ECE 448
Lecture 19b

I/O Register Map of an MMIO Core

Part 2:
Simplified Address Decoding

Developing a software/hardware implementation using an FPro system

Conceptual Design:

C1. Software/hardware partitioning

C2. I/O register map of the IP core

Hardware Design:

H1. Basic circuit performing the required functionality

* datapath * controller * top-level * functional verification

H2. A wrapper matching the interface of an MMIO core

* design adjustments * coding * functional verification

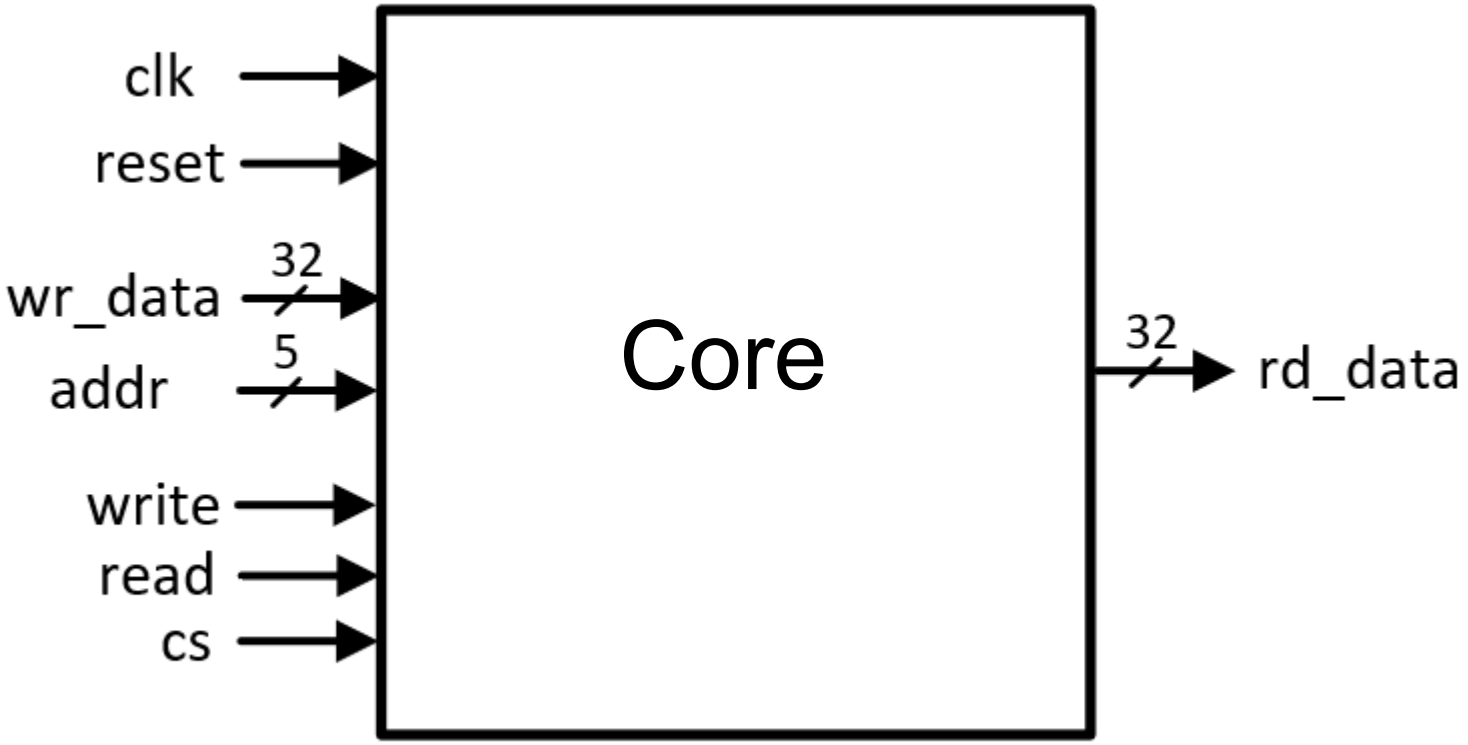
Software Design:

S1. Software driver

* declarations (.h) * implementations (.cpp) * testing

S2. Application based on functions of custom and standard cores

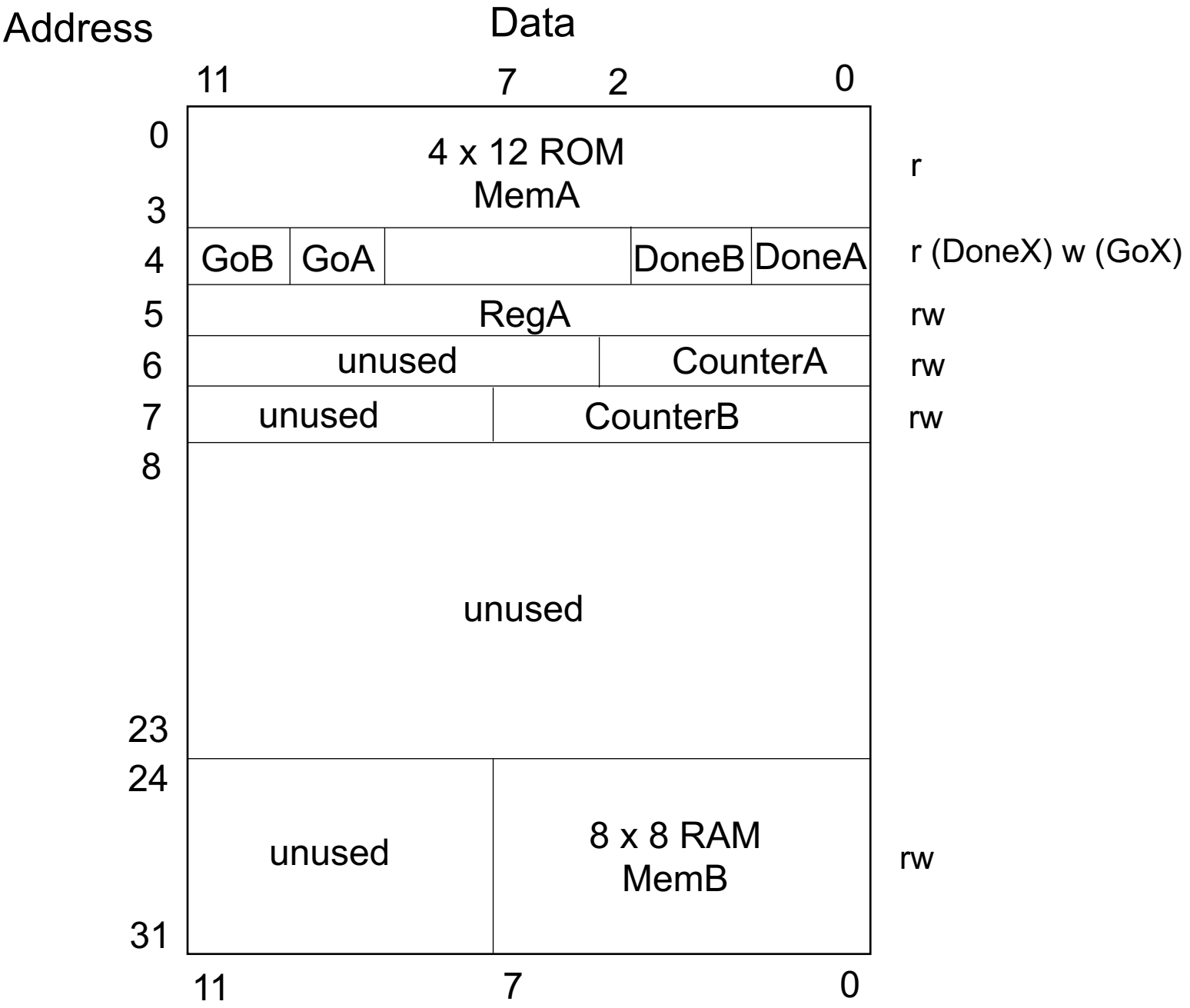
Interface of an MMIO Core



Class Exercise Specification

Task:

Draw a detailed internal block diagram of
an FPro MMIO unit with the standard interface
and the Memory & I/O Register Map
shown on the next page


Address
Data
11
7
2
0
0
3
4 x 12 ROM
MemA
r
4
GoB
GoA
DoneB
DoneA
r (DoneX) w (GoX)
5
RegA
rw
6
unused
CounterA
rw
7
unused
CounterB
rw
8
unused
23
24
unused
8 x 8 RAM
MemB
rw
31
11
7
0

Assumptions

- Registers must be implemented using actual registers, not memory. Register RegA can be written to and read from.
- CounterA is a 3-bit counter, CounterB is an 8-bit counter. Writing to a counter initializes this counter, reading from a counter, reads its current value.
  Assume that initializing a counter requires only asserting ld.
- GoA and GoB and 1-bit write-only flags located at the two most significant bit locations at address 4.
- DoneA and DoneB and 1-bit read-only flags located at the two least significant bit locations at address 4.

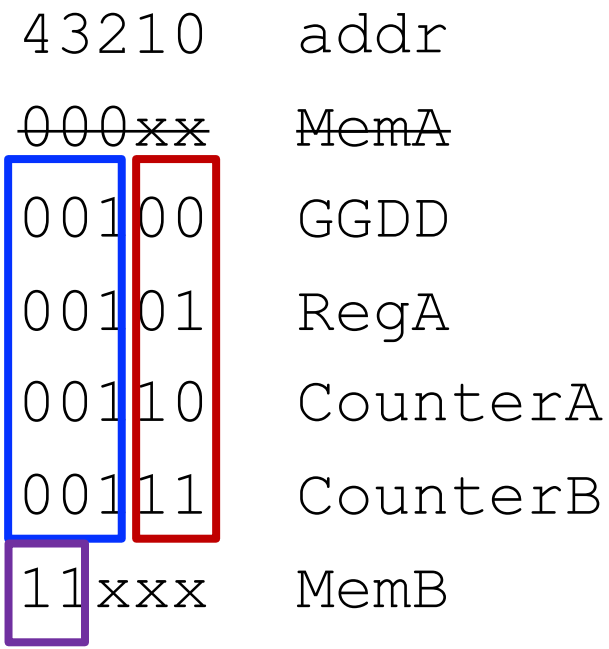
Address Decoding

	16	8	4	2	1
addr	4	3	2	1	0
min	0	0	0	0	0
max	0	0	0	1	1
MemA	0	0	0	x	x

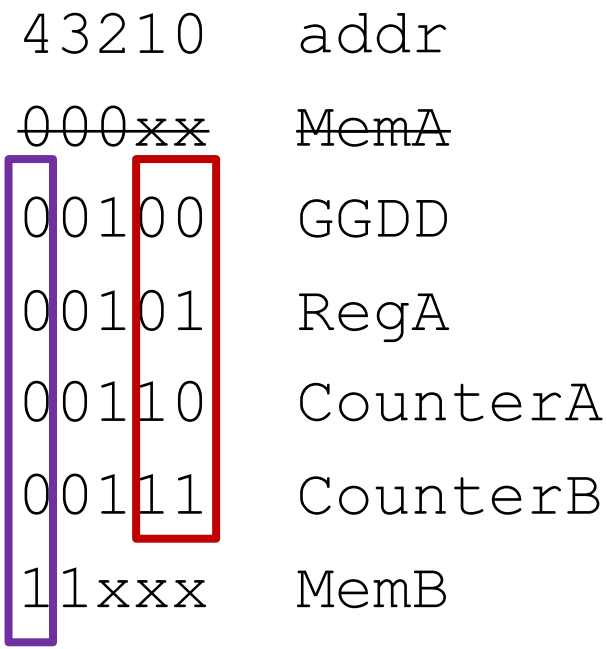
	16	8	4	2	1
addr	4	3	2	1	0
min	1	1	0	0	0
max	1	1	1	1	1
MemB	1	1	x	x	x

43210	addr
000xx	MemA
00100	GGDD
00101	RegA
00110	CounterA
00111	CounterB
11xxx	MemB

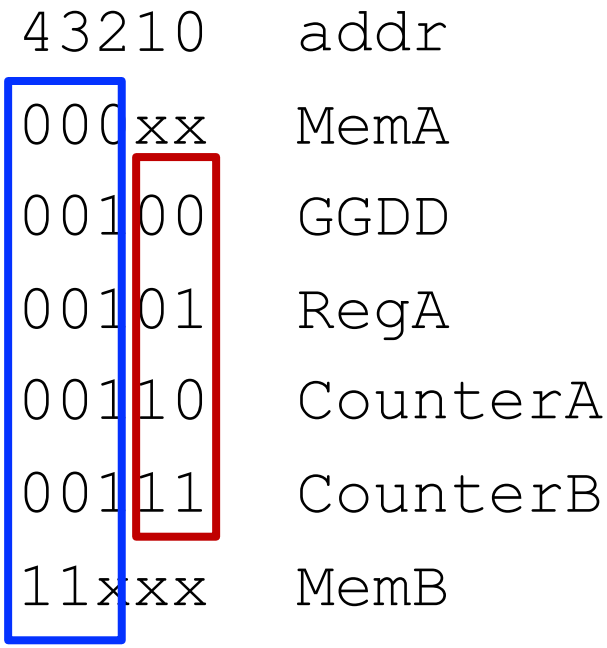
Exact Address Decoding for Writing



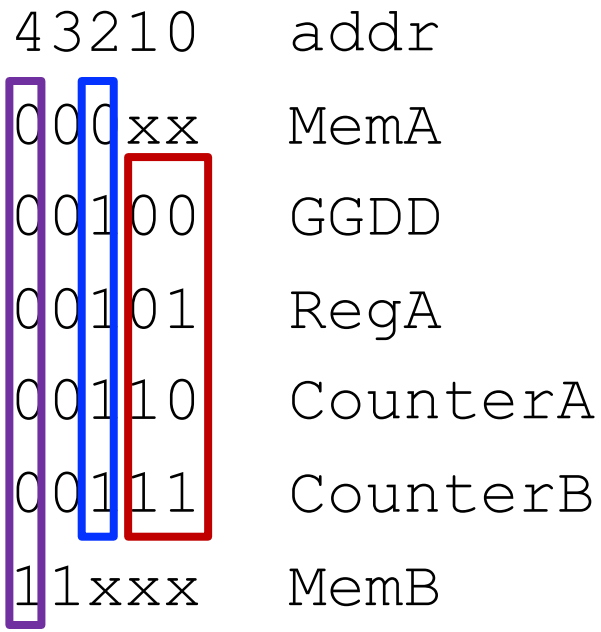
Simplified Address Decoding for Writing



Exact Address Decoding for Reading



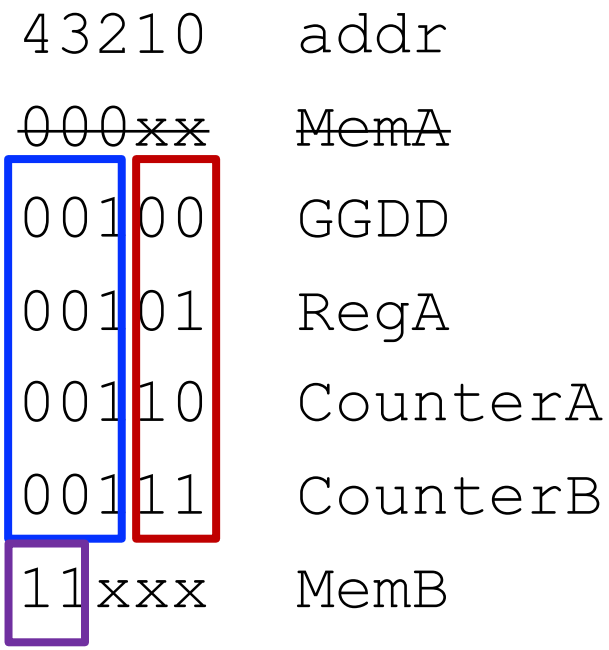
Simplified Address Decoding for Reading



Approach 2

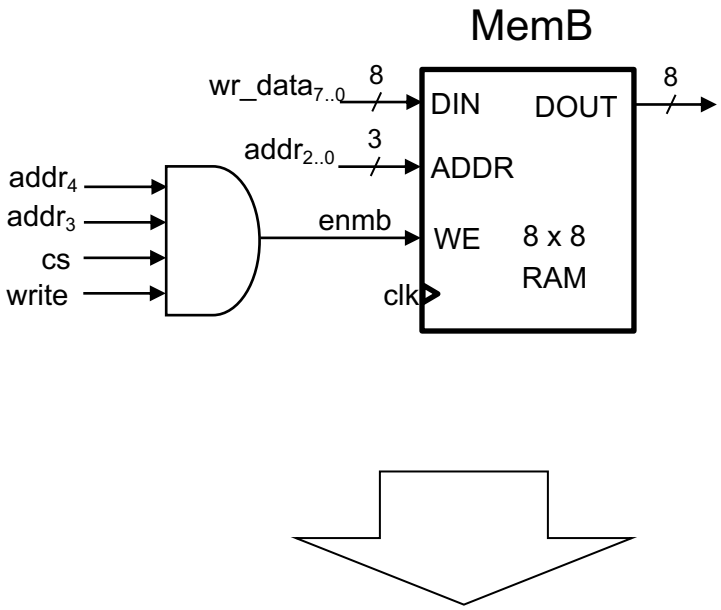
Simplified Address Decoding

Exact Address Decoding for Writing



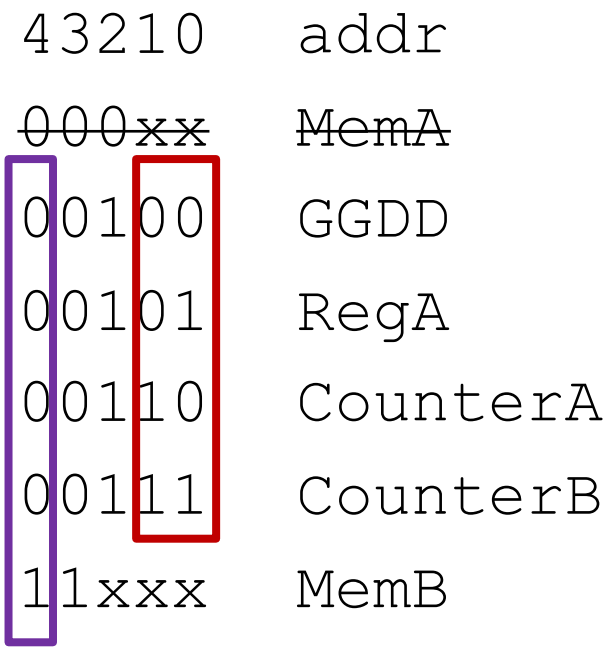
MemB

EXACT



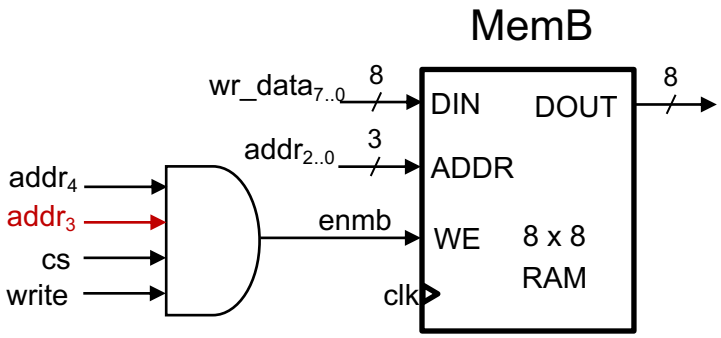
SIMPLIFIED

Simplified Address Decoding for Writing

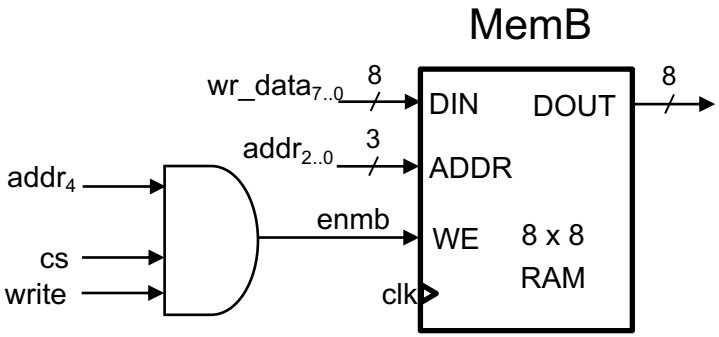


MemB

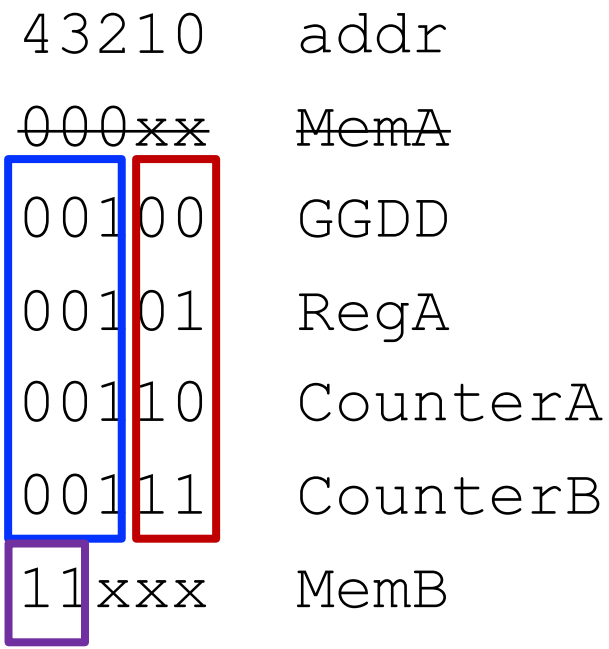
EXACT



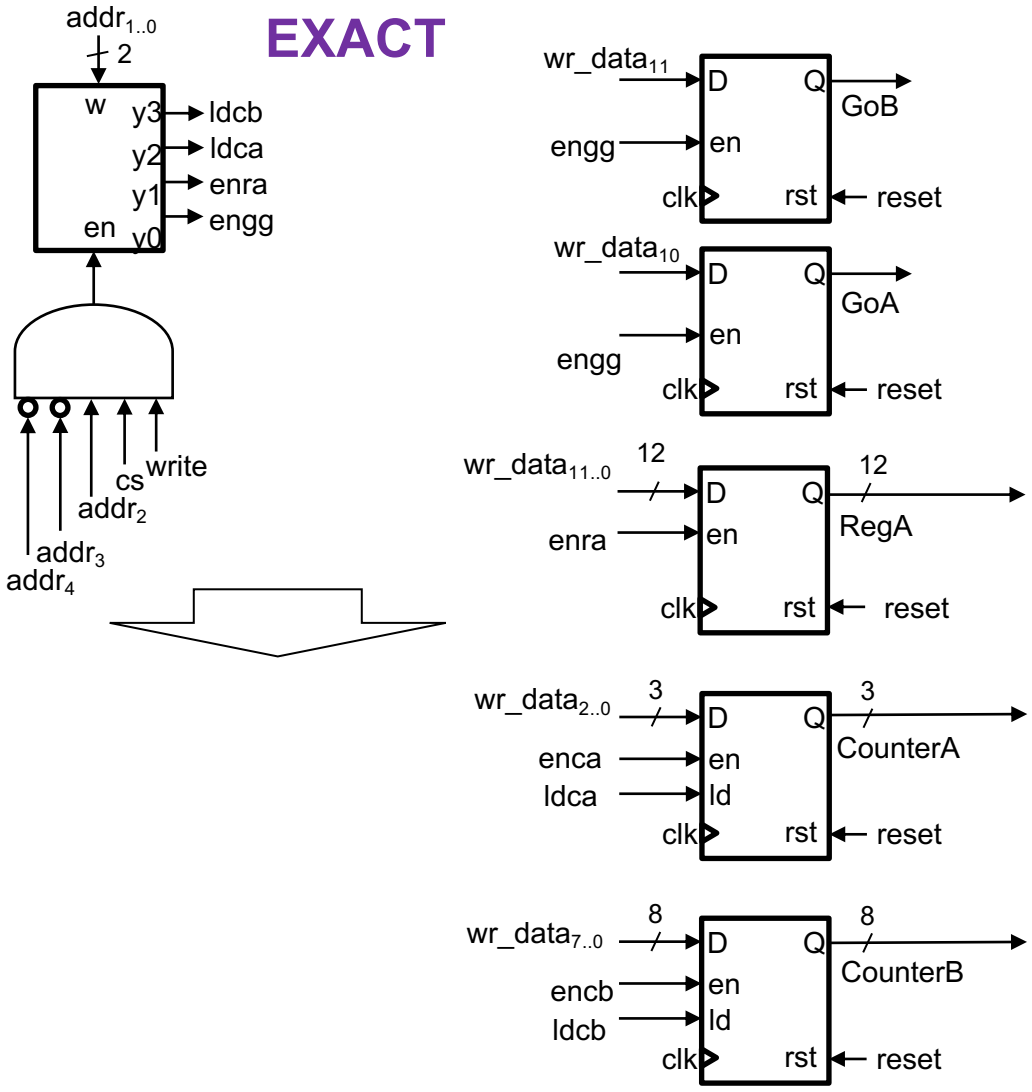
SIMPLIFIED



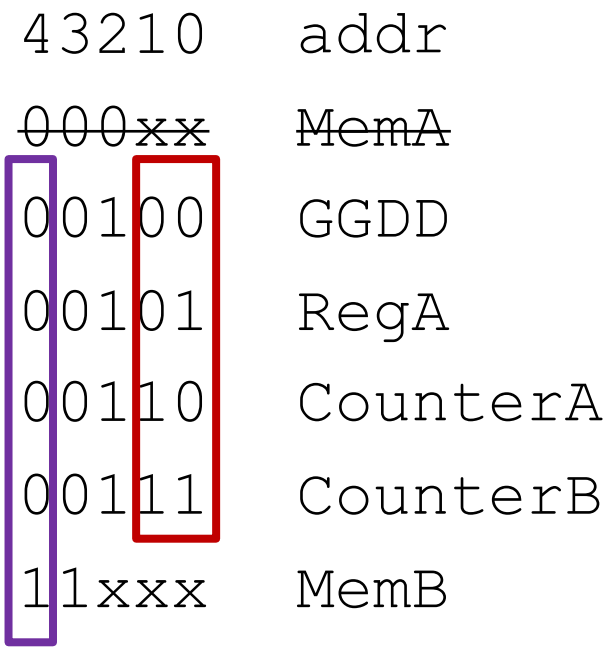
Exact Address Decoding for Writing



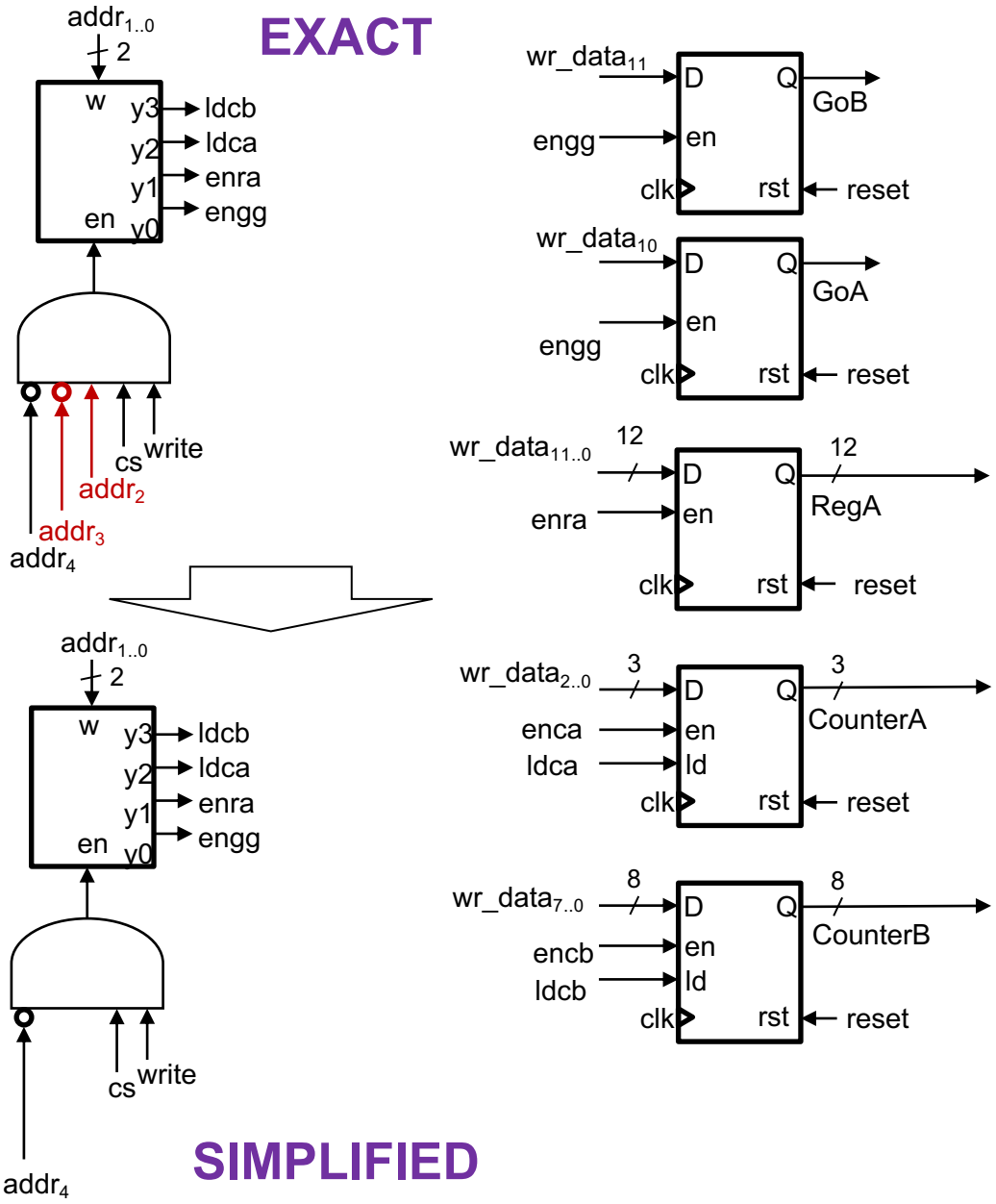
GoA, GoB, RegA, CounterA, and CounterB



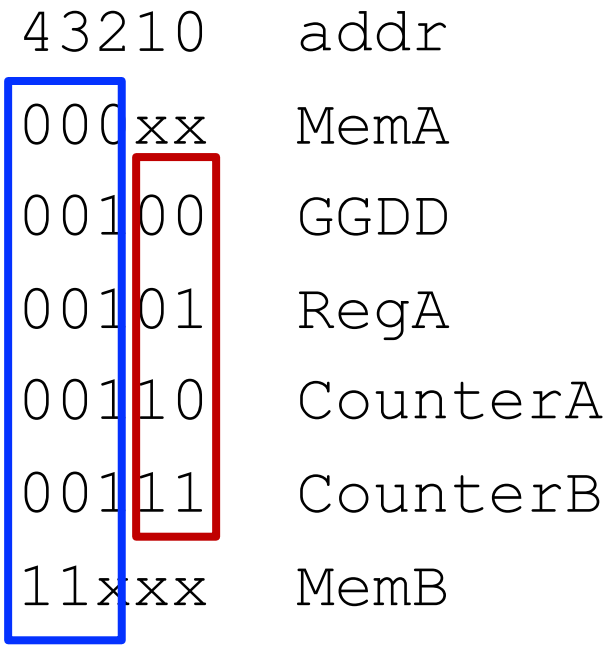
Simplified Address Decoding for Writing



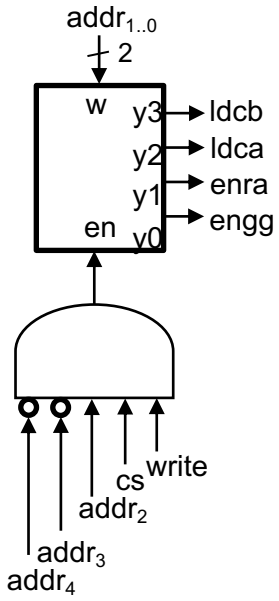
GoA, GoB, RegA, CounterA, and CounterB

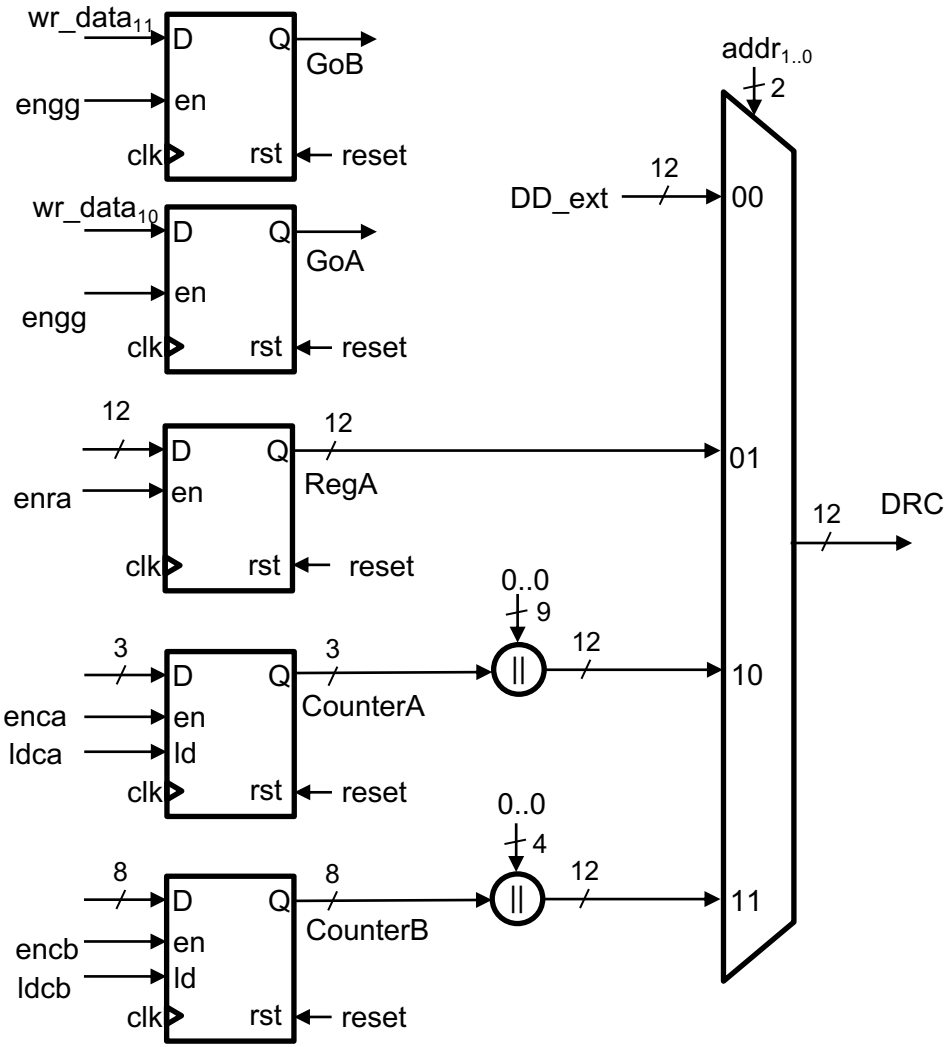


Exact Address Decoding for Reading



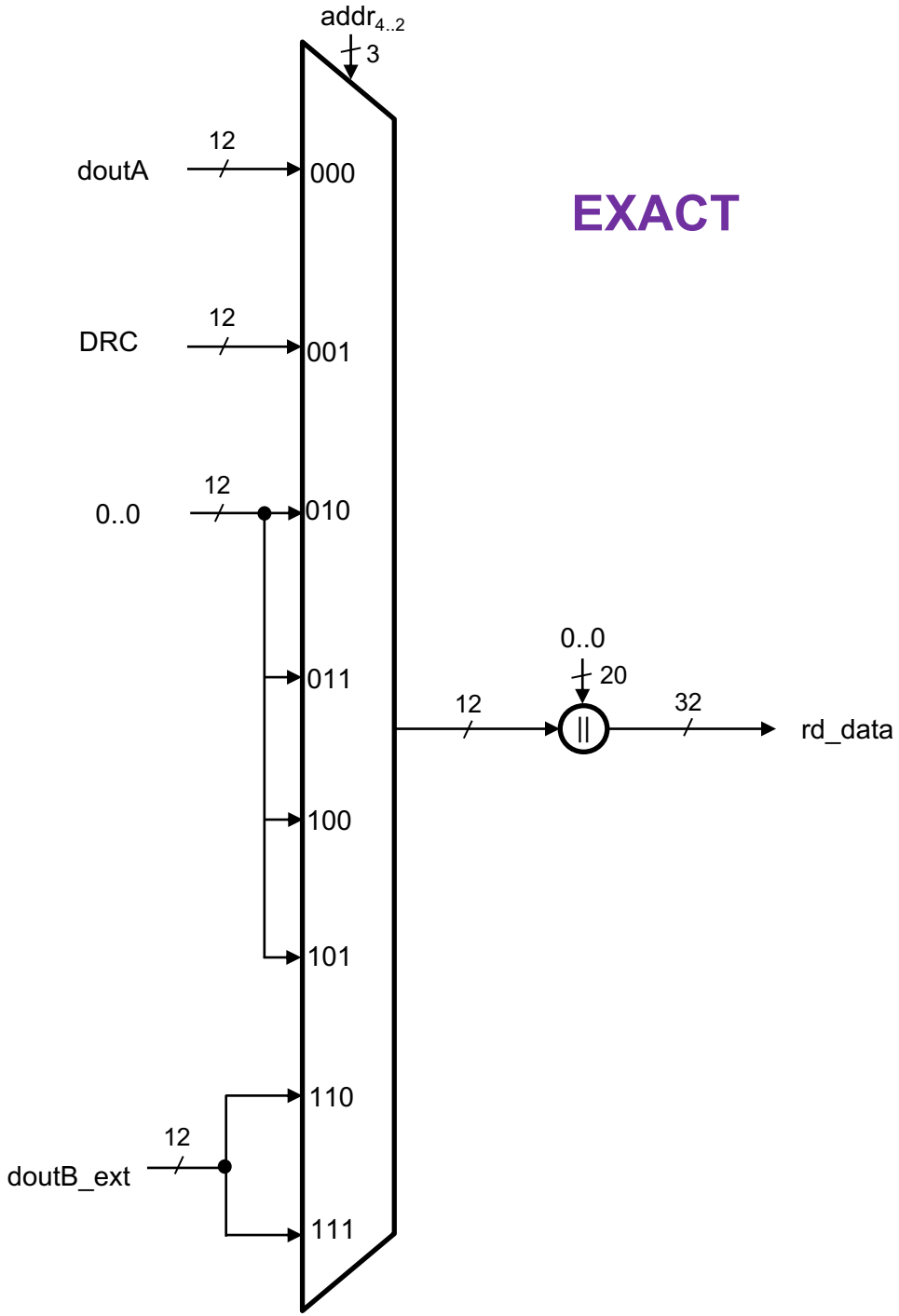
GoA, GoB, RegA, CounterA, and CounterB



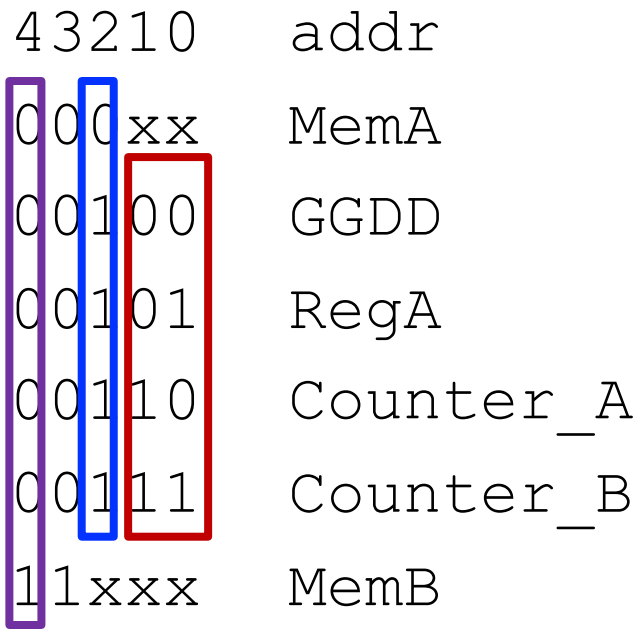


EXACT

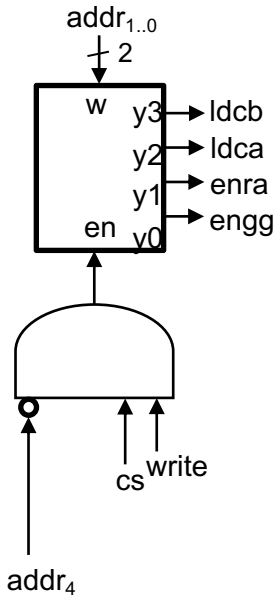
Output MUX

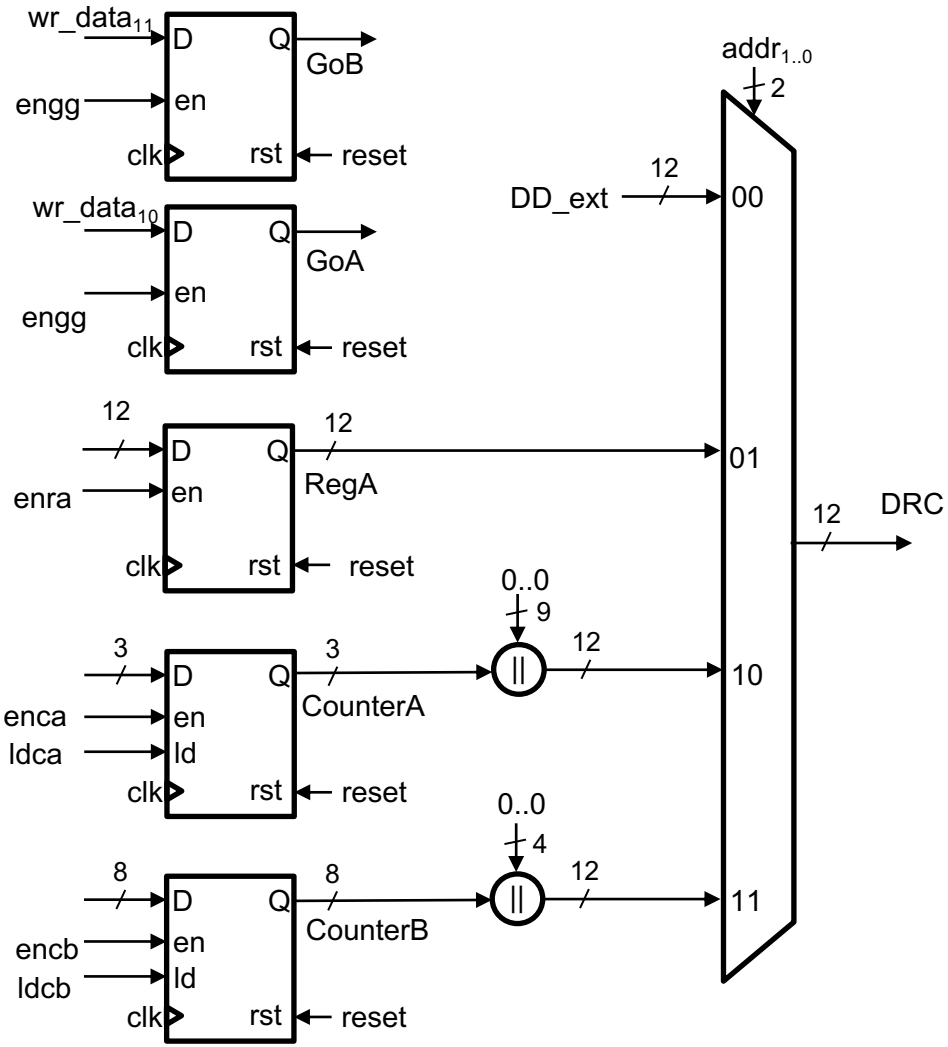

addr4..2
3
12
doutA
000
EXACT
12
DRC
001
12
0..0
010
0..0
20
011
12
32
rd_data
100
101
110
12
doutB_ext
111

Simplified Address Decoding for Reading



GoA, GoB, RegA, CounterA, and CounterB





Output MUXes

SIMPLIFIED