Unit 6:
Registers, Counters, Memories and Programmable Logic Devices

Registers

A register is a memory device that can be used to store more than one bit of information. A register is usually realized as several flip-flops with common control signals that control the movement of data to and from the register.

Common refers to the property that the control signals apply to all flip-flops in the same way

- A register is a generalization of a flip-flop. Where a flip-flop stores one bit, a register stores several bits
- The main operations on a register are the same as for any storage devices, namely
- Load or Store: Put new data into the register
- Read: Retrieve the data stored in the register (usually without changing the stored data.

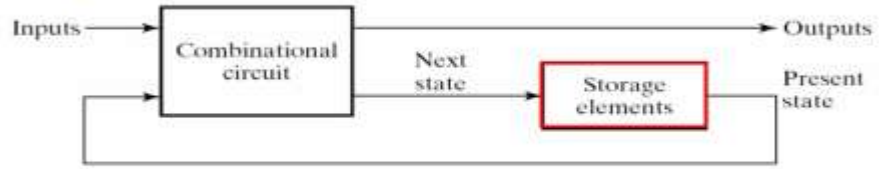
Shift register

A shift register is a device that allows additional inputs or outputs to be added to a microcontroller (in other words, in digital circuits, a **shift register** is a cascade of flip flops, sharing the same clock, in which the output of each flip-flop is connected to the 'data' input of the next flip-flop in the chain, resulting in a circuit that shifts by one position the 'bit array' stored in it, 'shifting in' the data present at its input and 'shifting out' the last bit in the array, at each transition of the clock input.). This is accomplished by converting data between parallel and serial formats. A microprocessor communicates with a shift register using serial information, and the shift register gathers or outputs information in a parallel (multi-pin) format.

Shift registers come in two basic types, SIPO, Serial-In-Parallel-Out, or PISO, Parallel-In-Serial-Out. Here is a SIPO, the 74HC595, and the PISO, the 74HC165. The first type, SIPO, is useful for controlling a large number of outputs, including LEDs, while the latter type, PISO, is good for gathering a large number of inputs, like buttons.


The 74HC595 Shift Register in a DIP package A breakout version of the 74HC165.

- Sequential circuit:



  - storage elements: devices capable of storing binary information (*state*)
  - (inputs, present state) ⇒ (outputs, next state)

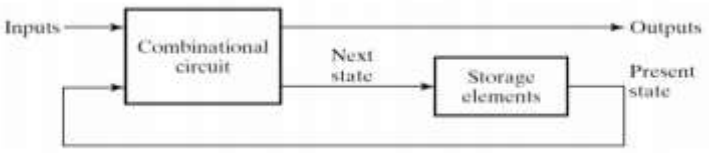
Types of Sequential Circuits

1. Synchronous seq. ckt: – Its behavior can be defined from the knowledge of its signals at discrete instants of time. – Storage elements: e.g., flip-flops
2. Asynchronous seq ckt: – Its behavior depends upon the input signals at any instant of time and the order in which the inputs change. – Storage elements: e.g., latches, feedback paths – Disadvantage: may be unstable & difficult to design

Synchronous Sequential Circuits

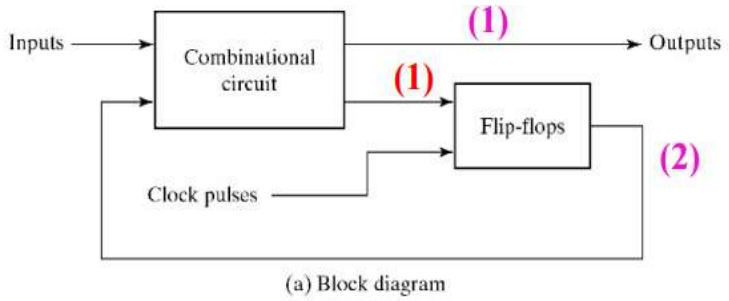
- Clocked seq ckts are the most commonly used sync seq ckts .
- They use clock pulses in the inputs of storage elements.
- It has a master-clock generator to generate a periodic train of clock pulses.
  - The clock pulses are distributed throughout the system.
  - Storage elements are affected only w/ the arrival of each pulse.
- Advantage.: seldom manifest(clear) instability problems
- Storage elements: flip-flops
  - Flip-flop: a binary cell capable of storing one bit of information
  - The state of a flip-flop can change only during a clock pulse transition.

  ⇒ the transition from one state to the next occurs only at predetermined time intervals dictated by the clock pulses.



(Sync) Sequential Circuit Analysis

- Synchronous sequential circuit (ckt): — includes f-fs, the clock inputs driven directly or indirectly by a clock signal and the direct sets and resets are unused during the normal functioning of the ckt
- The behavior of a sequential circuit (ckt)is determined from the inputs, outputs, and present state of the ckt.
- Analysis of a sequential circuit (ckt): — obtain a suitable description that demonstrates the time sequence of inputs, outputs, and states.



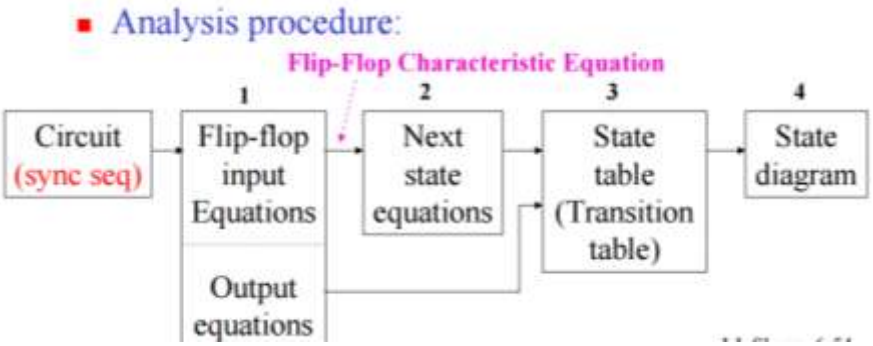
(a) Block diagram

6.2 Analysis of synchronous (Clocked) sequential circuit

Sequential circuit analysis is primarily about expressing the functionality of a sequential circuit in different forms. Each form provides a little more information or insight as to the functionality of the circuit. The sequencial circuit includes the following steps:

1. Excitation equations (or Flip-flop input equations, out put equations)
2. Next state equations
3. Output equations
4. State table
5. State diagram

This procedure can be presented diagrammatically as the following:

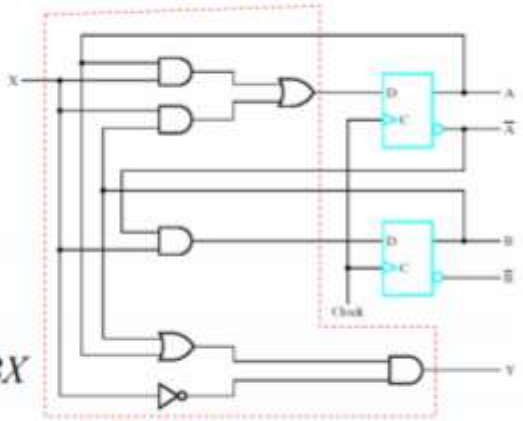


The diagram shows that the analysis of synchronous sequential circuit is facilitated by the use of *state tables* which consist of three vertical sections labeled present state, flip flop inputs, and next state. The present state represents the state of each flip flop before the occurrence of a clock pulse. The flip flop inputs section lists the logic levels (zeros or ones) at the flip flop inputs which are determined from the given sequential circuit. The next state section lists the states of the flip flop outputs after the clock pulse occurrence. The procedure is illustrated with the following examples.

The above stated procedure is exemplified below:

1. Flip-flop input equations & Output equations:

$$D_A = AX + BX$$
$$D_B = \overline{A}X$$
$$Y = (A+B)\overline{X}$$



comb. ckt.

2. Next state equation:

$(Q^+ = D)$

$$A(t+1) = D_A = AX + BX$$
$$B(t+1) = D_B = \overline{A}X$$

J.J. Shann 6-56

3. State table:

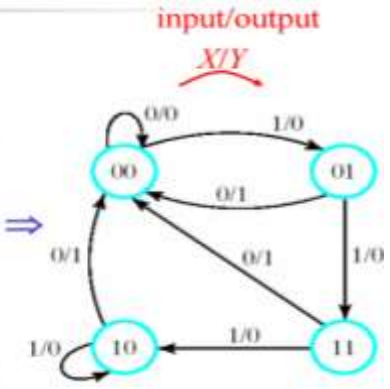
Next state equations: $A(t+1) = AX + BX$, $B(t+1) = \overline{A}X$

Output equations: $Y = A\overline{X} + B\overline{X}$

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

4. State diagram:

Present state		Next state				Output	
		X = 0		X = 1		X = 0	X = 1
A	B	A	B	A	B	Y	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0



6.3 Sequential Circuit Design

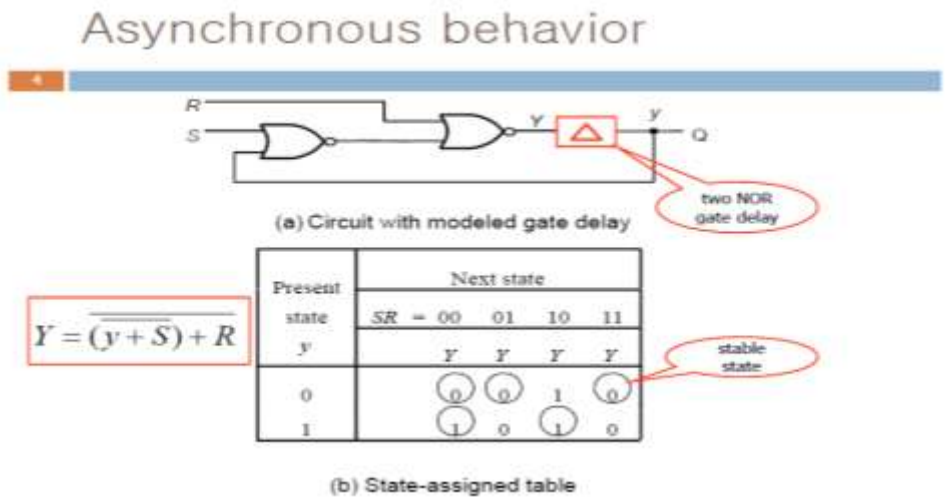
(Sync) Sequential Circuit Design

The following steps /procedural steps are undertaken for the designing of the synchronous sequential Circuit:

1. **Specification**: Writing a specification for the ckt.
2. **Formulation**: Obtain either a state diagram or state table from the statement of the problem.
   * State reduction: Reduce the # of states if necessary.
3. **State assignment**: Assign binary codes to the states and obtain the binary-coded state table.
4. **Flip-flop input equation determination**: Select the flip-flop type or types. Derive the flip-flop input equations from the next-state entries in the encoded state table.
5. **Output equation determination**: Derive output equations from the output entries in the state table.
6. **Optimization**: Optimize the flip-flop input equations and output equations.
7. **Technology mapping**: Draw a logic diagram of the ckt using flip-flops, ANDs, ORs, and inverters. Transform the logic diagram to a new diagram using the available flip-flop and gate technology.
8. **Verification**: Verify the correctness of the final design.

6.4 Asynchronous sequential circuits

In this circuit, internal states can change at **any instant** of time when there is a change in the input variables. (But in synchronous sequential circuits States change in a **periodic** train of clock pulses) . Like synchronous circuits, **No clock** signal is required. Changes in state are dependent on whether each of inputs to the circuit has the logic level 0 or 1 at any given time. It has better performance but hard to design due to timing problems.



Asynchronous sequential circuits

_ do not operate in synchronous with clock signal.

_ do not use F/Fs to represent state variables

_ Changes in state are dependent on whether each of inputs to the circuit has the logic level 0 or 1 at any given time

To achieve reliable operation

_ the inputs to the circuit must change one at a time

_ there must be sufficient time between the changes in input signals to allow the circuit to reach a stable state

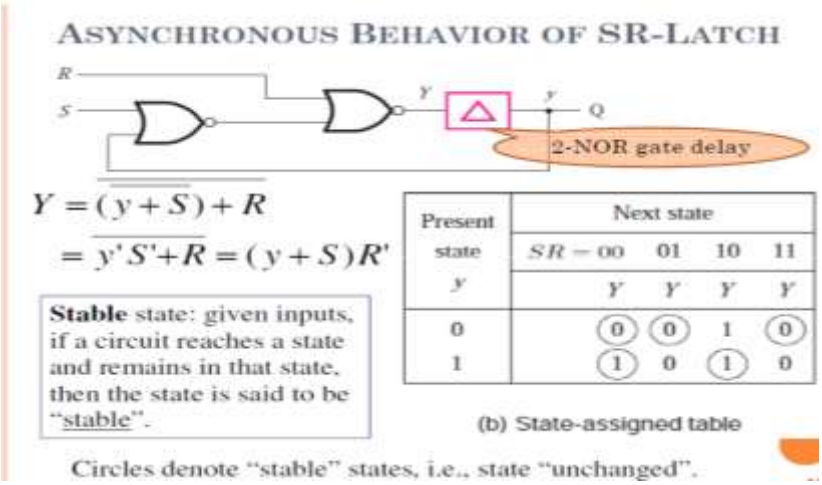
_ A circuit that adheres to these constraints is said to operate in the fundamental mode

6.4 Asynchronous Sequential Circuit Analysis

Steps in the asynchronous circuit analysis process

- Each feedback path is cut
- A delay element is inserted at the point where the cut is made
- A cut can be made anywhere in a particular loop formed by feedback connection, as long as there is only one cut per (state variable) loop
- Next-state and output expressions are derived from the circuit
- The excitation table is derived
- A flow table is obtained
- A corresponding state diagram is derived from the flow table if desired

Study of following analytic diagram



Advantages of asynchronous circuits

_No clock skew (clock signal arrives at different time)

_Lower power (Synchronous : clock signal must be present every time and everywhere.

_Average-case performance VS worst case performance

_Easing of global timing issues

_Partial optimization

_Better external input handling

Drawbacks (Disadvantages/Demerits)

_More difficult to design

_Concerns for hazards and glitches

_Unsure about faster performance

6.5 Problems of asynchronous sequential circuit design

Asynchronous logic is the logic required for the design of asynchronous digital systems. These function without a clock signal and so individual logic elements cannot be relied upon to have a discrete true/false state at any given time. Boolean logic is inadequate for this and so extensions are required. In spite of these problems, there are some problems or demerits of this logic. They are highlighted as follows:

- Area overhead may be up to double the number of circuit elements (transistors), due to addition of completion detection and design-for-test circuits.
- Fewer people are trained in this style compared to synchronous design.
- Synchronous designs are inherently easier to test and debug than asynchronous designs
- Clock gating in more conventional synchronous designs is an approximation of the asynchronous ideal, and in some cases, its simplicity may outweigh the advantages of a fully asynchronous design.
- Performance (speed) of asynchronous circuits may be reduced in architectures that require input-completeness (more complex data path).
- Incompatible with commercial EDA tools.

6.6 Memories: ROM, PROM, EPROM

ROM - Read-Only Memory, is used to store "permanent" copies of software (often) and data (more rarely - except for "lookup tables, dictionaries, and the like,) and comes in several different varieties. It is different from "RAM" - Random Access Memory, in that it is "non-volatile." That means that it doesn't lose its information when the power gets turned off.

Without a modifying prefix, "**ROM**" usually means "mask programmed Read Only Memory," and refers to those memory chips where the contents are actually built onto the chip in the silicon foundry, and can **never** be changed. They were the first of these kinds of devices to become available.

PROM - *Programmable* Read-Only Memory, is shipped from the factory with the possibly to write any kind of contents into the chip that you like … **BUT** it usually can *only be done once*. Typically, each bit of memory is stored with "fusible links," and you "burn" the fuses to set that bit to a 1 or 0. **PROM** chips are usually "programmed" in a special device using high voltage, called a "programmer" or, more informally, a "burner." They were an improvement on ROMs in that you could try out the program in small quantities before

committing to a manufacturing run of ROMs, that has extensive setup involved, and wouldn't be used unless you want to make thousands of copies.

Then there came **EPROM**, which is *Erasable* Programmable Read-Only Memory. That was a big improvement on PROMs, because not only could you burn the data or program that you wanted onto it, but *you didn't have to throw it away if you wanted to make a change*. The early **EPROM**s were erased by exposing them to Ultra-Violet (UV) light, through a glass or crystal window on the top of the chip. You would stick them in a little box with a "blacklight" bulb inside, and after a period of time ranging from half an hour to a couple of days, all of the data would be erased, and you could program ("burn") them all over again. To keep the U-V light from the sun or from regular fluorescent bulbs from erasing the data accidentally, you would cover the window with a sticky label or a small piece of foil when you weren't actually erasing it.

Finally, the memory chip manufacturers came out with **EEPROM** - *Electrically Erasable* Programmable Read-Only Memory. That was an improvement over the earlier EPROMs because it no longer needed U-V light to erase the memory chip. Instead, you would just apply a special voltage to one of the pins on the chip, and *voile!* your chip is back to its empty state and can be programmed all over again. Why was that important? Because *you no longer had to remove the chip from the board to put it into an eraser*. Instead, you could add some circuitry on the board, and program and erase the memory over and over, still installed in the device where it belongs.

Eventually, EEPROM was improved into **Flash memory**, in which the bits can be individually erased and programmed, so that you no longer wiped out the entire memory and reprogrammed ("burned") the whole thing again from scratch. When **Flash** came out, it became possible to add normal memory to a circuit, and have it non-volatile, keeping its contents between uses, and without power. It was immediately used to store images from digital cameras, and then smartphones, in those little SD, Micro-SD, and other "cards" we've come to know and love. **Flash memory** is also the basis for **SSD** - Solid-State Disks, which are used in the new laptops, tablets, and smartphones to provide permanent memory with no moving parts, and in a smaller size, than the hard disks they replace.

Programmable logic device (**PLD)**

A **programmable logic device** (**PLD**) is an electronic component used to build reconfigurable digital circuits. Unlike a logic gate (which has a fixed function), a PLD has an undefined function at the time of manufacture. Before the PLD can be used in a circuit it must be programmed, that is, reconfigured. To understand the idea more clearly, it can be said that PLD are chips which contain programmable circuit. These

chips contain large digital gates that can be programmed. This offered designers the flexibility to program the chip according to the circuit design requirements.

Before PLDs were invented, read-only memory (ROM) chips were used to create ***arbitrary*** combinational logic functions of a number of inputs. Consider a ROM with *m* inputs (the address lines) and *n* outputs (the data lines). When used as a memory, the ROM contains $\mathbf{2^m}$ words of ***n*** bits each.

Programmable logic array (PLA)

In 1970, Texas Instruments developed a mask-programmable IC based on the IBM read-only associative memory or ROAM. This device, the TMS2000, was programmed by altering the metal layer during the production of the IC. The TMS2000 had up to 17 inputs and 18 outputs with 8 JK flip flop for memory. Texas Instruments (TI) coined the term programmable logic array for this device.

A programmable logic array (PLA) has a programmable AND gate array, which links to a programmable OR gate array, which can then be conditionally complemented to produce an output.

Digital Logic Families: TTL, ECL, and CMOS

Different circuit configurations and production technologies are used during the production of digital integrated circuits. Each of these approaches is called a specific Logic Families. Now the idea of having different approaches or different logic families is that each ICs of the same family when fabricated will have identical electrical characteristics. The characteristics which are bound to be identical are supply voltage range, speed of response, dissipation of power, input and output logic levels, current sinking capability, current sourcing capability, noise margin, fan-out etc.

A "logic family" may also refer to a set of techniques used to implement logic within VLSI integrated circuits such as central processors, memories, or other complex functions.

6.8 Digital Logic Families: TTL, ECL, and CMOS

The digital ICs are designed using any of either bipolar devices or MOS or a combination of both. The logic families which fall under the first kind are called bipolar families, this include diode logic (DL), emitted coupled logic (ECL), resistor transistor logic (RTL), diode transistor logic (DTL), transistor transistor logic (TTL). The members of other logic family i.e. MOS family are PMOS, NMOS family, CMOS family. Now the Bi-MOS logic family is the one that uses both bipolar and MOS devices. Of the above mentioned families DL, RTL and DTL are not used these days they have become obsolete. TTL, CMOS, ECL, NMOS and Bi-CMOS are the families which are still used. We will discuss about few of them.

Transistor–transistor logic (TTL)

Transistor–transistor logic (**TTL**) is a class of digital circuits built from bipolar junction transistors (BJTs) and resistors. It is called *transistor–transistor logic* because transistors perform both the logic function (e.g., AND) and the amplifying function (compare with resistor–transistor logic (RTL) and diode–transistor logic (DTL)).

TTL integrated circuits (ICs) were widely used in applications such as computers, industrial controls, test equipment and instrumentation, consumer electronics, and synthesizers. The designation *TTL* is sometimes used to mean TTL-compatible logic levels, even when not associated directly with TTL integrated circuits, for example as a label on the inputs and outputs of electronic instruments.[1]

After their introduction in integrated circuit form in 1963 by Sylvania, TTL integrated circuits were manufactured by several semiconductor companies. The 7400 series (also called 74xx) by Texas Instruments became particularly popular. TTL manufacturers offered a wide range of logic gate, flip-flops, counters, and other circuits. Several variations of the original TTL circuit design were developed. The variations offered interchangeable functions that had higher speed or lower power dissipation to allow design optimization. TTL devices were originally made in ceramic and plastic dual-in-line (DIP) packages, and flat-pack form. TTL chips are now also made in surface-mount packages.

TTL became the foundation of computers and other digital electronics. Even after much larger scale integrated circuits made multiple-circuit-board processors obsolete, TTL devices still found extensive use as the glue logic interfacing more densely integrated components.

Emitter Coupled Logic (ECL)

Emitter-coupled logic (ECL) is the fastest of all logic families and therefore is used in applications where very high speed is essential. High speeds have become possible in ECL because the transistors are used in difference amplifier configuration, in which they are never driven into saturation and thereby the storage time is eliminated. Here, rather than switching the transistors from ON to OFF and vice-versa, they are switched between cut-off and active regions. Propagation delays of less than 1 ns per gate have become possible in ECL.

ECL is also referred to as **Current Mode Logic families**, is a digital technology with extremely high-speed. Transistors are not allowed to go into *deep saturation* thus, eliminating storage delays like in TTL logic families. Transistors are driven either in cut off or in active region. This is achieved by using voltage values

close to each other. For logic one it is -0.9 and for logic zero, it is -1.75v. In the active region, charge stored in the base region of transistors is kept to minimum.

Difference between these two logic states is very small. This improves the speed of operation at the expense of noise margin. Propagation time for an ECL gate is 0.5 to 2ns, which is very less when compared to its TTL counterpart. But the disadvantage of ECL logic families is that it uses a negative power supply such that the logic levels are not compatible with any other logic family and makes analysis and measurement inconvenient. ECL logic families requires large currents therefore power dissipation is 3 to 10 times higher than that of TTL logic families. Because of its large *power consumption* and high requirement of silicon area, CMOS logic gates are preferred over ECL logic families in large scale integrated circuits.

The ECL family was invented by IBM as current steering logic for use in the transistorized IBM 7030 Stretch computer, where it was implemented using discrete components. The first ECL logic family to be available in integrated circuits was introduced by Motorola as *MECL* in 1962

CMOS **(Complementary Metal Oxide Semiconductor)** logic

Because of high noise immunity and low static power dissipation, now CMOS logic families is most preferred in large scale integrated circuits. CMOS (Complementary Metal Oxide Semiconductor) has complementary and symmetrical NMOS and PMOS transistors.

Depending on the input value, only one transistor of the CMOS inverter will be ON at a time. So in both states, there is no direct connection between power supply and ground, thereby reducing static power loss of a transistor.

In contrast to TTL, CMOS uses almost no power in the static state (that is, when inputs are not changing). A CMOS gate draws no current other than leakage when in a steady 1 or 0 state. When the gate switches states, current is drawn from the power supply to charge the capacitance at the output of the gate. This means that the current draw of CMOS devices increases with switching rate (controlled by clock speed, typically).

The first CMOS family of logic integrated circuits was introduced by RCA as *CD4000 COS/MOS*, the 4000 series, in 1968. Initially CMOS logic was slower than LS-TTL. However, because the logic thresholds of CMOS were proportional to the power supply voltage, CMOS devices were well-adapted to battery-operated systems with simple power supplies. CMOS gates can also tolerate much wider voltage ranges than TTL gates

because the logic thresholds are (approximately) proportional to power supply voltage, and not the fixed levels required by bipolar circuits.

The required silicon area for implementing such digital CMOS functions has rapidly shrunk. VLSI technology incorporating millions of basic logic operations onto one chip, almost exclusively uses CMOS. The extremely small capacitance of the on-chip wiring caused an increase in performance by several orders of magnitude. On-chip clock rates as high as 4 GHz have become common, approximately 1000 times faster than the technology by 1970.

The end