Chapter 1: Introduction to CMOS Circuits

• 1.1 MOS (Metal Oxide Silicon) Transistor History
  – 1925: J. Lilienfeld proposed the basic principle of MOS FET (Field Effect Transistor).
  – 1935: O. Heil proposed a similar structure.
  – 1963: Frank Wanlass (Fairchild) invented inverter, NOR and NAND CMOS gates. This invention starts the era of CMOS low power applications.
  – 1965: The first MOS calculator.
  – 1971: Emergence of nMOS-silicon gate technology.
  – Note: Early research on MOS technology led to success of bipolar transistor. This in turns leads to a decline of interest in MOS transistor.
  – 1980: The market share of MOSFET exceeds bipolar device.

![Graph showing market share of MOSFET and bipolar in the IC market from 1980 to 2000.](image)
1.3 MOS Transistor

- Basic starting material: Single crystal of silicon formed as wafers (4-inch, 6-inch, 8-inch, 12-inch).

- MOS structure is created by superposing several layers of conducting, insulating, and transistor-forming materials to create a sandwich-like structure by way of a series of chemical processing steps such as:
  - oxidation of the silicon, diffusion of impurities into silicon to give it certain conduction characteristics, and
  - deposition and etching of aluminum on silicon to form interconnection.

- Two types of transistors (Figure 1.1):
  - $nMOS$: with negatively diffused (doped) source and drain on lightly p-doped substrate.
  - $pMOS$: with positively diffused source and drain on lightly n-doped substrate.
Four terminals of a transistor:

- **Gate**: usually formed by polycrystalline silicon (polysilicon for short). It is a control input that affects the flow of electrical current between source and drain.
- **Source and Drain**: Formed by diffusion. They are physically equivalent and the name assignment depends on the direction of current flow.
  - Source provides charges.
  - Drain sinks charges.
- **Substrate**: the fourth terminal of MOS transistor and will be discussed later.

Note that p-type transistor (pMOS) has n-doped substrate and n-type transistor (nMOS) has p-doped substrate.
• **1.4 MOS Transistor Switches**
  - The gate controls the flow of current between the source and the drain. This allows us to treat the MOS transistors as simple on/off switches.
  - Logic value system:
    - 1: Between 1.5 and 15 volts
    - z: High Impedance (a circuit node not connecting to either Power or Ground)
    - 0: Zero volts
  - Strength of the “1” and “0” signals:
    - Strength of a signal is measured by its ability to sink or source current.
    - Power (PWR, VDD): Strongest 1.
    - Ground (GND, VSS): Strongest 0.
    - By convention, current is sourced from Power, and Ground sinks current.
    - nMOS switch (N-SWITCH) is closed or ON if there is a “1” on the gate.
      - Pass a good 0.
      - Pass a poor 1.
    - pMOS switch (P-SWITCH) is closed or ON when there is a “0” on the gate.
      - Pass a good 1.
      - Pass a poor 0.
**Figure 1.2** nMOS and pMOS switch symbols and characteristics

**Table 1.1** The Output Logic Levels of N-Switches and P-Switches

<table>
<thead>
<tr>
<th>Level</th>
<th>Symbol</th>
<th>Switch Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong 1</td>
<td>1</td>
<td>P-SWITCH gate = 0, source = $V_{DD}$</td>
</tr>
<tr>
<td>Weak 1</td>
<td>1</td>
<td>N-SWITCH gate = 1, source = $V_{DD}$ or P-SWITCH connected to $V_{DD}$</td>
</tr>
<tr>
<td>Strong 0</td>
<td>0</td>
<td>N-SWITCH gate = 1, source = $V_{SS}$</td>
</tr>
<tr>
<td>Weak 0</td>
<td>0</td>
<td>P-SWITCH gate = 0, source = $V_{SS}$ or N-SWITCH connected to $V_{SS}$</td>
</tr>
<tr>
<td>High impedance</td>
<td>Z</td>
<td>N-SWITCH gate = 0 or P-SWITCH gate = 1</td>
</tr>
</tbody>
</table>
Complimentary switch (C-SWITCH) or Transmission gate

- Pass a good 1 and a good 0.

**FIGURE 1.3** A complementary CMOS switch
1.5 CMOS Logic

1.5.1 Inverter:

- Truth Table:

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- “Input 0 → Output 1” suggests a P-SWITCH connected from a “1” source ($V_{DD}$) to the output.
- “Input 1 → Output 0” suggests an N-SWITCH connected from a “0” source ($V_{SS}$) to the output.

Inverter Circuit
• Fully complementary CMOS gate always has an N-SWITCH array (pull-down) to connect the output to “0” ($V_{SS}$) and a P-SWITCH array (pull-up) to connect the output to “1” ($V_{DD}$).

![Diagram of CMOS gate with N-SWITCHES and P-SWITCHES]

- With independent inputs, the output logic level of a CMOS gate can be in any of the following four states:

<table>
<thead>
<tr>
<th>PULL-DOWN OUTPUT</th>
<th>PULL-UP OUTPUT</th>
<th>COMBINED OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Z</td>
<td>0</td>
</tr>
<tr>
<td>Z</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Z</td>
<td>Z</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Crowbarred</td>
</tr>
</tbody>
</table>
1.5.2 Combinational Logic:

- If two N-SWITCHES (or P-SWITCHES) are placed in series, this structure yields an “AND” function.

- If two N-SWITCHES (or P-SWITCHES) are placed in parallel, this structure yields an “OR” function.

- By using combinations of these structures, CMOS combinational gates can be constructed.

**FIGURE 1.5** Connection and behavior of series and parallel N-SWITCHES and P-SWITCHES
• **1.5.3 The NAND Gate:** \( F = \overline{A \cdot B} \) (derive the NAND function from schematics)
  
  Using the structures in Fig. 1.5(a) and Fig. 1.5(d)
- **Pull-down truth table**

<table>
<thead>
<tr>
<th>SWITCH A CONTROL INPUT</th>
<th>SWITCH B CONTROL INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Pull-up truth table**

<table>
<thead>
<tr>
<th>SWITCH A CONTROL INPUT</th>
<th>SWITCH B CONTROL INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Z</td>
</tr>
</tbody>
</table>

- **NAND gate truth table**

<table>
<thead>
<tr>
<th>OUTPUT</th>
<th>A INPUT</th>
<th>B INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Z</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Z</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Z</td>
<td>0</td>
</tr>
</tbody>
</table>
1.5.4 The NOR Gate $F = \overline{A + B}$ (Synthesize the NOR structure from NOR function)

- NOR gate Karnaugh Map and truth table

<table>
<thead>
<tr>
<th>TABLE 1.7</th>
<th>Karnaugh Map for 2-input NOR Gate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OUTPUT</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>B INPUT</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>A INPUT</strong></td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 1.8</th>
<th>2-input CMOS NOR Gate Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OUTPUT</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>A INPUT</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>B INPUT</strong></td>
<td>0</td>
</tr>
</tbody>
</table>

- From pull-down truth table, this suggests that the structure in Fig. 1.5(c) be used for N-SWITHCES array.
- From pull-up truth table, this suggests the structure in Fig. 1.5(b) be used for P-SWITHCES array.

Note that

1. the outputs of these are fully restored to “1” or “0” for any input combinations since there is always a path from $V_{DD}$ or $V_{SS}$ supplies to the output and the full supply voltages appear at the outputs. This considerably simplifies the circuit design.

2. the output is never at crowbarred state because there is never a path from the VDD to the VSS supplies for any input combination.
1.5.5 Compound Gates:

- A compound gate is formed by using a combination of series and parallel-switch structures.

- For example: to construct \( F = ((A \cdot B) + (C \cdot D)) \)

  - For n-side (pull-down),
    1. take the expression \( ((A \cdot B) + (C \cdot D)) \).
    2. From \( A \cdot B \) by using the structure in Fig. 1.5(a).
    3. Form \( C \cdot D \) by using the structure in Fig. 1.5(a).
    4. Treat the structures obtained in (2) and (3) as two simple switches and then use the structure in Fig. 1.5(c) to form \( (A \cdot B) + (C \cdot D) \).

  - For p-side (pull-up),
    1. Convert the expression \( ((A \cdot B) + (C \cdot D)) \) into sum-of-product \( (A + \overline{B}) \cdot (C + \overline{D}) \) and disregard the “bar” on each input variable, we obtain the expression \( (A + B) \cdot (C + D) \).
    2. Form \( (A + B) \) by using the structure in Fig. 1.5(c).
    3. Form \( (C + D) \) by using the structure in Fig. 1.5(c).
    4. Treat the structures obtained in (2) and (3) as two simple switches and then use the structure in Fig. 1.5(b) to form \( (A + B) \cdot (C + D) \).
Yet another example for \( F = (A + B + C) \cdot D \)

**FIGURE 1.9** CMOS compound gate for function \( F = ((A + B + C) \cdot D) \)

Implementation of 4 input AND gate \( F = A \cdot B \cdot C \cdot D \)

**FIGURE 1.10** Various implementations of a CMOS 4-input AND gate
1.5.6 Multiplexers:

- Complementary switches can be used to select between a number of inputs, thus forming a multiplexer function.

- 2-input CMOS multiplexer: Output = \( A \cdot S + B \cdot \overline{S} \)

**TABLE 1.9** 2-input Multiplexer Karnaugh Map

<table>
<thead>
<tr>
<th>AB INPUTS</th>
<th>S(\overline{S}) INPUTS</th>
<th>0(1)</th>
<th>1(0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
• **1.5.7 Memory-Latches and Registers**
  - Level sensitive latch: data input is passed to the output during the moment that clock is active. (for example, clock is high)
    - Inputs: D and CLK.
    - Outputs: Q and -Q.
    - when CLK=1, Q is set to D and -Q is set to -D.
    - When CLK=0, Q and -Q do not change even the input D would change.

![Diagram of a CMOS positive-level-sensitive D latch](image)

**FIGURE 1.12** A CMOS positive-level-sensitive D latch
- **edge-triggered register**: data is transmitted only when clock is changing its state.
  - By combining two level-sensitive latches, one positive sensitive (called slave) and one negative sensitive (called master) as shown in the following figure.

![Diagram of edge-triggered register](image)

**FIGURE 1.13** A CMOS positive edge-triggered D register

- **CLK = 0**: 
  - QM follows D
  - Q is stored (isolated from D)

- **CLK = 1**: 
  - QM transferred to Q
  - Q is still isolated from changes in D
1.6 Circuit and System Representations

- To facilitate the designing of complex digital systems, a specific set of abstractions have been developed to describe integrated electronic systems.

**FIGURE 1.14** Digital design domains and levels of abstraction
– Three distinct domains:
  – Behavioral domain specifies what a system does.
  – Structural domain specifies how entities are connected together to effect the prescribed behavior.
  – Physical domain specifies how to actually build a structure that has the required connectivity to implement the prescribed behavior.
– Each design domain may be specified at a variety of levels of abstraction.
– Six levels of abstraction:
  – Architectural level,
  – Algorithmic level,
  – Module or functional block level,
  – Logical level,
  – Switch level, and
  – Circuit level.
– Which domain and what level of abstraction can be used for a design?
  Ans.: A design is expressed in terms of the three design domains, while the levels of abstraction that are used depends on design style and circuit complexity.
1.6.1 Behavioral Representation

- A behavioral representation describes how a particular design should respond to a given set of inputs.
- Behavior may be specified by Boolean equations, tables of input and output values or algorithms.
- Behavior can be implemented by
  - Standard high level languages such as the C language, etc.
  - Hardware description language such as VHDL, Verilog, etc.
- Example: An n-bit adder by cascading n 1-bit adders

1-bit adder: A and B are two operand inputs and C is carry input; S is a sum output and CO is a carry output. The behavior of 1-bit adder can be represented by

- Boolean functions:
  
  \[ S = A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C + \overline{A} \cdot B \cdot \overline{C} + A \cdot B \cdot C \]

  \[ CO = A \cdot B + A \cdot C + B \cdot C \]

- Truth table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>CO</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
• A verilog module

```verilog
module carry (co, a, b, c);
    output co;
    input a, b, c;
    wire co = (a&b) | (a&c) | (b&c);
endmodule
```

• A primitive

```verilog
primitive carry (co, a, b, c);
    output co;
    input a, b, c;
    table
        // a b c    co
        1 1 ? : 1 ;
        1 ? 1 : 1 ;
        ? 1 1 : 1 ;
        ? 0 0 : 0 ;
        0 ? 0 : 0 ;
        0 0 ? : 0 ;
    endtable
endprimitive
```

• A verilog module with gate delay

```verilog
module carry (co, a, b, c);
    output co;
    input a, b, c;
    wire #10 co = (a&b) | (a&c) | (b&c);
endmodule
```
1.6.2 Structural Representation

- A structural specification specifies how components are connected to perform a certain function (behavior).

- 4-bit adder by connecting 4 1-bit modules (components):

```verilog
module add4 (s,c4,ci,a,b) ;
    input [3:0]a,b;
    input ci;
    output [3:0]s;
    output c4;
    wire [2:0] co;
    add a0 (co[0],s[0],a[0],b[0],ci);
    add a1 (co[1],s[1],a[1],b[1],co[0]);
    add a2 (co[2],s[2],a[2],b[2],co[1]);
    add a3 (c4,s[3],a[3],b[3],co[2]);
endmodule
```

- 1-bit adder consisting of sum and carry modules:

```verilog
module add (co,s,a,b,c) ;
    input a,b,c;
    output s,co;
    sum sl (s,a,b,c);
    carry cl (co,a,b,c);
endmodule
```
– Carry module consisting of logic gates:

```verilog
module carry (co,a,b,c) ;
  input a,b,c;
  output co;
  wire x,y,z;
  and g1 (x,a,b);
  and g2 (y,a,c);
  and g3 (z,b,c);
  or g4 (co,x,y,z);
endmodule
```

– Carry module consisting of MOS transistor:

```verilog
module carry (co,a,b,c) ;
  input a,b,c;
  output co;
  wire i1,i2,i3,i4,cn;
  nmos n1 (i1,vss,a);
  nmos n2 (i1,vss,b);
  nmos n3 (cn,i1,c);
  nmos n4 (i2,vss,b);
  nmos n5 (cn,i2,a);
  pmos p1 (i3,vdd,b);
  pmos p2 (cn,i3,a);
  pmos p3 (cn,i4,c);
  pmos p4 (i4,vdd,b);
  pmos p5 (i4,vdd,a);
  pmos p6 (co,vdd,cn);
  nmos n6 (co,vss,cn);
endmodule
```
A structural view of a 4-bit CMOS adder.

Circuit level representation for assessing the timing behavior of a logic gate by SPICE:

```
.SUBCKT CARRY VDD VSS A B C CO
MN1 I1 A VSS VSS NFET W=8U L=1U AD=8P AS=8P
MN2 I1 B VSS VSS NFET W=8U L=1U AD=8P AS=8P
MN3 CN C I1 VSS NFET W=8U L=1U AD=8P AS=8P
MN4 I2 B VSS NFET W=8U L=1U AD=8P AS=8P
MN5 CN A I2 NFET W=8U L=1U AD=8P AS=8P
MP1 I3 B VDD VDD PFET W=6U L=1U AD=8P AS=8P
MP2 CN A I3 VDD PFET W=8U L=1U AD=8P AS=8P
MP3 CN C I4 VDD PFET W=8U L=1U AD=8P AS=8P
MP4 I4 B VDD VDD PFET W=8U L=1U AD=8P AS=8P
MP5 I4 A VDD VDD PFET W=8U L=1U AD=8P AS=8P
MP6 CO CN VDD VDD PFET W=16U L=1U AD=16P AS=16P
MN6 CO CN VSS VSS NFET W=16U L=1U AD=16P AS=16P
C1 I1 VSS 50ff
C2 I2 VSS 50ff
C3 I3 VSS 50ff
C4 I4 VSS 50ff
CA A VSS 100ff
CB B VSS 100ff
CC C VSS 100ff
CCO CO VSS 150ff
.ENDS
```
• **1.6.3 Physical Representation**

  - The physical specification for a circuit is used to define how a particular part has to be constructed to create a specific structure and hence yield the prescribed behavior.

  - Physical layout for the 4-bit adder at module level.

    - **Written in HDL**

      ```
      module add4;
      input a[3:0],b[3:0];
      input ci;
      output s[3:0];
      output c4;
      boundary [0,0,100,400];
      port a[0] aluminum width=1 origin=[0,25];
      port b[0] aluminum width=1 origin=[0,75];
      port ci polysilicon width=1 origin=[50,0];
      port s[0] aluminum width=1 origin=[100,50];
      .
      add a0 origin = [0,0];
      add a1 origin = [0,100];
      .
      endmodule
      ```

    - **Represented by A picture**

      ![Figure 1.16](image-url)
- Physical layout for the 1-bit adder at transistor level.
  - Written in GDSII (a kind of graphical language)
  - Represented by polygons
1.7 An Example: Triangular waveform generator

1.7.1 Specification

- 4-bit precision
- Use an 8-bit DIP package
- Less than 10 milliwatts power dissipation

Signal description:

- RST: Reset input
- CLK: 1MHZ clock input
- VDD: 5V power supply
- VSS: Ground
- WAVE: 4-bit output
- System level diagram

**Figure 1.17** A system level diagram of the triangle generator chip
1.7.2 Behavioral Description

- **C subroutine**

```c
main (){
  triangle ()
}
triangle ()
{
  int j = 1;
  int i = wave = 0;
  while (1){
    if (wave == 15) j = -1;
    else if (wave == 0) j = 1;
    wave = wave + j;
    printf(stdout,"i=%d wave=%d\n", i++, wave);
  }
}
```

- **Verilog module**

```verilog
module triangle (wave);
output [0:3] wave;
reg clock;
reg [0:3] acc;
initial begin
  acc = 0;
end
// clock waveform
always
  begin
    #100 clock =0;
    #100 clock =1;
  end
// triangle functionality
always @ (posedge, clock)
  begin
    if (wave == 15)
      begin
        inc = -1;
      end
    else if (wave == 0)
      begin
        inc = 1;
      end
    acc = acc + inc;
    wave = acc;
  end
endmodule
```
1.7.3 Structural Description:

- Can be obtained by automatic tools that can convert a behavioral description into a structural one.

- A structural description for the whole system which consists of 8 I/O pads and a module called triangle-gen.

```verbatim
module chip (wave, clk, rst);
input clk, rst;
output [3:0] wave;
wire [3:0] output;
wire chip_clk, chip_rst;
    input_pad i1 (chip_clk, clk);
    input_pad i2 (chip_rst, rst);
triangle_gen tr (output, chip_clk, chip_rst);
output_pad o1 (wave[0], output[0]);
output_pad o2 (wave[1], output[1]);
output_pad o3 (wave[2], output[2]);
output_pad o4 (wave[3], output[3]);
endmodule
```

- A structural description of triangle-gen.

```verbatim
module triangle_gen (output, clk, rst);
output [3:0]output;
input clk, rst;
wire inc;
    inc_dec id1 (output, inc, clk, rst);
    and a1 (s1, output[0], output[1], output[2]);
    nor n1 (s2, output[0], output[1], output[2]);
    or o1 (s3, s1, s2);
    xor x1 (s4, s3, inc);
    dreg d1 (inc, s4, clk, rst);
endmodule
```
A structural description of inc-dec.

```verilog
module inc_dec (output,inc,clk,rst);
output [3:0] output;
input inc,cin,clk,rst;
wire [3:0]co;
    not inv1 (~inc,inc);
    inc_dec_bit id1 (output[0],co[0],inc,_inc,clk,rst);
    inc_dec_bit id2 (output[1],co[1],co[0],_inc,clk,rst);
    inc_dec_bit id3 (output[2],co[2],co[1],_inc,clk,rst);
    inc_dec_bit id4 (output[3],co[3],co[2],_inc,clk,rst);
endmodule
```

A structural description of inc-dec-bit.

```verilog
module inc_dec_bit (sum,co,ci,a,clk,rst);
output sum,co;
input ci,a,clk,rst;
    adder a1 (sum,co,a,q,ci);
    dreg r1 (q,sum,clk,rst);
endmodule
```
A conventional schematic diagram for triangle generator.

FIGURE 1.18 A schematic diagram for the triangle generator module
A hierarchy diagram of the whole chip.

- The cells at the bottom of the hierarchy tree are called leaf cells. There is a corresponding physical description for every leaf cell.
• **1.7.4 Physical Description**
  
  - All modules except leaf cells in a hierarchy tree can be transformed into modules that consist of only leaf cells. Thus a hierarchical design can be transformed into a (flat) design which consist of only leaf cells.
  
  - A physical layout of a design is a physical description that specifies the positions of leaf cells (called cell placement) and the routes of all connections among connections (called net routing).
  
  - Standard cell: A kind of leaf cells usually with the same height and varying width.
  
  - A symbolic representation of the chip layout.

*FIGURE 1.20* The triangle generator chip layout
• **1.7.5 Summary**
  
  – The majority of chip design is and must be highly automated to improve productivity.
  
  – The overall design flow.

**FIGURE 1.21** The design flow for a CMOS chip
• **1.8 CMOS Scorecard**
  
  – Variety of Technology
    
    • GaAs: fastest raw gate speed.
    • Bipolar: speed next to GaAS (Gallium Arsenide).
    • Advanced CMOS: speed close behind Bipolar.
    • CMOS: highest density and lowest power.

  – Main attributes of CMOS
    
    • Fully restored logic levels.
    • Rise and fall times are of the same order.
    • Dense and low power memory.
    • Transmission gates pass both logic levels well.
    • Power supply varies from 1.5 to 15 volts.
    • Precharging characteristics- Both n_type and p_type devices are available for pre charging a bus to \( V_{DD} \) and \( V_{SS} \).
    • Require 2n devices for an n-input gate.
    • Regular layout.

• **1.9 Summary**
  
  – MOS history
  – MOS transistor Switches
  – CMOS Logic
  – Circuit and System Representations