

CHAPTER XI

DATAPATH ELEMENTS

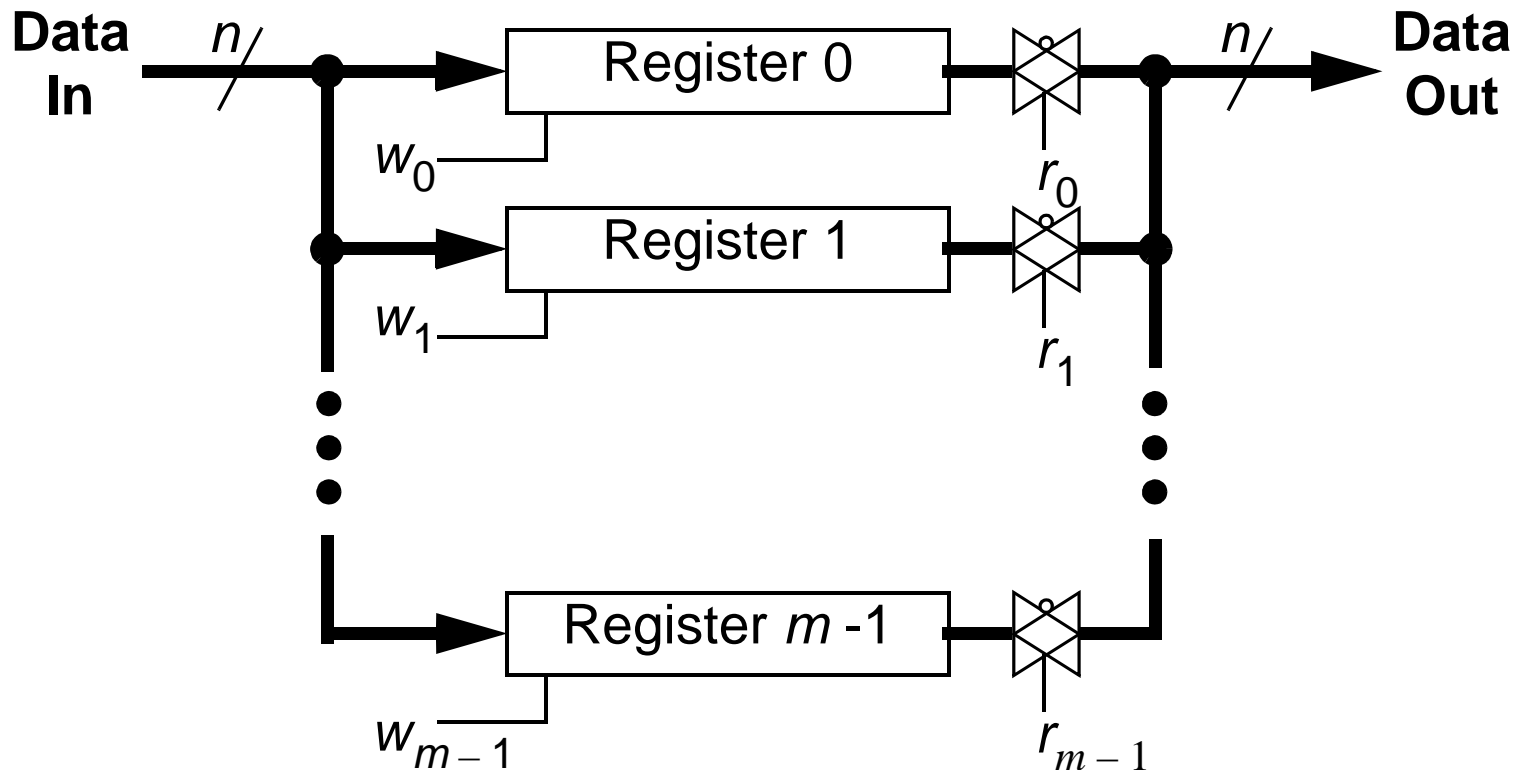
READ DATAPATH ELEMENTS FREE-DOC ON COURSE WEBPAGE

- **So far we have discussed many small components and building blocks.**
- **One final step in our building blocks before we can start to piece together a microprocessor is various datapath elements.**
 - We have already discussed portions of these datapath elements in terms of other components and building blocks.
 - We will now consider some of these components and building blocks in ways that will make the design of a microprocessor a little easier in the next chapter.

REGISTER FILES

REGISTER LAYOUT

- A general $m \times n$ register file with m registers that are each n -bits wide is illustrated below.

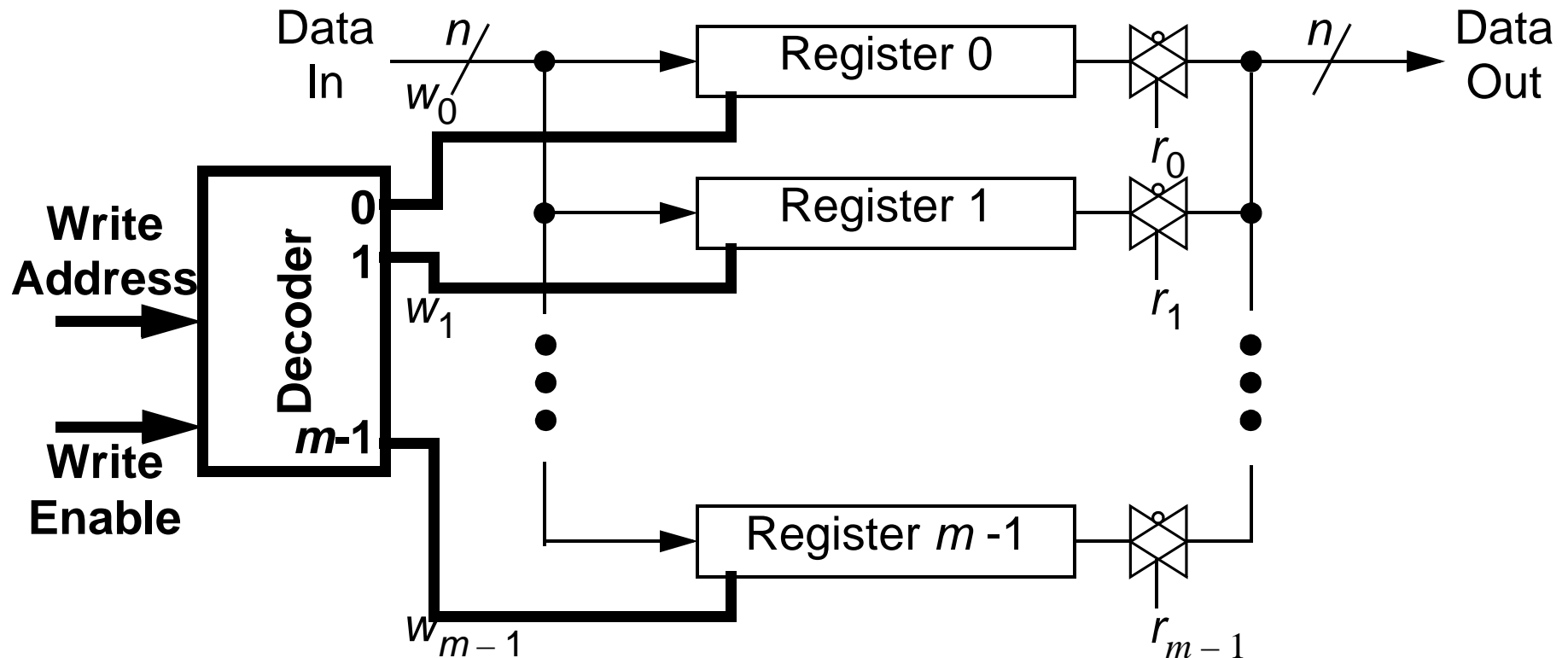


- The r_k and w_j signals indicate which register to read/write, respectively.

REGISTER FILES

WRITE DECODER

- For writing to a register, we include a write address with decoder.

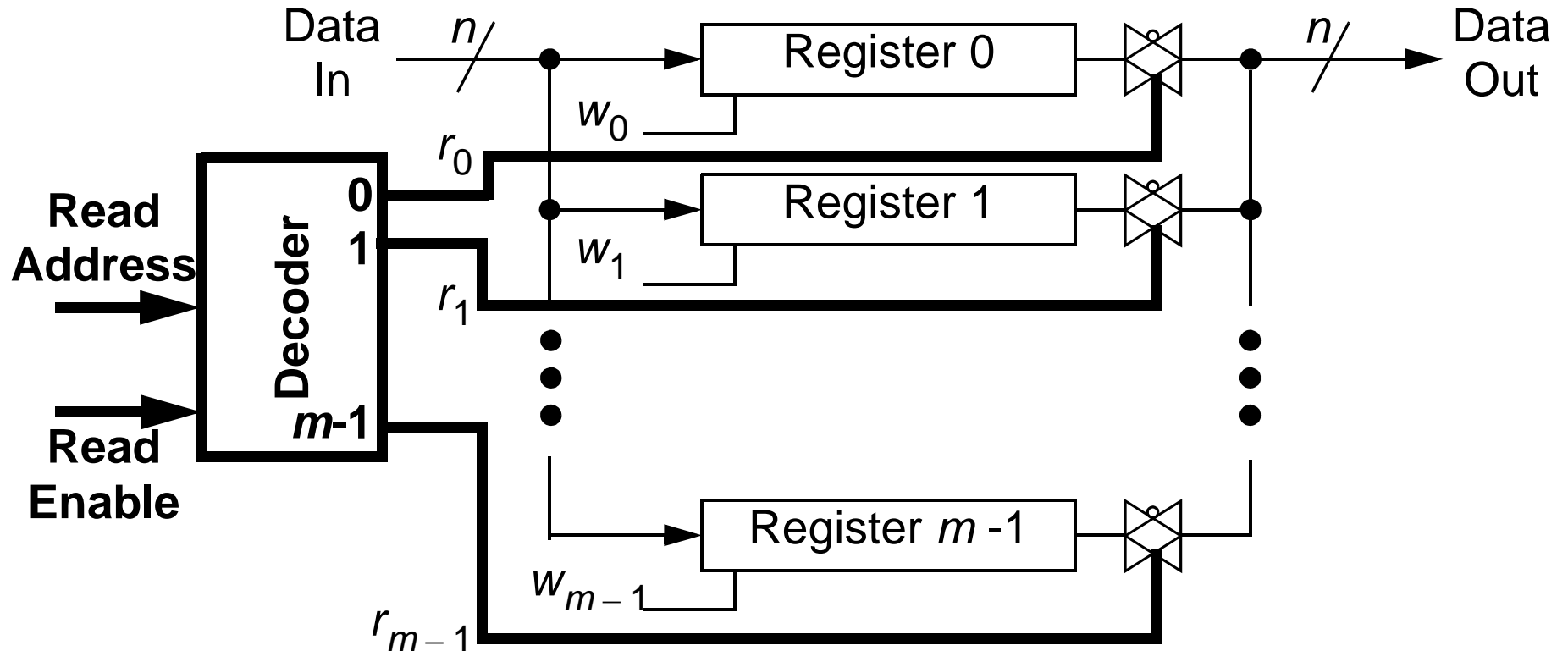


- A given **Write Address** (with **Write Enable = 1**) selects which register, 0 through $m - 1$, to store the input from **Data In**.

REGISTER FILES

READ DECODER

- For reading from a register, we include a read address with decoder.



- A given **Read Address** (with **Read Enable** = 1) selects which register, 0 through $m - 1$, to read from and output to Data Out.
- Could have multiple **data outputs** with multiple **read address** decoders.

REGISTER FILES

32-BIT WORD, 32 REGISTERS

- For the upcoming datapath designs in the next chapter, we want to have a 32x32 register file with one write input and two read outputs.

X_{ra} - X read address

Y_{ra} - Y read address

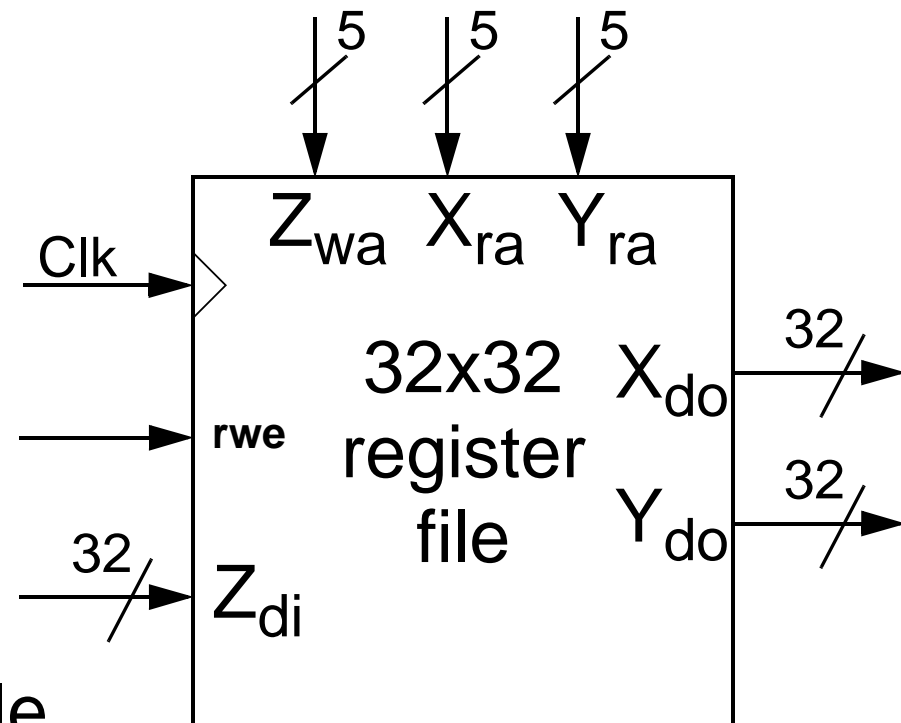
Z_{wa} - Z write address

X_{do} - X data out

Y_{do} - Y data out

Z_{di} - Z data in

rwe - register write enable

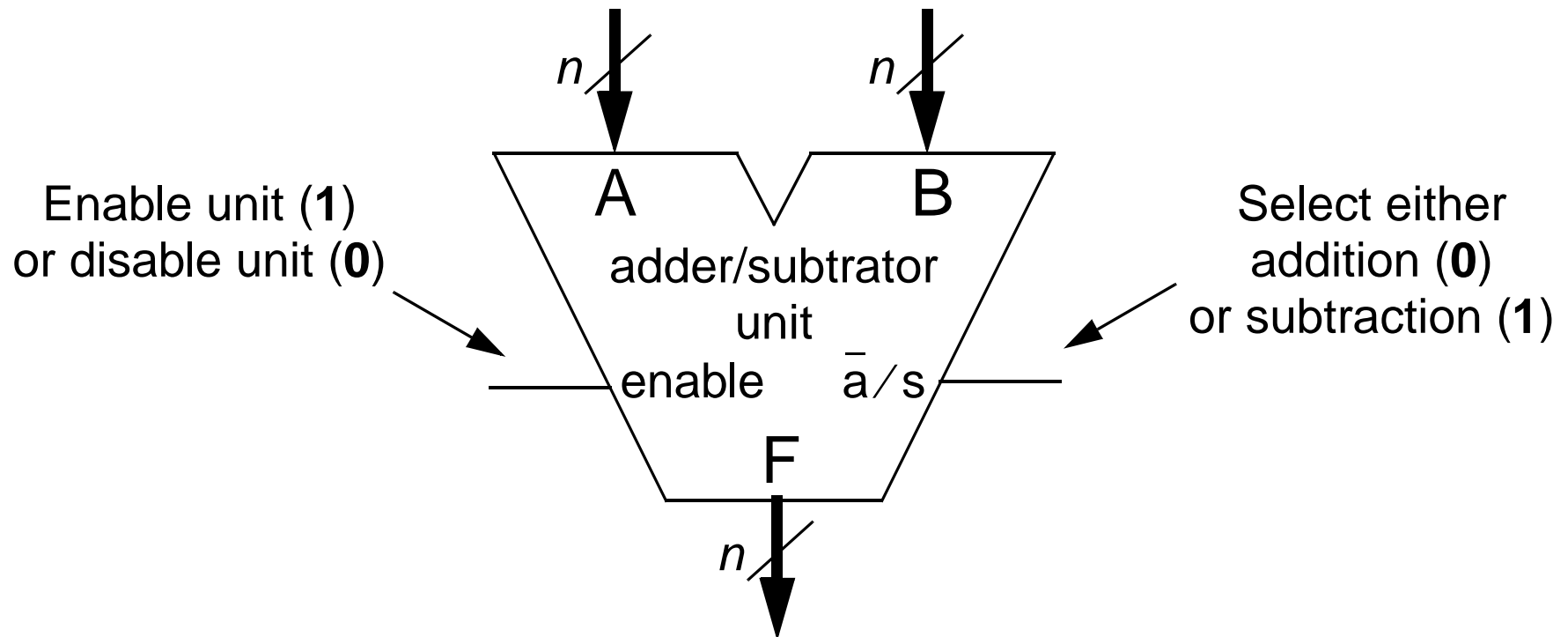


- Note: Two data outputs implemented with two read address decoders.

ADDER/SUBTRACTOR

GENERAL UNIT DIAGRAM

- An n -bit adder/subtractor unit is often illustrated as follows.

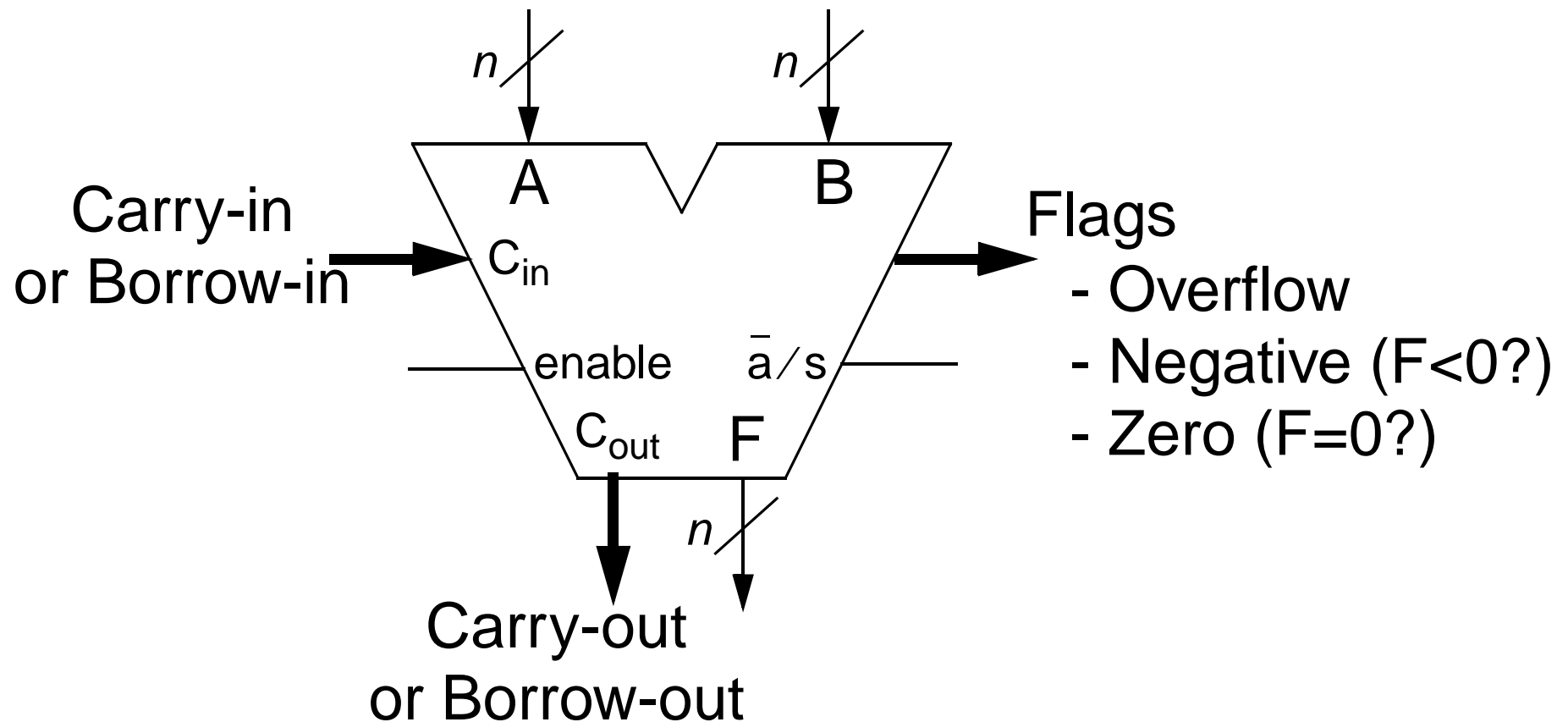


- This unit would have n full-adders internally.

ADDER/SUBTRACTOR

OTHER UNIT SIGNALS

- Other signals often included with an adder/subtractor are shown below.



LOGICAL UNIT

INTRODUCTION

- A useful unit would be one that can take two n -bit inputs and perform some logical operation between each of the bits to get an n -bit output.
- For example, given the 8-bit values **0001 1110** and **1001 1000**, we might want to find the **bit-wise logical OR**.

bit-wise	0001 1110
logical OR	1001 1000
	<hr/>
	1001 1110

- Or similarly, the **bit-wise logical AND** of the two 8-bit values.

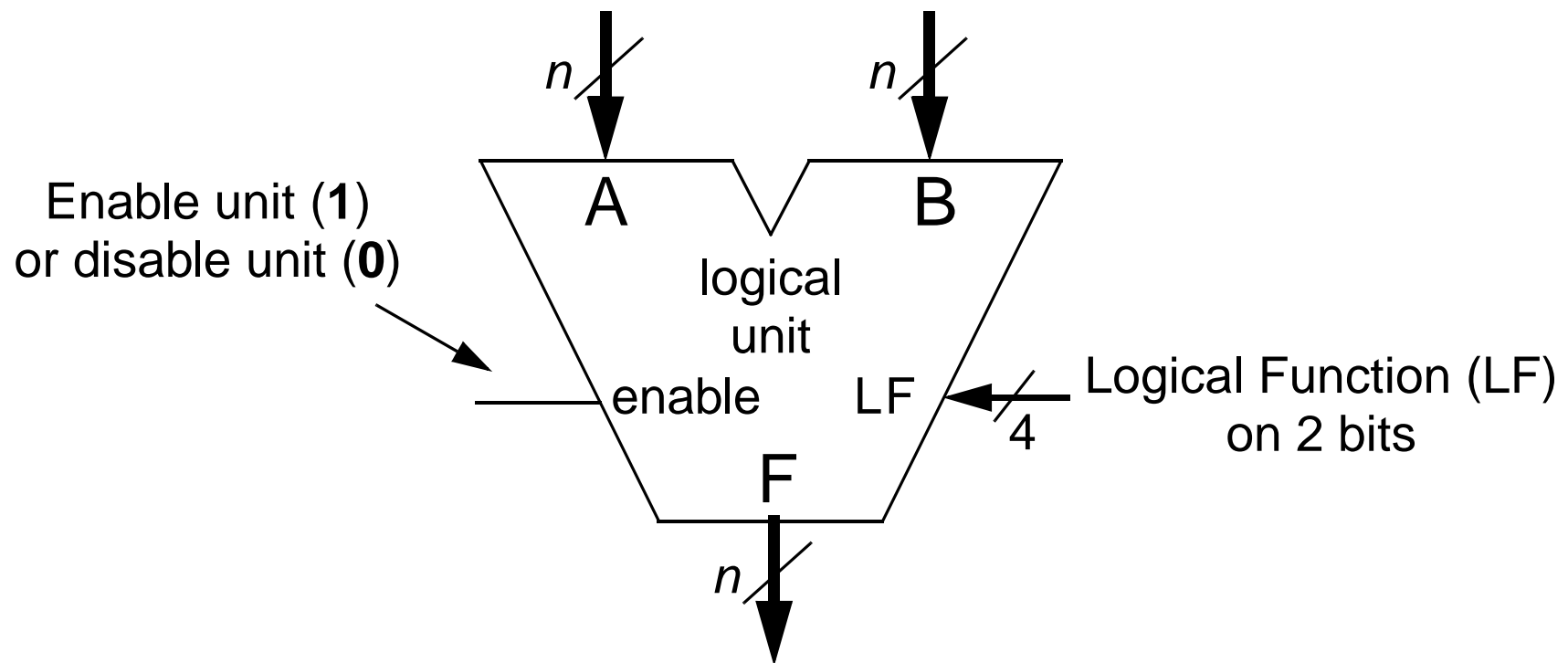
bit-wise	0001 1110
logical AND	1001 1000
	<hr/>
	0001 1000

- These types of operations are often used for masking and setting bits.

LOGICAL UNIT

GENERAL UNIT DIAGRAM

- Below is a general unit diagram for an n -bit logical unit.



- Logical operations, such as **AND/OR/NOT/NAND/NOR**/etc., are done for each bit of **A** and **B** to form **F**.

LOGICAL UNIT

4-BIT LOGICAL FUNCTIONS (LF)

- ADDER/SUBTRACTOR
- LOGICAL UNIT
 - INTRODUCTION
 - GENERAL UNIT DIAGRAM

- Recall the possible logic functions for two bits, A and B.
 - We can use the column F_n as the 4-bit LF input for the logical unit.

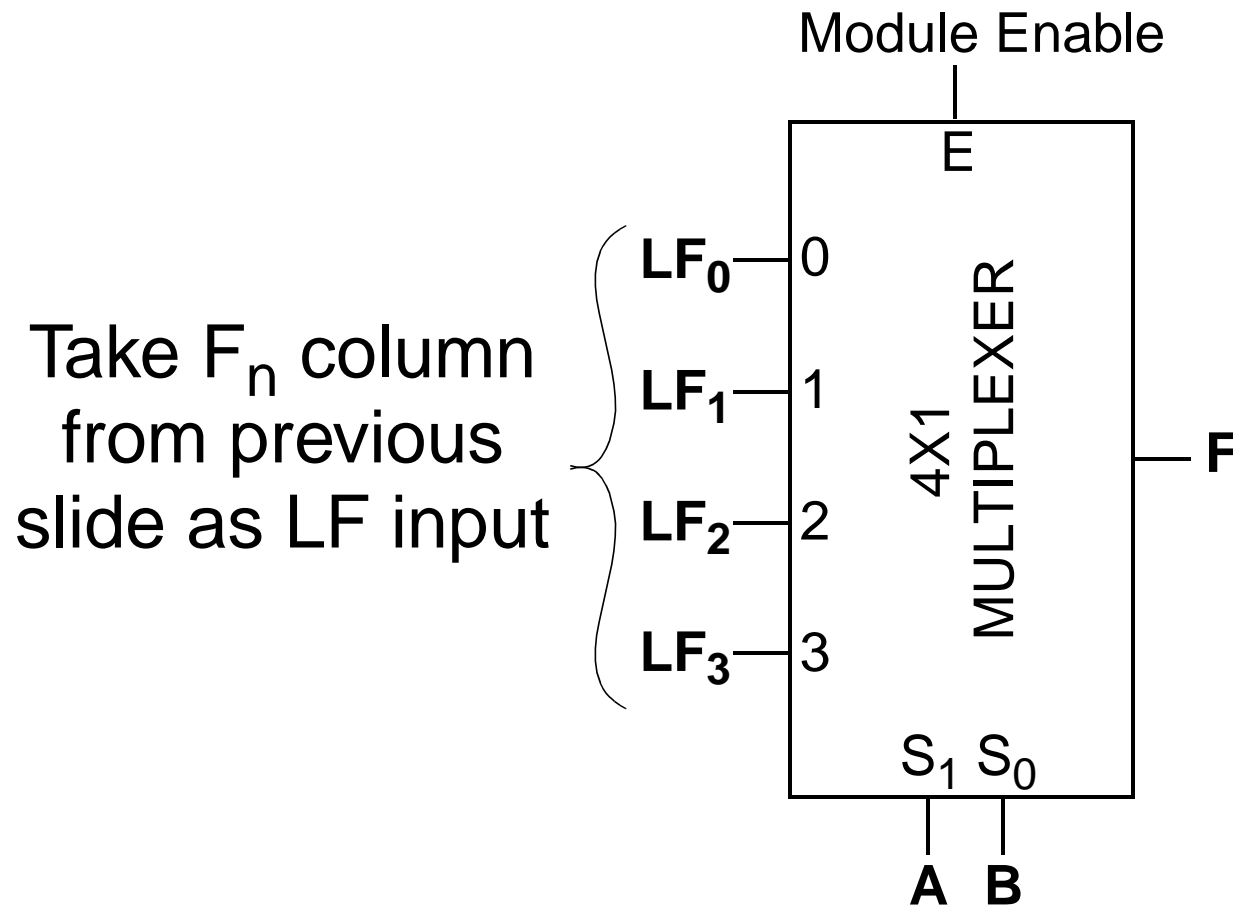
A	B	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

0	AB		A		B	A + B		A + B		B		A		A		AB	1
Null			Inhibition		A ⊕ B		A ⊕ B		A ⊕ B		Implication		Implication		Implication		Identity
						A + B		A + B									

LOGICAL UNIT

BIT SLICE IMPLEMENTATION

- A number of internal implementations exist for the logical unit.
 - The easiest is to use a **4-to-1 multiplexer** for **each bit** as follows



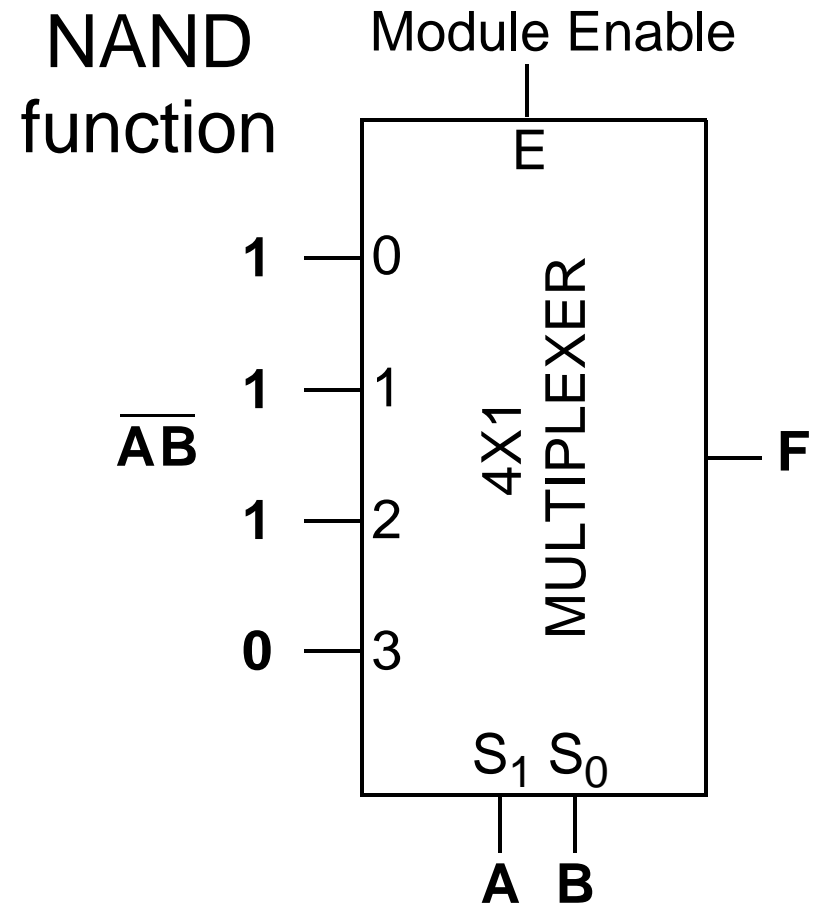
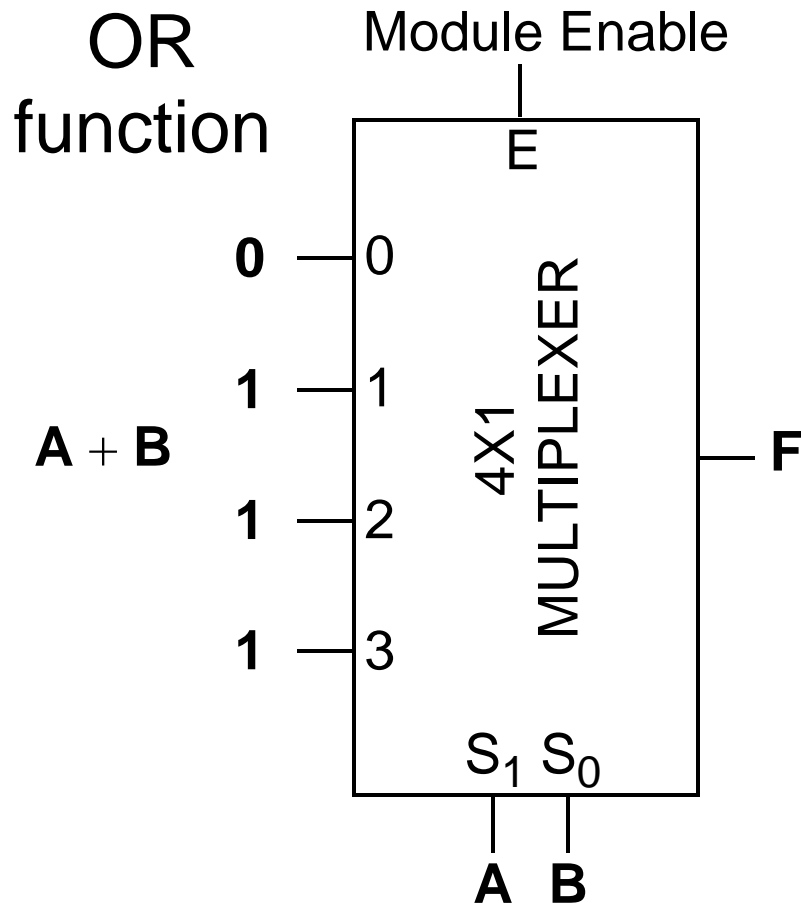
Require n of these to form our n -bit logical unit.

Note: When you look at a design for each bit, it is known as a **bit slice**

LOGICAL UNIT

BIT SLICE IMPLEMENTATION

- The following are example LF inputs for a logical unit bit slice.



SHIFT UNIT

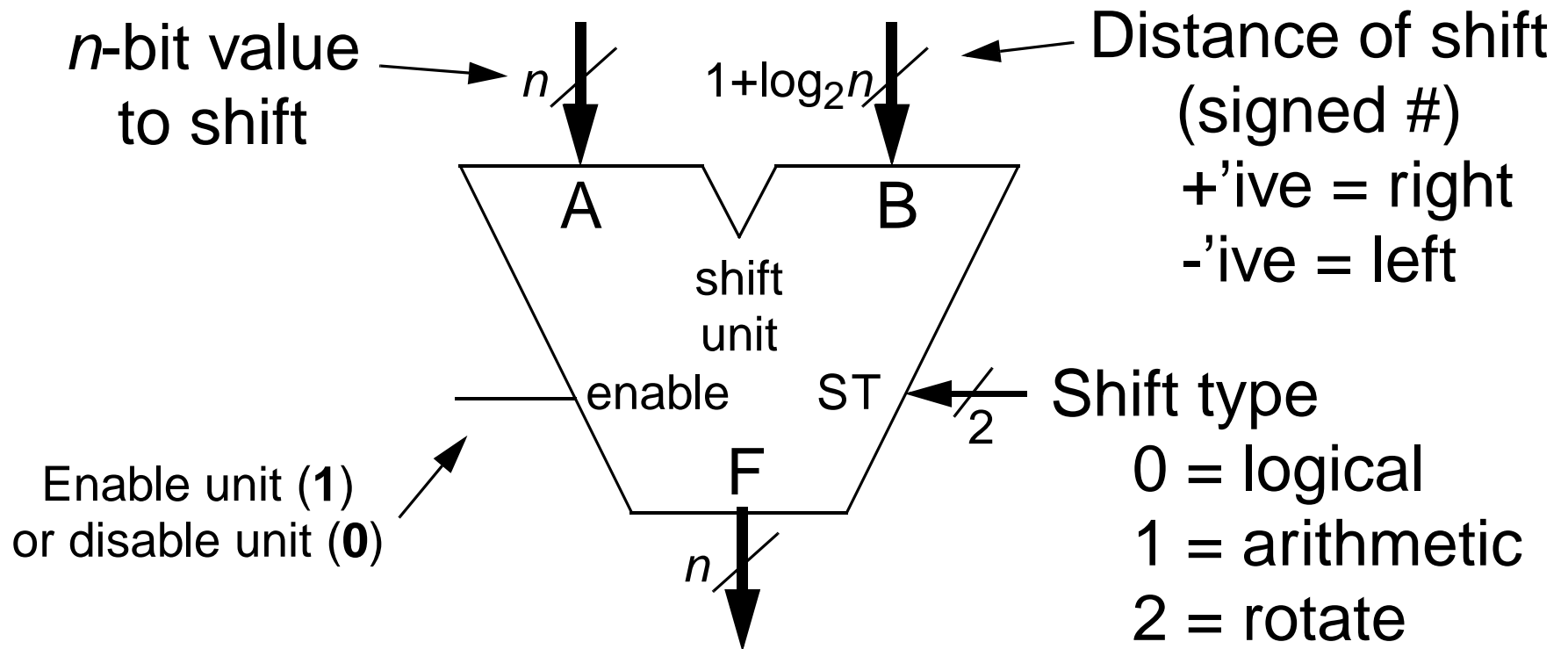
INTRODUCTION

- **We have already discussed the bulk about shift units in previous chapters.**
- **As given in the Free-Doc, there are different types of shift units.**
 - Logical shift
 - Arithmetic shift
 - Circular shift (this is just a rotate unit)
- **We want to discuss an implementation, the barrel shifter, that will be useful in our single cycle datapath computer we will design next chapter.**

SHIFT UNIT

GENERAL UNIT DIAGRAM

- Below is a general unit diagram for an n -bit shift unit.

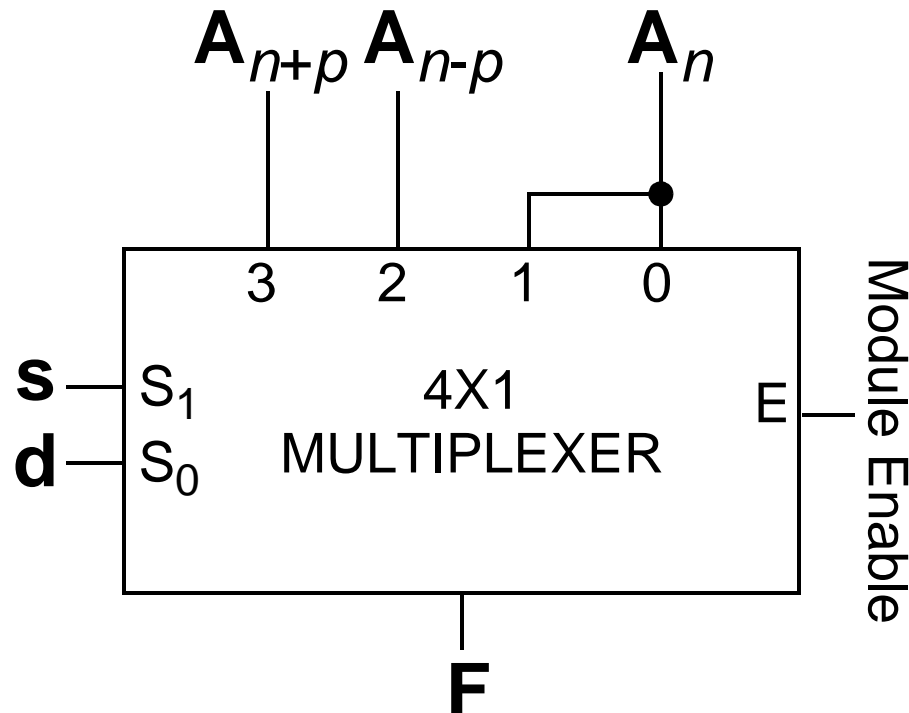


- Notice that the n -bit value **A** will be shifted according to the distance indicated with signed number **B**.

SHIFT UNIT

P-SHIFTER BIT SLICE

- Previously, we discussed the *p*-shifter but not its implementation.
- A *p*-shifter shifts the value to the left or right by *p*-bits.
- A bit slice view of a *p*-shifter for *n*th bit could be as follows.



 s
0 = no shift
1 = shift

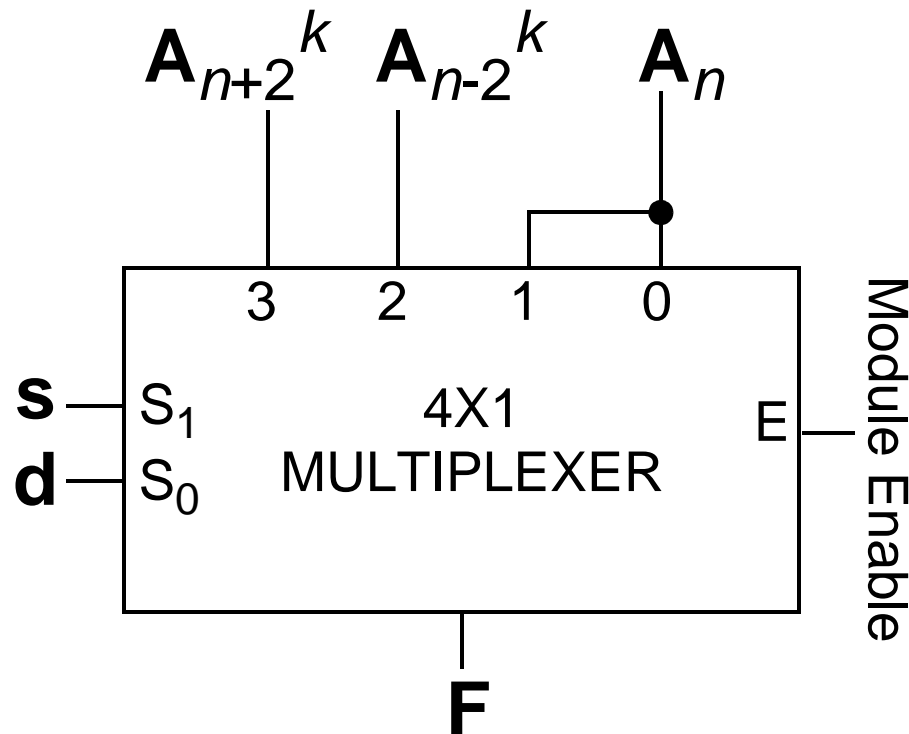
 d
0 = left
1 = right

- Notice that this can **ONLY** shift by *p*-bits. It is **hardwired** to shift *p*-bits.

SHIFT UNIT

2^k -SHIFTER BIT SLICE

- A useful type of p -shifter is when $p = 2^k$ for some positive integer k .



s

0 = no shift
1 = shift

d

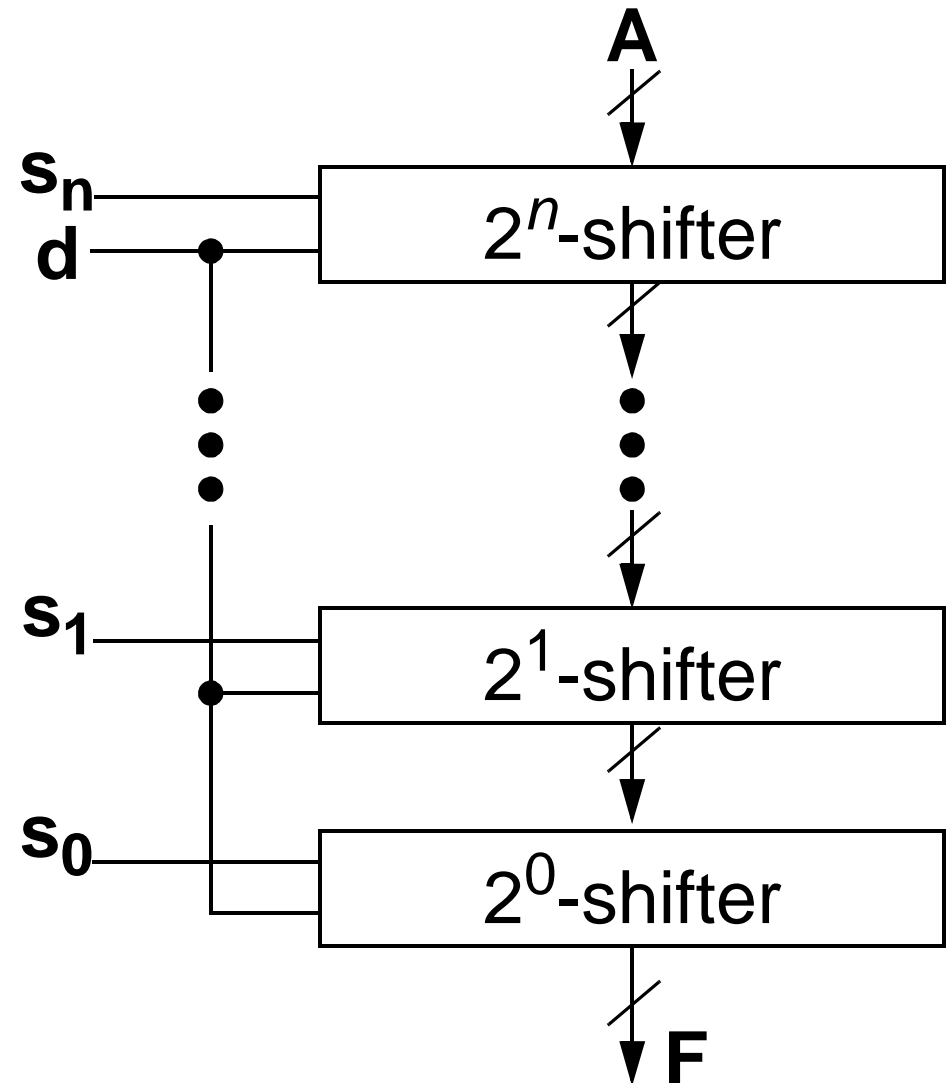
0 = left
1 = right

- A 2^k -shifter allows use to build a barrel shifter.

SHIFT UNIT

BARREL SHIFTER

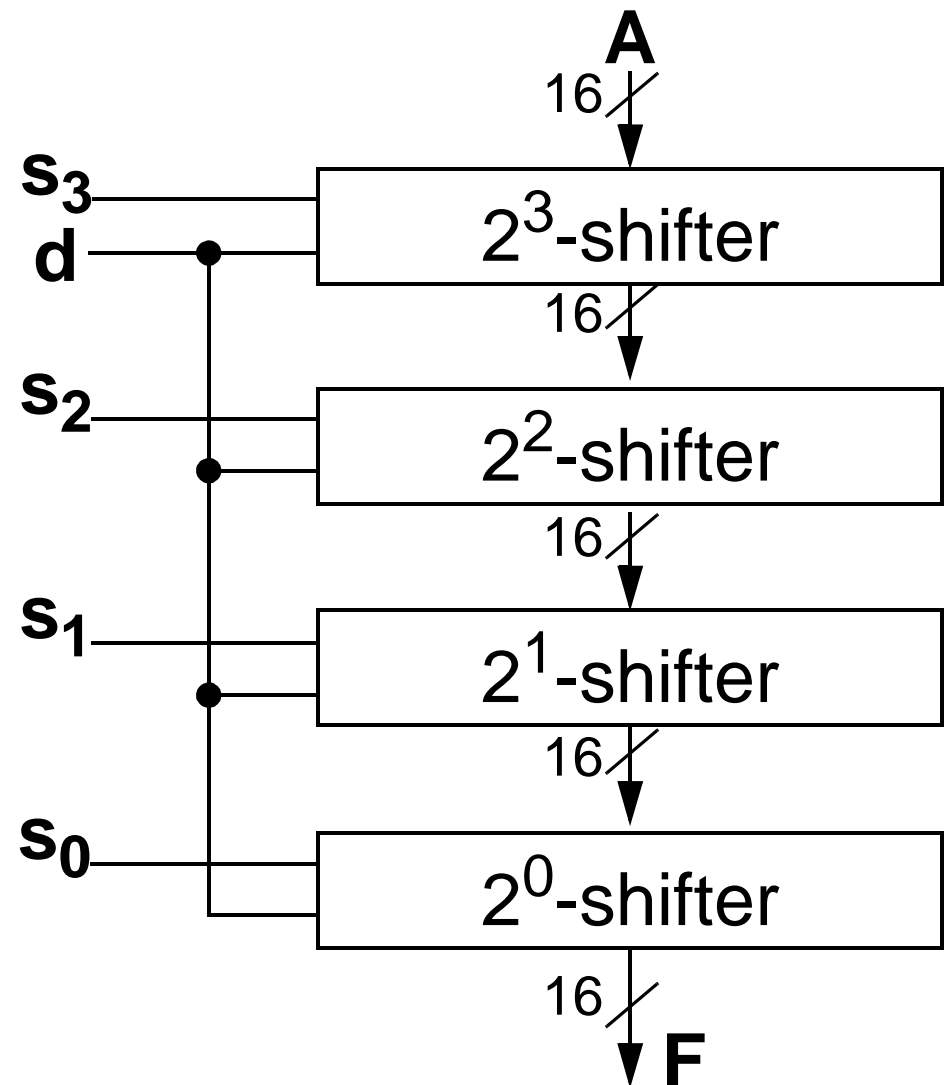
- We want to be able to shift a vector by an arbitrary distance instead of hardwired like the p -shifter and 2^k -shifter.
- The top level can shift A by n bits, depending on s_n .
- Subsequent levels can shift result by $n/2$ bits, depending on their input s_q .



SHIFT UNIT

SAMPLE BARREL SHIFTER

- We will do some examples with the following arbitrary n -shifter on a 16-bit input.
- Note that this barrel shifter can shift the input by 15 bits in either direction.



SHIFT UNIT

BARREL SHIFTER: EXAMPLE #1

- SHIFT UNIT
- 2^K -SHIFTER BIT SLICE
- BARREL SHIFTER
- SAMPLE BARREL SHIFTER

- For example, consider the input of

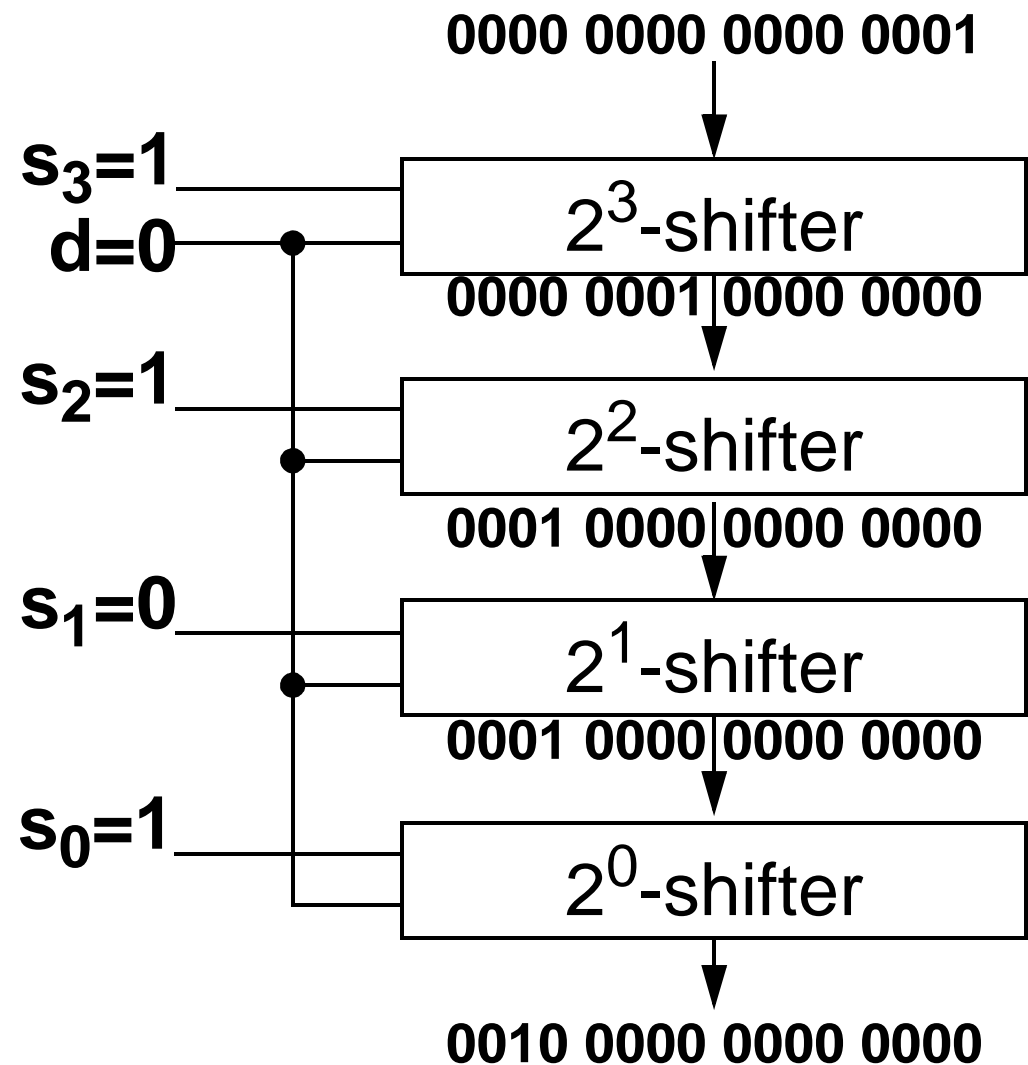
0000 0000 0000 0001

- If we want to shift this value to the **left** by **13**, we need the input

$d = 0$

$s = (s_3s_2s_1s_0) = 1101$

- Note: This example is for a logical shift.



SHIFT UNIT

BARREL SHIFTER: EXAMPLE #2

- SHIFT UNIT
 - BARREL SHIFTER
 - SAMPLE BARREL SHIFTER
 - BARREL SHIFTER EX. #1

- As another example, consider the input of
1001 0100 1110 0001
- If we want to shift this value to the **right** by **6**, we need the input
d = 1
s = (s₃s₂s₁s₀) = 0110
- Note: This example is for a logical shift.

