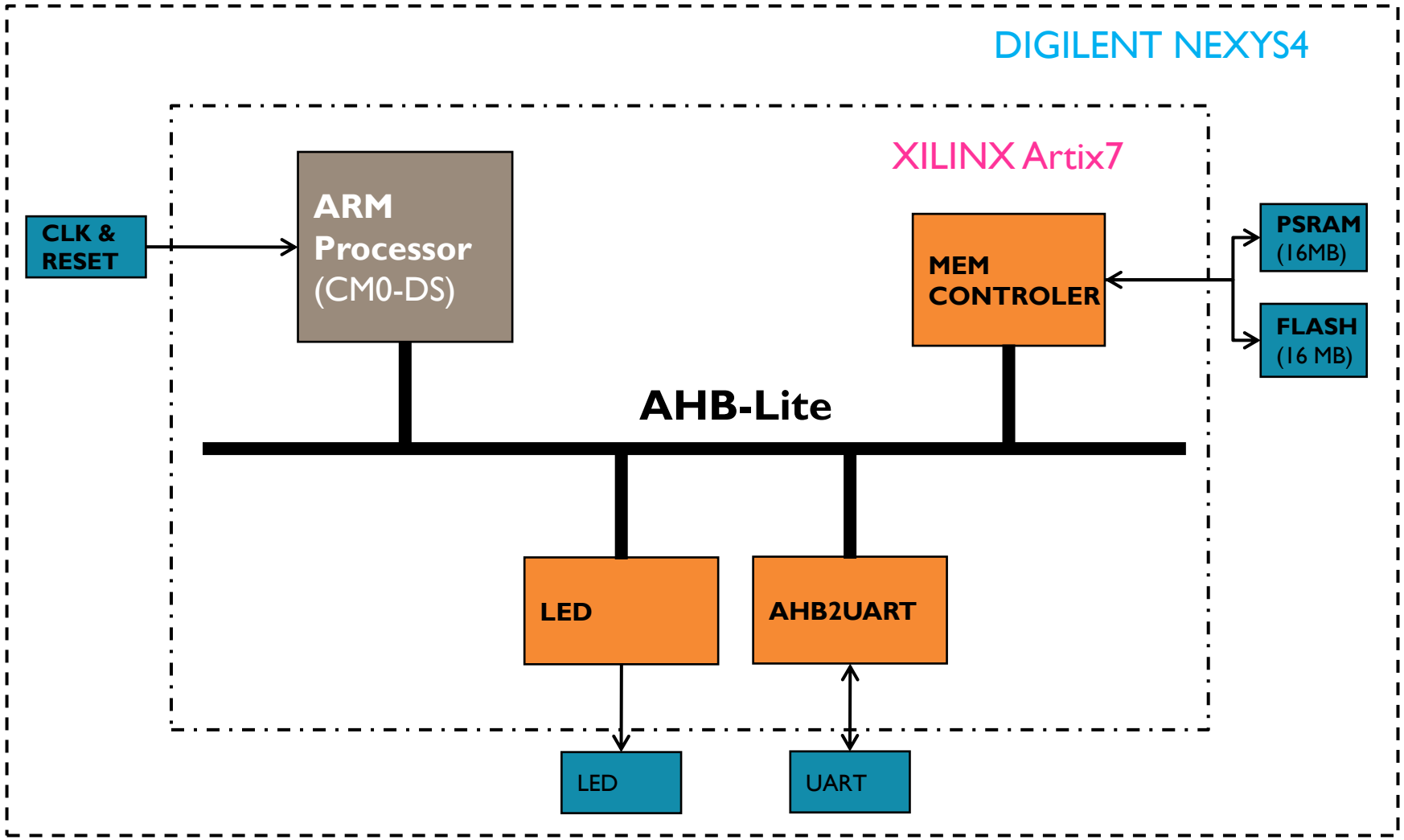
ARM

Design Summary

- Replace AHB2MEM peripheral with SRAM/FLASH memory controller
- Add UART peripheral for basic IO
- The design consists of 3 peripherals with below memory map

	Base Address	Size
PSRAM	0x0000_0000	8 MB
LED	0x5000_0000	0x00 General Purpose IO
UART	0x5100_0000	0x00 Send/Receive 0x04 Control Register

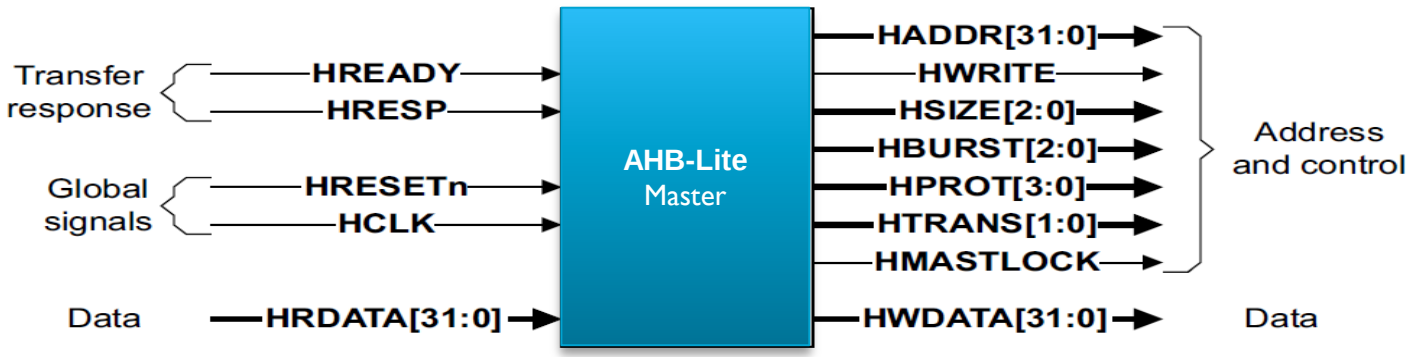
Design Summary



ARM

AHB-Lite Master

- **CM0-DS in our case**

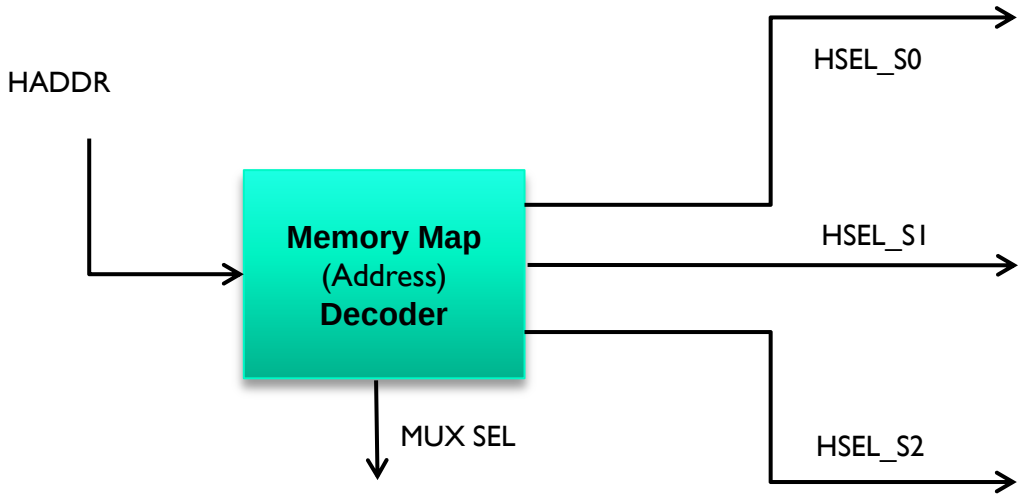



```
//AHBLite MASTER --> CM0-DS

CORTEXM0DS u_cortexm0ds (
   //Global Signals
   .HCLK          (HCLK),
   .HRESETn       (HRESETn),
   //Address, Control & Write Data
   .HADDR         (HADDR[31:0]),
   .HBURST        (HBURST[2:0]),
   .HMASTLOCK     (HMASTLOCK),
   .HPROT         (HPROT[3:0]),
   .HSIZE         (HSIZE[2:0]),
   .HTRANS        (HTRANS[1:0]),
   .HWDATA        (HWDATA[31:0]),
   .HWRITE        (HWRITE),
   //Transfer Response & Read Data
   .HRDATA        (HRDATA[31:0]),
   .HREADY        (HREADY),
   .HRESP         (HRESP),

   //CM0 Sideband Signals
   .NMI           (1'b0),
   .IRQ           (IRQ[15:0]),
   .TXEV          (),
   .RXEV          (1'b0),
   .LOCKUP        (LOCKUP),
   .SYSRESETREQ   (),
   .SLEEPING      ()
);
```


ARM

Address Decoder

- **Combination Block**
- **Decodes the address of each transfer and provides a select signal for slave**
- **Provides control signal to Multiplexer**

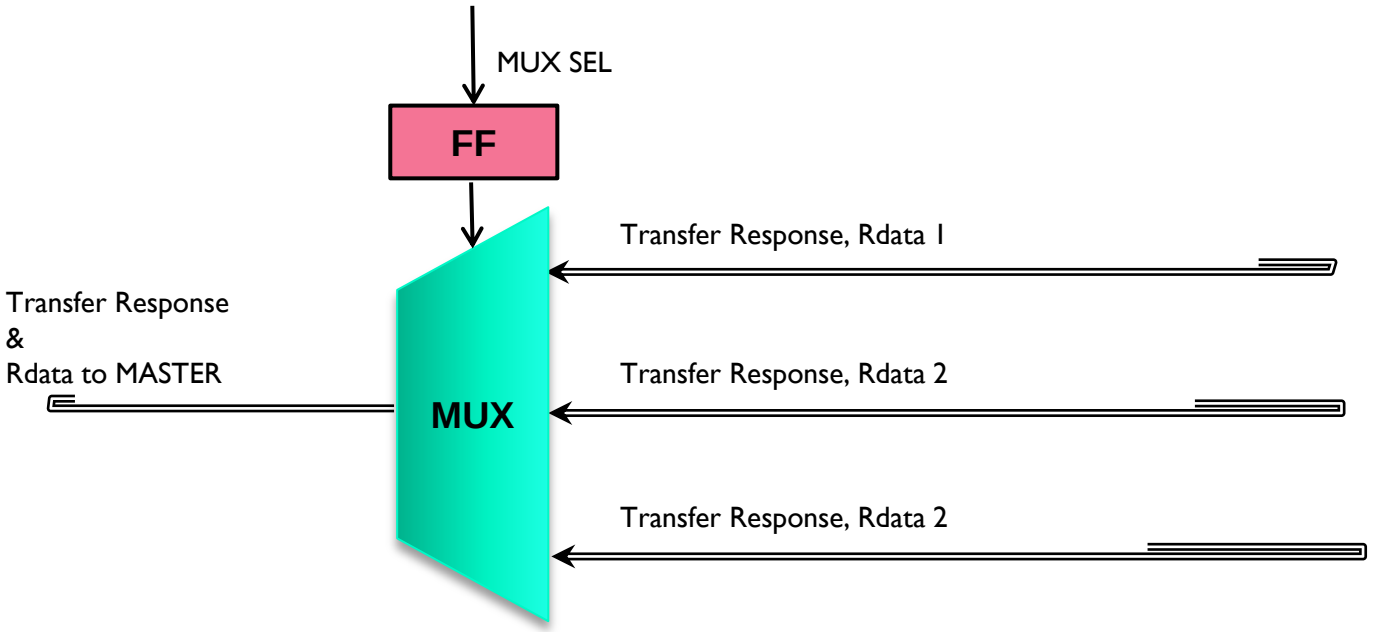



```
168 //Address Decoder
169
170 AHBDCD uAHBDCD (
171     .HADDR(HADDR[31:0]),
172
173     .HSEL_S0(HSEL_MEM),
174     .HSEL_S1(HSEL_LED),
175     .HSEL_S2(HSEL_UART),
176     .HSEL_S3(),
177     .HSEL_S4(),
178     .HSEL_S5(),
179     .HSEL_S6(),
180     .HSEL_S7(),
181     .HSEL_S8(),
182     .HSEL_S9(),
183     .HSEL_NOMAP(HSEL_NOMAP),
184
185     .MUX_SEL(MUX_SEL[3:0])
186 );
187
```


ARM

Slave-to-Master Multiplexer

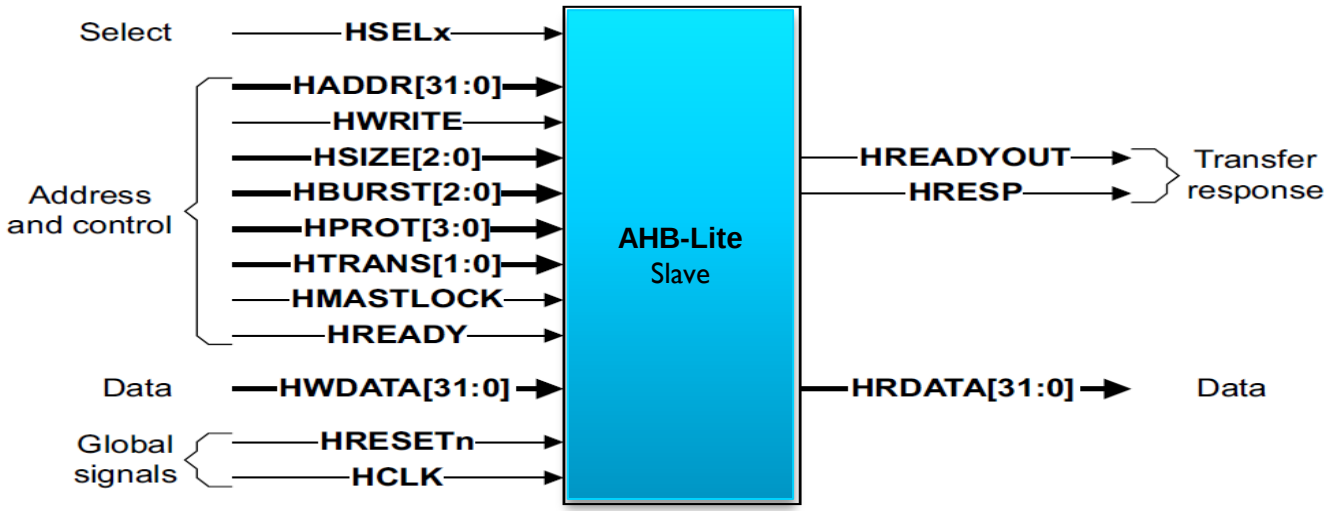
- **Multiplex the read data bus and response signals**
- **Decoder provides control (MUX_SEL)**
- **Remember Pipelined Operation**




```
190 AHBMUX uAHBMUX (
191     .HCLK(HCLK),
192     .HRESETn(HRESETn),
193     .MUX_SEL(MUX_SEL[3:0]),
194
195     .HRDATA_S0(HRDATA_MEM),
196     .HRDATA_S1(HRDATA_LED),
197     .HRDATA_S2(HRDATA_UART),
198     .HRDATA_S3(),
199     .HRDATA_S4(),
200     .HRDATA_S5(),
201     .HRDATA_S6(),
202     .HRDATA_S7(),
203     .HRDATA_S8(),
204     .HRDATA_S9(),
205     .HRDATA_NOMAP(32'hDEADBEEF),
206
207     .HREADYOUT_S0(HREADYOUT_MEM),
208     .HREADYOUT_S1(HREADYOUT_LED),
209     .HREADYOUT_S2(HREADYOUT_UART),
210     .HREADYOUT_S3(1'b1),
211     .HREADYOUT_S4(1'b1),
212     .HREADYOUT_S5(1'b1),
213     .HREADYOUT_S6(1'b1),
214     .HREADYOUT_S7(1'b1),
215     .HREADYOUT_S8(1'b1),
216     .HREADYOUT_S9(1'b1),
217     .HREADYOUT_NOMAP(1'b1),
218
219     .HRDATA(HRDATA[31:0]),
220     .HREADY(HREADY)
221 );
```


ARM

AHB Slaves




```
// AHBLite Memory Controller
AHB2SRAMFLSH uAHB2SRAMFLSH (
    .HCLK(HCLK),
    .HRESETn(HRESETn),
    .HADDR(HADDR[31:0]),
    .HSEL(HSEL_MEM),
    .HREADY(HREADY),
    .HSIZE(HSIZE[2:0]),
    .HTRANS(HTRANS[1:0]),
    .HWDATA(HWDATA[31:0]),
    .HWRITE(HWRITE),
    .HRDATA(HRDATA_MEM[31:0]),
    .HREADYOUT(HREADYOUT_MEM),

    .MemDB(MemDB),
    .MemAdr(MemAdr_SRAMCTRL),
    .RamCS(RamCEn),
    .MemWR(RamWEn),
    .MemOE(RamOEn),
    .RamUB(RamUBn),
    .RamLB(RamLBn),
    .RamCre(RamCRE),
    .RamAdv(RamADVn),
    .RamClk(RamCLK),
    .RamWait(RamWait)
);
```

```
//AHBLite Slave
AHB2LED uAHB2LED (
    //AHBLITE Signals
    .HSEL(HSEL_LED),
    .HCLK(HCLK),
    .HRESETn(HRESETn),
    .HREADY(HREADY),
    .HADDR(HADDR),
    .HTRANS(HTRANS[1:0]),
    .HWRITE(HWRITE),
    .HSIZE(HSIZE),
    .HWDATA(HWDATA[31:0]),

    .HRDATA(HRDATA_LED),
    .HREADYOUT(HREADYOUT_LED),
    //Sideband Signals
    .LED(LED[6:0])
);
```

```
AHBUART uAHBUART(
    .HCLK(HCLK),
    .HRESETn(HRESETn),
    .HADDR(HADDR[31:0]),
    .HTRANS(HTRANS[1:0]),
    .HWDATA(HWDATA[31:0]),
    .HWRITE(HWRITE),
    .HREADY(HREADY),
    .HREADYOUT(HREADYOUT_UART),
    .HRDATA(HRDATA_UART[31:0]),
    .HSEL(HSEL_UART),

    .RsRx(RsRx),
    .RsTx(RsTx)
//  .uart_irq(UART_IRQ)
    );
```


ARM

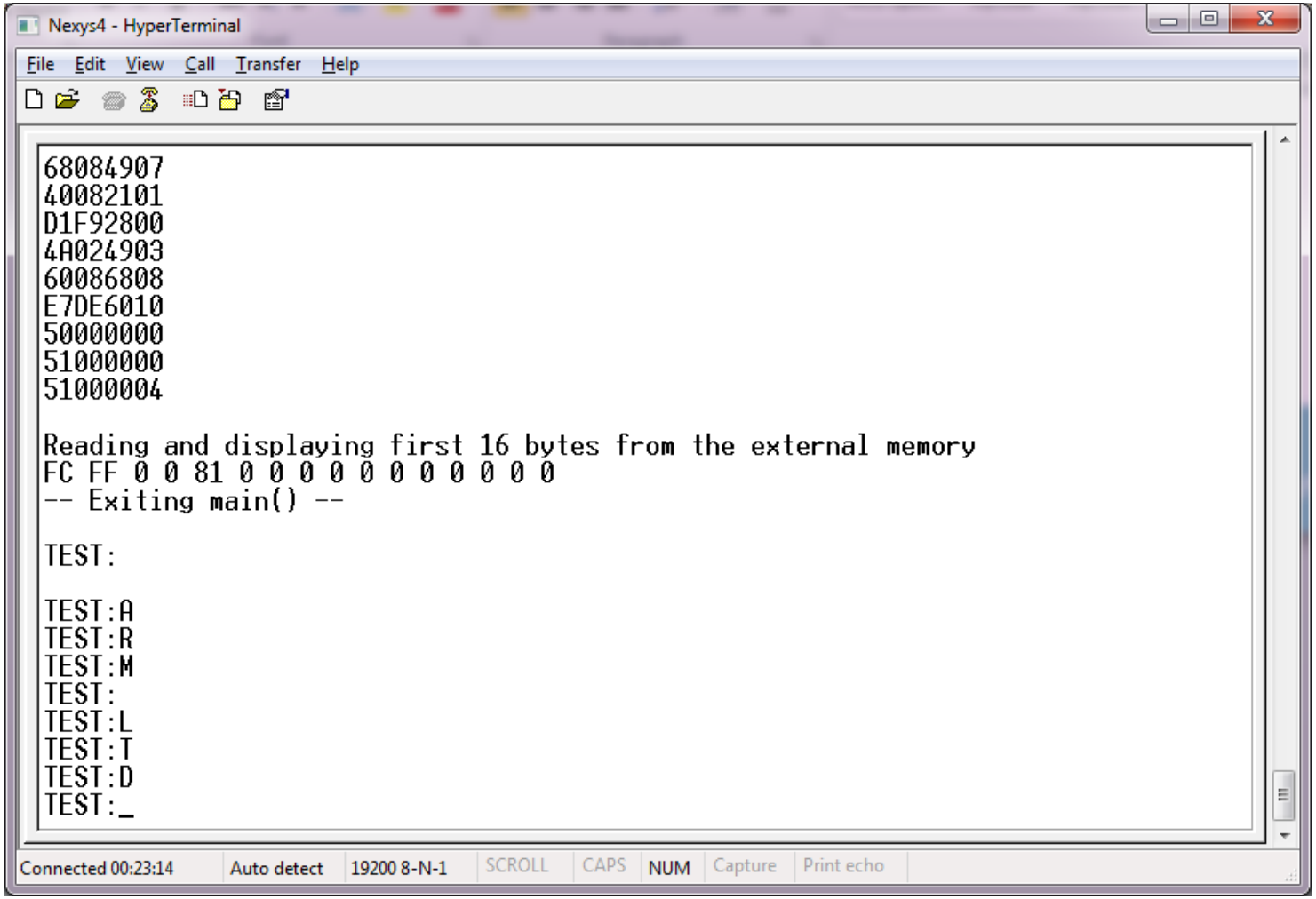
Lab Steps (with PSRAM)

1. Compile the Software using KEIL MDK ARM and generate code.hex file
2. Follow the steps given in the lab manual to download code.hex onto PSRAM
3. Open FPGA project under Vivado and implement the design
4. Use Vivado hardware manager to download the .bit file
5. Communicate with the board using HyperTerminal (or any other serial terminal)

Lab Steps (with BRAM)

1. Compile the Software using KEIL MDK ARM and generate code.hex file
2. Open FPGA project under Vivado and implement the design
3. Use Vivado hardware manager to download the .bit file
4. Communicate with the board using HyperTerminal (or any other serial terminal)

Output


Nexys4 - HyperTerminal
File Edit View Call Transfer Help
68084907
40082101
D1F92800
4A024903
60086808
E7DE6010
50000000
51000000
51000004
Reading and displaying first 16 bytes from the external memory
FC FF 0 0 81 0 0 0 0 0 0 0 0 0 0 0
-- Exiting main() --
TEST:
TEST:A
TEST:R
TEST:M
TEST:
TEST:L
TEST:T
TEST:D
TEST:_
Connected 00:23:14
Auto detect
19200 8-N-1
SCROLL
CAPS
NUM
Capture
Print echo

ARM