



CSCI 150

Introduction to Digital and Computer System Design

Lecture 5: Registers III



Jetic Gū

Overview

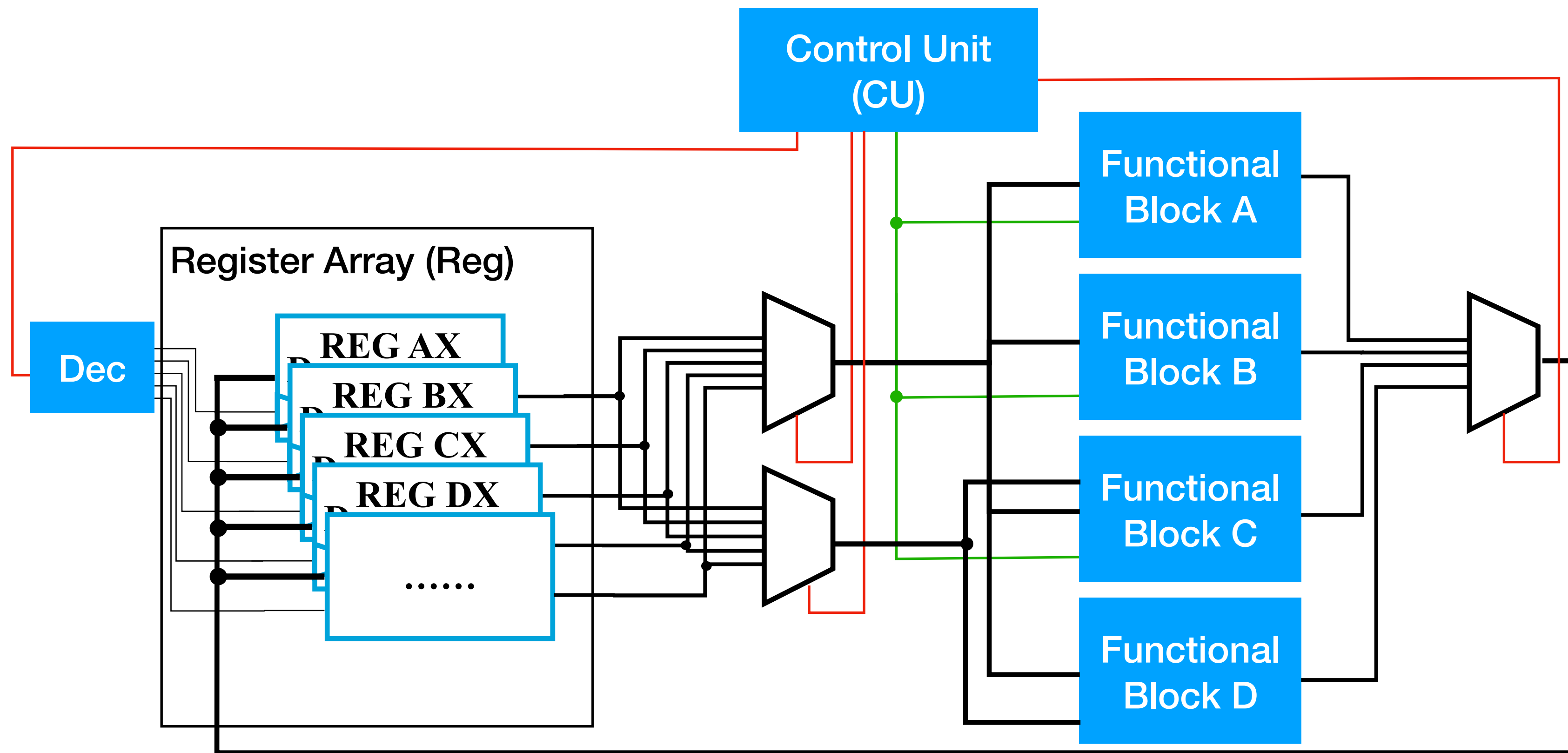
- Focus: Fundamentals of Complex Digital Circuit Design
- Architecture: von Neumann
- Textbook v4: Ch7 7.6; v5: Ch6 6.6
- Core Ideas:
 1. Implementation of Register Microoperations

Register Transfer Operations

	Operator	Example
Assignment	<code><=</code>	<code>ax <= 12h</code>
Reg. Transfer	<code><=</code>	<code>ax <= bx</code>
Addition	<code>+</code>	<code>ax + bx</code>
Subtraction	<code>-</code>	<code>ax - bx</code>
Shift Left	<code>sll</code>	<code>ax sll 2</code>
Shift Right	<code>srl</code>	<code>ax srl 2</code>

	Operator	Example
Bitwise AND	<code>and</code>	<code>ax and bx</code>
Bitwise OR	<code>or</code>	<code>ax or bx</code>
Bitwise NOT	<code>not</code>	<code>not ax</code>
Bitwise XOR	<code>xor</code>	<code>ax xor bx</code>
Vectors		<code>ax(3 down to 0)</code> <code>ax(3 down to 0)</code>
Concatenate	<code>&</code>	<code>ax(7 down to 4)</code> <code>&ax(3 down to 0)</code>

Example Datapath Architecture



Implementation of Datapath II

Individual Functional Blocks

Single Register Microoperations

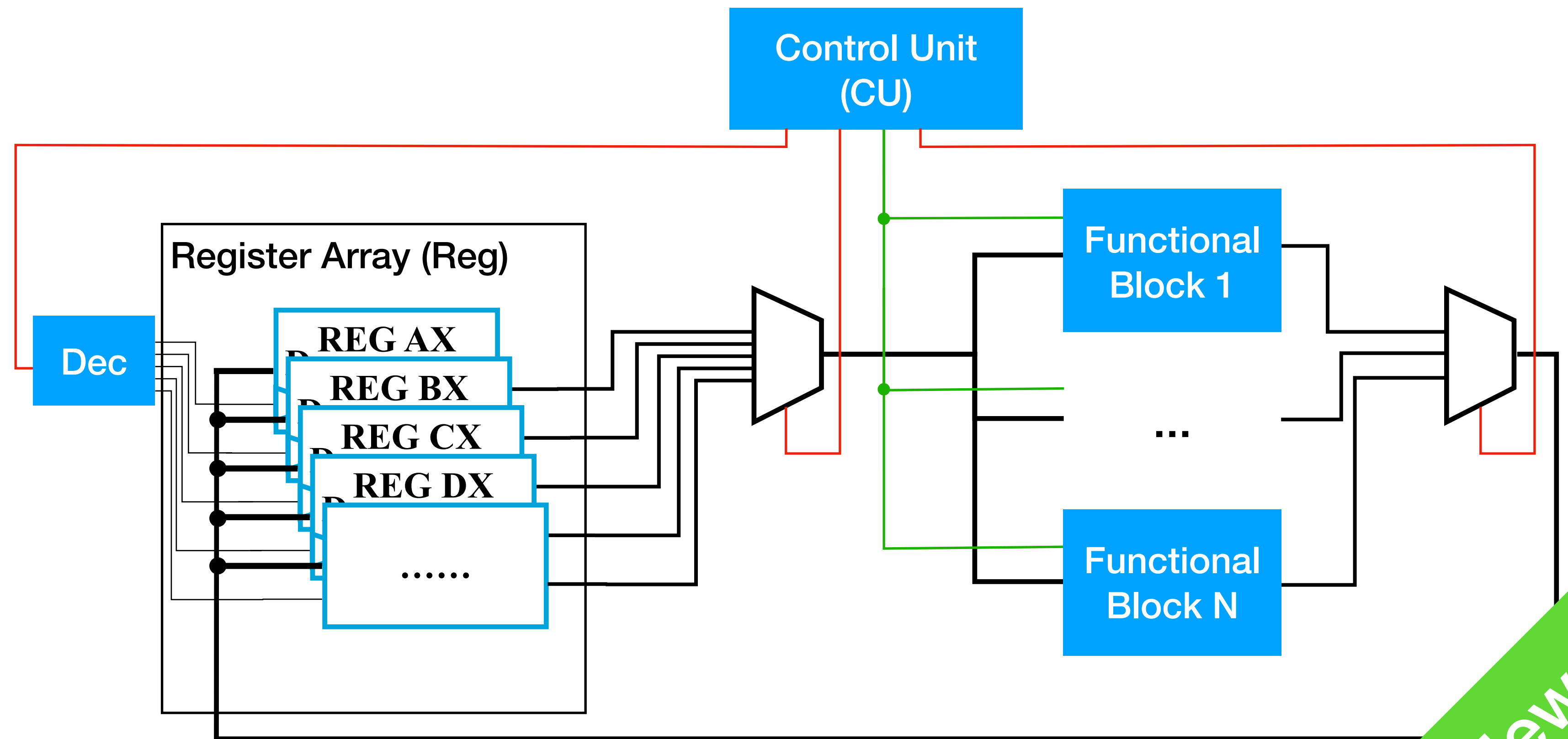
	Operator	Example		Operator	Example
Assignment	<code><=</code>	<code>ax <= 12h</code>	Bitwise AND	<code>and</code>	<code>ax and bx</code>
Reg. Transfer	<code><=</code>	<code>ax <= bx</code>	Bitwise OR	<code>or</code>	<code>ax or bx</code>
Addition	<code>+</code>	<code>ax + bx</code>	Bitwise NOT	<code>not</code>	<code>not ax</code>
Subtraction	<code>-</code>	<code>ax - bx</code>	Bitwise XOR	<code>xor</code>	<code>ax xor bx</code>
Shift Left	<code>sll</code>	<code>ax sll 2</code>	Vectors		<code>ax(3 down to 0) ax(3 down to 0)</code>
Shift Right	<code>srl</code>	<code>ax srl 2</code>	Concatenate	<code>&</code>	<code>ax(7 down to 4) &ax(3 down to 0)</code>

Single Register Microoperations

Red: Address

Green: Mode

- **Select** Input REG to be fed into Processing Blocks
- Select **mode** for Processing Blocks
- **Select** Processing Block for output to REG
- **Select** Receiving REG



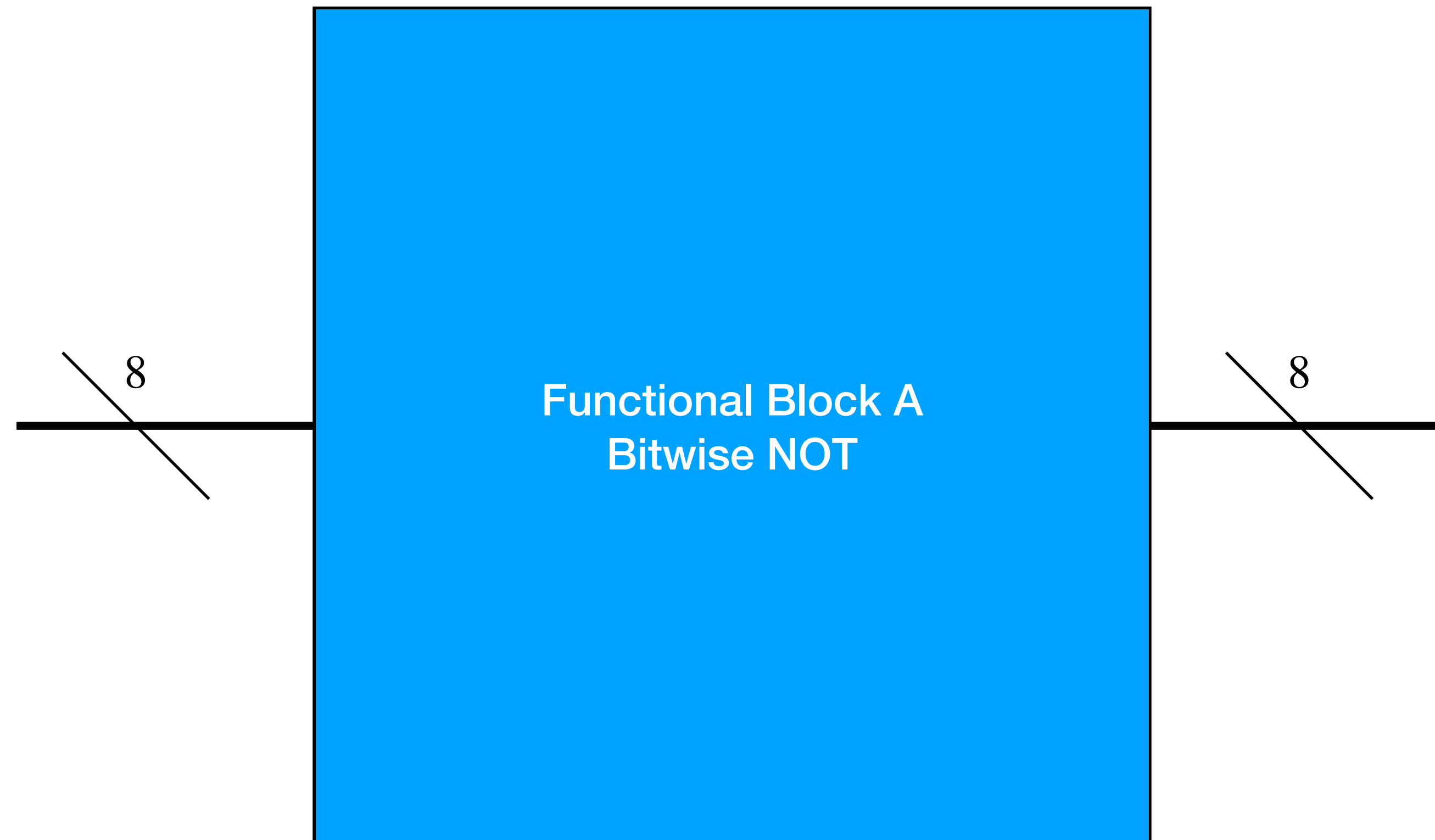
Review

Single Register Microoperations

- Each individual functional block design identical to the functional blocks we've covered in Lecture 3
- For operations completed in a single CLK tick, use the 5 step design procedures

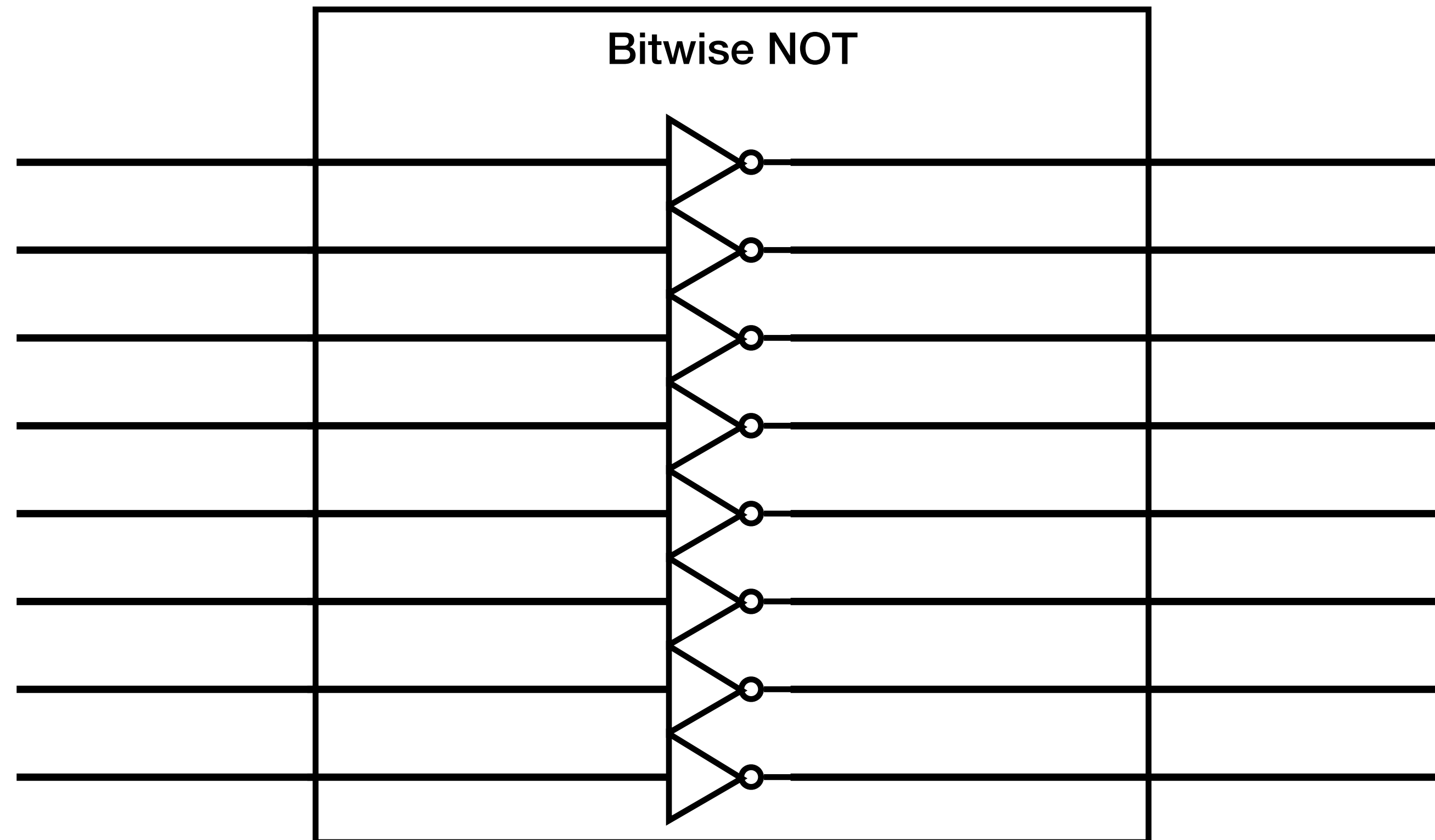
Single Register Microoperations

- Bitwise NOT



Single Register Microoperations

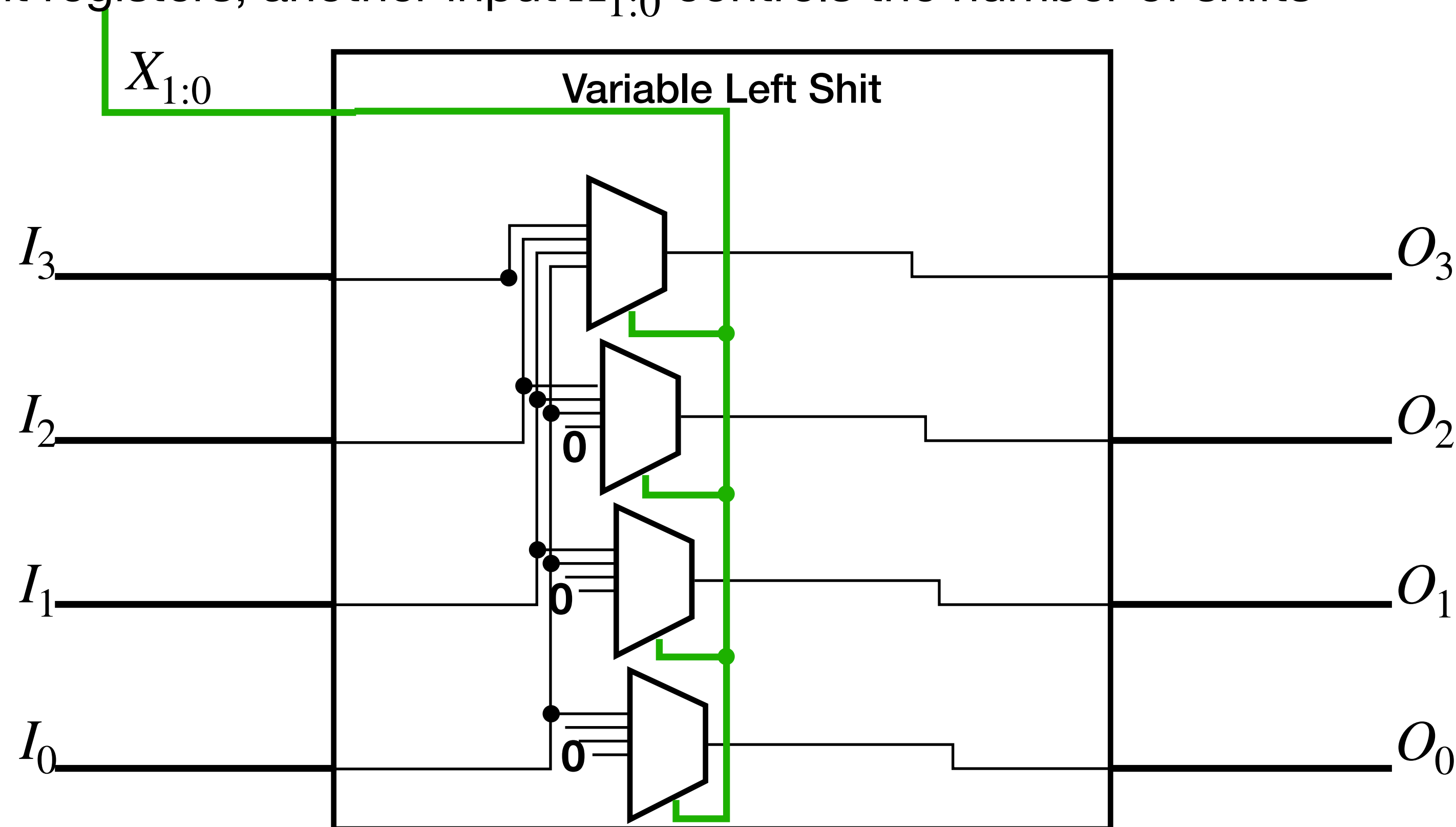
- Bitwise NOT



Single Register Microoperations

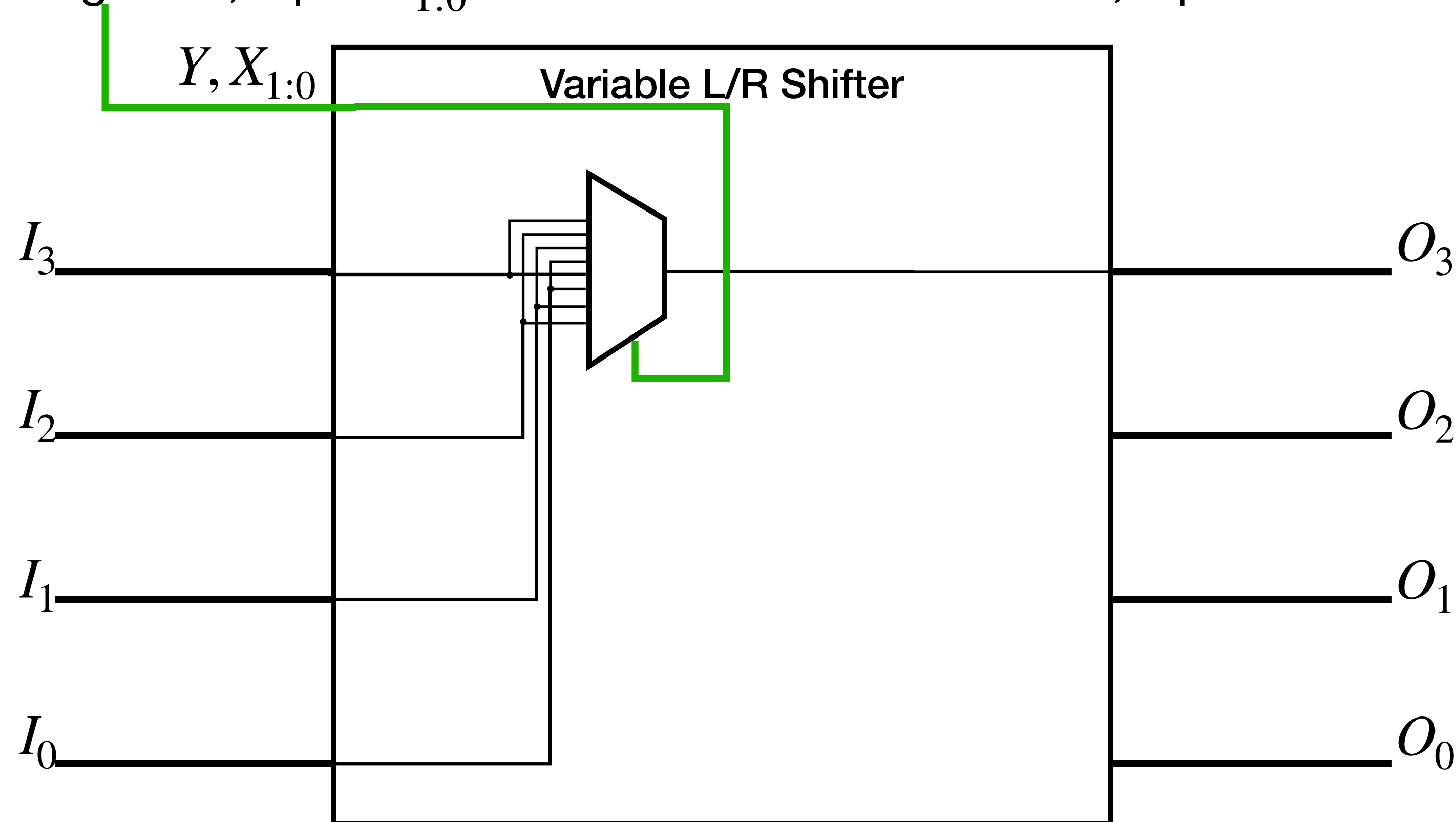
- Left Shifter: variable shifts

Assuming 4-bit registers, another input $X_{1:0}$ controls the number of shifts

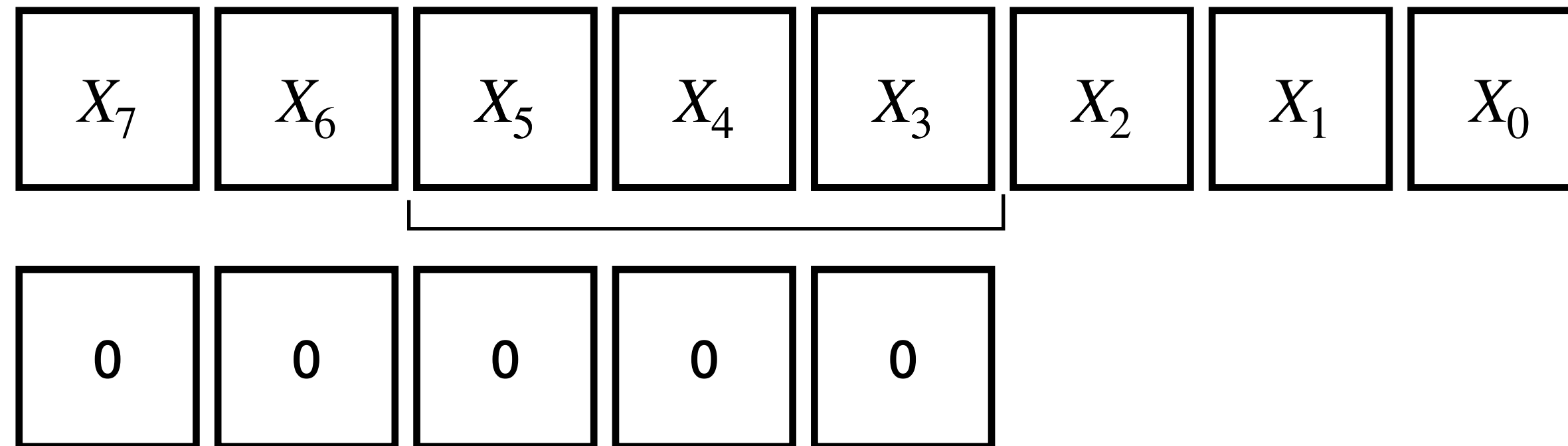


Single Register Microoperations

- Left/Right Shifter: variable shifts
Assuming 4-bit registers, input $X_{1:0}$ controls the number of shifts, input Y for direction (0:L; 1:R)



Single Register Microoperations



- Vector: I down to J (e.g. `ax (5 down to 3)`: $I = 5, J = 3$)
- Decompose the problem
- J : the number of right shifts to perform
- I : masking out everything before position I (X_7, \dots, X_{I+1}), to value 0

Multiple Register Microoperations

	Operator	Example		Operator	Example
Assignment	<code><=</code>	<code>ax <= 12h</code>	Bitwise AND	<code>and</code>	<code>ax and bx</code>
Reg. Transfer	<code><=</code>	<code>ax <= bx</code>	Bitwise OR	<code>or</code>	<code>ax or bx</code>
Addition	<code>+</code>	<code>ax + bx</code>	Bitwise NOT	<code>not</code>	<code>not ax</code>
Subtraction	<code>-</code>	<code>ax - bx</code>	Bitwise XOR	<code>xor</code>	<code>ax xor bx</code>
Shift Left	<code>sll</code>	<code>ax sll 2</code>	Vectors		<code>ax(3 down to 0)</code> <code>ax(3 down to 0)</code>
Shift Right	<code>srl</code>	<code>ax srl 2</code>	Concatenate	<code>&</code>	<code>ax(7 down to 4)</code> <code>&ax(3 down to 0)</code>

Concept

LogicWork Exercise

- Design your own Shifter
- Design your own register array
- Lab 4 Q2