2077 Chaitra, (Regular)

Ans

1) Explain different purpose of embedded System with example

⇒ The main purpose of an embedded System is to automate the human driven activities such that the task can be performed with higher reliability & efficiency. The different purpose are explain below:

(i) Data collection: In embedded system, the datas collected from other external devices for storage, analysis, manipulation or transmission. Data may be in analog or digital form. System working with digital data require analog & digital Converter if collected data is in analog form. The collected data can be ~~meaning for~~ used for meaningful purpose based on the functionality of embedded system. for instance, a digital camera collects data, stores it & finally provides graphical representation of data in the form of captured image

(ii) Data Communication: An embedded system is required to convert two or more devices which may be at close vicinity or at remote location. Embedded System are incorporated with different wireless modules or wireless modules for communication purpose

for example. If we have to transfer image captured using Camera into Laptop we can use either WIFI or serial communication. (using data cable).

(iii) Data processing : The collected data in embedded system is subjected to some sort of processing for which embedded systems are attributed with data processing modules. Speed coder, audio video codec etc can be few example of data preprocessing unit. Data processing include the manipulation of data for appropriate purpose.

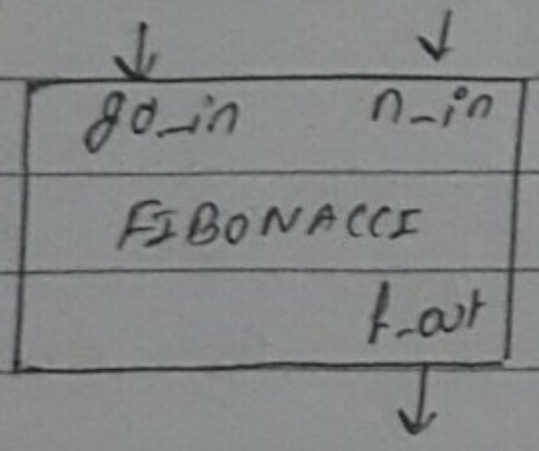
(iv) Application specific user interface :
To provide a better user interface based on application, has been one of the concerns of contemporary embedded systems. Keypads, Simple LCD modules, speakers etc are basic & common interface for users. However sensitive touchpad along with high definition display has been the sophisticated interface implemented in current scenario.

(v) Monitoring : Many embedded systems are incorporated with sensors to check state of different parameters. These parameters can be current, voltage, temperature etc.

(vi) Control : For control purpose, actuator along with sensors are present in embedded system. The sensor connected in input port detects the change in desired parameter & actuator at output port are controlled accordingly to implement desired functionality.

2) Design a single phase processor that output Fibonacci number upto n places. what are the optimization opportunities in single purpose processor.

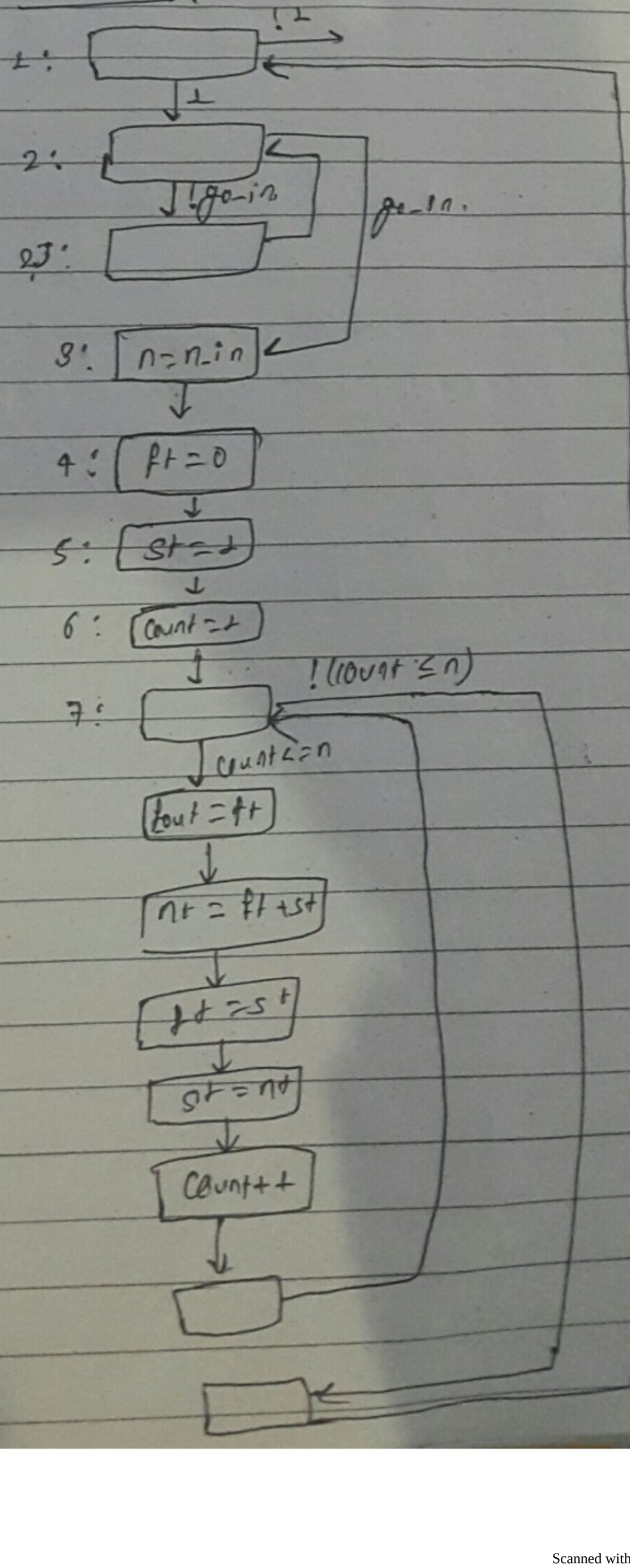
⇒. A. Block Box View



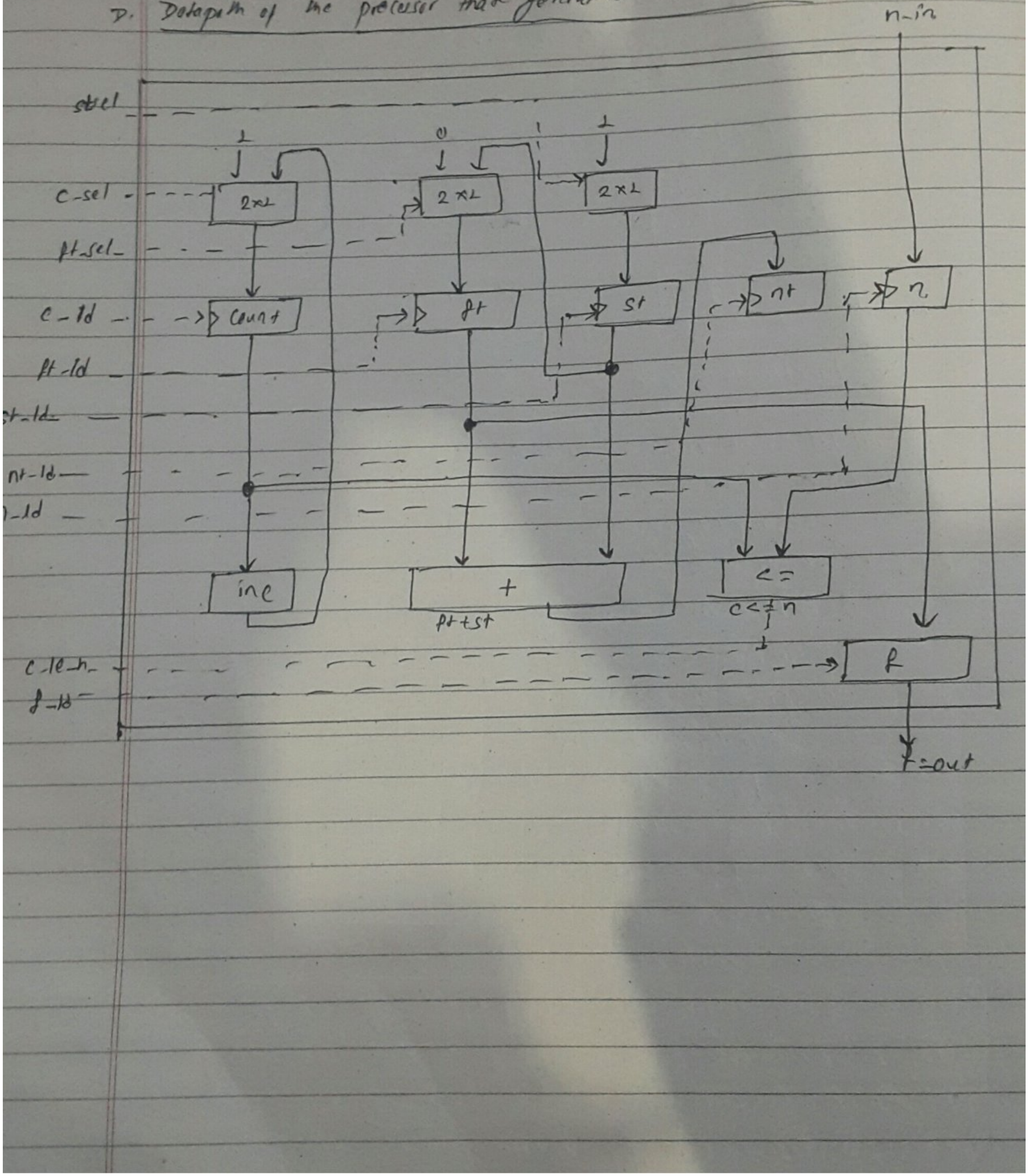
B. Functionality code.

```
int ft, st, nt, count, n;
while(1){
    while(!go_in);
    n = n_in;
    ft = 0;
    st = 1;
    count = 1;
    while (count <= n){
        t_out = ft;
        nt = ft + st;
        ft = st
        st = nt
        count++;
    }
}
```

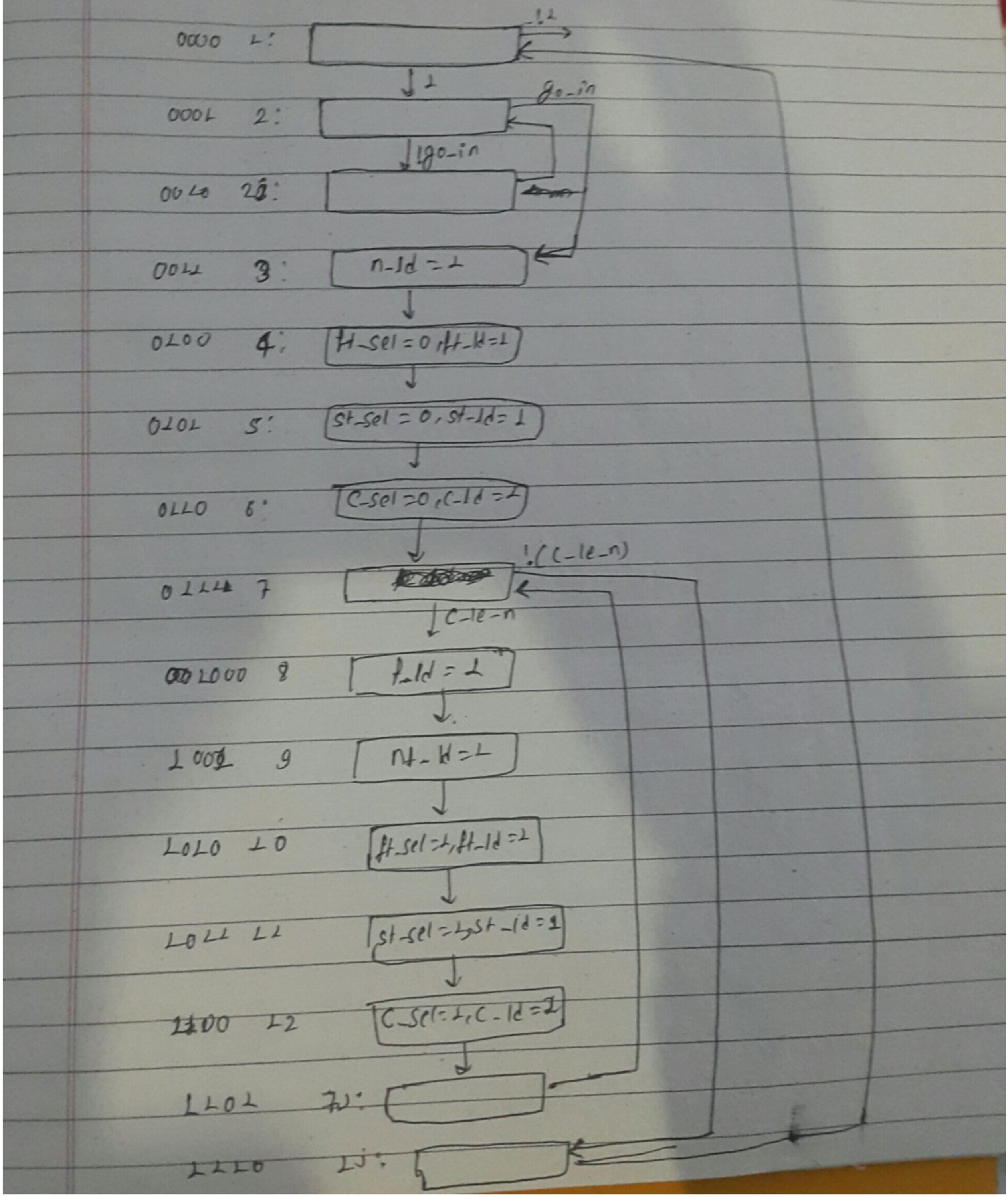
C. FSMD.



D. Datapath of the precursor that generates Fibonacci series:



E. FSM Controller for Fibonacci series generator :



⇒ The optimization opportunities in single purpose processor are as follows:

(i) optimizing the original program
(ii) optimizing the FSMD → merge state, Eliminate state, Separate state
(iii) Optimizing Datapath → sharing of functional unit
                          → use of multifunctional unit
(iv) Optimizing the FSM. → state encoding
                         → state minimization.

3) Explain the architecture of general purpose processor. what are the criteria to select the processor.

⇒

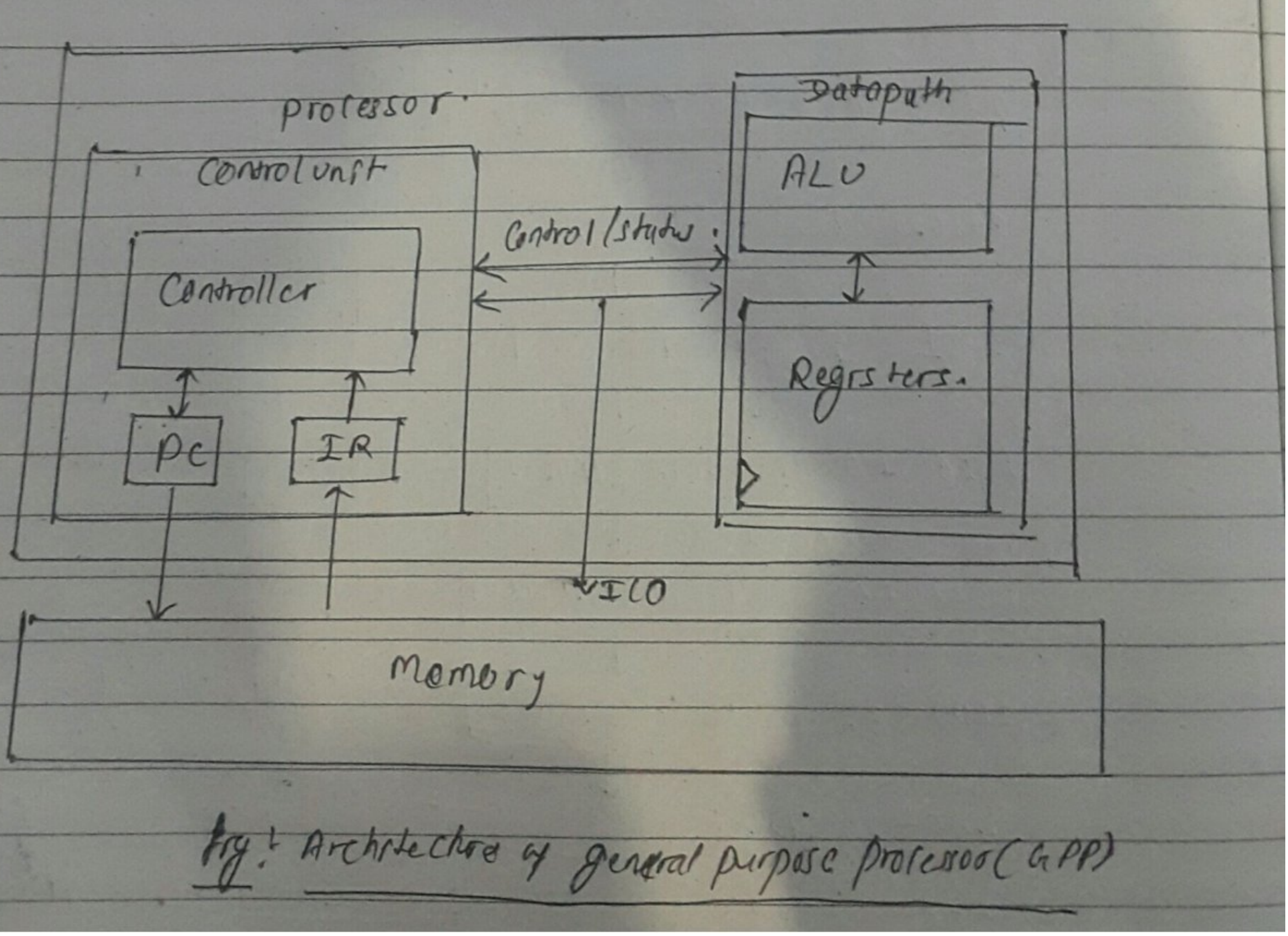


Fig: Architecture of general purpose processor (GPP)

The architecture of GPP can be explain as below

- **Datapath:**
  - → Circuitry for transferring data. ALU to perform operations
  - → mechanism for storing temporary data: Register to store data

- **Control unit:**

  Controller:
  - → Consist of a state register & control logic
  - → Generate Control signal
  - → Fetch cycle
  - → determines next value of PC

- **Memory:**
  - → Registers - short term storage
  - → Memory - medium or long term storage

⇒ The criteria to select the processor are as follows.

(i) Technical aspects: Selection of processor must be done based on required speed within limited power, size & cost

(ii) Non Technical aspect: Before selecting microprocessor, one must be aware of development environment, prior expertise of processor, licensing arrangements & so on.

Speed while base on Technical aspect, Speed of processor. Can be measured & compared using following method.

① Clock speed of processor : 100

→ No of instruction per clock cycle - may differ

→ Speed can be compared based on clock speed of processors.

(ii) Instruction per second:

→ Speed can be evaluated using number of instruction executed per second

→ . Complexity of each instruction may differ

(iii) Dhrystone benchmark :

→ It is a program that runs on different processors & evaluates their performance based on execution of certain operations.

→ Integer arithmetic & string-handling capabilities are examined

→ Since processors can execute such operations thousands of time in a second, Speed of processor may be expressed in terms of Dhrystones per seconds.

(iv) Millions of instruction per second (MIPS) :

→ general measure of computing performance & the amount of work a processor can do.

→ Based on Dhrystone benchmark

→ Can be useful when comparing performance of processors having similar architecture.

→ The origin of MIPS is based on VAX 11/780 which could execute one million instruction per second.

→ It could execute 1757 Dhrystone / second so

1 MIPS ≈ 1757 Dhrystone / second.

4). Design the internal architecture of 4×4 Rom. Explain the memory write ability and storage permanence with example.

⇒ As Rom can be represented by m×n, where m denotes number of word in Rom & n represent number of bits in each word.

For 4×4 Rom, m=4, n=4

Order of Decoder required = $\log_2(m) \times m = 2 \times 4$

Number of Data lines = n=4

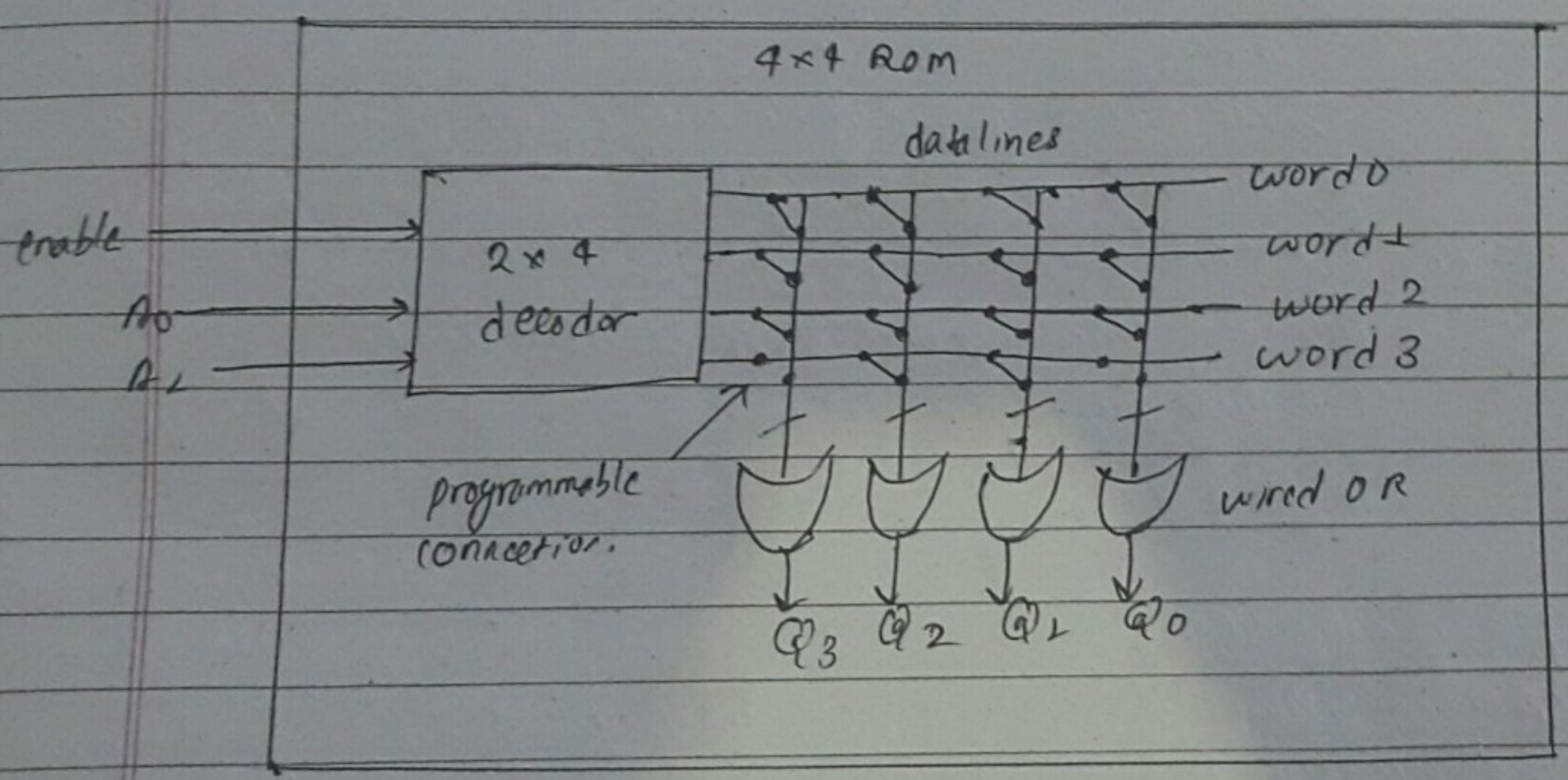


fig: Internal architecture of 4×4 Rom.

⇒ Memory write Ability:

Memory write Ability refers to the manner & speed that a particular memory can be written. It also can be used to represent the number of times a memory can be programmed or written into.

In-system programmables used to categorize memories into two along the write ability axes.

Range of write ability:

(i) High End: processor can write to memory simply & quickly
→ processor can write to memory simply & quickly
eg: RAM.

(ii) Middle range:
→ processor can write to memory a bits slower than high end.
eg: EEPROM, FLASH

(iii) Lower range:
→ special device called Programmer is used to write into memory
→ device must apply suitable voltage level to write in memory.
eg: EPROM (OTP ROM.

(iv) Low end:
→ bits are stored only once during fabrication.
eg: mask programmed rom

Storage permanence:

Storage permanence refers to the ability of memory to hold its stored bits after those bits have been written. Volatile & Non-volatile attribute are commonly used to divide memory types into two categories along storage permanence axis.

Range of storage permanence:

(i) Low End:
→ memory begins to its bits almost immediately after those bits are written.
→ must be refreshed periodically
→ eg: DRAM.

(ii) Low range:-

→ memory holds bits as long as power is supplied to memory

eg:- SRAM.

(iii) Middle Range:

→ memory in this range holds bits for days, months or even years after memory power source has been turned off.

eg:- NVRAM

(iv) High End:

→ memory never loses its bits as long as the memory chip is not damaged

eg:- Mask-programmed ROM.

---

5) Define memory interfacing & write about needs of interfacing. Explain priority arbitration with proper illustration & types.

⇒ Memory interfacing is the process of connecting a microprocessor to external memory device such as RAM, ROM etc.

The purpose of memory interfacing is to allow the microprocessor to store & retrieve data or instruction from external memory devices as needed. Effective memory interfacing is essential for efficient operation of microprocessors and digital system as it enables the system to access large amount of data & instructions quickly & efficiently. It allows for the expansion of memory capacity as needed by additional memory devices to system.

⇒ Arbitration is the mechanism through which a Service or shared resource is provided to a particular requesting device, out of many contending device for services

Priority orbiter is a single purpose processor which is used to orbitrate among various request from peripherals. Each of the peripherals, which are connected to the orbiter can make request for the service. Using certain priority mechanism, orbiter selects a peripheral to permit the request service. This processor mechanism is called priority orbitration.

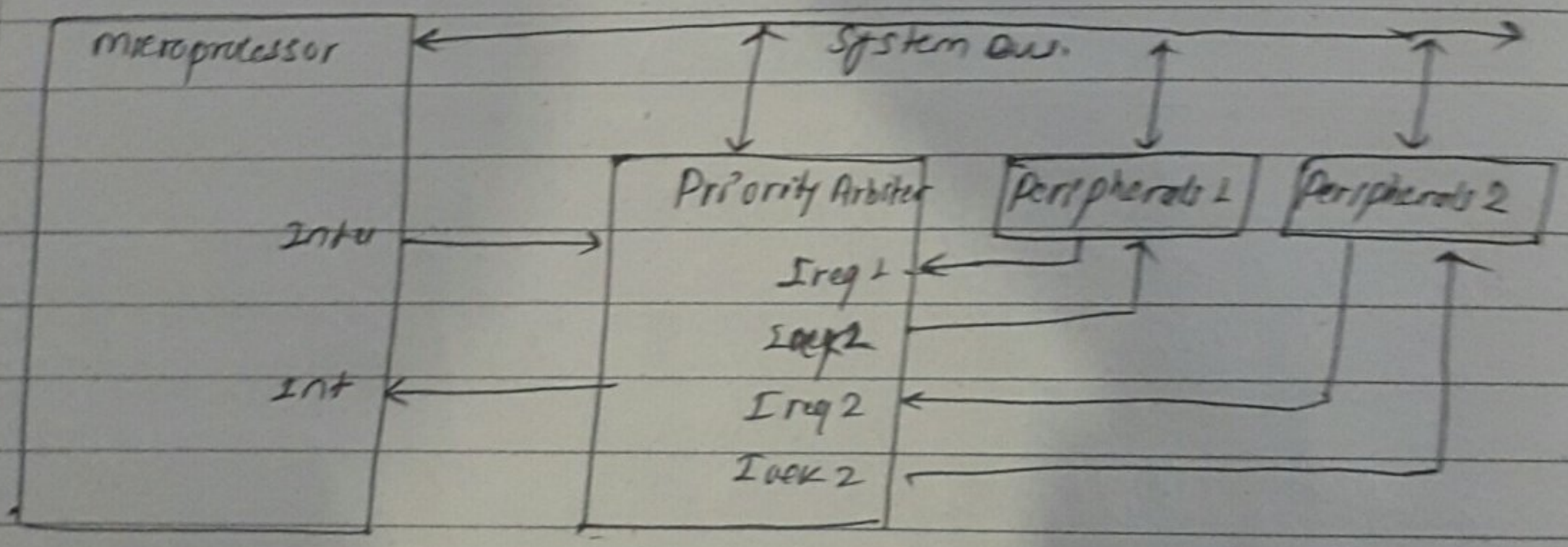


Fig:- Block diagram showing arbitration using a priority arbiter

The steps wise operation of Orbitration using priority arbiter is listed below:

→ Initially, microprocessor is busy in its own operation.
→ Both peripherals can assert request to priority arbiter which interrupts processor when at least one request is available from peripherals.
→ Processor stops its current operation, stores its state & asserts interrupt acknowledgement signal
→ After acknowledge by processor, Priority Arbiter asserts

acknaslodge signal to any one peripheral based on prrorety
→ The Scleeted perripherals puts interrupts address veetor on sys bus.
→ Macroprolessor reads ISR address from databus & jumps into sts, executes ISR.
→ After execution of reqused ISR, prolessor retrieves its state & resumes its operation.

There are two type of prrorety Arbiter.

(i) Fixed · Prrorety: Each perripheral is assigned a unique rank. If two peripherals simultaneously required for service then Orbiter chooses one with the higher rank.

(ii) Rotating Prrorety or Round-robin prrority: In this method, each peripheral gets almost equal time for service from arbiter. This prrority method is efficient when there is not much diff in prrority among peripherals.

6) Compare task, process and threads. Three processes with EDs P1, P2, P3 with estimated completion time 6, 8, 2 ms respectively enters the ready queue together. Process P4 with estimated execution completion time 4 milliseconds enters the ready queue after 1 millisecond. Calculate the waiting time & turn around time for each. Process & Print STAT in the non-preemptive shortest Job first scheduling. Describe basic function of real time kernel.

⇒

Task:

A task is defined as a program in execution & related information maintained by OS for the program. Task is also known as Job in operating system context.

Process:

~~A program or part of it is~~

A process is an instance of program or part of program in execution. A process requires various system resources such as CPU for executing process, memory for storing code. A program by itself is not a process; A program is a passive entity. A process is an active entity. A program becomes a process when an executable program file is loaded into memory.

Threads:

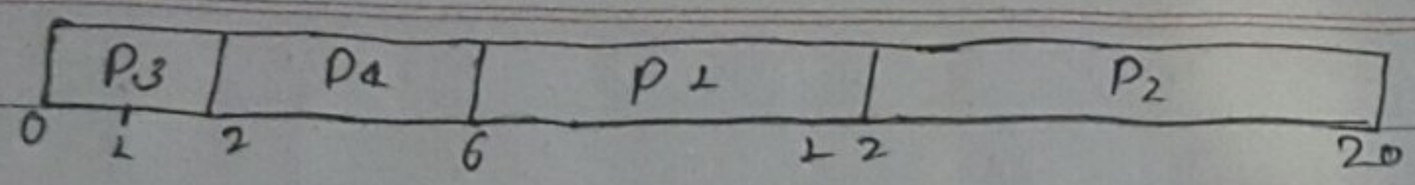
A thread is a basic unit of CPU utilization. It is a single sequential flow of control within a process. A process can have many threads of execution. Different threads which are part of process shares a data memory, code memory & heap memory,. A thread cannot live independently.

⇒

Process	Entered At	Completion Time (ms)
P1	0	6
P2	0	8
P3	0	2
P4	1ms after start of operation	4

Turn Around time = Completion time − entry point

Waiting time = Turn around time − Completion time



Execution sequences of Processes.

Turn Around time Calculation

$P_1 = 12 - 0 = 12$ ms

$P_2 = 20 - 0 = 20$ ms

$P_3 = 2 - 0 = 2$ ms

$P_4 = 6 - 1 = 5$ ms

$ATAT = \frac{12 + 20 + 2 + 5}{4}$

$= 9.75$ ms

Waiting time Calculation

$P_1 = 12 - 6 = 6$ ms

$P_2 = 20 - 8 = 12$ ms

$P_3 = 2 - 2 = 0$ ms

$P_4 = 5 - 4 = 1$ ms

$AWT = \frac{6 + 12 + 0 + 1}{4}$

$= 4.75$ ms

⇒ A real time kernel is software that manages the time of microprocessor to ensure that time-critical events are processed as efficiently as possible. The use of a kernel simplifies the design of embedded system because it allows the system to be divided into multiple independent elements called task. A task is a simple program that thinks it has microprocessor all to itself. Each task has its own stack space each task is dedicated to a specific function in your product. The function of real time kernel are:

(i) RT Kernel is responsible for keeping track of top of stack for each task

(ii) RT Kernal are preemptive i.e Kernel will always try to execute highest priority task that is ready to run.

(iii) Scheduling is a function performed by RT Kernel to determine where a more important task need to run

(iv) RT Kernel provides a mechanism called semaphore so that tasks needs to use in order to access a shared variable array, data structure & I/O device

7) Explain different performance metrics of control system using suitable performance response diagram. Write an algorithm for PID Control.

⇒ Performance metrics describe how well the output is tracking the change in reference input. The various aspects of performance is shown in the figure below.

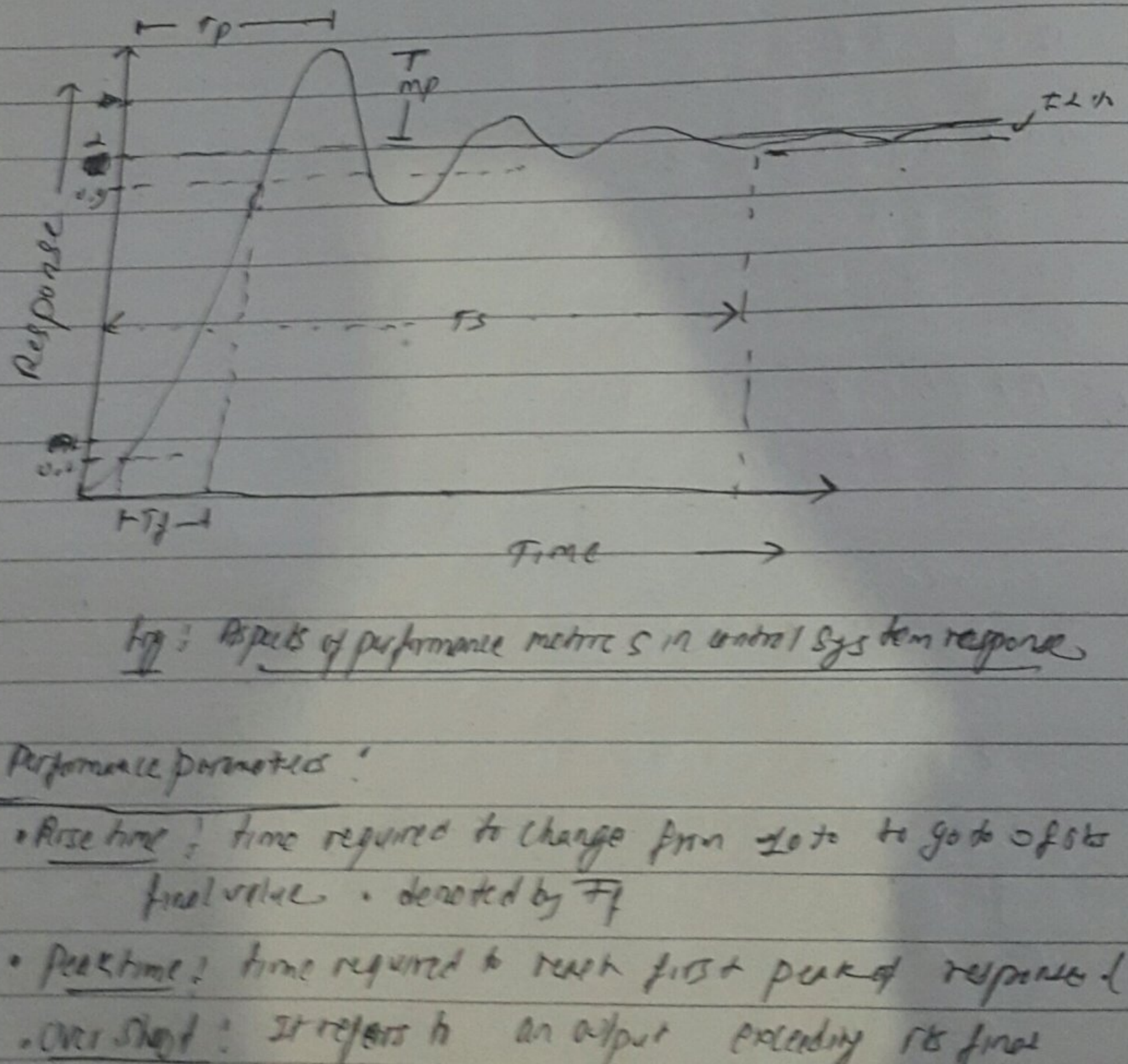


Fig: Aspects of performance metrics in control system response

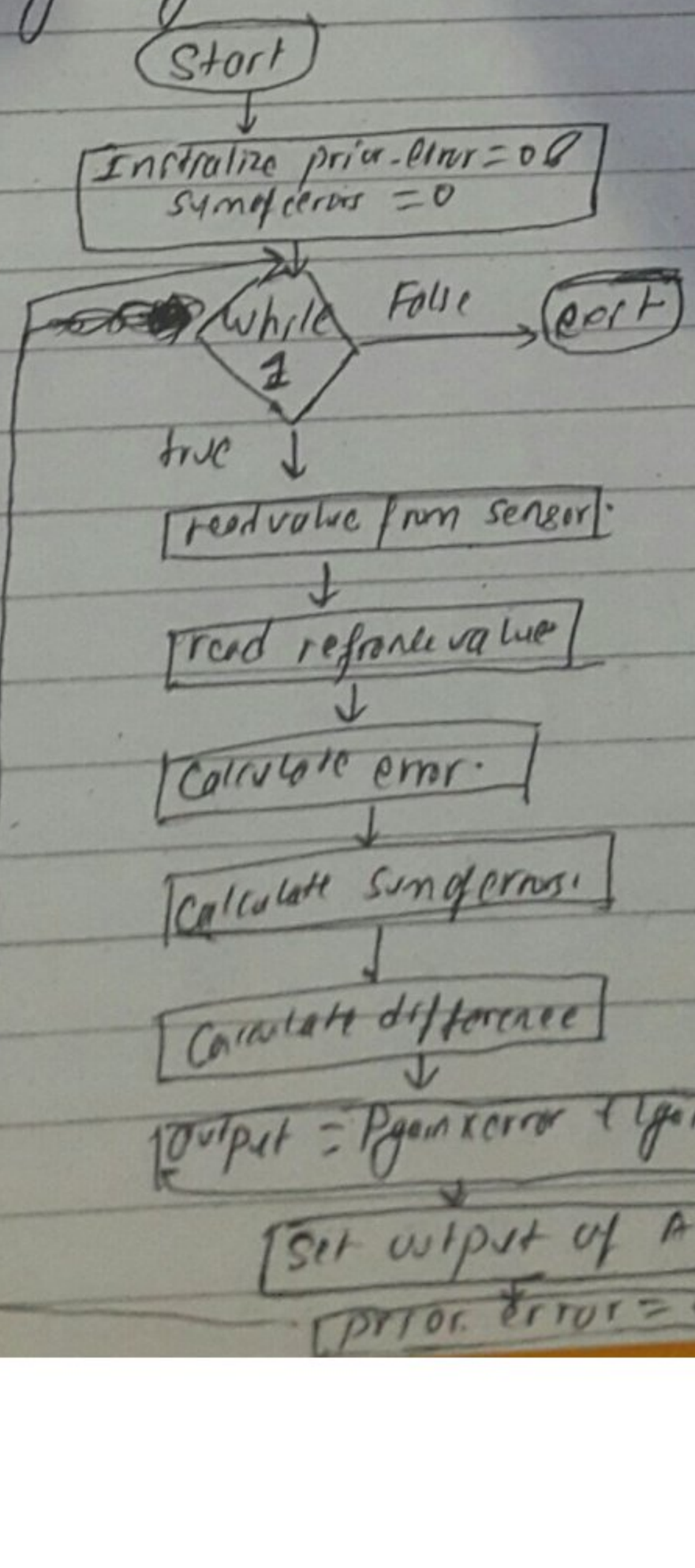
① Performance parameters:

- Rise time: time required to change from 10% to 90% of its final value. denoted by T_r
- Peak time: time required to reach first peak of response (T_p)
- Over Shoot: It refers to an output exceeding its final steady value. denoted by M_p
- Setting time: time required to settle down to within 2% of its final value. denoted by T_s

(ii) Transient response & Steady state response of CS:

Transient response occurs just after the system starts & when any undesired anderton occurs. The system's response during the settling time is transient response. Whereas the steady or state occurs when the system becomes settled. Steady state error is defined as the difference between the actual output & the desired output when system reaches steady state.

⇒ A PID controller can be implemented using software. At first, required initialization is done. Which is followed by reading reference value & Sensor value. Then after that error can be calculated which further is used to compute the output of PID controller. And the refined output is fed to actuator which in turns controls the plant. The flowchart showing algorith can be drawn below.



8) Describe briefly about semi custom IC technology. Explain various steps involved in photolithography.

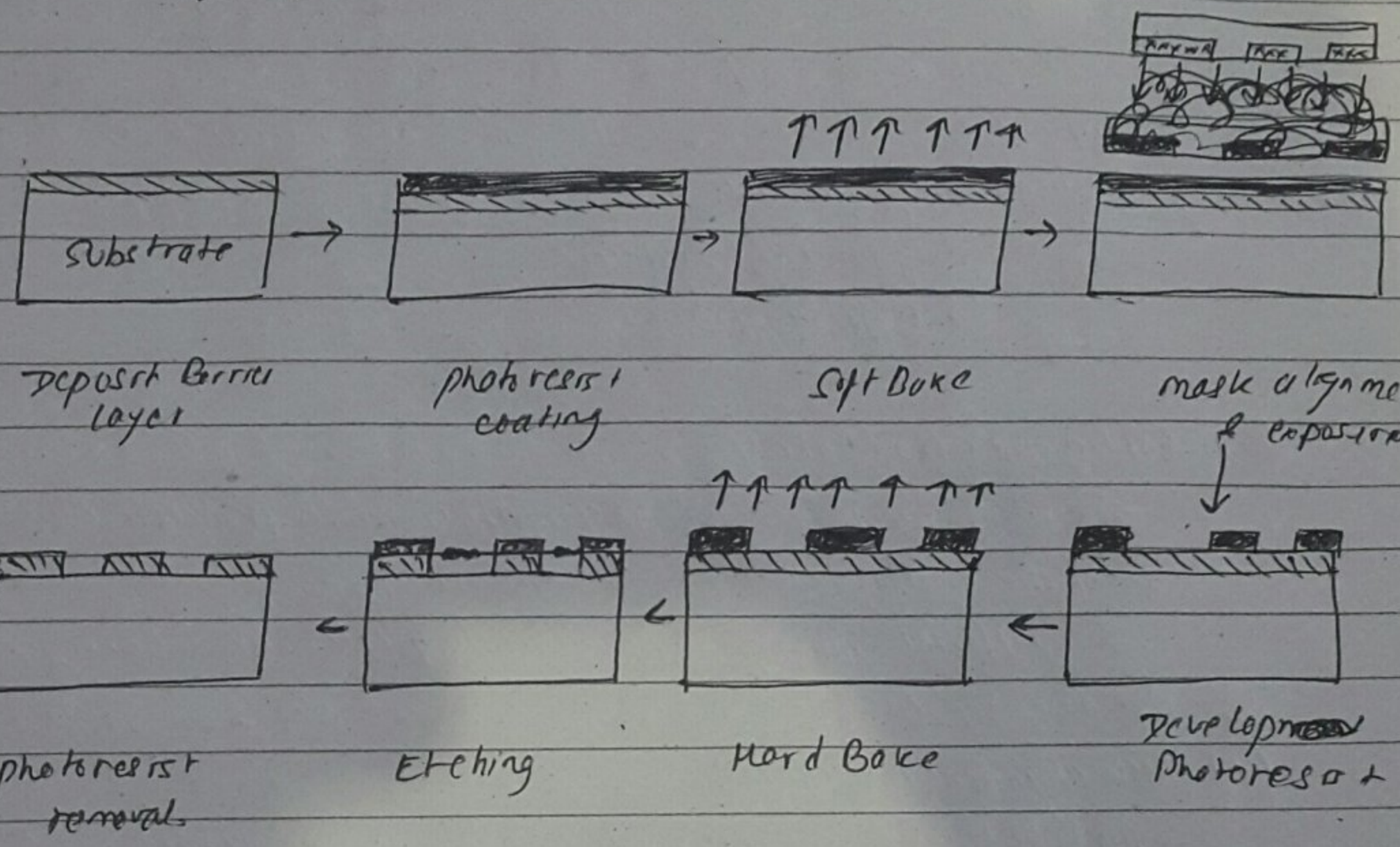
⇒ In semi custom IC technology, designer does not required to create full custom layout rather connects the pre fabricated building blocks. The use of chip with pre-existing gate will lessen the ~~dsg~~ design work of layout & mask creation. So the NRE cost is reduced while time to market is really fast as compared to full custom technology. But however, there will be a reduction in performance in term of power, size & speed. It is referred as Application specific Integrated Circuit (ASIC). They can be divided into two types:

(i) Gate Array semi custom IC technology: In a gate array IC technology, a chip with arrays of predesigned logic gates is utilized to implement the design circuit

(ii) Standard Cell semi Custom IC technology: In standard Cell semi Custom IC technology, functional blocks, which are also called cells with known electrical characteristics are utilized in the design to achieve very high gate density & good electrical performance

⇒ Photolithography is the process which transfers a pattern from mask to a light-sensitive Chemical photoresist on the substrate. It uses optical radiation to create pattern of complex circuit on a ~~wafer~~ ~~surface of silicon~~ or semiconductor

wafer (a piece of silicon or other semiconductor material designed in the form of a very thin disc). The various steps involved in photolithography are described below.



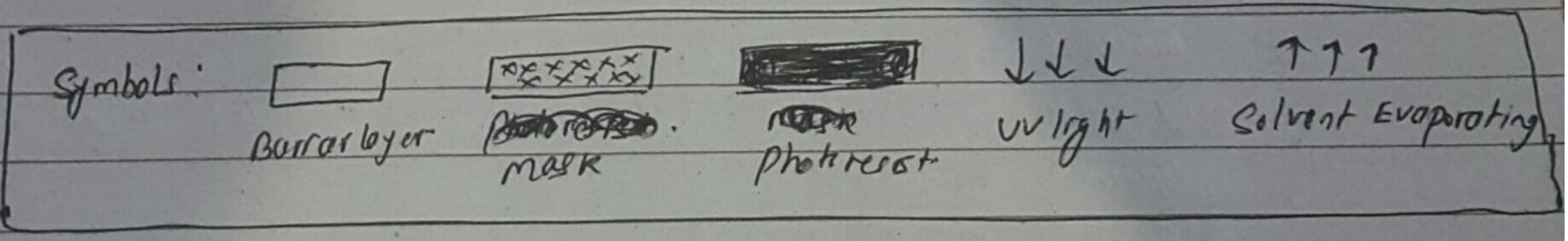


Fig: Various steps of photolithography process

① Deposit Barrier Layer:

→ Barrier layers are material which are required to be laid on the substrate

→ It may be Silicon dioxide, Silicon nitride etc

→ Different methods can be used for barrier formation such as thermal oxidation, chemical vapour deposition etc

(ii) Photoresist Coating:

→ Photoresist substance is a substance which changes its characteristics when exposed to UV light.

→ Before photoresist is coated, hexamethyldisilazane (HMDS) is used on the surface to improve adhesion.

→ After that liquid photoresist is coated over barrier layer using Spin coating method.

→ Types → positive photoresist
→ negative photoresist

(iii) Soft Bake or preBake:

→ The process of heating the wafer which removes solvent from photoresist.

→ Different Baking methods are hotplate, Oven baking etc.

(iv) Mask alignment & exposure:

→ Mask is simply a opaque plate with holes to pass UV rays. It contains pattern to be formed on wafer.

→ Mask is aligned with wafer accurately with the help of Special device called steppers.

→ Once the mask has been precisely aligned, the photoresist is exposed through the pattern on the mask with a controlled amount of UV light.

→ Exposure will caused positive photoresist to become soluble whereas negative photoresist to becomes insoluble.

→ Three method of exposure: Contact, proximity, projection.

(V) Develop photo resist:

→ Barrier layer is exposed when soluble photoresist is chemically washed away using developer solution.

(VI) Hard Bake or Post Bake:

→ Hard bake is used to stabilize and harden developed photoresist.
→ It improves the adhesion of photoresist & also removes the traces of ~~coating~~ Coating Solvent or developer solution.

(VII) Etch window in Barrier layer:

→ As hardened photoresist doesn't shield all part of barrier layer, etching method is implemented to remove the barrier layer which was left uncovered
→ Two methods of etching: wet etching & dry etching (chemical etching) (Plasma etching).

(VIII) Remove photoresist:

→ Finally the remaining photo resist is stripped from the surface exposing the required barrier layer.
→ Photoresist can be removed by using solvent strippers which causes the resist to swell & lose adhesion from substrate
→ Hence finally the pattern is created on surface of substrate.

9) Explain different configuration for seven segment display. write an assembly program to design a down counter that count from 99 to 00.

⇒ The different configuration for Seven Segment Display are:

~~(i) [illegible] configuration:~~

(i) Common anode Configuration: Anodes of all LEDs are connected together to form a common pin which must be connected to high logic voltage

(ii) Common cathode configuration: Cathodes of all LEDs are connected together to form a common pin which must be connected to low logic voltage.

⇒ An assembly program to design a down counter from 99 to 00:

```
ORG 00H
MOV P2, #00H
MOV R6, #09H      ; Counter for lower byte
MOV R7, #09H      ; Counter for upper byte.
MOV R5, #07H      ; to control speed of counter.
MOV P3, #00H.
MOV DPTR, #LABEL1; Loads the starting address of hex
                  ; code list to DPTR
MAIN:
    MOV A, R6
    SETB P3.0     ; Activates 2nd display to display lower byte
    CLR P3.1      ; deactive 1st display
    LCALL DISPLAY
```

```
"LCALL DELAY
MOV A,R7
SETB P3.1 ; activates 1st display
deactivate.
CLR P3.0 ; deactive the 2nd display
LCALL DISPLAY
LCALL DELAY
DJNZ R5, MAIN ; repetition of same display to control speed
MOV R5, #07H
DEC R6 ; decrease value of R6
CJNE R6, #-1, MAIN ; Compare unless R6 become less than zero
MOV R6, #09H.
DEC R7
CJNE R7, #-1, MAIN
MOV R7, #09H.
STMP MAIN.
DISPLAY:
MOV C A, @A+DPTR ; Load HEX value to accumulator from memory.
MOV P2, A ; Sending HEX value to Seven Segment
; through P2

RET
DELAY:
MOV R3, #0F0H
DEL1: MOV R2, #0FAH
DEL2: DJNZ R2, DEL2
DJNZ R3, DEL1

RET
LABEL1: ; represent storing address of HEX value 1
DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 6FH.
END
```

10. Using structural model, write a Code in VHDL to implement Full adder.

⇒ ~~Using structural model~~, we can implement Full adder using two half adders & an OR gate.

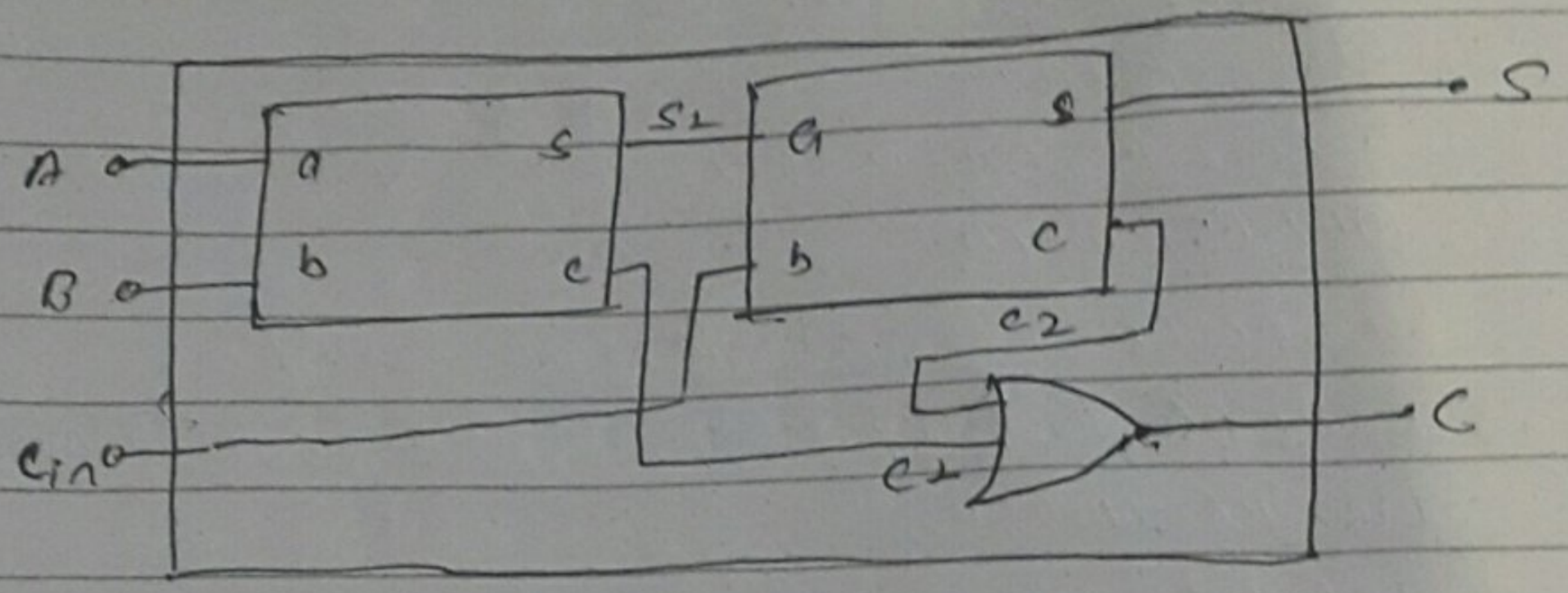


fig: Full adder

VHDL code for OR gate:

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
entity OR_Gate is
    Port ( x,y : in STD_LOGIC;
           z : out STD_LOGIC);

end OR_Gate;

architecture OR_Gate_arch of ORGate is
begin.
        z <= x or y;
end OR_Gate_arch;
```

VHDL code for Half Adder ;

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Half_Adder is
Port( a, b ; in STD_LOGIC;
      S, C ; out STD_LOGIC);
end Half_Adder;

architecture Half_Adder_arch of Half_Adder is
begin.
        ~~S <= a xor b;~~
        ~~C <= a and b;~~

~~architecture~~ process
  Process_adder : process(a,b)
  begin.
        S <= a XOR b ;
        C <= a AND b;
    end process process_adder
end Half_Adder_arch ;
```

VHDL Code for full adder using two half adders & OR gate :

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL
entity FULL_ADDER is
Port( A, B, Cin : in STD_LOGIC;
      S, C : out STD_LOGIC);

end FULL_ADDER;
```

Architecture structural of FULL_ADDER is

```
architecture FULL_ADDER_arch of FULL_ADDER is
    Component OR_Gate is
        Port ( x, y : in STD_LOGIC;
               z : out STD_LOGIC);
    end component OR_gate;

    Component Half_Adder is.
      Port ( a,b; in STD_LOGIC;
             S,C; out STD_LOGIC);
    end component Half_ADER;
    Signal S1, C2, C1 : STD_LOGIC
    begin
        HA1 : Half_Adder port map (a => A, b => B, S => S1, C => C1);
        HA2 : Half_Adder port map (a => S1, b => Cin, S => S, C => C2);
        OR1 : Or_Gate port map (x => C1, y => C2, z => C);

    end FULL_ADDER_arch;
```