RT Description of the Datapath

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

V ¬ C_n Å C_{n-1},
C ¬ C_n,
Z ¬ R_{n-1},R_{n-2},…,R_0,
N ¬ 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	
Function	Code
R0	000
R1	001
R2	010
R3	011
R4	100
R5	101
R6	110
R7	111

MB	
Function	Code
Register	0
Constant	1

MD	
Function	Code
Function	0
Data In	1

RW	
Function	Code
No Write	0
Write	1

FS	
Function	Code
G = A	00000
G = A + 1	00001
G = A + B	00010
G = A + B +1	00011
G = A + B-	00100
G = A + B +1	00101
G = A - 1	00110
G = A	00111
G = A ÙB	01000
G = A ÚB	01010
G = A ÅB	01100
G = 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←R2 + R3 +1

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

Microoperations Example

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

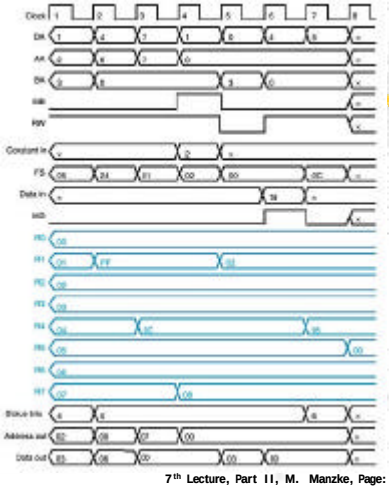
Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 + \overline{R3} + 1$	R1	R2	R3	Register	$F = A + \overline{B} + 1$	Function	Write
$R4 \leftarrow$ sl R6	R4	—	R6	Register	$F =$ sl B	Function	Write
$R7 \leftarrow R7 + 1$	R7	R7	—	Register	$F = A + 1$	Function	Write
$R1 \leftarrow R0 + 2$	R1	R0	—	Constant	$F = A + B$	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

Binary Control Words from 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

Wave

- Examine the figure:
  - Register transfer on Clock ↑.
  - R0 – R7 are initialised to $R_i \leftarrow i$.

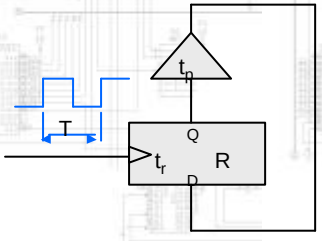


Status Bits

- 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.

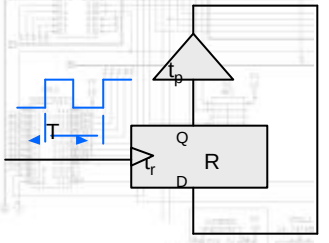


Combinational logic with delay t_p

Register with delay t_r

Timing

- For successful operations we must have:



$$T \geq t_p + t_r$$

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

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

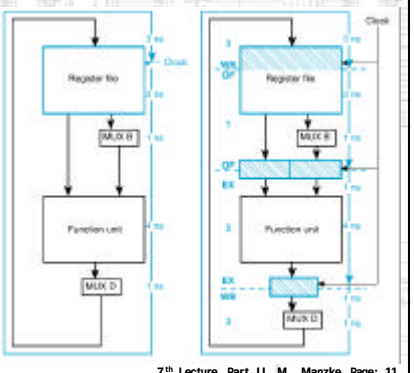
Conventional and Pipelined Datapath timing

t_r = 1ns

t_p = 3+1+4+1+2ns
= 11ns

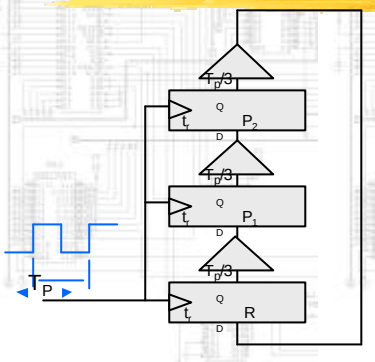
T_{min} = 12ns

f_{max} = 1 / T_{min}
= 83.3MHz



Three-Stage Pipeline

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



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

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

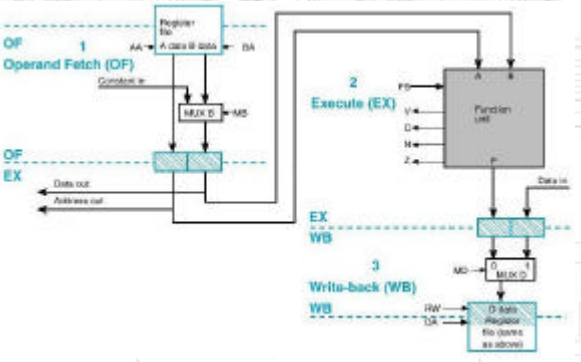
$$f_{max} = 1 / T_{max}$$

$$= 200MHz$$

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



Overlapped Execution

- This leads to overlapped execution of micro-ops, yielding an increased throughput.
  - See next slide.
  - DO PROBLEMS 25-29

Pipe Execution Pattern