



고려대학교
KOREA UNIVERSITY

COSE321 Computer Systems Design

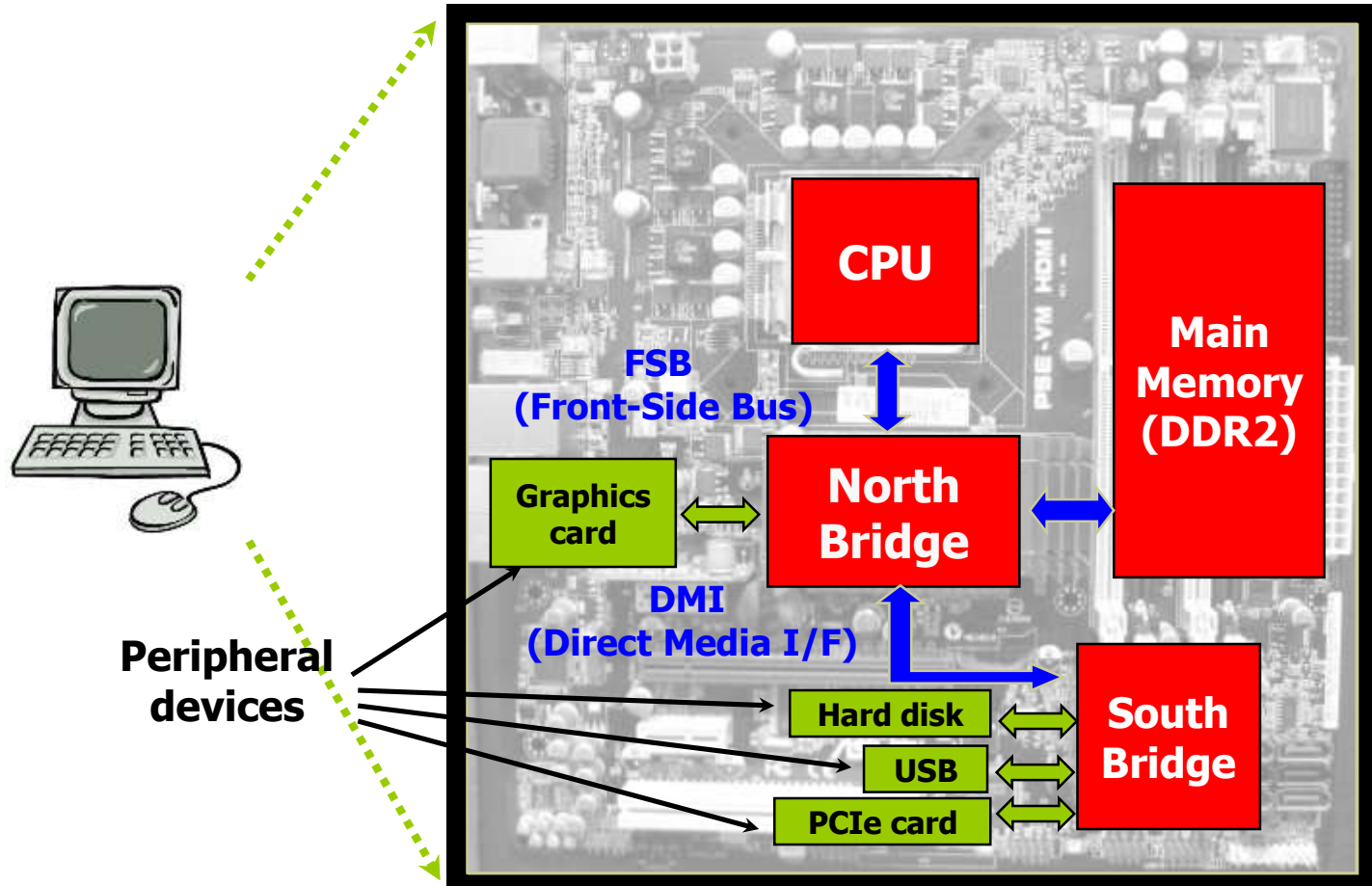
Lecture 4. Memory Map

Prof. Taeweon Suh

Computer Science & Engineering

Korea University

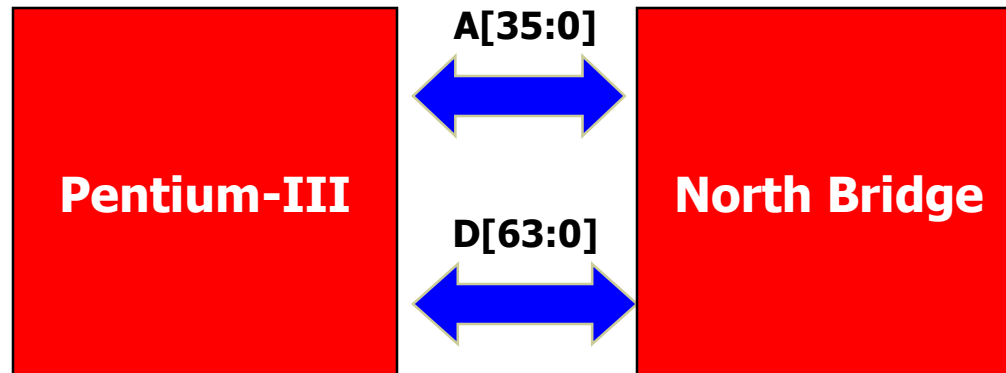
A Computer System (as of 2008)



But, don't forget the big picture!

Memory Space

- Pentium-III is 32-bit machine with 36-bit physical address
 - A[35:0] means that address bus has 36 pins for address
 - Note that we count from 0 in hardware
 - D[63:0] means that data bus has 64 pins for data



- What is the max address A[35:0] can address?
 - $0x3F_FFFF_FFFF = 64GB - 1$
 - So, it creates a 64GB space (0 ~ 64GB-1), meaning you can theoretically have up to a 64GB main memory installed in your PC

How does Computer Access I/Os?

- Computer only knows 2 kinds of devices
 - Memory and I/O (Input/output) devices
 - E.g., main memory and BIOS ROM in your PC are memory devices.
 - E.g., Keyboard and mouse are I/O devices
- When building a computer system, the address spaces for memory and I/O devices are typically assigned in advance by hardware.
- Then, how does CPU access these devices? Very simple!
 - To access memory, CPU generates memory transactions
 - To access I/O device,
 - CPU generates memory transaction if the I/O device is memory-mapped
 - CPU generates I/O transaction if the I/O device is I/O-mapped

What are Memory Map & I/O Map?

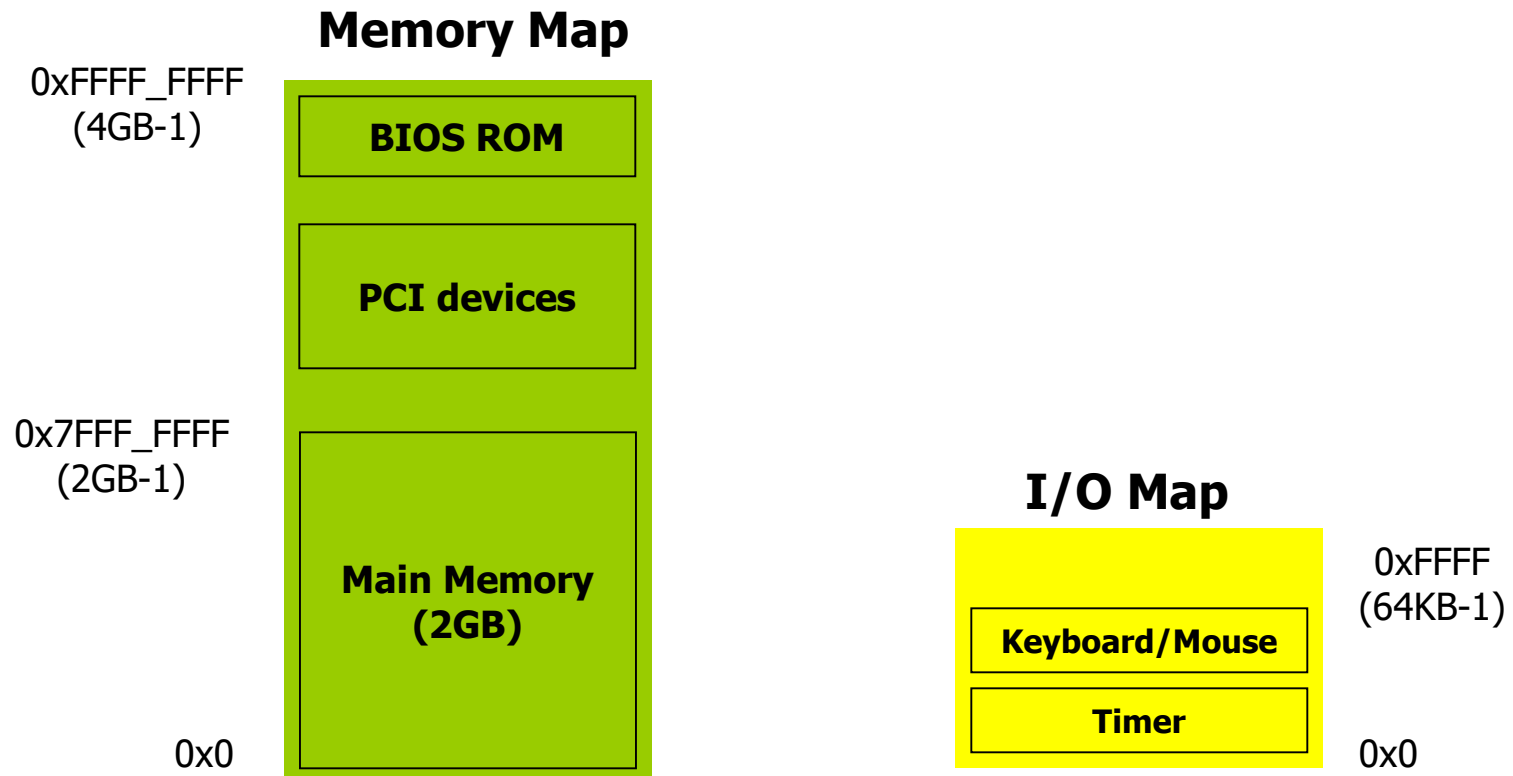
- Differentiating memory and I/O maps requires 2 things
 - ISA (Instruction Set Architecture) support
 - `mov` instruction in x86
 - `in`, `out` instruction in x86
 - Separate hardware signals from CPU to inform if it is memory or I/O transaction
- Memory Map
 - Allocate memory addresses for each resource (main memory, interrupt controller, PCI devices etc..) in a system
 - So, software can access those resources using assigned addresses
 - For example, `mov AX, [0x1000]` in x86 reads data from a memory location 0x1000 and stores it to AX register
- Memory-mapped I/O
 - I/O devices are mapped in memory space
 - For example, configuration registers in a PCI device are mapped from 0xF0000000 to 0xF000_00FF
 - `mov AX [0xF0000000]` in x86 generates memory transaction to read data from I/O device (PCI device) address 0xF000_0000

What are Memory Map & I/O Map?

- I/O Map (I/O-mapped I/O)
 - Likewise, a computer system can allocate I/O addresses for I/O devices
 - So, software can access I/O devices using assigned addresses
 - For example, `in AL, 0x21` in x86 reads data from I/O address 0x21 and stores it to the AL register

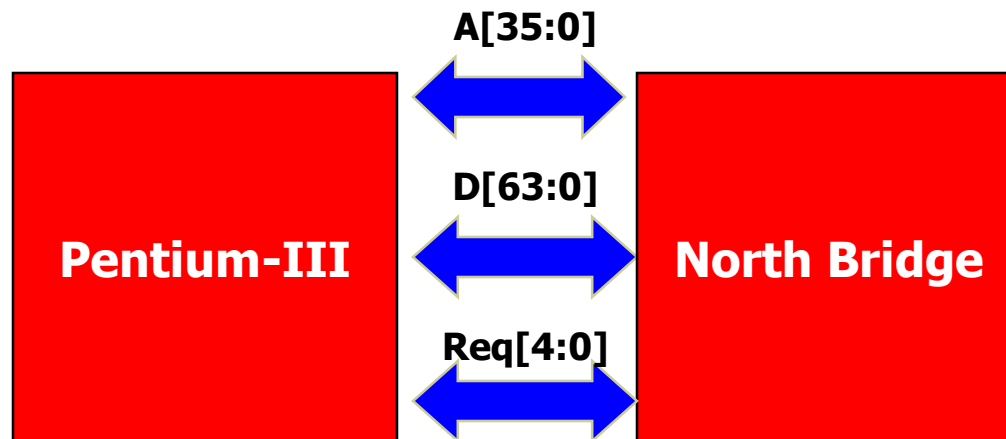
Memory & I/O Maps in x86-based System

- Assume a 32-bit physical address, meaning that CPU provides 32 pins to access memory or I/O
- Assume you have a 2GB DDR installed on your system



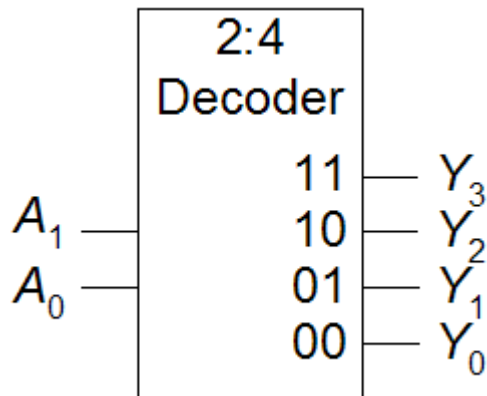
Memory & I/O Maps in x86-based System

- How does the system know if it is memory or I/O transaction?
 - There are CPU pins encoding the transaction type (Memory or I/O) when generating transaction
 - Req[4:0] in the P-III example below
 - I/O Read = b'10000, I/O Write = b'10001



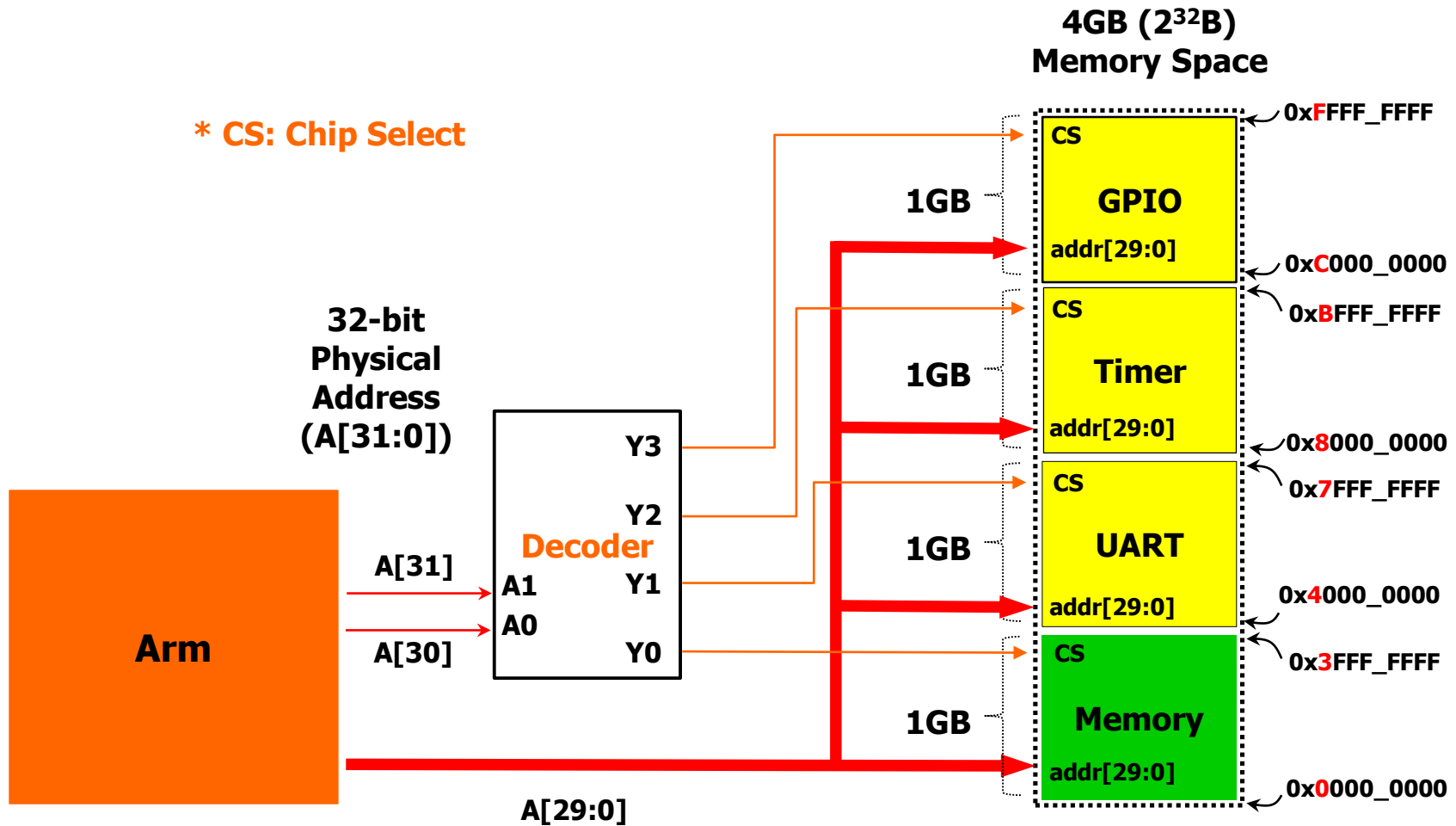
Example: Simple Memory Map

Devices	Address Range	Binary
GPIO	0xC000_0000 ~ 0xFFFF_FFFF	1100 (C) ~ 1111 (F)
Timer	0x8000_0000 ~ 0xBFFF_FFFF	1000 (8) ~ 1011 (B)
UART	0x4000_0000 ~ 0x7FFF_FFFF	0100 (4) ~ 0111 (7)
Memory	0x0000_0000 ~ 0x3FFF_FFFF	0000 (0) ~ 0011 (3)

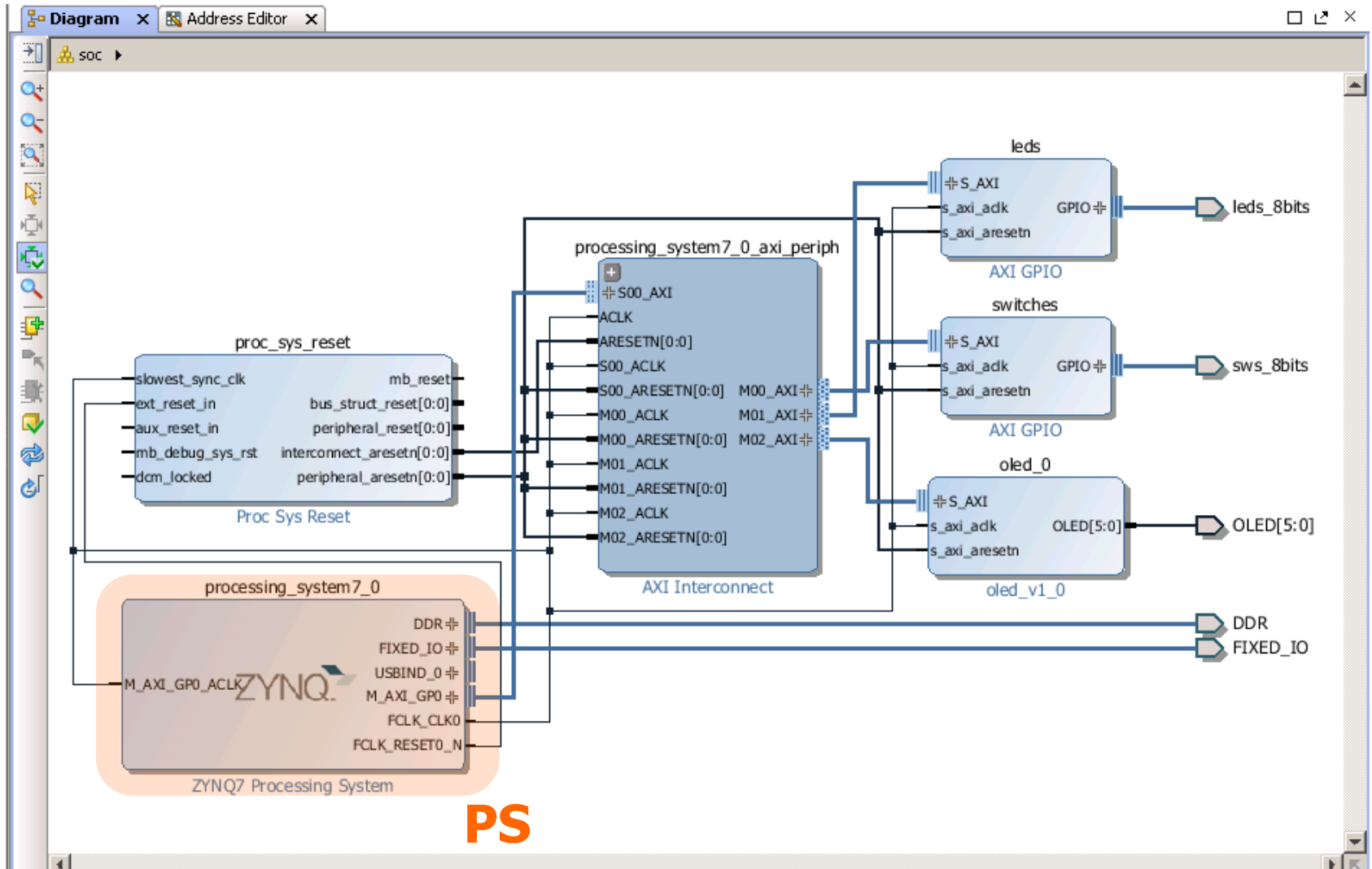


A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

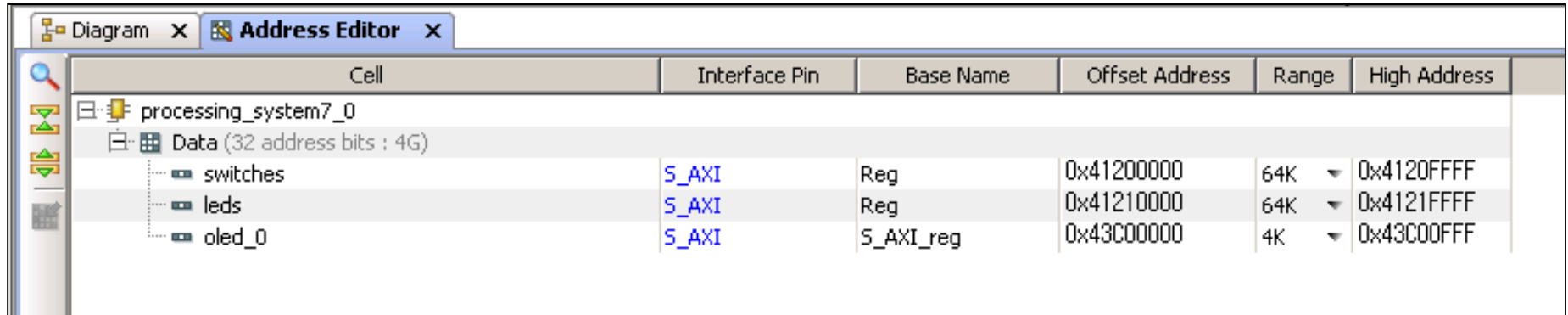
Example: Connections for Creating Memory Map



Our H/W System



Memory Map



Cell	Interface Pin	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 4G)					
switches	S_AXI	Reg	0x41200000	64K	0x4120FFFF
leds	S_AXI	Reg	0x41210000	64K	0x4121FFFF
oled_0	S_AXI	S_AXI_reg	0x43C00000	4K	0x43C00FFF

Backup Slides

Example: ARM-based Hardware System

