

# RT Description of the Datapath

$RW: R[DA] \leftarrow$  If MD then  
FS( $R[AA]$ , if MB then  $R[BA]$   
else Constant in)  
else DATA in,

$$V \leftarrow C_n \oplus C_{n-1},$$

$$C \leftarrow C_n,$$

$$Z \leftarrow R_{n-1}, R_{n-2}, \dots, R_0,$$

$$N \leftarrow R_{n-1},$$

# Symbolic Notation for Micro-ops

- ▶ Because human beings working in binary code tend to be highly error-prone, we usually employ intuitive symbols to specify datapath micro-ops.
- ▶ Typical **symbol/code** assignments are:

# Symbol-binary Map of Control Word Fields

DA, AA, BA		MB		FS	
Function	Code	Function	Code	Function	Code
R0	00000	Register	0	$G = A$	00000
R1	00001	Constant	1	$G = A + 1$	00001
R2	00010			$G = A + B$	00010
R3	00011	MD		$G = A + B + 1$	00011
R4	00100	Function	Code	$G = A + \overline{B}$	00100
R5	00101	Function	0	$G = A + \overline{B} + 1$	00101
R6	00110	Data In	1	$G = A - 1$	00110
R7	00111			$G = A$	00111
continue		RW		$G = A \wedge B$	01000
R31	11111	Function	Code	$G = A \vee B$	01010
		No Write	0	$G = A \oplus B$	01100
		Write	1	$G = \overline{A}$	01110
				$G = B$	10000
				$G = sr B$	10100
				$G = sl B$	11000

# Symbol Conversion

▶ With the symbolic notation it is easy to accurately specify control words which may then be automatically converted to binary.

▶ For example:  $R1 \leftarrow R2 + \overline{R3} + 1$

8 Register Example

Field:	DA	AA	BA	MB	FS	MD	RW
Symbol:	R1	R2	R3	Register	$F = A + \overline{B} + 1$	Function	Write
Binary:	001	010	011	0	00101	0	1



# CS2022 Microoperations Example

▶ DIY - Convert these to binary and check your results against the table on the next slide.

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 + \overline{R3} + 1$	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
$R4 \leftarrow \text{Data in}$	R4	—	—	—	—	Data in	Write
$R5 \leftarrow 0$	R5	R0	R0	Register	$F = A \oplus B$	Function	Write

# Binary Control Words from Example

## 8 Register Example

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	001	010	011	0	00101	0	1
$R4 \leftarrow sl\ R6$	100	000	110	0	11000	0	1
$R7 \leftarrow R7 + 1$	111	111	000	0	00001	0	1
$R1 \leftarrow R0 + 2$	001	000	000	1	00010	0	1
Data out $\leftarrow R3$	000	000	011	0	00000	0	0
$R4 \leftarrow$ Data in	100	000	000	0	00000	1	1
$R5 \leftarrow 0$	101	000	000	0	01100	0	1



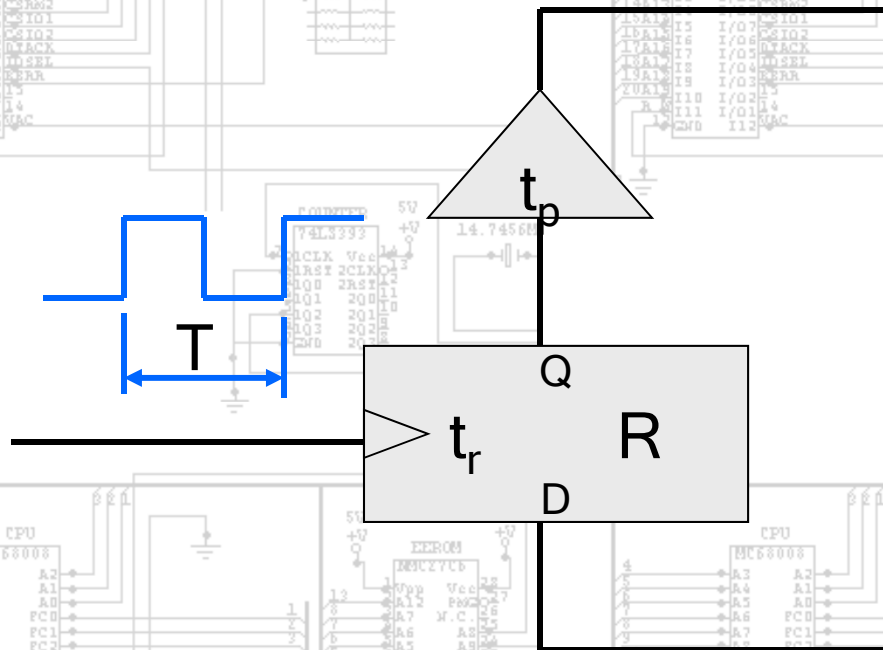
# Status Bits

- ▶ Status bits shows the input for the Zero, Negative, Carry-out and Overflow bits respectively.
- ▶ Hence the change following a control word change.



# Datapath Timing

▶ The total, worst case propagation delay determines the maximum rate at which we may clock a system.

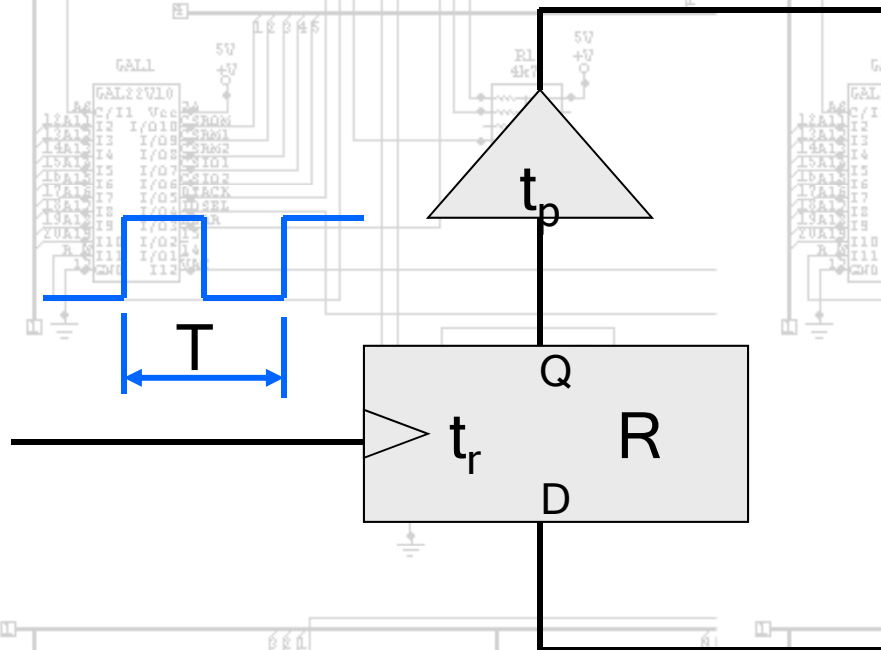


Combinational logic with delay  $t_p$

Register with delay  $t_r$

# Timing

► For successful operations we must have:



$$T \geq t_p + t_r$$

$$T_{\min} = t_p + t_r$$

$$f_{\max} = 1 / T_{\min}$$

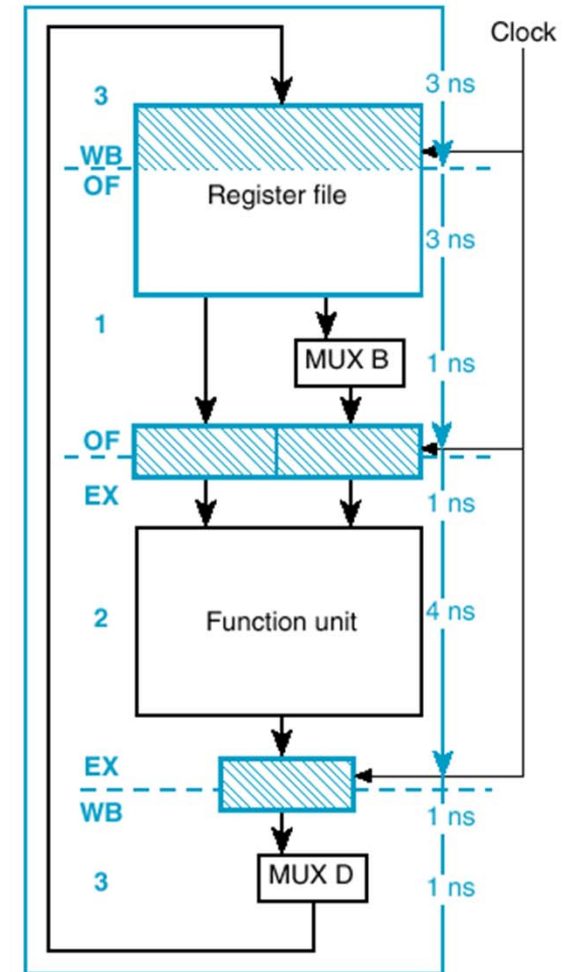
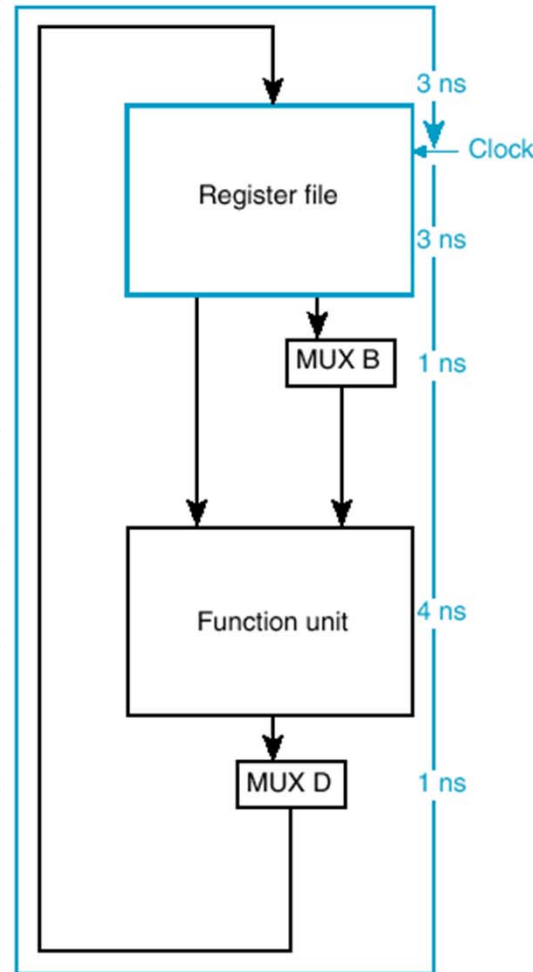
# Conventional and Pipelined Datapath Timing

$$t_r = 1\text{ns}$$

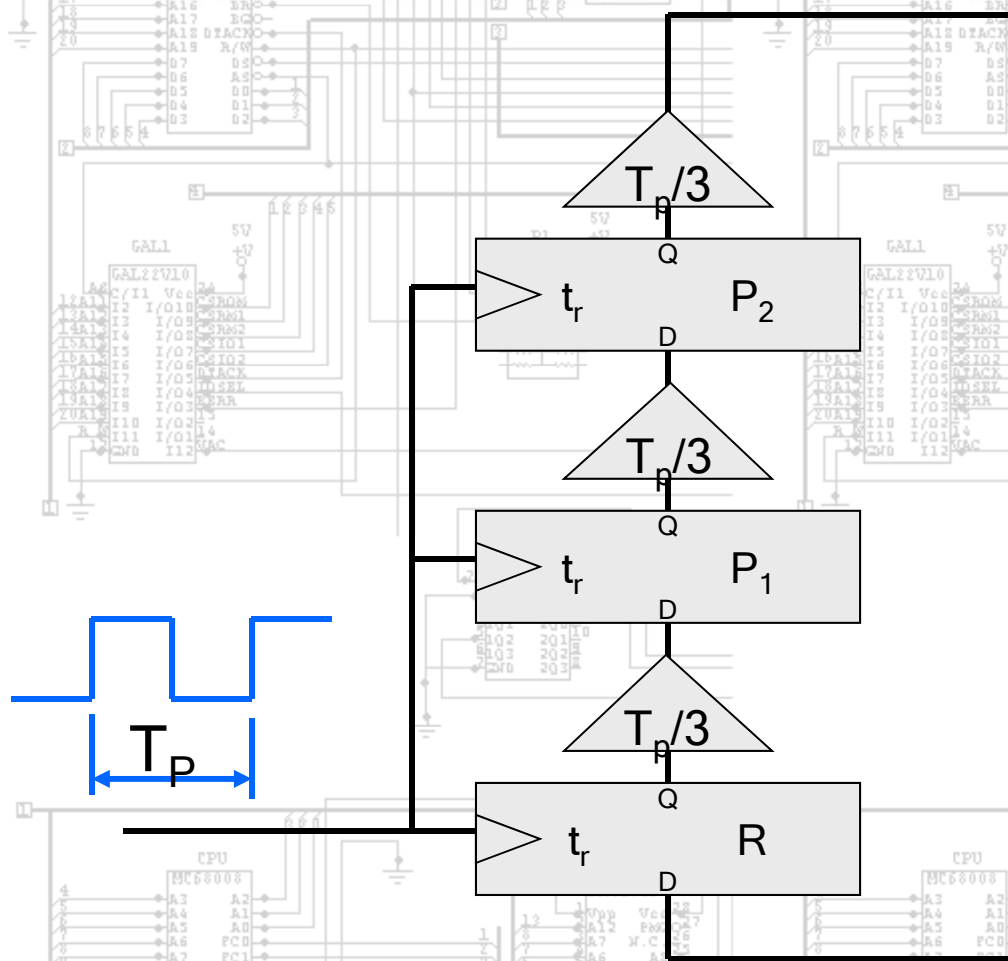
$$t_p = 3 + 1 + 4 + 1 + 2\text{ns} = 11\text{ns}$$

$$T_{\min} = 12\text{ns}$$

$$f_{\max} = 1 / T_{\min} = 83.3\text{MHz}$$



# Three-Stage Pipeline



- ▶ To speed things up we can introduce register into the combinational logic.

$$T_p \geq t_p/3 + t_r$$

$$T_{\max} = 1 + 4ns$$

$$f_{\max} = 1 / T_{\max} = 200MHz$$



# Pipelined Datapath

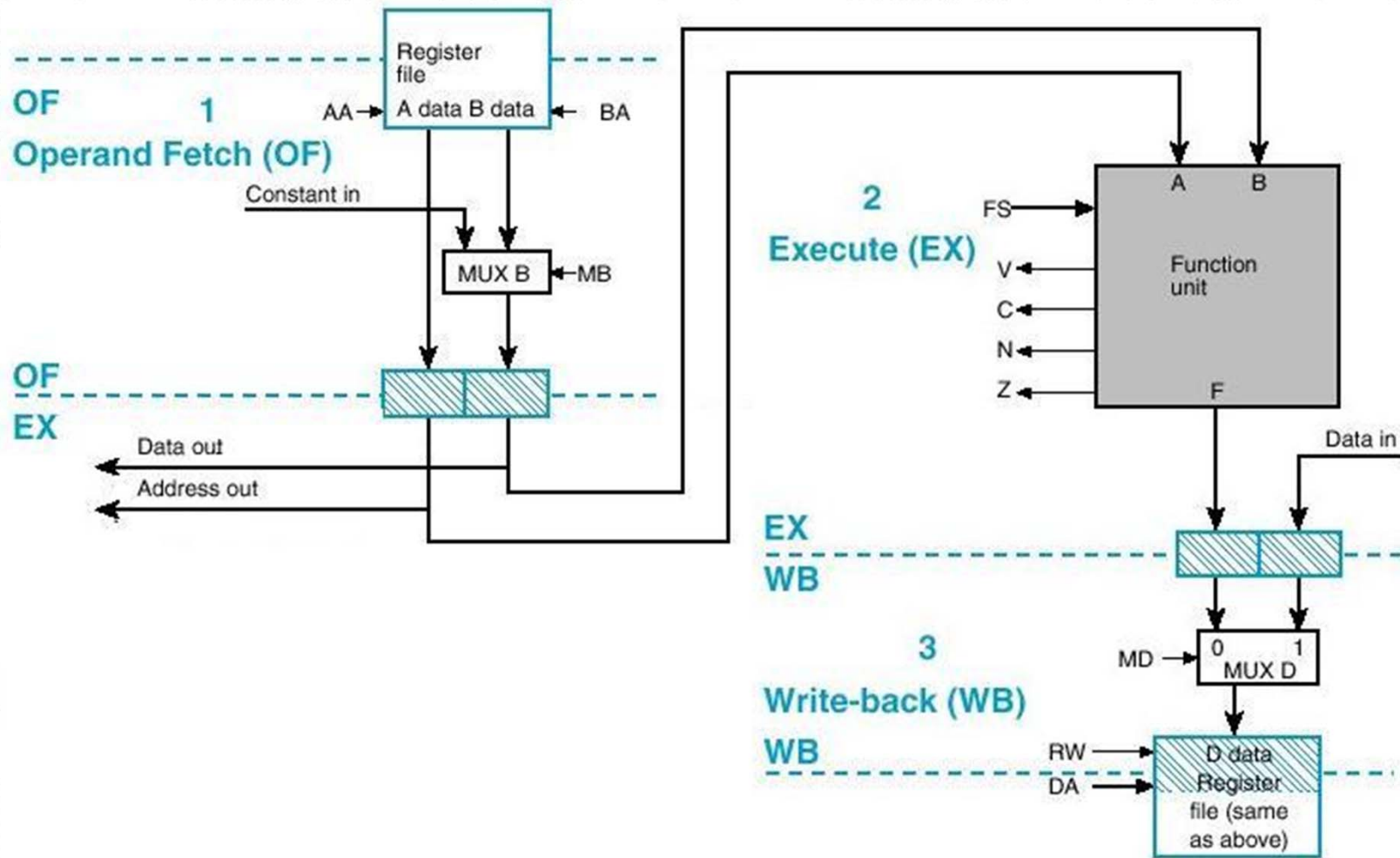
▶ When a datapath is pipelined it divides naturally into stages seen on the next slide.

▶ OF = Operand Fetch

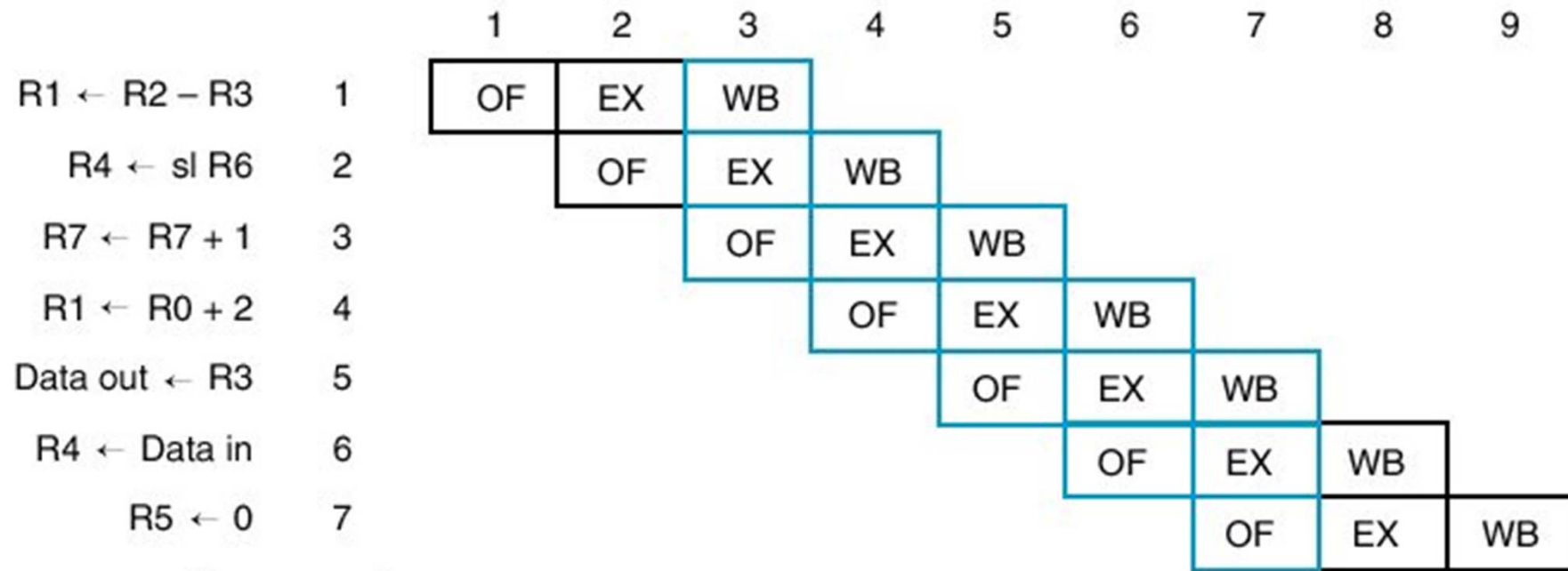
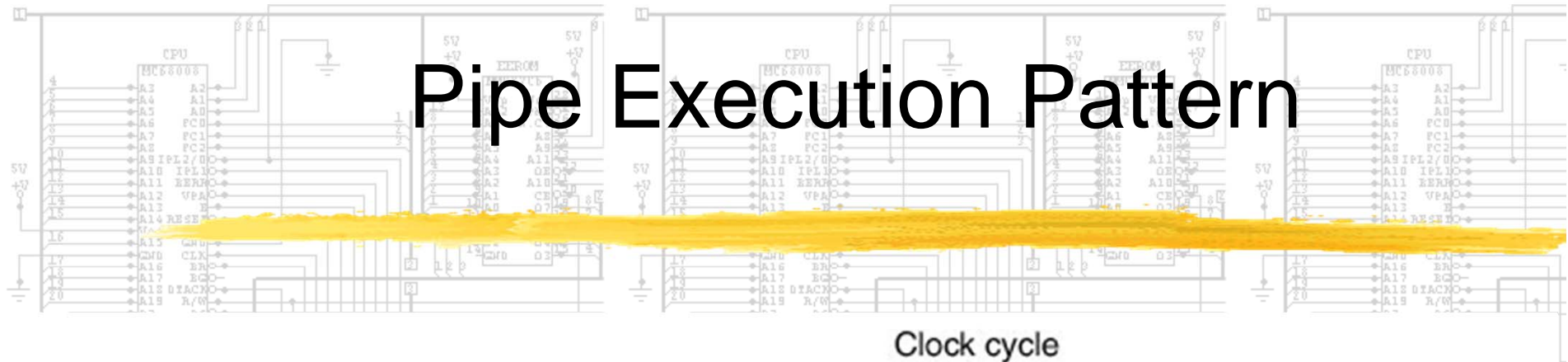
▶ EX = Execute

▶ WB = Write Back

# Pipelined Datapath Schematic



# Pipe Execution Pattern



Microoperation

