Module 4

PASS TRANSISTOR LOGIC

Switches and switch logic can be formed from simple n or p transistors and from the complementary switch ie the transmission gate. The complex transmission gate came into picture because of the undesirable threshold effects of the simple pass transistors. Transmission gate gives good non degraded logic levels. But this good package came at the cost of larger area and complementary signals required to drive the gates.

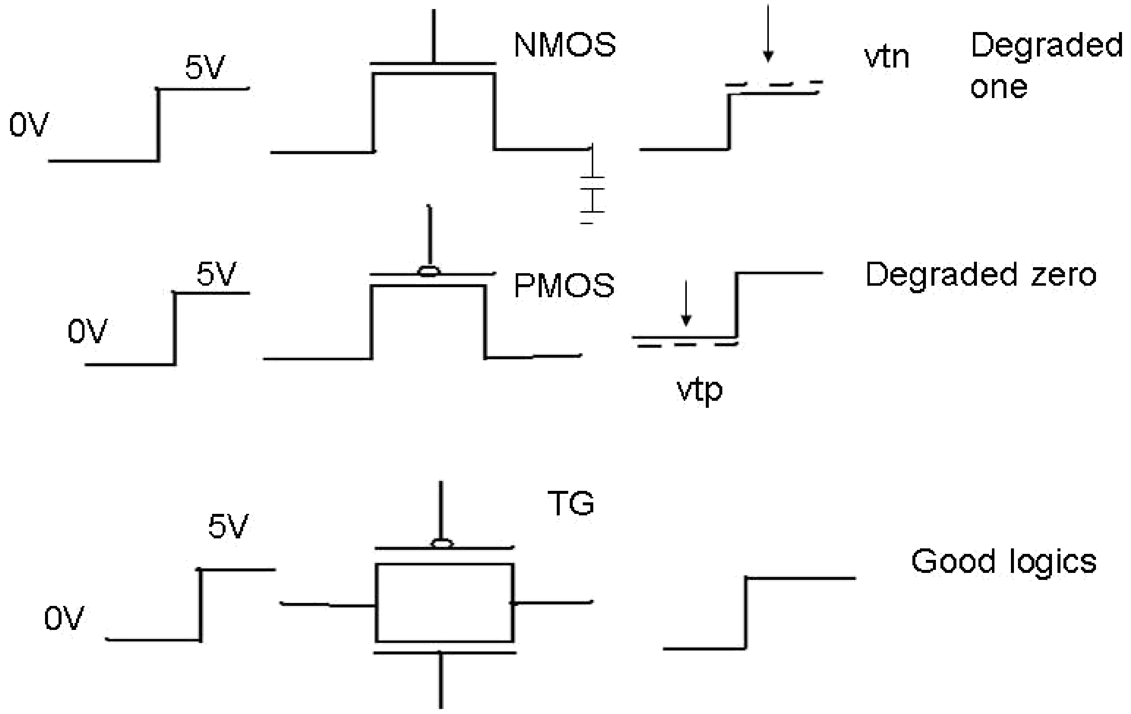


Figure 24: Some properties of pass transistor

CMOS Technology Logic Circuit Structures

Many different logic circuits utilizing CMOS technology have been invented and used in various applications. These can be divided into three types or families of circuits:

1.Complementary Logic

Standard CMOS

Clocked CMOS (C2MOS)

BICMOS (CMOS logic with Bipolar driver)

2.**Ratio Circuit Logic**

Pseudo-NMOS

Saturated NMOS Load

Saturated PMOS Load

Depletion NMOS Load (E/D)
Source Follower Pull-up Logic (SFPL)

3.Dynamic Logic:

CMOS Domino Logic

NP Domino Logic (also called Zipper CMOS)

NOR A Logic

Cascade voltage Switch Logic (CVSL)

Sample-Set Differential Logic (SSDL)

Pass-Transistor Logic

The large number of implementations shown so far may lead to a confusion as to what to use where. Here are some inputs

1.Complementary CMOS

The best option,because of the less dc power dissipation, noise immuned and fast.The logic is highly automated. Avoid in large fan outs as it leads to excessive levels of logic.

2.BICMOS

It can be used in high speed applications with large fanout. The economics must be justified.

PSUEDO –NMOS

Mostly useful in large fan in NOR gates like ROMS,PLA and CLA adders.The DC power can be reduced to 0 in case of power down situations

Clocked CMOS

Useful in hot electron susceptible processes.

CMOS domino logic

Used mostly in high speed low power application. Care must take of charge redistribution. Precharge robs the speed advantage.

CVSL

This is basically useful in fast cascaded logic .The size, design complexity and reduced noise immunity make the design not so popular.

Hybrid designs are also being tried for getting the maximum advantage of each of them into one.

PSEUDO NMOS LOGIC

This logic structure consists of the pull up circuit being replaced by a single pull up pmos whose gate is permanently grounded. This actually means that pmos is all the time on and that now for a n input logic we have only n+1 gates. This technology is equivalent to the depletion mode type and preceded the CMOS technology and hence the name pseudo. The two sections of the device are now called as load and driver. The ßn/ßp (ßdriver/ßload) has to be selected such that sufficient gain is achieved to get consistent pull up and pull down levels. This involes having ratioed transistor sizes so that correct operation is obtained. However if minimum size drivers are being used then the gain of the load has to be reduced to get adequate noise margin.

There are certain drawbacks of the design which is highlighted next

1.The gate capacitance of CMOS logic is two unit gate but for
  psuedo logic it is only one gate unit.

2.Since number of transistors per input is reduced area is reduced drastically.

The disadvantage is that since the pmos is always on, static power dissipation occurs whenever the nmos is on. Hence the conclusion is that in order to use psuedo logic a trade off between size & load or power dissipation has to be made.

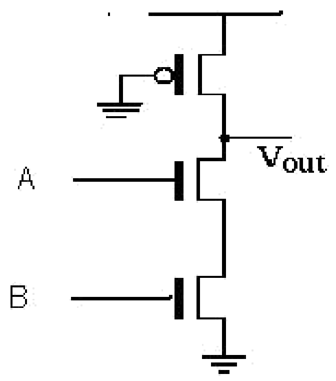


Figure 15 Pseudo Nmos

OTHER VARIATIONS OF PSEUDO NMOS

1.Multi drain logic

Oner way of implementing pseudo nmos is to use multidrain logic. It represents a merged transistor kind of implementation. The gates are combined in an open drain manner, which is useful in some automated circuits. Figure 16

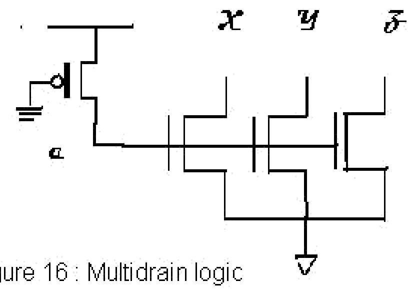
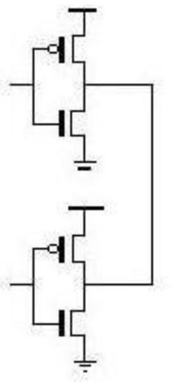


Figure 16 : Multidrain logic

GANGED LOGIC



The inputs are separately connected but the output is connected to a common terminal. The logic depends on the pull up and pull down ratio. If pmos is able to over come nmos it behaves as nandelse nor.

DYNAMIC CMOS LOGIC

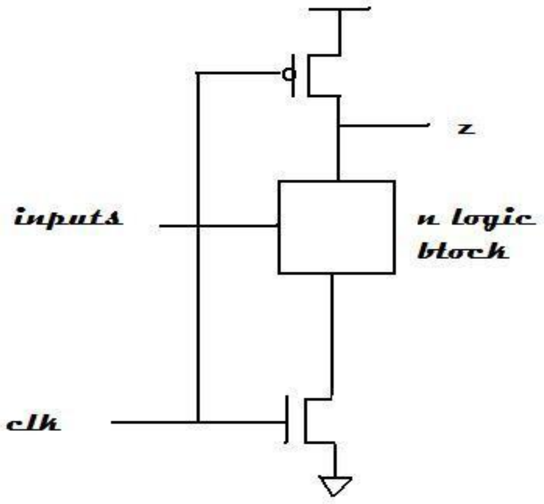


Figure 17 Dynamic cmos logic

This logic looks into enhancing the speed of the pull up device by precharging the output node to vdd. Hence we need to split the working of the device into precharge and evaluate stage for which we need a clock. Hence it is called as dynamic logic. The output node is precharged to vdd by the pmos and is discharged conditionally through the nmos. Alternatively you can also have a p block and precharge the n transistor to vss. When the clock is low the precharge phase occurs. The path to vss is closed by the nmos ie the ground switch . The pull up time is improved because of the active pmos which is already precharged. But the pull down time increases because of the ground switch .

There are a few problems associated with the design, like

1.Inputs have to change during the precharge stage and must be stable during the evaluate. If this condition cannot occur then charge redistribution corrupts the output node.

2.A simple single dynamic logic cannot be cascaded. During the evaluate phase the first gate will conditionally discharge but by the time the second gate evaluates, there is going to be a finite delay. By then the first gate may precharge.

CLOCKED CMOS LOGIC (C2MOS)

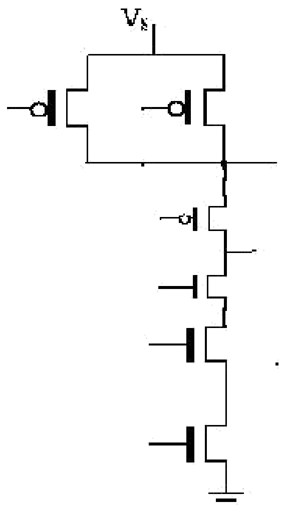


Figure 18 C2mos logic

CMOS DOMINO LOGIC

The disadvantage associated with the dynamic CMOS is over come in this logic. In this we are able to cascade logic blocks with the help of a single clock. The precharge and the evaluate phases retained as they were. The change required is to add a buffer at the end of each stage.

This logic works in the following manner. When the clk=0,ie during the precharge stage the output of the dynamic logic is high and the output of the buffer is low. Since the subsequent stages are fed from the buffer they are all off in the precharge stage. When the gate is evaluated in the next phase, the output conditionally goes low and the output of the buffer goes high. The the subsequent gates make a transition from high to low.

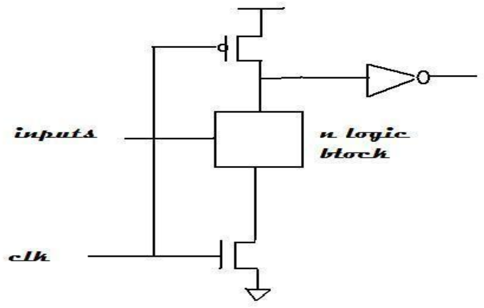


Figure 19:Cmos domino logic

Hence in one clock cycle the cascaded logic makes only one transition from 1 to 0 and buffer makes a transition from 0 to 1.In effect we can say that the cascaded logic falls like a line of dominos, and hence the name. The advantage is that any number of logic blocks can be cascaded provided the sequence can be evaluated in a single clock cycle. Single clock can be used to precharge and evaluate all the logic in a block. The limitation is that each stage must be buffered and only non- inverted structures are possible.

A further fine tuning to the domino logic can also be done. Cascaded logic can now consist of alternate p and n blocks and avoid the domino buffer. When clk=0,ie during the precharge stage, the first stage (with n logic) is precharged high and the second a p logic is precharged low and the third stage is high. Since the second stage is low, the n transistor is off. Hence domino connections can be made.

The advantages are we can use smaller gates, achieve higher speed and get a smooth operation. Care must be taken to ensure

design is correct.

NP DOMINO LOGIC (ZIPPER CMOS)

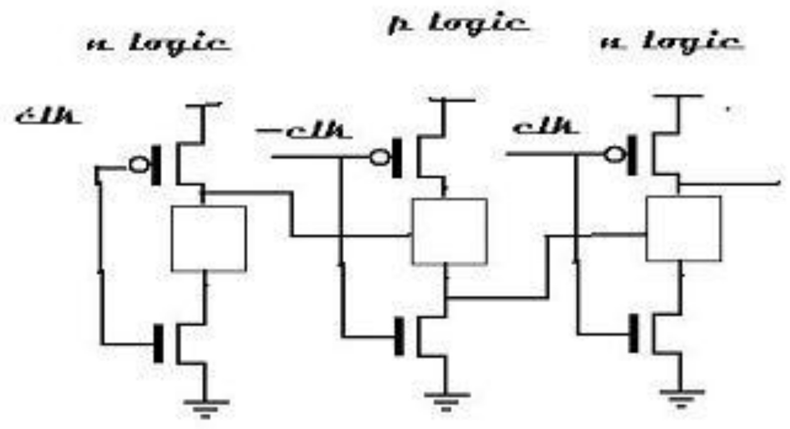


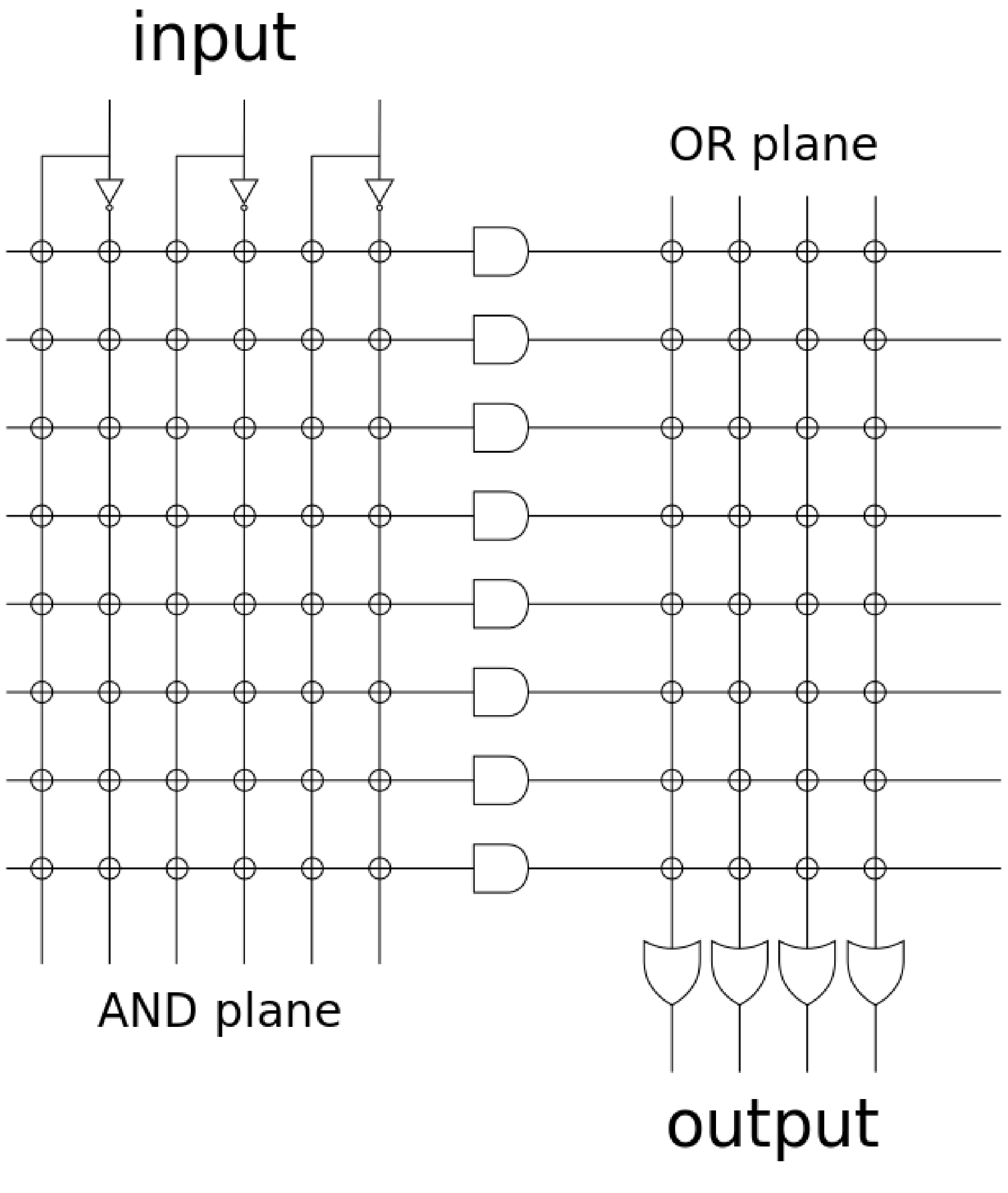
Figure 20: NP domino logic

CASCADED VOLTAGE SWITCH LOGIC

It is a differential kind of logic giving both true and complementary signal outputs. The switch logic is used to connect a combinational logic block to a high or a low output. There are static and dynamic variants .The dynamic variants use a clock. The static version (all the figures to shown next) is slower because the pull up devices have to over come the pull down devices. Hence the clocked versions with a latching sense amplifier came up. These switch logic are called sample set differential logic

Programmable logic array

A **programmable logic array** (**PLA**) is a kind of programmable logic device used to implement combinational logic circuits. The PLA has a set of programmable AND gate planes, which link to a set of programmable OR gate planes, which can then be conditionally complemented to produce an output. It has 2^N AND Gates for N input variables and for M outputs from PLA, there should be M OR Gates, each with programmable inputs from all of the AND gates. This layout allows for a large number of logic functions to be synthesized in the sum of products canonical forms.



Implementation procedure[edit]

1. Preparation in SOP (sum of products) form.
2. Obtain the minimum SOP form to reduce the number of product terms to a minimum.
3. Decide the input connection of the AND matrix for generating the required product term.
4. Then decide the input connections of OR matrix to generate the sum terms.
5. Decide the connections of invert matrix.
6. Program the PLA.

Advantages over read-only memory

The desired outputs for each combination of inputs *could* be programmed into a read-only memory, with the inputs being loaded onto the address bus and the outputs being read out as data. However, that would require a separate memory location for *every* possible combination of inputs, including combinations that are never supposed to occur, and also duplicating data for "don't care" conditions (for example, logic like "if input A is 1, then, as far as output X is concerned, we don't care what input B is": in a ROM this would have to be written out twice, once for each possible value of B, and as more "don't care" inputs are added, the duplication grows exponentially); therefore, a programmable logic array can often implement a piece of logic using fewer transistors than the equivalent in read-only memory. This is particularly valuable when it is part of a processing chip where transistors are scarce (for example, the original 6502 chip contained a PLA to direct various operations of the processor[2]).

Applications

One application of a PLA is to implement the control over a datapath. It defines various states in an instruction set, and produces the next state (by conditional branching). [e.g. if the machine is in state 2, and will go to state 4 if the instruction contains an immediate field; then the PLA should define the actions of the control in state 2, will set the next state to be 4 if the instruction contains an immediate field, and will define the actions of the control in state 4]. Programmable logic arrays should correspond to a state diagram for the system.

Other commonly used programmable logic devices are PAL, CPLD and FPGA.

Note that the use of the word "programmable" does not indicate that all PLAs are field-programmable; in fact many are mask-programmed during manufacture in the same manner as a mask ROM. This is particularly true of PLAs that are embedded in more complex and numerous integrated circuits such as microprocessors. PLAs that can be programmed after manufacture are called FPGA (Field-programmable gate array), or less frequently FPLA (Field-programmable logic array)..

n 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. TI coined the term programmable logic array for this device.[2]

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.

Field Programmable Gate Array (FPGA)

A **field-programmable gate array** (**FPGA**) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare).

FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together", like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory

The most common FPGA architecture[1] consists of an array of logic blocks (called configurable logic block, CLB, or logic array block, LAB, depending on vendor), I/O pads, and routing channels. Generally, all the routing channels have the same width (number of wires). Multiple I/O pads may fit into the height of one row or the width of one column in the array.

The FPGA is Field Programmable Gate Array. It is a type of device that is widely used in electronic circuits. FPGAs are semiconductor devices which contain programmable logic blocks and interconnection circuits. It can be programmed or reprogrammed to the required functionality after manufacturing.

Basics of FPGA

When a circuit board is manufactured and if it contains an FPGA as a part of it. This is programmed during the manufacturing process and further can be reprogrammed later to create an update or make necessary changes.

This feature of FPGA makes it unique from ASIC. Application Specific Integrated Circuits (ASIC) are custom manufactured for specific design task. In past FPGAs are used to develop low speed, complex and volume design, but today FPGA easily pushes the performance barrier up to 500MHz.

In microcontrollers, the chip is designed for a customer and they have to write the software and compile it to hex file to load onto the microcontroller. This software can be easily replaced as it is stored in flash memory.

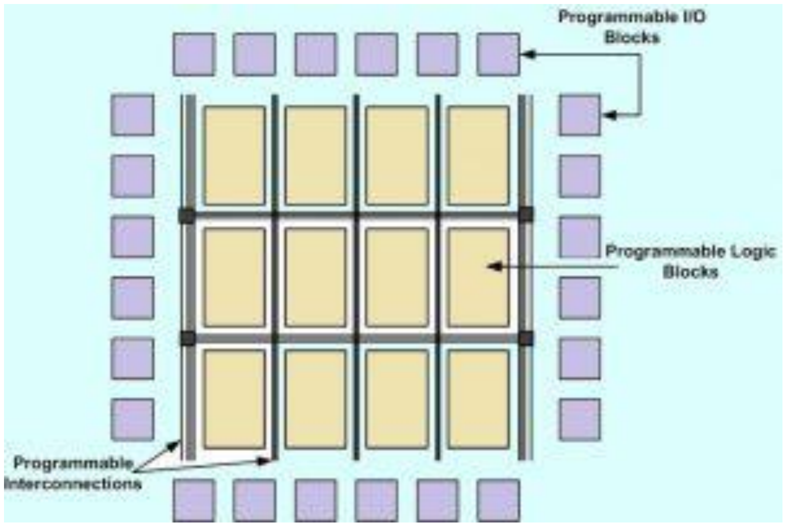
In FPGAs, there is no processor to run the software and we are the one designing the circuit. We can configure an FPGA as simple as an AND gate or a complex as the multi-core processor.

To create a design we write Hardware Description Language (HDL), which is of two types – Verilog and VHDL. Then the HDL is synthesized into a bit file using a BITGEN to configure the FPGA.

The FPGA stores the configuration in RAM, that is the configuration is lost when there is no power connectivity. Hence, they must be configured every time power is supplied.

FPGA Architecture

FPGAs are prefabricated silicon chips that can be programmed electrically to implement digital designs. The first static memory based FPGA called SRAM is used for configuring both logic and interconnection using a stream of configuration bits. Today's modern EPGA contains approximately 3,30,000 logic blocks and around 1,100 inputs and outputs.



FPGA Architecture

The FPGA Architecture consists of three major components

- Programmable Logic Blocks, which implement logic functions
- Programmable Routing (interconnects), which implements functions
- I/O blocks, which are used to make off-chip connections

Programmable Logic Blocks

The programmable logic block provides basic computation and storage elements used in digital systems. A basic logic element consists of programmable combinational logic, a flip-flop, and some fast carry logic to reduce area and delay cost.

Modern FPGAs contain a heterogeneous mixture of different blocks like dedicated memory blocks, multiplexers. Configuration memory is used throughout the logic blocks to control the specific function of each element.

Programmable Routing

The programmable routing establishes a connection between logic blocks and Input/Output blocks to complete a user-defined design unit.

It consists of multiplexers pass transistors and tri-state buffers. Pass transistors and multiplexers are used in a logic cluster to connect the logic elements.

Programmable I/O

The programmable I/O pads are used to interface the logic blocks and routing architecture to the external components. The I/O pad and the surrounding logic circuit form as an I/O cell.

These cells consume a large portion of the FPGA’s area. And the design of I/O programmable blocks is complex, as there are great differences in the supply voltage and reference voltage.

The selection of standards is important in I/O architecture design. Supporting a large number of standards can increase the silicon chip area required for I/O cells.

With advancement, the basic FPGA Architecture has developed through the addition of more specialized programmable function blocks.

The special functional blocks like ALUs, block RAM, multiplexers, DSP-48, and microprocessors have been added to the FPGA, due to the frequency of the need for such resources for applications.

An application circuit must be mapped into an FPGA with adequate resources. While the number of CLBs/LABs and I/Os required is easily determined from the design, the number of routing tracks needed may vary considerably even among designs with the same amount of logic. For example, a crossbar switch requires much more routing than a systolic array with the same gate count. Since unused routing tracks increase the cost (and decrease the performance) of the part without providing any benefit, FPGA manufacturers try to provide just enough tracks so that most designs that will fit in terms of lookup tables (LUTs) and I/Os can be routed. This is determined by estimates such as those derived from Rent's rule or by experiments with existing designs