Register Transfers and Datapaths

Instructor: Saraju P. Mohanty

PART 1

- Datapaths and Operations
- Register Transfer Operations
- Microoperations
- Multiplexer-Based Transfer
- Bus-Based Transfer

Sources

- Logic and Computer Design Fundamentals by M. M. Mano and C. R. Kime.
- Dr. Valavanis lectures

Datapaths and Operations

- A digital system is made up of interconnected flip-flops and gates.
- Large digital systems are designed using modular and hierarchical approach.
- In most digital system design we partition the system into two modules: the datapath (data processing operations) and a control unit (determines sequence).
- Control signals (binary) activate data-processing operations. Status signals (binary) describe aspects of the state of the datapath.
- Data movement stored in registers and processing performed on data are referred to as register transfer operations, described by:
    - the set of system registers
    - operations performed on data stored in the registers
    - control that supervises the sequence of operations in the system

Datapaths and Operations …

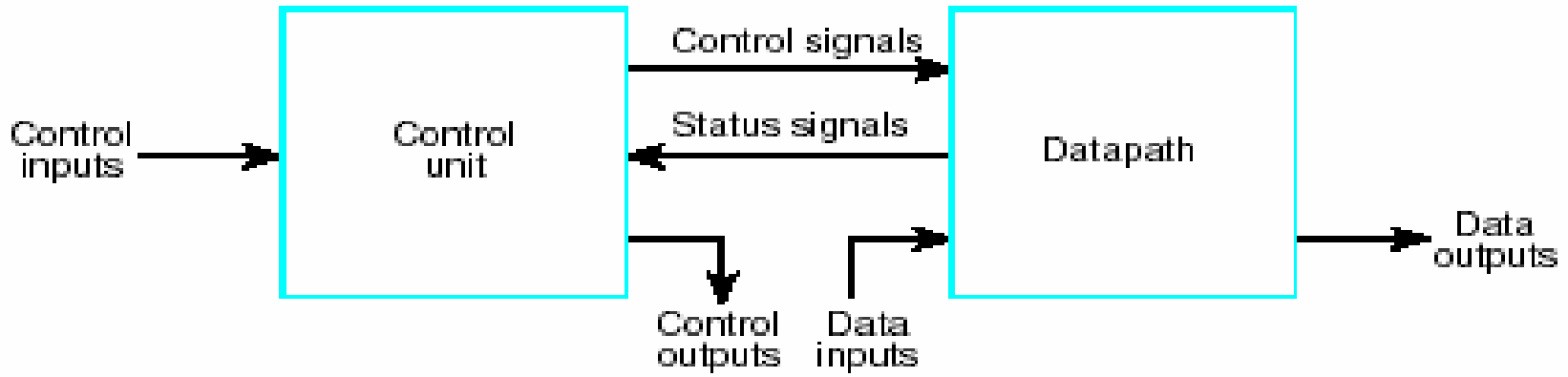


Fig. 7-1 Interaction between Datapath and Control Unit

- Register capable of performing elementary operations (load, count, add, subtract, shift).
- Elementary operations performed on data stored in registers are called microoperations.
- Example of microoperations: Loading the contents of one register into another, adding the content of two registers, etc.
- A microoperation is usually performed in parallel on string of bits during one clock cycle.

Register Transfer Operations

- The registers in a digital system is denoted by uppercase letters and sometimes followed by numerals.
- AR is the address register that holds an address for the memory unit.
- PC is the program counter and IR is instruction register.
- Similarly, R2 means register 2.

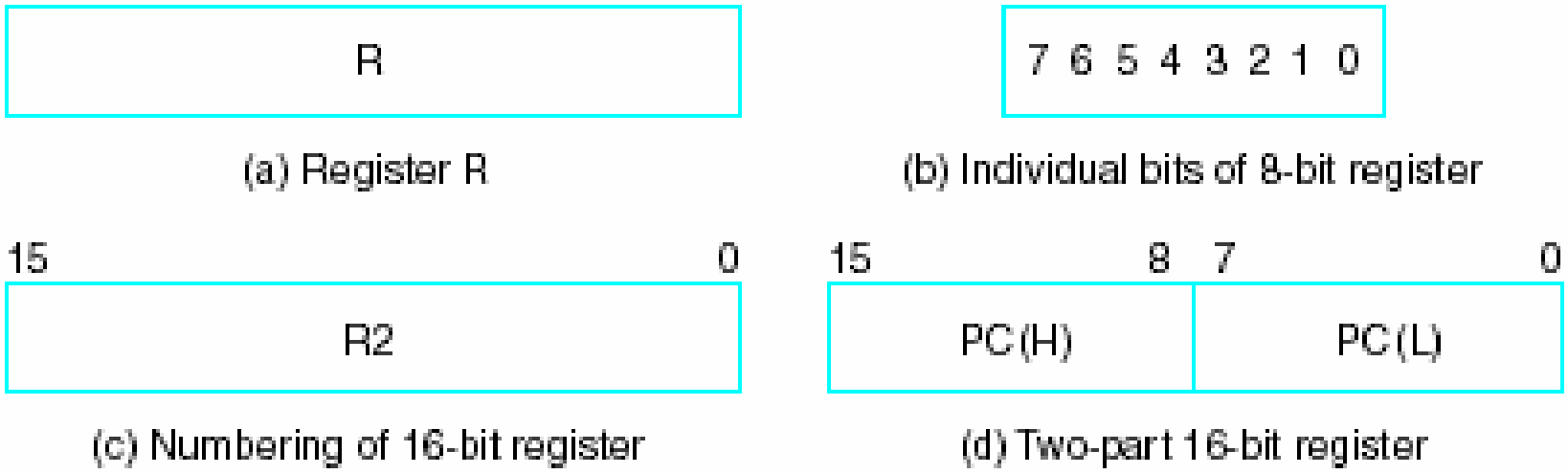
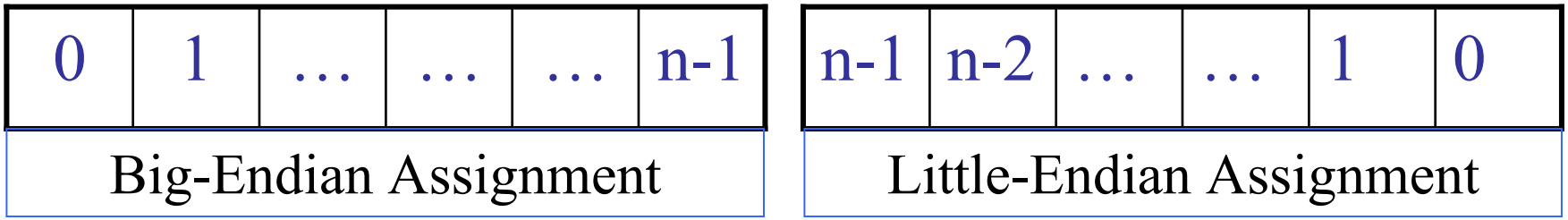


Fig. 7-2 Block Diagrams of Registers

Register Transfer Operations …

- Individual FFs in an n-bit register are numbered in sequence from 0 to n-1 or from n-1 to 0.
- Little-endian: 0 numbering at the rightmost or the least significant position
- Big-endian: 0 numbering at the leftmost or the most significant position
- In byte addressable computers above numbering scheme is applied in numbering the bytes.



Register Transfer Operations …

- Data transfer from one register to another register is designated in symbolic form by mean of replacement operator (←).
- For example, R2 ← R1 denotes transfer of contents of register R1 into register R2. The register R1 is referred to as source and R2 is referred to as destination.
- Conditional Statement: Transfer only for specified value of control signals, Option1- If (K_1=1) then (R2 ← R1) Option2- K_1: R2 ← R1
- Clock not included as a variable in the register transfer statement.

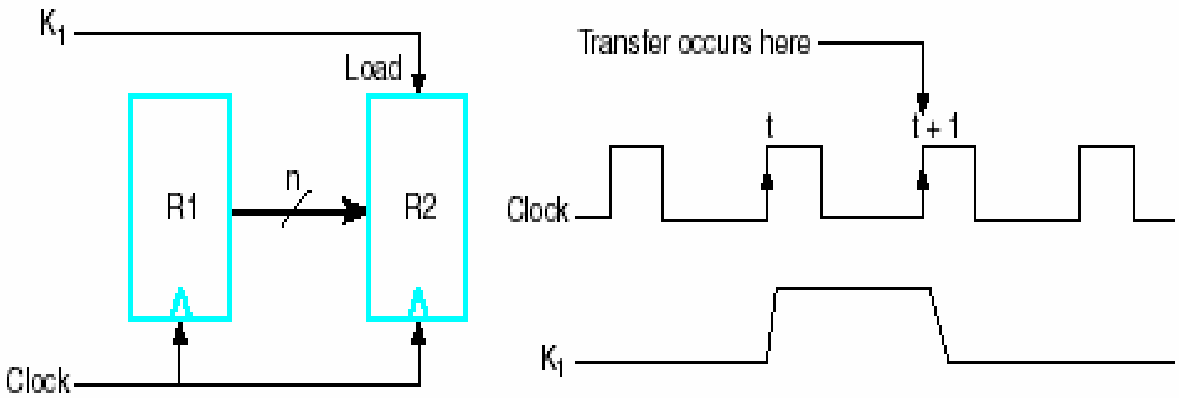


Fig. 7-3 Transfer from *R1* to *R2* when $K_1 = 1$

A comma is used to separate two or more register transfers that are executed at the same time. For example,

K3: R2 ← R1, R3 ← R1

Register Transfer Operations …

TABLE 7-1
Basic Symbols for Register Transfers

Symbol	Description	Examples
Letters (and numerals)	Denotes a register	AR, $R2$, DR, IR
Parentheses	Denotes a part of a register	$R2(1)$, $R2(7:0)$, $AR(L)$
Arrow	Denotes transfer of data	$R1 \leftarrow R2$
Comma	Separates simultaneous transfers	$R1 \leftarrow R2$, $R2 \leftarrow R1$
Square brackets	Specifies an address for memory	$DR \leftarrow M[AR]$

Table 7-1 Basic Symbols for Register Transfers

Microoperations

- The most often encountered are:
  - –Transfer data from one register to the other
  - –Arithmetic that perform arithmetic on register data
  - –Logic performing bit manipulation on register data
  - –Shift shifting data in registers.

- Arithmetic Microoperations: The statement,

  R0 ← R1+R2 specifies add operation

  In this operation R0 ← R1 + (R2)' + 1 (R2)' specifies the 1's complement of R2; (R2)'+1 gives the 2's complement of R2. Adding this to R1 is equivalent to R1-R2. So subtraction in terms of 2's complement.

Microoperations: Arithmetic

TABLE 7-3
Arithmetic Microoperations

Symbolic designation	Description
$R0 \leftarrow R1 + R2$	Contents of $R1$ plus $R2$ transferred to $R0$
$R2 \leftarrow \overline{R2}$	Complement of the contents of $R2$ (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement of the contents of $R2$
$R0 \leftarrow R1 + \overline{R2} + 1$	$R1$ plus 2's complement of $R2$ transferred to $R0$ (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of $R1$ (count up)
$R1 \leftarrow R1 - 1$	Decrement the contents of $R1$ (count down)

Table 7-3 Arithmetic Microoperations

Microoperations: Arithmetic …

There is a direct relationship between the statements written in register transfer notation and the registers and digital functions required for their implementations. For example Consider Implementation of :

$(X)'K_1$: R1 ← R1+R2

XK_1: R1 ← R1+ (R2)' + 1

Flip-flops

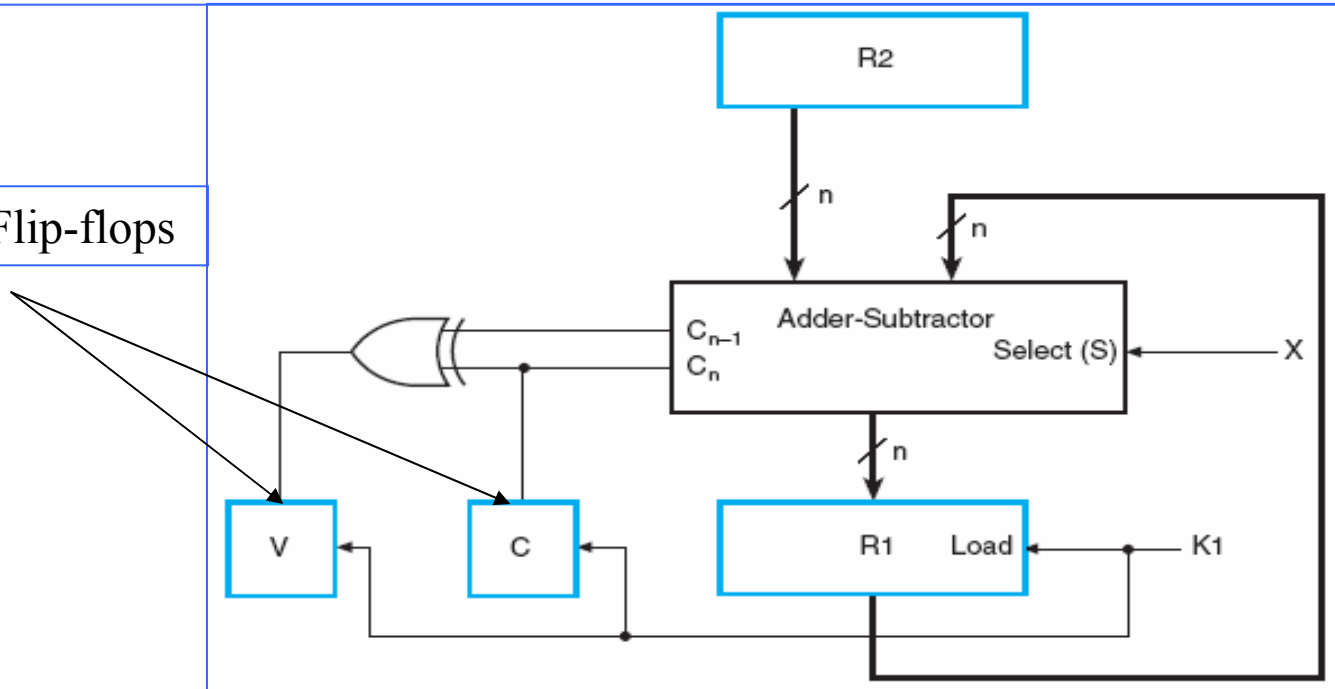


Fig. 7-4 Implementation of Add and Subtract Microoperations

S = 0 the two inputs are added.

S=1 they are subtracted.

Microoperations: Logic and Shift

Symbolic designation	Description
$R0 \leftarrow \overline{R1}$	Logical bitwise NOT (1's complement)
$R0 \leftarrow R1 \wedge R2$	Logical bitwise AND (clears bits)
$R0 \leftarrow R1 \vee R2$	Logical bitwise OR (sets bits)
$R0 \leftarrow R1 \oplus R2$	Logical bitwise XOR (complements bits)

Table 7-4 Logic Microoperations

		Eight-bit examples	
Type	Symbolic designation	Source *R2*	After shift: Destination *R1*
shift left	$R1 \leftarrow$ sl $R2$	10011110	00111100
shift right	$R1 \leftarrow$ sr $R2$	11100101	01110010

Table 7-5 Examples of Shifts

Multiplexer-based Transfer

There are occasions when register receives data from two or more different sources at different times, example If (K_1=1) then (R0 ← R1) else if (K_2=1) then (R0 ← R2). Conditional statement may be broken into two parts:

K_1=1: R0 ← R1, (K_1)' K_2: (R0 ← R2)

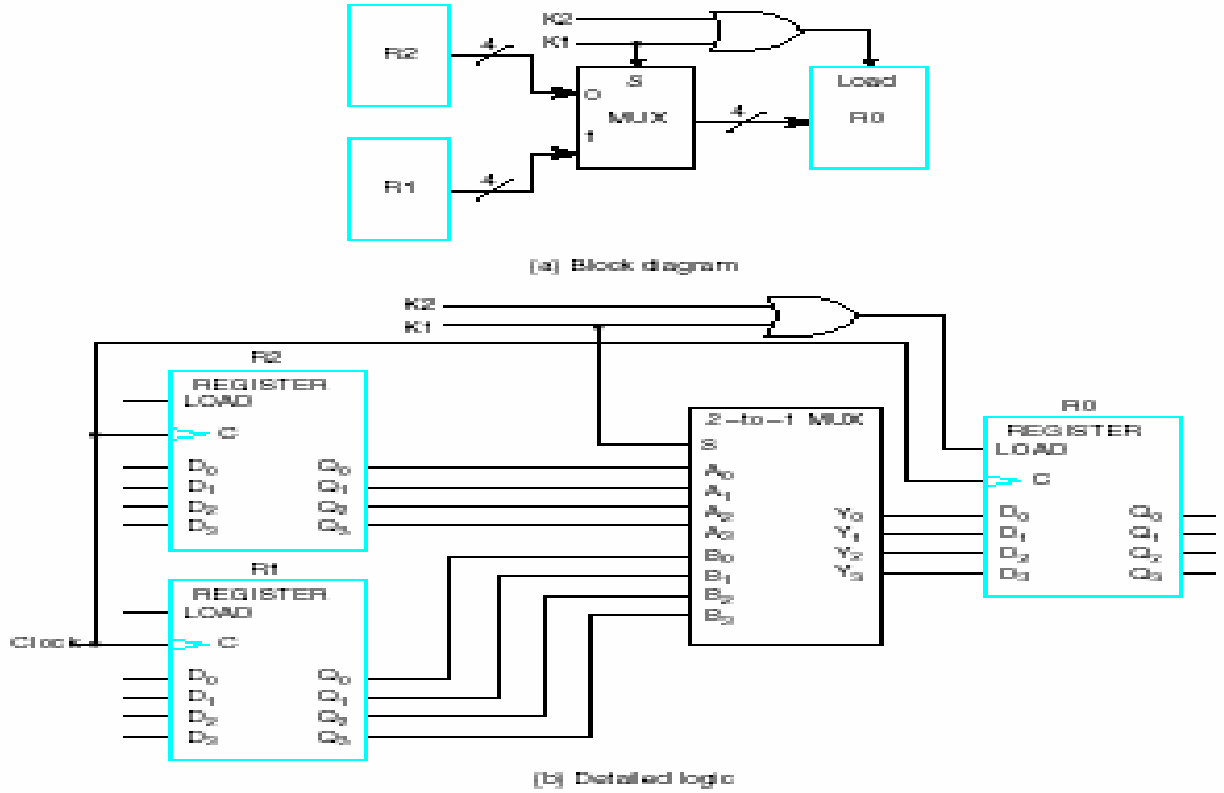
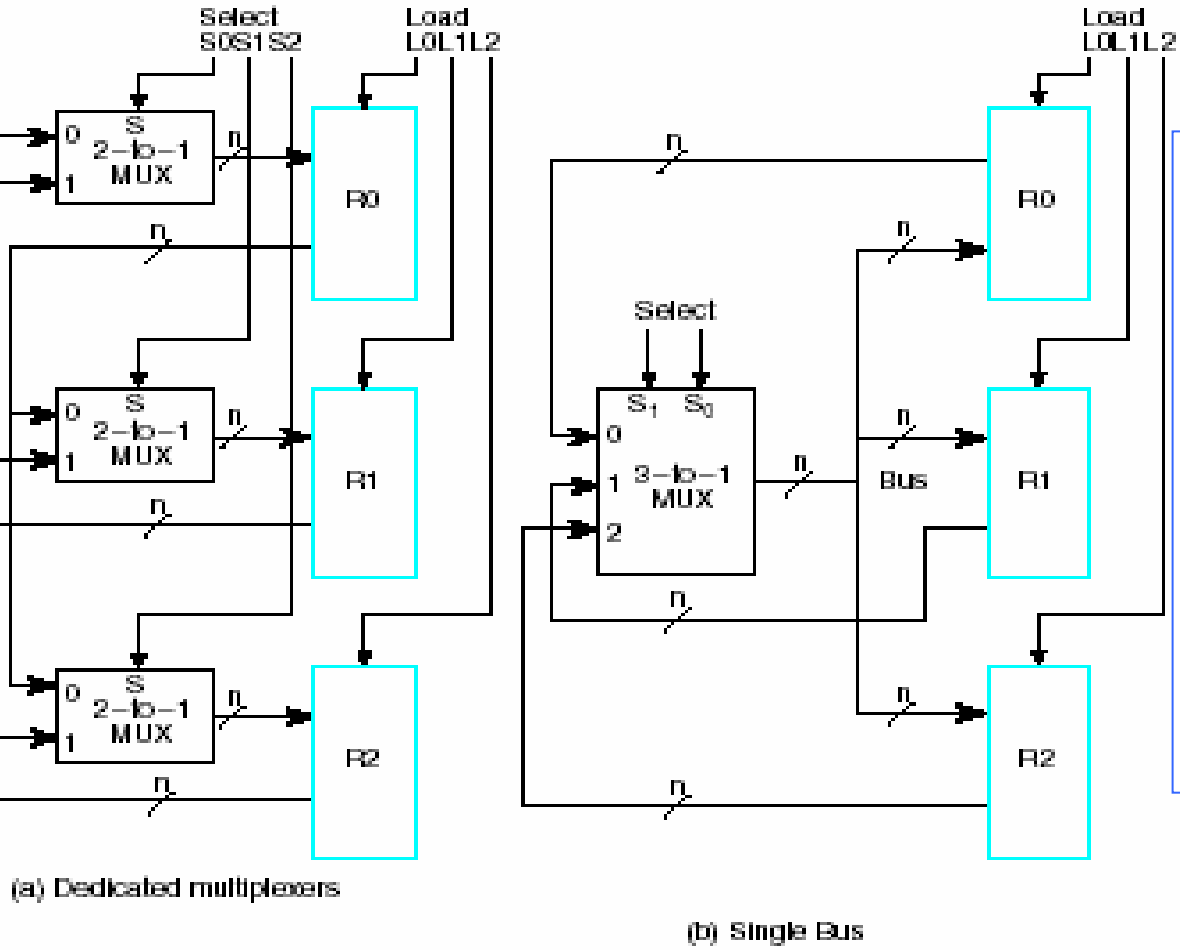


Fig. 7-5 Use of Multiplexers to Select between Two Registers

Bus-based Transfer

Observe the difference between dedicated MUX and single bus

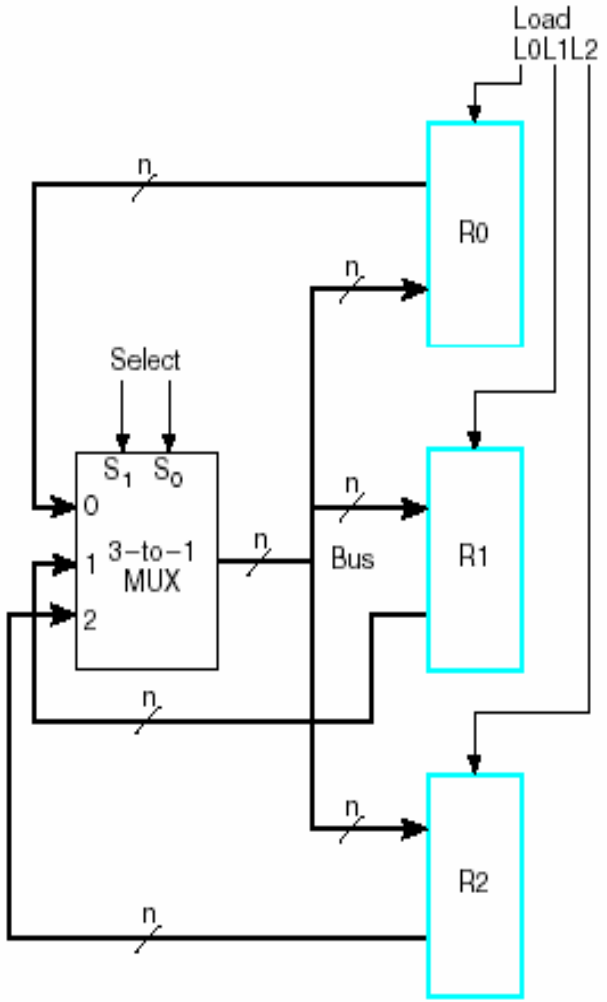
3 n-bit 2-to-1 MUX, each with its own select signal.



Same system implemented with single n-bit 3-to-1 MUX, and parallel load registers.

Fig. 7-6 Single Bus versus Dedicated Multiplexers

Bus-based Transfer …



The control input pair Select, determines the contents of single source register that will appear on the MUX outputs, on the bus. Load inputs determine the destination register on registers to be loaded with the bus data.

TABLE 7-6
Examples of Register Transfers Using the Single Bus in Figure 7-6(b)

	Select		Load		
Register Transfer	S1	S0	L2	L1	L0
$R0 \leftarrow R2$	1	0	0	0	1
$R0 \leftarrow R1, R2 \leftarrow R1$	0	1	1	0	1
$R0 \leftarrow R1, R1 \leftarrow R0$	Impossible				

e 7-6 Examples of Register Transfers Using the Single Bus in Figure 7-6(b)

Bus-based Transfer: Three-state bus

Compare: In the three-state bus, there are only three data connections to the set of register blocks for each bit of the bus. The MUX-implemented bus has six data connections per bit to the set of register blocks.

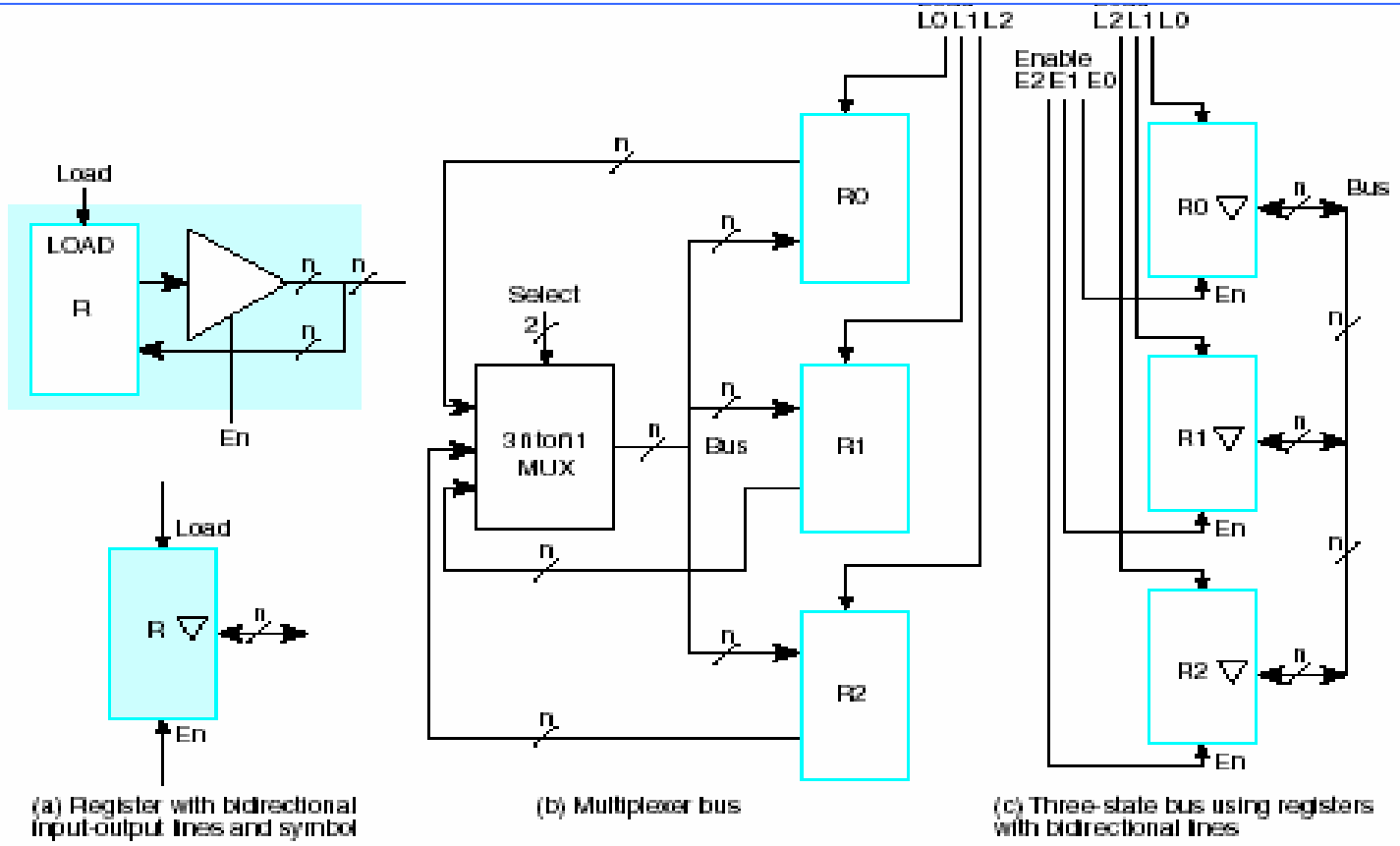


Fig. 7-7 Three-State Bus versus Multiplexer Bus

Bus-based Transfer: Memory Transfer

Read: DR ← M[AR], *Write*: M[AR] ← DR (address register, data register

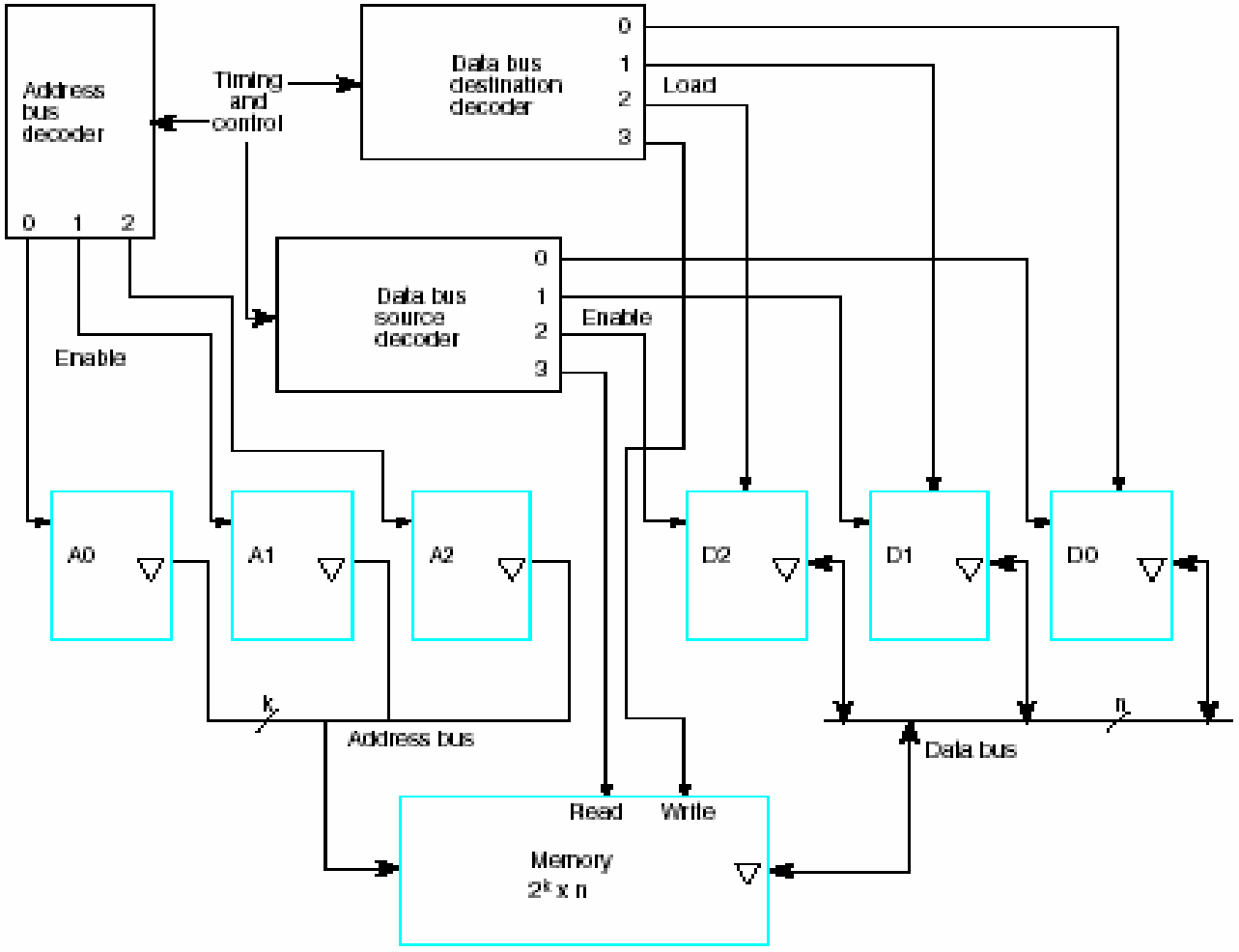


Fig. 7-8 Memory Unit Connected to Address and Data Buses

Write: M[A1] ← D2

Read: D1 ← M[A2]

Three source registers A0, A1, A2. Data bus is bidirectional. Contents of selected memory word can go to one of three registers D0, D1, D2, selected by the data bus destination decoder.