Logic and Computer Design Fundamentals

Chapter 9 – Computer Design Basics

Part 1 – Datapaths

Charles Kime & Thomas Kaminski

(Hyperlinks are active in View Show mode)

Overview

- **Part 1 – Datapaths**
  - **Introduction**
  - **Datapath Example**
  - **Arithmetic Logic Unit (ALU)**
  - **Shifter**
  - **Datapath Representation and Control Word**
- **Part 2 – A Simple Computer**
- **Part 3 – Multiple Cycle Hardwired Control**

Introduction

- **Computer Specification**
  - ***Instruction Set Architecture (ISA)*** **- the specification of a computer's appearance to a programmer at its lowest level**
  - ***Computer Architecture*** **- a high-level description of the hardware implementing the computer derived from the ISA**
  - **The architecture usually includes additional specifications such as speed, cost, and reliability.**

Introduction (continued)

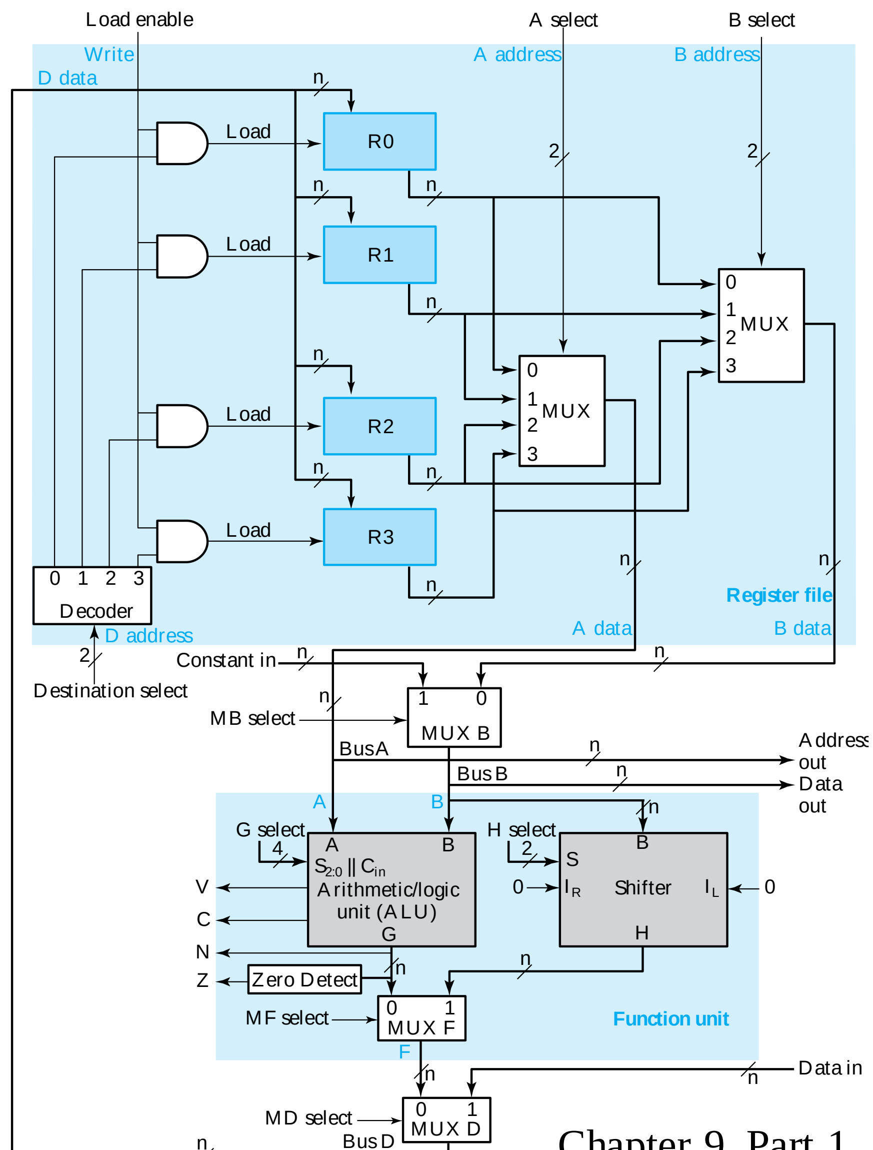
- **Simple computer architecture decomposed into:**
  - **Datapath for performing operations**
  - **Control unit for controlling datapath operations**
- **A *datapath* is specified by:**
  - **A set of registers**
  - **The microoperations performed on the data stored in the registers**
  - **A control interface**

Datapaths

- **Guiding principles for basic datapaths:**
  - **The set of registers**
    - **Collection of individual registers**
    - **A set of registers with common access resources called a *register file***
    - **A combination of the above**
  - **Microoperation implementation**
    - **One or more shared resources for implementing microoperations**
    - **Buses - shared transfer paths**
    - ***Arithmetic-Logic Unit (ALU)* - shared resource for implementing arithmetic and logic microoperations**
    - **Shifter - shared resource for implementing shift microoperations**

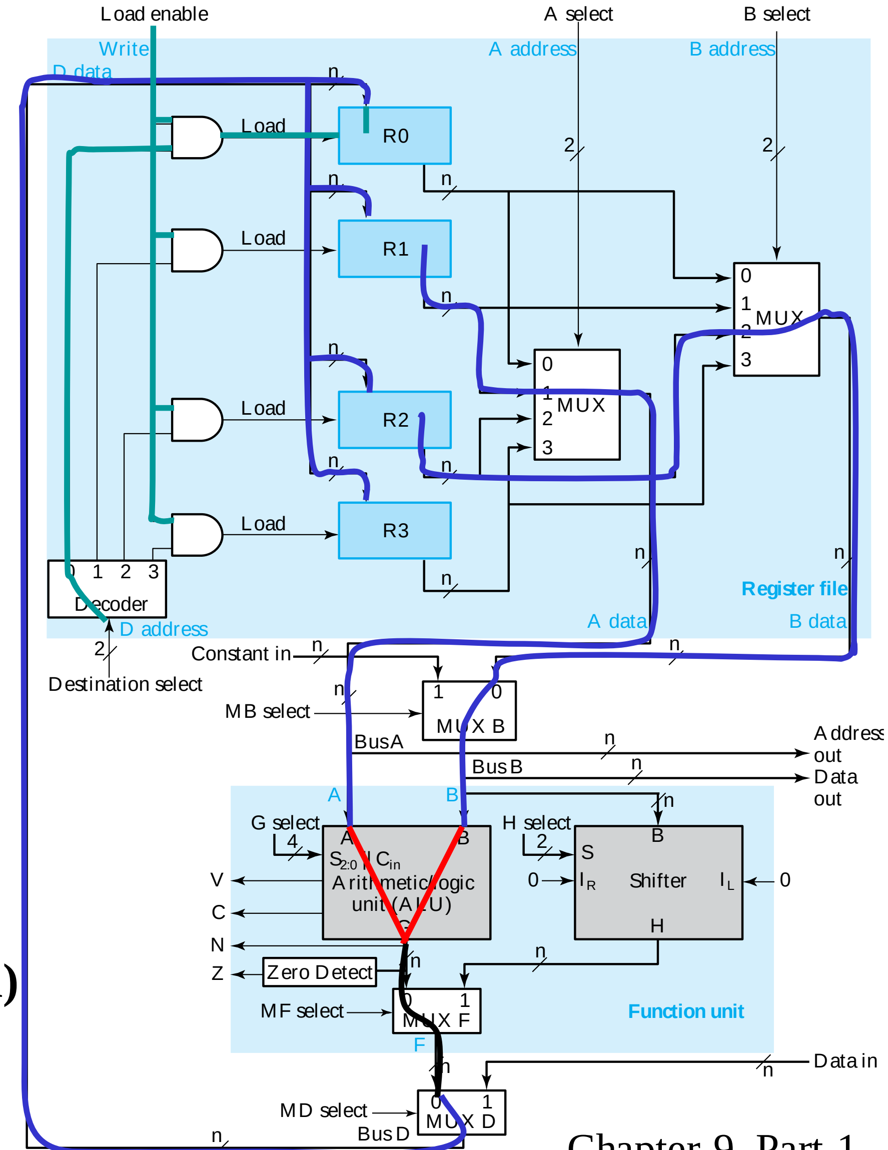
Datapath Example

- Four parallel-load registers
- Two mux-based register selectors
- Register destination decoder
- Mux B for external constant input
- Buses A and B with external address and data outputs
- ALU and Shifter with Mux F for output select
- Mux D for external data input
- Logic for generating status bits V, C, N, Z



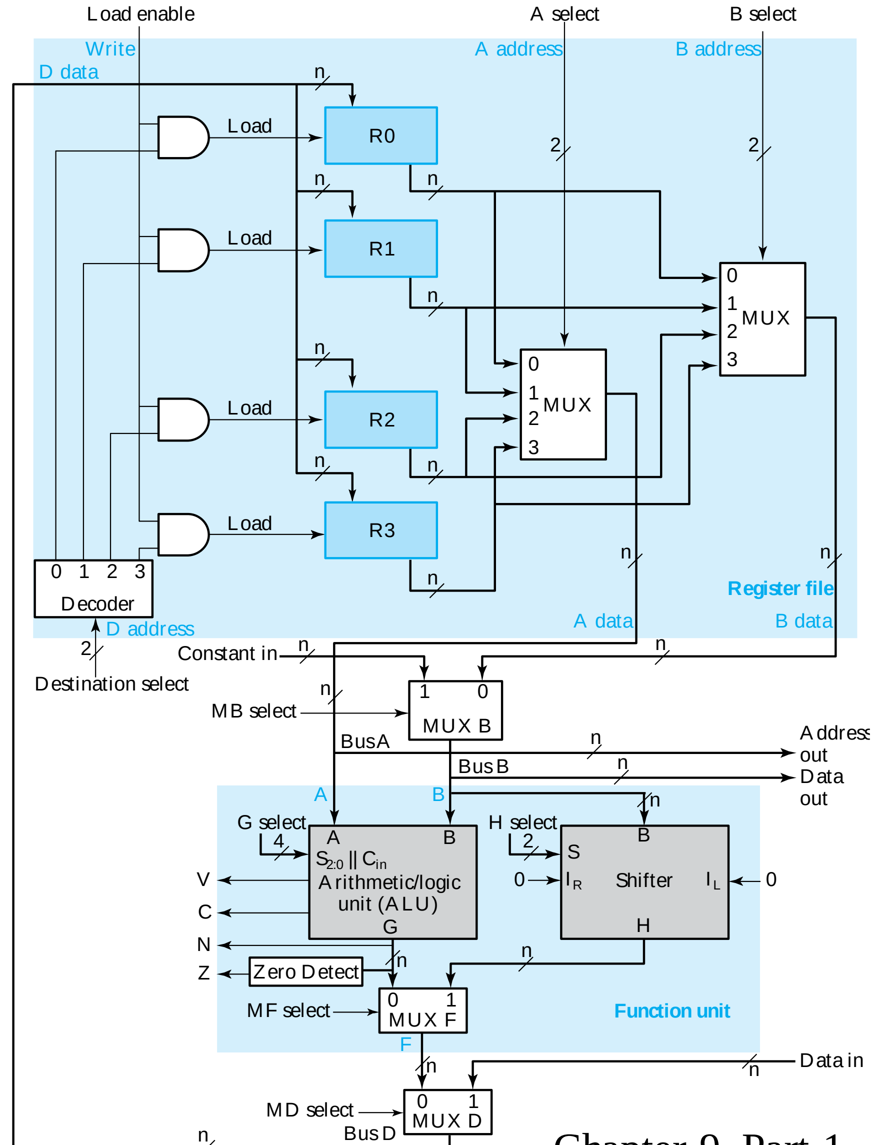
Datapath Example: Performing a Microoperation

- Microoperation: R0 ← R1 + R2
- Apply 01 to A select to place contents of R1 onto Bus A
- Apply 10 to B select to place contents of R2 onto B data and apply 0 to MB select to place B data on Bus B
- Apply 0010 to G select to perform addition G = Bus A + Bus B
- Apply 0 to MF select and 0 to MD select to place the value of G onto BUS D
- Apply 00 to Destination select to enable the Load input to R0
- Apply 1 to Load Enable to force the Load input to R0 to 1 so that R0 is loaded on the clock pulse (not shown)
- The overall microoperation requires 1 clock cycle



Datapath Example: Key Control Actions for Microoperation Alternatives

- Perform a shift microoperation – apply 1 to MF select
- Use a constant in a micro-operation using Bus B – apply 1 to MB select
- Provide an address and data for a memory or output write microoperation – apply 0 to Load enable to prevent register loading
- Provide an address and obtain data for a memory or output read microoperation – apply 1 to MD select
- For some of the above, other control signals become don't cares



Arithmetic Logic Unit (ALU)

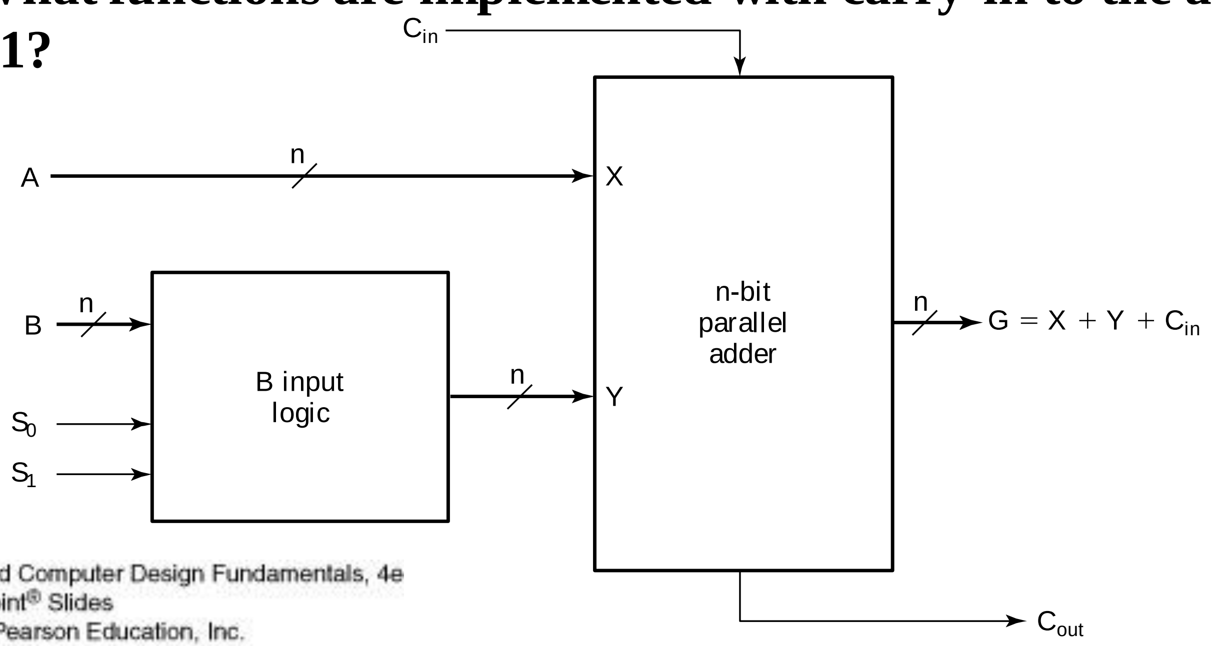
- In this and the next section, we deal with detailed design of typical ALUs and shifters
- Decompose the ALU into:
  - An arithmetic circuit
  - A logic circuit
  - A selector to pick between the two circuits
- Arithmetic circuit design
  - Decompose the arithmetic circuit into:
    - An n-bit parallel adder
    - A block of logic that selects four choices for the B input to the adder
    - See next slide for diagram

Arithmetic Circuit Design (continued)

- **There are only four functions of B to select as Y in G = A + Y:**

	$C_{in} = 0$	$C_{in} = 1$
• 0	G = A	G = A + 1
• B	G = A + B	G = A + B + 1
• $\overline{B}$	G = A + $\overline{B}$	G = A + $\overline{B}$ + 1
• 1	G = A – 1	G = A

- **What functions are implemented with carry-in to the adder = 0? =1?**



Arithmetic Circuit Design (continued)

- **Adding selection codes to the functions of B:**

TABLE 9-1
Function Table for Arithmetic Circuit

Select		Input	$G = (A + Y + C_{in})$	
S_1	S_0	Y	$C_{in} = 0$	$C_{in} = 1$
0	0	all 0s	$G = A$ (transfer)	$G = A + 1$ (increment)
0	1	B	$G = A + B$ (add)	$G = A + B + 1$
1	0	$\overline{B}$	$G = A + \overline{B}$	$G = A + \overline{B} + 1$ (subtract)
1	1	all 1s	$G = A - 1$ (decrement)	$G = A$ (transfer)

- **The useful arithmetic functions are labeled in the table**
- **Note that all four functions of B produce at least one useful function**

Logic Circuit

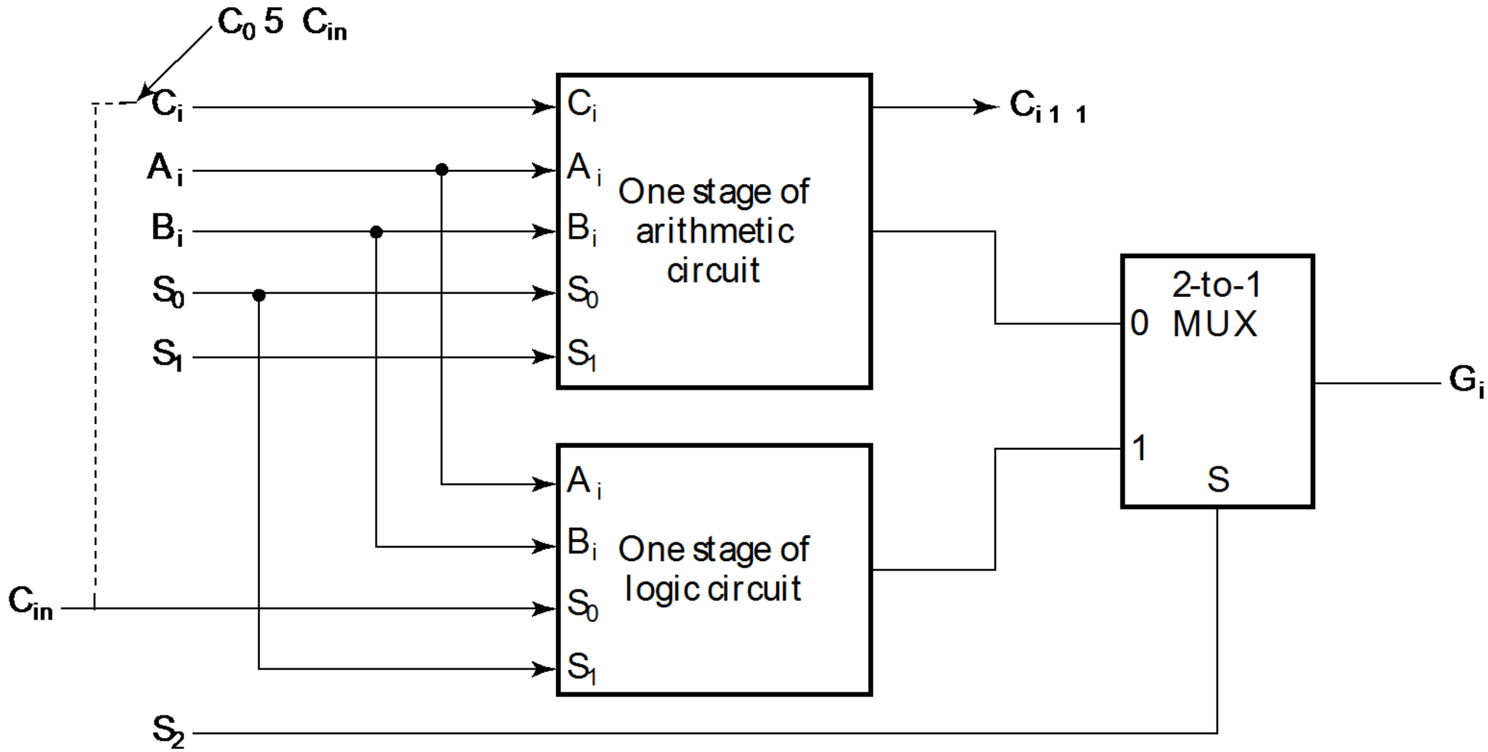
- The text gives a circuit implemented using a multiplexer plus gates implementing: AND, OR, XOR and NOT
- Here we custom design a circuit for bit G_i by beginning with a truth table organized as a K-map and assigning (S1, S0) codes to AND, OR, etc.
- $G_i = S_0 \overline{A_i} B_i + \overline{S_1} A_i B_i + S_0 A_i \overline{B_i} + S_1 \overline{S_0} \overline{A_i}$
- Gate input count for MUX solution > 29
- Gate input count for above circuit < 20
- Custom design better

S_1S_0	AND	OR	XOR	NOT
A_iB_i	0 0	0 1	1 1	1 0
0 0	0	0	0	1
0 1	0	1	1	1
1 1	1	1	0	0
1 0	0	1	1	0

Arithmetic Logic Unit (ALU)

- The custom circuit has interchanged the (S_1,S_0) codes for XOR and NOT compared to the MUX circuit. To preserve compatibility with the text, we use the MUX solution.
- Next, use the arithmetic circuit, the logic circuit, and a 2-way multiplexer to form the ALU. See the next slide for the bit slice diagram.
- The input connections to the arithmetic circuit and logic circuit have been been assigned to prepare for seamless addition of the shifter, keeping the selection codes for the combined ALU and the shifter at 4 bits:
  - Carry-in C_i and Carry-out C_{i+1} go between bits
  - A_i and B_i are connected to both units
  - A new signal S_2 performs the arithmetic/logic selection
  - The select signal entering the LSB of the arithmetic circuit, C_{in}, is connected to the least significant selection input for the logic circuit, S_0.

Arithmetic Logic Unit (ALU) (continued)

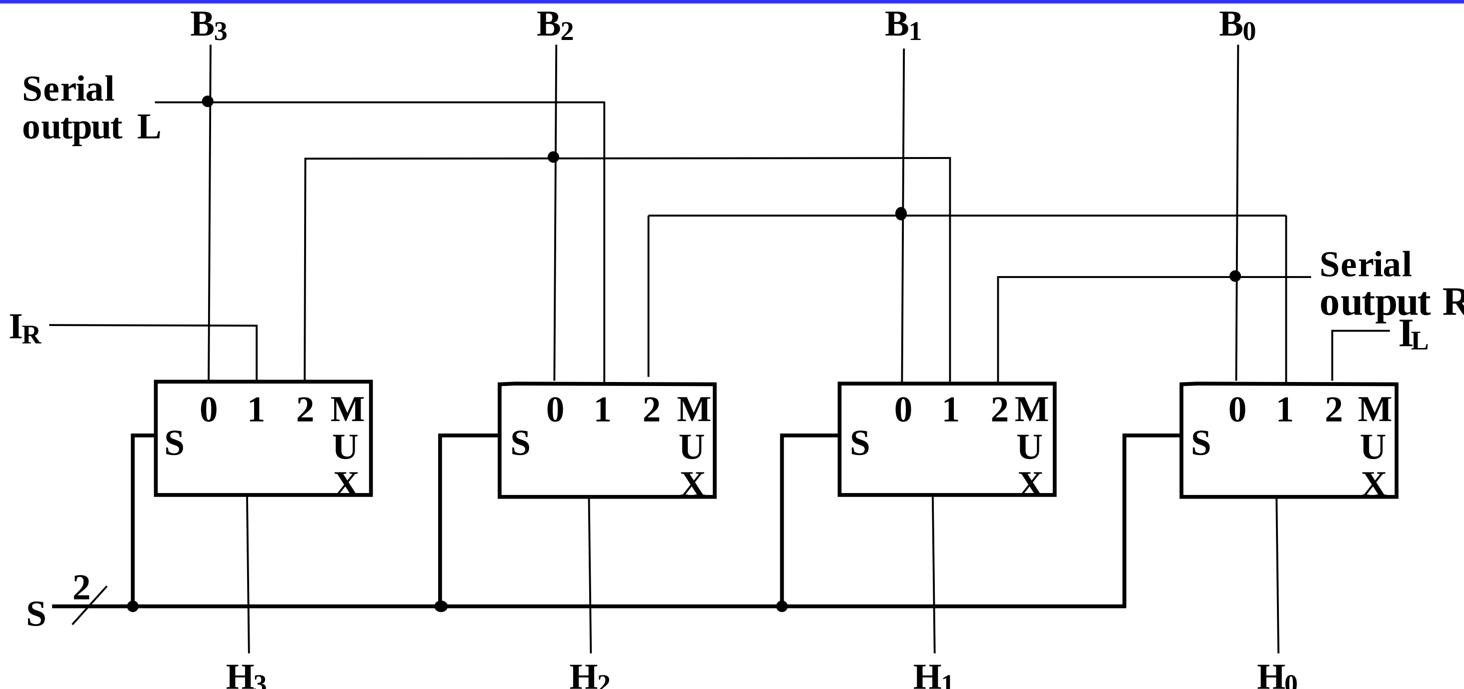


- The next most significant select signals, S0 for the arithmetic circuit and S1 for the logic circuit, are wired together, completing the two select signals for the logic circuit.
- The remaining S1 completes the three select signals for the arithmetic circuit.

Combinational Shifter Parameters

- Direction: Left, Right
- Number of positions with examples:
  - Single bit:
    - 1 position
    - 0 and 1 positions
  - Multiple bit:
    - 1 to n – 1 positions
    - 0 to n – 1 positions
- Filling of vacant positions
  - Many options depending on instruction set
  - Here, will provide input lines or zero fill

4-Bit Basic Left/Right Shifter

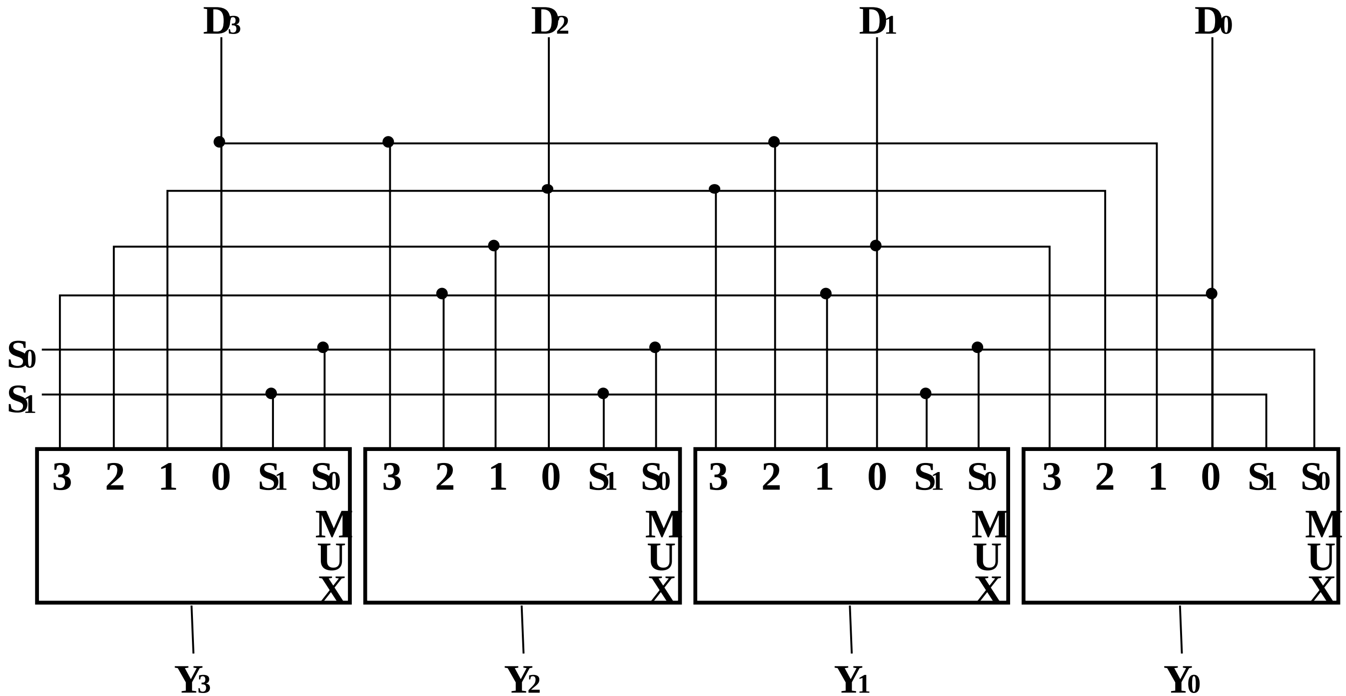


- **Serial Inputs:**
  - **I_R for right shift**
  - **I_L for left shift**
- **Serial Outputs**
  - **R for right shift (Same as MSB input)**
  - **L for left shift (Same as LSB input)**

- **Shift Functions:**

(S_1, S_0) = 00 Pass B unchanged
01 Right shift
10 Left shift
11 Unused

Barrel Shifter



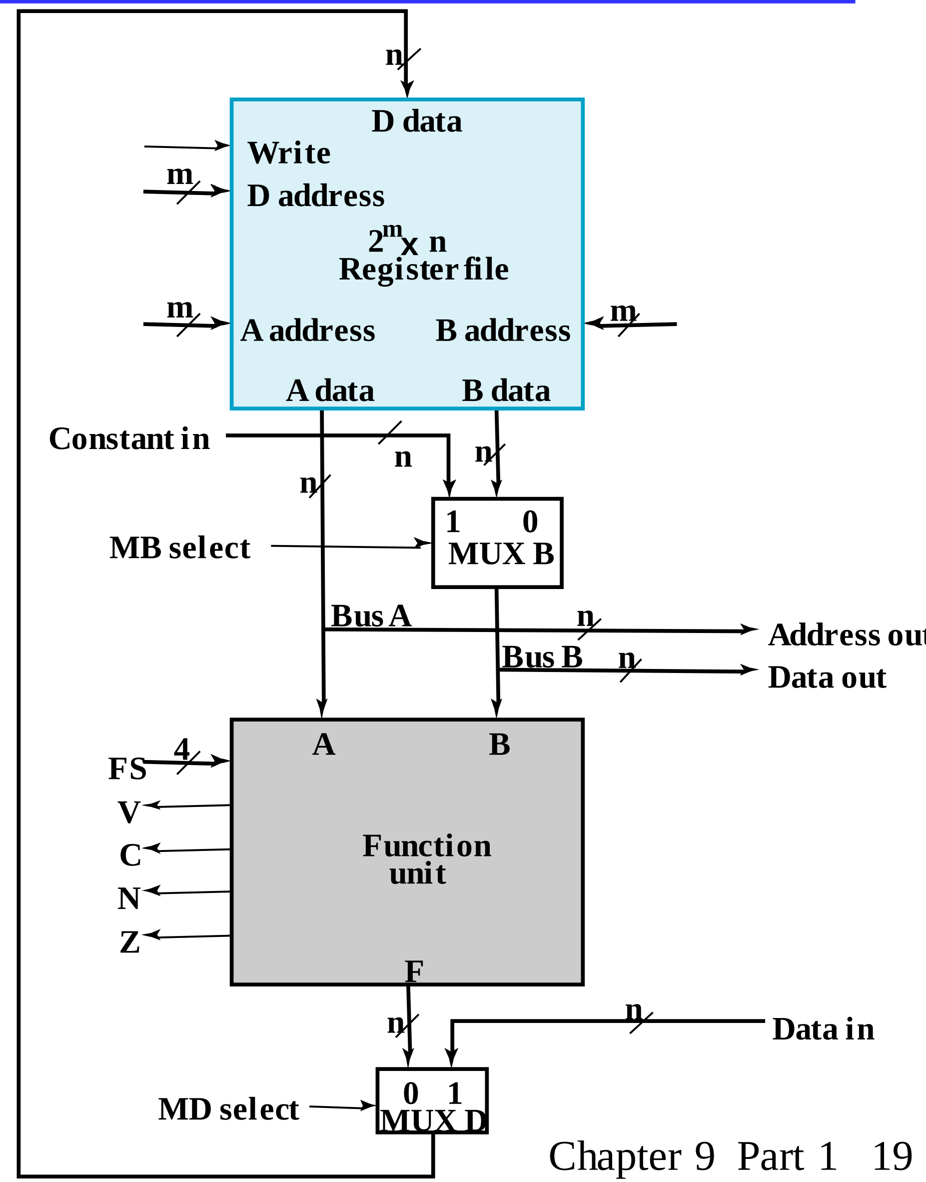
- A rotate is a shift in which the bits shifted out are inserted into the positions vacated
- The circuit rotates its contents left from 0 to 3 positions depending on S:
  S = 00 position unchanged S = 10 rotate left by 2 positions
  S = 01 rotate left by 1 positions S = 11 rotate left by 3 positions
- See Table 10-3 in text for details

Barrel Shifter (continued)

- Large barrel shifters can be constructed by using:
  - Layers of multiplexers - Example 64-bit:
    - Layer 1 shifts by 0, 16, 32, 48
    - Layer 2 shifts by 0, 4, 8, 12
    - Layer 3 shifts by 0, 1, 2, 3
    - See example in section 12-2 of the text
  - 2 - dimensional array circuits designed at the electronic level

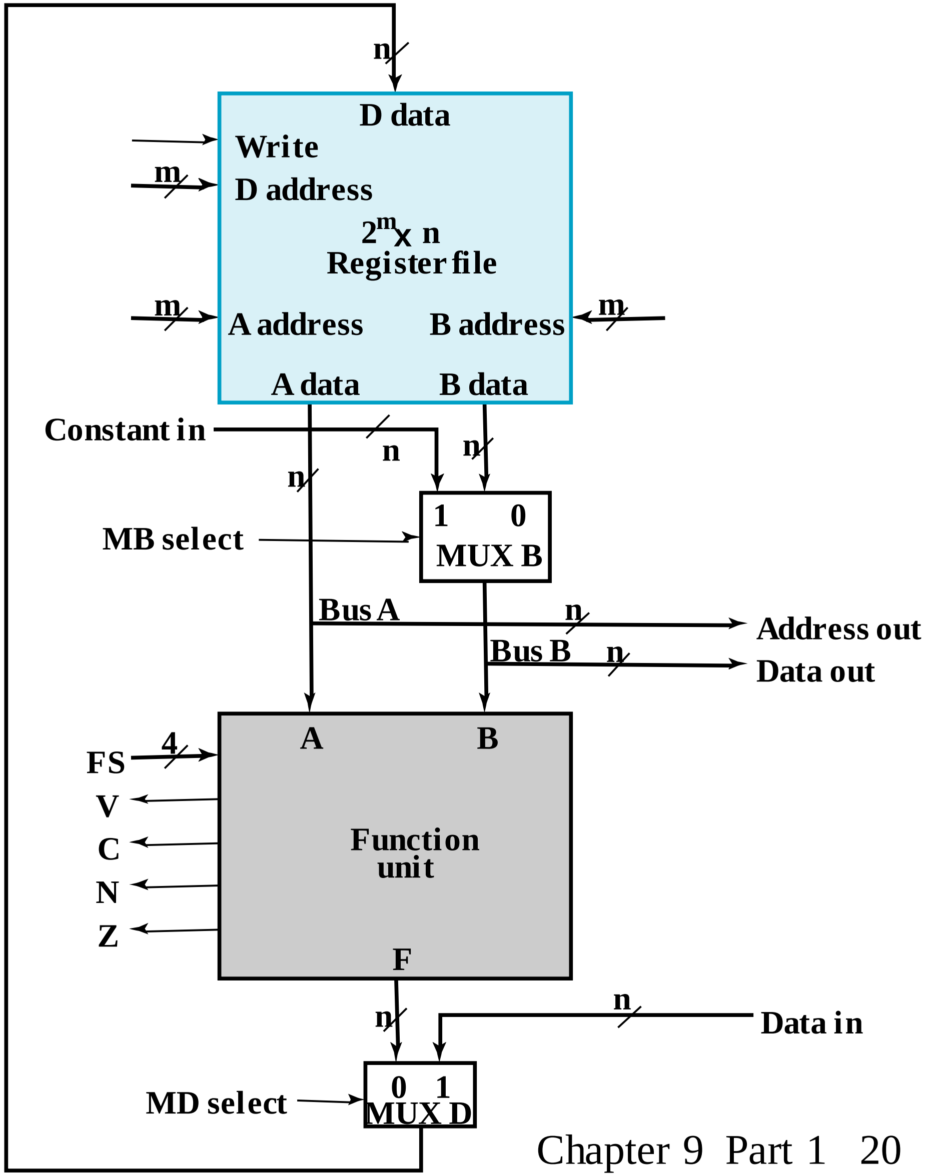
Datapath Representation

- Have looked at detailed design of ALU and shifter in the datapath in slide 8
- Here we move up one level in the hierarchy from that datapath
- The registers, and the multiplexer, decoder, and enable hardware for accessing them become a *register file*
- The ALU, shifter, Mux F and status hardware become a *function unit*
- The remaining muxes and buses which handle data transfers are at the new level of the hierarchy



Datapath Representation (continued)

- In the register file:
  - Multiplexer select inputs become A address and B address
  - Decoder input becomes D address
  - Multiplexer outputs become A data and B data
  - Input data to the registers becomes D data
  - Load enable becomes write
- The register file now appears like a memory based on clocked flip-flops (the clock is not shown)
- The function unit labeling is quite straightforward except for FS



Definition of Function Unit Select (FS) Codes

FS(3:0)	MF Select	G Select(3:0)	H Select(3:0)	Microoperation
0000	0	0000	XX	$F \leftarrow A$
0001	0	0001	XX	$F \leftarrow A + 1$
0010	0	0010	XX	$F \leftarrow A + B$
0011	0	0011	XX	$F \leftarrow A + B + 1$
0100	0	0100	XX	$F \leftarrow A + \overline{B}$
0101	0	0101	XX	$F \leftarrow A + \overline{B} + 1$
0110	0	0110	XX	$F \leftarrow A - 1$
0111	0	0111	XX	$F \leftarrow A$
1000	0	1 X00	XX	$F \leftarrow A \wedge B$
1001	0	1 X01	XX	$F \leftarrow A \vee B$
1010	0	1 X10	XX	$F \leftarrow A \oplus B$
1011	0	1 X11	XX	$F \leftarrow \overline{A}$
1100	1	XXXX	00	$F \leftarrow B$
1101	1	XXXX	01	$F \leftarrow \mathrm{sr}\ B$
1110	1	XXXX	10	$F \leftarrow \mathrm{sl}\ B$

Boolean Equations:

$MFS = F_3 F_2$

$GS_i = F_i$

$HS_i = F_i$

The Control Word

- The datapath has many control inputs
- The signals driving these inputs can be defined and organized into a *control word*
- To execute a microinstruction, we apply control word values for a clock cycle. For most microoperations, the positive edge of the clock cycle is needed to perform the register load
- The datapath control word format and the field definitions are shown on the next slide

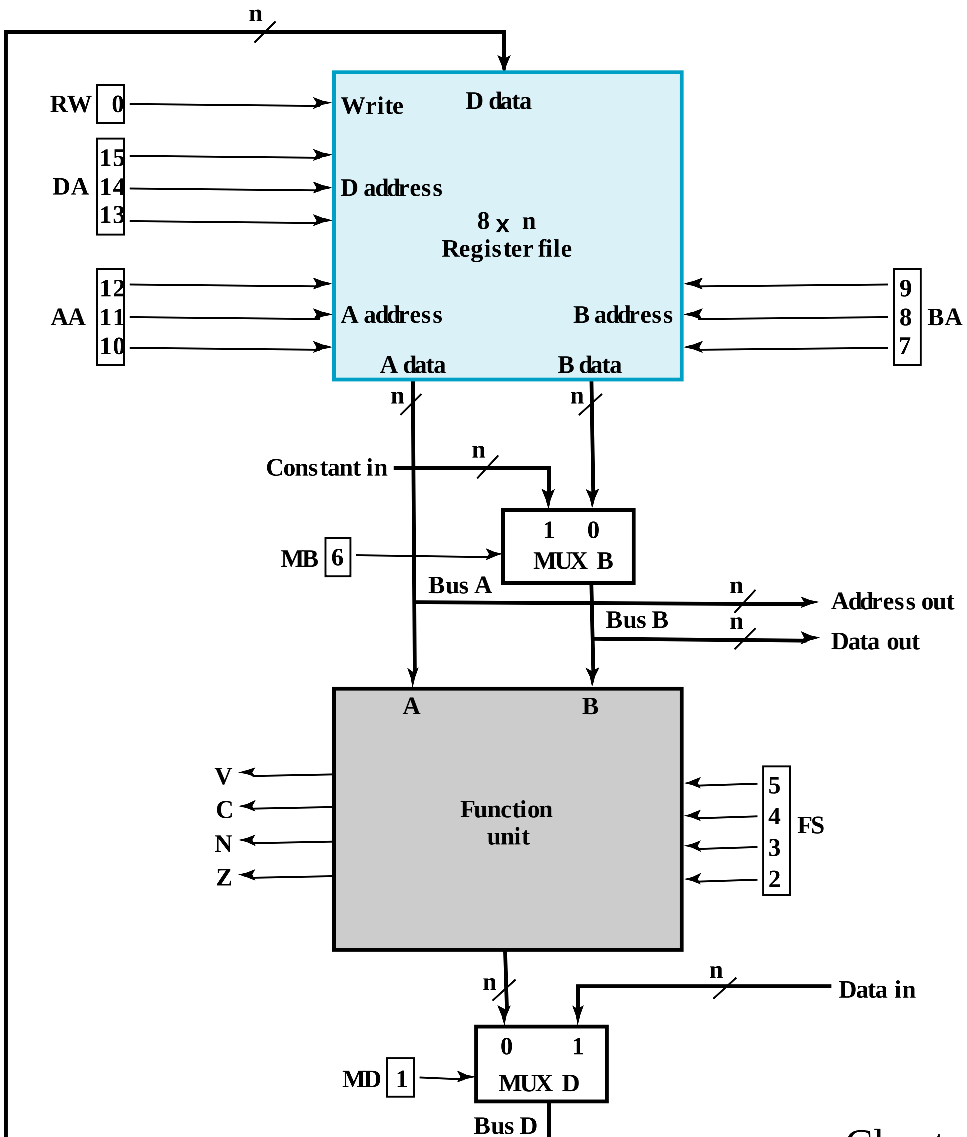
The Control Word Fields

15 14 13	12 11 10	9 8 7	6	5 4 3 2	1	0
DA	AA	BA	MB	FS	MD	RW

Control word

- **Fields**
  - DA – D Address
  - AA – A Address
  - BA – B Address
  - MB – Mux B
  - FS – Function Select
  - MD – Mux D
  - RW – Register Write
- **The connections to datapath are shown in the next slide**

Control Word Block Diagram



Control Word Encoding

DA, AA, BA		MB		FS		MD		RW	
Function	Code	Function	Code	Function	Code	Function	Code	Function	Code
$R0$	000	Register	0	$F \leftarrow A$	0000	Function	0	No write	0
$R1$	001	Constant	1	$F \leftarrow A + 1$	0001	Data In	1	Write	1
$R2$	010			$F \leftarrow A + B$	0010				
$R3$	011			$F \leftarrow A + B + 1$	0011				
$R4$	100			$F \leftarrow A + \overline{B}$	0100				
$R5$	101			$F \leftarrow A + \overline{B} + 1$	0101				
$R6$	110			$F \leftarrow A - 1$	0110				
$R7$	111			$F \leftarrow A$	0111				
				$F \leftarrow A \wedge B$	1000				
				$F \leftarrow A \vee B$	1001				
				$F \leftarrow A \oplus B$	1010				
				$F \leftarrow \overline{A}$	1011				
				$F \leftarrow B$	1100				
				$F \leftarrow \text{sr } B$	1101				
				$F \leftarrow \text{sl } B$	1110				

Microoperations for the Datapath - Symbolic Representation

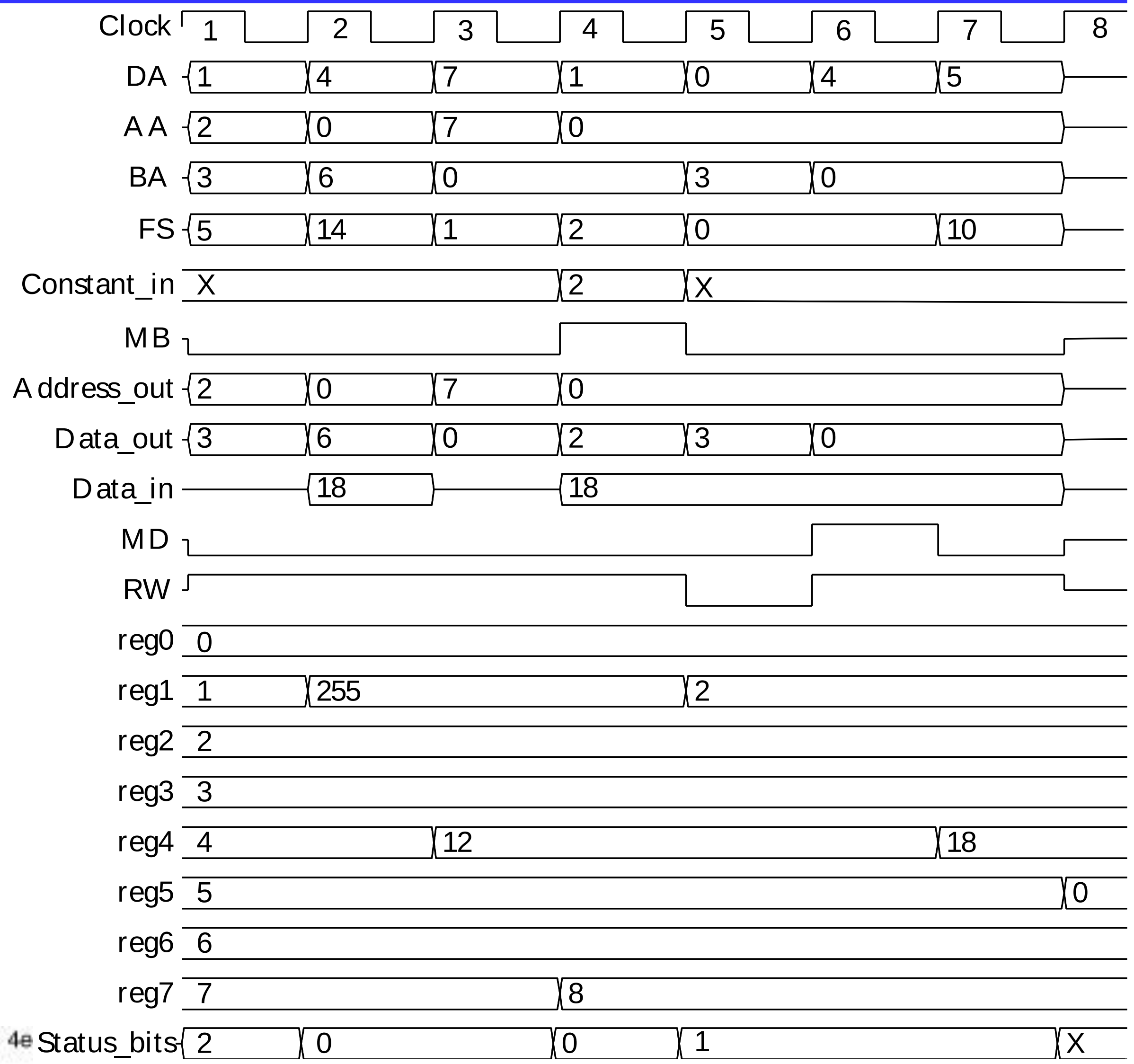
Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	$R1$	$R2$	$R3$	Register	$F = A + \overline{B} + 1$	Function	Write
$R4 \leftarrow \text{sl R6}$	$R4$	—	$R6$	Register	$F = \text{sl } B$	Function	Write
$R7 \leftarrow R7 + 1$	$R7$	$R7$	—	Register	$F = A + 1$	Function	Write
$R1 \leftarrow R0 + 2$	$R1$	$R0$	—	Constant	$F = A + B$	Function	Write
Data out $\leftarrow R3$	—		$R3$	Register	—	—	No Write
$R4 \leftarrow$ Data in	$R4$	—		—	—	Data in	Write
$R5 \leftarrow 0$	$R5$	$R0$	$R0$	Register	$F = A \oplus B$	Function	Write

Microoperations for the Datapath - Binary Representation

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	001	010	011	0	0101	0	1
$R4 \leftarrow \mathrm{sl}\ R6$	100	XXX	110	0	1110	0	1
$R7 \leftarrow R7 + 1$	111	111	XXX	0	0001	0	1
$R1 \leftarrow R0 + 2$	001	000	XXX	1	0010	0	1
Data out $\leftarrow R3$	XXX	XXX	011	0	XXXX	X	0
$R4 \leftarrow$ Data in	100	XXX	XXX	X	XXXX	1	1
$R5 \leftarrow 0$	101	000	000	0	1010	0	1

- **Results of simulation of the above on the next slide**

Datapath Simulation



Terms of Use