DISCIPLINE: ELECTRONICS & TELECOMMUNICATION ENIGINEERING SEMESTER : V

Subject: Advanced Microprocessor & VLSI

Content Developed by :

Er. Srikanta Sahu
AMIE (India) Electronics &Telecom.,
M.Tech. Computer Sc., LMISTE, MIE

&

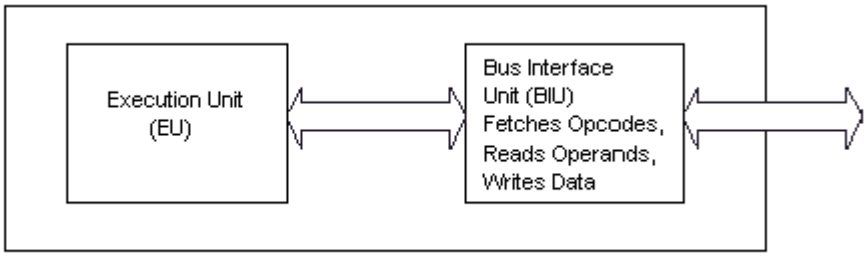
Er. Asman Kumar Sahu
B Tech Electronics &Telecom.

CHAPTER -1.

ADVANCED MICROPROCESSORS AND STANDARDS

1.1 The block diagram of advanced microprocessor, bus interface unit-Microprocessor cache super scalar issue of instructions, integer unit-floating point unit-MMU.

The 8086 CPU is organized as two separate processors, called the Bus Interface Unit (BIU) and the Execution Unit (EU). The BIU provides various functions, including generation of the memory and I/O addresses for the transfer of data between outside the CPU, and the EU.

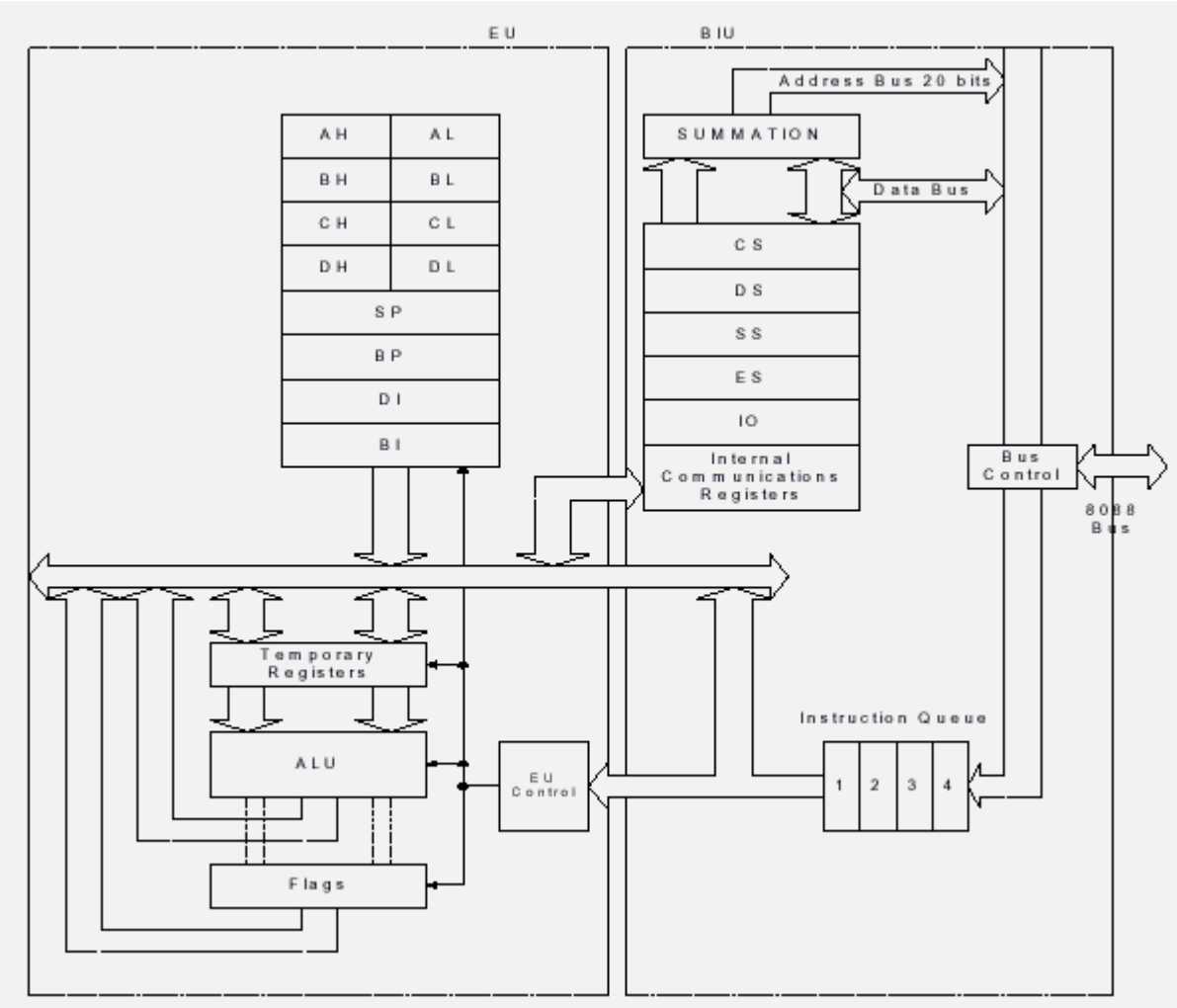


The EU receives program instruction codes and data from the BIU, executes these instructions, and store the results in the general registers. By passing the data back to the BIU, data can also be stored in a memory location or written to an output device. Note that the EU has no connection to the system buses. It receives and outputs all its data through the BIU.

Superscalar Issue

A superscalar CPU architecture implements a form of parallelism called instruction-level parallelism within a single processor. It therefore allows faster CPU throughput than would otherwise be possible at a given clock rate. A superscalar processor executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to different functional units on the processor. Each functional unit is not a separate CPU core but an execution resource within a single CPU such as an arithmetic logic unit, a bit shifter, or a multiplier.

In Flynn's taxonomy, a single-core superscalar processor is classified as an SISD processor (Single Instructions, Single Data), while a multi-core superscalar processor is classified as an MIMD processor (Multiple Instructions, Multiple Data).

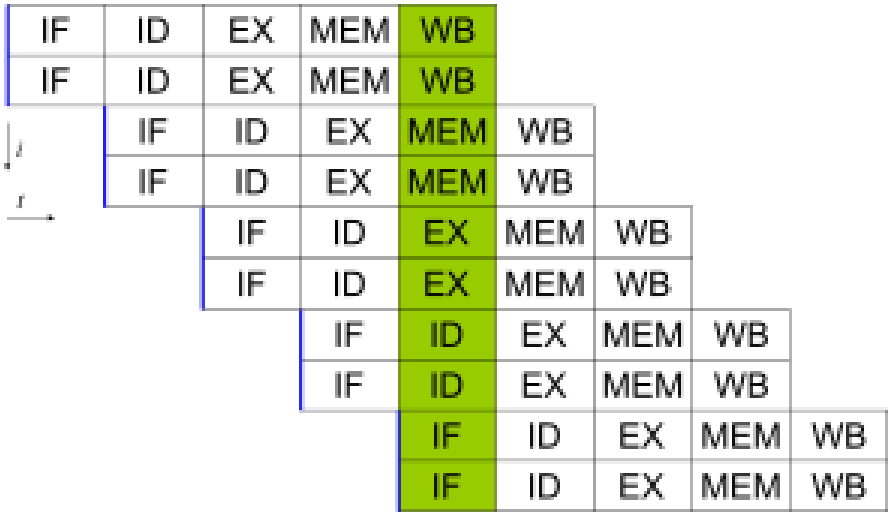


Block diagram of the 8086 Central Processing Unit (CPU)

While a superscalar CPU is typically also pipelined, pipelining and superscalar architecture are considered different performance enhancement techniques.

The superscalar technique is traditionally associated with several identifying characteristics (within a given CPU core):

- Instructions are issued from a sequential instruction stream
- CPU hardware dynamically checks for data dependencies between instructions at run time (versus software checking at compile time)
- The CPU processes multiple instructions per clock cycle



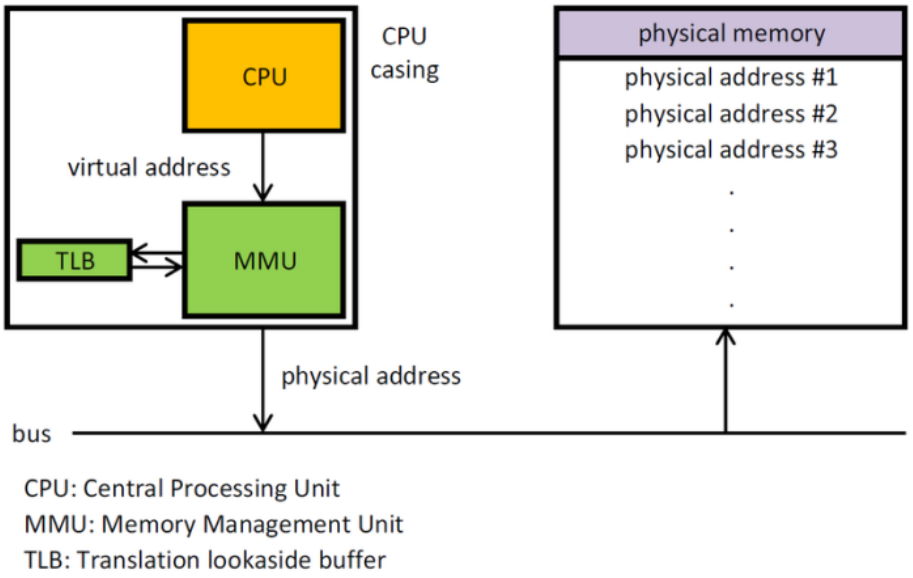
(Simple superscalar pipeline. By fetching and dispatching two instructions at a time, a maximum of two instructions per cycle can be completed. (IF = Instruction Fetch, ID = Instruction Decode, EX = Execute, MEM = Memory access, WB = Register write back, i = Instruction number, t = Clock cycle [i.e., time])

Memory management unit

- A memory management unit (MMU), sometimes called paged memory management unit (PMMU), is a computer hardware unit having all memory references passed through itself, primarily performing the translation of virtual memory addresses to physical addresses. It is usually implemented as part of the central processing unit (CPU), but it also can be in the form of a separate integrated circuit.
- An MMU is effectively performing the virtual memory management, handling at the same time memory protection, cache control, bus arbitration and, in simpler computer architectures (especially 8-bit systems), bank switching.
- Modern MMUs typically divide the virtual address space (the range of addresses used by the processor) into pages, each having a size which is a power of 2, usually a few kilobytes, but they may be much larger. The bottom bits of the address (the offset within a page) are left unchanged. The upper address bits are the virtual page numbers.[2]
- Page table entries
- Most MMUs use an in-memory table of items called a "page table," containing one "page table entry" (PTE) per page, to map virtual page numbers to physical page numbers in main memory. An associative cache of PTEs is called a translation lookaside buffer (TLB) and is used to avoid the necessity of accessing the main memory every time a virtual address is mapped. Other MMUs may have a private array of memory[3] or registers

that hold a set of page table entries. The physical page number is combined with the page offset to give the complete physical address.

- A PTE may also include information about whether the page has been written to (the "dirty bit"), when it was last used (the "accessed bit," for a least recently used (LRU) page replacement algorithm), what kind of processes (user mode or supervisor mode) may read and write it, and whether it should be cached.
- Sometimes, a PTE prohibits access to a virtual page, perhaps because no physical random access memory has been allocated to that virtual page. In this case, the MMU signals a page fault to the CPU. The operating system (OS) then handles the situation, perhaps by trying to find a spare frame of RAM and set up a new PTE to map it to the requested virtual address. If no RAM is free, it may be necessary to choose an existing page (known as a "victim"), using some replacement algorithm, and save it to disk (a process called "paging"). With some MMUs, there can also be a shortage of PTEs, in which case the OS will have to free one for the new mapping.



- The MMU may also generate illegal access error conditions or invalid page faults upon illegal or non-existing memory accesses, respectively, leading to segmentation fault or bus error conditions when handled by the operating system.

1.2 Memory Hierarchy – Register file –cache-address mapping- virtual memory and paging segmentation.

The term memory hierarchy is used in the theory of computation when discussing performance issues in computer architectural design, algorithm predictions, and the lower

level programming constructs such as involving locality of reference. A 'memory hierarchy' in computer storage distinguishes each level in the 'hierarchy' by response time. Since response time, complexity, and capacity are related, the levels may also be distinguished by the controlling technology. The many trade-offs in designing for high performance will include the structure of the memory hierarchy, i.e. the size and technology of each component. So the various components can be viewed as forming a hierarchy of memories (m1,m2,...,mn) in which each member mi is in a sense subordinate to the next highest member mi-1 of the hierarchy. To limit waiting by higher levels, a lower level will respond by filling a buffer and then signaling to activate the transfer.

There are four major storage levels.

This is a most general memory hierarchy structuring. Many other structures are useful. For example, a paging algorithm may be considered as a level for virtual memory when designing a computer architecture.

- Adding complexity slows down the memory hierarchy.
- CMOx memory technology stretches the Flash space in the memory hierarchy
- One of the main ways to increase system performance is minimising how far down the memory hierarchy one has to go to manipulate data.
- Latency and bandwidth are two metrics associated with caches and memory. Neither of them is uniform, but is specific to a particular component of the memory hierarchy.
- Predicting where in the memory hierarchy the data resides is difficult.
- ...the location in the memory hierarchy dictates the time required for the prefetch to occur.

Application of the concept

The memory hierarchy in most computers is:

- Processor registers – fastest possible access (usually 1 CPU cycle), only hundreds of bytes in size
- Level 1 (L1) cache – often accessed in just a few cycles, usually tens of kilobytes
- Level 2 (L2) cache – higher latency than L1 by 2× to 10×, often 512 KiB or more
- Level 3 (L3) cache – higher latency than L2, often 2048 KiB or more
- Main memory – may take hundreds of cycles, but can be multiple gigabytes. Access times may not be uniform, in the case of a NUMA machine.

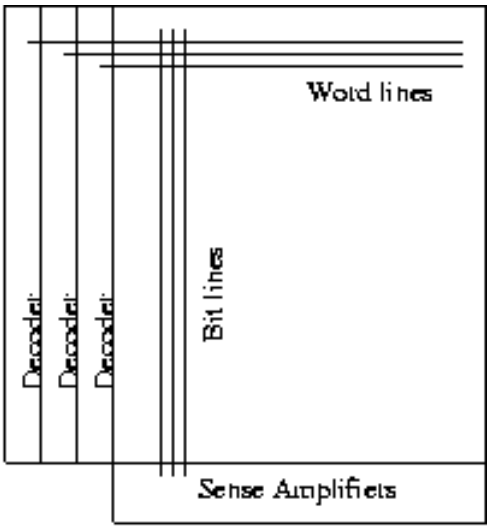
- Disk storage – millions of cycles latency if not cached, but very large
- Tertiary storage – several seconds latency, can be huge

Register file

A register file is an array of processor registers in a central processing unit (CPU). Modern integrated circuit-based register files are usually implemented by way of fast static RAMs with multiple ports. Such RAMs are distinguished by having dedicated read and write ports, whereas ordinary multiported SRAMs will usually read and write through the same ports.

The instruction set architecture of a CPU will almost always define a set of registers which are used to stage data between memory and the functional units on the chip. In simpler CPUs, these architectural registers correspond one-for-one to the entries in a physical register file within the CPU. More complicated CPUs use register renaming, so that the mapping of which physical entry stores a particular architectural register changes dynamically during execution.

Implementation



The usual layout convention is that a simple array is read out vertically. That is, a single word line, which runs horizontally, causes a row of bit cells to put their data on bit lines, which run vertically. Sense amps, which convert low-swing read bitlines into full-swing logic levels, are usually at the bottom (by convention). Larger register files are then sometimes constructed by tiling mirrored and rotated simple arrays.

Register files have one word line per entry per port, one bit line per bit of width per read port, and two bit lines per bit of width per write port. Each bit cell also has a Vdd and Vss. Therefore, the wire pitch area increases as the square of the number of ports, and the transistor area increases linearly. At some point, it may be smaller and/or faster to have

multiple redundant register files, with smaller numbers of read ports, than a single register file with all the read ports. The MIPS R8000's integer unit, for example, had a 9 read 4 write port 32 entry 64-bit register file implemented in a 0.7 μm process, which could be seen when looking at the chip from arm's length. In principle anything that could be done with a 64-bit-wide register file with many read and write ports could be done with a single 8-bit-wide register file with a single read port and a single write port. However, the bit-level parallelism of wide register files with many ports allows them to run much faster -- they can do things in a single cycle that would take many cycles with fewer ports or a narrower bit width or both.

Cache-address mapping

A cache in the primary storage hierarchy contains cache lines that are grouped into sets. If each set contains k lines then we say that the cache is k-way associative. A data request has an address specifying the location of the requested data. Each cache-line sized chunk of data from the lower level can only be placed into one set. The set that it can be placed into depends on its address.This mapping between addresses and sets must have an easy, fast implementation. The fastest implementation involves using just a portion of the address to select the set. When this is done, a request address is broken up into three parts

An offset part identifies a particular location within a cache line.

A set part identifies the set that contains the requested data.

A tag part must be saved in each cache line along with its data to distinguish different addresses that could be placed in the set.

An Example

A computer uses 32-bit byte addressing. The computer uses a 2-way associative cache with a capacity of 32KB. Each cache block contains 16 bytes. Calculate the number of bits in the TAG, SET, and OFFSET fields of a main memory address.

Answer

Since there are 16 bytes in a cache block, the OFFSET field must contain 4 bits (24 = 16). To determine the number of bits in the SET field, we need to determine the number of sets. Each set contains 2 cache blocks (2-way associative) so a set contains 32 bytes. There are 32KB bytes in the entire cache, so there are 32KB/32B = 1K sets. Thus the set field contains 10 bits (210 = 1K).

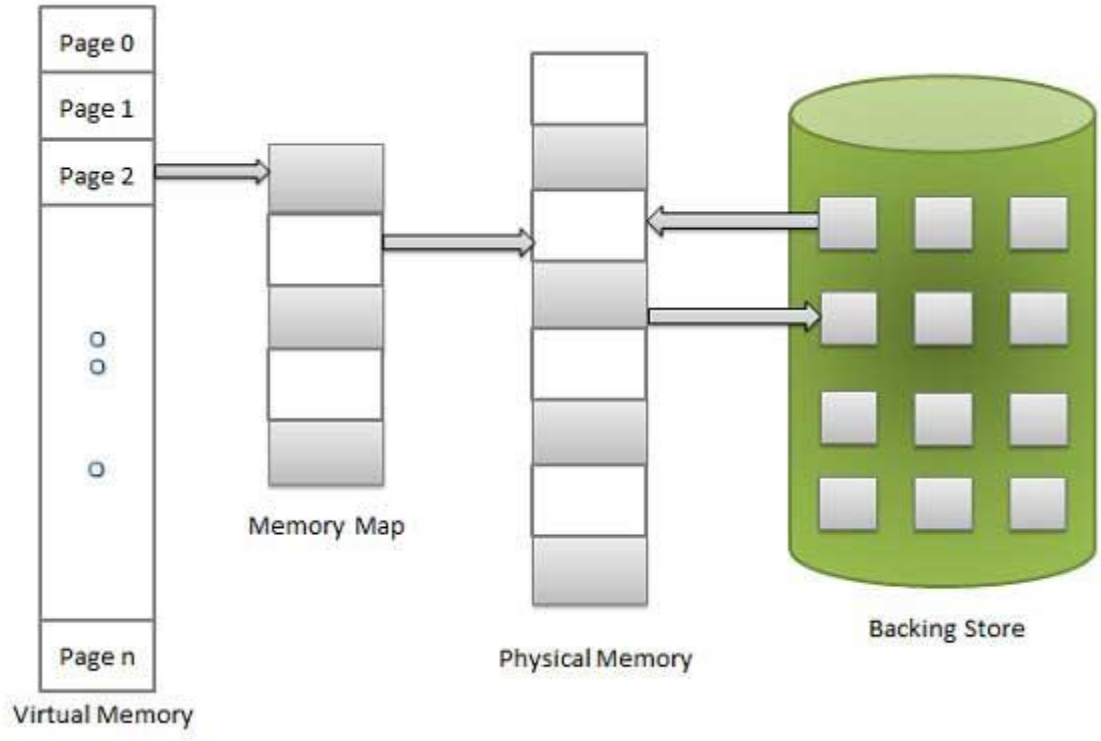
Virtual memory

Virtual memory is a technique that allows the execution of processes which are not completely available in memory. The main visible advantage of this scheme is that programs

can be larger than physical memory. Virtual memory is the separation of user logical memory from physical memory.

This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. Following are the situations, when entire program is not required to be loaded fully in main memory.

- User written error handling routines are used only when an error occured in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.



Memory segmentation

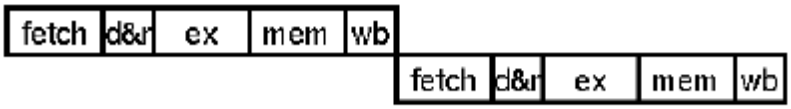
- Memory segmentation is the division of a computer's primary memory into segments or sections. In a computer system using segmentation, a reference to a memory location includes a value that identifies a segment and an offset within that segment. Segments or sections are also used in object files of compiled programs when they are linked together into a program image and when the image is loaded into memory.
- Segments usually correspond to natural divisions of a program such as individual routines or data tables so segmentation is generally more visible to the programmer than paging alone.[1] Different segments may be created for different program modules, or for different classes of memory usage such as code and data segments. Certain segments may be shared between programs
- In a system using segmentation computer memory addresses consist of a segment id and an offset within the segment. A hardware memory management unit (MMU) is responsible for translating the segment and offset into a physical memory address, and for performing checks to make sure the translation can be done and that the reference to that segment and offset is permitted.
- Each segment has a length and set of permissions (for example, read, write, execute) associated with it. A process is only allowed to make a reference into a segment if the type of reference is allowed by the permissions, and if the offset within the segment is within the range specified by the length of the segment. Otherwise, a hardware exception such as a segmentation fault is raised.
- Segments may also be used to implement virtual memory. In this case each segment has an associated flag indicating whether it is present in main memory or not. If a segment is accessed that is not present in main memory, an exception is raised, and the operating system will read the segment into memory from secondary storage.
- Segmentation is one method of implementing memory protection.[2] Paging is another, and they can be combined. The size of a memory segment is generally not fixed and may be as small as a single byte.[3]
- Segmentation has been implemented in several different ways on different hardware, with or without paging. The Intel x86 implementation of segments does not fit either model and is discussed separately below.

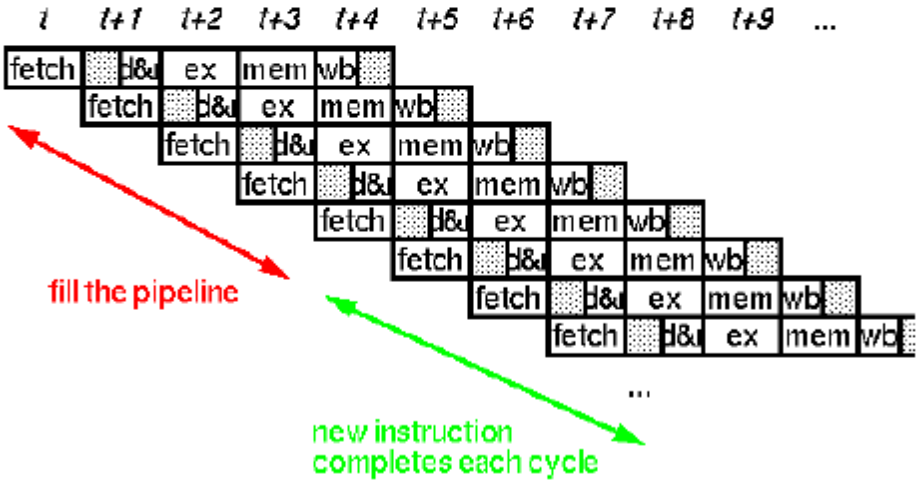
- Segmentation without paging
- Associated with each segment is information that indicates where the segment is located in memory— the segment base. When a program references a memory location the offset is added to the segment base to generate a physical memory address.
- An implementation of virtual memory on a system using segmentation without paging requires that entire segments be swapped back and forth between main memory and secondary storage. When a segment is swapped in, the operating system has to allocate enough contiguous free memory to hold the entire segment. Often memory fragmentation results in there being not enough contiguous memory even though there may be enough in total.
- Segmentation with paging
- Instead of an actual memory location the segment information includes the address of a page table for the segment. When a program references a memory location the offset is translated to a memory address using the page table. A segment can be extended simply by allocating another memory page and adding it to the segment's page table.
- An implementation of virtual memory on a system using segmentation with paging usually only moves individual pages back and forth between main memory and secondary storage, similar to a paged non-segmented system. Pages of the segment can be located anywhere in main memory and need not be contiguous. This usually results in less paging input/output and reduced memory fragmentation

1.3 Discuss Pipe lining – pipe line hazards Instruction level parallelism, RISC versus CISC.

Pipelining

Implementation technique (but it is visible to the architecture) overlaps execution of different instructions execute all steps in the execution cycle simultaneously, but on different instructions Exploits ILP by executing several instructions “in parallel” Goal is to increase instruction throughput





pipeline hazards is of three types 1. structural, 2. data, 3. Control place a "soft limit" on the number of stages increase instruction latency (a little) write & read pipeline registers for data that is computed in a stage time for clock & control lines to reach all stages all stages are the same length which is determined by the longest stage stage length determines clock cycle time.

Structural Hazards

Cause: Instructions in different stages want to use the same resource in the same cycle e.g., 4 FP instructions ready to execute & only 2 FP units

Solutions: more hardware (eliminate the hazard)

- stall (so still execute correct programs)
- less hardware, lower cost
- only for big hardware components

Data Hazards

Cause:

• an instruction early in the pipeline needs the result produced by an instruction farther down the pipeline before it is written to a register would not have occurred if the implementation was not pipelined

Types

RAW (data: flow), WAR (name: antidependence), WAW (name: output)

HW solutions

• forwarding hardware (eliminate the hazard)

• stall via pipelined interlocks if can't forward Compiler solution

• code scheduling (for loads)

Control Hazards

Cause: condition & target determined after next fetch

Early HW solutions

• stall

• assume an outcome & flush pipeline if wrong

• move branch resolution hardware forward in the pipeline

Compiler solutions

• code scheduling

• static branch prediction

RISC versus CISC.

The simplest way to examine the advantages and disadvantages of RISC architecture is by contrasting it with it's predecessor: CISC (Complex Instruction Set Computers) architecture.

Multiplying Two Numbers in Memory

On the right is a diagram representing the storage scheme for a generic computer. The main memory is divided into locations numbered from (row) 1: (column) 1 to (row) 6: (column) 4. The execution unit is responsible for carrying out all computations. However, the execution unit can only operate on data that has been loaded into one of the six registers (A, B, C, D, E, or F). Let's say we want to find the product of two numbers - one stored in location 2:3 and another stored in location 5:2 - and then store the product back in the location

The CISC Approach

The primary goal of CISC architecture is to complete a task in as few lines of assembly as possible. This is achieved by building processor hardware that is capable of understanding and executing a series of operations. For this particular task, a CISC processor would come prepared with a specific instruction (we'll call it "MULT"). When executed, this instruction loads the two values into separate registers, multiplies the operands in the execution unit, and then stores the product in the appropriate register. Thus, the entire task of multiplying two numbers can be completed with one instruction:

MULT 2:3, 5:2

MULT is what is known as a "complex instruction." It operates directly on the computer's memory banks and does not require the programmer to explicitly call any loading or storing functions. It closely resembles a command in a higher level language. For instance, if we let "a" represent the value of 2:3 and "b" represent the value of 5:2, then this command is identical to the C statement "a = a * b."

One of the primary advantages of this system is that the compiler has to do very little work to translate a high-level language statement into assembly. Because the length of the code is relatively short, very little RAM is required to store instructions. The emphasis is put on building complex instructions directly into the hardware.

The RISC Approach

RISC processors only use simple instructions that can be executed within one clock cycle. Thus, the "MULT" command described above could be divided into three separate commands: "LOAD," which moves data from the memory bank to a register, "PROD," which finds the product of two operands located within the registers, and "STORE," which moves data from a register to the memory banks. In order to perform the exact series of steps described in the CISC approach, a programmer would need to code four lines of assembly:

```
LOAD    A,    2:3
LOAD    B,    5:2
PROD    A,    B
STORE   2:3,  A
```

At first, this may seem like a much less efficient way of completing the operation. Because there are more lines of code, more RAM is needed to store the assembly level

instructions. The compiler must also perform more work to convert a high-level language statement into code of this form.

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
Small code sizes, high cycles per second	Low cycles per second, large code sizes
Transistors used for storing complex instructions	Spends more transistors on memory registers

However, the RISC strategy also brings some very important advantages. Because each instruction requires only one clock cycle to execute, the entire program will execute in approximately the same amount of time as the multi-cycle "MULT" command. These RISC "reduced instructions" require less transistors of hardware space than the complex instructions, leaving more room for general purpose registers. Because all of the instructions execute in a uniform amount of time (i.e. one clock), pipelining is possible.

1.4 Comparison of pentium processor with 80386 and 80486

- ′THE 80386 MICROPROCESSOR A 32-bit microprocessor ′introduced by Intel in 1985. ′The chip of 80386 contains 132 pins. ′It has total 129 instructions. It has 32 bit data bus 32 bit address ′bus. The execution of the instructions is highly pipelined and the processor is designed to operate in a multiuser and ′multitasking. Software written for the 8088,8086,80186 and 80286 will also run on 386.
- ′ The address bus is capable of addressing over 4 ′gigabytes of physical memory. Virtual addressing pushing this over 64 terabytes of ′storage. ′80387 coprocessor is used. The processor can operate in two modes: In the real mode physical address space is 1Mbytes and maximum size of segment is 64KB. In the protected mode address space is 4G bytes and maximum size of segment is upto entire physical addressing space.
- ′ 80386 processor is available in 2 different versions. ¬ 386DX ′ 32 bit address bus and 32 bit data bus. ′ 132 pins package. ¬ 386SX ′ 24 bit address bus and 16 bit

data bus. ′ 100 pin package. ′ The lower cost package and ease of interfacing 8 bit and 16 bit memory and peripherals. ′ But the address range and memory transfer rate are lower than that of 386DX.

- ′REGISTER SET-80386 It included all eight general purpose registers plus the ′four segment registers. The general purpose registers were 16 bit wide in earlier machines, but in 386 these registers can be extended to ′32 bit. Their new names are ′EAX,EBX,ECX and so on. Two additional 16 bit segment are included FS and GS.
- ′THE 80486 MICROPROCESSOR 80486 is the next in Intel's upward compatible 80x86 ′architecture. Only few differences between the 80486 and 80386, but these differences created a significant performance ′improvement. 32 bit microprocessor and ′same register set as 80386. Few additional instructions were added to ′its instruction set. 4 gigabyte addressing space .
- ′IMPROVEMENTS MADE IN 80486 OVER 80386 80486 was ′powered with a 8KB cache memory. This improved the speed of 80486 ′processor to great extent. Some new 80486 instructions are included to maintain the ′cache. ′It uses four way set associative cache. 80486 also uses a co-processor similar to 80387 used with ′80386. But this co-processor is integrated on the chip allows it to execute instructions 3 times faster as 386/387 combination.
- ′ The new design of 80486 allows the instruction to ′execute with fewer clock cycles. 486 is packed with 168 pin grid array package instead of ′the 132 pin used for 386 processor. This additional pin's made room for the additional ′signals. This new design of 80486 allows the instruction to execute with fewer ′clock cycles. These small differences made 80486 more powerful processor.

THE PENTIUM PROCESSOR

- ′WHY THE NAME PENTIUM ????? Intel wanted to prevent their competitors from branding their processors with similar names, as AMD had done ′with their Am486. The name Pentium is originally derived from the Greek word pente meaning five as the series was Intel's 5th generation micro architecture.
- ′THE PENTIUM PROCESSOR Upward compatibility has ′been maintained. It can run all programs written for any 80x86 line, but ′does so at a double the speed of fastest 80486. Pentium is ′mixture of both CISC and RISC technologies. All the prior 80x86

processor are considered as CISC ′processor. The addition of RISC aspects lead to additional performance improvement.

- It uses 64 bit data bus to address memory′ organized in Each bank can′8 banks, each bank contains 512 MB of data. store a byte of data. BE7 BE6 BE5 BE4 BE3 BE2 BE1 BE0 B7 B6 B5 B4 B3 B2 B1 B0 64 bit Memory System of ′Pentium All these bank enable signals are active low.
- ′IMPROVEMENTS OF PENTIUM OVER 80X86 Separate 8KB ′data and instruction cache memory. Dual Integer pipelines are present but only single ′integer pipeline is present in 80486. Branch Prediction Logic.
- ′CACHE MEMORY The Pentium contains two 8K-byte ′cache. ′An 8 byte instruction cache, which stores the instruction. An 8 byte data cache, stores the data used by the ′instructions. In the 80486 with unified cache, a program that was data intensive quickly fills the cache, allowing less room for ′instructions. In Pentium this cannot occur because of the separate instruction cache.
- ′PIPELINING It is a technique used to enable one instruction to complete with ′each clock cycle. In Pentium there are two instruction pipelines, the U pipeline ′and V pipeline. These pipelines are responsible for executing 80x86 ′instructions. During Execution the U and V pipelines are capable of executing two integer instructions at the same time and one floating point instructions.
- ′ On a non pipelined machine 9 clock cycles are needed for the ′individual fetch, decode and execute cycle. On a pipelined machine fetch, decode and execute operations are performed in parallel only 5 cycles are needed to execute the same three ′instructions. ′The First instructions needed 3 cycles to complete. Additional instructions complete at rate of 1 per cycle.
- ′ The Instruction pipelines are five-stage pipelines and capable ′of independent operations. The Five-Stages are, PF – Pre-Fetch D1 – Instruction Decode D2 – Address Generate EX - Execute Cache and ALU Access. ′WB – Write Back The U pipeline can execute any processor instruction where as V pipeline only execute Simple Instruction.
- The Pentium -90 MHz′ The 80486 - 60 MHz′ The 80386 - 40MHz′ The 80286 - 25 MHz′SPEED OF PROCESSORS

1.5 Bus Standards:

The Centronics parallel interface is an older and still widely-used standard I/O interface for connecting printer s and certain other devices to computers. The interface typically includes a somewhat cumbersome cable and a 36- pin male and female connector at the printer or other device. The cable plugs into a 25-pin parallel port on the computer. Data flows in one direction only, from the computer to the printer or other device. In addition to eight parallel data lines, other lines are used to read status information and send control signals. Centronics Corporation designed the original Centronics parallel interface for dot matrix printers. In 1981, IBM used this interface as an alternative to the slower one-bit-at-a-time-serial interface.

When the Centronics parallel interface was first developed, the main peripheral was the printer. Since then, portable disk drives, tape backup drives, and CD-ROM players are among devices that have adopted the parallel interface. These new uses caused manufacturers to look at new ways to make the Centronics parallel interface better. In 1991, Lexmark, IBM, Texas instruments, and others met to discuss a standard that would offer more speed and bi-directional communication. Their effort and the sponsorship of the IEEE resulted in the IEEE 1284 committee. The IEEE 1284 standard was approved for release in March, 1994.

The IEEE 1284 standard specifies five modes of operation, each mode providing data transfer in either the forward direction (computer to peripheral), backward direction (peripheral to computer), or bi-directional (one direction at a time).

- Compatibility mode is the original Centronics parallel interface and intended for use with dot matrix printers and older laser printers. The compatibility mode can be combined with the nibble mode for bi-directional data transfer.
- Nibble mode allows data transfer back to the computer. The nibble mode uses the status lines to send 2 nibble s (4-bit units) of data to the computer in two data transfer cycles. This mode is best used with printers.
- Byte mode uses software driver s to disable the drivers that control the data lines in order for data to be sent from the printer to the computer. The data is sent at the same speed as when data is sent from the computer to the printer. One byte of data is transferred instead of the two data cycles required by the nibble mode.
- ECP mode (Enhanced Capability Port mode) is an advanced bi-directional mode for use with printers and scanner s. It allows data compression for image s, FIFO (first in, first out) for items in queue s, and high-speed, bi-directional communication. Data transfer

occurs at two to four megabytes per second. An advanced feature of ECP is channel addressing . This is used for multifunction devices such as printer/fax/modem devices. For example, if a printer/fax/modem device needs to print and send data over the modem at the same time, the channel address software driver of the ECP mode assigns a new channel to the modem so that both devices can work simultaneously.

- EPP mode (Enhanced Parallel Port mode) was designed by Intel, Xircom, and Zenith Data Systems to provide a high-performance parallel interface that could also be used with the standard interface. EPP mode was adopted as part of the IEEE 1284 standard. The EPP mode uses data cycles that transfer data between the computer and the peripheral and address cycles that assign address, channel, or command information. This allows data transfer speeds of 500 kilobytes to 2 megabytes per second, depending on the speed of the slowest interface. The EPP mode is bi-directional. It is suited for network adapters, data acquisition, portable hard drives, and other devices that need speed.

- **RS 232**

In telecommunications, RS-232 is a standard for serial communication transmission of data. It formally defines the signals connecting between a DTE (data terminal equipment) such as a computer terminal, and a DCE (data circuit-terminating equipment, originally defined as data communication equipment[1]), such as a modem. The RS-232 standard is commonly used in computer serial ports. The standard defines the electrical characteristics and timing of signals, the meaning of signals, and the physical size and pinout of connectors. The current version of the standard is TIA-232-F Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange, issued in 1997.

An RS-232 serial port was once a standard feature of a personal computer, used for connections to modems, printers, mice, data storage, uninterruptible power supplies, and other peripheral devices. However, RS-232 is hampered by low transmission speed, large voltage swing, and large standard connectors. In modern personal computers, USB has displaced RS-232 from most of its peripheral interface roles. Many computers do not come equipped with RS-232 ports and must use either an external USB-to-RS-232 converter or an internal expansion card with one or more serial ports to connect to RS-232 peripherals. RS-232 devices are widely used, especially in industrial machines, networking equipment and scientific instruments.

Chapter 2. Introduction to VLSI

2.1 Historical Perspective

The electronics industry has achieved a phenomenal growth over the last few decades, mainly due to the rapid advances in integration technologies and large-scale systems design. The use of integrated circuits in high-performance computing, telecommunications, and consumer electronics has been growing at a very fast pace. Typically, the required computational and information processing power of these applications is the driving force for the fast development of this field. Figure 1 gives an overview of the prominent trends in information technologies over the next decade. The current leadingedge technologies (such as low bit-rate video and cellular communications) already provide the end-users a certain amount of processing power and portability. This trend is expected to continue, with very important implications for VLSI and systems design. One of the most important characteristics of information services is their increasing need for very high processing power and bandwidth (in order to handle real-time video, forexample).

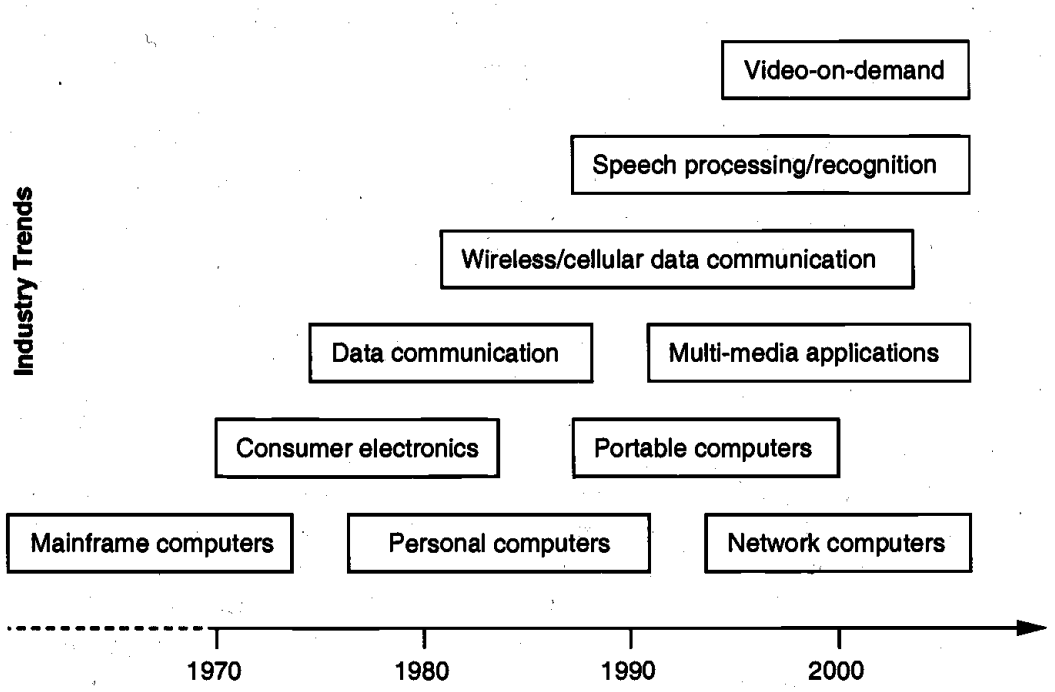


FIG.1(Prominent "driving" trends in information service technologies.)

ERA	DATE	COMPLEXITY (# of logic blocks per chip)
Single transistor	1958	< 1
Unit logic (one gate)	1960	1
Multi-function	1962	2 - 4
Complex function	1964	5 - 20
Medium Scale Integration (MSI)	1967	20 - 200
Large Scale Integration (LSI)	1972	200 - 2,000
Very Large Scale Integration (VLSI)	1978	2,000 - 20,000
Ultra Large Scale Integration (ULSI)	1989	20,000 - ?

2.2 VLSI DESIGN METHODOLOGIES AND VLSI DESIGN FLOW

VLSI DESIGN METHODOLOGIES

The demands of the rapidly rising chip complexity has created significant challenges in many areas; practically hundreds of team members are involved in the development of a typical VLSI product, including the development of technology,computer-aided design (CAD) tools, chip design, fabrication, packaging, testing and reliability qualification. The level of circuit performance which can be reached within a certain design time strongly depends on the efficiency of the design methodologies, as well as on the design style two different VLSI design styles are compared for their relative merits in the design of the same product.

1. FULL CUSTOM DESIGN
2. SEMICUSTOM DESIGN

FULL CUSTOM DESIGN - Using the full-custom design style (where the geometry and the placement of every transistor can be optimized individually) requires a longer time until design maturity can be reached, yet the inherent flexibility of adjusting almost every aspect of circuit design allows far more opportunity for circuit performance improvement during the design cycle. The final

product typically has a high level of performance (e.g. high processing speed, low power dissipation) and the silicon area is relatively small because of better area utilization. But this comes at a larger cost in terms of design time.

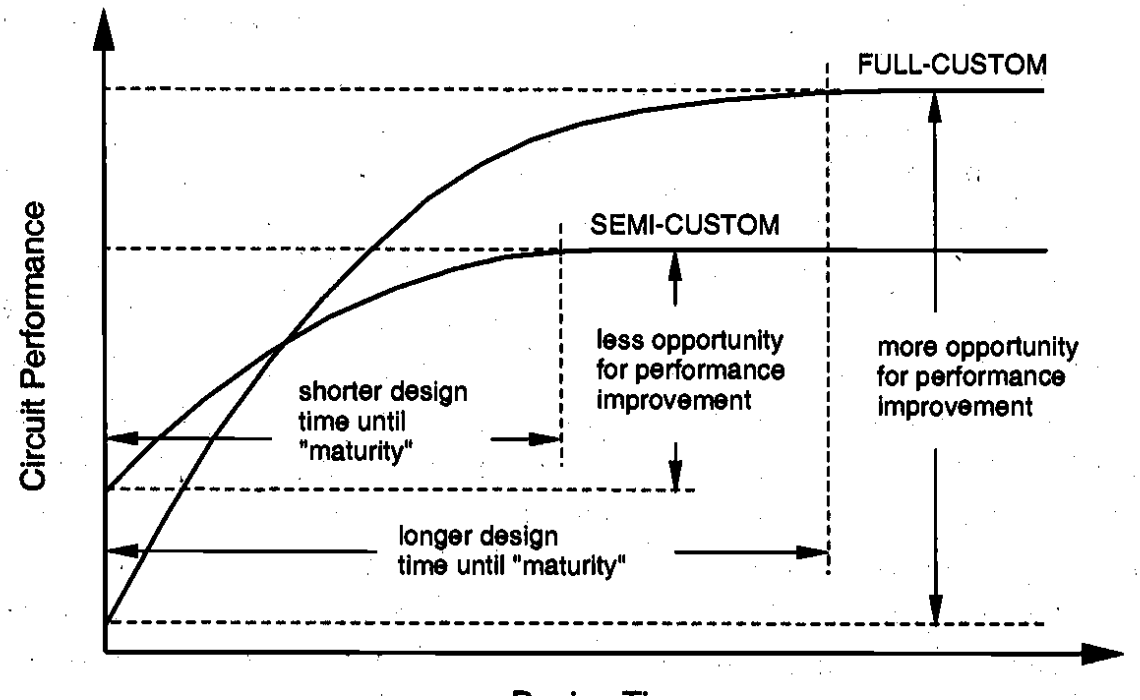


Figure -.2(Impact of different VLSI design styles upon the design cycle time and the achievable circuit performance).

SEMI-CUSTOM DESIGN - Using a semi-custom design style (such as standard-cell based design or FPGA) will allow a shorter design time until design maturity can be achieved. In the early design phase, the circuit performance can be even higher than that of a full-custom design, since some of the components used in semi-custom design are already optimized. But the semi-custom design style offers less opportunity for performance improvement over the long run, and the overall performance of the final product will inevitably be less than that of a full-custom design.

In addition to the proper choice of a VLSI design style, there are other issues which must be addressed in view of the constantly evolving nature of the VLSI manufacturing technologies. Approximately every two years, a new generation of technology is introduced, which typically allows for smaller device dimensions and consequently, higher integration density and higher performance.

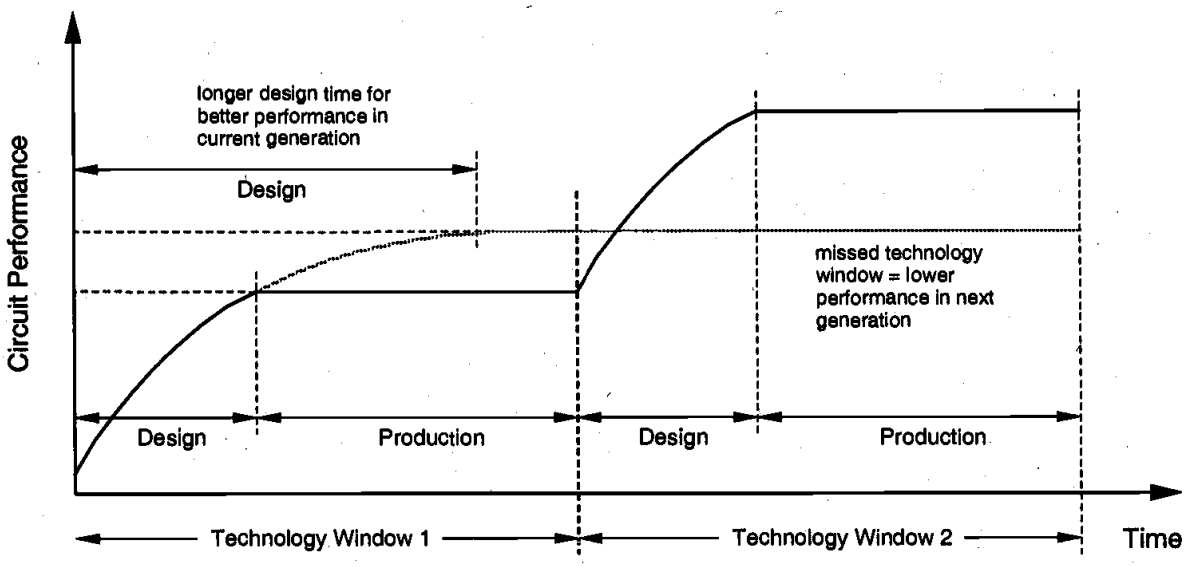


Figure 3. *(*Progressive performance improvement of a VLSI product)

VLSI Design Flow

The design process, at various levels, is usually evolutionary in nature. It starts with a given set of requirements. Initial design is developed and tested against the requirements. When requirements are not met, the design has to be improved. If such improvement is either not possible or too costly, then a revision of requirements and an impact analysis must be considered. The Y-chart (first introduced by D. Gajski) shown in Fig. 4 illustrates a design flow for most logic chips, using design activities on three different axes (domains) which resemble the letter "Y."

The Y-chart consists of three domains of representation, namely (i) behavioural domain, (ii) structural domain, and (iii) geometrical layout domain. The design flow starts from the algorithm that describes the behavior of the target chip. The corresponding architecture of the processor is first defined. It is mapped onto the chip surface by floorplanning. The next design evolution in the behavioral domain defines finite state machines (FSMs) which are structurally implemented with functional modules such as registers and arithmetic logic units (ALUs). These modules are then geometrically placed onto the chip surface using CAD tools for automatic module placement followed by routing, with a goal of minimizing the interconnects area and signal delays. The third evolution starts with a behavioral module description. Individual modules are then implemented with leaf cells. At this stage the chip is described in terms of logic gates (leaf cells), which can be placed and interconnected by using a cell placement and routing program.

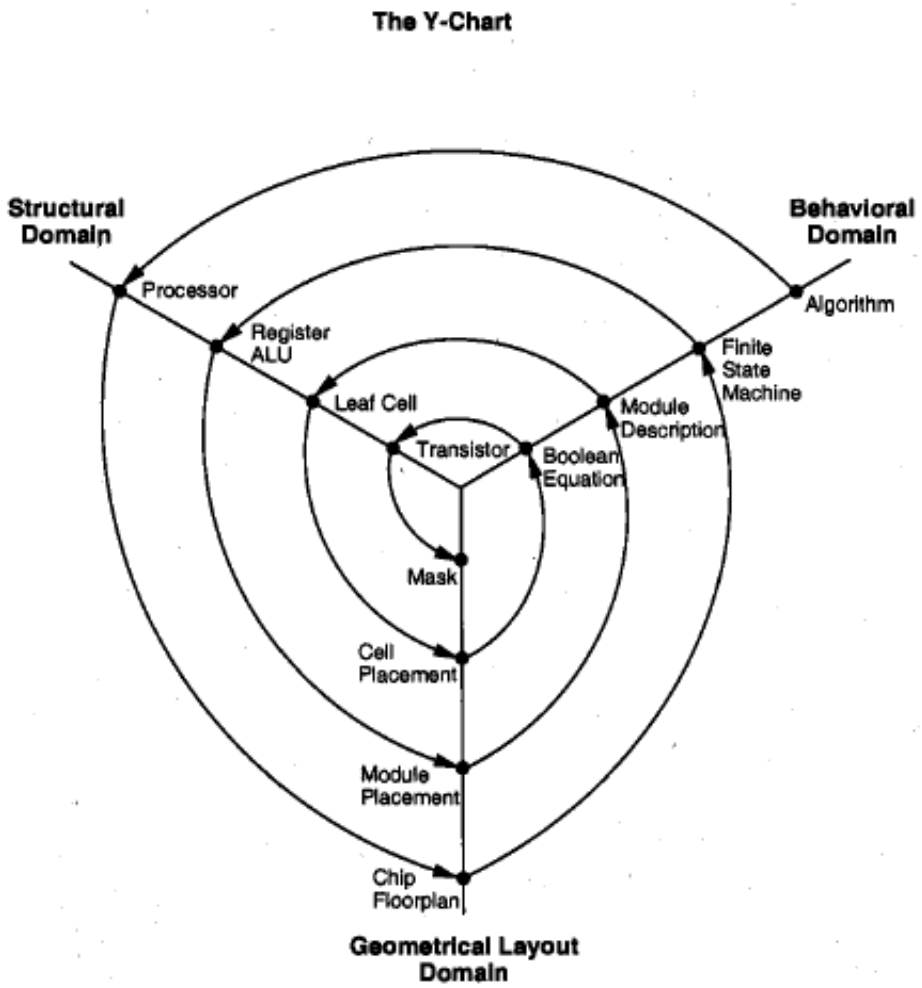


Figure 4.3. Typical VLSI design flow in three domains (Y-chart representation).

The last evolution involves a detailed Boolean description of leaf cells followed by a transistor level implementation of leaf cells and mask generation. In the standardcell based design style, leaf cells are pre designed (at the transistor level) and stored in a library for logic implementation, effectively eliminating the need for the transistor level design. Figure 4 provides a more simplified view of the VLSI design flow, taking into account the various representations, or abstractions of design: behavioral, logic, circuit and mask layout. Note that the verification of design plays a very important role in every step during this process. The failure to properly verify a design in its early phases typically causes significant and expensive re-design at a later stage, which ultimately increases the time-to-market.

2.3 Design Hierarchy, Design Styles & CAD Technology.

Design Hierarchy

The use of the hierarchy, or "divide and conquer" technique involves dividing a module into sub-modules and then repeating this operation on the sub-modules until the complexity of the smaller parts becomes manageable. This approach is very similar to software development wherein large programs are split into smaller and smaller sections until simple subroutines, with well-defined functions and interfaces, can be written. In the physical domain, partitioning a complex system into its various functional blocks will provide a valuable guide for the actual realization of these blocks on the chip.

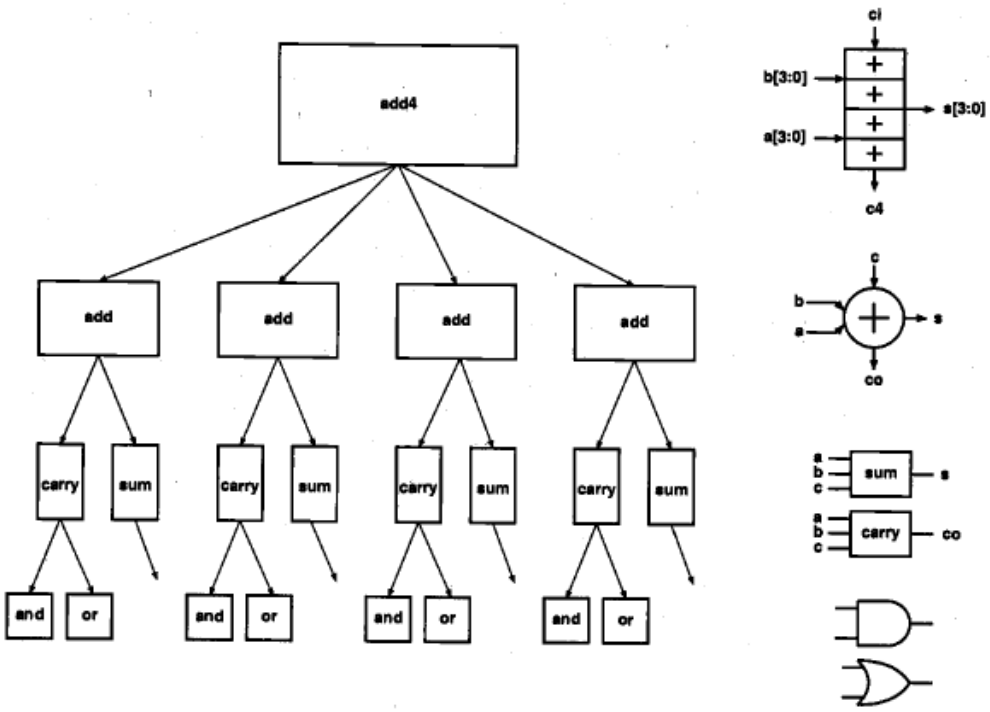


FIG- 5 (Structural decomposition of a 4-bit adder, showing the levels of hierarchy).

Regularity, Modularity and Locality

Regularity means that the hierarchical decomposition of a large system should result in not only simple, but also similar blocks, as much as possible. A good example of regularity is the design of array structures consisting of identical cells - such as a parallel multiplication array.

Modularity in design means that the various functional blocks which make up the larger system must have *well-defined functions* and *interfaces.* Modularity allows that each

block or module can be designed relatively independently from each other, since there is no ambiguity about the function and the signal interface of these blocks.

Locality also ensures that connections are mostly between neighboring modules, avoiding long-distance connections as much as possible. This last point is extremely important for avoiding long interconnect delays

Computer-Aided Design Technology

CHAPTER 14 Computer-aided design (CAD) tools are essential for timely development of integrated circuits. Although CAD tools cannot replace the creative and inventive parts of the design activities, the majority of time-consuming and computation intensive mechanistic parts of the design can be executed by using CAD tools. The CAD technology for VLSI chip design can be categorized into the following areas:

* High level synthesis
* Logic synthesis
* Circuit optimization
* Layout
* Simulation
* Design rules checking

Synthesis Tools

The high-level synthesis tools using hardware description languages (HDLs), such as VHDL or Verilog, address the automation of the design phase in the top level of the design hierarchy.

***Layout** Tools*

The tools for circuit optimization are concerned with transistor sizing for minimization of delays and with process variations, noise, and reliability hazards. The layout CAD tools include floorplanning, place-and-route and module generation. Sophisticated layout tools are goal driven and include some degree of optimization functions.

Simulation and Verification Tools

The simulation category, which is the most mature area of VLSI CAD, includes many tools ranging from circuit-level simulation (SPICE or its derivatives, such as HSPICE), timing level simulation, logic level simulation, and behavioral simulation. Many other simulation tools have also been developed for device-level simulation and process simulation for technology development. The aim of all simulation CAD tools is to determine if the designed circuit meets the required specifications, at all stages of the design process**.**

Chapter 3 .FABRICATION OF MOSFETS

3.1 Explain Fabrication processes (NMOS Fabrication, CMOS n-well process)

FABRICATION PROCESS

The process starts with the creation of the n-well regions for pMOS transistors, by impurity implantation into the substrate. Then, a thick oxide is grown in the regions surrounding the nMOS and pMOS active regions. The thin gate oxide is subsequently grown on the surface through thermal oxidation. These steps are followed by the creation of n+ and p+ regions (source, drain, and channel stop implants) and by final metallization (creation of metal interconnects).

Each processing step requires that certain areas are defined on chip by appropriate masks.As a result patterned layers of doped silicon, polysilicon, metal, and insulating silicon dioxide. In general, a layer must be patterned before the next layer of material is applied on the chip. The process used to transfer a pattern to a layer on the chip is called *lithography.* Since each layer has its own distinct patterning requirements, the lithographic sequence must be repeated for every layer, using a different mask.

starts with the thermal oxidation of the silicon surface, by which an oxide layer of about 1 m thickness, for example, is created on the substrate (Fig. (b)). The entire oxide surface is then covered with a layer of photo-resist, which is essentially a light-sensitive, acid-resistant organic polymer, initially insoluble in the developing solution (Fig.(c)).

If the photo-resist material is exposed to ultraviolet (UV) light, the exposed areas become soluble so that they are no longer resistant to etching solvents. To selectively expose the photo-resist, we have to cover some of the areas on the surface with a mask during exposure. Thus, when the structure with the mask on top is exposed to UV light, areas which are covered by the opaque features on the mask are shielded. In the areas where the UV light can pass through, on the other hand, the photo-resist is exposed and becomes soluble (Fig. (d)).

The type of photo-resist which is initially insoluble and becomes soluble after exposure to UV light is called *positive photo-resist.*

There is another type of photo-resist which is initially soluble and becomes insoluble (hardened) after exposure to UV light, called negative photo-resist. If negative photo-resist is used in the photolithography process,

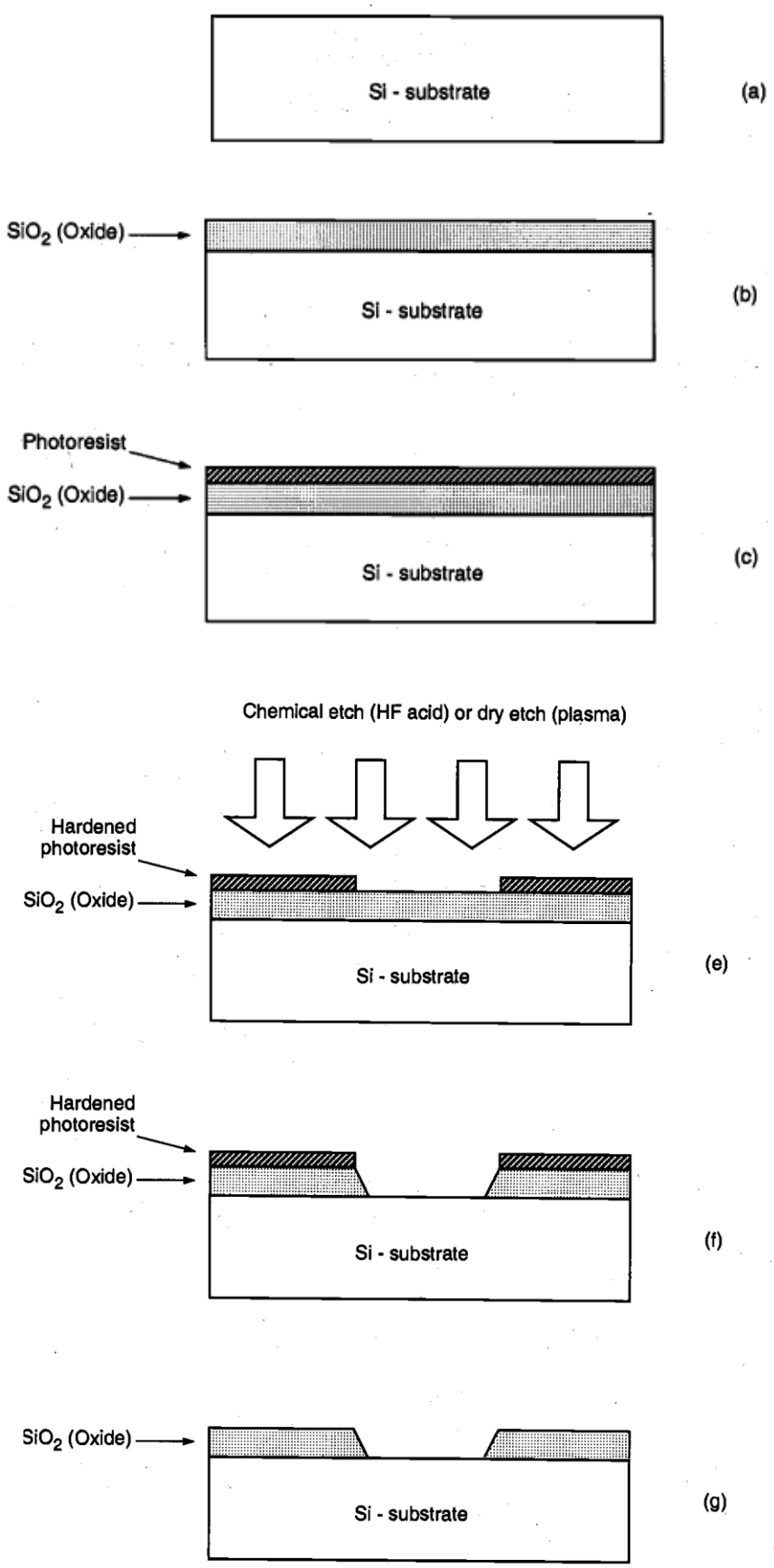


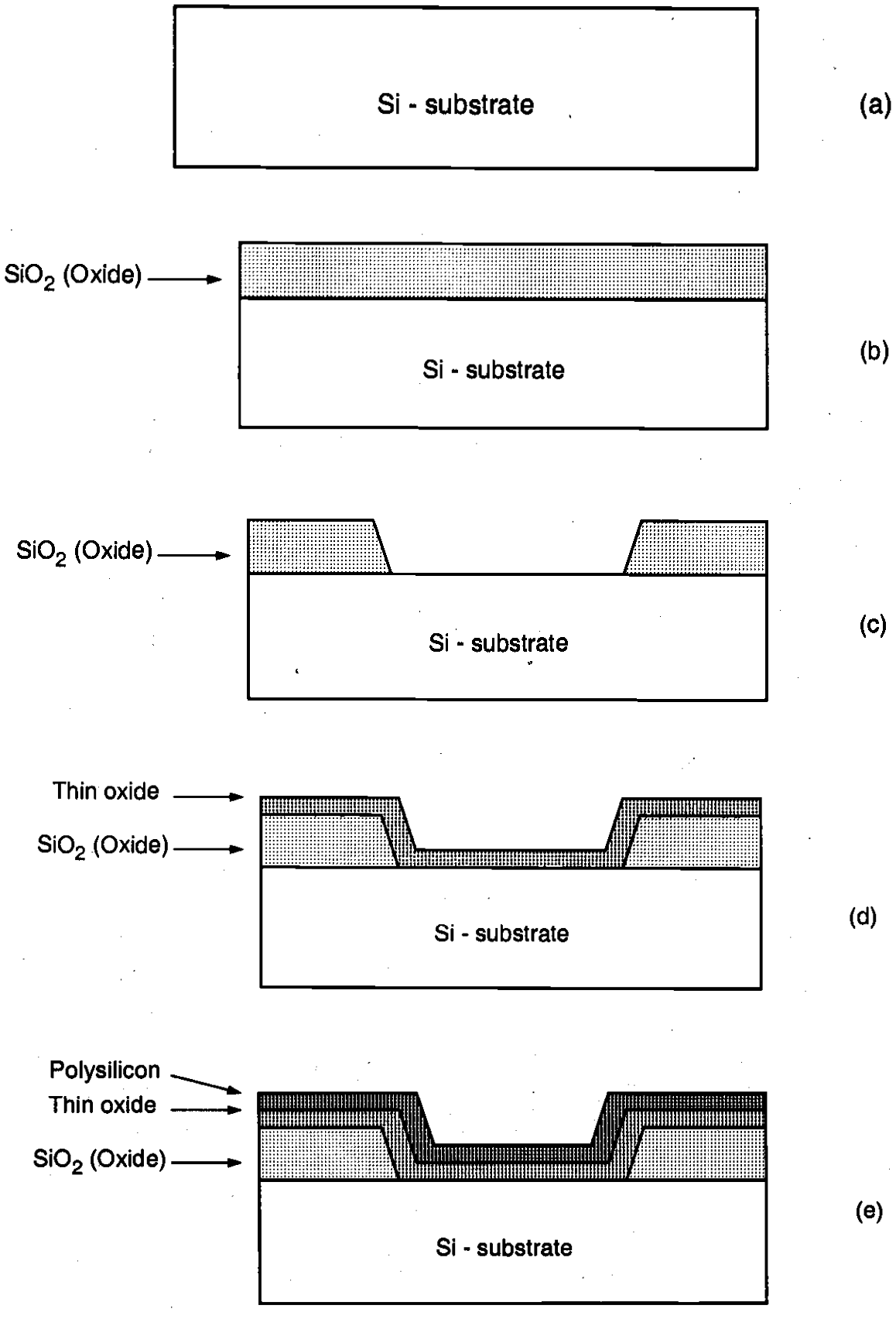
FIG - Process steps required for patterning of silicon dioxide.

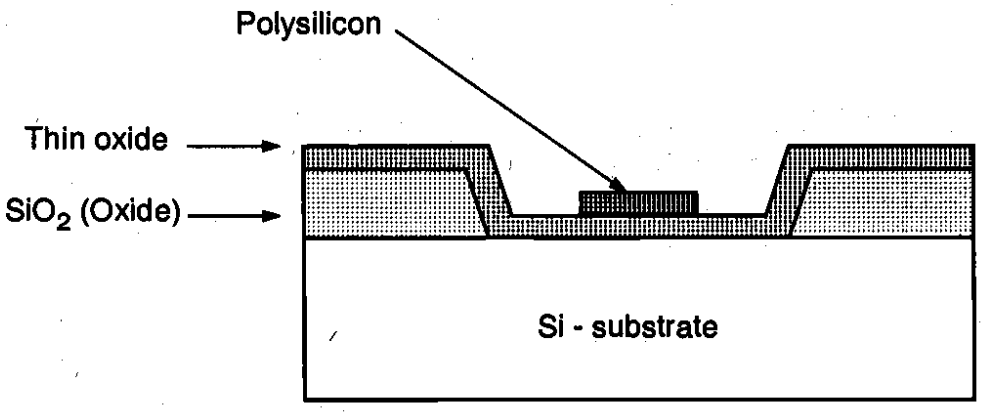
The areas which are not shielded from the UV light by the opaque mask features become insoluble, whereas the shielded areas can subsequently be etched away by a developing solution. Negative photo-resists are more sensitive to light, but their photolithographic resolution is not as high as that of the positive photo-resists. Therefore, negative photo-resists are-used less commonly in the manufacturing of high-density integrated circuits.

Fabrication the Nmos Transistor

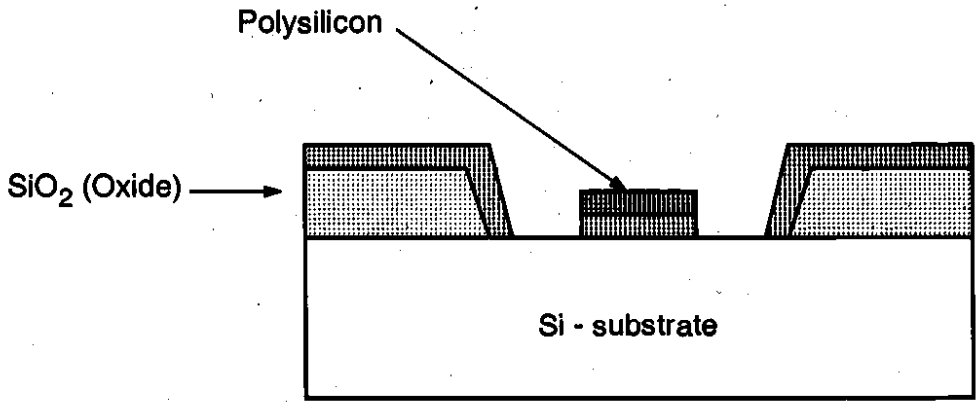
The process starts with of silicon substrate (Fig. (a)), in which a relatively thick dioxide layer, also called field oxide, is created on the surface (Fig. (b)). Then, the field oxide is selectively etched to expose the silicon surface on which the MOS transistor will be created (Fig. (c)). Following this step, the surface is covered with a thin, high-quality oxide layer, which will eventually form the gate oxide of the n- mos transistor On top of the thin oxide layer, a layer of polysilicon (polycrystalline silicon) is deposited (Fig. (e)). Polysilicon is used both as gate electrode material for MOS transistors and also as an interconnect medium in silicon integrated circuits. Undoped polysilicon has relatively high resistivity. The-resistivity of polysilicon can be reduced, however, by doping it with impurity atoms.

After deposition, the polysilicon layer is patterned and etched to form the interconnects and the MOS transistor gates (Fig. (f)). The thin gate oxide not covered by polysilicon is also etched away, which exposes the bare silicon surface on which the source and drainjunctions are to be formed (Fig. (g)). The entire silicon surface is then doped with a high concentration of impurities, either through diffusion or ion implantation (in this case with donor atoms to produce n-type doping). Figure (h) shows that the doping penetrates the exposed areas on the silicon surface, ultimately creating two ntype regions (source and drain junctions) in the p-type substrate. The impurity doping also penetrates the polysilicon on the surface, reducing its resistivity. Note that the polysilicon gate, which is patterned *before* doping, actually defines the precise location of the channel region and, hence, the location of the source and the drain regions. Since this procedure allows very precise positioning of the two regions relative to the gate, it is also called the *self-aligned process.*


Si - substrate
(a)
SiO_2 (Oxide)
Si - substrate
(b)
SiO_2 (Oxide)
Si - substrate
(c)
Thin oxide
SiO_2 (Oxide)
Si - substrate
(d)
Polysilicon
Thin oxide
SiO_2 (Oxide)
Si - substrate
(e)

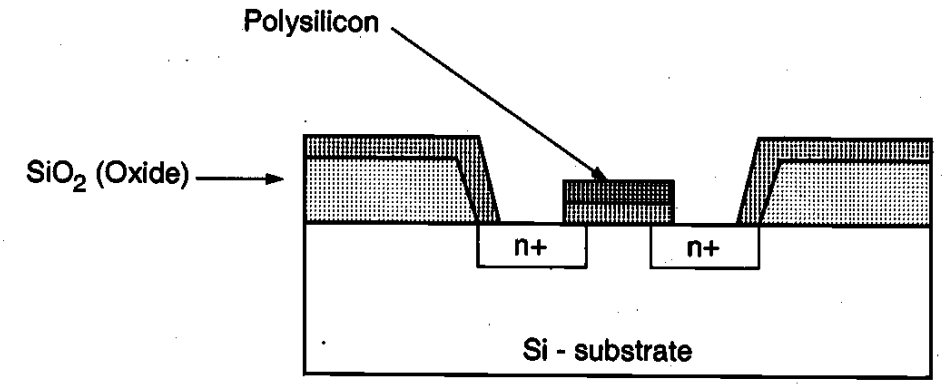


(f)

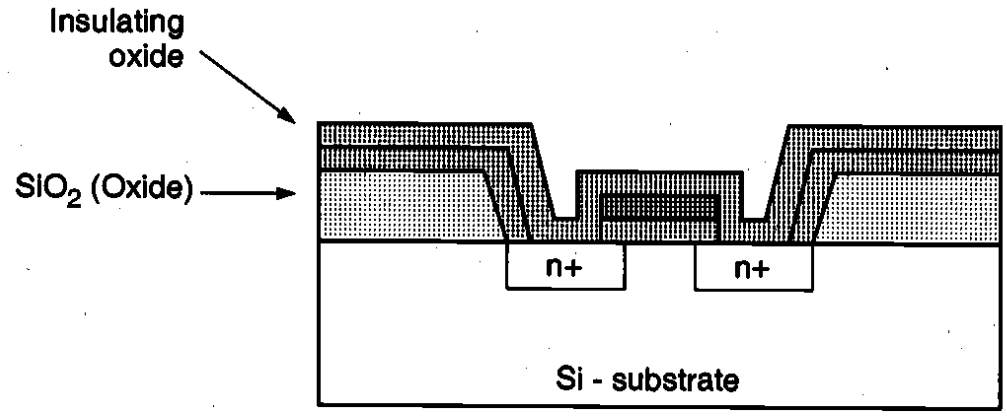


(g)

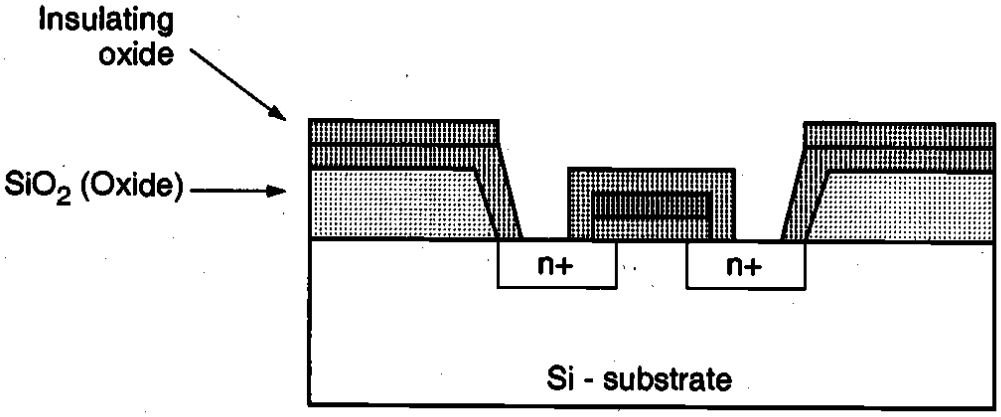
Once the source and drain regions are completed, the entire surface is again covered with an insulating layer of silicon dioxide (Fig. (i)). The insulating oxide layer is then patterned in order to provide contact windows for the drain and source junctions (Fig. (j)). The surface is covered with evaporated aluminum which will form the interconnects (Fig. (k)). Finally, the metal layer is patterned and etched, completing the interconnection of the MOS transistors on the surface (Fig. (1)). Usually, a second (and third) layer of metallic interconnect can also be added on top of this structure by creating another insulating oxide layer, cutting contact (via) holes, depositing, and patterning the metal. The major process steps for the fabrication of an nMOS transistor on p-type silicon substrate are also illustrated in Plate 1 and Plate 2.


Polysilicon
SiO_2 (Oxide)
n+
n+
Si - substrate

(h)


Insulating oxide
SiO_2 (Oxide)
n+
n+
Si - substrate

(i)


Insulating oxide
SiO_2 (Oxide)
n+
n+
Si - substrate

(j)

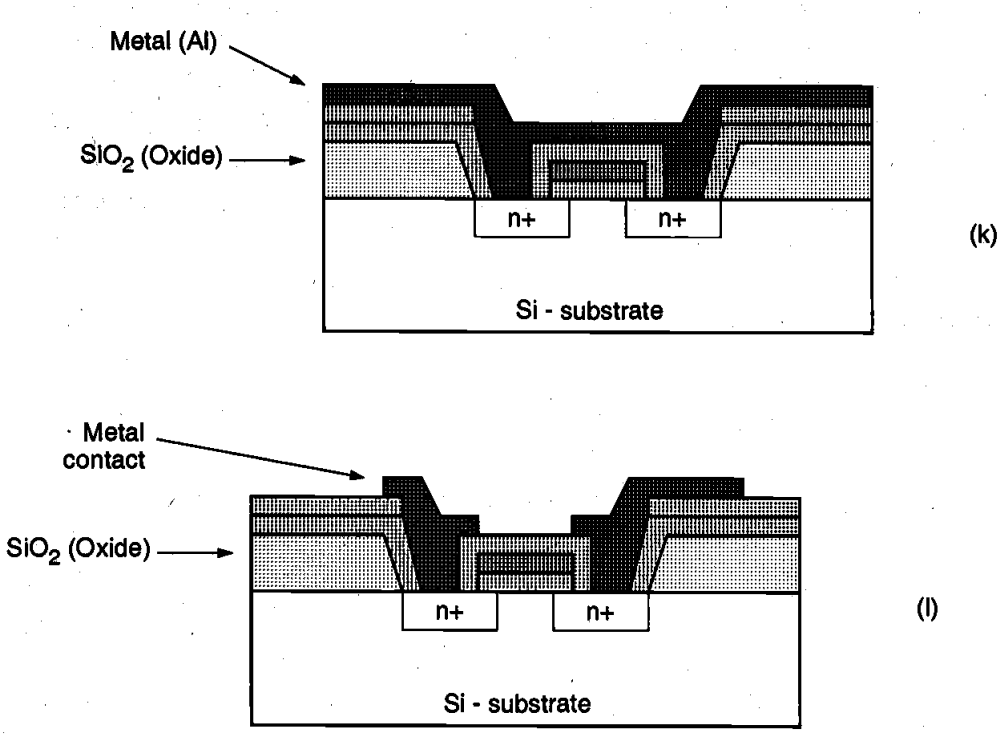
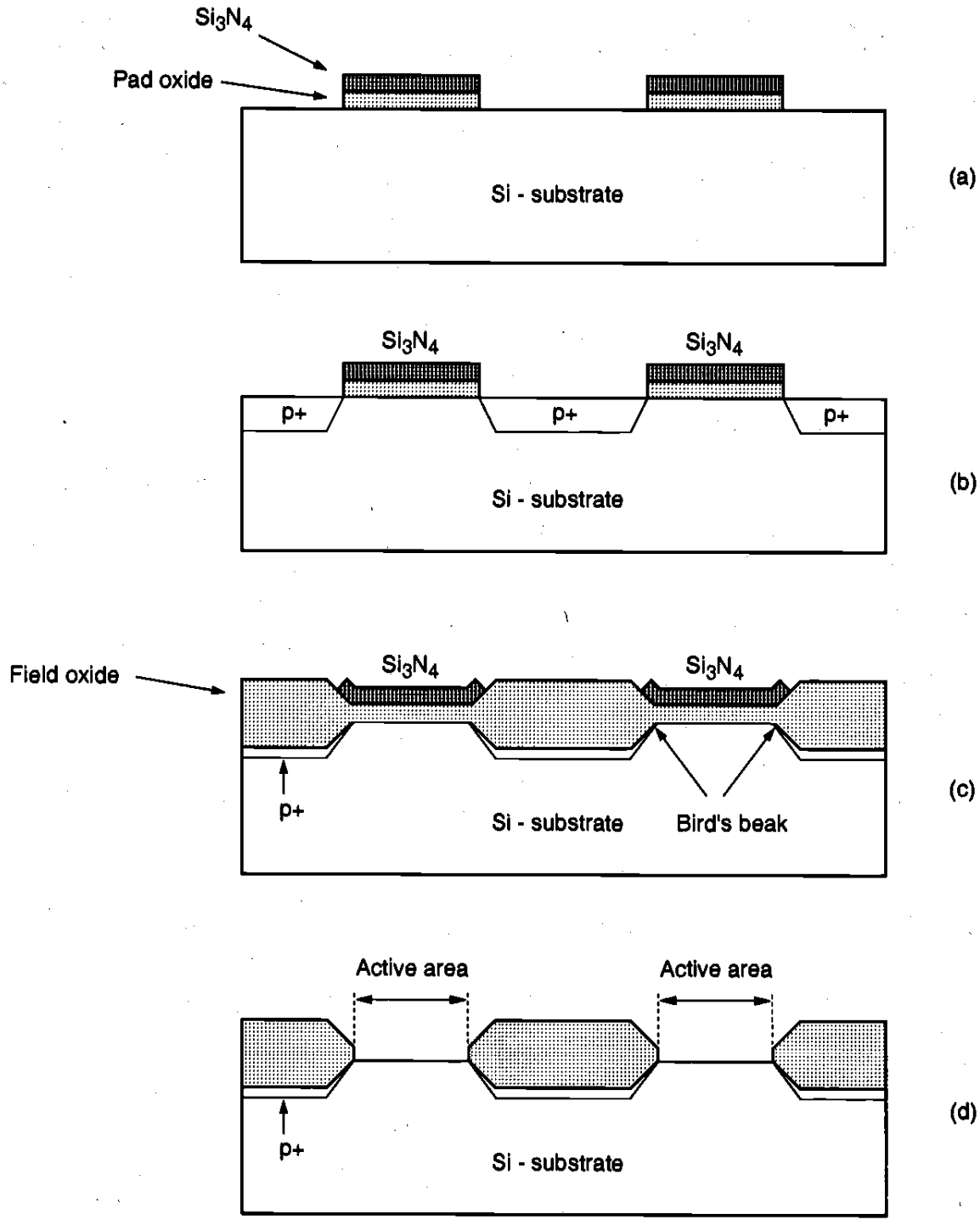


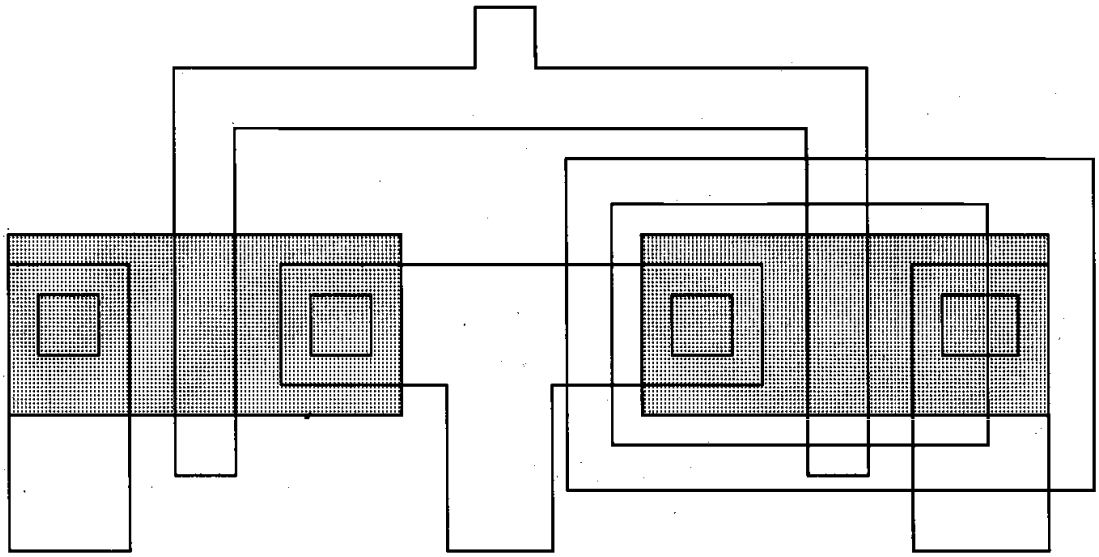
Figure 2.4. Process flow for the fabrication of an n-type MOS transistor

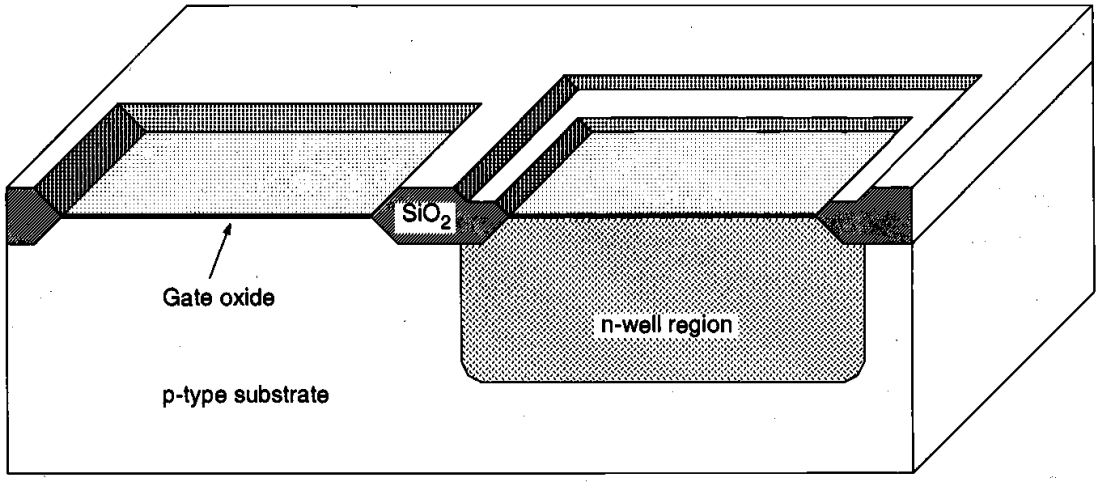
The CMOS n-Well Process

Having examined the basic process steps for pattern transfer through lithography and having gone through the fabrication procedure of a single n-type MOS transistor, we can now return to the generalized fabrication sequence of n-well CMOS integrated circuits, as shown in Fig. 1. In the following figures, some of the important process steps involved in the fabrication of a CMOS inverter will be shown by a top view of the lithographic masks and a cross-sectional view of the relevant areas. The n-well CMOS process starts with a moderately doped (with impurity concentration typically less than 1015 cm-3) p-type silicon substrate. Then, an initial oxide layer is grown on the entire surface. The first lithographic mask defines the n-well region. Donor atoms, usually phosphorus, are implanted through this window in the oxide.

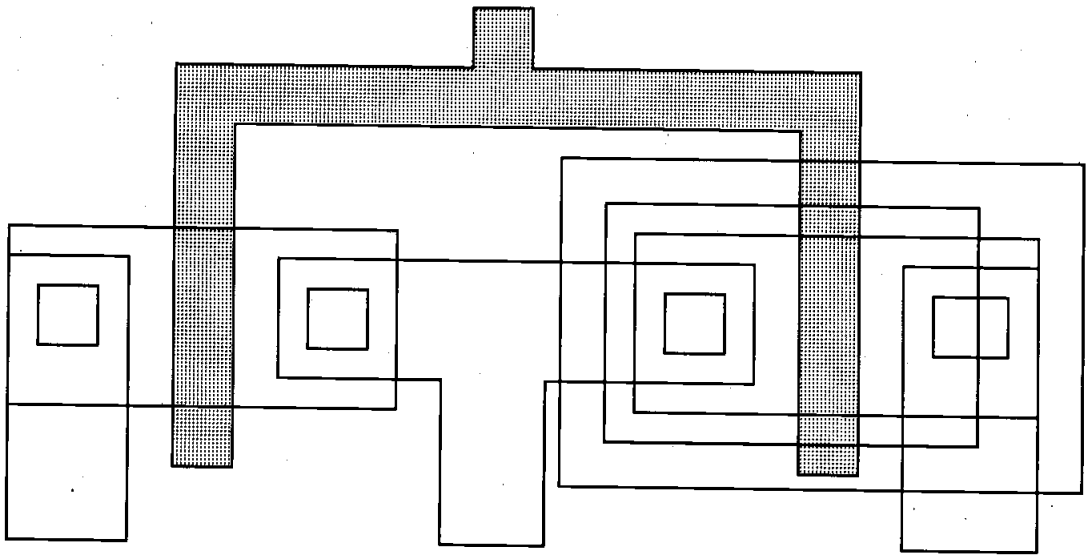
Once the n-well is created, the active areas of the nMOS and pMOS transistors can be defined. Figures 2.6 through 2.11 illustrate the significant milestones that occur during the fabrication process of a CMOS inverter. The main process steps for the fabrication of a CMOS inverter are also illustrated in Plate 3, Plate 4 and Plate 5.

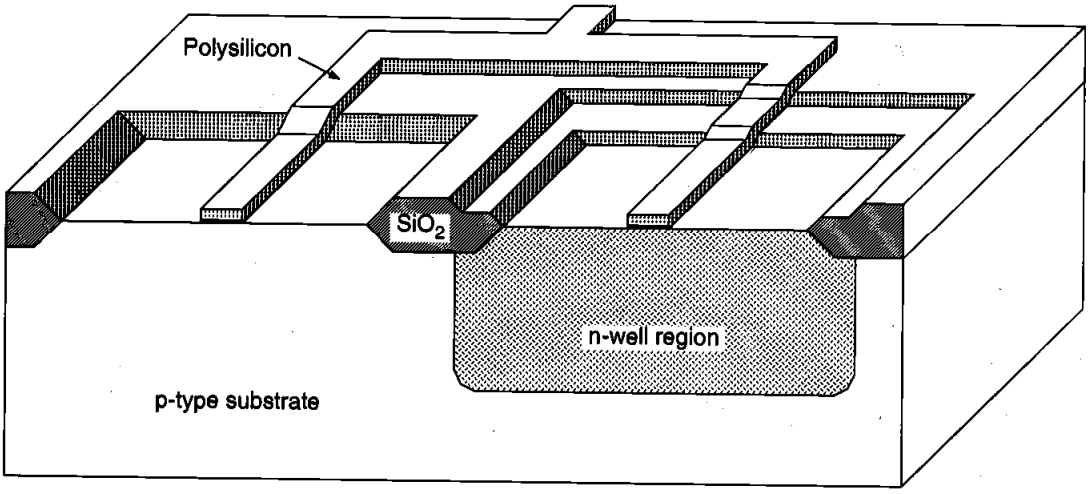

Si_3N_4
Pad oxide
Si - substrate
(a)
Si_3N_4
Si_3N_4
p+
p+
p+
Si - substrate
(b)
Field oxide
Si_3N_4
Si_3N_4
p+
Si - substrate
Bird's beak
(c)
Active area
Active area
p+
Si - substrate
(d)



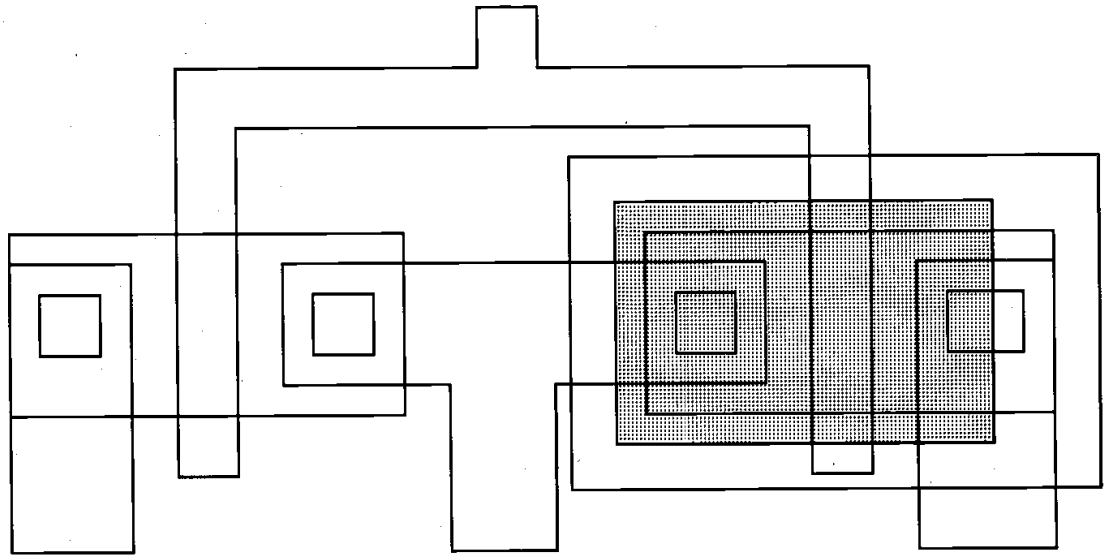


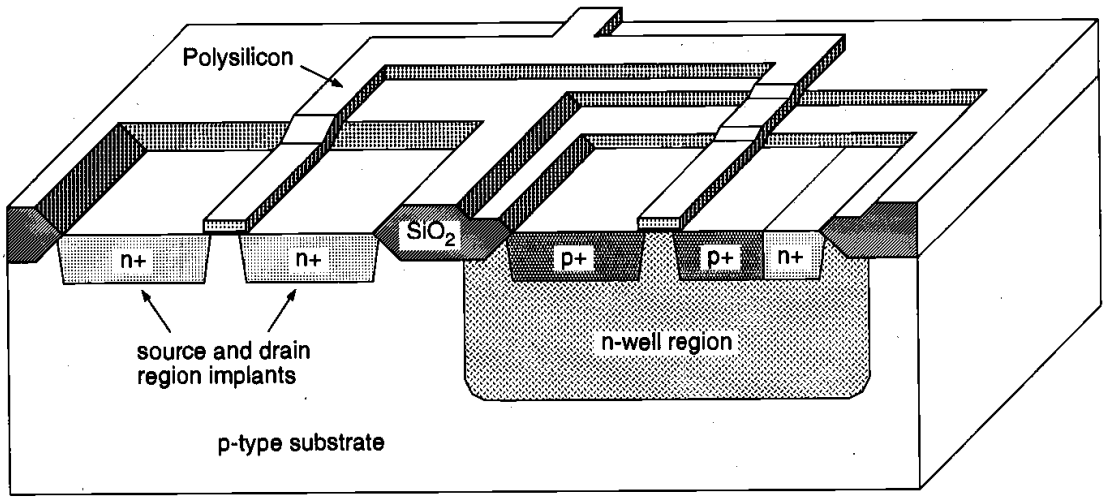
Following the creation of the n-well region, a thick field oxide is grown in the areas surrounding the transistor's active regions, and a thin gate oxide is grown on top of the active regions



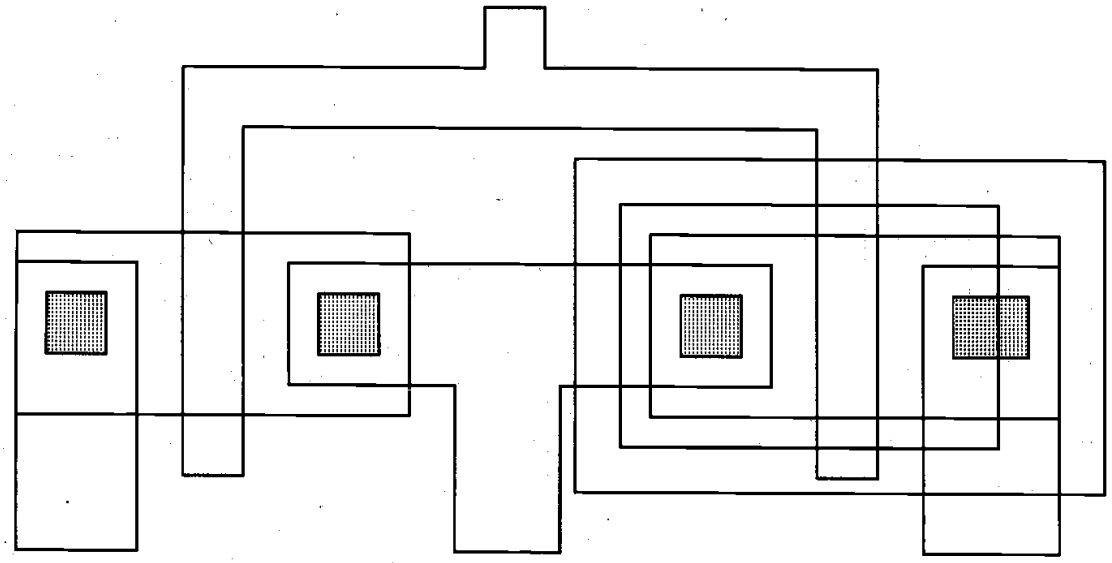


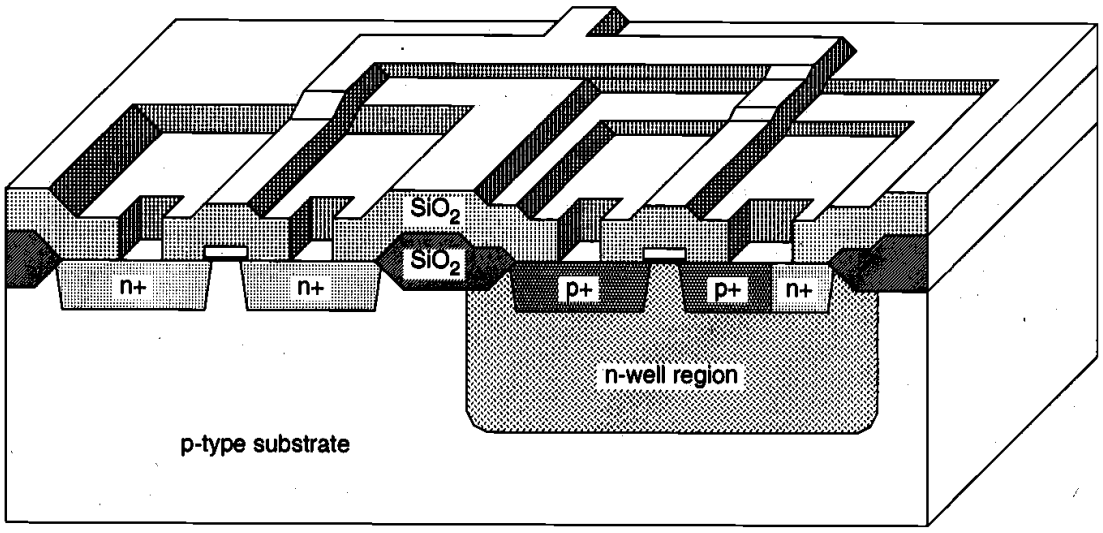
The polysilicon layer is deposited using chemical vapor deposition (CVD) and patterned by dry (plasma) etching. The created polysilicon lines will function as the gate electrodes of the nMOS and the pMOS transistors and their interconnects.



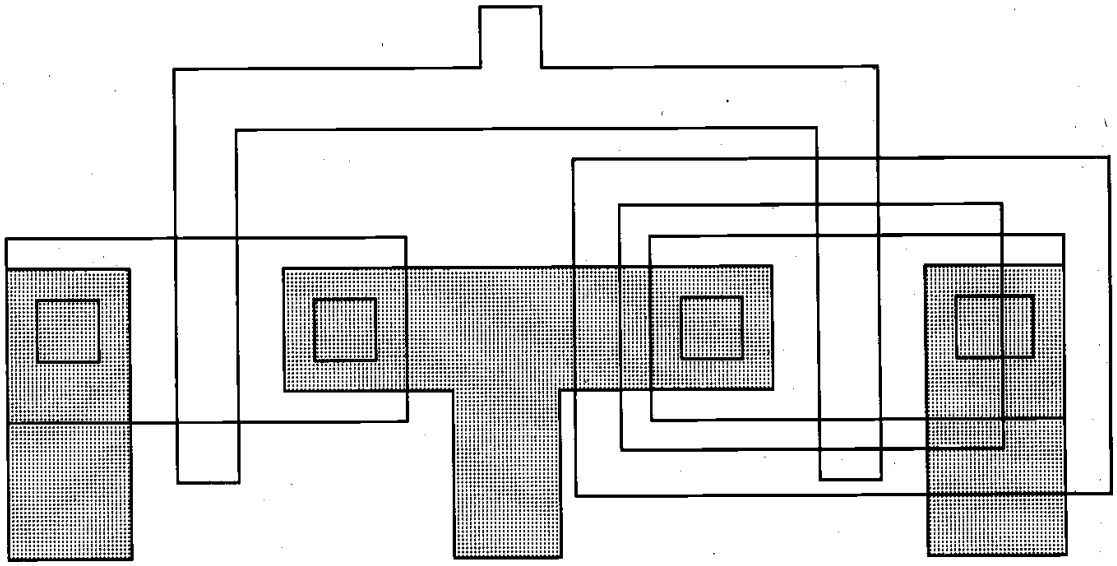


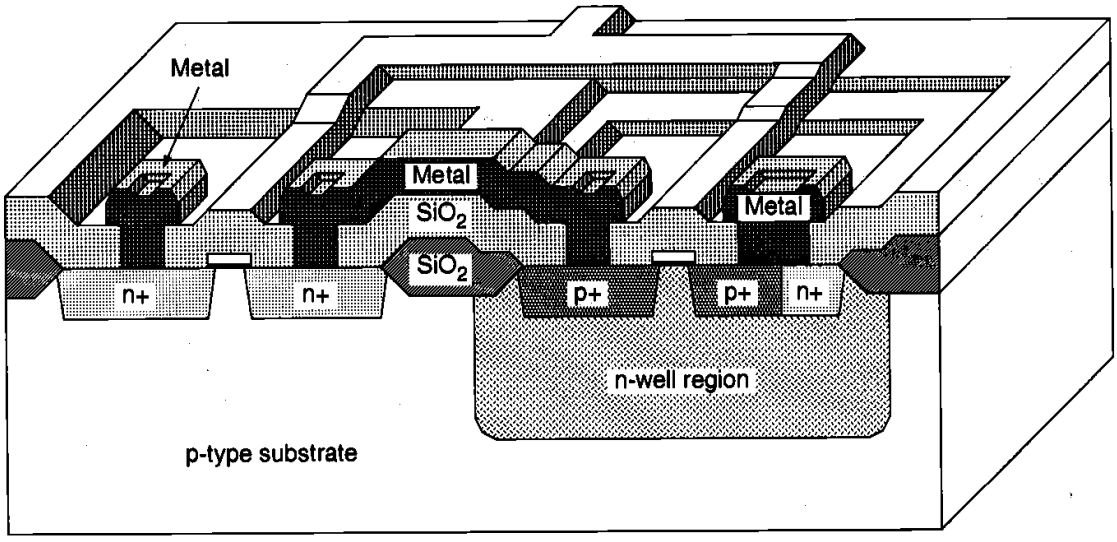
Using a set of two masks, the n+ and p+ regions are implanted into the substrate and into the n-well, respectively.



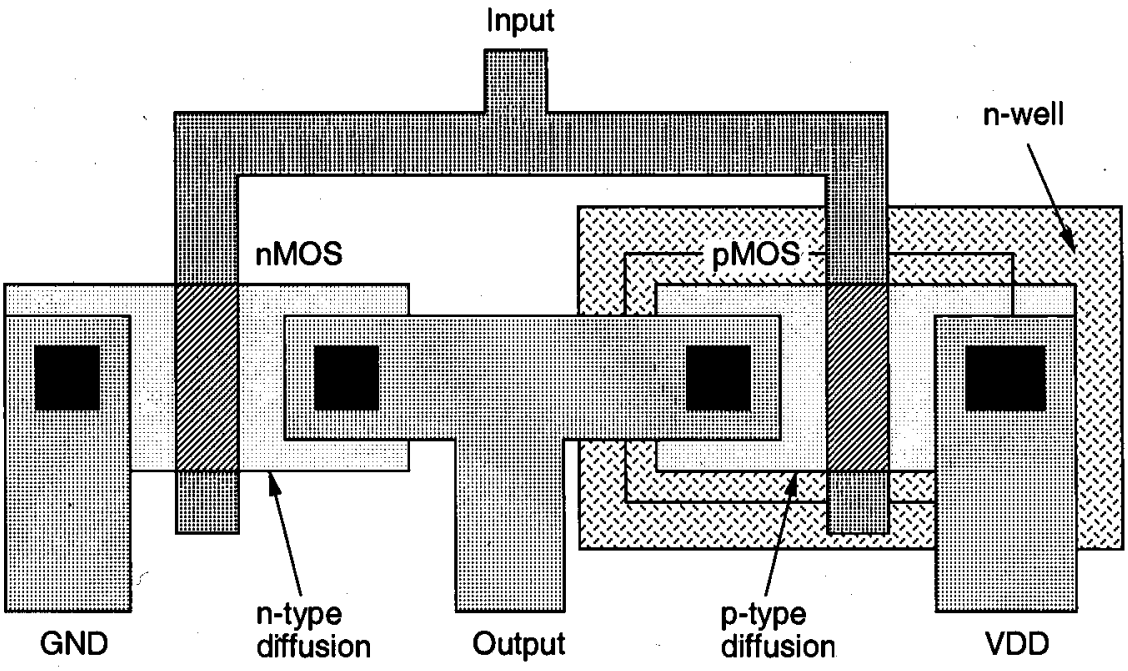


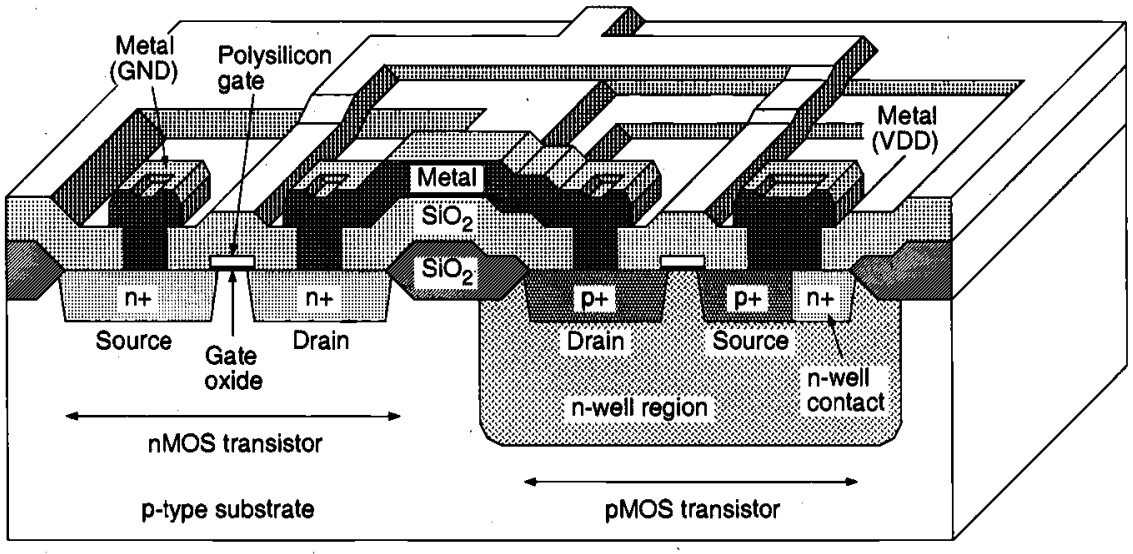
An insulating silicon dioxide layer is deposited over the entire wafer using CVD. Then, the contacts are defined and etched away to expose the silicon or polysilicon contact windows. These contact windows are necessary to complete the circuit interconnections using the metal layer,





Metal (aluminum) is deposited over the entire chip surface using metal evaporation, and the metal lines are patterned through etching. Since the wafer surface is non-planar, the quality and the integrity of the metal lines created in this step are very critical and are ultimately essential for circuit reliability.





The composite layout and the resulting cross-sectional view of the chip, showing one nMOS and one pMOS transistor (in the n-well), and the poly-silicon and metal interconnections. The final step is to deposit the passivation layer (for protection) over the chip, except over wire-bonding pad areas

3.2 Explain Design rules Layout

Layout Design Rules

The physical mask layout of any circuit to be manufactured using a particular process must conform to a set of geometric constraints or rules, which are generally called layout design rules. These rules usually specify the minimum allowable line widths for physical objects on-chip such as metal and polysilicon interconnects or diffusion areas, minimum feature dimensions, and minimum allowable separations between two such features.

The design rules are usually described in two ways:

(i) Micron rules, in which the layout constraints such as minimum feature sizes and minimum allowable feature separations are stated in terms of absolute dimensions in micrometers, or,

(ii) Lambda rules, which specify the layout constraints in terms of a single parameter () and thus allow linear, proportional scaling of all geometrical constraints.

Lambda-based layout design rules were originally devised to simplify the industry standard micron-based design rules and to allow scaling capability for various processes. It must be emphasized,

Active area rules

Minimum active area width 3λ

Minimum active area spacing 3 λ

Polysilicon rules

Minimum poly width 2 λ

Minimum poly spacing 2 λ

Minimum gate extension of poly over active 2 λ

Minimum poly-active edge spacing L λ

(poly outside active area)

Minimum poly-active edge spacing 3 λ

(poly inside active area)

Metal rules

Minimum metal width 3 λ

Minimum metal spacing 3 λ

Contact rules

Poly contact size 2 λ

Minimum poly contact spacing 2 λ

STICK DIAGRAM

It is the Stick and colour representation of the n- mos and c-mos circuit presentation and constructed as per the given rules

We have follow the rules as per the NAND and NOR gate constructions that is series connection in n- mos for NAND and parallel connection in p-mos similarly parallel connection for NOR gate in n-mos and series in p-mos

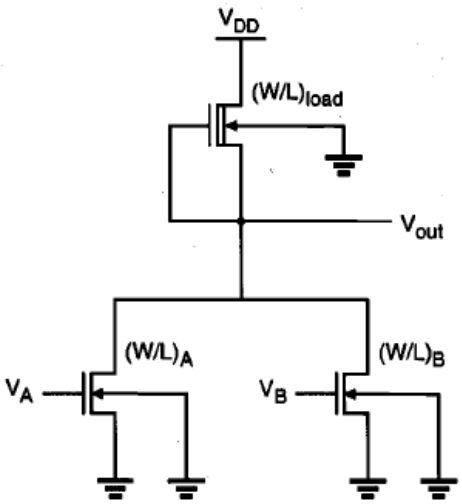


FIG. CIRCUIT DIAGRAM OF NOR GATE

Example- NOR GATE

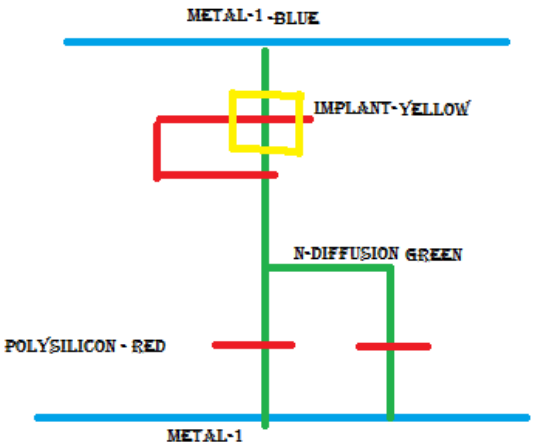


FIG. STICK DIAGRAM OF NOR GATE.

Similarly every other gate can constructed following these conditions .

Full-Custom Mask Layout Design

In this section, the basic mask layout principles for CMOS inverters and logic gates will be presented. The design of physical layout is very tightly linked to overall circuit performance (area, speed, and power dissipation) since the physical structure directly determines the trans-conductance of the transistors, the parasitic capacitances and resistances, and obviously, the silicon area which is used for a certain function. On the other hand, the detailed mask layout of logic gates requires a very intensive and time consuming design effort, which is justifiable only in special circumstances where the area and/or the performance of the circuit must be optimized under very tight constraints. Therefore, automated layout generation (e.g., using a standard cell library, computer aided placement-and-routing) is typically preferred for the design of most digital VLSI circuits. In order to judge the physical constraints and limitations, however, the VLSI designer must also have a good understanding of the physical mask layout process.

Chapter 4 MOS Transistor

4.1. The Metal Oxide Semiconductor (MOS) Structure

We will start our investigation by considering the electrical behavior of the simple two-. terminal MOS structure shown in Fig. 3.1. Note that the structure consists of three layers: The metal gate electrode, the insulating oxide (SiO2) layer, and the p-type bulk semiconductor (Si), called the substrate. As such, the MOS structure forms a capacitor, with the gate and the substrate acting as the two terminals (plates) and the oxide layer as the dielectric. The thickness of the silicon dioxide layer is usually between 10 nm and 50 nm. The carrier concentration and its local distribution within the semiconductor substrate can now be manipulated by the external voltages applied to the gate and substrate terminals. A basic understanding of the bias conditions for establishing different carrier concentrations in the substrate will also provide valuable insight into the operating conditions of more complicated MOSFET structures.

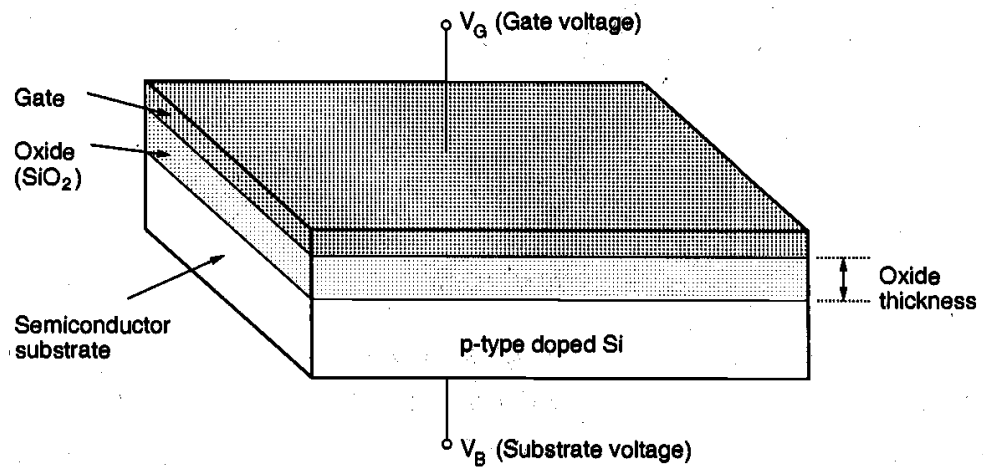


Figure 3.1. Two-terminal MOS structure.

Consider first the basic electrical properties of the semiconductor (Si) substrate, which acts as one of/the electrodes of the MOS capacitor. The equilibrium concentrations of mobile carriers in a semiconductor always obey the *Mass Action Law* given by

$$n \cdot p = n_i^2 \qquad (3.1)$$

Here, *n* and p denote the mobile carrier concentrations of electrons and holes, respectively, and n_i denotes the intrinsic carrier concentration of silicon, which is a function

of the temperature T. At room temperature, i.e., *T*= 300 K, *ni* is approximately equal to 1.45 x 1010 cm-3. Assuming that the substrate is uniformly doped with an acceptor (e.g., Boron) concentration *NA*, the equilibrium electron and hole concentrations in the p-type substrate are approximated by

$$n_{po} \cong \frac{n_i^2}{N_A}$$
$$p_{po} \cong N_A \qquad (3.2)$$

The doping concentrationNA is typically on the order of 1015 to 1016 cm-3; thus, it is much

greater than the intrinsic carrier concentration n_i

The MOS System under External Bias

We now turn our attention to the electrical behavior of the MOS structure under externally applied bias voltages. Assume that the substrate voltage is set at VB = 0, and let the gate voltage be the controlling parameter. Depending on the polarity and the magnitude of VG, three different operating regions can be observed for the MOS system: accumulation, depletion, and inversion. If a negative voltage VG is applied to the gate electrode, the holes in the p-type substrate are attracted to the semiconductor-oxide interface. The majority carrier concentration near the surface becomes larger than the equilibrium hole concentration in the substrate; hence, this condition is called carrier accumulation on the surface (Fig. 3.5). Note that in this case, the oxide electric field is directed towards the gate electrode. The negative surface potential also causes the energy bands to bend upward near the surface.

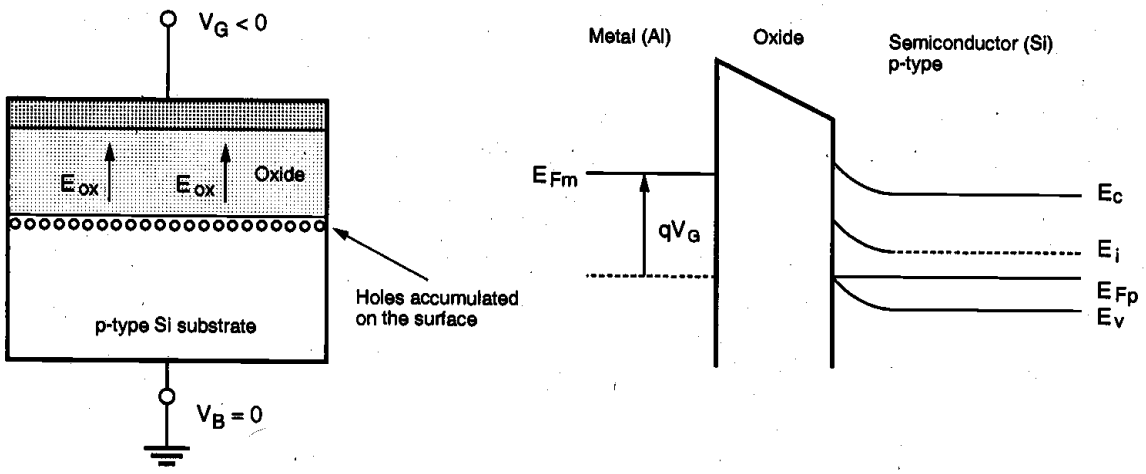
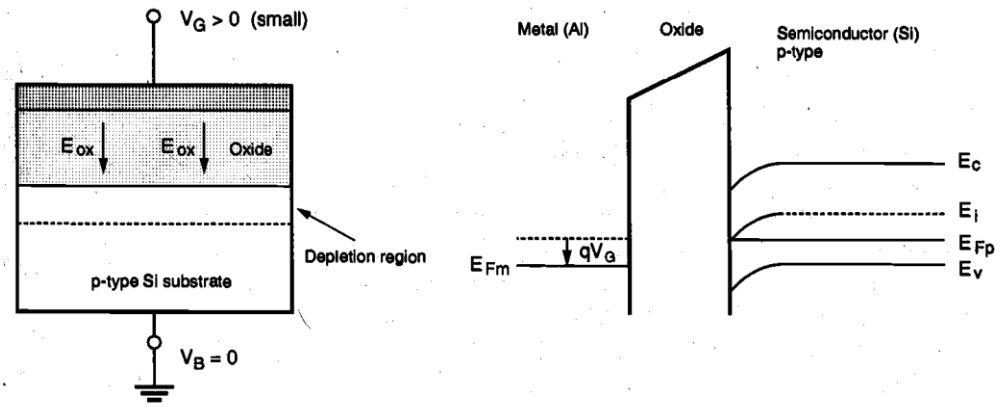


Figure 3.5. The cross-sectional view and the energy band diagram of the MOS structure

While the hole density near the surface increases as a result of the applied negative gate bias, the electron (minority carrier) concentration decreases as the negatively charged electrons are pushed deeper into the substrate operating in accumulation region.



$$dQ = -q \cdot N_A \cdot dx \tag{3.7}$$

The *change* in surface potential required to displace this charge sheet dQ by a distance x_d away from the surface can be found by using the Poisson equation.

$$d\phi_s = -x \cdot \frac{dQ}{\varepsilon_{Si}} = \frac{q \cdot N_A \cdot x}{\varepsilon_{Si}} dx \tag{3.8}$$

Integrating (3.7) along the vertical dimension (perpendicular to the surface) yields

$$\int_{\phi_F}^{\phi_s} d\phi_s = \int_0^{x_d} \frac{q \cdot N_A \cdot x}{\varepsilon_{Si}} \, dx \tag{3.9}$$

$$\phi_s - \phi_F = \frac{q \cdot N_A \cdot x_d^2}{2\,\varepsilon_{Si}} \tag{3.10}$$

Thus, the depth of the depletion region is

$$x_d = \sqrt{\frac{2\varepsilon_{Si} \cdot |\phi_s - \phi_F|}{q \cdot N_A}} \tag{3.11}$$

and the depletion region charge density, which consists solely of fixed acceptor ions in this region, is given by the following expression

$$Q = -q \cdot N_A \cdot x_d = -\sqrt{2q \cdot N_A \cdot \varepsilon_{Si} \cdot |\phi_s - \phi_F|} \tag{3.12}$$

Now consider the next case in which a small positive gate bias VG is applied to the gate electrode. Since the substrate bias is zero, the oxide electric field will be directed towards the substrate in this case. The positive surface potential causes the energy bands to bend downward near the surface, as shown in Fig. 3.6. The majority carriers, i.e., the holes in the substrate, will be repelled back into the substrate as a result of the positive gate bias, and these holes will leave negatively charged fixed acceptor ions behind. Thus, a depletion region is created near the surface. Note that under this bias condition, the region near the semiconductor-oxide interface is nearly devoid of all mobile carriers.

To complete our qualitative overview of different bias conditions and their effects upon the MOS system, consider next a further increase in the positive gate bias. As a result of the increasing surface potential, the downward bending of the energy bands will increase as well. Eventually, the mid-gap energy level Ei becomes smaller than the Fermi level EFP on the surface, which means that the substrate semiconductor in this region becomes n-type. Within this thin layer, the electron density is larger than the majority hole density, since the positive gate potential attracts additional minority carriers (electrons) from the bulk substrate to the surface (Fig. 3.7).

$$x_{dm} = \sqrt{\frac{2 \cdot \varepsilon_{Si} \cdot |2\phi_F|}{q \cdot N_A}} \qquad (3.13)$$

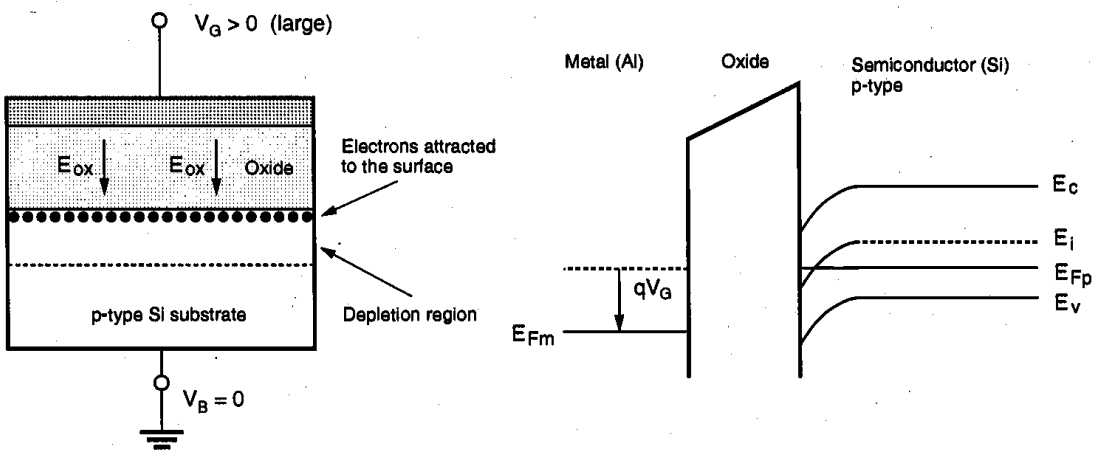


Figure 3.7. The cross-sectional view and the energy band diagram of the MOS structure in surface inversion, under larger gate bias voltage.

The n-type region created near the surface by the positive gate bias is called the inversion layer, and this condition is called surface inversion. It will be seen that the thin inversion layer on the surface with a large mobile electron concentration can be utilized for conducting current between two terminals of the MOS transistor. The creation of a conducting surface inversion layer through externally applied gate bias is an essential phenomenon for current conduction in MOS transistors. In the following we will examine the structure and the operation of the MOS Field Effect Transistor (MOSFET).

4.2 MOSFET Current-Voltage Characteristics

The analytical derivation of the MOSFET current-voltage relationships for various bias conditions requires that several approximations be made to simplify the problem.

Without these simplifying assumptions, analysis of the actual three-dimensional MOS system would become a very complex task and would prevent the derivation of closed form current-voltage equations. In the' following, we will use the *gradual channel approximation* (GCA) for establishing the MOSFET current-voltage relationships, which will effectively reduce the analysis to a one-dimensional current-flow problem. This will allow us to devise relatively simple current equations that agree well with experimental results. As in every approximate approach, however, the GCA also has its limitations, especially for small-geometry MOSFETs. We will investigate the most significant limitations and examine some of the possible remedies.

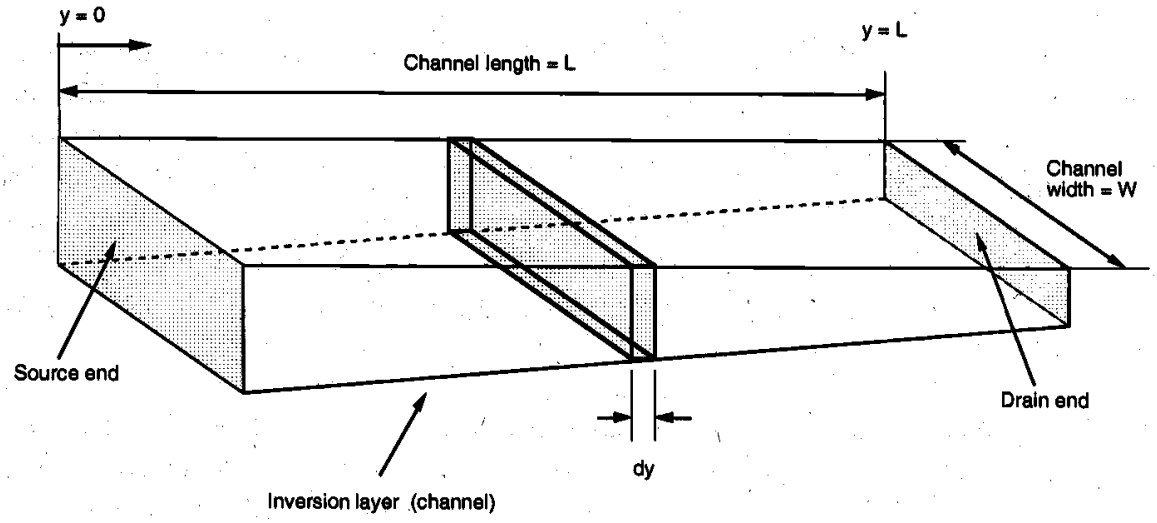


Figure 3.16. Simplified geometry of the surface inversion layer (channel region).

Now consider the incremental resistance *dR* of the differential channel segment shown in Fig. 3.16. Assuming that all mobile electrons in the inversion layer have a constant

surface mobility jun, the incremental resistance can be expressed as follows. Note that the minus sign is due to the negative polarity of the inversion layer charge Q_l.

$$dR = -\frac{dy}{W \cdot \mu_n \cdot Q_l(y)} \quad \text{.....(1)}$$

The *electron surface mobility yun* used , depends on the doping concentration of the channel region, and its magnitude is typically about one-half of that of the bulk electron mobility. We will assume that the channel current density is uniform across this segment. According to our one-dimensional model, the channel (drain) current *ID* flows between the source and the drain regions in the y-coordinate direction. Applying Ohm's law for this segment yields the voltage drop along the incremental segment *dy*, in the ydirection.

$$dV_c = I_D \cdot dR = -\frac{I_D}{W \cdot \mu_n \cdot Q_I(y)} \cdot dy \quad \text{.....(2)}$$

This equation can now be integrated along the channel, i.e., from y = 0 to y = L, using the

boundary conditions given in above equation

$$\int_0^L I_D \cdot dy = -W \cdot \mu_n \int_0^{V_{DS}} Q_I(y) \cdot dV_c \quad \text{.....(3)}$$

The left-hand side of this equation is simply equal to L ID. The integral on the right-hand side is evaluated by replacing Q_l in

$$Q_l(y) = -Cox[VGS \text{ v } VC(Y) - VTO]$$

$$I_D = \frac{\mu_n \cdot C_{ox}}{2} \cdot \frac{W}{L} \cdot \left[2 \cdot (V_{GS} - V_{T0}) V_{DS} - V_{DS}^2\right] \quad \text{.....(4)}$$

Equation (3.32) represents the drain current ID as a simple second-order function of the

two external voltages, VGS and VDS. This current equation can also be rewritten as

$$I_D = \frac{k'}{2} \cdot \frac{W}{L} \cdot \left[2 \cdot (V_{GS} - V_{T0}) V_{DS} - V_{DS}^2\right]$$

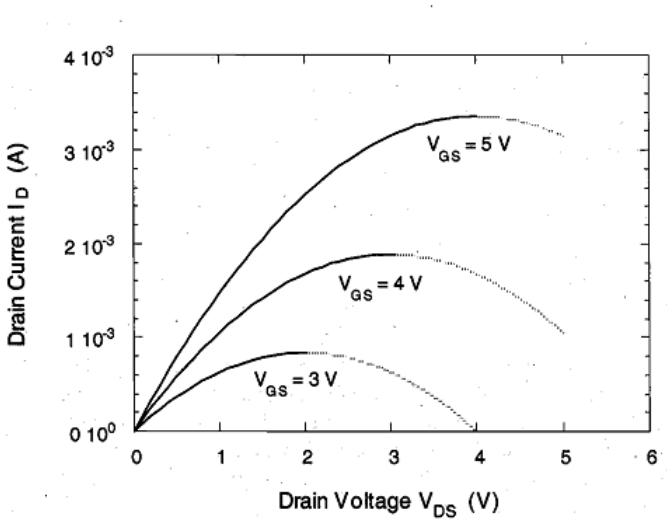
$$I_D = \frac{k}{2} \cdot \left[2 \cdot (V_{GS} - V_{T0}) V_{DS} - V_{DS}^2\right] \quad \text{.....(5)}$$

where the parameters k and k' are defined as

$$k' = \mu_n \cdot C_{ox}$$

$$k = k' \cdot \frac{W}{L} \qquad \text{.....(6)}$$

The drain current equation given in (5) is the simplest analytical approximation for the MOSFET current-voltage relationship. Note that, in addition to the process dependent constants k' and V, the current-voltage relationship is also affected by the device dimensions, W and L. In fact, we will see that the ratio of WIL is one of the most important design parameters in MOS digital circuit design. Now, we must determine the region of validity for this equation and what this means for the practical use of the equation.



The drain current-drain voltage curves shown above reach their peak value for VDS = V_{GS} - V_{TO} Beyond this maximum, each curve exhibits a negative differential conductance, which is not observed in actual MOSFET current-voltage measurements (section shown by the dashed lines). We must remember now that the drain current equation (4) has been derived under the following voltage assumptions,

$$V_{GS} \geq V_{T0}$$
$$V_{GD} = V_{GS} - V_{DS} \geq V_{T0}$$

$$V_{DS} \geq V_{DSAT} = V_{GS} - V_{T0} \qquad \text{.....(7)}$$

Also, drain current measurements with constant V_S show that the current I_D does not show much variation as a function of the drain voltage. V_{DS} beyond the saturation boundary, but rather remains approximately constant around the peak value reached for $V_{DS} = VD_{SAT}$ This saturation drain current level can be found simply by substituting (7) for VDS in (1).

$$I_D(sat) = \frac{\mu_n \cdot C_{ox}}{2} \cdot \frac{W}{L} \cdot \left[2 \cdot (V_{GS} - V_{T0}) \cdot (V_{GS} - V_{T0}) - (V_{GS} - V_{T0})^2\right]$$

$$= \frac{\mu_n \cdot C_{ox}}{2} \cdot \frac{W}{L} \cdot (V_{GS} - V_{T0})^2 \quad \text{.....(8)}$$

Thus, the drain current ID becomes a function only of the gate-to-source voltage VGS, beyond the saturation boundary. Note that this constant saturation current approximation is not very accurate in reality, and that the saturation-region drain current continues to have a certain dependence on the drain voltage. For simple hand calculations, however,(8) provides a sufficiently accurate approximation of the MOSFET drain (channel) current in saturation.

4.3 MOSFET scaling and small geometry effects.

MOSFET Scaling and Small-Geometry Effects The design of high-density chips in MOS VLSI (Very Large Scale Integration) technology requires that the packing density of MOSFETs used in the circuits is as high as possible and, consequently, that the sizes of the transistors are as small as possible. The reduction of the size, i.e., the dimensions of MOSFETs, is commonly referred to as scaling.

There are two basic types ofsize-reduction strategies:

full scaling (also called constant-field scaling)

constant voltage scaling.

Full Scaling (Constant-Field Scaling)

This scaling option attempts to preserve the magnitude of internal electric fields in the MOSFET, while the dimensions are scaled down by a factor of S. To achieve this goal, all potentials must be scaled down proportionally, by the same scaling factor. Note that this potential scaling also affects the threshold voltage V_{TO} Finally, the Poisson equation describing the relationship between charge densities and electric fields dictates that the charge densities must be increased by a factor of S in order to maintain the field conditions. Table 1 lists the scaling factors for all significant dimensions, potentials, and doping densities of the MOS transistor.

Quantity	Before Scaling	After Scaling
Channel length	L	$L' = L / S$
Channel width	W	$W' = W / S$
Gate oxide thickness	t_{ox}	$t_{ox}' = t_{ox} / S$
Junction depth	x_j	$x_j' = x_j / S$
Power supply voltage	V_{DD}	$V_{DD}' = V_{DD} / S$
Threshold voltage	V_{T0}	$V_{T0}' = V_{T0} / S$
Doping densities	N_A N_D	$N_A' = S \cdot N_A$ $N_D' = S \cdot N_D$

Table 1. Full scaling of MOSFET dimensions, potentials, and doping densities.

Constant-Voltage Scaling

While the full scaling strategy dictates that the power supply voltage and all terminal voltages be scaled down proportionally with the device dimensions, the scaling of voltages may not be very practical in many cases. In particular, the peripheral and interface circuitry may require certain voltage levels for all input and output voltages, which in turn would necessitate multiple power supply voltages and complicated level shifter arrangements. For these reasons, constant-voltage scaling is usually preferred over full scaling.

Quantity	Before Scaling	After Scaling
Oxide capacitance	C_{ox}	$C_{ox}' = S \cdot C_{ox}$
Drain current	I_D	$I_D' = I_D / S$
Power dissipation	P	$P' = P / S^2$
Power density	$P / Area$	$P' / Area' = P / Area$

Table 2. Effects of full scaling upon key device characteristics.

Explain MOSFET capacitances.

The majority of the topics covered in this chapter has been related to the steady-state behavior of the MOS transistor. The current-voltage characteristics investigated here can be

applied for investigating the DC response of MOS circuits under various operating conditions.

the gate electrode overlaps both the source region and the drain region at the edges. The two overlap capacitances that arise as a result of this structural arrangement are called CGD (overlap) and CGS (overlap), respectively. Assuming that both the source and the drain diffusion regions have the same width W, the overlap capacitances can be found as

$$C_{GS}(overlap) = C_{ox} \cdot W \cdot L_D$$
$$C_{GD}(overlap) = C_{ox} \cdot W \cdot L_D$$

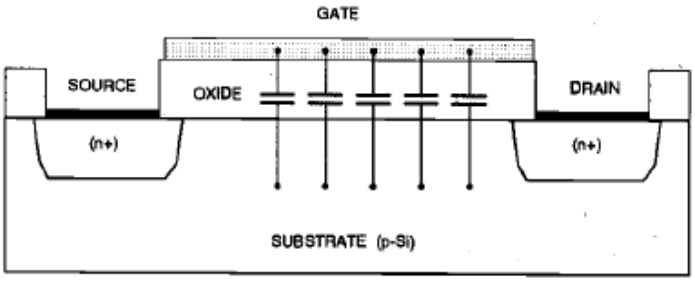
$$C_{ox} = \frac{\varepsilon_{ox}}{t_{ox}}$$

Note that both of these overlap capacitances do not depend on the bias conditions, i.e., they are voltage-independent. Now consider the capacitances which result from the interaction between the gate voltage and the channel charge. Since the channel region is connected to the source, the drain, and the substrate, we can identify three capacitances between the gate and these regions, i.e., Cgs, C d and C b respectively. Notice that in reality, the gate-to-channel capacitance is distributed and voltage-dependent. Then, the gate-to-source capacitance Cgs is actually the gate-to-channel capacitance seen between the gate and the source terminals; the gate-to-drain capacitance C ad is actually the gate-to-channel capacitance seen between the gate and the drain terminals. A simplified view of their bias-dependence can be obtained by observing the conditions in the channel region during cut-off, linear, and saturation modes.

In cut-off mode (Fig. (a)), the surface is not inverted. Consequently, there is no conducting channel that links the surface to the source and to the drain. Therefore, the gate-to-source and the gate-to-drain capacitances are both equal to zero: Cgs = Cgd= 0. The gate-to-substrate capacitance can be approximated by

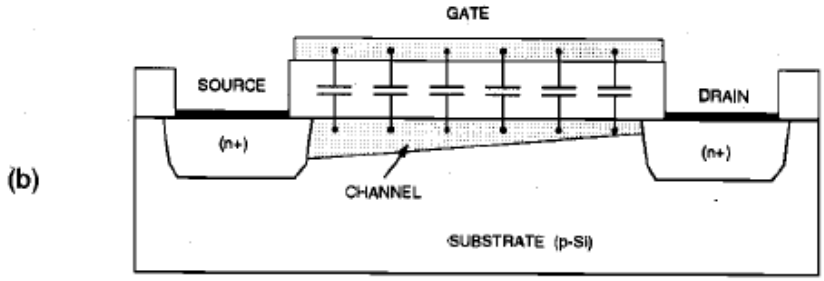
$$C_{gb} = C_{ox} \cdot W \cdot L$$

(a)



In linear-mode operation, the inverted channel extends across the MOSFET, between the source and the drain (Fig. (b)). This conducting inversion layer on the surface effectively shields the substrate from the gate electric field; thus, Cgb = 0. In this case, the distributed gate-to-channel capacitance may be viewed as being shared equally between the source and the drain, yielding

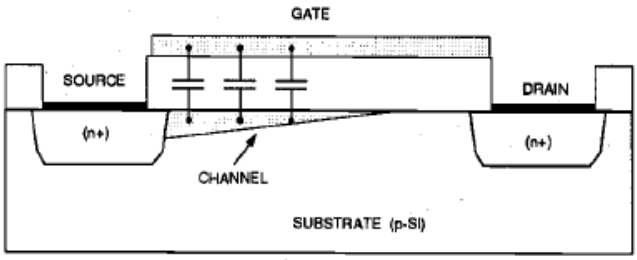
$$C_{gs} \cong C_{gd} \cong \frac{1}{2} \cdot C_{ox} \cdot W \cdot L$$



(b)

When the MOSFET is operating in saturation mode, the inversion layer on the surface does not extend to the drain, but it is pinched off (Fig. (c)). The gate-to-drain cca pacitance component is therefore equal to zero (Cgd = 0) . Since the source is still linked to the conducting channel, its shielding effect also forces the gate-to-substrate capacitance to be zero, Cgb = 0. Finally, the distributed gate-to-channel capacitance as seen between the gate and the source can be approximated by

$$C_{gs} \cong \frac{2}{3} \cdot C_{ox} \cdot W \cdot L$$



(c)

Figure .. Schematic representation of MOSFET oxide capacitances during (a) cut-off, (b) linear, and (c) saturation modes.

Capacitance	Cut-off	Linear	Saturation
C_{gb} (total)	$C_{ox}WL$	0	0
C_{gd} (total)	$C_{ox}WL_D$	$\frac{1}{2}C_{ox}WL + C_{ox}WL_D$	$C_{ox}WL_D$
C_{gs} (total)	$C_{ox}WL_D$	$\frac{1}{2}C_{ox}WL + C_{ox}WL_D$	$\frac{2}{3}C_{ox}WL + C_{ox}WL_D$

Table for Approximate oxide capacitance values for three operating modes of the MOS

4.4 Modeling of MOS Transistors including Basic concept the SPICE level-1 models, the level-2 and level-3 model

The LEVEL 1 Model Equations

The LEVEL 1 model is the simplest current-voltage description of the MOSFET, which is basically the GCA-based quadratic model originally proposed by Shichman and Hodges. The equations used for the LEVEL 1 n-channel MOSFET model in SPICE are as follows.

Linear Region

$$I_D = \frac{k'}{2}\cdot\frac{W}{L_{eff}}\cdot\left[2\cdot(V_{GS}-V_T)V_{DS}-V_{DS}^2\right]\cdot(1+\lambda V_{DS}) \quad \text{for} \quad V_{GS}\geq V_T$$

$$\text{and} \quad V_{DS} < V_{GS}-V_T \qquad \text{.....(1)}$$

Saturation Region

$$I_D = \frac{k'}{2}\cdot\frac{W}{L_{eff}}\cdot(V_{GS}-V_T)^2\cdot(1+\lambda\cdot V_{DS}) \qquad \text{for} \quad V_{GS}\geq V_T$$

$$\text{and} \quad V_{DS}\geq V_{GS}-V_T$$

where the threshold voltage V_T is calculated as

$$V_T = V_{T0}+\gamma\cdot\left(\sqrt{|2\phi_F|+V_{SB}}-\sqrt{|2\phi_F|}\right)$$

Note that the effective channel length *Le* used in these equations is found as follows:

$$L_{eff} = L-2\cdot L_D$$

The LEVEL 2 Model Equations

To obtain a more accurate model for the drain current, it is necessary to eliminate some of the simplifying assumptions made in the original GCA analysis. Specifically, the bulk depletion charge must be calculated by taking into account its dependence on the channel voltage. Solving the drain current equation using the voltage-dependent bulk charge term, the following current-voltage characteristics can be obtained:

$$I_D = \frac{k'}{(1-\lambda\cdot V_{DS})}\cdot\frac{W}{L_{eff}}\cdot\left\{\left(V_{GS}-V_{FB}-|2\phi_F|-\frac{V_{DS}}{2}\right)\cdot V_{DS} - \frac{2}{3}\cdot\gamma\cdot\left[(V_{DS}-V_{BS}+|2\phi_F|)^{3/2}-(-V_{BS}+|2\phi_F|)^{3/2}\right]\right\}$$

the saturation voltage V_{DSAT} can be calculated as

$$V_{DSAT} = V_{GS} - V_{FB} - |2\phi_F| + \gamma^2 \cdot \left(1 - \sqrt{1 + \frac{2}{\gamma^2} \cdot (V_{GS} - V_{FB})}\right)$$

The saturation mode current is

$$I_D = I_{Dsat} \cdot \frac{1}{(1 - \lambda \cdot V_{DS})}$$

The LEVEL 3 Model Equations

The LEVEL 3 model has been developed for simulation of short-channel MOS' transistors; it can represent the characteristics of MOSFETs quite precisely for channel lengths down to 2 gum. The current-voltage equations are formulated in the same way as for the LEVEL 2 model.

$$I_D = \mu_s \cdot C_{ox} \cdot \frac{W}{L_{eff}} \cdot \left(V_{GS} - V_T - \frac{1 + F_B}{2} \cdot V_{DS}\right) \cdot V_{DS}$$

Where

$$F_B = \frac{\gamma \cdot F_s}{4 \cdot \sqrt{|2\phi_F| + V_{SB}}} + F_n$$

The empirical parameter *FB* expresses the dependence of the bulk depletion charge on the

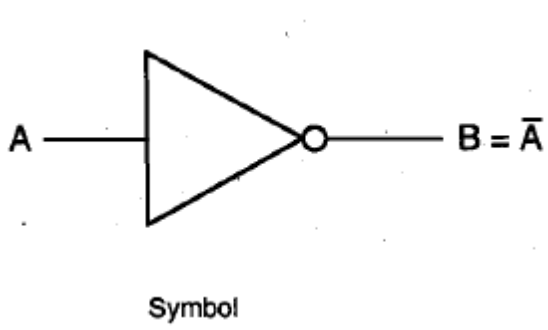
three-dimensional geometry of the MOSFET. Here, the parameters *VT, Fs,* and *us* are influenced by the short-channel effects, while the parameter Fn is influenced by the narrow-channel effects. The dependence of the surface mobility on the gate electric field is simulated as follows:

$$\mu_s = \frac{\mu}{1 + \theta \cdot (V_{GS} - V_T)}$$

Chapter 5 MOS Inverter

5.1 Basic NMOS inverters, characteristics,

The logic symbol and the truth table of the ideal inverter are shown in Fig. 1. In MOS inverter circuits, both the input variable A and the output variable B are represented by node voltages, referenced to the ground potential. Using positive logic convention, the Boolean (or logic) value of "1" can be represented by a high voltage of V_{DD}, and the Boolean (or logic) value of "0" can be represented by a low voltage of 0. The DC voltage transfer characteristic (V_{TC}) of the ideal inverter circuit is shown in Fig. 2. The voltage V_{th} is called the inverter threshold voltage. Note that for any input voltage between 0 and Vth= V_{DD}/2 , the output voltage is equal to V_{DD} (logic" 1).



Symbol

A	B
0	1
1	0

Truth Table

Fig.1,2 Logic symbol and truth table of the inverter

The output switches from V_{DD} to 0 when the input is equal to V_{th}. For any input voltage between V_{th} and V_{DD}, the output voltage assumes a value of 0 (logic "0"). Thus, an input voltage 0 < Vi. < V is interpreted by this ideal inverter as a logic "0," while an input voltage V_{th} <Vin < V_{DD} s interpreted as a logic " 1." The DC characteristics of actual inverter circuits will obviously differ in various degrees from the ideal characteristic shown in Fig. 2. The accurate estimation and the manipulation of the shape of V_{TC} for various inverter types are actually important parts of the design process.

Figure 3 shows the generalized circuit structure of an nMOS inverter. The input voltage of the inverter circuit is also the gate-to-source voltage of the nMOS transistor (V_{in} = V_{Gs}), while the output voltage of the circuit is equal to the drain-to-source voltage (V_{out}= V_{DS}). The source and the substrate terminals of the nMOS transistor, also called the driver transistor, are connected to ground potential; hence,

the source-to-substrate voltage is $V_{SB} = 0$. In this generalized representation, the load device is represented as a two-terminal circuit element with terminal current IL and terminal voltage $V_L(I_L)$.

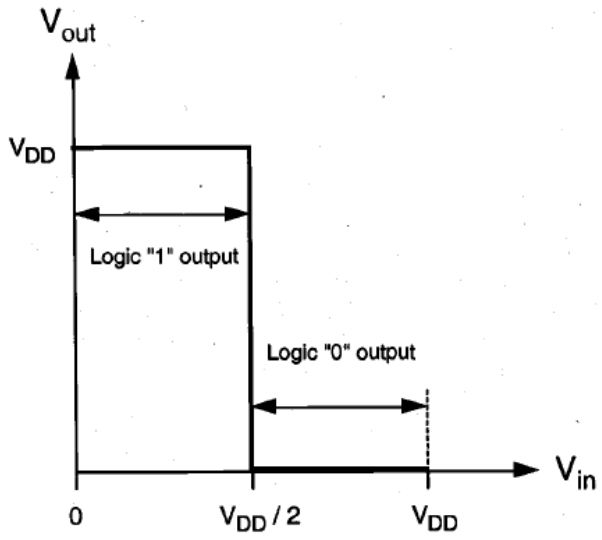


Fig 3 Voltage transfer characteristic (VTC) of the ideal inverter.

Voltage Transfer Characteristic (VTC*)*

Applying Kirchhoff s Current Law (KCL) to this simple circuit, we see that the load current is always equal to the nMOS drain current.

$$I_D(V_{in}, V_{out}) = I_L(V_L) \quad \text{.....(1)}$$

The voltage transfer characteristic describing ***V*** as a function of Vin under DC conditions can then be found by analytically solving equation (1) for various input voltage values. The typical VTC of a realistic nMOS inverter is shown in Fig. Upon examination, we can identify a number of important properties of this DC transfer characteristic.

The general shape of the VTC in Fig. 5.4 is qualitatively similar to that of the ideal inverter transfer characteristic shown in Fig. 5.2. There are, however, several significant differences that deserve special attention. For very low input voltage levels, the output voltage V is equal to the high value of *VOH (output high voltage).* In this case, the driver nMOS transistor is in cut-off, and hence, does not conduct any current. Consequently, the voltage drop across the load device is very small in magnitude, and the output voltage level is high. As the input voltage V increases, the driver transistor starts conducting a certain drain current, and the output voltage eventually starts to decrease. Notice that this drop in the

output voltage level does not occur abruptly, such as the vertical drop assumed for the ideal inverter VTC, but rather gradually and with a finite slope.

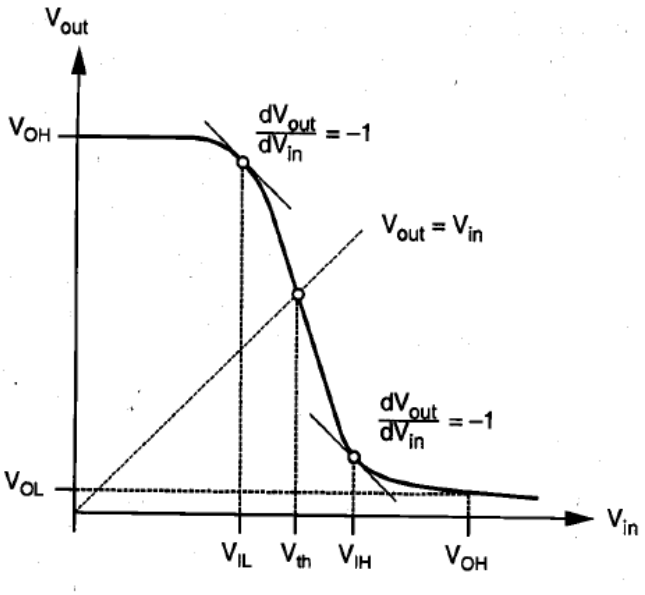


*Figure **5.4.*** Typical voltage transfer characteristic (VTC) of a realistic nMOS inverter.

We identify two *critical voltage points* on this curve, where the slope of the Vt(Vin) characteristic becomes equal to -1, i.e.,

$$\frac{dV_{out}}{dV_{in}} = -1$$

VOH: Maximum output voltage when the output level is logic " 1"

VOL Minimum output voltage when the output level is logic "0"

VIL: Maximum input voltage which can be interpreted as logic "0"

VIH: Minimum input voltage which can be interpreted as logic " 1"

5.2 Describe inverters with resistive load and with n-type & MOSFET load

Resistive-Load Inverter

The basic structure of the resistive-load inverter circuit is shown in Fig. 1. As in the general inverter circuit as shown in Fig. 2, an enhancement-type nMOS transistor acts as the driver device. The load consists of a simple linear resistor, RL. The power supply voltage of this circuit is VDD. Since the following analysis concentrates on the static behavior of the circuit, the output load capacitance is not shown in this figure.

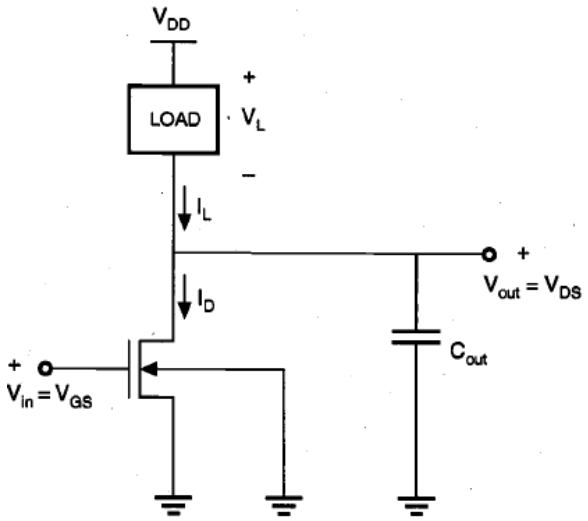


Fig .1

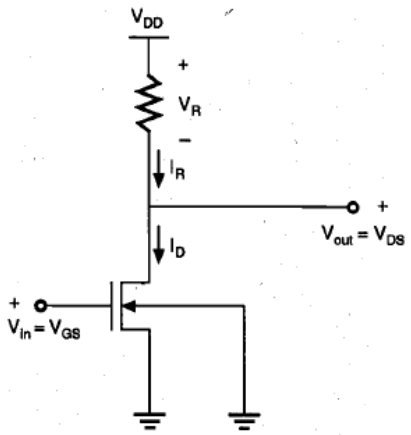


Fig .2

As already noted in Section equation 1, the drain current ID of the driver MOSFET is equal to the load current R in DC steady-state operation. To simplify the calculations, the channel-length modulation effect will be neglected in the following, i.e., A = 0. Also, note that the source andthe substrate terminals of the driver transistor are both connected to the ground; hence, VSB = 0. Consequently, the threshold voltage of the driver transistor is always equal to V.. We start our analysis by identifying the various operating regions of the driver transistor under steady-state conditions. For input voltages smaller than the threshold voltage V., the transistor is in cut-off, and does not conduct any drain current. Since the voltage drop across the load resistor is equal to zero, the output voltage must be equal to the power supply voltage, VDD. As the input voltage is increased beyond V_{th}., the driver transistor starts conducting a nonzero drain current. Note that the driver MOSFET is initially in saturation, since its drain-tosource voltage. (VDs = V_{out}) is larger than ($V_{in} - V_{GS}$.). Thus,

$$I_R = \frac{k_n}{2} \cdot (V_{in} - V_{T0})^2$$

Inverters with n-Type MOSFET Load

The simple resistive-load inverter circuit examined in the previous section is not a suitable candidate for most digital VLSI system applications, primarily because of the large area occupied by the load resistor. In this section, we will introduce inverter circuits, which use an nMOS transistor as the *active load* device, instead of the linear load resistor.The main advantage of using a MOSFET as the load device is that the silicon area occupied by the

transistor is usually smaller than that occupied by a comparable resistive load. Moreover, inverter circuits with active loads can be designed to have better overall performance compared to that of passive-load inverters. In a chronological view, the development of inverters with an enhancement-type MOSFET load precedes other active-load inverter types, since its fabrication process was perfected earlier

Enhancement-Load nMOS Inverter

The circuit configurations of two inverters with enhancement-type load devices are shown in Fig. 3 and 4. Depending on the bias voltage applied to its gate terminal, the load transistor can be operated either in the saturation region or in the linear region. Both types of inverters have some distinct advantages and disadvantages from the circuit design point of view.

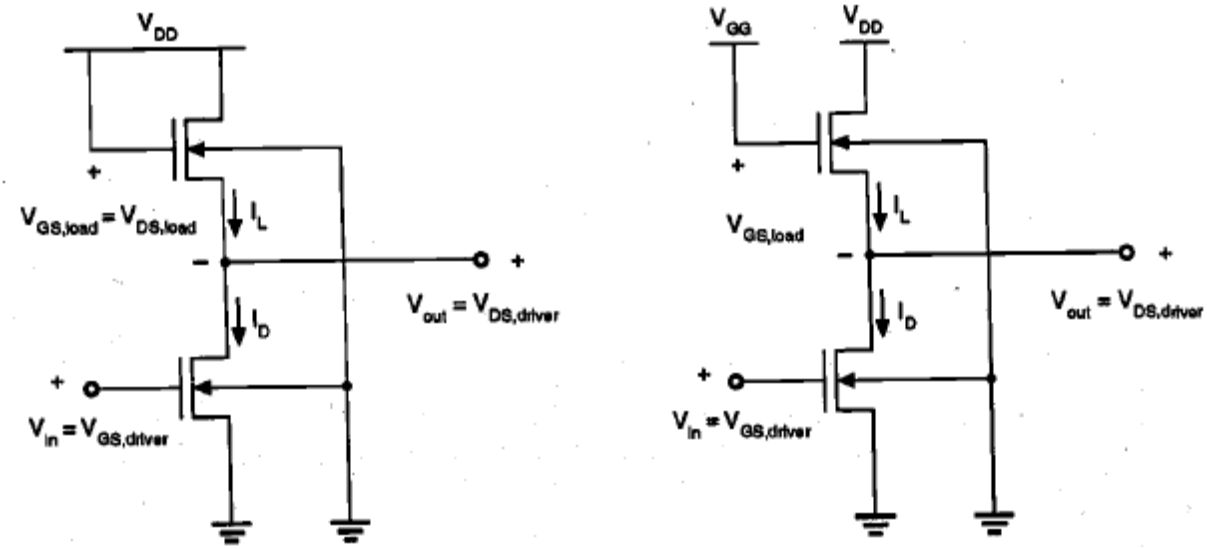


Fig 3 and 4

Depletion-Load nMOS Inverter

Several of the disadvantages of the enhancement-type load inverter can be avoided by using a depletion type nMOS transistor as the load device.-The fabrication process for producing an inverter with an enhancement-type nMOS driver and a depletion-type nMOS load is slightly more complicated and requires additional processing steps, especially for the channel implant to adjust the threshold voltage of the load device.

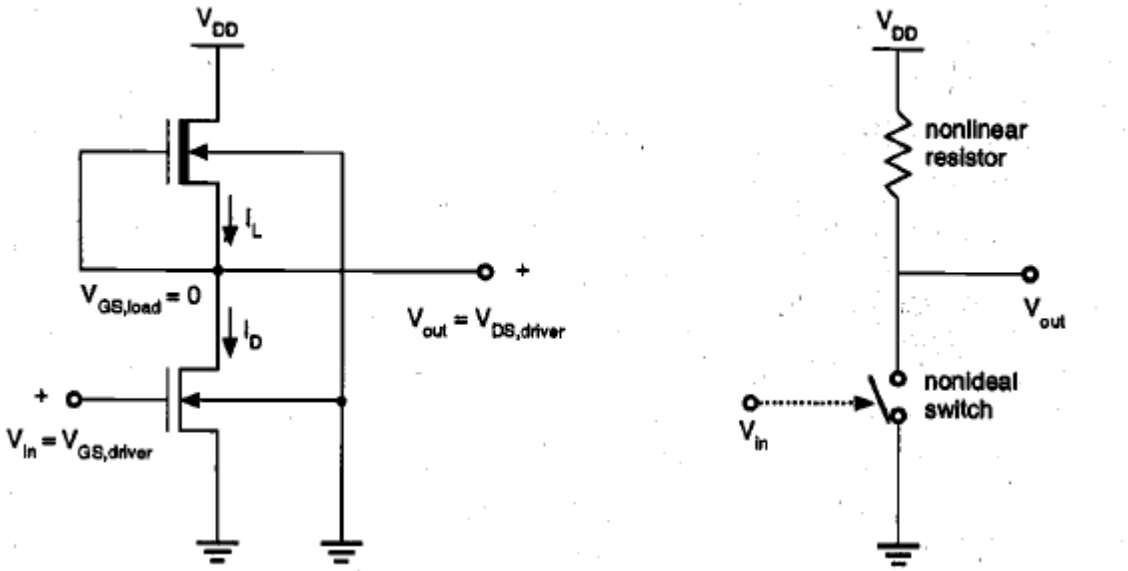


Fig 5 and 6

5.3 CMOS inverter and characteristics and interconnect effects: Delay time definitions

C – mos which consists of an enhancement-type nMOS transistor and an enhancement-type pMOS transistor, operating in complementary mode (Fig.7 and 8). This configuration is called Complementary MOS (CMOS). The circuit topology is complementary push-pull in the sense that for high input, the nMOS transistor drives (pulls down) the output node while the pMOS transistor acts as the load, and for low input the pMOS transistor drives (pulls up) the output node while the nMOS transistor acts as the load. Consequently, both devices contribute equally to the circuit operation characteristics

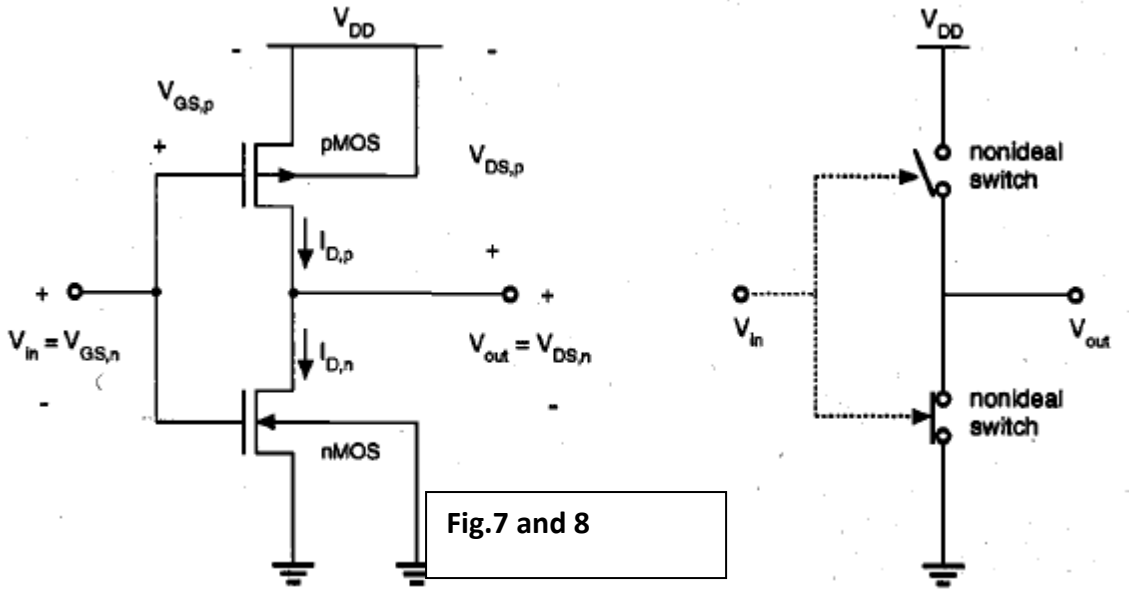


Fig.7 and 8

(7) CMOS inverter circuit. (8) Simplified view of the CMOS inverter, consisting of two complementary nonideal switches.

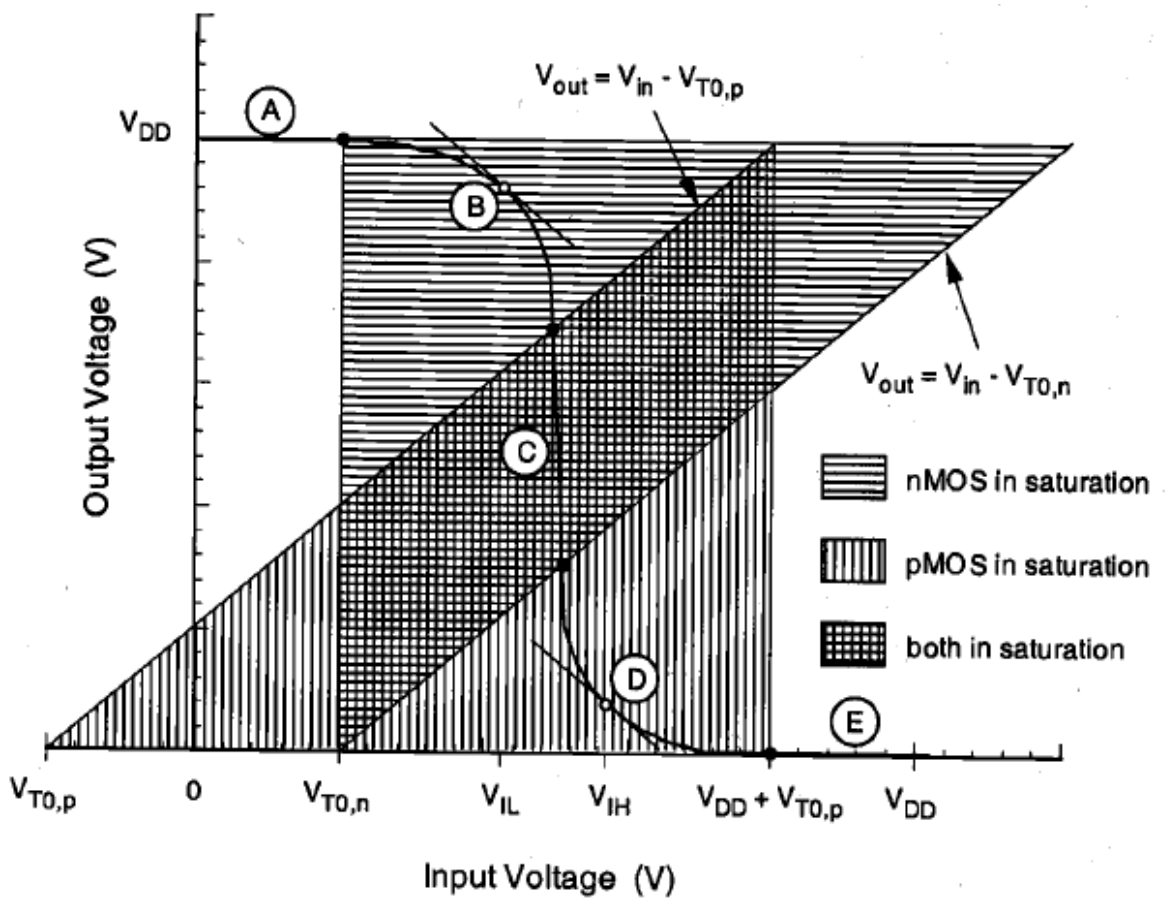


Fig -9 **Operating regions of the nMOS and the pMOS transistors.**

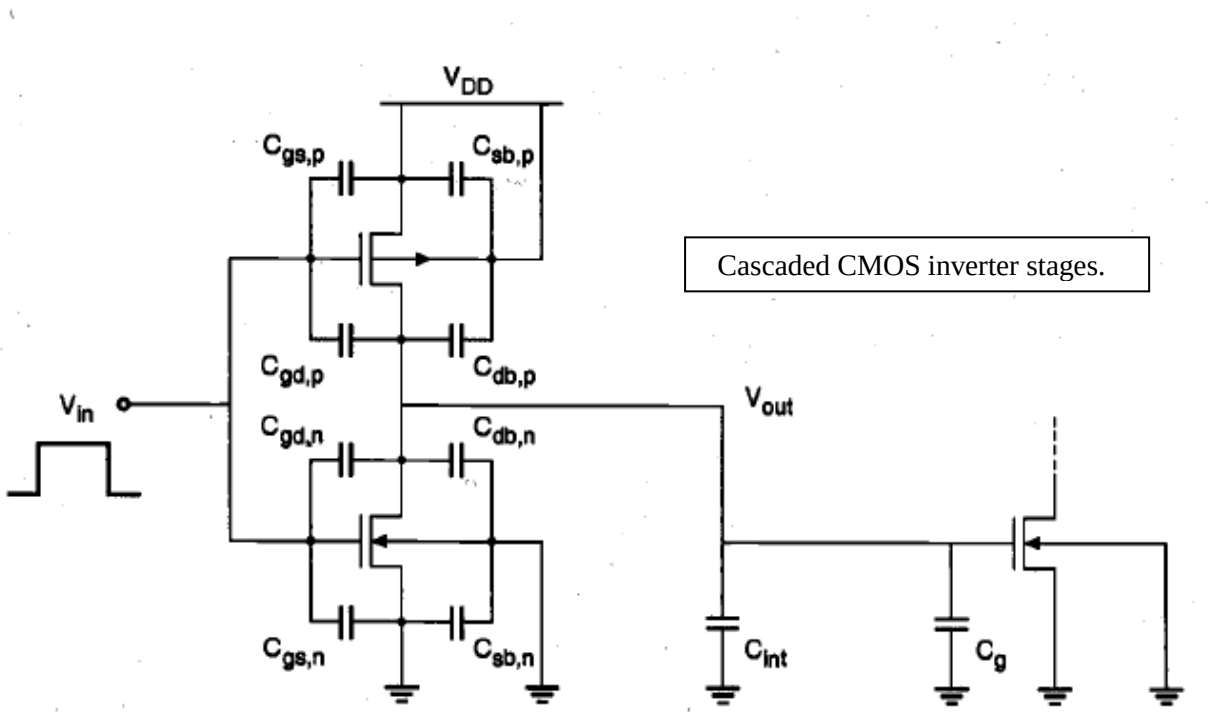
Both of these conditions for device saturation are illustrated graphically as shaded areas on the Vu -V plane in Fig. 9 A typical CMOS inverter voltage transfer characteristic is also superimposed for easy reference. Here, we identify five distinct regions, labeled A through E, each corresponding to a different set of operating conditions. The table below lists these regions and the corresponding critical input and output voltage levels.

Region	V_{in}	V_{out}	nMOS	pMOS
A	$< V_{T0,n}$	V_{OH}	cut-off	linear
B	V_{IL}	high $\approx V_{OH}$	saturation	linear
C	V_{th}	V_{th}	saturation	saturation
D	V_{IH}	low $\approx V_{OL}$	linear	saturation
E	$>\left(V_{DD} + V_{T0,p}\right)$	V_{OL}	linear	cut-off

Consider the cascade connection of two CMOS inverter circuits shown in Fig. 6.1. The parasitic capacitances associated with each MOSFET are illustrated individually. Here, the capacitances C_{gd} and C_{gs} are primarily due to gate overlap with diffusion, while C_{db} and C_{sb} are voltage-dependent junction capacitances, as discussed in Chapter 3. The 'capacitance component C_g is due to the thin-oxide capacitance over the gate area. In addition, we also consider the lumped interconnect capacitance C_{int}, which represents the parasitic capacitance contribution of the metal or polysilicon connection between the two inverters. It is assumed that a pulse waveform is applied to the input of the first-stage inverter.

The problem of analyzing the output voltage waveform is fairly complicated, even for this relatively simple circuit, because a number of nonlinear, voltage-dependent capacitances are involved. To simplify the problem, we first combine the capacitances seen in Fig. into an equivalent *lumped* linear capacitance, connected between the output node of the inverter and the ground. This combined capacitance at the output node will be called the load capacitance, $\mathbf{C}_{load}$

$$C_{load} = C_{gd,n} + C_{gd,p} + C_{db,n} + C_{db,p} + C_{int} + C_g$$



Cascaded CMOS inverter stages.

Fig - 10

The propagation delay times τ_{PHL} and τ_{PLH} determine the input-to-output signal delay during the high to-low and low-to-high transitions of the output, respectively. By definition, τ_{PHL} is the time delay between the V50%-transition of the *rising* input voltage and the V50 -

transition of *the falling* output voltage. Similarly, τ_{PLH} is defined as the time delay between the V50 -transition of *the falling* input voltage and the V50%-transition of the *rising* output voltage.

$$V_{50\%} = V_{OL} + \frac{1}{2}(V_{OH} - V_{OL}) = \frac{1}{2}(V_{OL} + V_{OH})$$

Thus, the propagation delay times τ_{PHL} and τ_{PLH} are found from Fig. 11 as

$$\tau_{PHL} = t_1 - t_0$$
$$\tau_{PLH} = t_3 - t_2$$

The average propagation delay τ_P of the inverter characterizes the average time required for the input signal to propagate through the inverter.

$$\tau_P = \frac{\tau_{PHL} + \tau_{PLH}}{2}$$

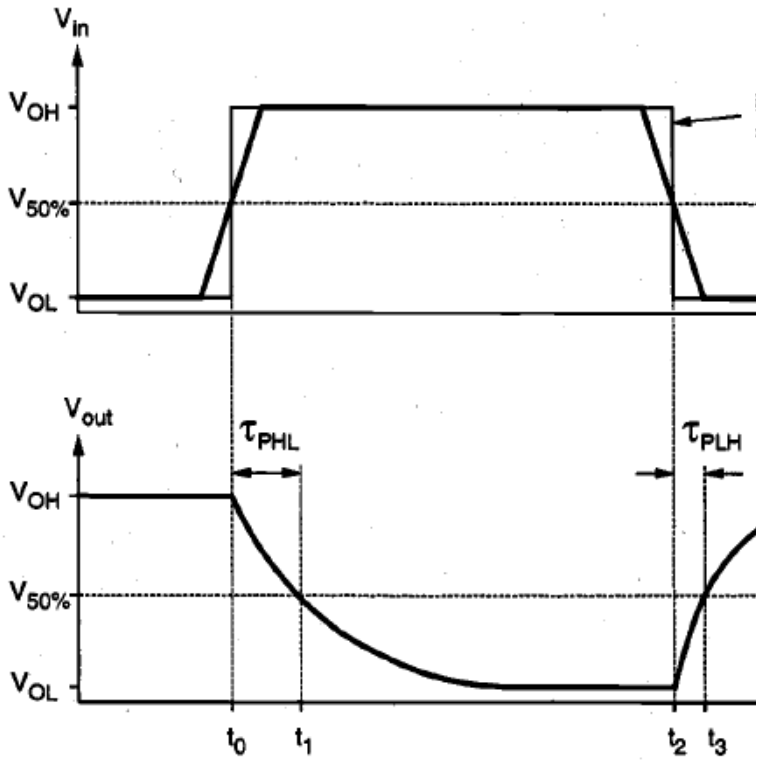


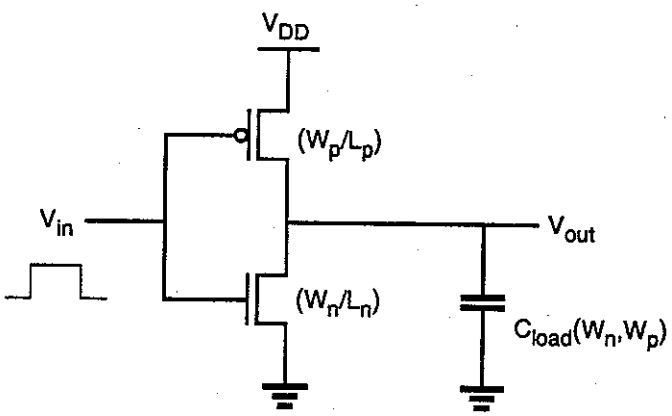
FIG 11(Input and output voltage waveforms of a typical inverter, and the definitions of propagation delay times. The input voltage waveform is idealized as a step pulse for simplicity.)

5.4 Invertor design with delay constraints.

The propagation delay equations on chart 4-5 can be rearranged to solve for W/L, as shown below, where we substituted $C_{ox}\mu_n(W_n/L_n)$ for k_n and similarly for k_p These

equations can then be used to "size" a CMOS circuit to achieve a desired minimum rising or falling propagation delay assuming C_{load} and other parameters are known

After determining the desired W/L values, we can obtain the device widths W based on the technology minimum design device lengths L . Other constraints such as rise time/fall time or rise/fall symmetry may also need to be considered in addition to rise and fall delay.



$$\left(\frac{W_n}{L_n}\right)=\frac{C_{load}}{\tau^*_{PHL}\,\mu_n\,C_{ox}\left(V_{DD}-V_{T,n}\right)}\left[\frac{2\,V_{T,n}}{V_{DD}-V_{T,n}}+\ln\left(\frac{4\left(V_{DD}-V_{T,n}\right)}{V_{DD}}-1\right)\right]$$

$$\left(\frac{W_p}{L_p}\right)=\frac{C_{load}}{\tau^*_{PLH}\,\mu_p\,C_{ox}\left(V_{DD}-\left|V_{T,p}\right|\right)}\left[\frac{2\left|V_{T,p}\right|}{V_{DD}-\left|V_{T,p}\right|}+\ln\left(\frac{4\left(V_{DD}-\left|V_{T,p}\right|\right)}{V_{DD}}-1\right)\right]$$

5.5 Estimation of parasitics switching power dissipation of CMOS inverters.

equals the instantaneous drain current of the pMOS transistor. When the input voltage switches from low to high, the Pmos transistor in the circuit is turned off, and the nMOS transistor starts conducting. During this phase, the output load capacitance C_L is being discharged through the nMOS transistor. Thus, the capacitor current equals the instantaneous drain current of the nMOS transistor. When the input voltage switches from high to low, the nMOS transistor in the circuit is turned off, and the pMOS transistor starts conducting. During this phase, the output load capacitance C_L ,ad is being charged up through the pMOS transistor; therefore, the capacitor current

- For complementary CMOS circuits where no dc current flows, average dynamic power is given by

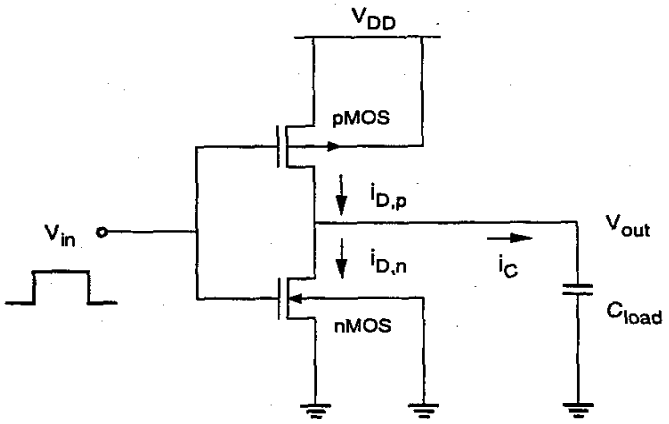
$$P_{ave} = C_L V_{DD}^2 f$$

where CL represents the total load capacitance, VDD is the power supply, and f is the frequency of the signal transition

– above formula applies to a simple CMOS inverter or to complex, combinational CMOS logic

– applies only to dynamic (capacitive) power

– dc power and/or short-circuit power must be computed separately



Chapter 6

Combinational, Sequential & Dynamics logic circuits

6.1 MOS logic circuits & CMOS logic circuits. state style, complex logic circuits, pass transistor logic.

In its most general form, a combinational logic circuit, or gate, performing a Boolean function can be represented as a multiple-input single-output system, as depicted in Fig. 6.1. All input variables are represented by node voltages, referenced to the ground potential. Using *positive logic convention*, the Boolean (or logic) value of " 1 " can be represented by a high voltage of *VDD*, and the Boolean (or logic) value of "0" can be represented by a low voltage of 0. The output node is loaded with a capacitance *CL*, which represents the combined parasitic device capacitances in the circuit and the interconnect capacitance components *seen* by the output node. This output load capacitance certainly plays a very significant role in the dynamic operation of the logic gate.

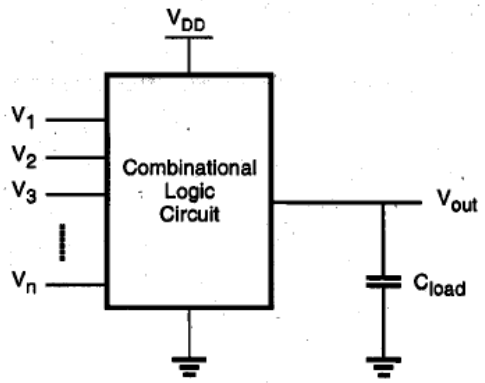
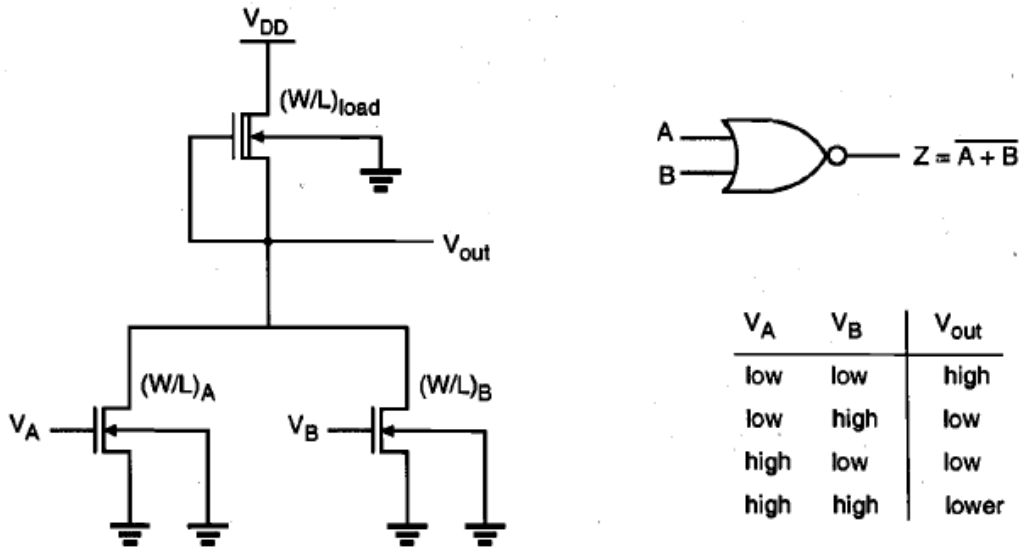


Fig. 6.1.

MOS Logic Circuits with Depletion nMOS Loads

Two-Input NOR Gate

The first circuit to be examined in this section is the two-input NOR gate. The circuit diagram, the logic symbol, and the corresponding truth table of the gate are given in Fig. 6.2. The Boolean OR operation is performed by the parallel connection of the two enhancement-type nMOS driver transistors.

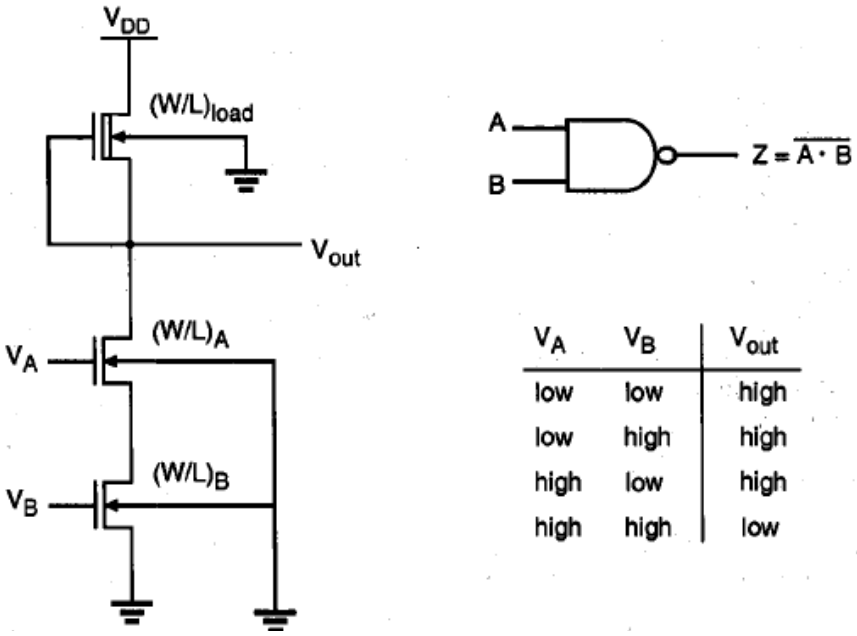


V_A	V_B	V_{out}
low	low	high
low	high	low
high	low	low
high	high	lower

Fig. 6.2.

Two-Input NAND Gate

Next, we will examine the two-input NAND (NAND2) gate. The circuit diagram, the logic symbol, and the corresponding truth table of the gate are given in Fig. 6.3 The Boolean AND operation is performed by the series connection of the two enhancement type nMOS driver transistors. There is a conducting path between the output node and the ground only if the input voltage V_A *and* the input voltage V_B are equal to logic-high, i.e., only if both of the series-connected drivers are turned on. In this case, the output voltage will be low, which is the complemented result of the AND operation. Otherwise, either one or both of the driver transistors will be off, and the output voltage will be pulled to a logic-high level by the depletion-type nMOS load transistor.



V_A	V_B	V_{out}
low	low	high
low	high	high
high	low	high
high	high	low

Fig. 6.3

CMOS Logic Circuits

CMOS NOR2 (Two-Input NOR) Gate

The design and analysis of CMOS combinational logic circuits can be based on the basic principles developed for the nMOS depletion-load logic circuits in the previous section. Figure 6.4 shows the circuit diagram of a two-input CMOS NOR gate. Note that the circuit consists of a parallel-connected n-not and a series-connected complementary p not. The input voltages *VA* and *VB* are applied to the gates of one nMOS and one Pmos transistor

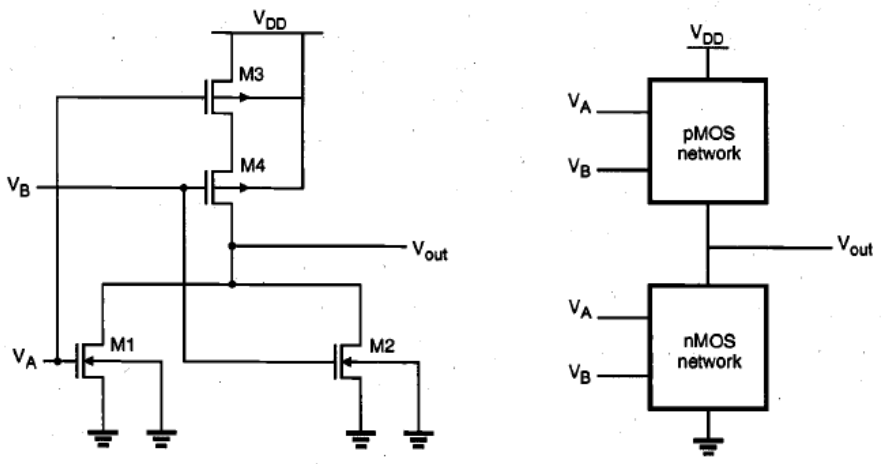


Figure 6.4

Complex Logic Circuits

To realize arbitrary Boolean functions of multiple input variables, the basic circuit structures and design principles developed for simple NOR and NAND gates in the

previous sections can easily be extended to complex logic gates. The ability to realize complex logic functions using a small number of transistors is one of the most attractive features of nMOS and CMOS logic circuits.

Consider the following Boolean function as an example.

$$Z = \overline{A(D+E)+BC}$$

The nMOS depletion-load complex logic gate that is used to realize this function is shown in Fig. 6.5 Inspection of the circuit topology reveals the simple design principle of the pull-down network:

* OR operations are performed by parallel-connected drivers.
* AND operations are performed by series-connected drivers.
* Inversion is provided by the nature of MOS circuit operation.

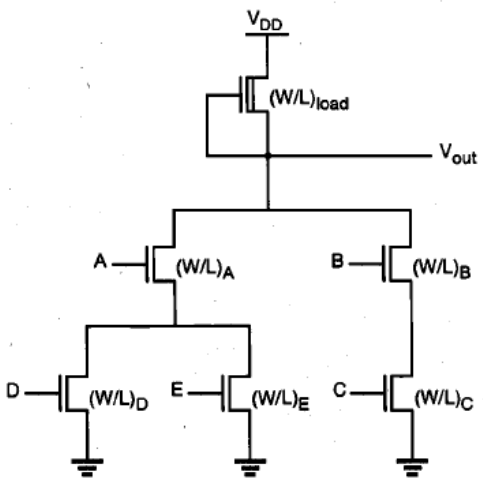


Fig. 6.5 (nMOS complex logic gate realizing the Boolean function)

The CMOS transmission gate (TG) or pass gate, a new class of logic circuits which use the TGs as their basic building blocks. As shown in Fig 6.6 the CMOS transmission gate consists of one VnMOS and one pMOS transistor, connected in parallel. The gate voltages applied to these two transistors are also set to be complementary signals. As such, the CMOS TG operates as a bidirectional switch between the nodes A and B which is controlled by signal C.

If the control signal C is logic-high, i.e., equal to *VDD,* then both transistors are turned on and provide a low-resistance current path between the nodes A and B. If, on the other hand, the control signal C is low, then both transistors will be off, and the path between the nodes A and B will be an open circuit. This condition is also called the high-impedance

state. Note that the substrate terminal of the nMOS transistor is connected to ground and the substrate terminal of the pMOS transistor is connected to ***VDD.*** Thus, we must take into account the substrate-bias effect for both transistors, depending on the bias conditions. Figure 7.33 also shows three other commonly used symbolic representations of the CMOS transmission gate.

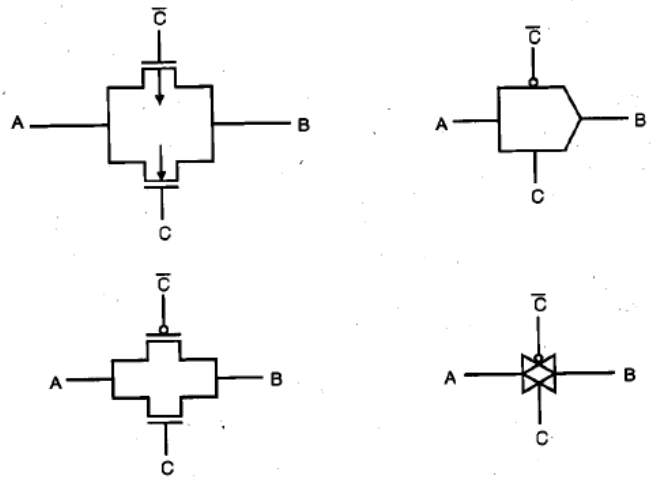


Fig 6.6 Four different representations of the CMOS transmission gate (TG).

6.2 Explain SR latch, clocked latch & flip-flop circuits.

The SR Latch Circuit

The bistable element consisting of two cross-coupled inverters (Fig. 8.2) has two stable operating modes, or states. The circuit preserves its state (either one of the two possible modes) as long as the power supply voltage is provided; hence, the circuit can perform a simple memory function of *holding* its state. However, the simple two-inverter circuit examined above has no provision for allowing its state to be changed externally from one stable operating mode to the other. To allow such a change of state, we must add simple switches to the bistable element, which can be used to force or trigger the circuit from one operating point to the other. Figure 6.7 shows the circuit structure of the simple CMOS SR latch, which has two such triggering inputs, S (set) and R (reset). In the literature, the SR latch is also called an SR flip-flop, since two stable states can be switched back and forth. The circuit consists of two CMOS NOR2 gates. One of the input terminals of each NOR gate is used to cross-couple to the output of the other NOR gate, while the second input enables triggering of the circuit.

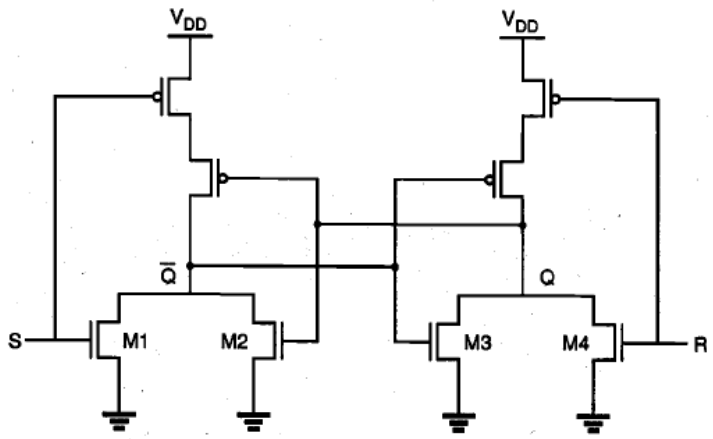


Fig 6.7

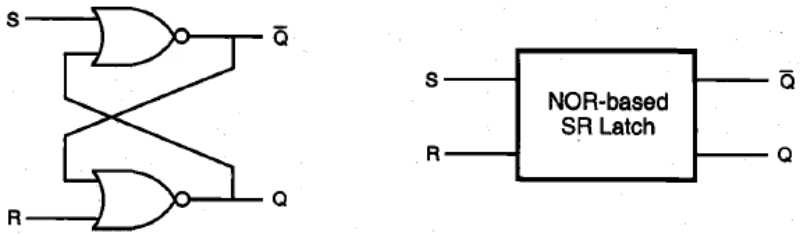


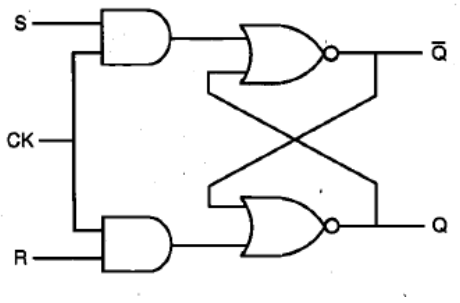
Fig 6.8 (Gate-level schematic and block diagram of the NOR-based SR latch.)

The truth table of the NOR-based SR latch is summarized in the following table :

S	R	Q_{n+1}	$\overline{Q_{n+1}}$	Operation
0	0	Q_n	$\overline{Q_n}$	hold
1	0	1	0	set
0	1	0	1	reset
1	1	0	0	not allowed

Clocked SR Latch

All of the SR latch circuits examined in the previous section are essentially asynchronous sequential circuits, which will respond to the changes occurring in input signals at a circuit-delay-dependent time point during their operation. To facilitate synchronous operation, the circuit response can be controlled simply by adding a gating clock signal to the circuit, so that the outputs will respond to the input levels only during the active period of a clock pulse. For simple reference, the clock pulse will be assumed to be a periodic square waveform, which is applied simultaneously to all clocked logic gates in the system.



6.9 (Gate-level schematic of the clocked NOR-based SR latch).

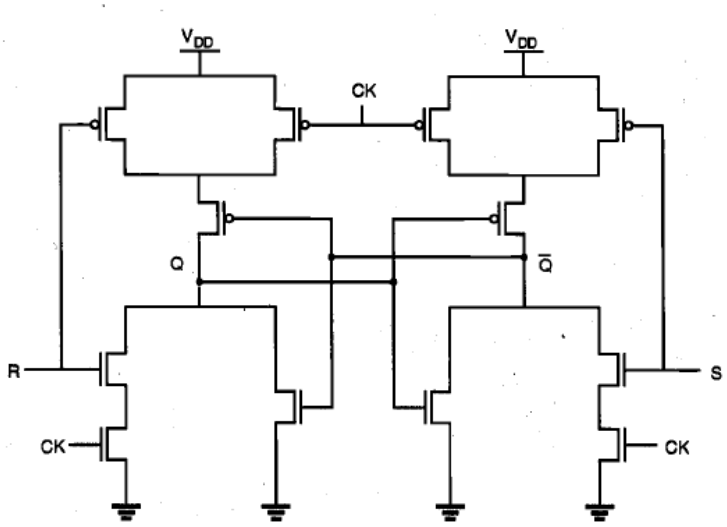


Fig 6.10(AOI-based implementation of the clocked NOR-based SR latch circuit.)

6.3 Explain Dynamic logic & basic principles.

In high-density, high-performance digital implementations where reduction of circuit delay and silicon area is a major objective, *dynamic logic circuits* offer several significant advantages over static logic circuits. The operation of all dynamic logic gates depends on temporary (transient) storage of charge in parasitic node capacitances, instead of relying on steady-state circuit behavior. This operational property necessitates periodic updating of internal node voltage levels, since stored charge in a capacitor cannot be retained indefinitely.

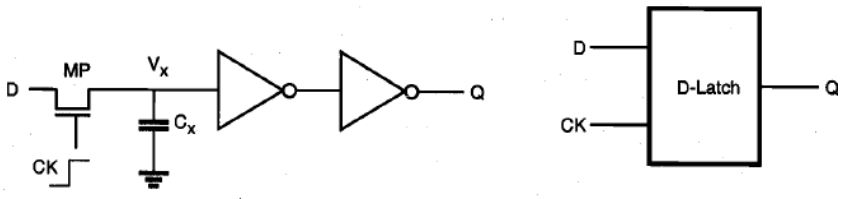


Fig 6.11(Dynamic latch circuit.)

* When the clock is high (CK = 1), the pass transistor turns on. The capacitor *C*, is either charged up, or charged down through the pass transistor MP, depending on the input (D) voltage level. The output (Q) assumes the same logic level as the input.

* When the clock is low (CK = 0), the pass transistor MP turns off, and the capacitor C is isolated from the input D. Since there is no current path from the intermediate node X to either *VDD* or ground, the amount of charge stored in C. during the previous cycle determines the output voltage level Q.

6.4 High performance dynamics CMOS circuits.

The circuits presented here are variants of the basic dynamic CMOS logic gate structure.

They are designed to take full advantage of the obvious benefits of dynamic operation and at the same time, to allow unrestricted cascading of multiple stages. The ultimate goal is to achieve reliable, high-speed, compact circuits using the least complicated clocking scheme possible.

Domino CMOS Logic

Consider the generalized circuit diagram of a domino CMOS logic gate shown in Fig. 9.28. A dynamic CMOS logic stage, such as the one shown in Fig. 9.26, is cascaded with a static CMOS inverter stage. The addition of the inverter allows us to operate a number of such structures in cascade, as explained in the following.

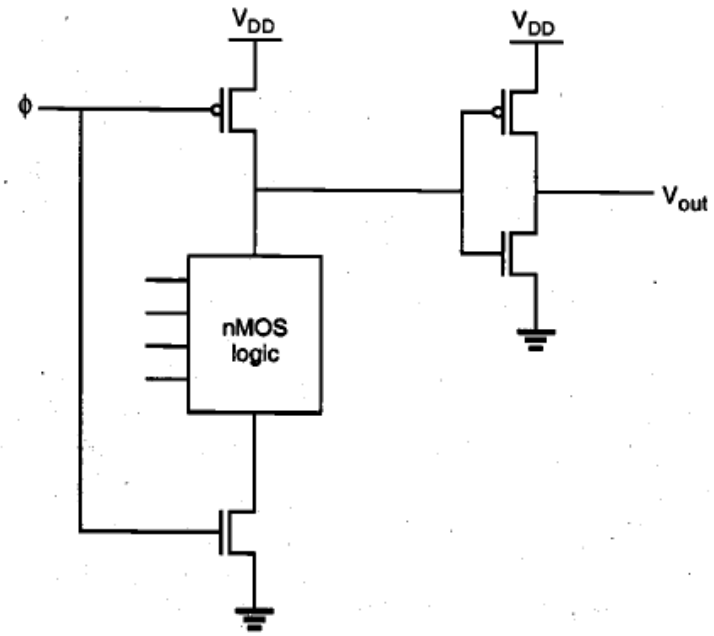


Figure 6.12. Generalized circuit diagram of a domino CMOS logic gate.

During the precharge phase (when CK = 0), the output node of the dynamic CMOS stage is precharged to a high logic level, and the output of the CMOS inverter (buffer) becomes low. When the clock signal rises at the beginning of the evaluation phase, there are two possibilities: The output node of the dynamic CMOS stage is either discharged to a low level through the nMOS circuitry (1 to 0 transition), or it remains high.

Consequently, the inverter output voltage can also make at most one transition during the evaluation phase, from 0 to 1. Regardless of the input voltages applied to the dynamic CMOS stage, it is not possible for the buffer output to make a 1 to 0 transition during the evaluation phase.

6.5 Define Dynamic Ram, SRAM, flash memory.

Read-write (R/W) memory circuits, on the other hand, must permit the modification (writing) of data bits stored in the memory array, as well as their retrieval (reading) on demand. This requires that the data storage function be *volatile,* i.e., the stored data are lost when the power supply voltage is turned off. The read-write memory circuit is commonly called *Random Access Memory* (RAM), mostly due to historical reasons.

Compared to sequential-access memories such as magnetic tapes, any cell in the R/W memory array can be accessed with nearly equal access time. Based on the operation type of individual data storage cells, RAMs are classified into two main categories: *Static S*RAMs SRAM) and *Dynamic* RAMs (DRAM).

Chapter 7 System Design method & Testing

7.1 **Design capture tools, hardware definition languages such as VHDL and packages. Xlinx (introduction)**

Hardware description language (HDL): allows designer to specify logic function only. Then a computer-aided design (CAD) tool produces or *synthesizes* the optimized gates. Most commercial designs built using HDLs Two leading HDLs:

– **Verilog**

• developed in 1984 by Gateway Design Automation

• became an IEEE standard (1364) in 1995

– VHDL

VHDL was originally developed at the behest of the U.S Department of Defense in order to document the behavior of the ASICs that supplier companies were including in equipment.

The idea of being able to simulate the ASICs from the information in this documentation was so obviously attractive that logic simulators were developed that could read the VHDL files. The next step was the development of logic synthesis tools that read the VHDL, and output a definition of the physical implementation of the circuit.

The initial version of VHDL, designed to IEEE standard 1076-1987, included a wide range of data types, including numerical (integer and real), logical (bit and boolean), character and time, plus arrays of bit called bit_vector and of character called string

Standardization

The IEEE Standard 1076 defines the VHSIC Hardware Description Language or VHDL. It was originally developed under contract F33615-83-C-1003 from the United States Air Force awarded in 1983 to a team with Intermetrics, Inc. as language experts and prime contractor, with Texas Instruments as chip design experts and IBM as computer system design experts. The language has undergone numerous revisions and has a variety of sub-standards associated with it that augment or extend it in important ways.

Design

VHDL is commonly used to write text models that describe a logic circuit. Such a model is processed by a synthesis program, only if it is part of the logic design. A simulation program is used to test the logic design using simulation models to represent the logic circuits that interface to the design. This collection of simulation models is commonly called a *testbench*.

VHDL has constructs to handle the parallelism inherent in hardware designs, but these constructs (processes) differ in syntax from the parallel constructs in Ada (tasks). Like Ada, VHDL is strongly typed and is not case sensitive. In order to directly represent operations which are common in hardware, there are many features of VHDL which are not found in Ada, such as an extended set of Boolean operators including nand and nor. VHDL also allows arrays to be indexed in either ascending or descending direction; both conventions are used in hardware, whereas in Ada and most programming languages only ascending indexing is available.

Advantages

The key advantage of VHDL, when used for systems design, is that it allows the behavior of the required system to be described (modeled) and verified (simulated) before synthesis tools translate the design into real hardware (gates and wires).

Another benefit is that VHDL allows the description of a concurrent system. VHDL is a dataflow language, unlike procedural computing languages such as BASIC, C, and assembly code, which all run sequentially, one instruction at a time.

A VHDL project is multipurpose. Being created once, a calculation block can be used in many other projects. However, many formational and functional block parameters can be tuned (capacity parameters, memory size, element base, block composition and interconnection structure).

Xilinx

Xilinx ISE[1] (Integrated Software Environment) is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer.

Technology

The Spartan-3 platform was the industry's first 90nm FPGA, delivering more functionality and bandwidth per dollar than was previously possible, setting new standards in the programmable logic industry.

Xilinx designs, develops and markets programmable logic products, including integrated circuits (ICs), software design tools, predefined system functions delivered as intellectual property (IP) cores, design services, customer training, field engineering and technical support. Xilinx sells both FPGAs and CPLDs for electronic equipment manufacturers in end markets such as communications, industrial, consumer, automotive and data processing.

Xilinx's FPGAs have been used for the ALICE (A Large Ion Collider Experiment) at the CERN European laboratory on the French-Swiss border to map and disentangle the trajectories of thousands of subatomic particles. Xilinx has also engaged in a partnership with the United States Air Force Research Laboratory's Space Vehicles Directorate to develop FPGAs to withstand the damaging effects of radiation in space, which are 1,000 times less sensitive to space radiation than the commercial equivalent, for deployment in new satellites.

The Virtex-II Pro, Virtex-4, Virtex-5, and Virtex-6 FPGA families, which include up to two embedded IBM PowerPC cores, are targeted to the needs of system-on-chip (SoC) designers.

Xilinx FPGAs can run a regular embedded OS (such as Linux or vxWorks) and can implement processor peripherals in programmable logic.

Xilinx's IP cores include IP for simple functions (BCD encoders, counters, etc.), for domain specific cores (digital signal processing, FFT and FIR cores) to complex systems (multi-gigabit networking cores, the Micro Blaze soft microprocessor and the compact Picoblaze microcontroller). Xilinx also creates custom cores for a fee.

The main design toolkit Xilinx provides engineers is the Vivado Design Suite, an integrated design environment (IDE) with a system-to-IC level tools built on a shared scalable data model and a common debug environment. Vivado includes electronic system level (ESL) design tools for synthesizing and verifying C-based algorithmic IP; standards based packaging of both algorithmic and RTL IP for reuse; standards based IP stitching and systems integration of all types of system building blocks; and the verification of blocks and systems. A free version WebPACK Edition of Vivado provides designers with a limited version of the design environment.

Xilinx's Embedded Developer's Kit (EDK) supports the embedded PowerPC 405 and 440 cores (in Virtex-II Pro and some Virtex-4 and -5 chips) and the Microblaze core. Xilinx's System Generator for DSP implements DSP designs on Xilinx FPGAs. A freeware version of its EDA software called ISE WebPACK is used with some of its non-high-performance chips. Xilinx is the only (as of 2007) FPGA vendor to distribute a native Linux freeware synthesis tool chain

7.3 Introduction to IRSIM and GOSPL (open source packages).

IRSIM

IRSIM is a tool for simulating digital circuits. It is a "switch-level" simulator; that is, it treats transistors as ideal switches. Extracted capacitance and lumped resistance values are used to make the switch a little bit more realistic than the ideal, using the RC time constants to predict the relative timing of events.

IRSIM shares a history with magic, although it is an independent program. Magic was designed to produce, and IRSIM to read, the ".sim" file format, which is largely unused outside of these two programs. IRSIM was developed at Stanford, while Magic was developed at Berkeley. Parts of Magic were developed especially for use with IRSIM, allowing IRSIM to run a simulation in the "background" (i.e., a forked process communicating through a pipe), while displaying information about the values of signals directly on the VLSI layout.

For "quick" simulations of digital circuits, IRSIM is still quite useful for confirming basic operation of digital circuit layouts. The addition of scheduling commands ("at", "every", "when", and "whenever") put IRSIM into the same class as Verilog simulators. It is, in my opinion, much easier to write complicated testbench simulations using Tcl and IRSIM. I have used IRSIM to validate the digital parts of several production chips at MultiGiG, including the simulation of analog behavior such as PLL locking.

IRSIM version 9.5 was a long-standing and stable version that corresponded to the relatively stable Magic version 6.5. When magic was recast in a Tcl/Tk interpreter framework (versions 7.2 and 7.3), IRSIM could no longer operate as a background process. However, it was clear that if IRSIM could also be recast in the same Tcl/Tk interpreter framework, the level of interaction between it and Magic would be greatly increased.

7.4 Design verification and testing, simulation at various levels including timing verification, faults models.

Design verification

Design verification is the most important aspect of the product development process illustrated in Figures , consuming as much as 80% of the total product development time. The intent is to verify that the design meets the system requirements and specifications. Approaches to design verification consist of (1) logic simulation/emulation and circuit simulation, in which detailed functionality and timing of the design are checked by means of simulation or emulation; (2) functional verification, in which functional models describing the functionality of the design are developed to check against the behavioral specification of the design without detailed timing simulation; and (3) formal verification, in which the functionality is checked against a “golden” model. Formal verification further includes

property checking (or model checking), in which the property of the design is checked against some presumed "properties" specified in the functional or behavioral model (e.g., a finite-state machine should not enter a certain state), and equivalence checking, in which the functionality is checked against a "golden" model .

Simulation-based techniques are the most popular approach to verification, even though these are time-consuming and may be incomplete in finding design errors. Logic simulation is used throughout every stage of logic design automation, whereas circuit simulation is used after physical design. The most commonly used logic simulation techniques are compiled-code simulation and event-driven simulation .The former is most effective for cyclebased two-valued simulation; the latter is capable of handling various gate and wire delay models. Although versatile and low in cost, logic simulation is too slow for complex SOC designs or hardware/software co-simulation applications. For more accurate timing information and dynamic behavior analysis, devicelevel circuit simulation is used. However, limited by the computation complexity, circuit simulation is, in general, only applied to critical paths, cell library components, and memory analysis.

Emulation-based verification by use of FPGAs provides an attractive alternative to simulation-based verification as the gap between logic simulation capacity and design complexity continues growing. Before the introduction of FPGAs in the 1980s, ASICs were often verified by construction of a breadboard by use of small-scale integration (SSI) and medium-scale integration (MSI) devices on a wire-wrap board. This became impractical as the complexity and scale of ASICs moved into the VLSI realm. As a result, FPGAs became the primary hardware for emulation-based verification. Although these approaches are costly and may not be easy to use, they improve verification time by two to three orders of magnitude compared with software simulation.

Formal verification techniques are a relatively new paradigm for equivalence checking. Instead of input stimuli, these techniques perform exhaustive proof through rigorous logical reasoning. The primary approaches used for formal verification include binary decision diagrams (BDDs) and Boolean satisfiability (SAT). These approaches, along with other algorithms specific to EDA applications. The BDD approach successively applies Shannon expansion on all variables of a combinational logic function until either the constant function "0" or "1"is reached.

TEST AUTOMATION

Advances in manufacturing process technology have also led to very complex designs. As a result, it has become a requirement that design-for-testability (DFT) features be incorporated in the register-transfer level (RTL) or gatelevel design before physical design to ensure the quality of the fabricated devices. In fact, the traditional VLSI development process illustrated in Figure involves some form of testing at each stage, including design verification. Once verified, the VLSI design then goes to fabrication and, at the same time, test engineers develop a test procedure based on the design specification and fault models associated with the implementation technology. Because the resulting product quality is in general unsatisfactory, modern VLSI test development planning tends to start when the RTL design is near completion. This test development plan defines what test requirements the product must meet, often in terms of defect level and manufacturing yield, test cost, and whether it is necessary to perform self-test and diagnosis.

Fault models

A defect is a manufacturing flaw or physical imperfection that may lead to a fault, a fault can cause a circuit error, and a circuit error can result in a failure of the device or system. Because of the diversity of defects, it is difficult to generate tests for real defects. Fault models are necessary for generating and evaluating test patterns. Generally, a good fault model should satisfy two criteria: (1) it should accurately reflect the behavior of defects and (2) it should be computationally efficient in terms of time required for fault simulation and test generation. Many fault models have been proposed but, unfortunately, no single fault model accurately reflects the behavior of all possible defects that can occur. As a result, a combination of different fault models is often used in the generation and evaluation of test patterns. Some well-known and commonly used fault models for general sequential logic include the following:

1. Gate-level stuck-at fault model: The stuck-at fault is a logical fault model that has been used successfully for decades. A stuck-at fault transforms the correct value on the faulty signal line to appear to be stuck-at a constant logic value, either logic 0 or 1, referred to as stuck-at-0 (SA0) or stuck-at-1 (SA1), respectively. This model is commonly referred to as the line stuck-at fault model where any line can be SA0 or SA1, and also referred to as the gate-level stuck-at fault model where any input or output of any gate can be SA0 or SA1.

2. **Transistor-level stuck fault model**: At the switch level, a transistor can be stuck-off or stuck-on, also referred to as stuck-open or stuckshort, respectively. The line stuck-at fault model cannot accurately reflect the behavior of stuck-off and stuck-on transistor faults in complementary metal oxide semiconductor (CMOS) logic circuits because of the multiple transistors used to construct CMOS logic gates. A stuckopen transistor fault in a CMOS combinational logic gate can cause the gate to behave like a level-sensitive latch. Thus, a stuck-open fault in a CMOS combinational circuit requires a sequence of two vectors for 1.3 Test automation 19 detection instead of a single test vector for a stuck-at fault. Stuck-short faults, on the other hand, can produce a conducting path between power(VDD) and ground (VSS) and may be detected by monitoring the power supply current during steady state, referred to as IDDQ. This technique of monitoring the steady state power supply current to detect transistor stuck-short faults is called IDDQ testing.

3. Bridging fault models: Defects can also include opens and shorts in the wires that interconnect the transistors that form the circuit. Opens tend to behave like line stuck-at faults. However, a resistive open does not behave the same as a transistor or line stuck-at fault, but instead affects the propagation delay of the signal path. A short between two wires is commonly referred to as a bridging fault. The case of a wire being shorted to VDD or VSS is equivalent to the line stuck-at fault model. However, when two signal wires are shorted together, bridging fault models are needed; the three most commonly used bridging fault models

7.5 Design strategies for testing chip level and system level test techniques.

CHIP-LEVEL TEST TECHNIQUES

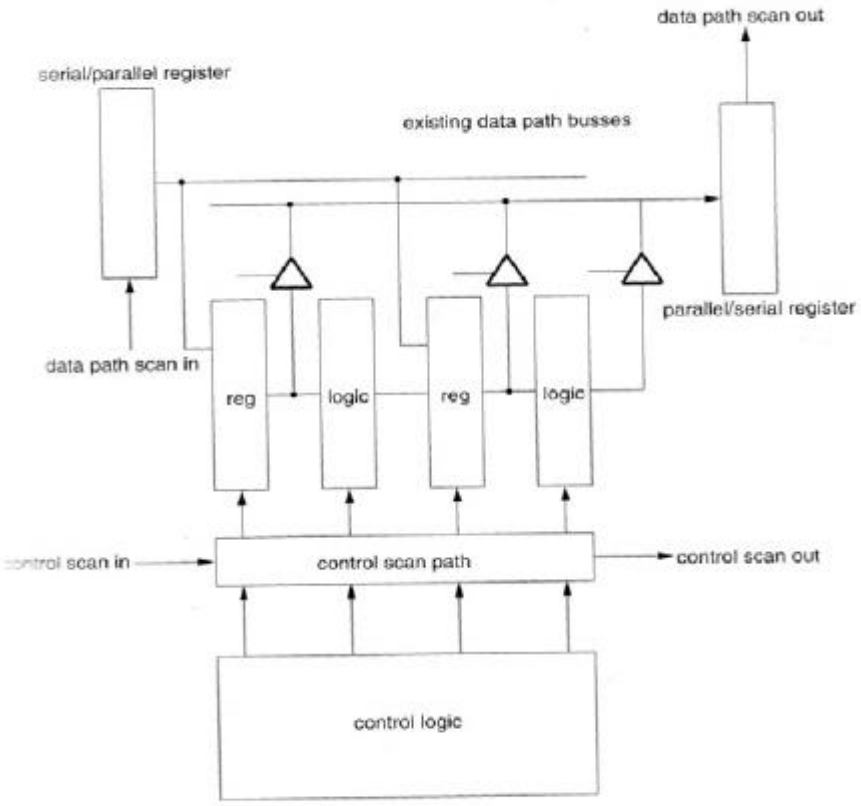
In the past the design process was frequently divided between a designer who designed the circuit and a test engineer who designed the test to apply to that circuit. The advent of the ASIC, small design teams, the desire for reliable ICs and rapid times to market have all forced the “test problem” earlier in the design cycle. In fact, the designer who is only thinking about what functionality has to be implemented and not about how to test the circuit will quite likely cause product deadlines to be slipped and in extreme cases precuts to be stillborn. In this section some practical methods of incorporating test requirements into a design. This discussion is structured around the main types of circuit structure that will been countered in a digital CMOS chip

Regular Logic Array

Partial serial scan or parallel scan is probably the best approach for structure such as dat a paths. One approach that has been used in a Lisp microprocessor is shown in figure . Here the input busses may be driven by a serially loaded register. These in turn may be sourced onto a bus, and this bus may be loaded into a register that may be serially accessed. All of the control signals to the data path are also made scannable

Memories

Memories may use the self-testing techniques mentioned in section 5.3.4.2.alternatively, the provision of multiplexers on data inputs and addresses and convenient external access to data outputs enables the testing of embedded memories. It is a mistake to have memories indirectly accessible (i.e., data is written by passing through logic, data is observed after passing through logic, addresses cannot be conveniently sequenced). Because memories have to be tested exhaustively, any overhead on writing and reading the memories can substantially increase the test time and, probably more significantly, turn the testing task into an effort inscrutability



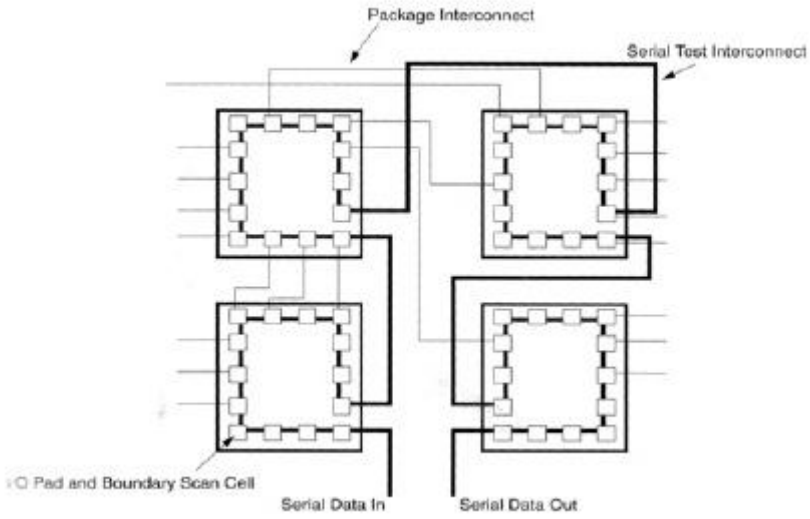
SYSTEM-LEVEL TEST TECHNIQUES

Traditionally at the board level, “bed-of-nails” testers have been used to test boards. In this type of a tester, the board under test is lowered onto a set of test points that probe points of interest on the board. These may be sensed and driven to test the complete board. At the chassis level, software programs are frequently used to test a complete board set. For instance, when a computer boots, it might run a memory test on the installed memory to detect possible faults. The increasing complexity of boards and the movement to technologies like Multichip Modules (MCMs) and surface-mount technologies resulted in system designers agreeing on a unified scan-based methodology for testing chips at the board(and system level). This is called Boundary Scan

Boundary Scan

Introduction

The IEEE 1149 Boundary Scan architecture is shown in figure. In essence it provides a standardized serial scan path through the I/O pins of an IC. At the board level, ICs obeying the standard may be connected in a variety of series and parallel combinations to enable testing of a complete board or, possibly, collection of boards. The description here is a précis of the published standard .The standard allows for the following types of tests to be run in a unified testing framework



The Test Access Port (TAP)

The Test Access Port (TAP) is a definition of the interface that needs to be included in an IC to make it capable of being included in a Boundary-Scan architecture. The port has four or five single-bit connections, as follows

TCK (The Test Clock Input) – used to clock tests into and out of chips.

TMS (The Test Mode Select) –used to control test operations.

TDI (The Test data Input) – used to input test data to a chip.

TDO (the Test Data Output) –used to output test data from a chip. It also has an optional signal

TRST (The Test Reset Signal) used to asynchronously reset the TAP controller, also used if a power-up reset signal is not available in the chip being tested. The TDO signal is defined as a tri-state signal that is only driven when the TAP controller is outputting test data

The TDO signal is defined as a tri-state signal that is only driven when the TAP controller is outputting test data.

The Test Architecture

The basic test architecture that must be implemented on a chip is shown in figure5.25 it consists of: TAP interface pins a set of test-data registers to collect data from the chip an instruction register to enable test inputs to be applied to the

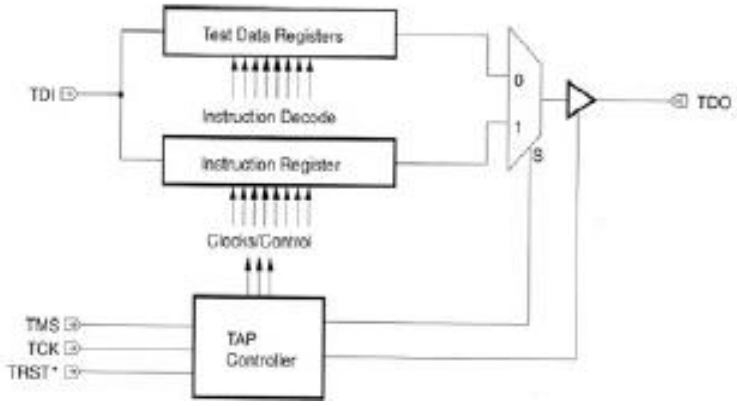


Figure 5.25 TAP architecture

chip a TAP controller, which interprets test instructions and controls the flow of data onto and out of the TAP .Data that is input via the TDI port may be fed to one or more test data registers or an instruction register. An output MUX selects between the instruction register and the data registers to be output to the tri-state TDO pin.