ASIC Logic

Speaker: Hao-Yun Chin

Advisor: Prof. **Tian-Sheuan Chang**

Apr. 13, 2004

Goal of This Lab

- ❑ Rapid Prototyping
- ❑ Lint Checking
- ❑ Coverage Verification
- ❑ Familiarize with ARM Logic Module (LM)
- ❑ Know how to program LM
- ❑ HW/SW co-verification using LM and CM

Outline

- ❑ *Introduction*
- ❑ ARM System Overview
- ❑ ARM Integrator System Memory Map
- ❑ Prototyping with Logic Module
- ❑ Lab – ASIC Logic

Introduction

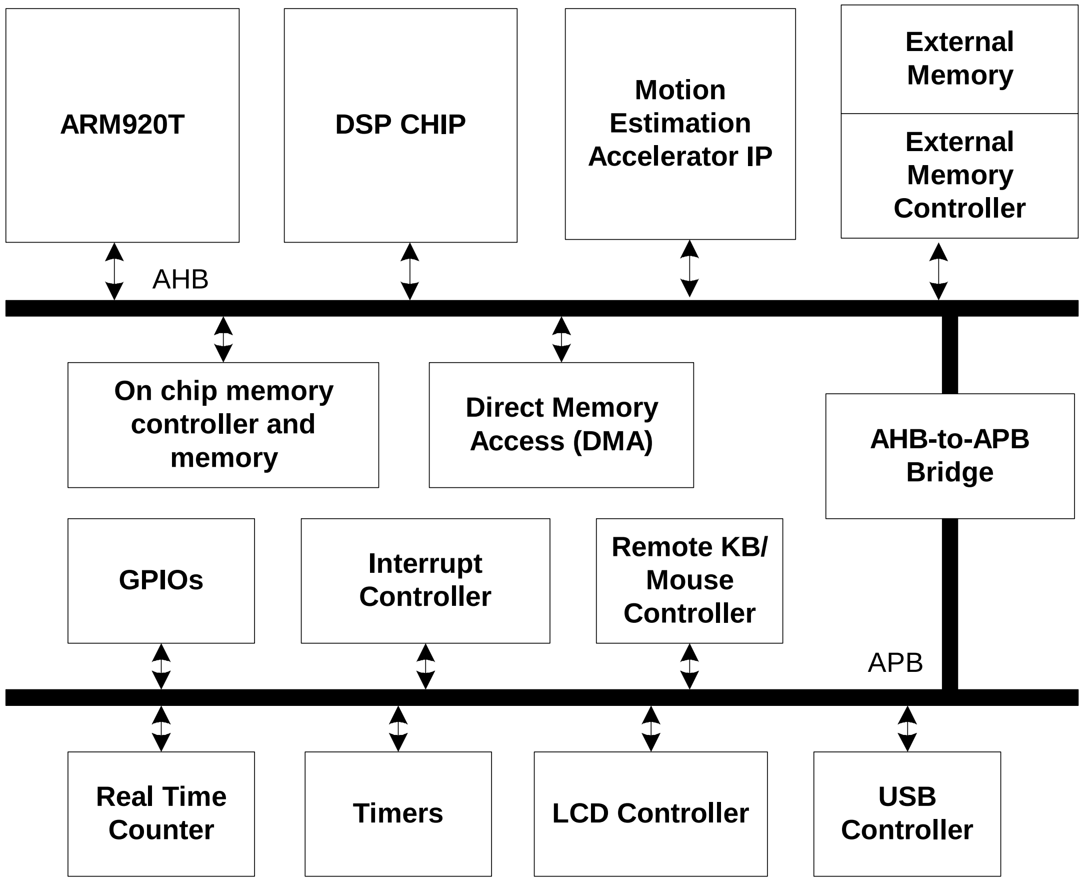
- ❑ Rapid Prototyping – A fast way to verify your prototype design.
  - – Enables you to discover problems before tape out.
  - – Helps to provide a better understanding of the design’s behavior.
- ❑ ARM Integrator and Logic Module can be used for Hardware Design Verification and HW/SW co-verification.
  - – Hardware Design Verification: using LM stand alone.
  - – HW/SW co-verification: using LM, CM, Integrator together.

Outline

- ❑ Introduction
- ❑ ***ARM System Overview***
  - – ARM Synchronization Scheme: Interrupt
  - – ARM Synchronization Scheme: Polling
- ❑ ARM Integrator System Memory Map
- ❑ Prototyping with Logic Module
- ❑ Lab – ASIC Logic

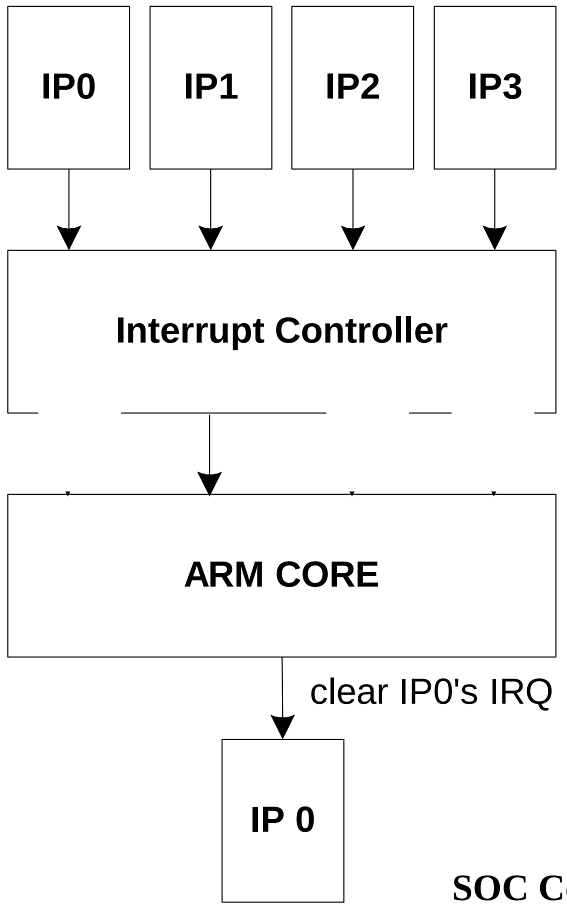
ARM System Overview

- A typical ARM system consists of an ARM core, a DSP chip for application-specific needs, some dedicated hardware accelerator IPs, storages, and some peripherals and controls.


ARM920T
DSP CHIP
Motion Estimation Accelerator IP
External Memory
External Memory Controller
AHB
On chip memory controller and memory
Direct Memory Access (DMA)
AHB-to-APB Bridge
GPIOs
Interrupt Controller
Remote KB/ Mouse Controller
APB
Real Time Counter
Timers
LCD Controller
USB Controller

ARM System Synchronization Scheme: Interrupt

- A device asserts an interrupt signal to request the ARM core handle it.
- The ARM core can perform tasks while the device is in use.
- Needs Interrupt Controller. More hardware.



IP0, IP1, IP2, and IP3 raised interrupt request (IRQ) at the same time. The IRQs are sent to the interrupt controller.

Interrupt controller receives the IRQs and update the IRQ status indicating the IRQ sources.

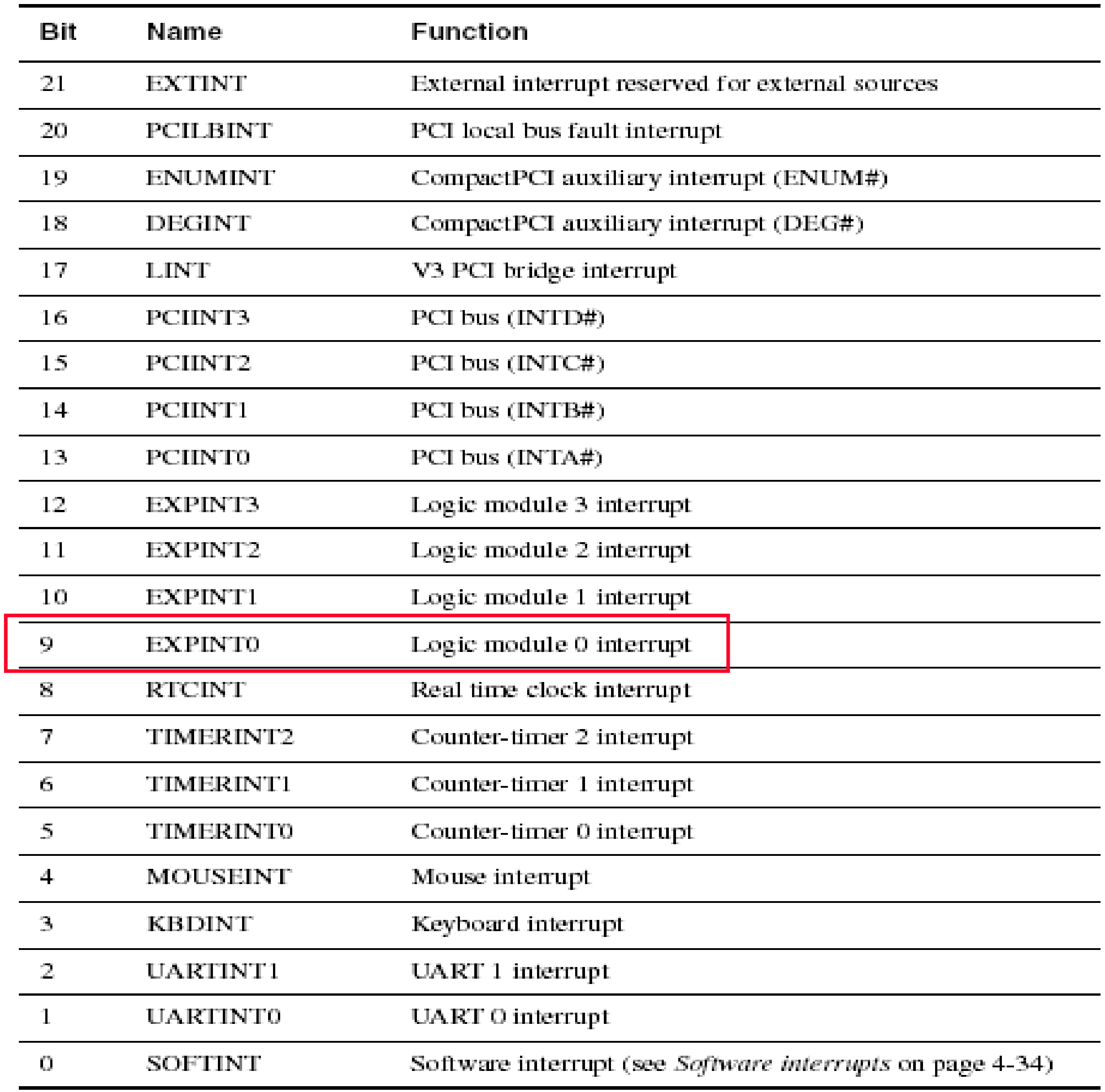
ARM core receives the IRQs, deteremines which IRQ should be handled according to programmed priorities. and then executes the corresponding interrupt service routine (ISR).

The ISR performs its operations and clears the IP0's interrupt.

IRQ registers

Address	Name	Type	Size	Function
0x14000000	IRQ0_STATUS	Read	22	IRQ0 status
0x14000004	IRQ0_RAWSTAT	Read	22	IRQ0 interrupt request status
0x14000008	IRQ0_ENABLESET	Read/write	22	IRQ0 enable set
0x1400000C	IRQ0_ENABLECLR	Write	22	IRQ0 enable clear
0x14000040	IRQ1_STATUS	Read	22	IRQ1 status register
0x14000044	IRQ1_RAWSTAT	Read	22	IRQ1 raw status
0x14000048	IRQ1_ENABLESET	Read/write	22	IRQ1 enable set
0x1400004C	IRQ1_ENABLECLR	Write	22	IRQ1 enable clear
0x14000080	IRQ2_STATUS	Read	22	IRQ2 status register
0x14000084	IRQ2_RAWSTAT	Read	22	IRQ2 raw status
0x14000088	IRQ2_ENABLESET	Read/write	22	IRQ2 enable set
0x1400008C	IRQ2_ENABLECLR	Write	22	IRQ2 enable clear
0x140000C0	IRQ3_STATUS	Read	22	IRQ3 status register
0x140000C4	IRQ3_RAWSTAT	Read	22	IRQ3 raw status
0x140000C8	IRQ3_ENABLESET	Read/write	22	IRQ3 enable set
0x140000CC	IRQ3_ENABLECLR	Write	22	IRQ3 enable clear

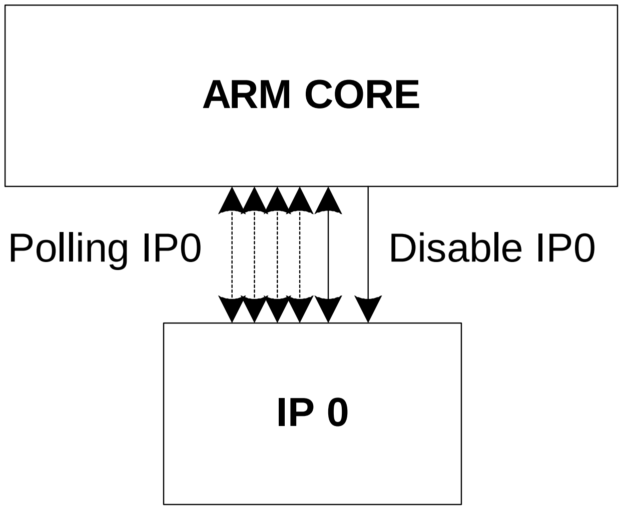
IRQ register Bit assignment



Bit	Name	Function
21	EXTINT	External interrupt reserved for external sources
20	PCILBINT	PCI local bus fault interrupt
19	ENUMINT	CompactPCI auxiliary interrupt (ENUM#)
18	DEGINT	CompactPCI auxiliary interrupt (DEG#)
17	LINT	V3 PCI bridge interrupt
16	PCIINT3	PCI bus (INTD#)
15	PCIINT2	PCI bus (INTC#)
14	PCIINT1	PCI bus (INTB#)
13	PCIINT0	PCI bus (INTA#)
12	EXPINT3	Logic module 3 interrupt
11	EXPINT2	Logic module 2 interrupt
10	EXPINT1	Logic module 1 interrupt
9	EXPINT0	Logic module 0 interrupt
8	RTCINT	Real time clock interrupt
7	TIMERINT2	Counter-timer 2 interrupt
6	TIMERINT1	Counter-timer 1 interrupt
5	TIMERINT0	Counter-timer 0 interrupt
4	MOUSEINT	Mouse interrupt
3	KBDINT	Keyboard interrupt
2	UARTINT1	UART 1 interrupt
1	UARTINT0	UART 0 interrupt
0	SOFTINT	Software interrupt (see *Software interrupts* on page 4-34)

ARM System Synchronization Scheme: Polling

- The ARM core keeps checking a register indicating if the device has done its task.
- The ARM core is busy "polling" the device while the device is in use.
- Less hardware.



ARM core polls IP0's ready register after IP0 has been enabled.

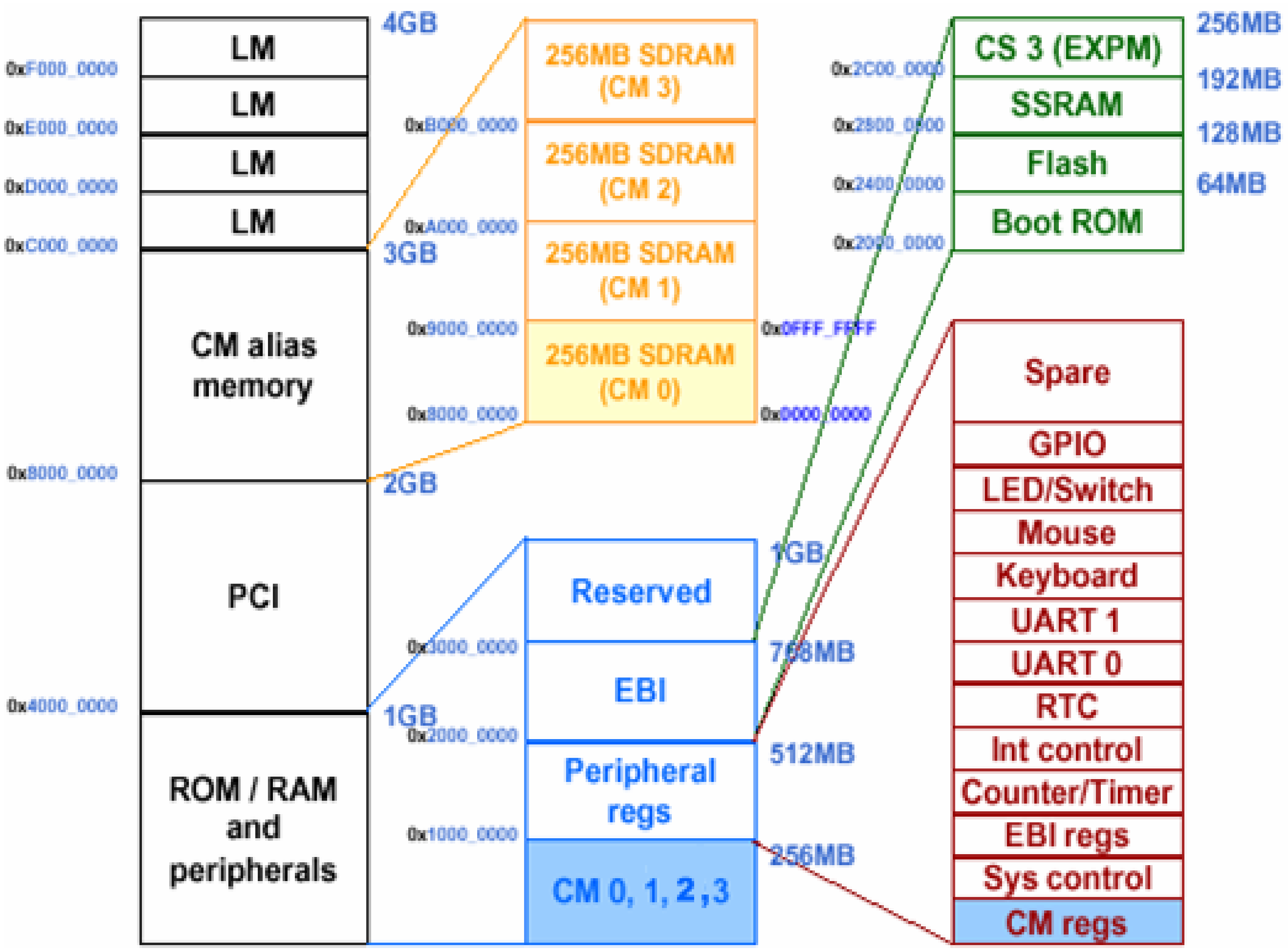
Once IP0 is done with its operation, ARM core will know from the changed value of the ready register.

ARM core will execute the corresponding operations and then disable IP0.

Outline

- ❑ Introduction
- ❑ ARM System Overview
- ❑ ***ARM Integrator System Memory Map***
- ❑ Prototyping with Logic Module
- ❑ Lab – ASIC Logic

Overview of System Memory Map



Core Module Memory Map

- Core Module Control Register CM_CTRL

Bits	Name	Access	Function
31:6	Reserved		
5	BIGEND	R/W	0=little-endian 1=big-endian
4	Reserved		
3	RESET	W	Reset core module
2	REMAP	R/W	0=access Boot ROM 1=access SSRAM
1	nMBDET	R	0=mounted on MB 1=stand alone
0	LED	R/W	0=LED OFF 1=LED ON

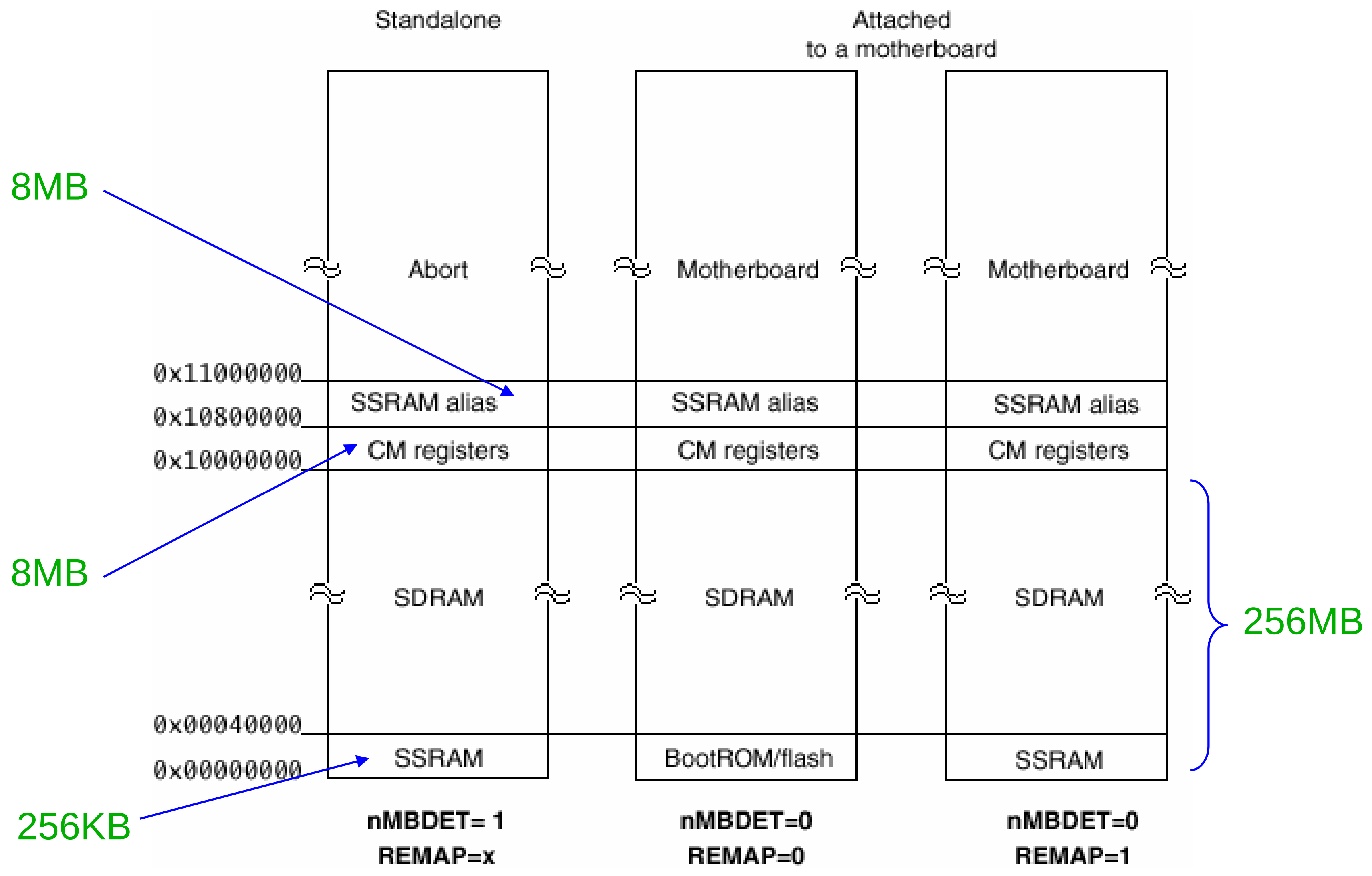
- 4-pole DIP switch (S1) on motherboard
  - S1[1]=ON: code starts execution from boot ROM
  - S1[1]=OFF: code starts execution from flash

Core Module Memory Map (cont.)

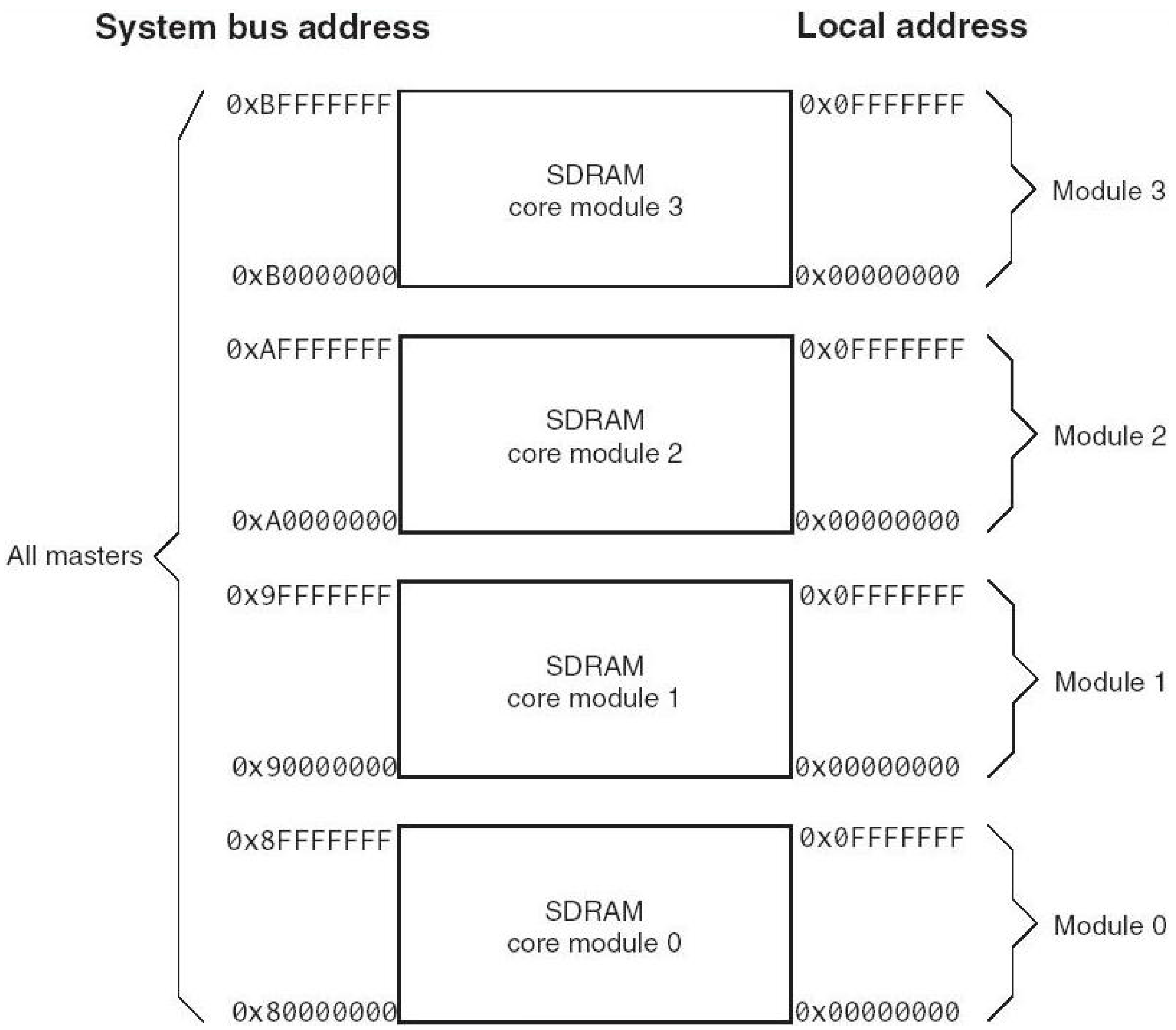
nMBDET	REMAP	Address range	Region size	Description
0	0	0x00000000 to 0x0003FFFF	256KB	Boot ROM (on motherboard)
0	1	0x00000000 to 0x0003FFFF	256KB	SSRAM
1	X	0x00000000 to 0x0003FFFF	256KB	SSRAM
X	X	0x00040000 to 0x0FFFFFFF	256MB	Local SDRAM
X	X	0x10000000 to 0x107FFFFF	8MB	Core Module registers
X	X	0x10800000 to 0x10FFFFFF	8MB	SSRAM alias
0	X	0x11000000 to 0xFFFFFFFF	272MB to 4GB	System bus address space
1	X	0x11000000 to 0xFFFFFFFF	272MB to 4GB	Abort

- The **nMBDET** signal is permanently grounded by the motherboard so that it is pulled LOW on the core module when it is fitted.
- The **REMAP** bit only has effect if the core module is attached to a motherboard (**nMBDET** = 0).

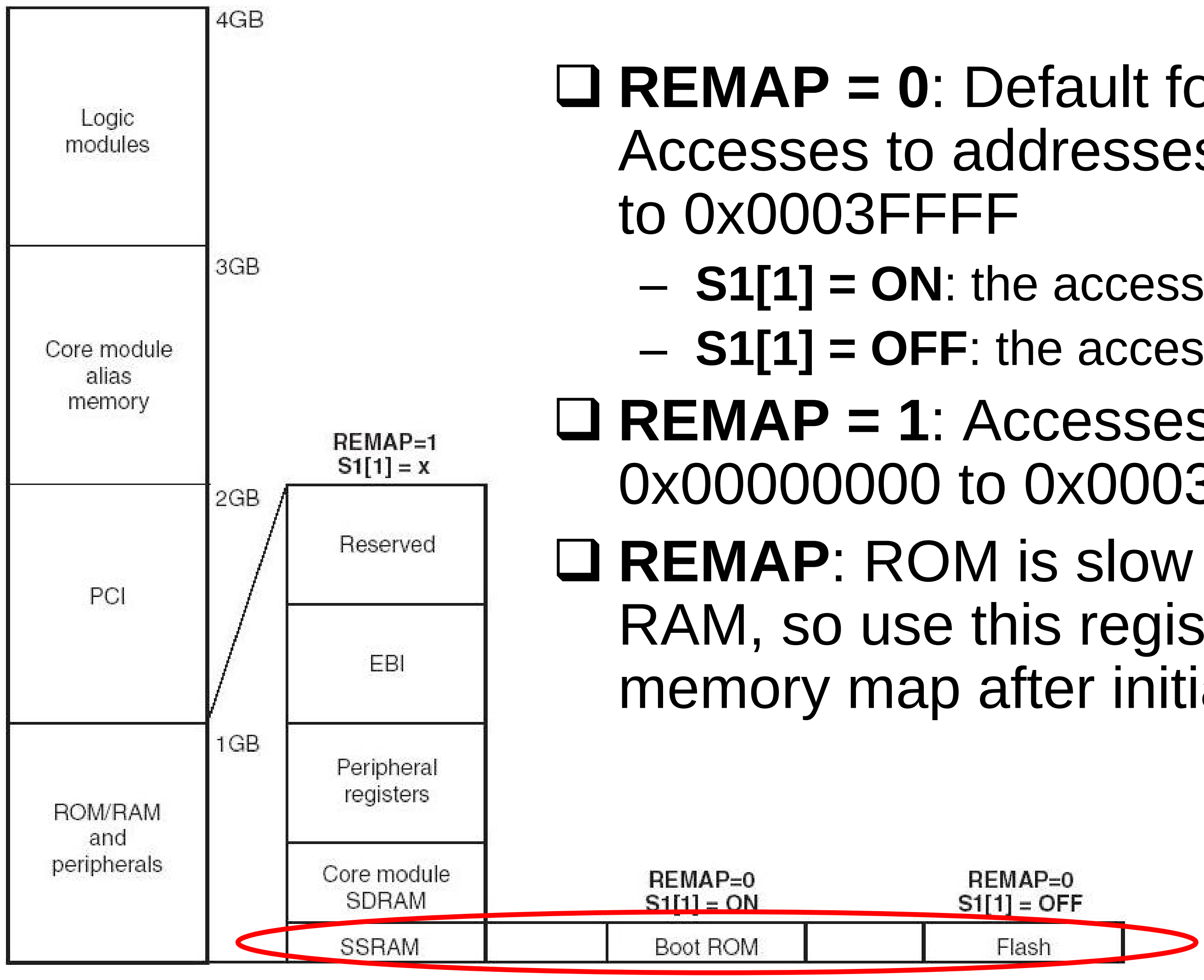
Core Module Memory Map (cont.)



Core Module Alias Address

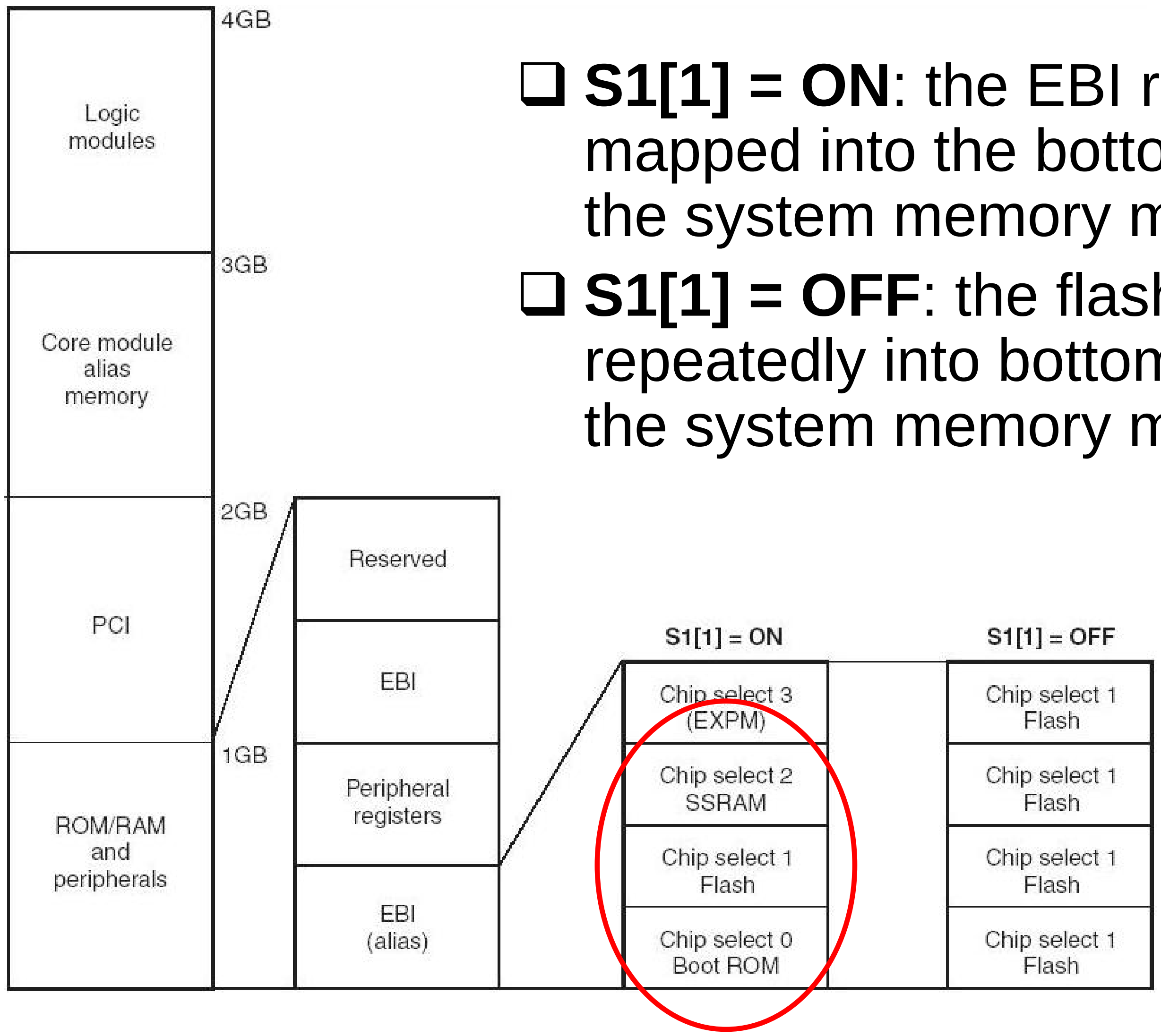


Integrator Memory Map for Core Modules



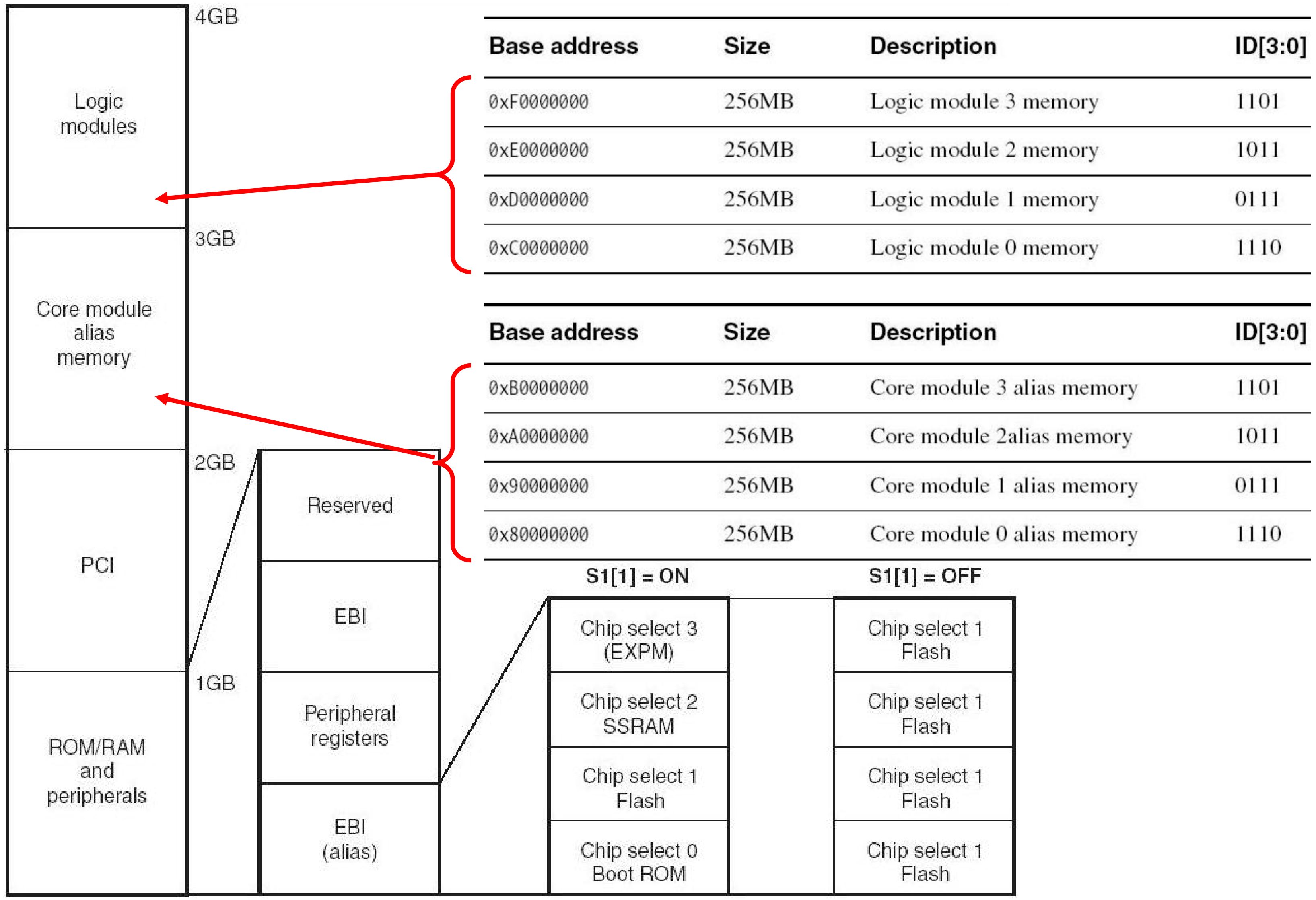
- **REMAP = 0**: Default following reset. Accesses to addresses 0x00000000 to 0x0003FFFF
  - **S1[1] = ON**: the access is to boot ROM
  - **S1[1] = OFF**: the access is to flash
- **REMAP = 1**: Accesses to address 0x00000000 to 0x0003FFFF
- **REMAP**: ROM is slow & narrow to RAM, so use this register to change memory map after initialization

Integrator Memory Map for Logic Modules



- **S1[1] = ON**: the EBI resources are mapped into the bottom 256MB of the system memory map.
- **S1[1] = OFF**: the flash is mapped repeatedly into bottom 256MB of the system memory map.

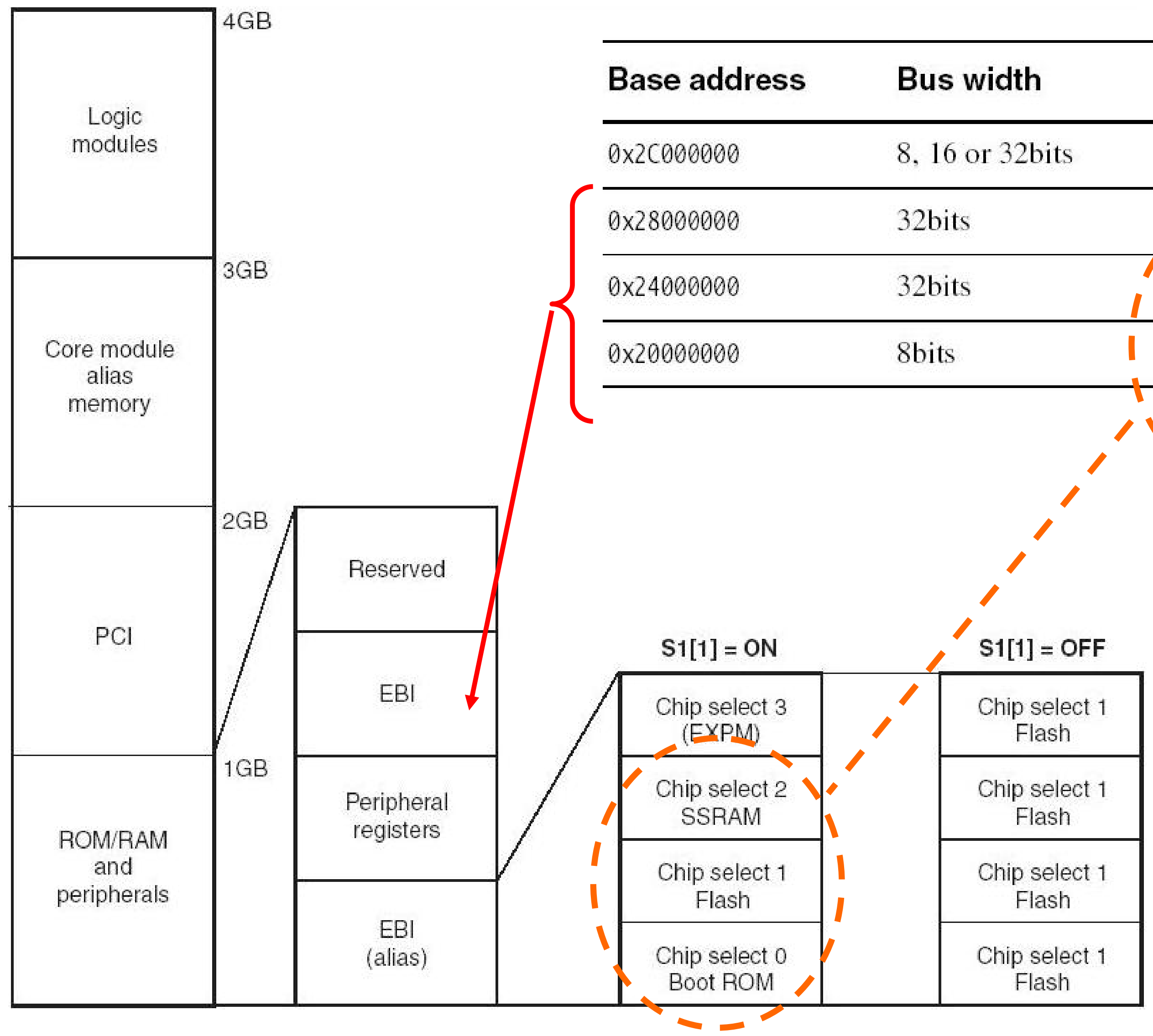
System Memory Map (1/3)



Base address	Size	Description	ID[3:0]
0xF0000000	256MB	Logic module 3 memory	1101
0xE0000000	256MB	Logic module 2 memory	1011
0xD0000000	256MB	Logic module 1 memory	0111
0xC0000000	256MB	Logic module 0 memory	1110

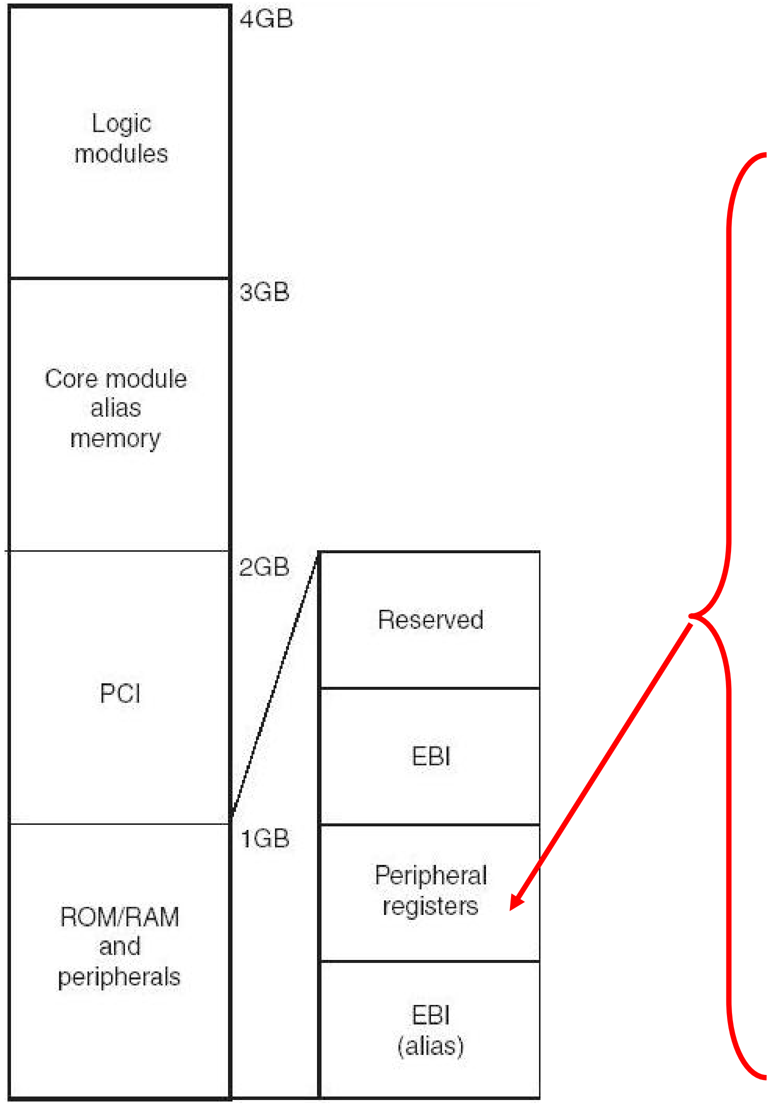
Base address	Size	Description	ID[3:0]
0xB0000000	256MB	Core module 3 alias memory	1101
0xA0000000	256MB	Core module 2alias memory	1011
0x90000000	256MB	Core module 1 alias memory	0111
0x80000000	256MB	Core module 0 alias memory	1110

System Memory Map (2/3)



Base address	Bus width	Description
0x2C000000	8, 16 or 32bits	Chip select 3 (for expansion)
0x28000000	32bits	SSRAM
0x24000000	32bits	Flash
0x20000000	8bits	Boot ROM

System Memory Map (3/3)

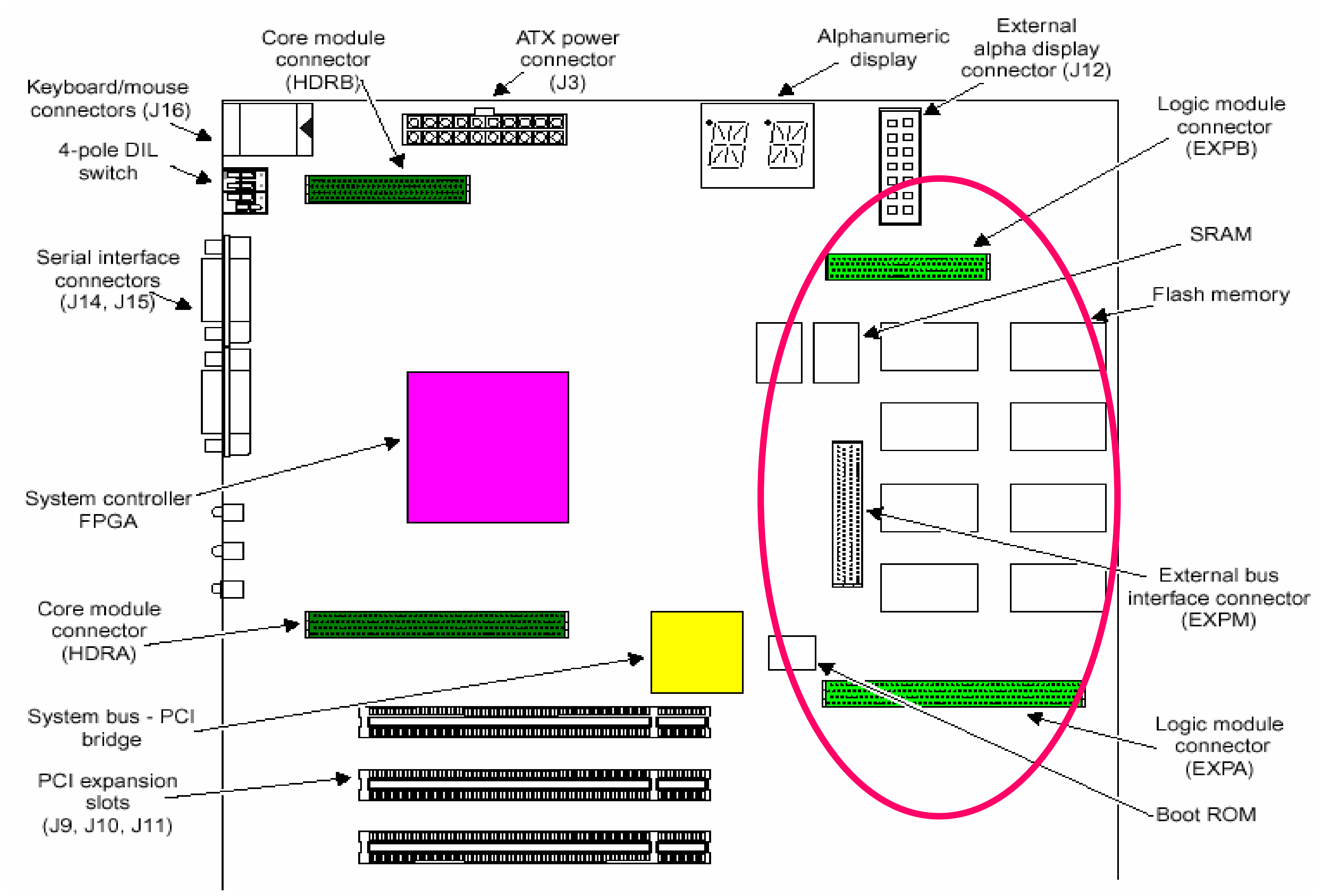


Base address	Size	Description
0x1F000000	16MB	Spare
0x1E000000	16MB	Spare
0x1D000000	16MB	Spare
0x1C000000	16MB	Spare
0x1B000000	16MB	GPIO
0x1A000000	16MB	LED display and boot switch
0x19000000	16MB	Mouse
0x18000000	16MB	Keyboard
0x17000000	16MB	UART1
0x16000000	16MB	UART0
0x15000000	16MB	RTC
0x14000000	16MB	Interrupt controller
0x13000000	16MB	Counter/timers
0x12000000	16MB	EBI configuration registers
0x11000000	16MB	System controller registers
0x10000000	16MB	Core module registers (for core modules) Spare (for logic modules)

Outline

- ❑ Introduction
- ❑ ARM System Overview
- ❑ ARM Integrator System Memory Map
- ❑ ***Prototyping with Logic Module***
  - – ARM Integrator AP & ARM LM
  - – FPGA tools
  - – Example 0
  - – Example 1
  - – Example 2
  - – Exercise
- ❑ Lab – ASIC Logic

AP Layout


Core module connector (HDRB)
ATX power connector (J3)
Alphanumeric display
External alpha display connector (J12)
Keyboard/mouse connectors (J16)
4-pole DIL switch
Logic module connector (EXPB)
SRAM
Flash memory
Serial interface connectors (J14, J15)
System controller FPGA
External bus interface connector (EXPM)
Core module connector (HDRA)
System bus - PCI bridge
Logic module connector (EXPA)
PCI expansion slots (J9, J10, J11)
Boot ROM

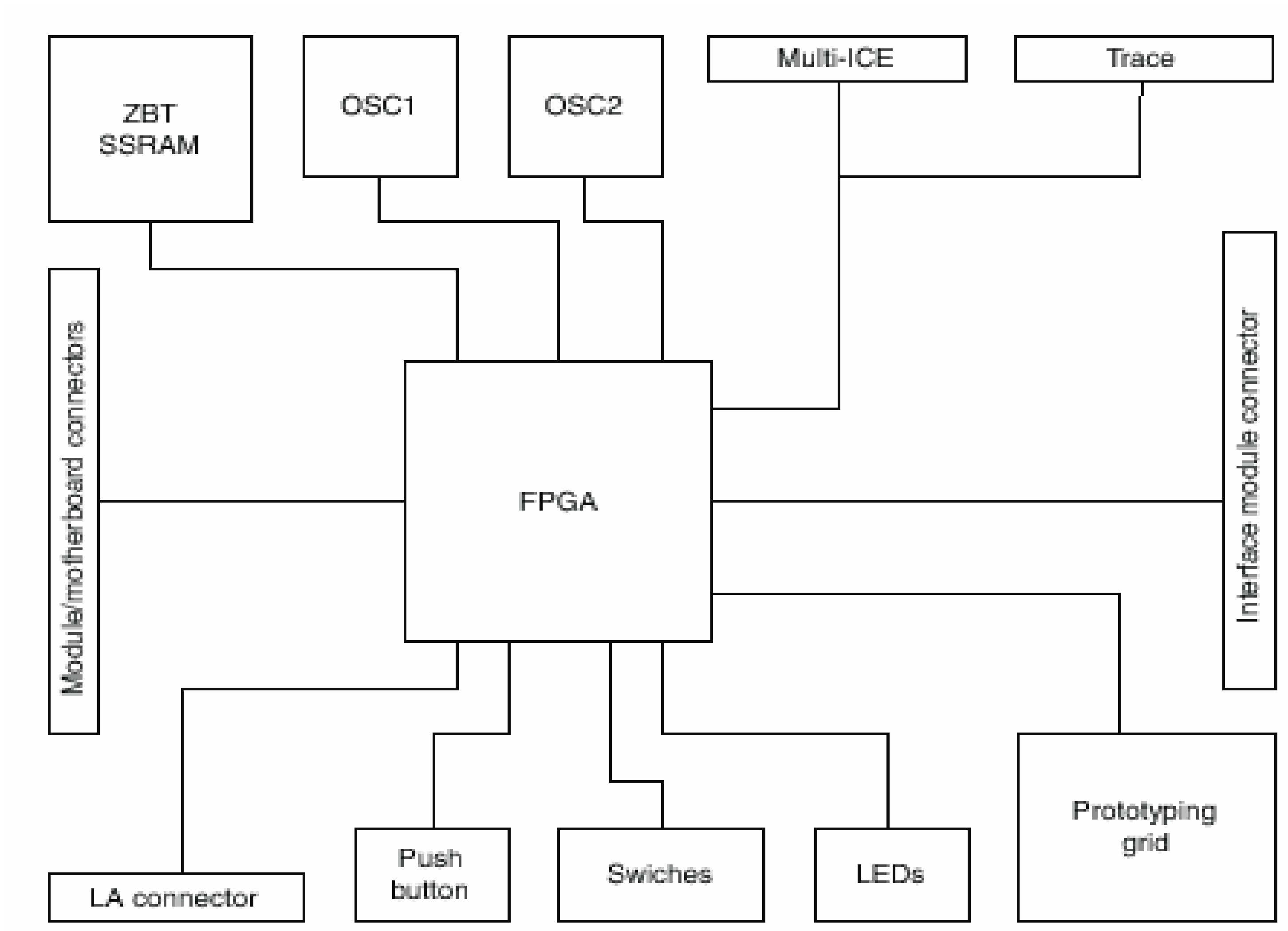
What is LM

- *Logic Module*
- A platform for developing ***Advanced Microcontroller Bus Architecture*** (AMBA), ***Advanced System Bus*** (ASB), ***Advanced High-performance Bus*** (AHB), and ***Advanced Peripheral Bus*** (APB) peripherals for use with ARM cores.

Using the LM

- It can be used in the following ways:
  - As a standalone system
  - With an CM, and a AP or SP motherboard
  - As a CM with either AP or SP motherboard if a synthesized ARM core is programmed into the FPGA
  - Stacked without a motherboard, if one module in the stack provides system controller functions of a motherboard

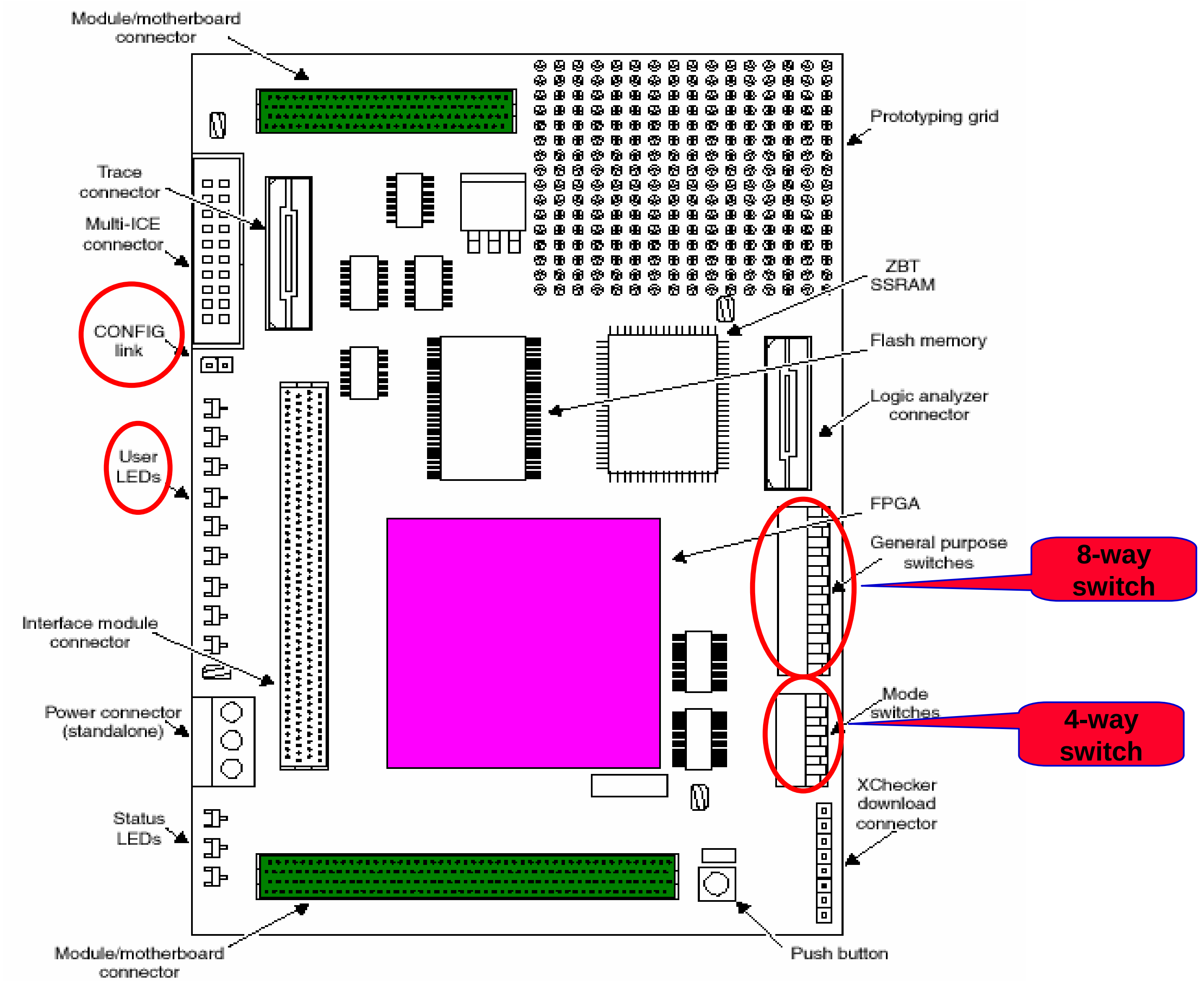
LM Architecture



Components of LM

- Altera or ***Xilinx*** FPGA
- Configuration PLD and flash memory for storing FPGA configurations
- 1MB ZBT SSRAM
- Clock generators and reset sources
- A 4-way flash image selection switch and an 8-way user definable switch
- 9 user-definable surface-mounted LEDs (8G1R)
- User-definable push button
- Prototyping grid
- System bus connectors to a motherboard or other modules

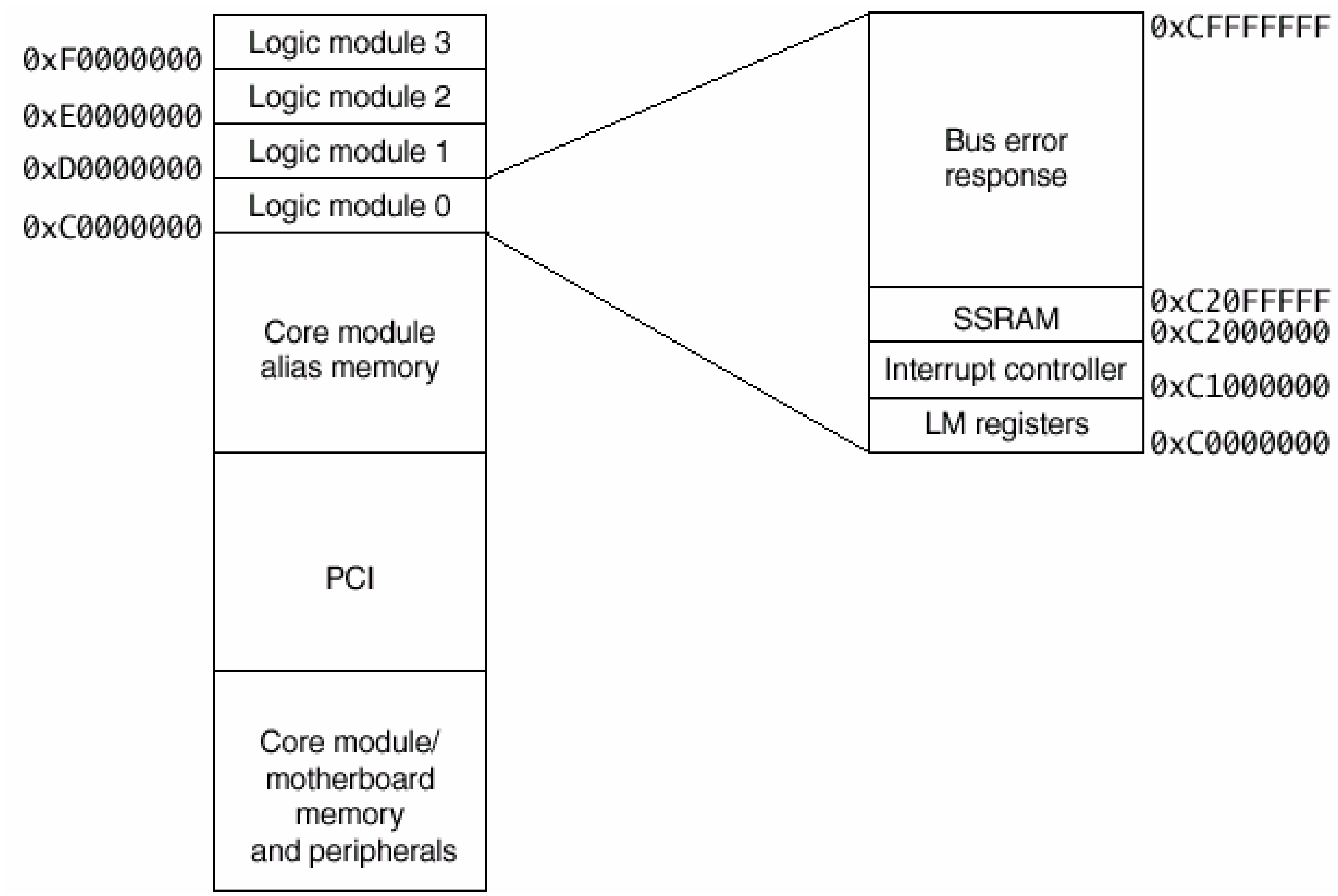
LM Layout


Module/motherboard connector
Prototyping grid
Trace connector
Multi-ICE connector
ZBT SSRAM
CONFIG link
Flash memory
Logic analyzer connector
User LEDs
FPGA
General purpose switches
8-way switch
Interface module connector
Mode switches
4-way switch
Power connector (standalone)
XChecker download connector
Status LEDs
Module/motherboard connector
Push button

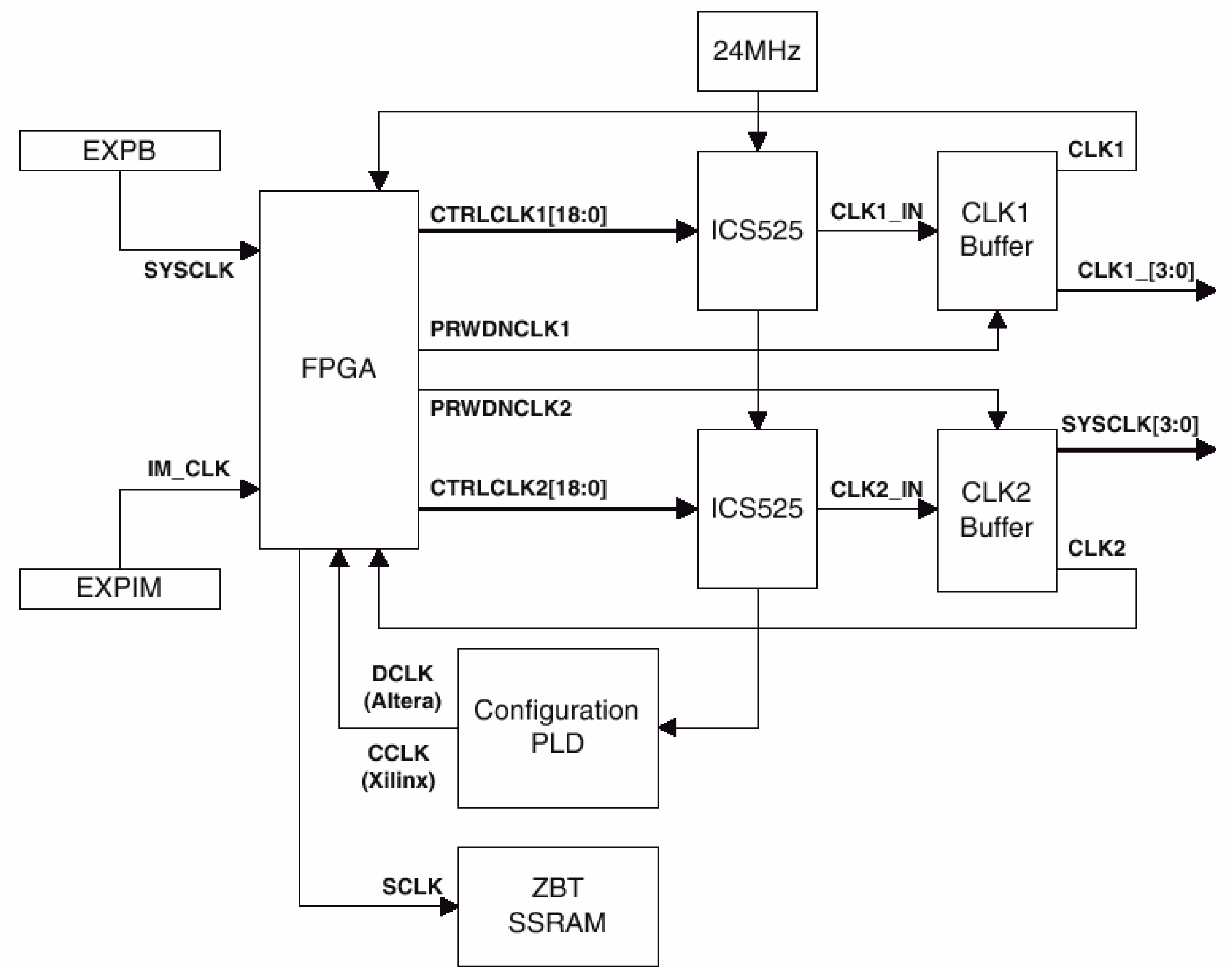
Links

- CONFIG link
  - Enable ***configuration mode***, which changes the JTAG signal routing and is used to ***download new PLD or FPGA configurations***.
- JTAG, Trace, and logic analyzer connectors
- Other links, switches, and small ICs can be added to the prototyping grid if required.

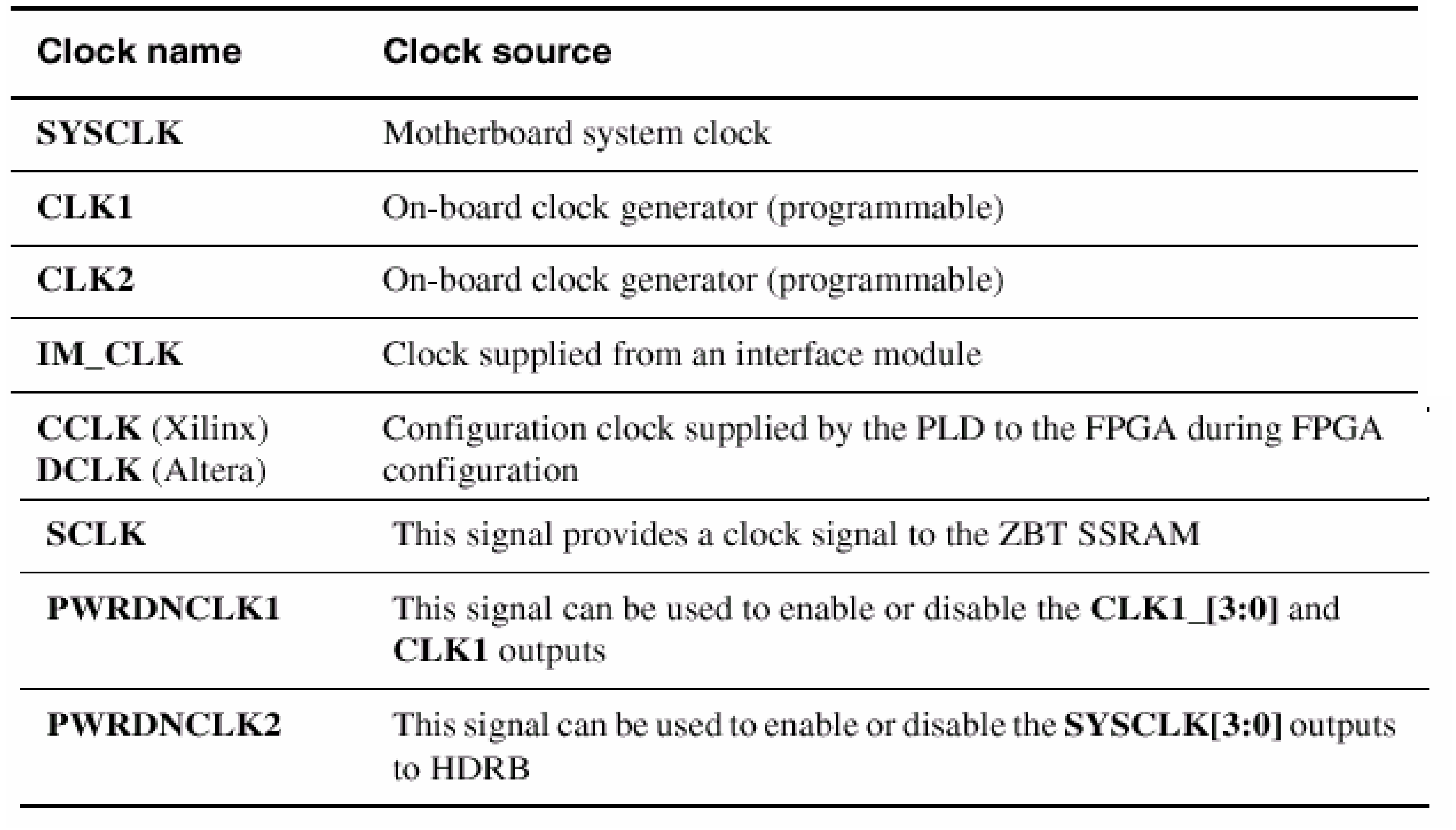
Integrator Memory Map



On-board Clock Generators



Clock Signal Summary



Clock name	Clock source
SYSCLK	Motherboard system clock
CLK1	On-board clock generator (programmable)
CLK2	On-board clock generator (programmable)
IM_CLK	Clock supplied from an interface module
CCLK (Xilinx) **DCLK** (Altera)	Configuration clock supplied by the PLD to the FPGA during FPGA configuration
SCLK	This signal provides a clock signal to the ZBT SSRAM
PWRDNCLK1	This signal can be used to enable or disable the **CLK1_[3:0]** and **CLK1** outputs
PWRDNCLK2	This signal can be used to enable or disable the **SYSCLK[3:0]** outputs to HDRB

Programming the LM Clock

Signals	Control parameter	Label
CTRLCLKx[18:16]	Output divider	S[2:0]
CTRLCLKx[15:9]	Reference divider	R[6:0]
CTRLCLKx[8:0]	VCO divider	V[8:0]

S	S[2:0]
2	001
4	011
5	100
6	111
7	101
8	010
9	110
10	000

$$\text{Frequency} = 48\text{MHz} \cdot \frac{(V[8:0]+8)}{(R[6:0]+2)\cdot S}$$

1MHz: CTRLCLKx=19'b1100111110000000100
2MHz: CTRLCLKx=19'b1100011110000000100
5MHz: CTRLCLKx=19'b1100001110000000111
10MHz: CTRLCLKx=19'b1100000110000000111

Constraint:

$$10\text{MHz} < 48\text{MHz} \cdot \frac{(V[8:0]+8)}{(R[6:0]+2)}$$

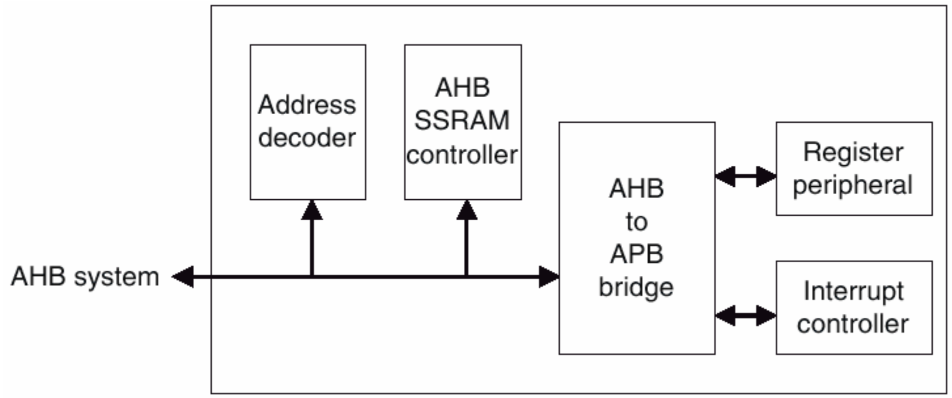
$$R[6:0] < 118$$

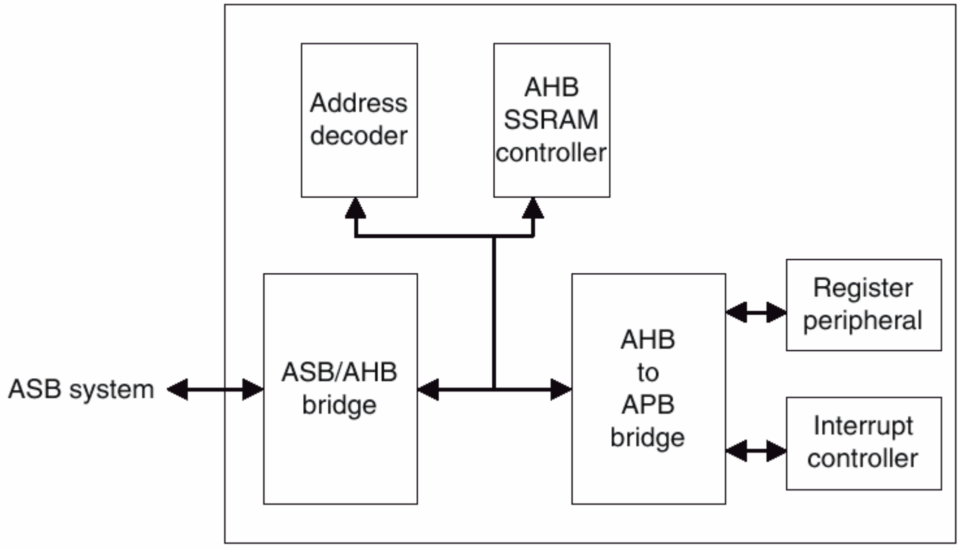
Example

- The example code operates as follows:
  1. Determines DRAM size on the core module and sets up the system controller
  2. Checks that the logic module is present in the AP expansion position
  3. Reports module information
  4. Sets the logic module clock frequencies
  5. Tests SSRAM for word, halfword, and byte accesses.
  6. Flashes the LEDs
  7. Remains in a loop that displays the switch value on the LEDs

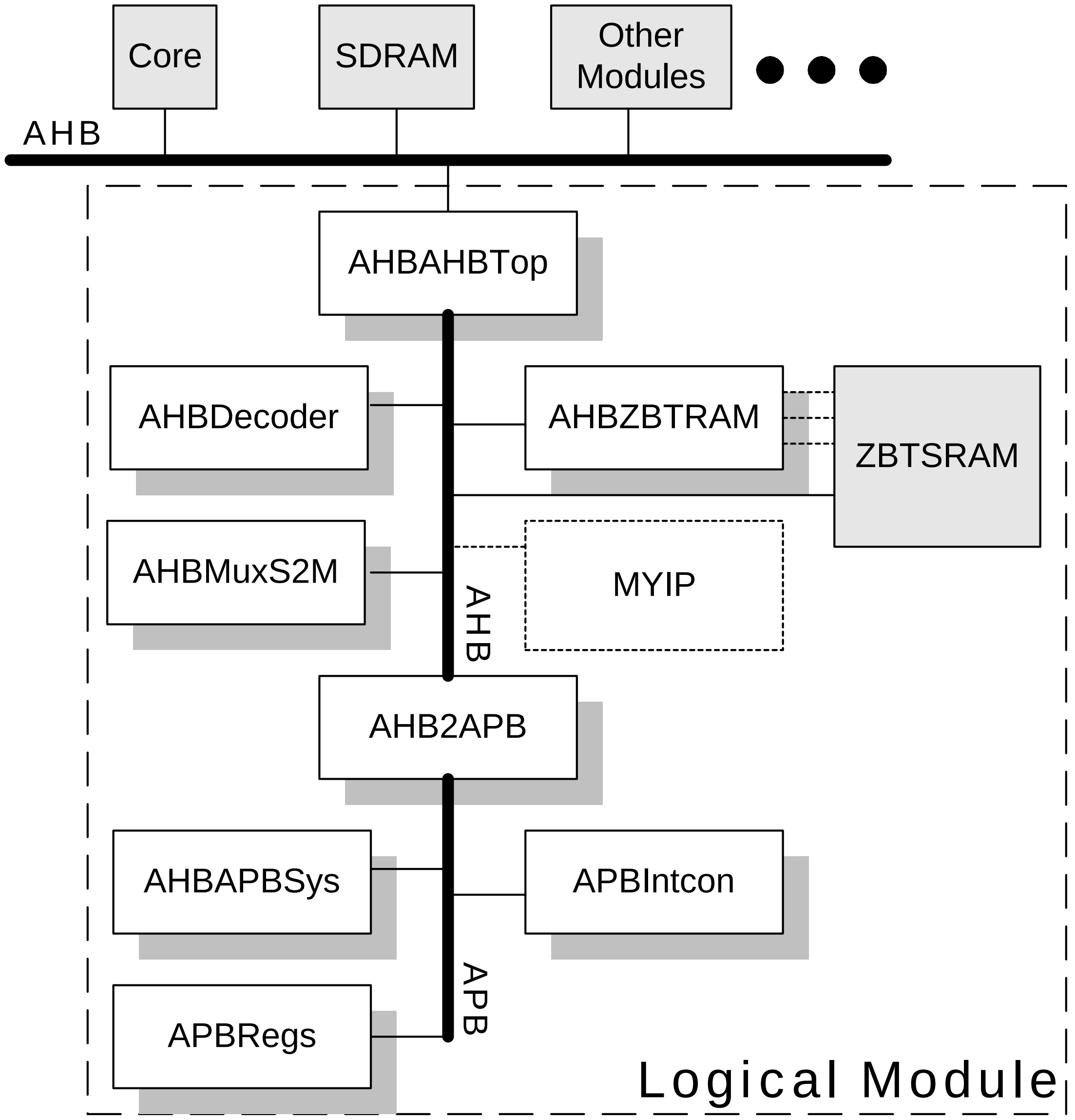
Two Platform – AHB & ASB

- Two versions of example 2 are provided to support the following implementations:
  - AHB motherboard and AHB peripherals
  - ASB motherboard and AHB peripherals
- Which AMBA has been downloaded on board can be observed by the alphanumber display
  - **H**: AHB
  - **S**: ASB





AHB Platform



Software Description

- 5 files included in .*Lab7\Codes\SW\example2\*
  - sw.mcp: project file
  - logic.c: the main C code
  - logic.h: constant definitions
  - platform.h: constant definitions
  - rw_support.s: assembly functions for SSRAM testing

Outline

- ❑ Introduction
- ❑ ARM System Overview
- ❑ ARM Integrator System Memory Map
- ❑ Prototyping with Logic Module
- ❑ ***Lab – ASIC Logic***

Lab 7: ASIC Logic

- Goal
  - HW/SW Co-verification using Rapid Prototyping
- Principles
  - Basics and work flow for prototyping with ARM Integrator
  - Target platform: AMBA AHB sub-system
- Guidance
  - Overview of examples used in the Steps
- Steps
  - Understand the files for the example designs and FPGA tool
  - Steps for synthesis with Xilinx ISE 5.1i/5.2i or Altera Quartus II