Digital Design and Computer Organization

Module 4:

Memory System: Basic Concepts, Semiconductor RAM Memories, Read Only Memories, Speed, Size, and Cost, Cache Memories Mapping Functions.

Arithmetic: Signed Operand Multiplication, Fast multiplication**,** Integer Division, Floating-point Numbers and Operations , IEEE standard for floating point numbers.

Basic Processing Unit: Some Fundamental Concepts, Execution of a Complete Instruction, Multiple Bus Organization, Hard,wired Control.

Memory System

BASIC CONCEPTS

Address	Memory Locations
16 Bit	$2^{16} = 64$ K
32 Bit	$2^{32} = 4$G (Giga)
40 Bit	$2^{40} = 1$T (Tera)

- Maximum size of memory that can be used in any computer is determined by addressing mode.
- If MAR is k-bits long then

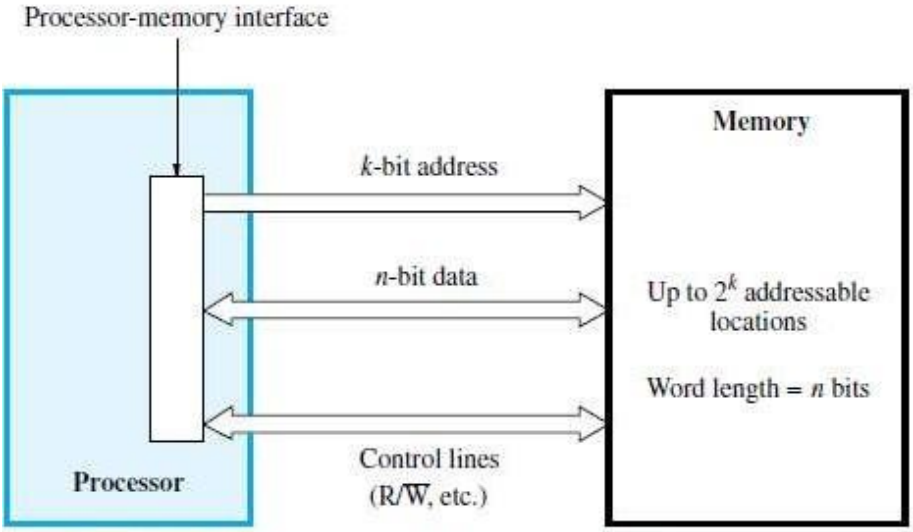


Figure 8.1 Connection of the memory to the processor.

→ memory may contain upto 2^K addressable-locations

- If MDR is n-bits long, then
  → n-bits of data are transferred between the memory and processor.
- The data-transfer takes place over the processor-bus (Figure 8.1).
- The processor-bus has
  1) Address-Line
  2) Data-line &
  3) Control-Line (R/W‟, MFC – Memory Function Completed).
- The Control-Line is used for coordinating data-transfer.
- The processor reads the data from the memory by
  → loading the address of the required memory-location into MAR and
  → setting the R/W‟ line to 1.
- The memory responds by
  → placing the data from the addressed-location onto the data-lines and
  → confirms this action by asserting MFC signal.
- Upon receipt of MFC signal, the processor loads the data from the data-lines into MDR.
- The processor writes the data into the memory-location by
  → loading the address of this location into MAR &
  → setting the R/W‟ line to 0.

- **Memory Access Time:** It is the time that elapses between
  → initiation of an operation &
  → completion of that operation.
- **Memory Cycle Time:** It is the minimum time delay that required between the initiation of the two successive memory-operations.

RAM (Random Access Memory)

- In RAM, any location can be accessed for a Read/Write-operation in fixed amount of time,

Cache Memory

- It is a small, fast memory that is inserted between
  → larger slower main-memory and
  → processor.
- It holds the currently active segments of a program and their data.

Virtual Memory

- The address generated by the processor is referred to as a **virtual/logical address**.
- The virtual-address-space is mapped onto the physical-memory where data are actuallystored.
- The mapping-function is implemented by MMU. (MMU = memory management unit).
- Only the active portion of the address-space is mapped into locations in the physical-memory.
- The remaining virtual-addresses are mapped onto the bulk storage devices such as magnetic disk.
- As the active portion of the virtual-address-space changes during program execution, theMMU
  → changes the mapping-function &
  → transfers the data between disk and memory.
- During every memory-cycle, MMU determines whether the addressed-page is in the memory.
  If the page is in the memory.
  Then, the proper word is accessed and execution proceeds.
  Otherwise, a page containing desired word is transferred from disk to memory.

- Memory can be classified as follows:
  1) RAM which can be further classified as follows:
     i) Static RAM
     ii) Dynamic RAM (DRAM) which can be further classified as synchronous & asynchronousDRAM.
  2) ROM which can be further classified as follows:
     i) PROM
     ii) EPROM
     iii) EEPROM &
     iv) Flash Memory which can be further classified as Flash Cards & Flash Drives.

SEMI CONDUCTOR RAM MEMORIES

INTERNAL ORGANIZATION OF MEMORY-CHIPS

- Memory-cells are organized in the form of array (Figure 8.2).
- Each cell is capable of storing 1-bit of information.
- Each row of cells forms a memory-word.
- All cells of a row are connected to a common line called as **Word-Line**.
- The cells in each column are connected to **Sense/Write** circuit by 2-bit-lines.
- The Sense/Write circuits are connected to data-input or output lines of the chip.
- During a write-operation, the sense/write circuit
  → receive input information &
  → store input info in the cells of the selected word.

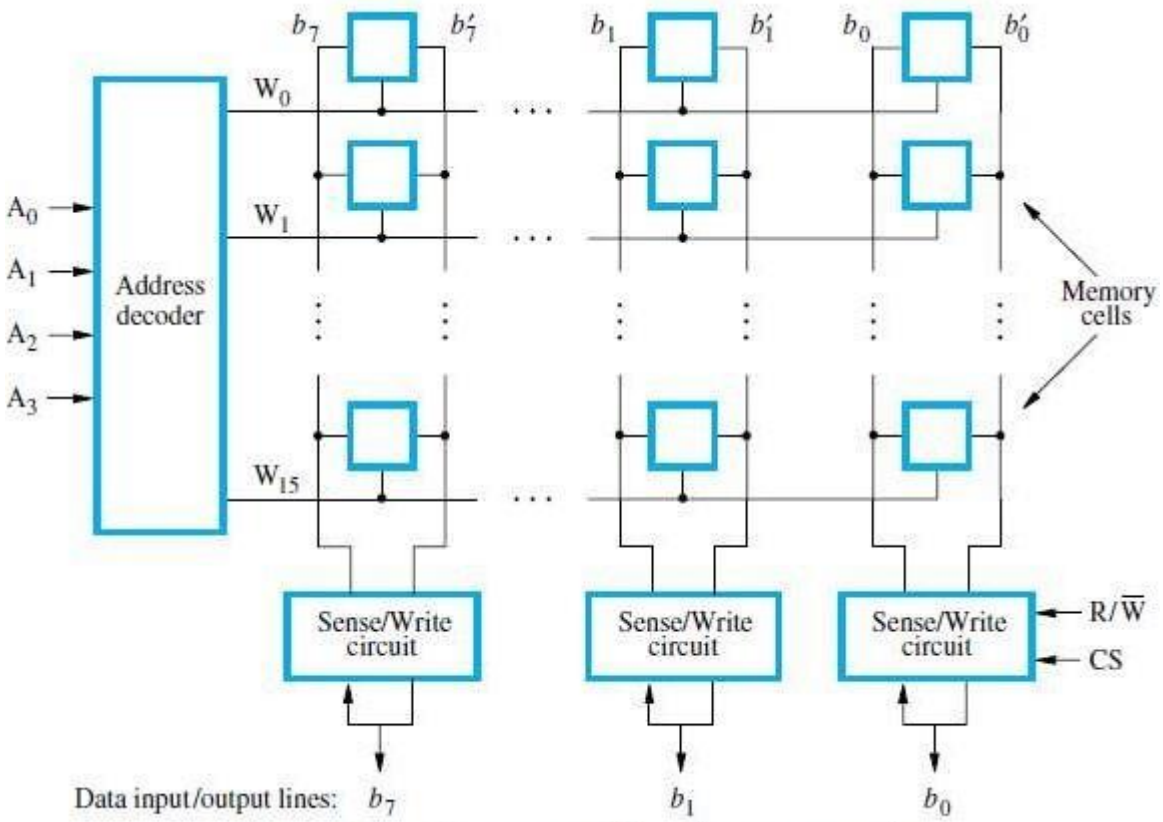


Figure 8.2 Organization of bit cells in a memory chip.

- The data-input and data-output of each Sense/Write circuit are connected to a single bidirectionaldata-line.
- Data-line can be connected to a data-bus of the computer.
- Following 2 control lines are also used:
  - **1) R/W'** □ Specifies the required operation.
  - **2) CS'** □ Chip Select input selects a given chip in the multi-chip memory-system.

Bit Organization	Requirement of external connection for address, data and control lines
128 (16x8)	14
(1024) 128x8(1k)	19

STATIC RAM (OR MEMORY)

- Memories consist of circuits capable of retaining their state as long as power is applied are known**.**

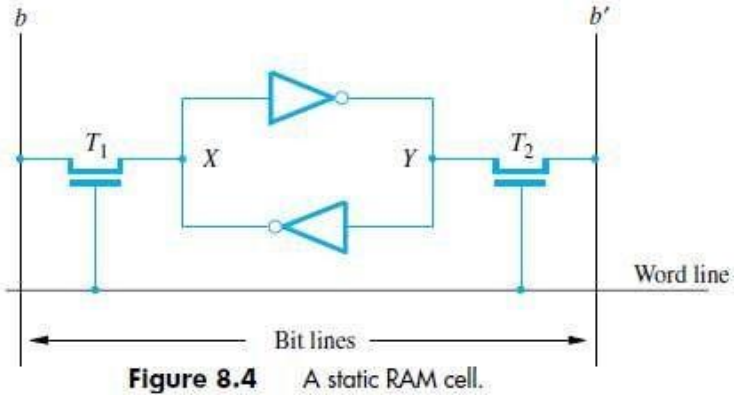


Figure 8.4 A static RAM cell.

- Two inverters are cross connected to form a latch (Figure 8.4).
- The latch is connected to 2-bit-lines by transistors T1 and T2.
- The transistors act as switches that can be opened/closed under the control of the word-line.

- When the word-line is at ground level, the transistors are turned off and the latch retain its state.

Read Operation

- To read the state of the cell, the word-line is activated to close switches T1 and T2.
- If the cell is in state 1, the signal on bit-line b is high and the signal on the bit-line b‟ is low.
- Thus, b and b‟ are complement of each other.
- Sense/Write circuit
  → monitors the state of b & b‟ and
  → sets the output accordingly.

Write Operation

- The state of the cell is set by
  → placing the appropriate value on bit-line b and its complement on b‟ and
  → then activating the word-line. This forces the cell into the corresponding state.
- The required signal on the bit-lines is generated by Sense/Write circuit.

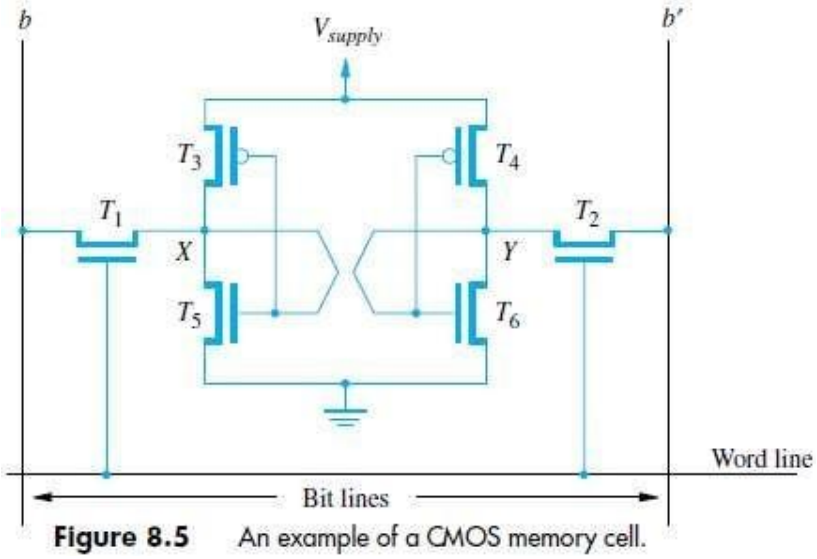


Figure 8.5 An example of a CMOS memory cell.

CMOS Cell

- Transistor pairs (T3, T5) and (T4, T6) form the inverters in the latch (Figure 8.5).
- In state 1, the voltage at point X is high by having T5, T6 ON and T4, T5 are OFF.
- Thus, T1 and T2 returned ON (Closed), bit-line b and b‟ will have high and low signals respectively.
- **Advantages**:
  1) It has low power consumption „.‟ the current flows in the cell only when the cell is active.
  2) Static RAM‟s can be accessed quickly. It access time is few nanoseconds.
- **Disadvantage:** SRAMs are said to be volatile memories „.‟ their contents are lost when poweris interrupted.

ASYNCHRONOUS DRAM

- Less expensive RAMs can be implemented if simple cells are used.
- Such cells cannot retain their state indefinitely. Hence they are called **Dynamic RAM (DRAM).**
- The information stored in a dynamic memory-cell in the form of a charge on a capacitor.
- This charge can be maintained only for tens of milliseconds.
- The contents must be periodically refreshed by restoring this capacitor charge to its full value.

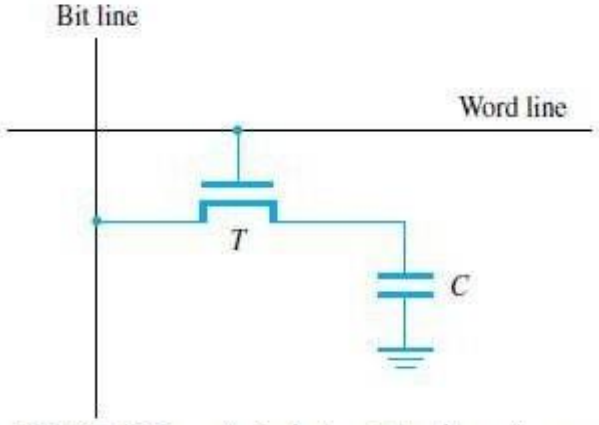


Figure 8.6 A single-transistor dynamic memory cell.

- In order to store information in the cell, the transistor T is turned „ON" (Figure 8.6).
- The appropriate voltage is applied to the bit-line which charges the capacitor.
- After the transistor is turned off, the capacitor begins to discharge.
- Hence, info. stored in cell can be retrieved correctly before threshold value of capacitor drops down.
- During a read-operation,
  → transistor is turned „ON"
  → a sense amplifier detects whether the charge on the capacitor is above the threshold value.
  - If (charge on capacitor) > (threshold value) □ Bit-line will have logic value „1".
  - If (charge on capacitor) < (threshold value) □ Bit-line will set to logic value „0".

ASYNCHRONOUS DRAM DESCRIPTION

- The 4 bit cells in each row are divided into 512 groups of 8 (Figure 5.7).
- 21 bit address is needed to access a byte in the memory. 21 bit is divided as follows:
  1) 12 address bits are needed to select a row.
     i.e. A8-0 → specifies row-address of a byte.
  2) 9 bits are needed to specify a group of 8 bits in the selected row.
     i.e. A20-9 → specifies column-address of a byte.

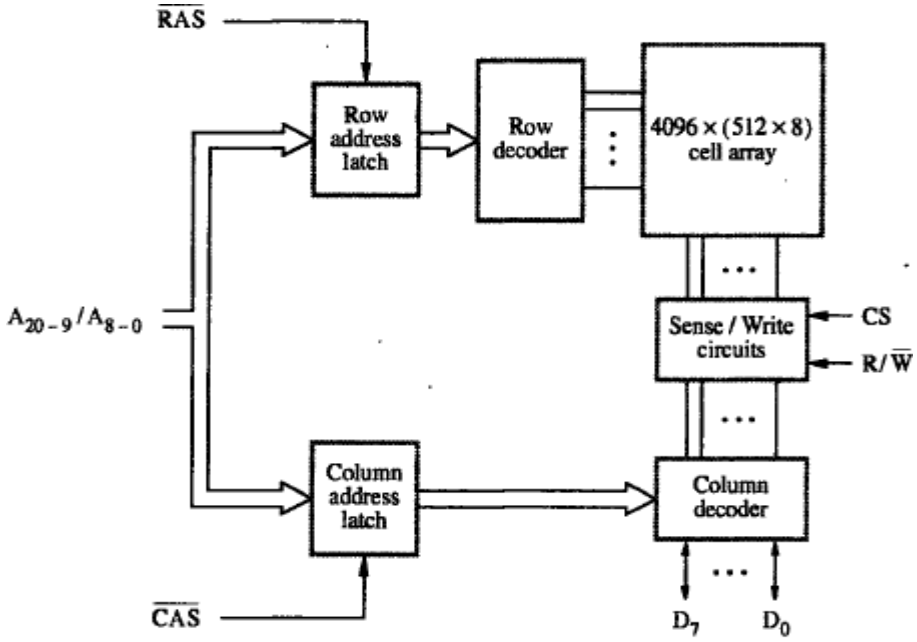


Figure 5.7 Internal organization of a 2M × 8 dynamic memory chip

- During Read/Write-operation,
  → row-address is applied first.
  → row-address is loaded into row-latch in response to a signal pulse on **RAS'** input of chip.(RAS = Row-address Strobe CAS = Column-address Strobe)
- When a Read-operation is initiated, all cells on the selected row are read and refreshed.
- Shortly after the row-address is loaded, the column-address is
  → applied to the address pins &
  → loaded into **CAS'.**
- The information in the latch is decoded.
- The appropriate group of 8 Sense/Write circuits is selected.
  R/W'=1(read-operation) □ Output values of selected circuits are transferred to data-lines D0-D7.
  R/W'=0(write-operation) □ Information on D0-D7 are transferred to the selected circuits.
- RAS" & CAS" are active-low so that they cause latching of address when they change from highto low.
- To ensure that the contents of DRAMs are maintained, each row of cells is accessed periodically.
- A special memory-circuit provides the necessary control signals RAS" & CAS" that govern the timing.
- The processor must take into account the delay in the response of the memory.

Fast Page Mode

- Transferring the bytes in sequential order is achieved by applying the consecutive sequenceof column-address under the control of successive CAS" signals.
- This scheme allows transferring a block of data at a faster rate.
- The block of transfer capability is called as *fast page mode***.**

SYNCHRONOUS DRAM

- The operations are directly synchronized with clock signal (Figure 8.8).
- The address and data connections are buffered by means of registers.
- The output of each sense amplifier is connected to a latch.
- A Read-operation causes the contents of all cells in the selected row to be loaded in these latches.
- Data held in latches that correspond to selected columns are transferred into data-output register.
- Thus, data becoming available on the data-output pins.

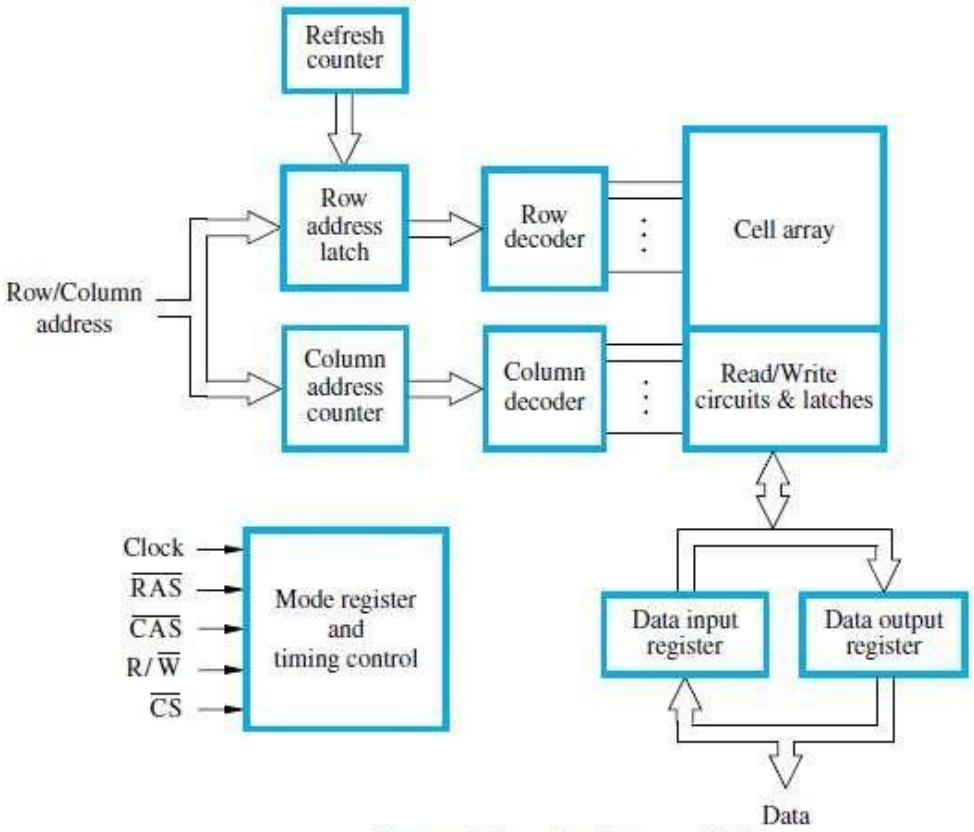


Figure 8.8 Synchronous DRAM.

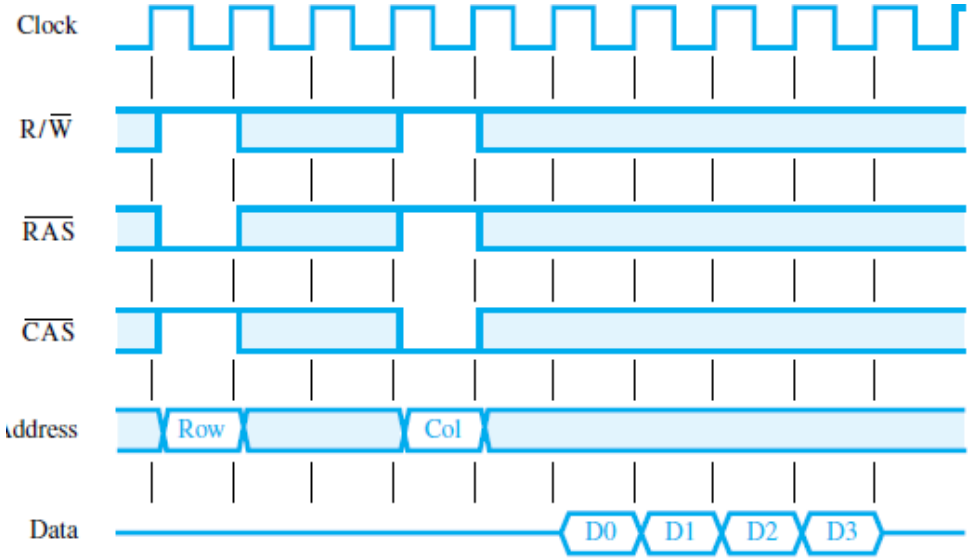


Figure 8.9 A burst read of length 4 in an SDRAM.

- First, the row-address is latched under control of RAS‟ signal (Figure 8.9).
- The memory typically takes 2 or 3 clock cycles to activate the selected row.
- Then, the column-address is latched under the control of CAS‟ signal.
- After a delay of one clock cycle, the first set of data bits is placed on the data-lines.
- SDRAM automatically increments column-address to access next 3 sets of bits in the selected row.

READ ONLY MEMORY (ROM)

- Both SRAM and DRAM chips are volatile, i.e. They lose the stored information if power is turned off.
- Many application requires non-volatile memory which retains the stored information if power isturned off.
- For ex:
  OS software has to be loaded from disk to memory i.e. it requires non-volatile memory.
- Non-volatile memory is used in embedded system.
- Since the normal operation involves only reading of stored data, a memory of this type is called ROM.
  - **At Logic value '0'** □ Transistor(T) is connected to the ground point (P).
    Transistor switch is closed & voltage on bit-line nearly drops to zero (Figure 8.11).
  - **At Logic value '1'** □ Transistor switch is open.The bit-line remains at high voltage.

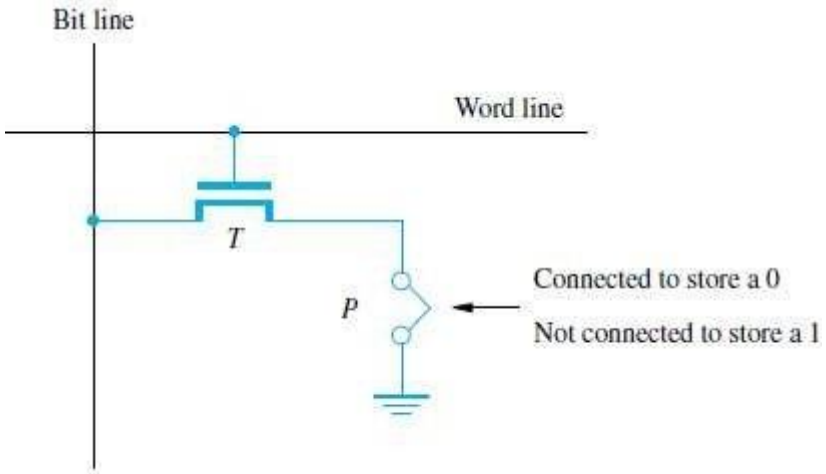


Figure 8.11 A ROM cell.

- To read the state of the cell, the word-line is activated.
- A Sense circuit at the end of the bit-line generates the proper output value.

TYPES OF ROM

- Different types of non-volatile memory are
  1) PROM
  2) EPROM
  3) EEPROM &
  4) Flash Memory (Flash Cards & Flash Drives)

PROM (PROGRAMMABLE ROM)

- PROM allows the data to be loaded by the user.
- Programmability is achieved by inserting a „fuse" at point P in a ROM cell.
- Before PROM is programmed, the memory contains all 0"s.
- User can insert 1"s at required location by burning-out fuse using high current-pulse.
- This process is irreversible.
- **Advantages:**
  1) It provides flexibility.
  2) It is faster.
  3) It is less expensive because they can be programmed directly by the user.

EPROM (ERASABLE REPROGRAMMABLE ROM)

- EPROM allows
  → stored data to be erased and
  → new data to be loaded.
- In cell, a connection to ground is always made at „P" and a special transistor is used.
- The transistor has the ability to function as
  → a normal transistor or
  → a disabled transistor that is always turned „off".

- Transistor can be programmed to behave as a permanently open switch, by injecting charge into it.
- Erasure requires dissipating the charges trapped in the transistor of memory-cells.This can be done by exposing the chip to ultra-violet light.
- **Advantages**:
  1) It provides flexibility during the development-phase of digital-system.
  2) It is capable of retaining the stored information for a long time.
- **Disadvantages:**
  1) The chip must be physically removed from the circuit for reprogramming.
  2) The entire contents need to be erased by UV light.

EEPROM (ELECTRICALLY ERASABLE ROM)

- **Advantages:**
  1) It can be both programmed and erased electrically.
  2) It allows the erasing of all cell contents selectively.
- **Disadvantage:** It requires different voltage for erasing, writing and reading the stored data.

FLASH MEMORY

- In EEPROM, it is possible to read & write the contents of a single cell.
- In Flash device, it is possible to read contents of a single cell & write entire contents of a block.
- Prior to writing, the previous contents of the block are erased.
  Eg. In MP3 player, the flash memory stores the data that represents sound.
- Single flash chips cannot provide sufficient storage capacity for embedded-system.
- **Advantages**:
  1) Flash drives have greater density which leads to higher capacity & low cost per bit.
  2) It requires single power supply voltage & consumes less power.
- There are 2 methods for implementing larger memory: 1) Flash Cards & 2) Flash Drives

Flash Cards

One way of constructing larger module is to mount flash-chips on a small card.

- Such flash-card have standard interface.
- The card is simply plugged into a conveniently accessible slot.
- Memory-size of the card can be 8, 32 or 64MB.
- Eg: A minute of music can be stored in 1MB of memory. Hence 64MB flash cards can store an hour of music.

Flash Drives

- Larger flash memory can be developed by replacing the hard disk-drive.
- The flash drives are designed to fully emulate the hard disk.
- The flash drives are solid state electronic devices that have no movable parts.
- **Advantages:**
  1) They have shorter seek & access time which results in faster response.
  2) They have low power consumption. ∴ they are attractive for battery driven application.
  3) They are insensitive to vibration.
- **Disadvantages:**
  1) The capacity of flash drive (<1GB) is less than hard disk (>1GB).
  2) It leads to higher cost per bit.
  3) Flash memory will weaken after it has been written a number of times (typically atleast 1 million times).

SPEED, SIZE COST

Characteristics	SRAM	DRAM	Magnetis Disk
Speed	Very Fast	Slower	Much slower than DRAM
Size	Large	Small	Small
Cost	Expensive	Less Expensive	Low price

Memory	Speed	Size	Cost
Registers	Very high	Lower	Very Lower
Primary cache	High	Lower	Low
Secondary cache	Low	Low	Low
Main memory	Lower than Seconadry cache	High	High
Secondary Memory	Very low	Very High	Very High

- The main-memory can be built with DRAM (Figure 8.14)
- Thus, SRAM‟s are used in smaller units where speed is of essence.
- The Cache-memory is of 2 types:
  1) Primary/Processor Cache (Level1 or L1 cache)
  - It is always located on the processor-chip.

  2) Secondary Cache (Level2 or L2 cache)
  - It is placed between the primary-cache and the rest of the memory.
- The memory is implemented using the dynamic components (SIMM, RIMM, DIMM).
- The access time for main-memory is about 10 times longer than the access time for L1 cache.

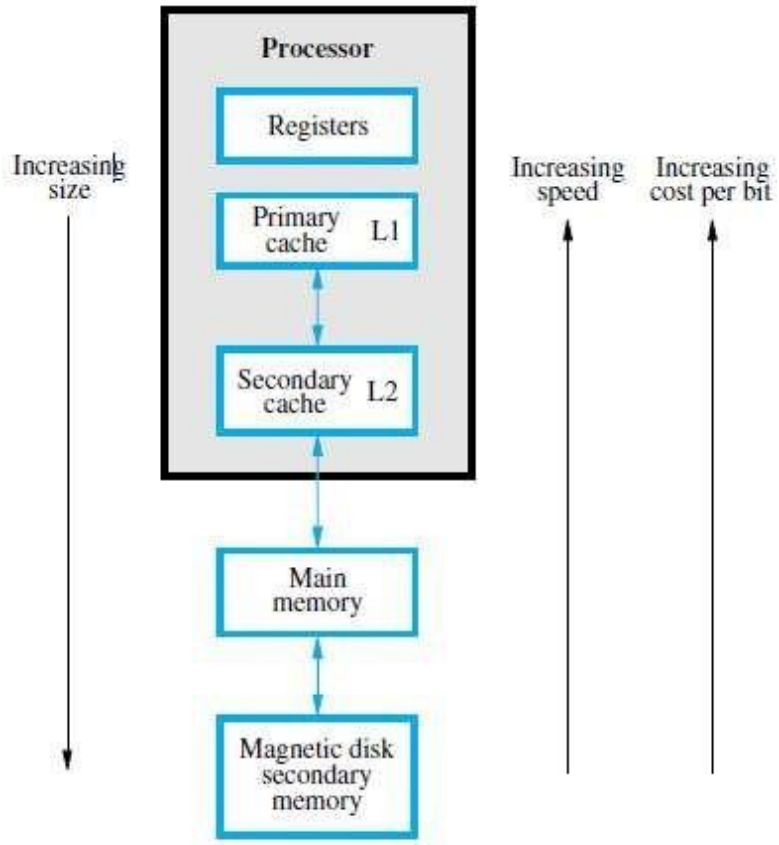


Figure 8.14 Memory hierarchy.

CACHE MEMORIES

- The effectiveness of cache mechanism is based on the property of „**Locality of Reference'.Locality of Reference**
- Many instructions in the localized areas of program are executed repeatedly during some time period
- Remainder of the program is accessed relatively infrequently (Figure 8.15).
- There are 2 types:

1) Temporal

- The recently executed instructions are likely to be executed again very soon.

2) Spatial

- Instructions in close proximity to recently executed instruction are also likely to be executed soon.

- If active segment of program is placed in cache-memory, then total execution time can be reduced.
- **Block** refers to the set of contiguous address locations of some size.
- The cache-line is used to refer to the cache-block.

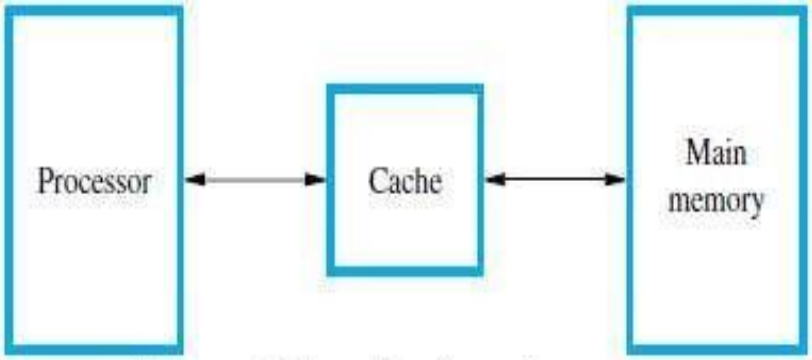


Figure 8.15 Use of a cache memory.

- The Cache-memory stores a reasonable number of blocks at a given time.
- This number of blocks is small compared to the total number of blocks available in main-memory.
- Correspondence b/w main-memory-block & cache-memory-block is specified by mapping-function.
- Cache control hardware decides which block should be removed to create space for the new block.
- The collection of rule for making this decision is called the **Replacement Algorithm.**
- The cache control-circuit determines whether the requested-word currently exists in the cache.
- The write-operation is done in 2 ways: 1) Write-through protocol & 2) Write-back protocol.

- **Write-Through Protocol**
  - Here the cache-location and the main-memory-locations are updated simultaneously.
- **Write-Back Protocol**
  - This technique is to
    → update only the cache-location &
    → mark the cache-location with associated flag bit called **Dirty/Modified Bit.**
  - The word in memory will be updated later, when the marked-block is removed from cache.

During Read-operation

- If the requested-word currently not exists in the cache, then **read-miss** will occur.
- To overcome the read miss, *Load–through/Early restart protocol* is used.

- **Load–Through Protocol**
  - The block of words that contains the requested-word is copied from the memory into cache.
  - After entire block is loaded into cache, the requested-word is forwarded to processor.
- **During Write-operation**

- If the requested-word not exists in the cache, then **write-miss** will occur.
  1) If **Write Through Protocol** is used, the information is written directly into main-memory.
  2) If **Write Back Protocol** is used,
     → then block containing the addressed word is first brought into the cache &
     → then the desired word in the cache is over-written with the new information.

MAPPING-FUNCTION

- Here we discuss about 3 different mapping-function:
  1) Direct Mapping
  2) Associative Mapping
  3) Set-Associative Mapping

DIRECT MAPPING

- The block-j of the main-memory maps onto block-j modulo-128 of the cache (Figure 8.16).
- When the memory-blocks 0, 128, & 256 are loaded into cache, the block is stored in cache-block 0. Similarly, memory-blocks 1, 129, 257 are stored in cache-block 1.
- The contention may arise when
  1) When the cache is full.
  2) When more than one memory-block is mapped onto a given cache-block position.
- The contention is resolved by
  allowing the new blocks to overwrite the currently resident-block.
- Memory-address determines placement of block in the cache.

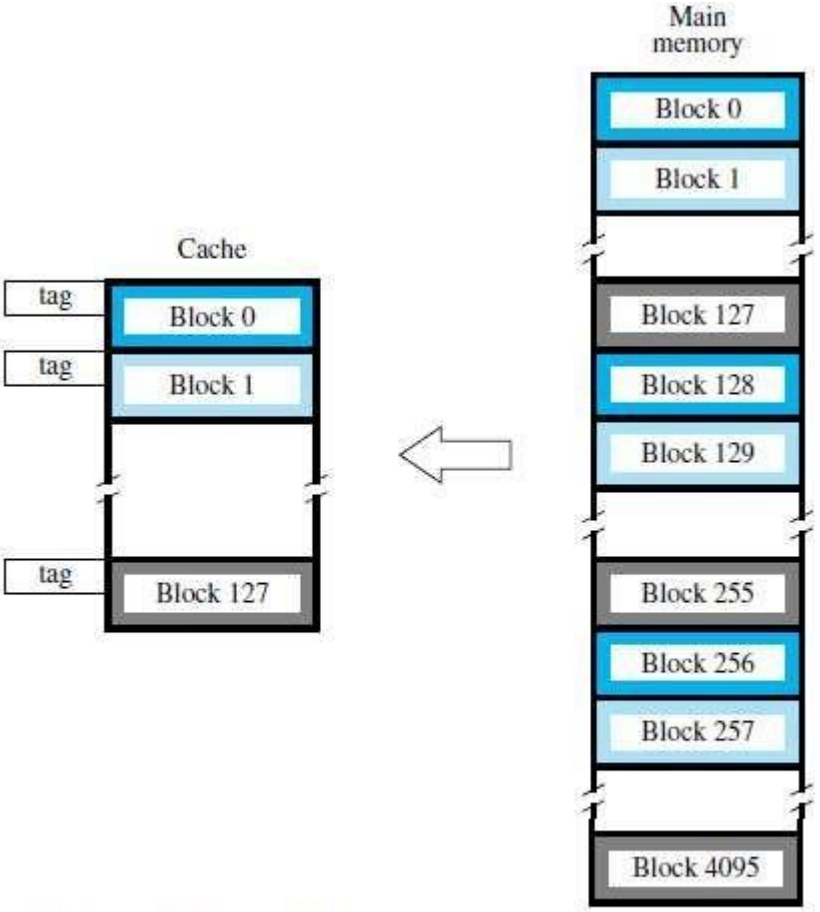


Figure 8.16 Direct-mapped cache.

- The memory-address is divided into 3 fields:

1) Low Order 4 bit field

- Selects one of 16 words in a block.

2) 7 bit cache-block field

- 7-bits determine the cache-position in which new block must be stored.

3) 5 bit Tag field

- 5-bits memory-address of block is stored in 5 tag-bits associated with cache-location.

- As execution proceeds,
  5-bit tag field of memory-address is compared with tag-bits associated with cache-location.
  If they match, then the desired word is in that block of the cache.
  Otherwise, the block containing required word must be first read from the memory.
  And then the word must be loaded into the cache.

ASSOCIATIVE MAPPING

- The memory-block can be placed into any cache-block position. (Figure 8.17).
- 12 tag-bits will identify a memory-block when it is resolved in the cache.
- Tag-bits of an address received from processor are compared to the tag-bits of each block of cache.
- This comparison is done to see if the desired block is present.

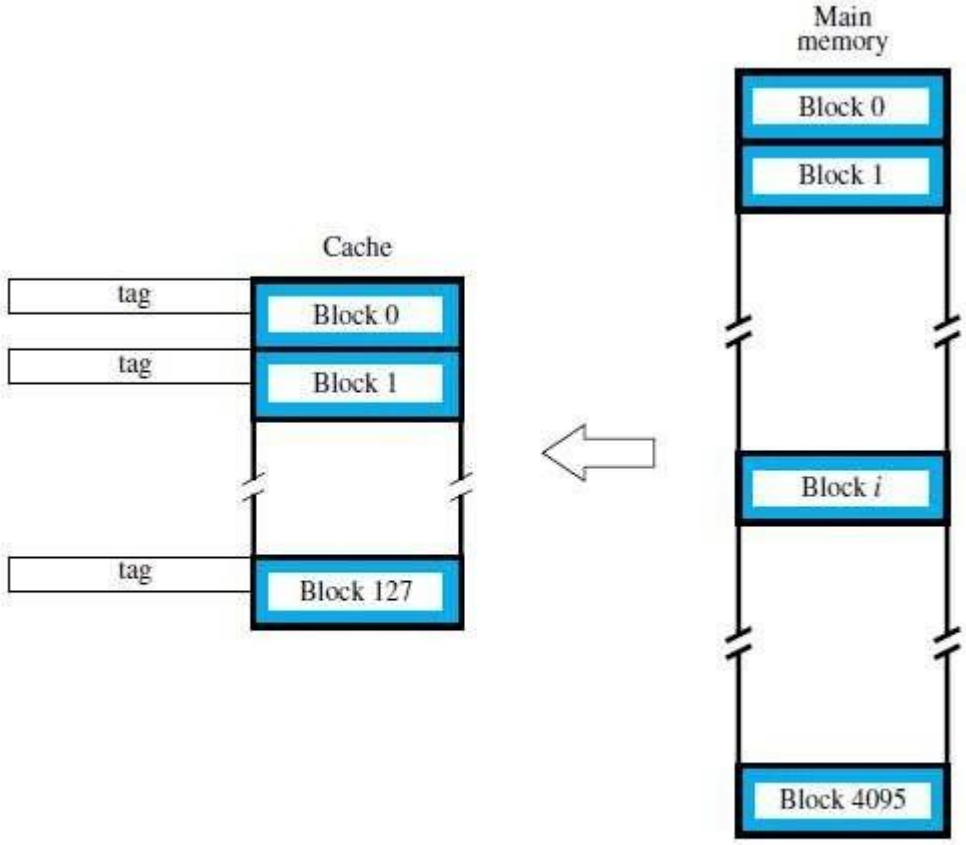


Figure 8.17 Associative-mapped cache.

- It gives complete freedom in choosing the cache-location.
- A new block that has to be brought into the cache has to replace an existing block if the cache is full.
- The memory has to determine whether a given block is in the cache.
- **Advantage:** It is more flexible than direct mapping technique.
- **Disadvantage:** Its cost is high.

SET-ASSOCIATIVE MAPPING

- It is the combination of direct and associative mapping. (Figure 8.18).
- The blocks of the cache are grouped into sets.
- The mapping allows a block of the main-memory to reside in any block of the specified set.
- The cache has 2 blocks per set, so the memory-blocks 0, 64, 128…….. 4032 maps into cache set „0".
- The cache can occupy either of the two block position within the set.
  - **6 bit set field**
    - Determines which set of cache contains the desired block.
  - **6 bit tag field**
    - The tag field of the address is compared to the tags of the two blocks of the set.
    - This comparison is done to check if the desired block is present.

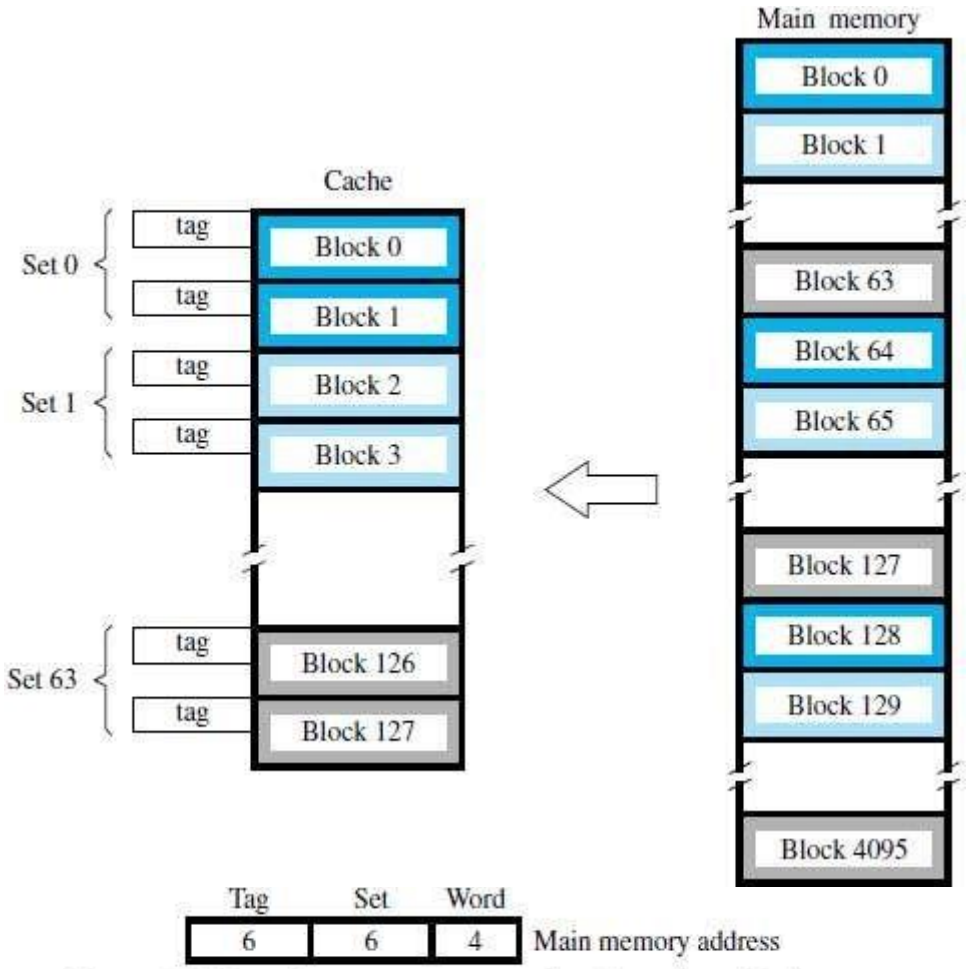


Figure 8.18 Set-associative-mapped cache with two blocks per set.

- The cache which contains 1 block per set is called **direct mapping.**
- A cache that has „k" blocks per set is called as"**k-way set associative cache**".
- Each block contains a control-bit called a **valid-bit**.
- The Valid-bit indicates that whether the block contains valid-data.
- The dirty bit indicates that whether the block has been modified during its cache residency.

Valid-bit=0 □ When power is initially applied to system.

Valid-bit=1 □ When the block is loaded from main-memory at first time.

- If the main-memory-block is updated by a source & if the block in the source is already exists in the cache, then the valid-bit will be cleared to "0".
- If Processor & DMA uses the same copies of data then it is called as **Cache Coherence Problem**.
- **Advantages:**

1) Contention problem of direct mapping is solved by having few choices for block placement.
2) The hardware cost is decreased by reducing the size of associative search.

Arithmetic

ADDITION OF POSITIVE NUMBERS

- Consider adding two 1-bit numbers.
  The sum of 1 & 1 requires the 2-bit vector 10 to represent the value 2. We say that sum is 0 and thecarry-out is 1.

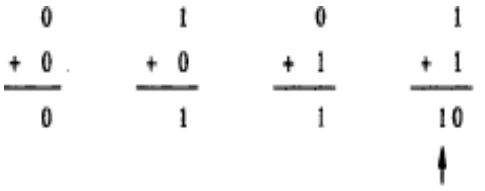


Figure 2.2 Addition of 1-bit numbers.

ADDITION & SUBTRACTION OF SIGNED NUMBERS

- Following are the two rules for addition and subtraction of n-bit signed numbers using the 2's complement representation system (Figure 1.6).
  - **Rule 1:**
    - **To Add** two numbers, add their n-bits and ignore the carry-out signal from the MSB position.
    - Result will be algebraically correct, if it lies in the range -2^{n-1} to $+2^{n-1}-1$.
  - **Rule 2:**
    - **To Subtract** two numbers X and Y (that is to perform X-Y), take the 2's complement of Y and then add it to X as in rule 1.
    - Result will be algebraically correct, if it lies in the range (2^{n-1}) to $+(2^{n-1}-1)$.
- When the result of an arithmetic operation is outside the representable-range, an arithmetic overflowis said to occur.
- To represent a signed in 2's complement form using a larger number of bits, repeat the sign bit asmany times as needed to the left. This operation is called **sign extension**.
- In 1's complement representation, the result obtained after an addition operation is not always correct. The carry-out(cn) cannot be ignored. If cn=0, the result obtained is correct. If cn=1, then a 1 must be added to the result to make it correct.

OVERFLOW IN INTEGER ARITHMETIC

- When result of an arithmetic operation is outside the representable-range, an **arithmetic overflow** is said to occur.
- For example: If we add two numbers +7 and +4, then the output sum S is 1011(□0111+0100),which is the code for -5, an incorrect result.

- An overflow occurs in following 2 cases
  1. Overflow can occur only when adding two numbers that have the same sign.
  2. The carry-out signal from the sign-bit position is not a sufficient indicator of overflow when adding signed numbers

(a)	0 0 1 0 + 0 0 1 1 0 1 0 1	(+2) (+3) (+5)	(b)	0 1 0 0 + 1 0 1 0 1 1 1 0	(+4) (−6) (−2)
(c)	1 0 1 1 + 1 1 1 0 1 0 0 1	(−5) (−2) (−7)	(d)	0 1 1 1 + 1 1 0 1 0 1 0 0	(+7) (−3) (+4)
(e)	1 1 0 1 − 1 0 0 1	(−3) (−7)	⇒	1 1 0 1 + 0 1 1 1 0 1 0 0	 (+4)
(f)	0 0 1 0 − 0 1 0 0	(+2) (+4)	⇒	0 0 1 0 + 1 1 0 0 1 1 1 0	 (−2)
(g)	0 1 1 0 − 0 0 1 1	(+6) (+3)	⇒	0 1 1 0 + 1 1 0 1 0 0 1 1	 (+3)
(h)	1 0 0 1 − 1 0 1 1	(−7) (−5)	⇒	1 0 0 1 + 0 1 0 1 1 1 1 0	 (−2)
(i)	1 0 0 1 − 0 0 0 1	(−7) (+1)	⇒	1 0 0 1 + 1 1 1 1 1 0 0 0	 (−8)
(j)	0 0 1 0 − 1 1 0 1	(+2) (−3)	⇒	0 0 1 0 + 0 0 1 1 0 1 0 1	 (+5)

Figure 1.6 2's-complement Add and Subtract operations.

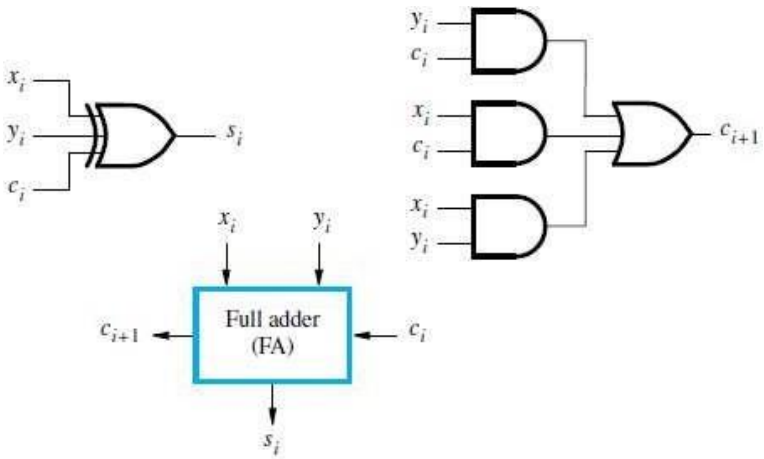
ADDITION & SUBTRACTION OF SIGNED NUMBERSn-BIT RIPPLE CARRY ADDER

- A cascaded connection of n full-adder blocks can be used to add 2-bit numbers.
- Since carries must propagate (or ripple) through cascade, the configuration is called an n-bit ripplecarry adder (Figure 9.1).

x_i	y_i	Carry-in c_i	Sum s_i	Carry-out c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$s_i = \overline{x}_i\overline{y}_i c_i + \overline{x}_i y_i \overline{c}_i + x_i\overline{y}_i\overline{c}_i + x_i y_i c_i = x_i \oplus y_i \oplus c_i$$

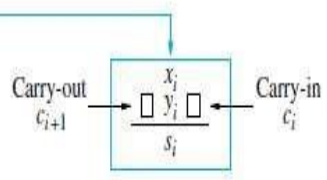
$$c_{i+1} = y_i c_i + x_i c_i + x_i y_i$$



(a) Logic for a single stage

Example:

$$\begin{array}{rcl} X & 7 & 0\ 1\ \boxed{1\ 1}\ 1 \\ +Y & +6 & +_0\,0_1\,1_1\,1_0\,0_0 \\ \hline Z & 13 & 1\ 1\ \boxed{0\ 1} \end{array}$$



Legend for stage i

Figure 9.1 Logic specification for a stage of binary addition.

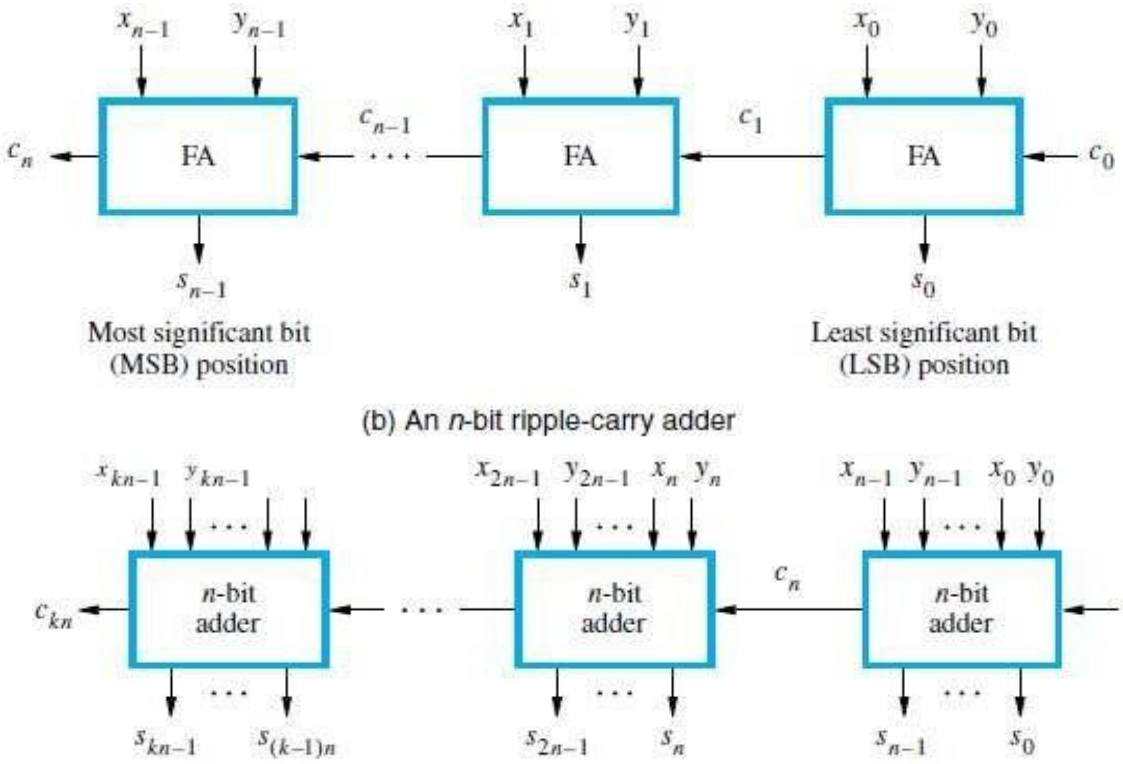


Figure 9.2 Logic for addition of binary numbers.

ADDITION/SUBTRACTION LOGIC UNIT

- The n-bit adder can be used to add 2's complement numbers X and Y (Figure 9.3).
- **Overflow** can only occur when the signs of the 2 operands are the same.
- In order to perform the subtraction operation X-Y on 2's complement numbers X and Y; we form the2's complement of Y and add it to X.
- Addition or subtraction operation is done based on value applied to the Add/Sub input control-line.
- Control-line=0 for addition, applying the Y vector unchanged to one of the adder inputs.Control-line=1 for subtraction, the Y vector is 2's complemented.

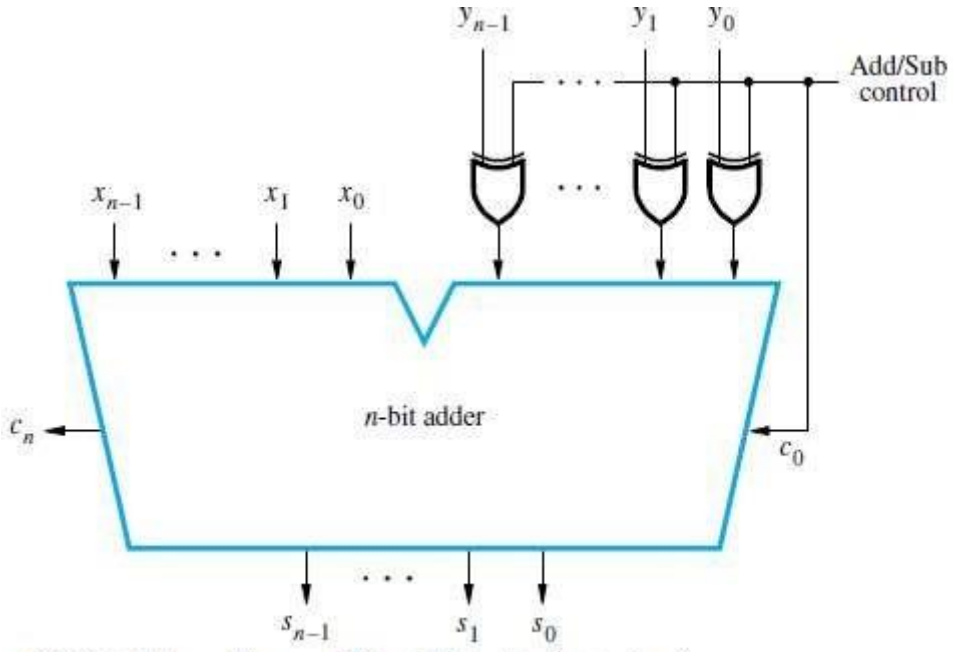
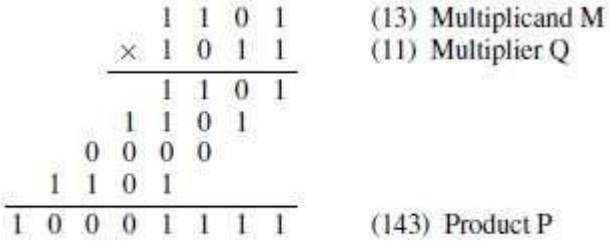
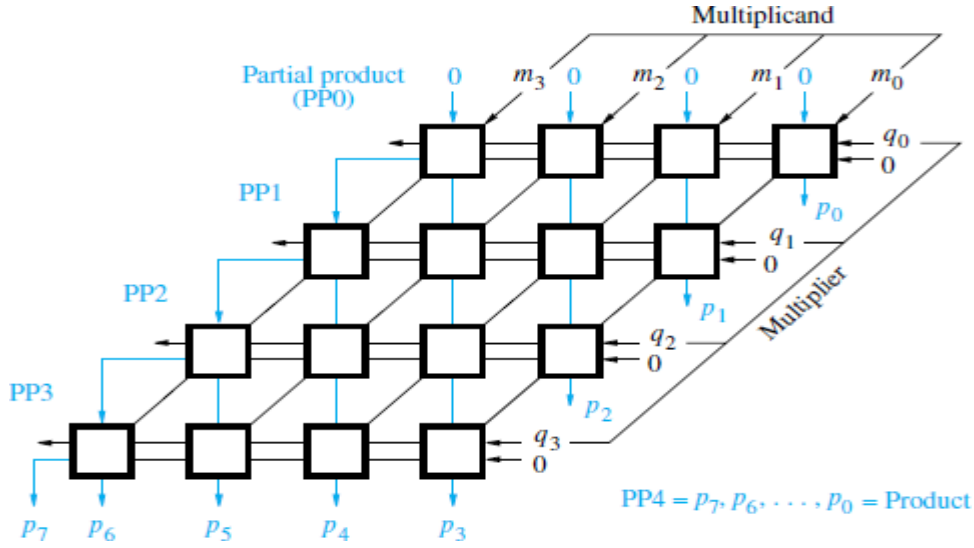


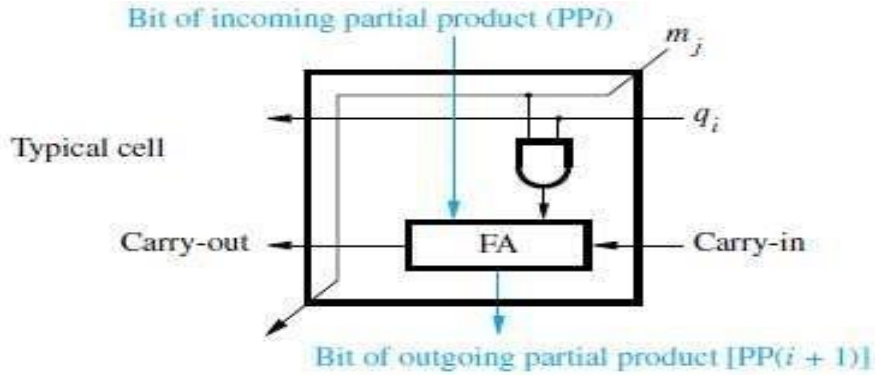
Figure 9.3 Binary addition/subtraction logic circuit.

MULTIPLICATION OF POSITIVE NUMBERS



(a) Manual multiplication algorithm





(b) Array implementation

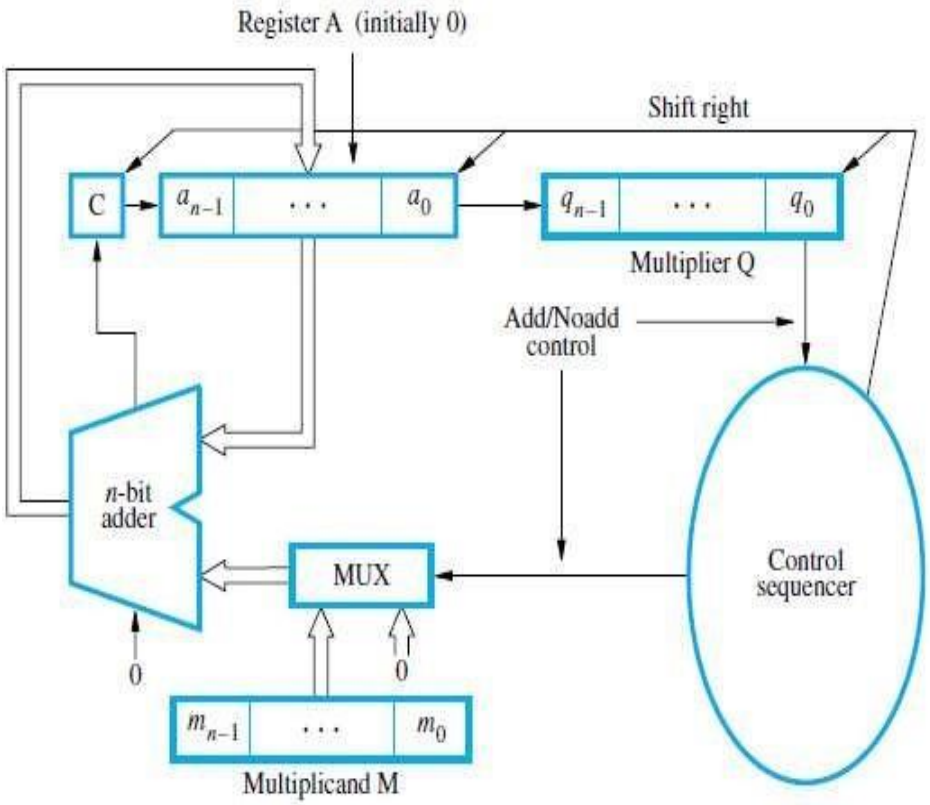
Figure 9.6 Array multiplication of unsigned binary operands.

ARRAY MULTIPLICATION

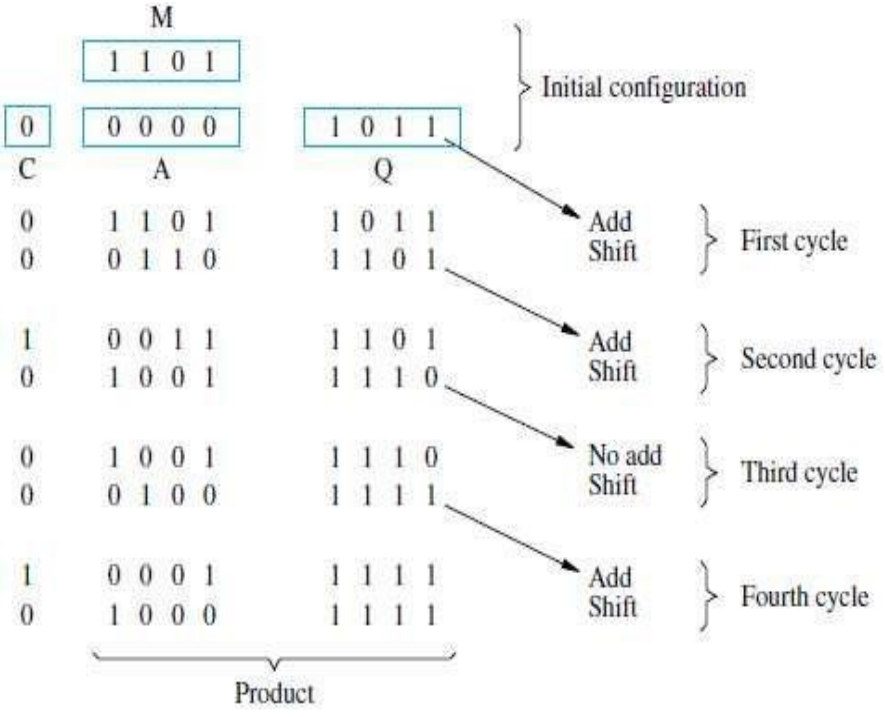
- The main component in each cell is a full adder(FA)..
  The AND gate in each cell determines whether a multiplicand bit mj, is added to the incoming partial-product bit, based on the value of the multiplier bit qi (Figure9.6).

SEQUENTIAL CIRCUIT BINARY MULTIPLIER

- Registers A and Q combined hold PPi(partial product)
  while the multiplier bit qi generates the signal Add/Noadd.
- The carry-out from the adder is stored in flip-flop C (Figure 9.7).
- Procedure for multiplication:
  1) Multiplier is loaded into register Q, Multiplicand is loaded into register M and C & A are cleared to 0.
  2) If q0=1, add M to A and store sum in A. Then C, A and Q are shifted right one bit-position.If q0=0, no addition performed and C, A & Q are shifted right one bit-position.
  3) After n cycles, the high-order half of the product is held in register A andthe low-order half is held in register Q.



(a) Register configuration



(b) Multiplication example

Figure 9.7 Sequential circuit binary multiplier.

SIGNED OPERAND MULTIPLICATIONBOOTH ALGORITHM

- This algorithm
  → generates a 2n-bit product
  → treats both positive & negative 2's-complement n-bit operands uniformly(Figure 9.9-9.12).
- Attractive feature: This algorithm achieves some efficiency in the number of addition required whenthe multiplier has a few large blocks of 1s.
- This algorithm suggests that we can reduce the number of operations required for multiplication by representing multiplier as a difference between 2 numbers.

  For e.g. multiplier(Q) 14(001110) can be represented as010000 (16)
  -000010 (2)
  001110 (14)

- Therefore, product P=M*Q can be computed by adding 2^4 times the M to the 2's complement of 2^1 times the M.

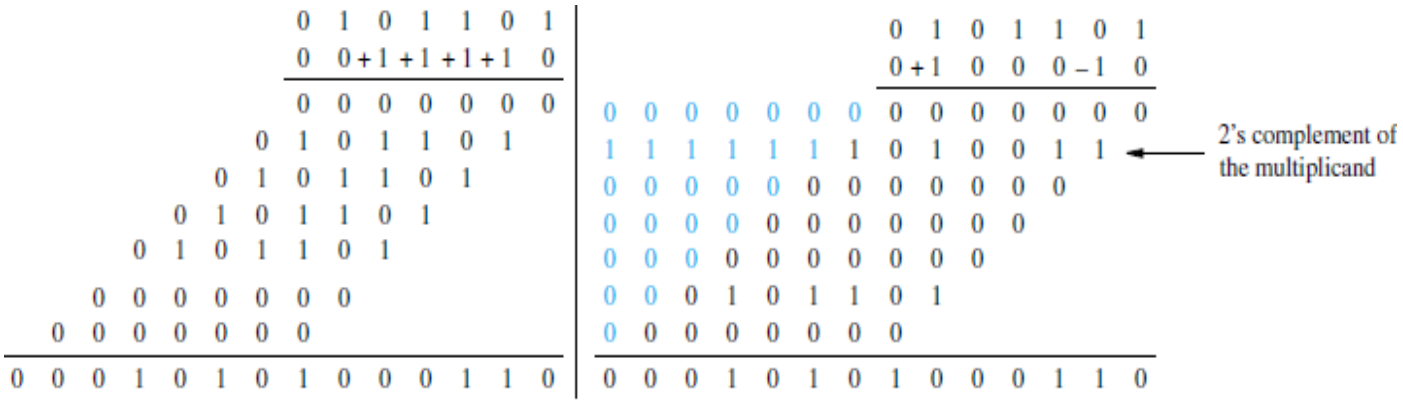


Figure 9.9 Normal and Booth multiplication schemes.

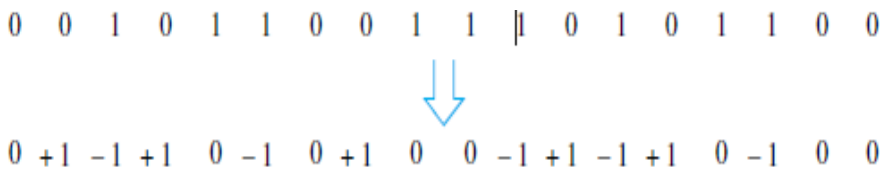


Figure 9.10 Booth recoding of a multiplier.

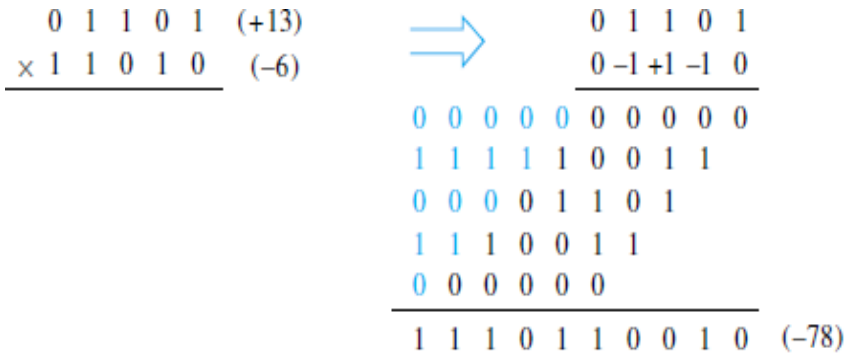


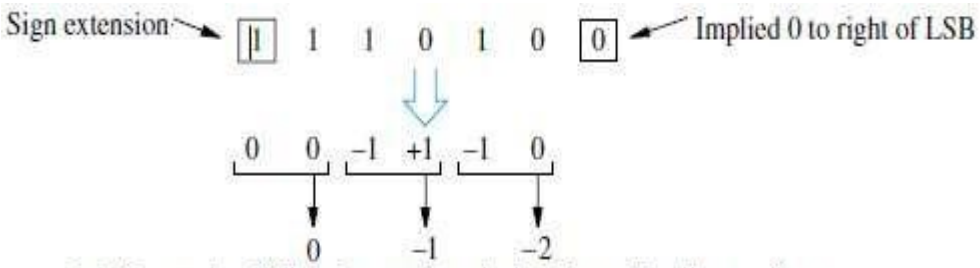
Figure 9.11 Booth multiplication with a negative multiplier.

Multiplier		Version of multiplicand selected by bit i
Bit i	Bit $i-1$	
0	0	$0 \times M$
0	1	$+1 \times M$
1	0	$-1 \times M$
1	1	$0 \times M$

Figure 9.12 Booth multiplier recoding table.

FAST MULTIPLICATION BIT-PAIR RECODING OF MULTIPLIERS

- This method
  - → derived from the booth algorithm
  - → reduces the number of summands by a factor of 2
- Group the Booth-recoded multiplier bits in pairs. (Figure 9.14 & 9.15).
- The pair (+1 -1) is equivalent to the pair (0 +1).



(a) Example of bit-pair recoding derived from Booth recoding

Multiplier bit-pair		Multiplier bit on the right	Multiplicand selected at position i
$i+1$	i	$i-1$	
0	0	0	$0 \times M$
0	0	1	$+1 \times M$
0	1	0	$+1 \times M$
0	1	1	$+2 \times M$
1	0	0	$-2 \times M$
1	0	1	$-1 \times M$
1	1	0	$-1 \times M$
1	1	1	$0 \times M$

(b) Table of multiplicand selection decisions

Figure 9.14 Multiplier bit-pair recoding.

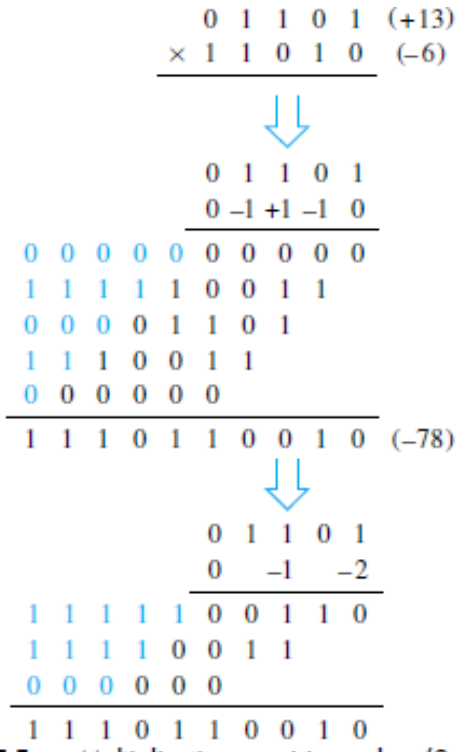
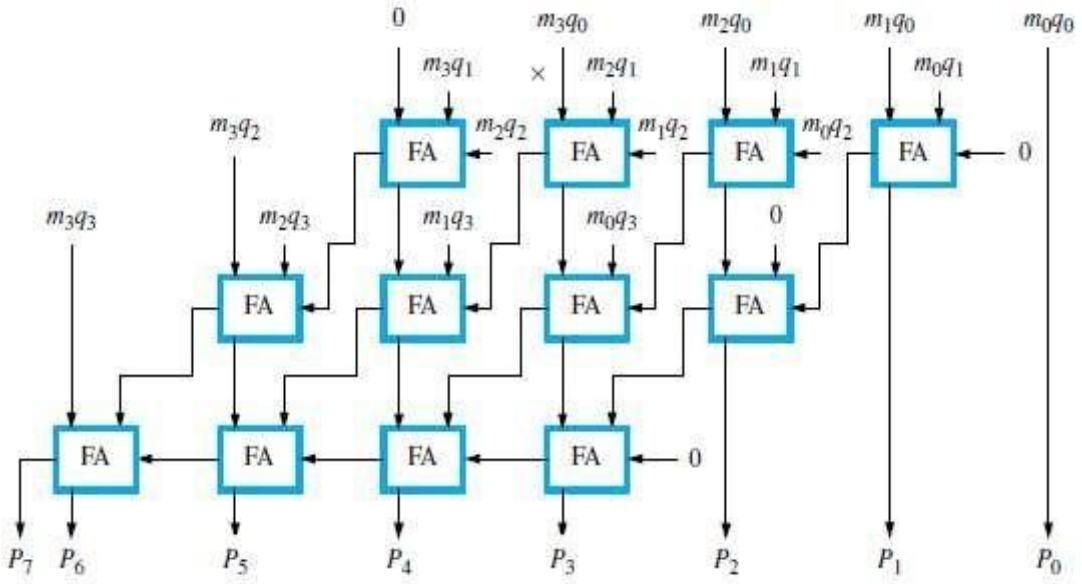


Figure 9.15 Multiplication requiring only $n/2$ summands.

CARRY-SAVE ADDITION OF SUMMANDS

- Consider the array for 4*4 multiplication. (Figure 9.16 & 9.18).
- Instead of letting the carries ripple along the rows, they can be "saved" and introduced into the next row, at the correct weighted positions.



(b) Carry-save array

Figure 9.16 carry-save arrays for a 4 × 4 multiplier.

```
                1  0  1  1  0  1   (45)  M
             ×  1  1  1  1  1  1   (63)  Q
             ------------------------------
                1  0  1  1  0  1         A
             1  0  1  1  0  1            B
          1  0  1  1  0  1               C
       1  0  1  1  0  1                  D
    1  0  1  1  0  1                     E
 1  0  1  1  0  1                        F
 ------------------------------------------
 0  1  1  0  0  0  1  0  0  1  1  (2,835) Product
```

Figure 9.17 A multiplication example used to illustrate carry-save addition as shown in Figure 9.18.

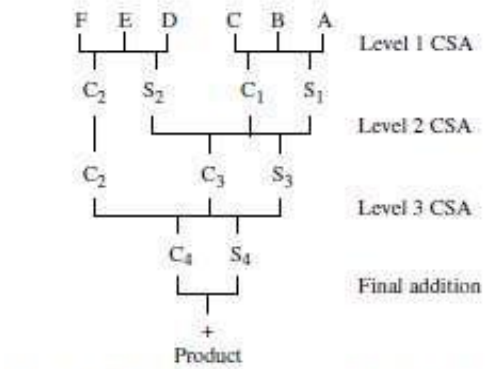


Figure 9.19 Schematic representation of the carry-save addition operations in Figure 9.18.

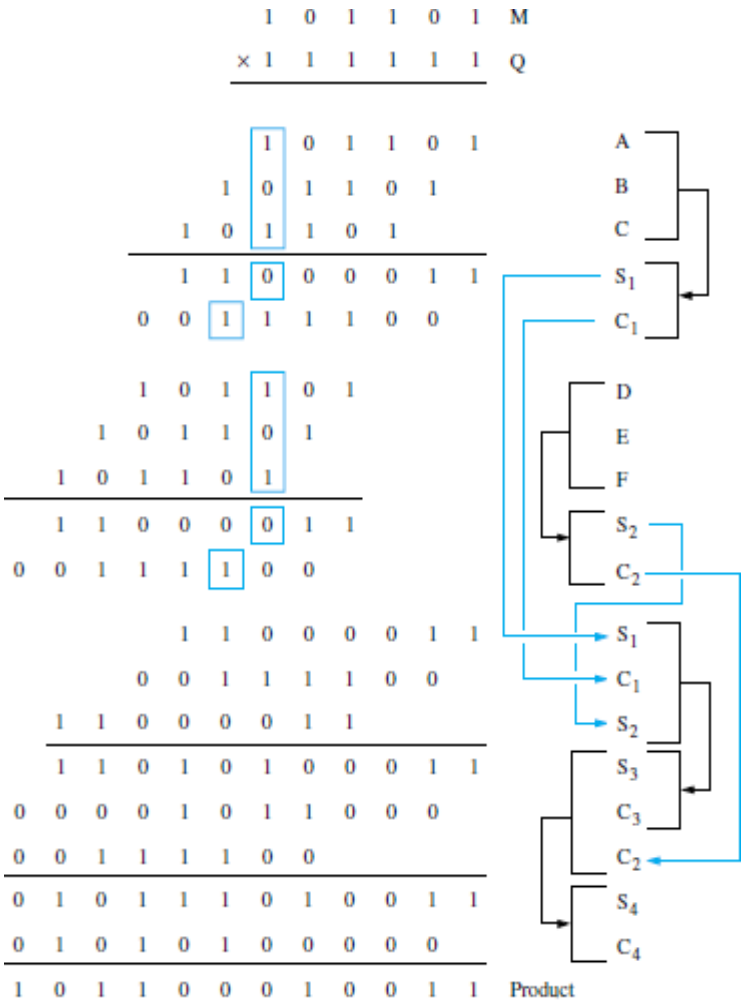


Figure 9.18 The multiplication example from Figure 9.17 performed using carry-save addition.

- The full adder is input with three partial bit products in the first row.
- Multiplication requires the addition of several summands.
- CSA speeds up the addition process.
- Consider the array for 4x4 multiplication shown in fig 9.16.
- First row consisting of just the AND gates that implement the bit products m3q0, m2q0, m1q0 and m0q0.
- The delay through the carry-save array is somewhat less than delay through the ripple-carry array. This is because the S and C vector outputs from each row are produced in parallel in one full-adder delay.
- Consider the addition of many summands in fig 9.18.
- Group the summands in threes and perform carry-save addition on each of these groups in parallel to generate a set of S and C vectors in one full-adder delay
- Group all of the S and C vectors into threes, and perform carry-save addition on them, generating a further set of S and C vectors in one more full-adder delay
- Continue with this process until there are only two vectors remaining
- They can be added in a RCA or CLA to produce the desired product.
- When the number of summands is large, the time saved is proportionally much greater.
- Delay: AND gate + 2 gate/CSA level + CLA gate delay, Eg., 6 bit number require 15 gate delay,array 6x6 require 6(n-1)-1 = 29 gate Delay.
- In general, CSA takes 1.7 *log*2k-1.7 levels of CSA to reduce k summands.

INTEGER DIVISION

- An n-bit positive-divisor is loaded into register M.
  An n-bit positive-dividend is loaded into register Q at the start of the operation.
  Register A is set to 0 (Figure 9.21).
- After division operation, the n-bit quotient is in register Q, and
  the remainder is in register A.

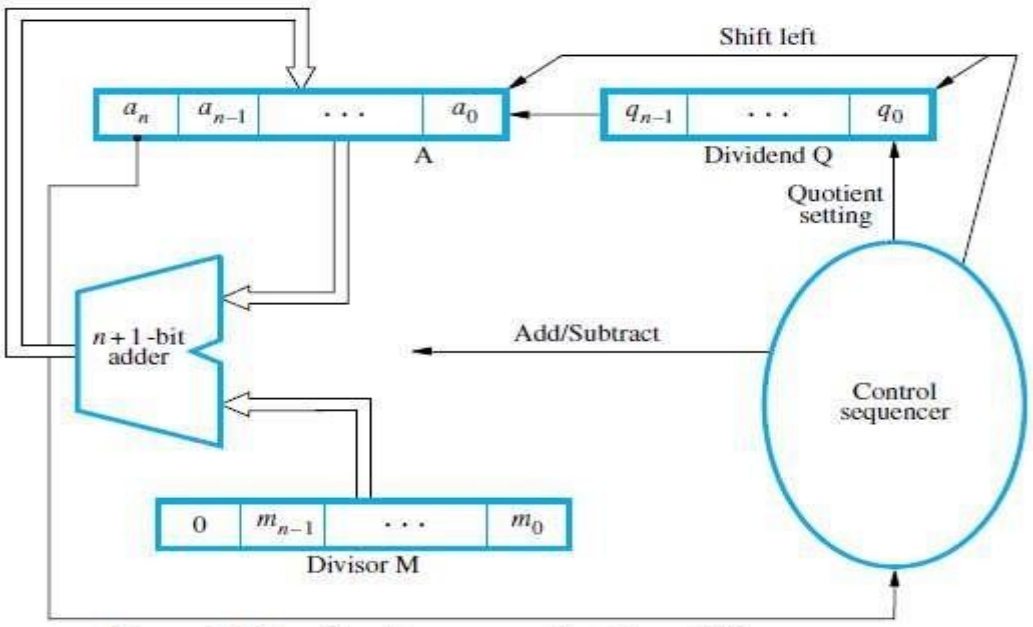


Figure 9.23 Circuit arrangement for binary division.

```
      21                    10101
13 ) 274          1101 ) 100010010
     26                    1101
     14                   10000
     13                    1101
     --                   ----
      1                    1110
                           1101
                           ----
                              1
```

Figure 9.22 Longhand division examples.

NON-RESTORING DIVISION

- Procedure:

Step 1: Do the following n times

i) If the sign of A is 0, shift A and Q left one bit position and subtract M from A;otherwise, shift A and Q left and add M to A (Figure 9.23).

ii) Now, if the sign of A is 0, set q0 to 1; otherwise set q0 to 0.Step 2: If the sign of A is 1, add M to A (restore).

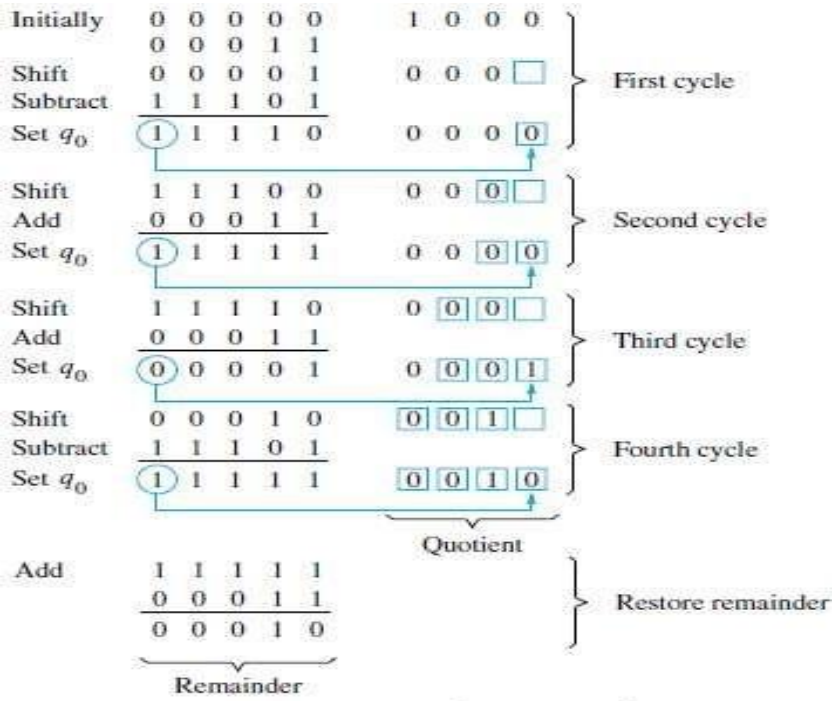


Figure 9.25 A non-restoring division example.

RESTORING DIVISION

- Procedure: Do the following n times

1) Shift A and Q left one binary position (Figure 9.22).
2) Subtract M from A, and place the answer back in A
3) If the sign of A is 1, set q0 to 0 and add M back to A(restore A).If the sign of A is 0, set q0 to 1 and no restoring done.

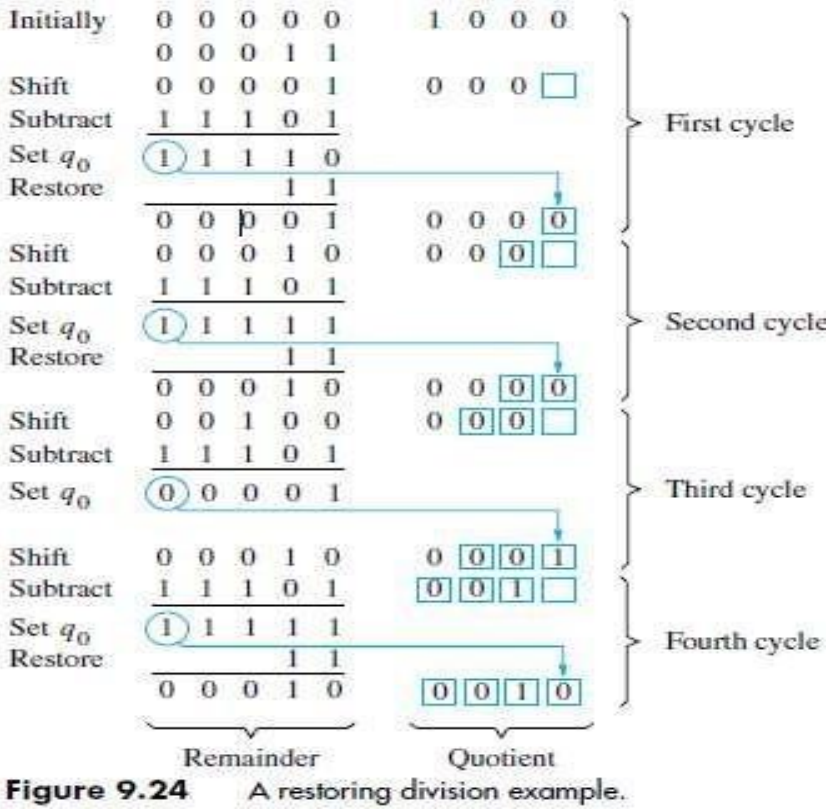
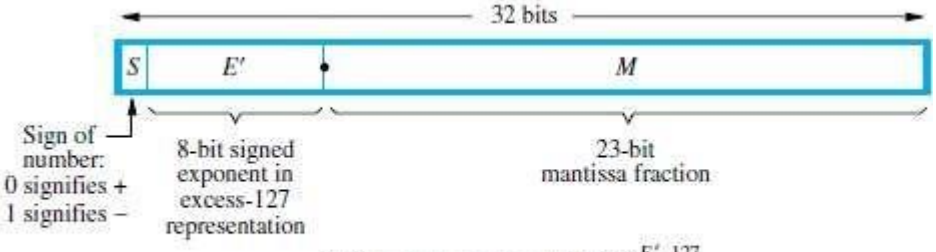


Figure 9.24 A restoring division example.

FLOATING-POINT NUMBERS & OPERATIONS IEEE STANDARD FOR FLOATING POINT NUMBERS

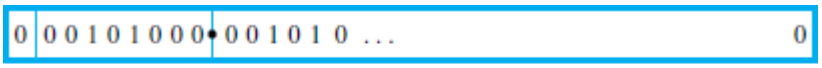
- Single precision representation occupies a single 32-bit word.
  The scale factor has a range of 2^{-126} to 2^{+127} (which is approximately equal to 10^{+38}).
- The 32 bit word is divided into 3 fields: sign(1 bit), exponent(8 bits) and mantissa(23 bits).
- Signed exponent=E.
  Unsigned exponent E'=E+127. Thus, E' is in the range 0<E'<255.
- The last 23 bits represent the mantissa. Since binary normalization is used, the MSB of the mantissais always equal to 1. (M represents fractional-part).
- The 24-bit mantissa provides a precision equivalent to about 7 decimal-digits (Figure 9.24).
- Double precision representation occupies a single 64-bit word. And E' is in the range 1<E'<2046.



Value represented = $\pm 1.M \times 2^{E'-127}$

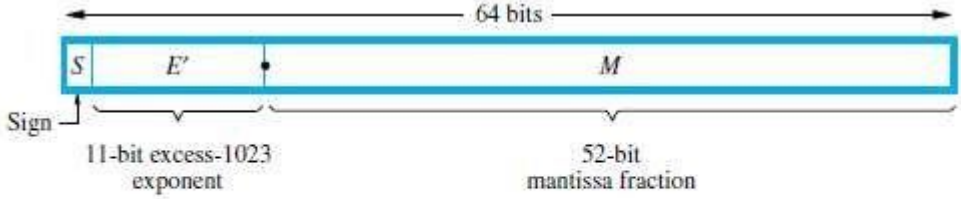
(a) Single precision

- The 53-bit mantissa provides a precision equivalent to about 16 decimal-digits.



Value represented = $1.001010\ldots0 \times 2^{-87}$

(b) Example of a single-precision number



Value represented = $\pm 1.M \times 2^{E'-1023}$

(c) Double precision

Figure 9.26 IEEE standard floating-point formats.

ARITHMETIC OPERATIONS ON FLOATING-POINT NUMBERS

- **Multiply Rule**
  1) Add the exponents & subtract 127.
  2) Multiply the mantissas & determine sign of the result.
  3) Normalize the resulting value if necessary.
- **Divide Rule**
  1) Subtract the exponents & add 127.
  2) Divide the mantissas & determine sign of the result.
  3) Normalize the resulting value if necessary.
- **Add/Subtract Rule**
  1) Choose the number with the smaller exponent & shift its mantissa right a number of stepsequal to the difference in exponents(n).
  2) Set exponent of the result equal to larger exponent.
  3) Perform addition/subtraction on the mantissas & determine sign of the result.
  4) Normalize the resulting value if necessary.

IMPLEMENTING FLOATING-POINT OPERATIONS

- First compare exponents to determine how far to shift the mantissa of the number with the smaller exponent.
- The shift-count value n
  → is determined by 8 bit subtractor &
  → is sent to SHIFTER unit.
- In step 1, sign is sent to SWAP network (Figure 9.26).
  If sign=0, then EA>EB and mantissas MA & MB are sent straight through SWAP network. If sign=1, then EA<EB and the mantissas are swapped before they are sent to SHIFTER.
- In step 2, 2:! MUX is used. The exponent of result E is tentatively determined as EA if EA>EB or EB if EA<EB
- In step 3, CONTROL logic
  → determines whether mantissas are to be added or subtracted.
  → determines sign of the result.
- In step 4, result of step 3 is normalized. The number of leading zeros in M determines number of bit shifts(X) to be applied to M.

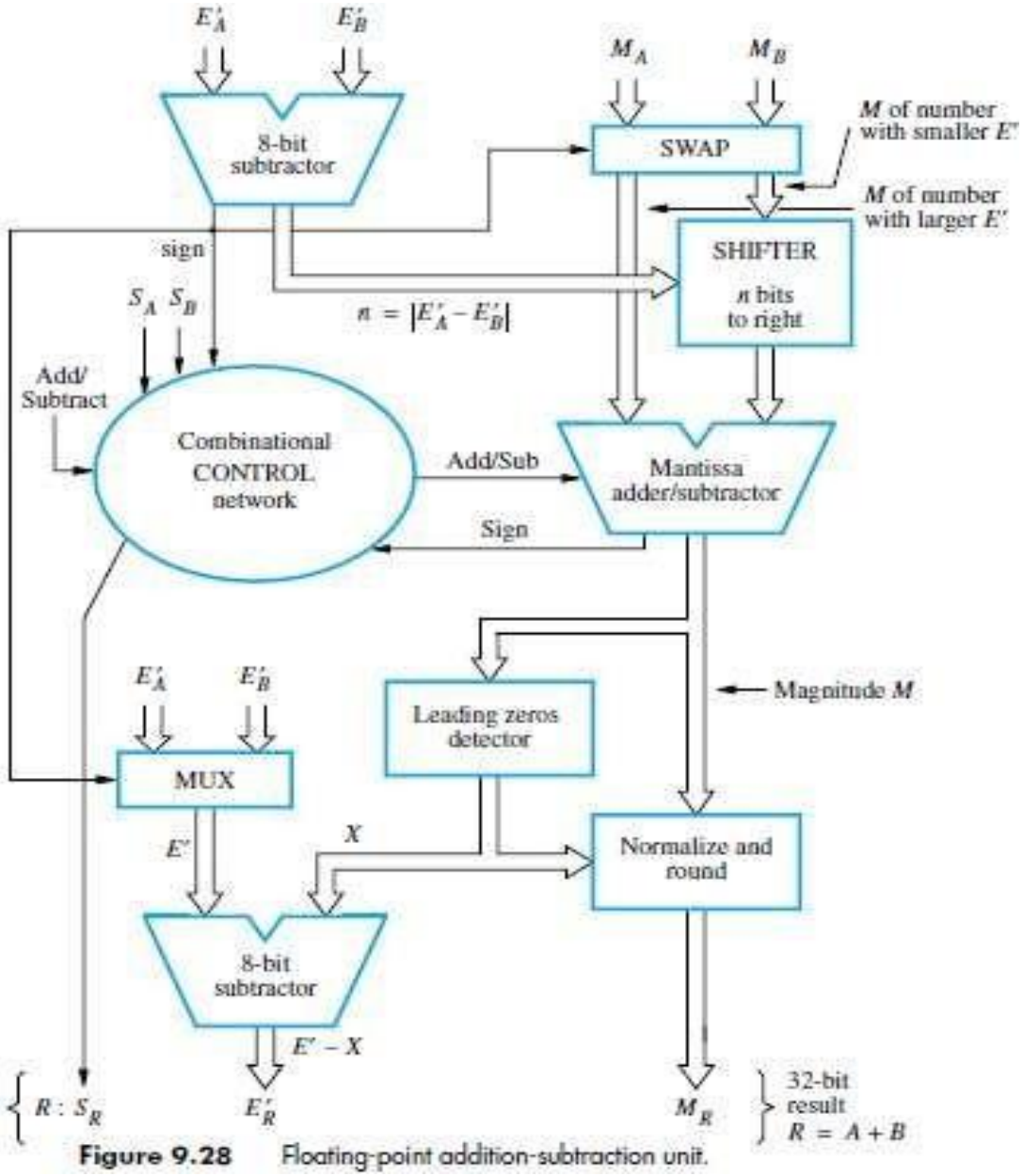


Figure 9.28 Floating-point addition-subtraction unit.

Basic Processing Unit

SOME FUNDAMENTAL CONCEPTS

- To execute an instruction, processor has to perform following 3 steps:
  1) Fetch contents of memory-location pointed to by PC. Content of this location is an instruction to be executed. The instructions are loaded into IR, Symbolically, this operation is written as:
     IR□ [[PC]]
  2) Increment PC by 4.
     PC□ [PC] +4
  3) Carry out the actions specified by instruction (in the IR).
- The first 2 steps are referred to as **Fetch Phase**.
  Step 3 is referred to as **Execution Phase**.

- The operation specified by an instruction can be carried out by performing one or more of thefollowing actions:
  1) Read the contents of a given memory-location and load them into a register.
  2) Read data from one or more registers.
  3) Perform an arithmetic or logic operation and place the result into a register.
  4) Store data from a register into a given memory-location.
- The hardware-components needed to perform these actions are shown in Figure 5.1.

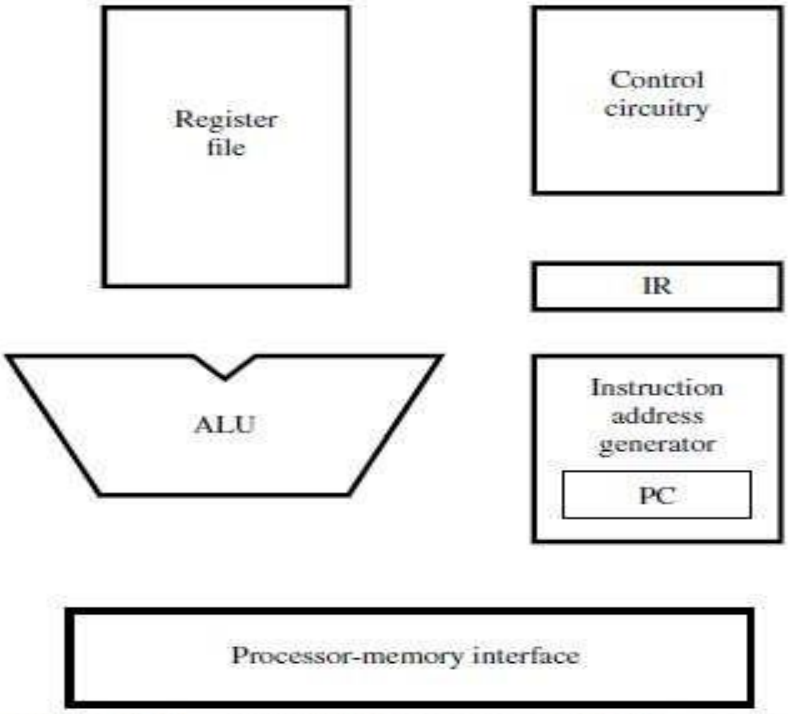


Figure 5.1 Main hardware components of a processor.

SINGLE BUS ORGANIZATION

- ALU and all the registers are interconnected via a **Single Common Bus** (Figure 7.1).
- Data & address lines of the external memory-bus is connected to the internal processor-bus via MDR & MAR respectively. (MDR□ Memory Data Register, MAR □ Memory Address Register).
- **MDR** has 2 inputs and 2 outputs. Data may be loaded
  → into MDR either from memory-bus (external) or
  → from processor-bus (internal).
- **MAR**‟s input is connected to internal-bus; MAR‟s output is connected to external-bus.
- **Instruction Decoder & Control Unit** is responsible for
  → issuing the control-signals to all the units inside the processor.
  → implementing the actions specified by the instruction (loaded in the IR).
- Register R0 through R(n-1) are the **Processor Registers**.
  The programmer can access these registers for general-purpose use.
- Only processor can access 3 registers **Y**, **Z** & **Temp** for temporary storage during program-execution.
  The programmer cannot access these 3 registers.
- In **ALU**, 1) „A‟ input gets the operand from the output of the multiplexer(MUX).
  2) „B‟ input gets the operand directly from the processor-bus.

- There are 2 options provided for „A" input of the ALU.
- MUX is used to select one of the 2 inputs.
- **MUX** selects either
  → output of Y or
  → constant-value 4(which is used to increment PC content).

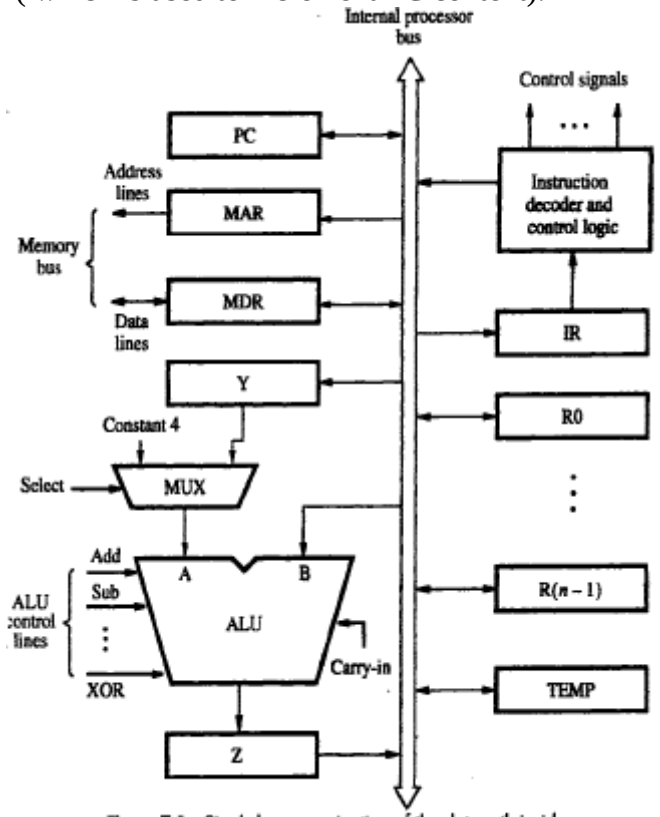


Figure 7.1 Single-bus organization of the datapath inside a processor.

- An instruction is executed by performing one or more of the following operations:
  1) Transfer a word of data from one register to another or to the ALU.
  2) Perform arithmetic or a logic operation and store the result in a register.
  3) Fetch the contents of a given memory-location and load them into a register.
  4) Store a word of data from a register into a given memory-location.
- **Disadvantage:** Only one data-word can be transferred over the bus in a clock cycle.

Solution: Provide multiple internal-paths. Multiple paths allow several data-transfers to take place in parallel.

EXECUTION OF A COMPLETE INSTRUCTION

- Consider the instruction *Add (R3),R1* which adds the contents of a memory-location pointed by R3 to register R1. Executing this instruction requires the following actions:
  1) Fetch the instruction.
  2) Fetch the first operand.
  3) Perform the addition &
  4) Load the result into R1.

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

Figure 7.6 Control sequence for execution of the instruction Add (R3),R1

- Instruction execution proceeds as follows:

Step1--> The instruction-fetch operation is initiated by
→ loading contents of PC into MAR &
→ sending a Read request to memory.

The Select signal is set to Select4, which causes the Mux to select constant 4. This value is added to operand at input B (PC‟s content), and the result is stored in Z.

Step2--> Updated value in Z is moved to PC. This completes the PC increment operation andPC will now point to next instruction.

Step3--> Fetched instruction is moved into MDR and then to IR.The step 1 through 3 constitutes the **Fetch Phase**.

At the beginning of step 4, the instruction decoder interprets the contents of the IR. This enables the control circuitry to activate the control-signals for steps 4 through 7.

The step 4 through 7 constitutes the **Execution Phase**.

Step4--> Contents of R3 are loaded into MAR & a memory read signal is issued. Step5--> Contents of R1 are transferred to Y to prepare for addition.

Step6--> When Read operation is completed, memory-operand is available in MDR, and the addition is performed.

Step7--> Sum is stored in Z, then transferred to R1.The End signal causes a new instructionfetch cycle to begin by returning to step1.

BRANCHING INSTRUCTIONS

- Control sequence for an **unconditional branch instruction** is as follows:

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	Offset-field-of-IR_{out}, Add, Z_{in}
5	Z_{out}, PC_{in}, End

Figure 7.7 Control sequence for an unconditional Branch instruction.

- Instruction execution proceeds as follows:

Step 1-3--> The processing starts & the fetch phase ends in step3.

Step 4--> The offset-value is extracted from IR by instruction-decoding circuit.

Since the updated value of PC is already available in register Y, the offset X is gated onto the bus, and an addition operation is performed.

Step 5--> the result, which is the branch-address, is loaded into the PC.

- The branch instruction loads the branch target address in PC so that PC will fetch the next instruction from the branch target address.
- The branch target address is usually obtained by adding the offset in the contents of PC.
- The offset X is usually the difference between the branch target-address and the addressimmediately following the branch instruction.
- In case of **conditional branch**,

we have to check the status of the condition-codes before loading a new value into the PC.e.g.:

Offset-field-of-IRout, Add, Zin, If N=0 then End

If N=0, processor returns to step 1 immediately after step 4.If N=1, step 5 is performed to load a new value into PC.

MULTIPLE BUS ORGANIZATION

- **Disadvantage of Single-bus organization:** Only one data-word can be transferred over the bus ina clock cycle. This increases the steps required to complete the execution of the instruction

Solution: To reduce the number of steps, most processors provide multiple internal-paths. Multiplepaths enable several transfers to take place in parallel.

- As shown in fig 7.8, three buses can be used to connect registers and the ALU of the processor.
- All general-purpose registers are grouped into a single block called the **Register File**.
- Register-file has 3 ports:
  1) Two output-ports allow the contents of 2 different registers to be simultaneously placed on buses A & B.
  2) Third input-port allows data on bus C to be loaded into a third register during the sameclock-cycle.
- Buses A and B are used to transfer source-operands to A & B inputs of ALU.
- The result is transferred to destination over bus C.

Step	Action
1	PC_{out}, R=B, MAR_{in}, Read, IncPC
2	WMFC
3	MDR_{outB}, R=B, IR_{in}
4	$R4_{outA}$, $R5_{outB}$, SelectA, Add, $R6_{in}$, End

Figure 7.9 Control sequence for the instruction Add R4,R5,R6

- **Incrementer Unit** is used to increment PC by 4.
- Instruction execution proceeds as follows:

Step 1--> Contents of PC are
→ passed through ALU using R=B control-signal &
→ loaded into MAR to start memory Read operation. At the same time, PC is incremented by 4.
Step2--> Processor waits for MFC signal from memory.
Step3--> Processor loads requested-data into MDR, and then transfers them to IR. Step4--> The instruction is decoded and add operation takes place in a single step.

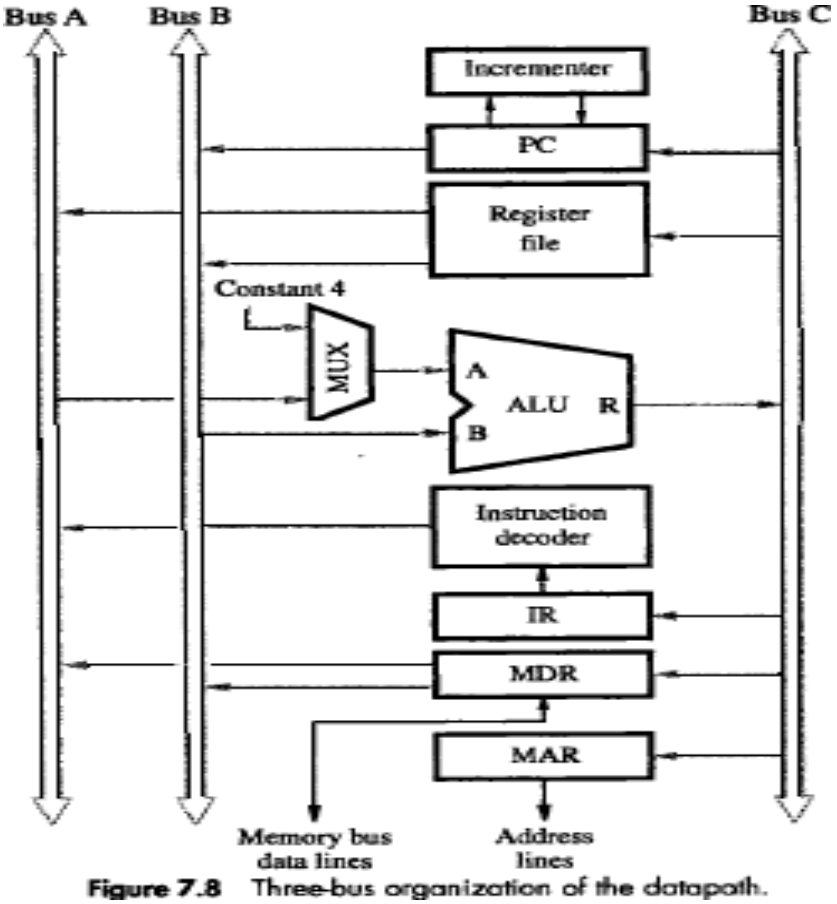


Figure 7.8 Three-bus organization of the datapath.

COMPLETE PROCESSOR

- This has separate processing-units to deal with integer data and floating-point data.
  Integer Unit→ To process integer data. (Figure 7.14).

  Floating Unit→ To process floating –point data.

- **Data-Cache** is inserted between these processing-units & main-memory.The integer and floating unit gets data from data cache.
- **Instruction-Unit** fetches instructions
  → from an instruction-cache or
  → from main-memory when desired instructions are not already in cache.
- Processor is connected to system-bus &
  hence to the rest of the computer by means of a **Bus Interface.**

- Using separate caches for instructions & data is common practice in many processors today.
- A processor may include several units of each type to increase the potential for concurrentoperations.
- The 80486 processor has 8-kbytes single cache for both instruction and data.
  Whereas the Pentium processor has two separate 8 kbytes caches for instruction and data.

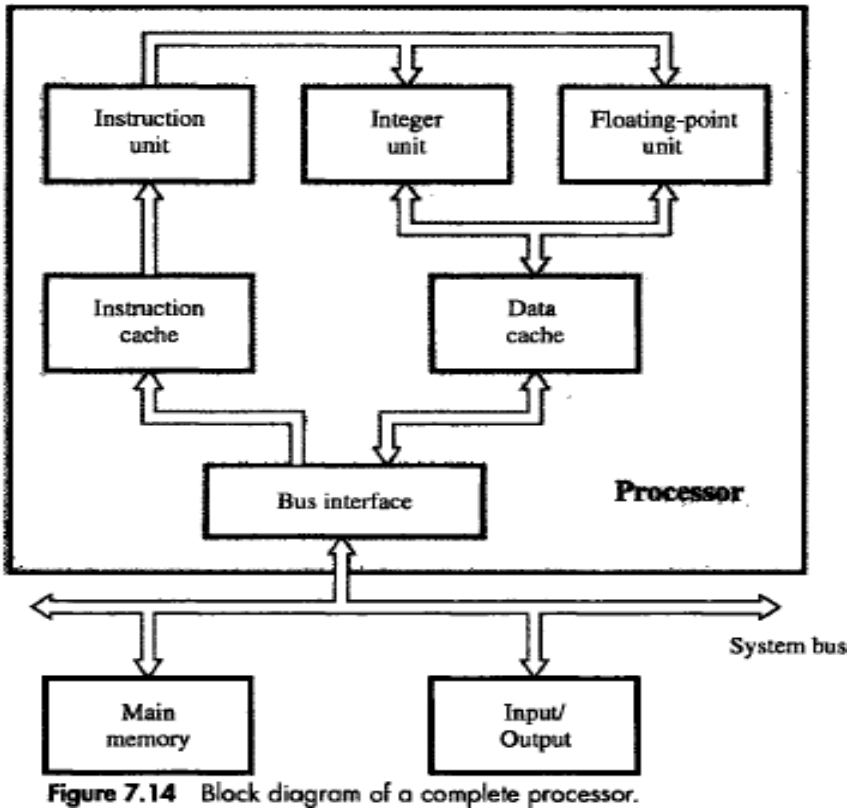


Figure 7.14 Block diagram of a complete processor.

Note:

To execute instructions, the processor must have some means of generating the control-signals. Thereare two approaches for this purpose:

1) Hardwired control and 2) Microprogrammed control.

HARDWIRED CONTROL

- Hardwired control is a method of control unit design (Figure 7.11).
- The control-signals are generated by using logic circuits such as gates, flip-flops, decoders etc.
- **Decoder/Encoder Block** is a combinational-circuit that generates required control-outputsdepending on state of all its inputs.
- Instruction Decoder
  - It decodes the instruction loaded in the IR.
  - If IR is an 8 bit register, then instruction decoder generates 2^8(256 lines); one for each instruction.
  - It consists of a separate output-lines INS1 through INSm for each machine instruction.
  - According to code in the IR, one of the output-lines INS1 through INSm is set to 1, and all

other lines are set to 0.

- **Step-Decoder** provides a separate signal line for each step in the control sequence.
- Encoder
  - It gets the input from instruction decoder, step decoder, external inputs and condition codes.
  - It uses all these inputs to generate individual control-signals: Yin, PCout, Add, End and so on.
  - For example (Figure 7.12), *Zin=T1+T6.ADD+T4.BR*

;This signal is asserted during time-slot T1 for all instructions.
during T6 for an Add instruction.
during T4 for unconditional branch instruction

- When **RUN**=1, counter is incremented by 1 at the end of every clock cycle.When RUN=0, counter stops counting.
- After execution of each instruction, **end** signal is generated. End signal resets step counter.
- Sequence of operations carried out by this machine is determined by wiring of logic circuits, hence the name "**hardwired**".
- **Advantage**: Can operate at high speed.
- Disadvantages:

1) Since no. of instructions/control-lines is often in hundreds, the complexity of control unit is very high.
2) It is costly and difficult to design.
3) The control unit is inflexible because it is difficult to change the design.

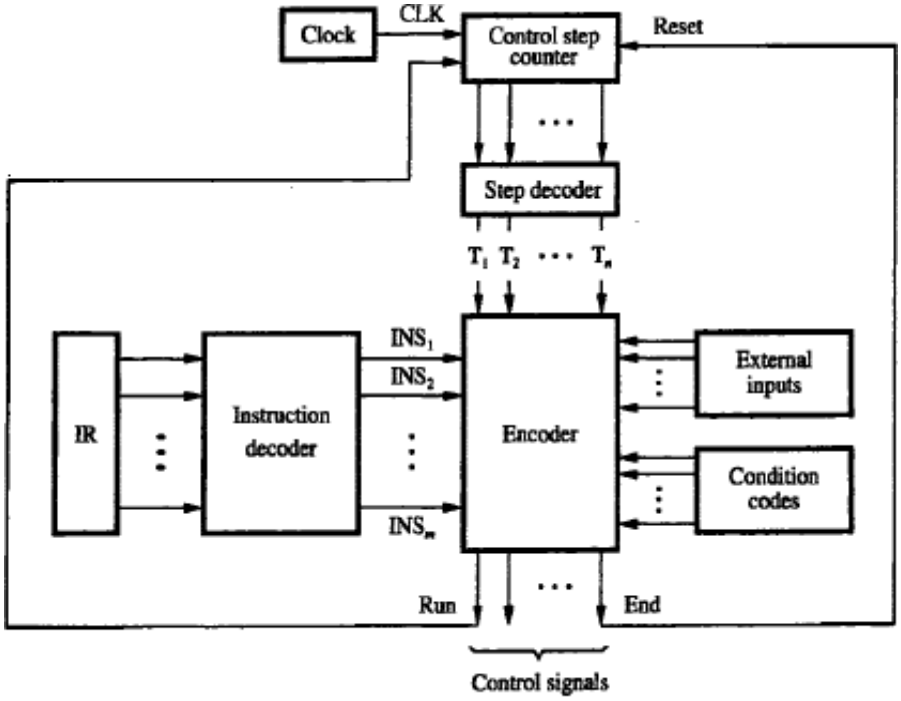


Figure 7.11 Separation of the decoding and encoding functions.

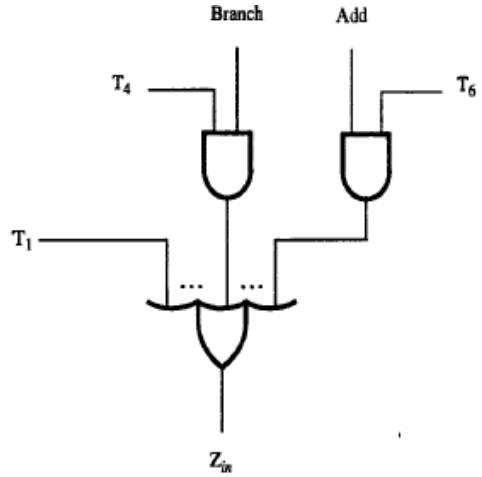


Figure 7.12 Generation of the Z_{in} control signal

MICROPROGRAMMED CONTROL

- Microprogramming is a method of control unit design (Figure 7.16).
- Control-signals are generated by a program similar to machine language programs.
- **Control Word(CW)** is a word whose individual bits represent various control-signals (like Add, PCin).
- Each of the control-steps in control sequence of an instruction defines a unique combination of 1s &0s in CW.
- Individual control-words in microroutine are referred to as **microinstructions** (Figure 7.15).
- A sequence of CWs corresponding to control-sequence of a machine instruction constitutes the **microroutine**.
- The microroutines for all instructions in the instruction-set of a computer are stored in a specialmemory called the **Control Store (CS)**.
- Control-unit generates control-signals for any instruction by sequentially reading CWs of corresponding microroutine from CS.
- **µPC** is used to read CWs sequentially from CS. (µPC Microprogram Counter).
- Every time new instruction is loaded into IR, o/p of **Starting Address Generator** is loaded into µPC.

- Then, µPC is automatically incremented by clock;
  causing successive microinstructions to be read from CS.
  Hence, control-signals are delivered to various parts of processor in correct sequence.

Advantages

- It simplifies the design of control unit. Thus it is both, cheaper and less error prone implement.
- Control functions are implemented in software rather than hardware.
- The design process is orderly and systematic.
- More flexible, can be changed to accommodate new system specifications or to correct the designerrors quickly and cheaply.
- Complex function such as floating point arithmetic can be realized efficiently.

Disadvantages

- A microprogrammed control unit is somewhat slower than the hardwired control unit, because time is required to access the microinstructions from CM.
- The flexibility is achieved at some extra hardware cost due to the control memory and its access circuitry.

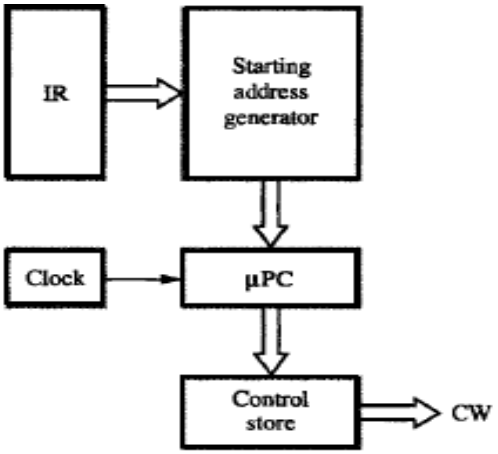


Figure 7.16 Basic organization of a microprogrammed control unit.

Micro-instruction	--	PC_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	Select	Add	Z_{in}	Z_{out}	$R1_{out}$	$R1_{in}$	$R3_{out}$	WMFC	End	--
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	

Figure 7.15 An example of microinstructions for Figure 7.6.