ECE290 Fall 2012
Lecture 17

Dr. Zbigniew Kalbarczyk

Today

- Register Transfer, RTL
- Datapath and control unit

Register Transfer Language

- Moving data from one register to another
- Register transfer language (RTL) notation:

Symbol	Description	Examples
Letters	Register	PC, F, IR
Arrow	Data transfer	F <- A
Brackets, parenthesis	Register bits	F[3:0], F[1]
Brackets	Memory address	DR <- M[AR]
comma	Simultaneous transfers	F <- A, B <- R3

Micro-Operations Using RTL

Symbols	Operations
F=A+B	Addition
F=A’	1’s complement
F=A’+1	2’s complement
F=A+B’+1	subtraction
F=A+1	Increment register
F=A-1	Decrement register
F=A’	Bitwise NOT
F=A $\wedge$ B	Bitwise AND
F=A $\vee$ B	Bitwise OR
F=A $\oplus$ B	Bitwise XOR
F=sl A	Shift left
F=sr A	Shift right

ARITHMETIC

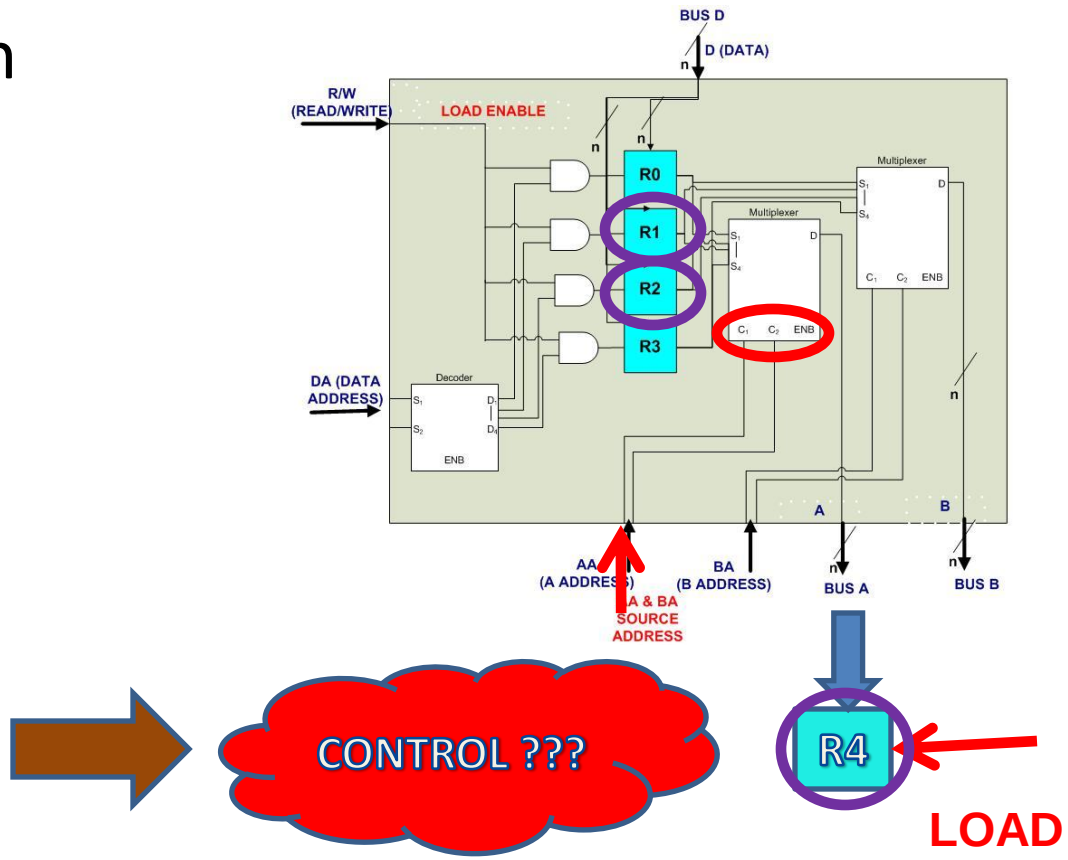
LOGIC

Example of Register Transfer

- How should we design control interfaces for register transfers?

Example:

- If (S1=1) then R4=R1
  - Else
    - If (S2=1) then R4=R2

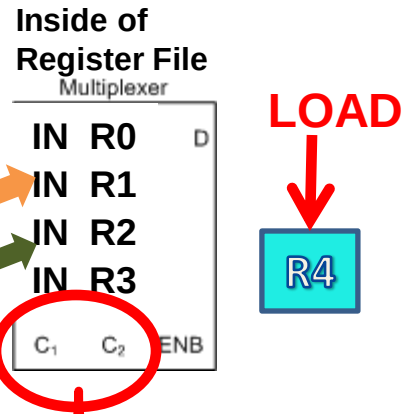


RED arrows - Control
BLUE arrow- Data
PURPLE – Register involved

Example of Register Transfer

- How should we design control interfaces for register transfers?

S1	S2	Operation
1	X	R4=R1
0	1	R4=R2



Example:

- If (S1=1) then R4=R1
  - Else
    - If (S2=1) then R4=R2

S1	S2	AA[C1]	AA[C2]	Load Enable for R4
1	X	1	0	1
0	1	0	1	1
0	0	X	X	0

Load Enable	S2 = 0	S2 = 1
S1 = 0	0	1
S1 = 1	1	1

$$AA[C1] = S1$$

$$AA[C2] = \bar{S}1\,S2$$

$$Load\ Enable = S1 + S2$$

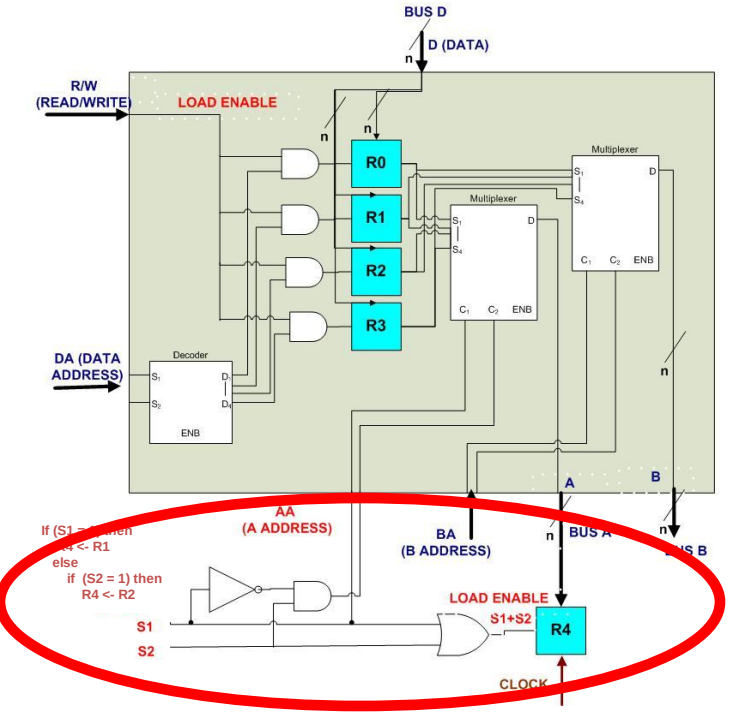
Example Control Interface for Register Transfer

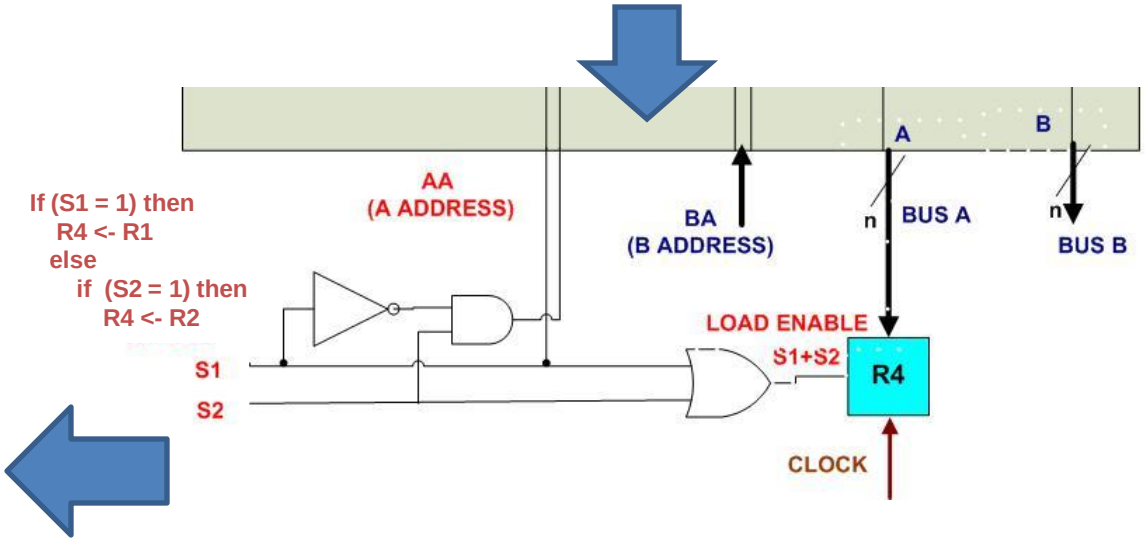
- If (S1=1) then R4=R1
  - Else
    - If (S2=1) then R4=R2

$$AA[C1] = S1$$

$$AA[C2] = \bar{S}1\,S2$$

$$Load\ Enable\ for\ R4 = S1 + S2$$





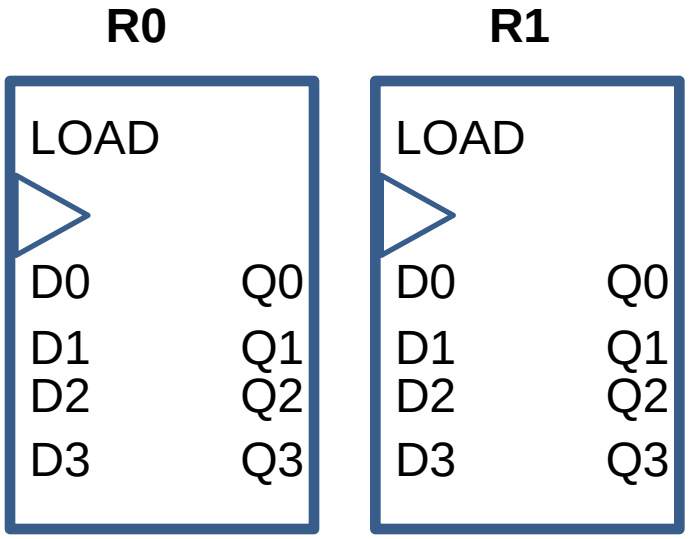
What if destination registry is inside of the same File Registry?

S0	S1	Operation
1	0	R1<- R1 ‘
0	1	R1<- R0

Example Register Transfer

- Given two 4-bit registers R0 & R1, implement the following transfers using two control signals S0 & S1, and AND, OR and NOT circuits
  - Control S0: R1 <- R1’
  - Control S1: R1 <- R0

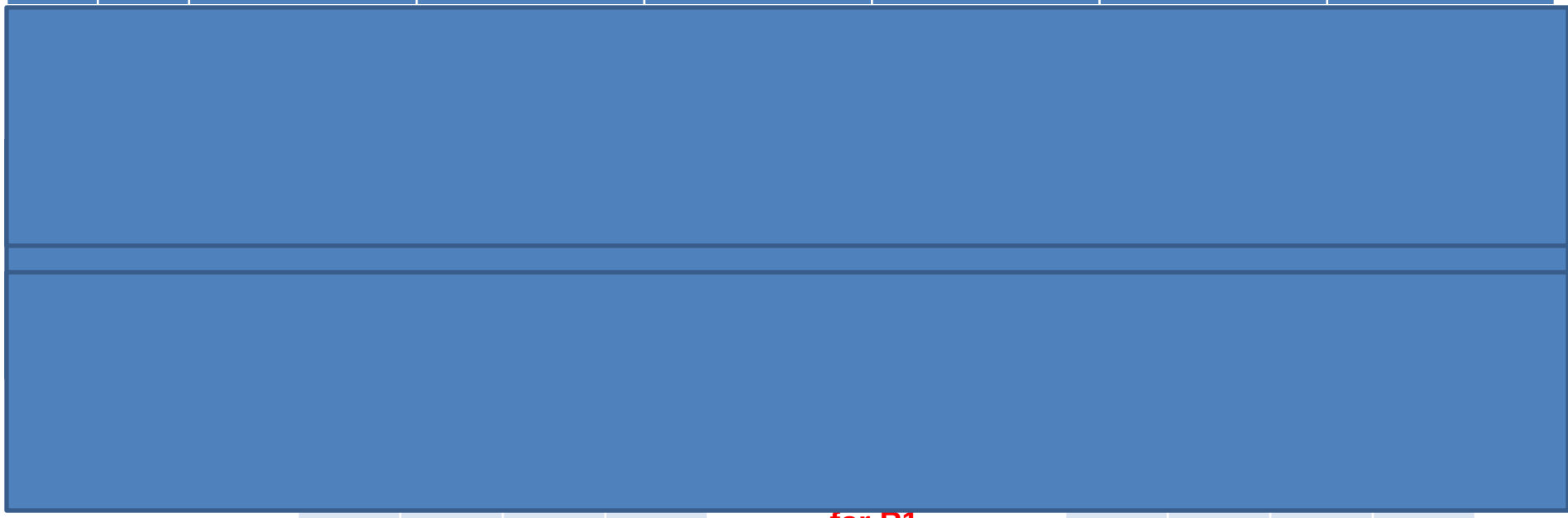
S0	S1	Operation
1	0	R1<- R1‘
0	1	R1<- R0



Example Register Transfer

S0	S1	Operation
1	0	R1<- R1‘
0	1	R1<- R0

S0	S1	Q0(R0)	Q0(R1)	D0(R0)	D0(R1)	Load Enable for R0	Load Enable for R1



for R1

Q0(R0)Q0(R1)

x	0	x	0
x	1	x	0
x	1	x	1

Q0(R0)Q0(R1)

0	1	0	1
0	1	0	1
0	1	0	1

Example Register Transfer

S0	S1	Operation
1	0	R1<- R1‘
0	1	R1<- R0

S0S1

D0(R1)

Q0(R0)Q0(R1)				
	x	0	x	1
	x	0	x	0
	x	1	x	0
	x	1	x	1

S0S1

Load Enable for R1

Q0(R0)Q0(R1)				
	0	1	0	1
	0	1	0	1
	0	1	0	1
	0	1	0	1

$$D_i(R0) = anything; i = 0,1,2,3$$

$$D_i(R1) = S0 * \bar{Q}_i(R1) + S1 * Q_i(R0)$$

$$LOAD(R0) = 0$$

$$LOAD(R1) = S0 \oplus S1$$

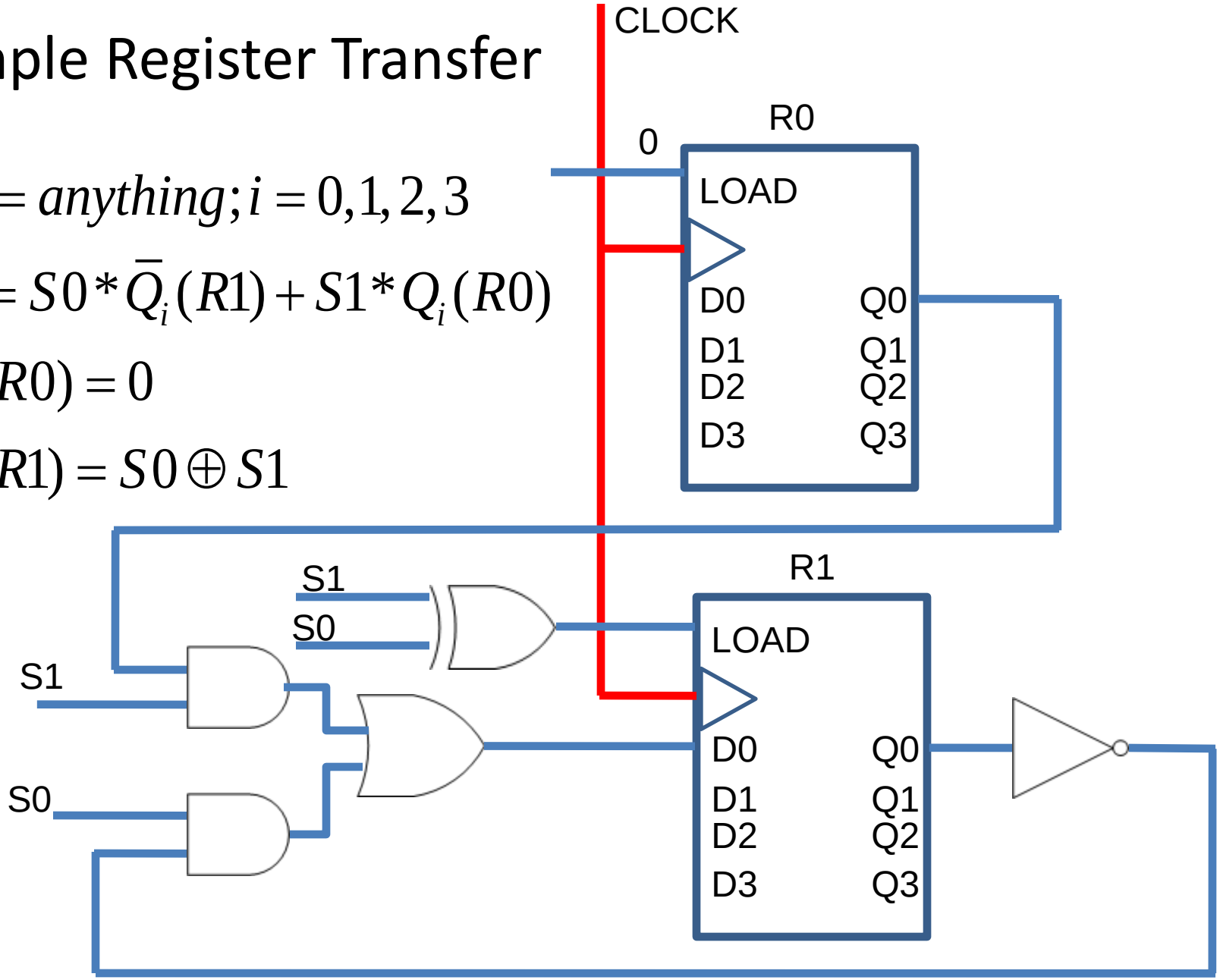
Example Register Transfer

$$D_i(R0) = anything; i = 0,1,2,3$$

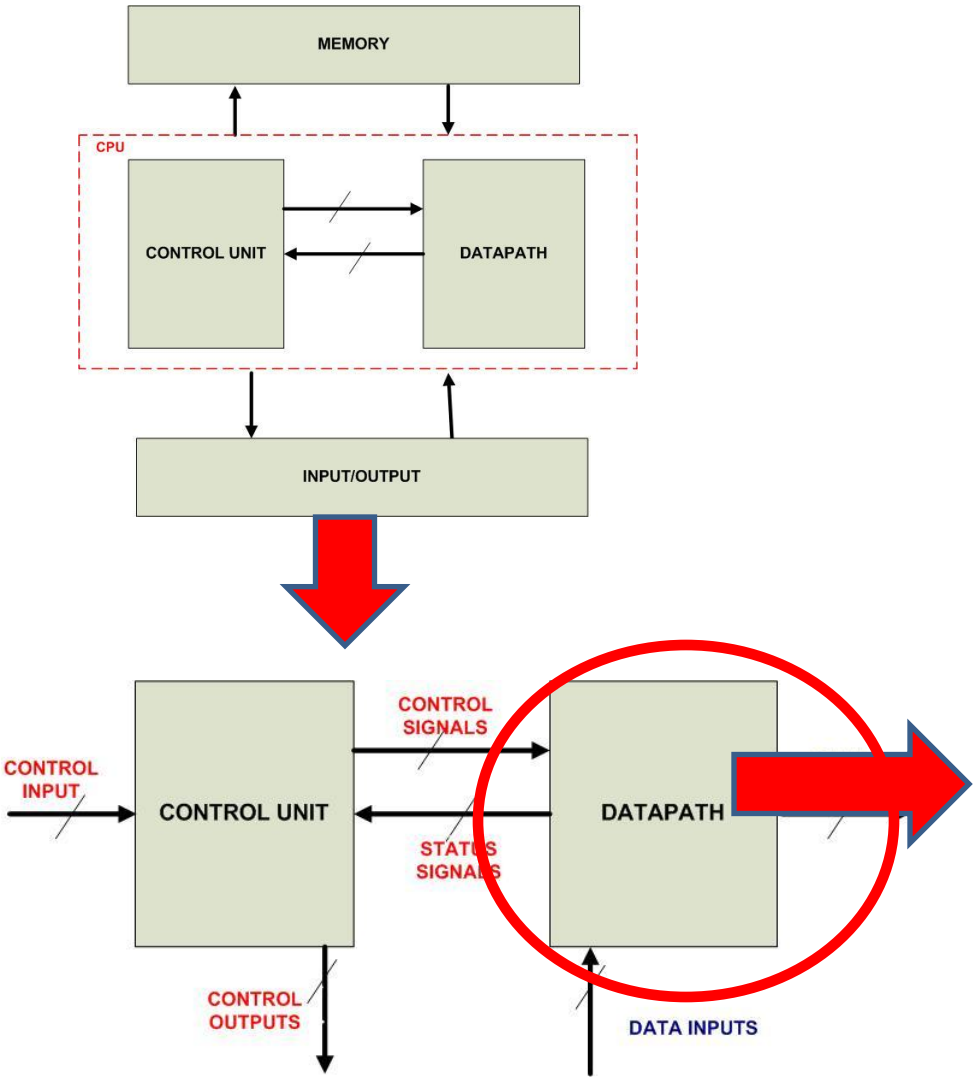
$$D_i(R1) = S0 * \bar{Q}_i(R1) + S1 * Q_i(R0)$$

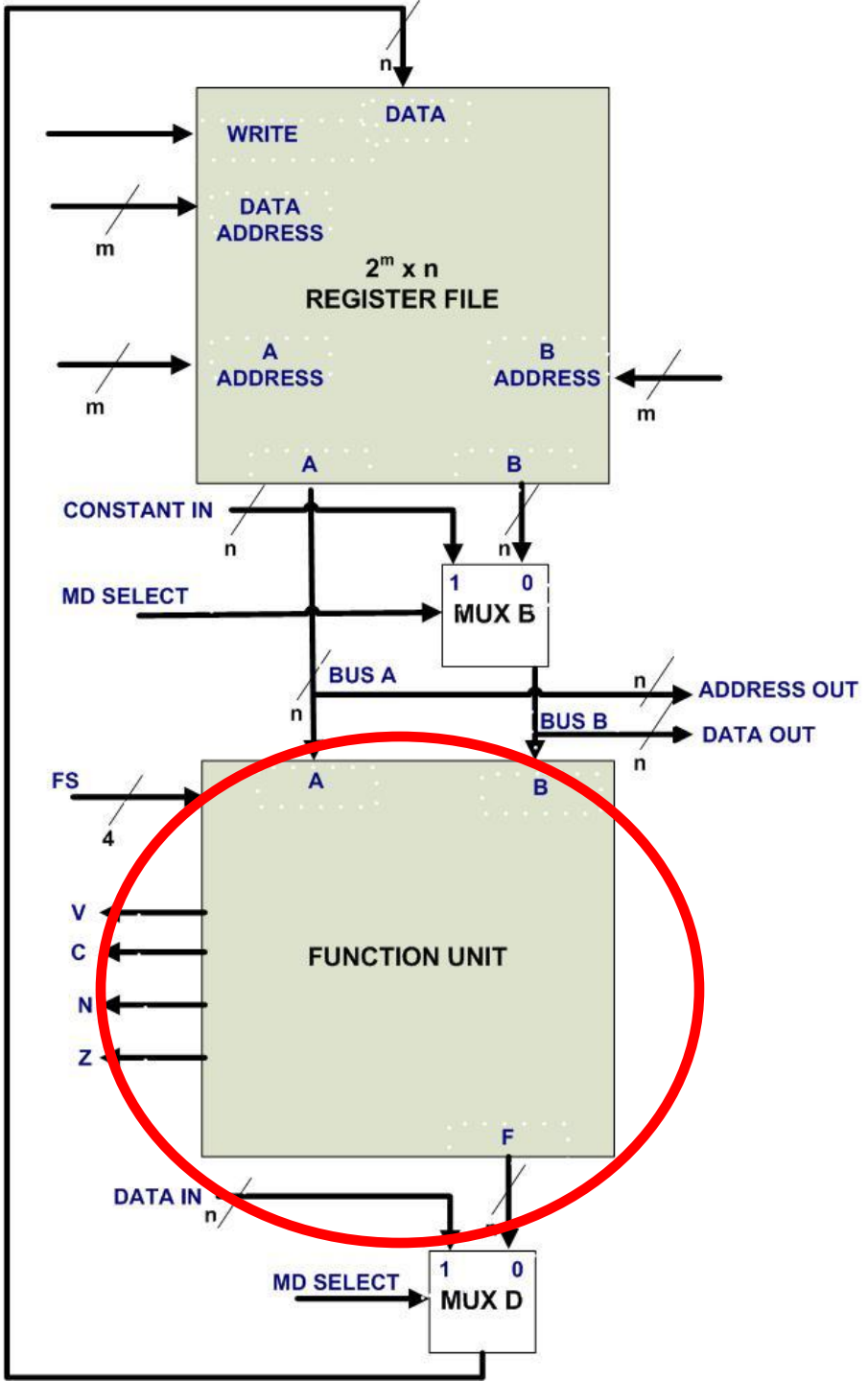
$$LOAD(R0) = 0$$

$$LOAD(R1) = S0 \oplus S1$$



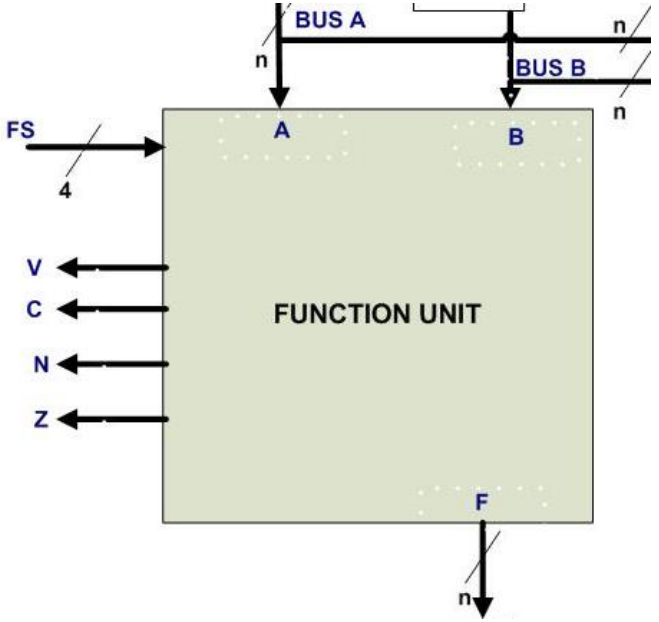
Data Path Design:
Function Unit





Data Path Design: Function Unit

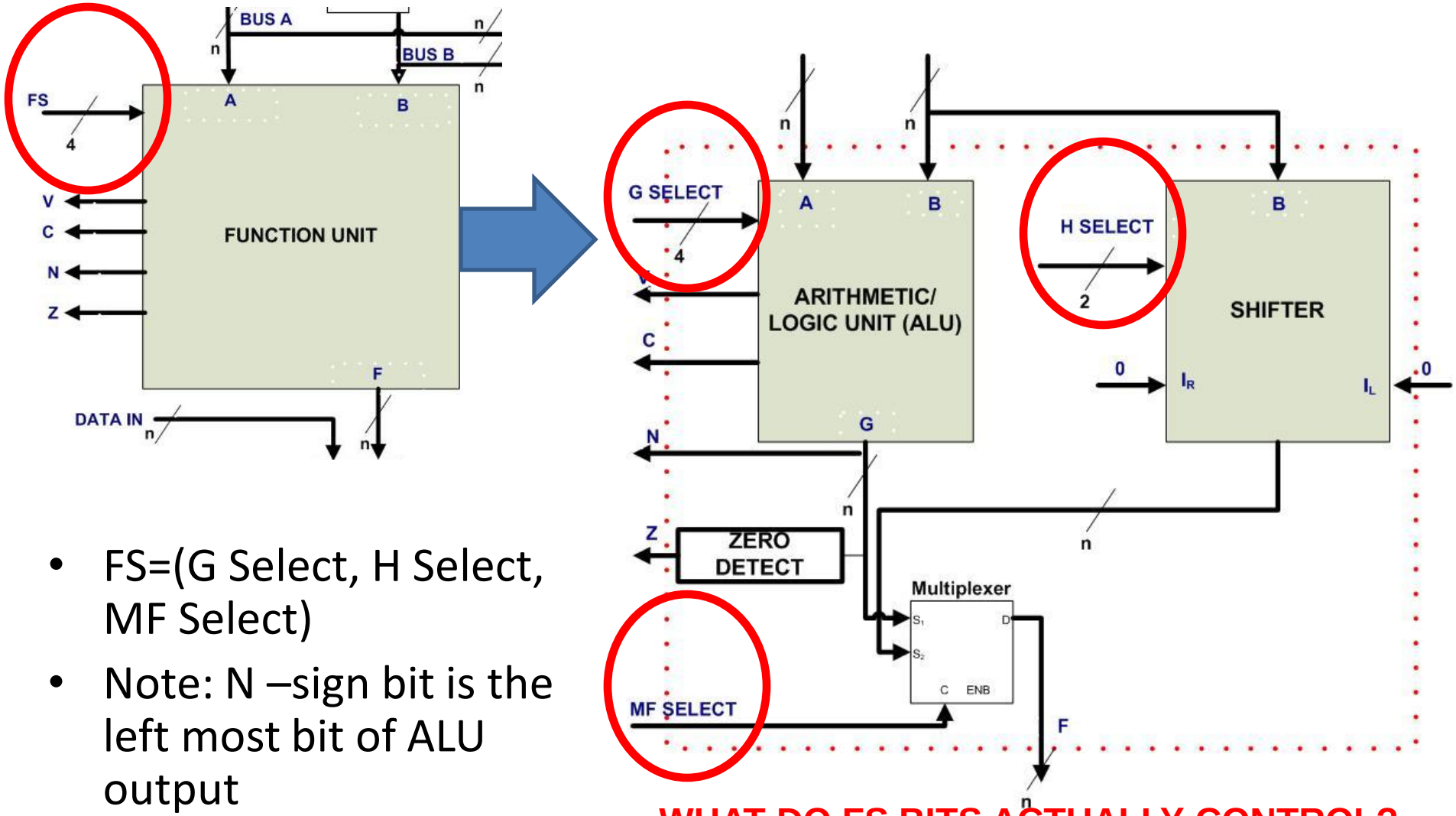
- Function unit does the following:
  - Gets inputs from the Register File
  - Performs arithmetic, logical and shift micro-operations
  - Outputs to the Register File
  - Controls inputs for selecting functions by using Function Select (FS)
  - Provides information back to the control unit about Status (V, C, N, Z)



Status Bits

- V – overflow bit
  - 1 ~ overflow, 0 ~ no overflow
- C – carry bit
- N – sign bit
  - 1 ~ negative number, 0 ~ positive number
- Z – result of compare to zero operation
  - 1 ~ result is zero, 0 ~ result is non-zero

Function Unit Model and Function Select Bits



- FS=(G Select, H Select, MF Select)
- Note: N –sign bit is the left most bit of ALU output

WHAT DO FS BITS ACTUALLY CONTROL?

Function Select Bits

- **Goal:** To encode micro-operations applied to A and B, and add copy/move micro-operation

Symbols	Operations
F=A+B	Addition
F=A’	1’s complement
F=A’+1	2’s complement
F=A+B’+1	Subtraction
F=A+1	Increment register
F=A-1	Decrement register
F=A’	Bitwise NOT
F=A $\wedge$ B	Bitwise AND
F=A $\vee$ B	Bitwise OR
F=A $\oplus$ B	Bitwise XOR
F=sl B	Shift left
F=sr B	Shift right

ARITHMETIC

LOGIC

ORANGE – identical operations

Note: Two registers A & B are involved => Move/Copy operation is needed

Function Select Bits (cont.)

FS[3:0]	MF Select	G Select	H Select	Symbols	Operations
0000	0	0000	XX	F=A	Move A
0001	0	0001	XX	F=A+1	Increment register
0010	0	0010	XX	F=A+B	Addition
0011	0	0011	XX	F=A+B+1	Addition plus one
0100	0	0100	XX	F=A+B’	Addition with 1’s complement of B
0101	0				
0110	0				
0111	0				
1000	0				
1001	0	1X01	XX	F=A $\vee$ B	Bitwise OR
1010	0	1X10	XX	F=A $\oplus$ B	Bitwise XOR
1011	0	1X11	XX	F=A’	Bitwise NOT
1100	1	XXXX	00	F=B	Move B
1101	1	XXXX	01	F=sl B	Shift left
1110	1	XXXX	10	F=sr B	Shift right

$$MF = F[3]F[2]$$

$$G[3] = F[3]; G[2] = F[2]; G[1] = F[1]; G[0] = F[0]$$

$$H[1] = F[1]; H[0] = F[0]$$