UNIT - I

Embedded Systems and its applications :→

① An embedded system is a system that has Software embedded into Computer-hardware, which makes a System dedicated for an application or Specific part of an application or Product or part of a larger System

② It is any device that includes a programmable Computer but is not itself intended to be a general Purpose Computer.

③ Embedded System are the electronic Systems that Contain a microprocessor or a microcontroller, but we do not think of them as computers - the Computer is hidden or embedded in the system

Three main embedded Components →

① Embeds hardware to give computer like functionalities

② Embeds main Application Software generally into flash or Rom and the application Software performs Concurrently the Numbers of tasks.

③ Embeds a real time operating System (RTOS), which Supervises the application Software tasks running on the hardware and organizes the accesses to system Resources according to priorities and timing

Constraints of tasks in the Systems.

Embeded System RTOS →

→ Enables execution of Concurrent processes or threads or tasks

→ Provides a Mechanism to let the processor run each process as per Scheduling and to do context-switch B/W the various processes (Threads & tasks)

→ RTOS Sets the rules during execution of applications processes to enable finishing of a Process within the assigned time interval and with assigned priority.

Real Time operations → Defines the ways in which the system works, reacts to the events and interrupts, schedules the system functioning in real time and executes by following a plan to control the latencies and to meet the deadlines [Latency - Waiting interval B/W the instance at which a need to run the codes arises for tasks (or interrupt service routine) following an event and instance of start executing the codes]

Multi rate operations → Different operations may take place at distinct rates. for example the audio, video, Network data or stream and events have the different Rates and time constraints to finish associated processes.

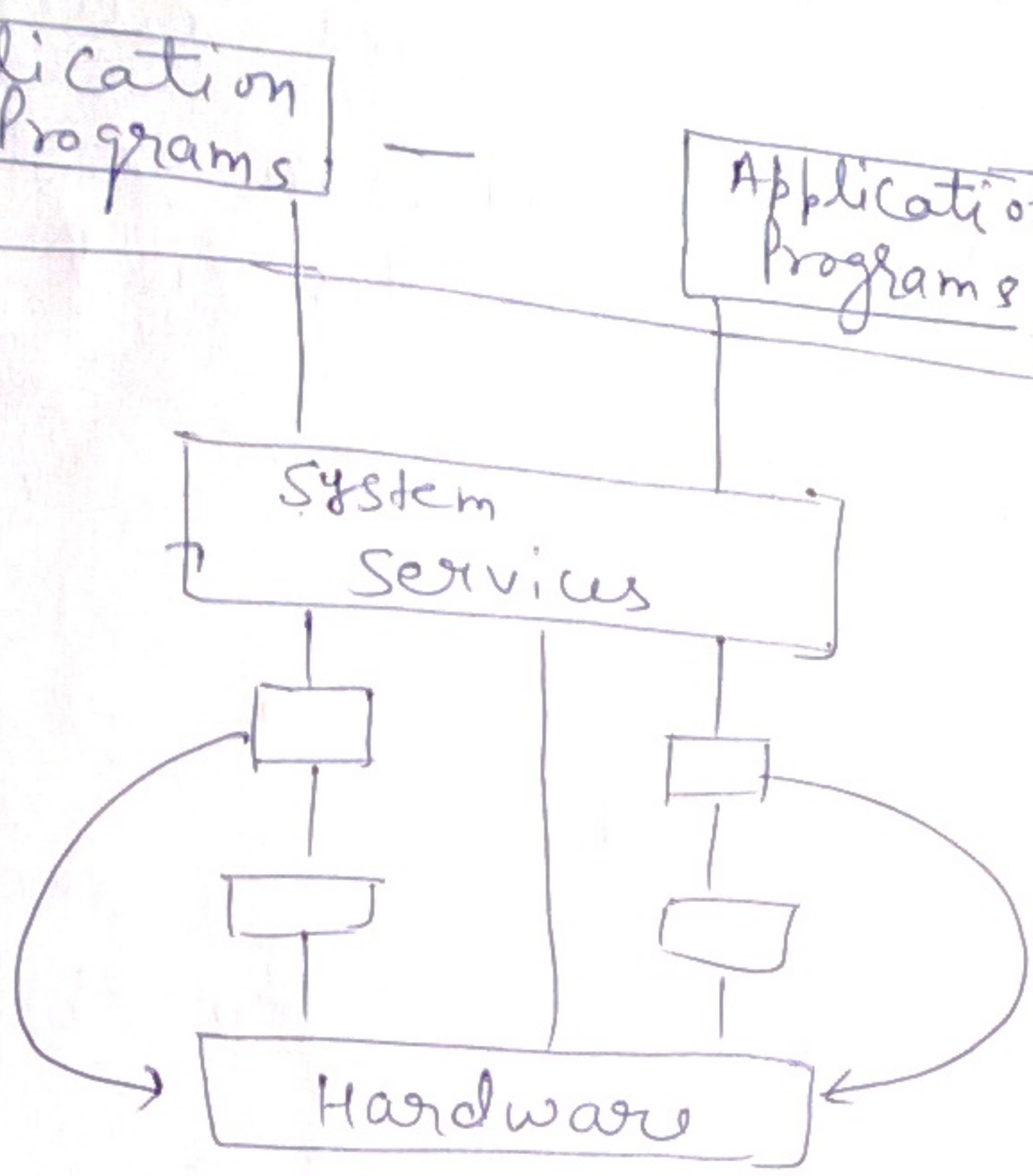
Embedded operating System →

→ As the complexities in the embedded applications increase, use of an operating System brings a lot of advantage.

→ A Simple Microwave oven does not require an operating System. But as the Complexity of applications expands beyond Simple tasks the Benefits of an operating System far outweights the associated cost. Since embedded systems (PDAs, cell phones, VCRs, industrial robot control, or even the toster) are becoming more complex hardware-wise with every generation and more features are put into them in each iteration. So Some operating System is always Required to Simplifies the complexity

→ Real Time operating system allows real time application to be designed and expanded easily. function can be added without major changes to the software.

① Operating System Architectures →



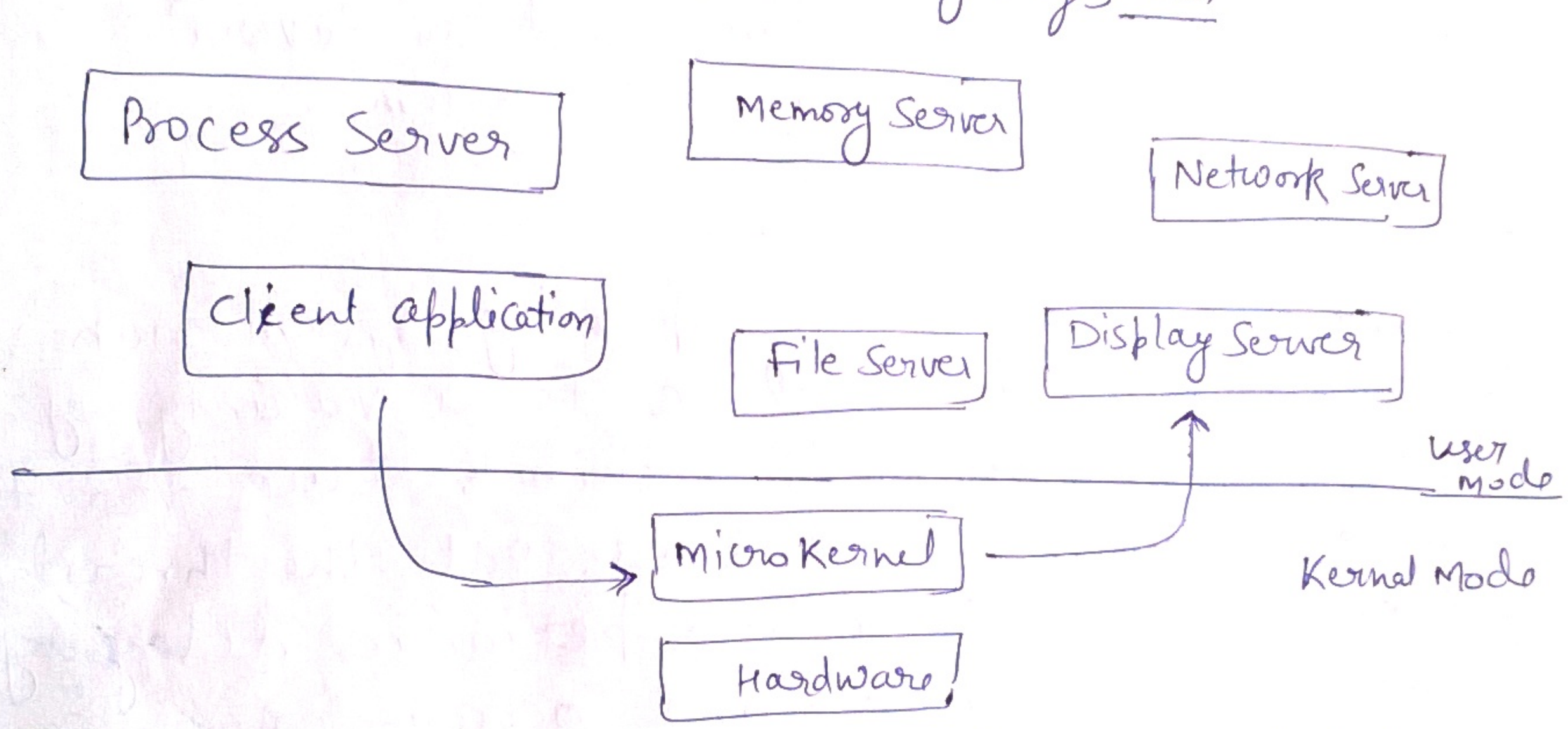
Monolithic operating System

→ In this case the OS is just one piece of code composed of different Modules. One Module calls the other in one or more way. Here More the Modules, more will be the interconnections and more complex the Software becomes

② Layered Operating System →

→ This is better approach compared to a Monolithic OS. A system call goes directly to each individual layer. Here an application can access the BIOS or even the hardware

(3) Client-Server operating system



→ The basics of OS limited to a strict minimum (scheduler and Synchronization Primitive) and all other functionality is on another level implemented as system threads or tasks.

Some of the Popular RTOSs are

(1) QNX RTOS v 6.1 → The QNX RTOS v 6.1 has a client-server based architecture. QNX adopts the approach of implementing an OS with a 10 K bytes micro-kernel surrounded by a team of optional processes that provide

higher-level OS services. Every process including the device driver has its own virtual memory space. The system can be distributed over several Nodes and is N/W transparent. The system performance is fast and predictable and is robust.

QNX has successfully been used in tiny ROM - based embedded systems and in several hundred node distributed systems.

(2) Windows CE 3.0 –

window CE 3.0 is an operating system rich in features and is available for a variety of hardware platforms. it exhibits true real time behaviour most of the times. But the thread creation and deletion has periodic delay of more than 1 millisecond occuring every second.

The system is complex and highly configurable The configuration of CE 3.0 is a complicated process. The system is robust and no memory leak occurs even under stressed condition

(3) pSOSystem/x86 2.2.6 →

pSOS+ is a small Kernal suitable for embedded applications. This uses the software bus to communicate Between different modules. the choice of module to be used can be done at compile time making it suitable for embedded applications.

pSOS has a multiprocessor version of pSOS+, which Can have one Node as Master and a Number of Nodes as Slaves. failure in Master will however lead to System Crash. The Integrated Development Environment is Comprehensive and is available for both windows and UNIX Systems. The draw-back of this RTOS is That it is available Only for Selected processors and that lack of Mutexes in Some Versions leads to priority inversions.

(4) Vx Works (Wind River Systems) →

Vx Works is the Premier development and execution environment for complex real-time and embedded applications on a wide Variety of target processors. Three highly integrated Components are included with Vx Works,

① A high performance scalable real-time operating System which executes on a target processor

② A Set of powerful cross-development tools

③ A full Range of communication Software options such as Ethernet or Serial Line for the target connection to the host

The heart of the OS is the wind microkernel

which supports multitasking, scheduling, intertask management and memory Management.

(5) windows NT —

The overall architecture is good and may be a suitable RTOS for control systems that need a good user interface and can tolerate the heavy recourse requirements demanded for installation. it need hardisk and powerful processor. Configuration and user interaction requires a dedicated screen and Keyboard.

Generally an RTOS for embedded applications Should have the following features →

(1) Open Source
(2) portable
(3) Rom able
(4) Scalable
(5) pre-emptive
(6) multi-tasking
(7) Deterministic
(8) Efficient memory management
(9) Rich in Services
(10) Good Interrupt management
(11) Robust and Reliable.

Design parameters of an embedded system and its significance →

- All Embedded systems contain a processor and Software. The processor may be 8051 micro-controller or a pentium-IV processor (having a clock speed of 2.4GHz). All embedded system contain memory, and inputs and outputs. for Example in a microwave oven the inputs are the buttons on the front panel and a temperature Probe and the output are human readable display and the microwave radiation.

Inputs to the system generally take the form of Sensors and Probes, communication signals, or control Knobs and buttons. outputs are generally displays, communication signals, or changes to the Physical world

The common critical features and design requirements of an embedded hardware are

(1) Processing power → selection of the Processor is based on the amount of processing power to get the job done and also on the basis of register width required.

(2) Throughput → The system may need to handle a lot of data in short period of time.

(3) Response → The system has to react to events quickly.

(4) Memory → Hardware designer must make his best estimate of the memory

Requirement and must make provision for expansion

(5) Power Consumption → Systems generally work on battery and design of both Software and hardware must take care of power saving techniques.

(6) Number of Units → The no of Units expected to be produced and sold will dictate the trade-off Between production cost and development cost.

(7) Expected lifetime → Design decisions like Selection of components to system development cost will depend on how long the system is expected to Run.

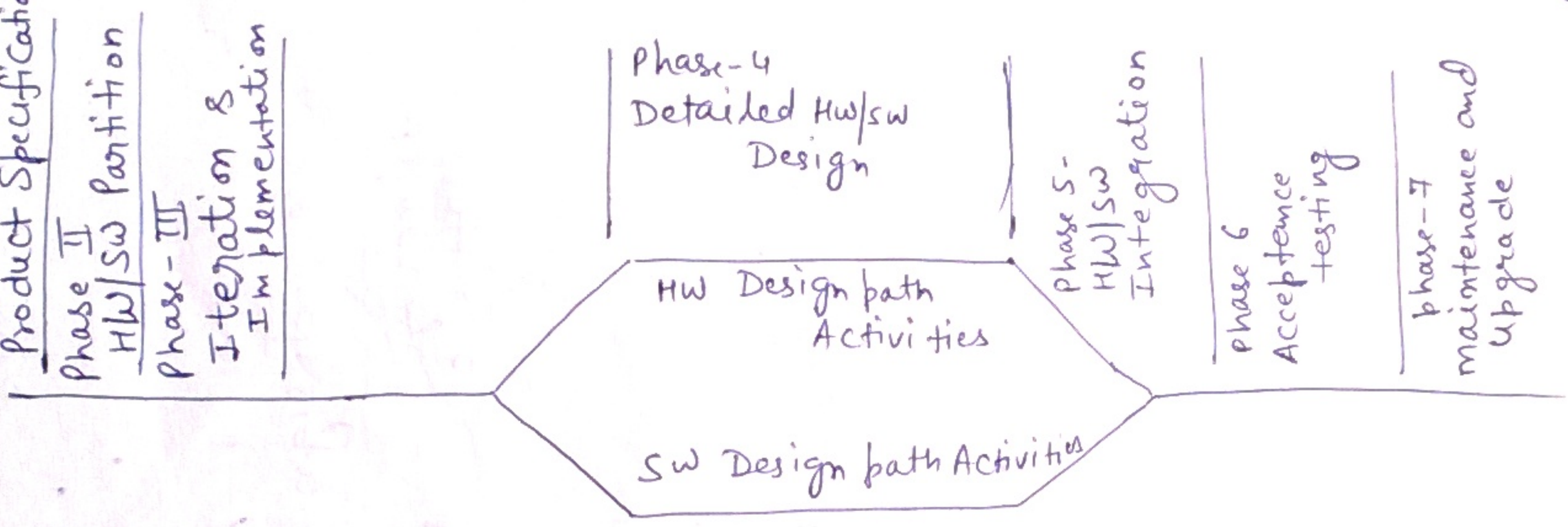
(8) Program Installation → Installation of the Software on to the embedded System needs Special tools.

(9) Testability & Debugability → setting up test conditions and equipment will be difficult and finding out what is wrong with the Software will become a difficult task without a Keyboard and the usual display screen.

(10) Reliability → Is critical if it is a Space shuttle or a car but in Case of a toy it doesn't always have to work right.

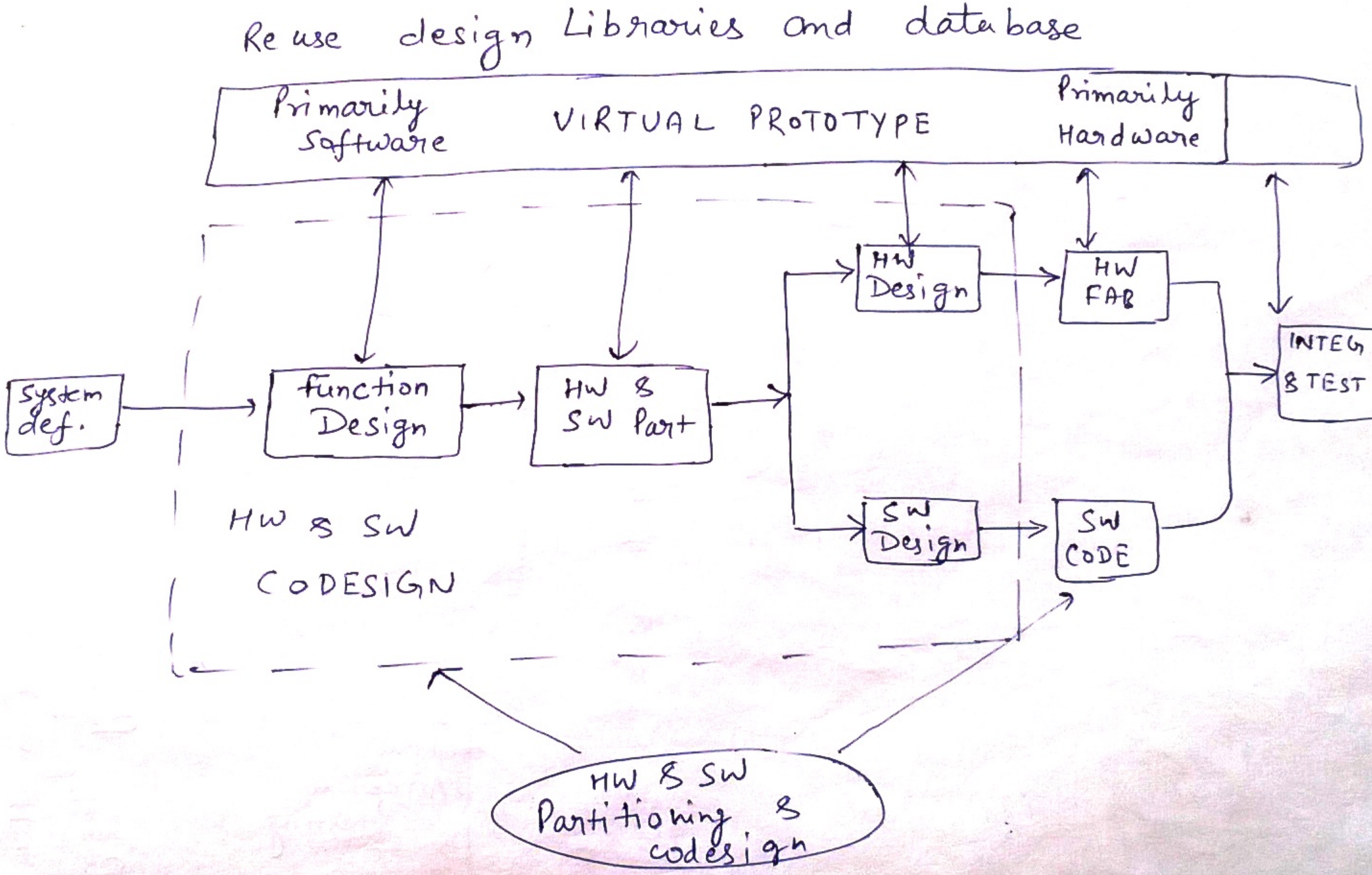
Embedded Design Life cycle →

Unlike the design of a Software application on a standard platform, the design of an embedded System implies that both Software and hardware are being designed in parallel. Although this is not always the case, it is reality for many designs today



Design life Cycle

Hardware and Software partitioning and co-design →



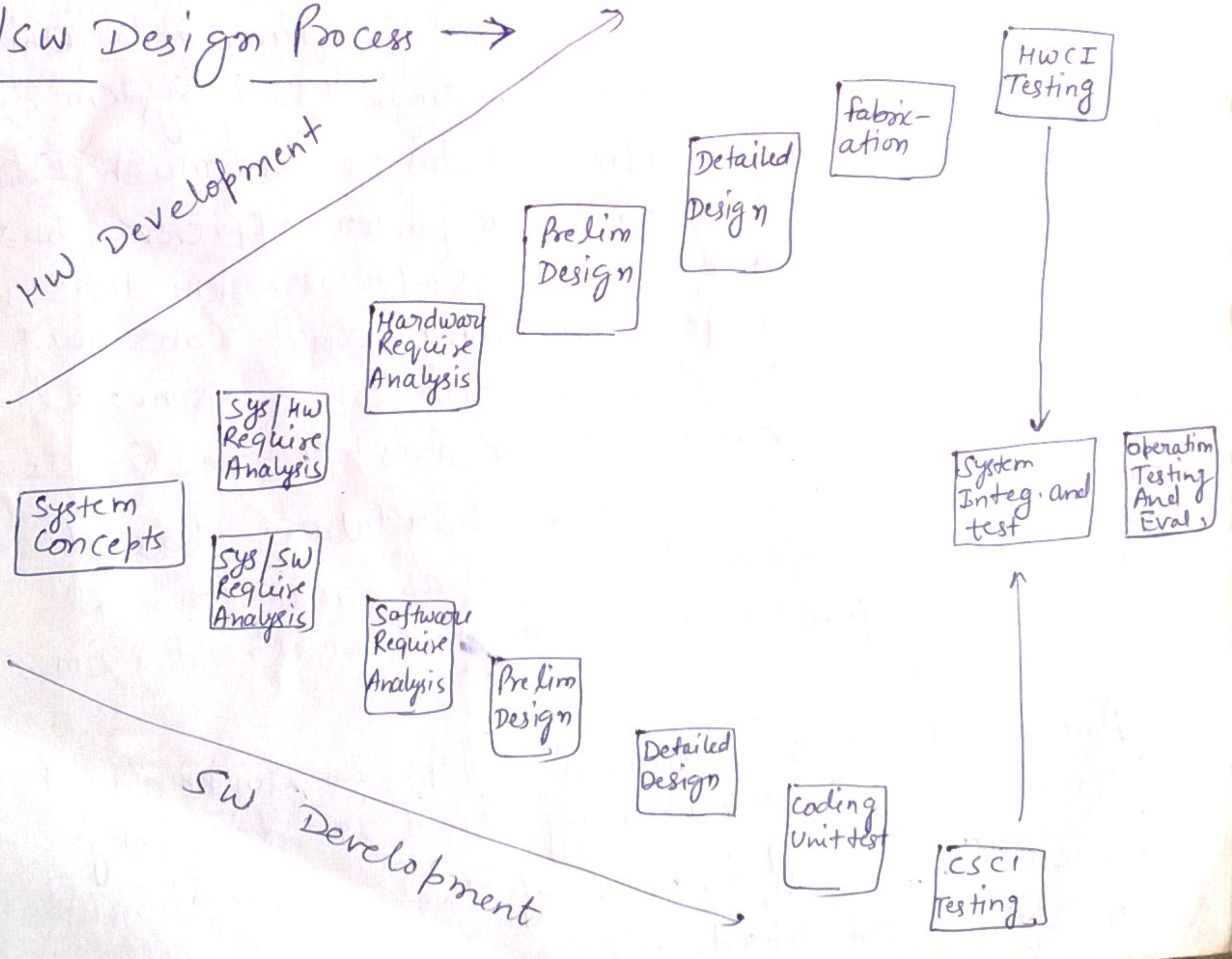
Co-design → The meeting of system-level objectives by exploiting the trade-offs between hardware and Software in a system through their concurrent design

The importance of co-design in designing hardware / Software systems: →

(1) Improves design Quality, design cycle time, and cost, Reduces integration and test time

(2) Takes Advantage of advances in tools and technologies
- processor Cores
- High-level hardware synthesis Capabilities
- ASIC development.

HW/SW Design Process →



Hardware and software partitioning :→

An embedded System is a Computing system rather than desktop computers / Laptops / palmtops.
Typical applications that employ embedded system include fax machines, copiers, printers, scanners, Cash Registers, Alarm Systems, Card readers, Mobile phones, digital Cameras, Washing machines DVD players, speech recognizers and many more
Like typical computer systems, embedded system too Include hardware and Software components, In other words, it is common to use both application-specific hardware accelerator Circuits and general-purpose programmable Units with appropriate software for embedded system design usually application specific hardware is much faster than software and also more power efficient, but expensive at the same time. Software on the other hand is cheaper, but slow and consumes much power when implemented on a general purpose processor. Hence for faster realization or power-critical situations hardware based systems are preferred, whereas non-critical modules of embedded system are realized in Software.

Among the most Crucial steps in embedded System design, partitioning, that is, deciding which components/modules of the system should be realized on hardware and which ones in

Software is a fundamental problem. this is referred to HW/SW partitioning problem in Embedded System. Traditionally HW/SW Partitioning was accomplished manually. However with the increased complexity in embedded system researchers currently prefer an Automatic approach to handle this problem.

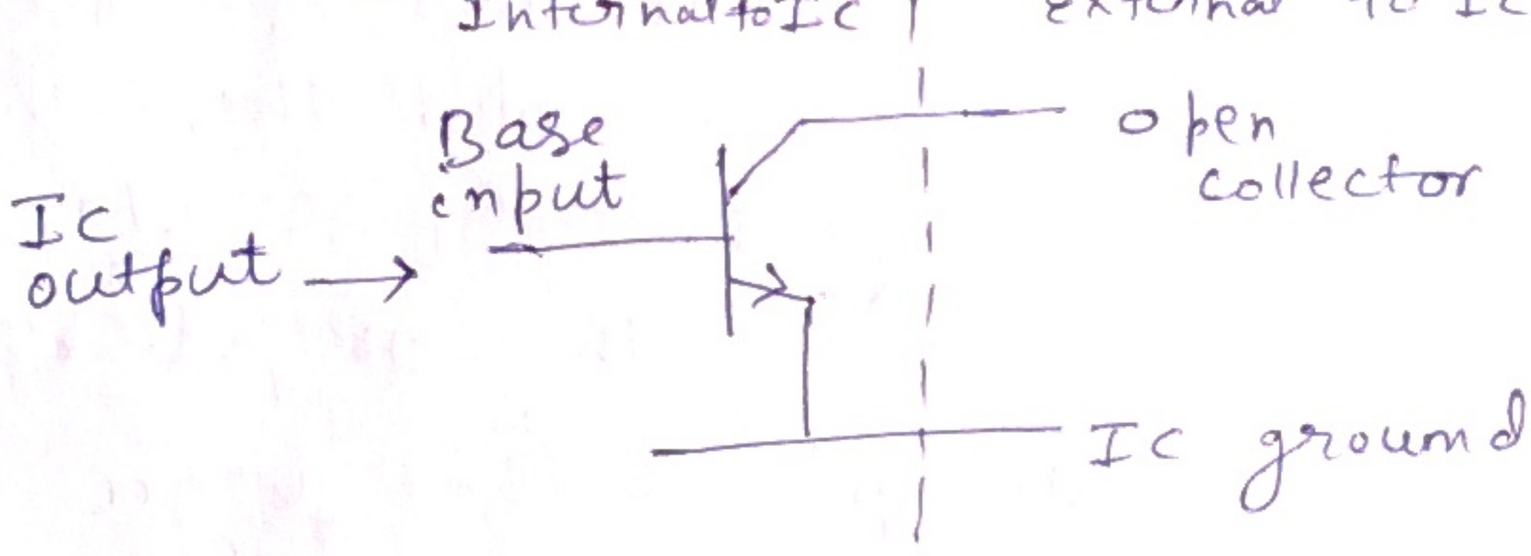
Classically there exist two approaches, exact and heuristic, to handle the HW/SW partitioning Problem.

→ Exact Algorithms include branch, dynamic programming, and integer linear programming

→ heuristic approaches include, Genetic Algorithm Simulated annealing, tabu search Algorithm, greedy Algorithms., Ant colony optimization

Open Collector output →

An open collector is a common type of output found on many integrated circuit (Ic). instead of outputing a signal of a specific voltage or current, the output signal is applied to the base of an internal NPN transistor whose collector is externalized (open) on a Pin of the Ic. The emitter of the transistor is connected internally to the ground Pin. if the o/p device is MOSFET the output is called open drain and it function in a similar way



The output essentially acts as either an open circuit (no connection to anything) or a connection to ground The output usually has an external pull-up resistor, which raises the output voltage when the transistor is turned off. when the transistor connected to this resistor is turned on, the output is forced to nearly '0'. open-collector output can be useful for analog weighting, summing limiting etc.

→ open collector output is Normally used in logic circuits and can be viewed as a common emitter configuration for a BJT transistor and Normally of Type NPN

→ A typical common emitter configuration requires the emitter to be connected to the ground and a resistor ~~Rcc~~ R_c from V_{cc} to the collector. when the transistor becomes forward biased, the collector will Pulled the voltage accross the resistor down, from approximately V_{cc} down to almost 0V, normally in the order of .1V at the collector.

→ we can see that the output at the collector will swing B/w V_{cc} and ground. In open collector output the collector resistor R_c and V_{cc} have been left out. This enables the engineers to use any voltage amd any pull-up resistor that will meet the specification of the output resistor.

→ Normally it is acceptable in logic circuits that multiple inputs may be connected to one o/p but not Multiple outputs to one input due to the Nature of active outputs. when open collectors however multiple outputs may be connected to one input without damaging the device.

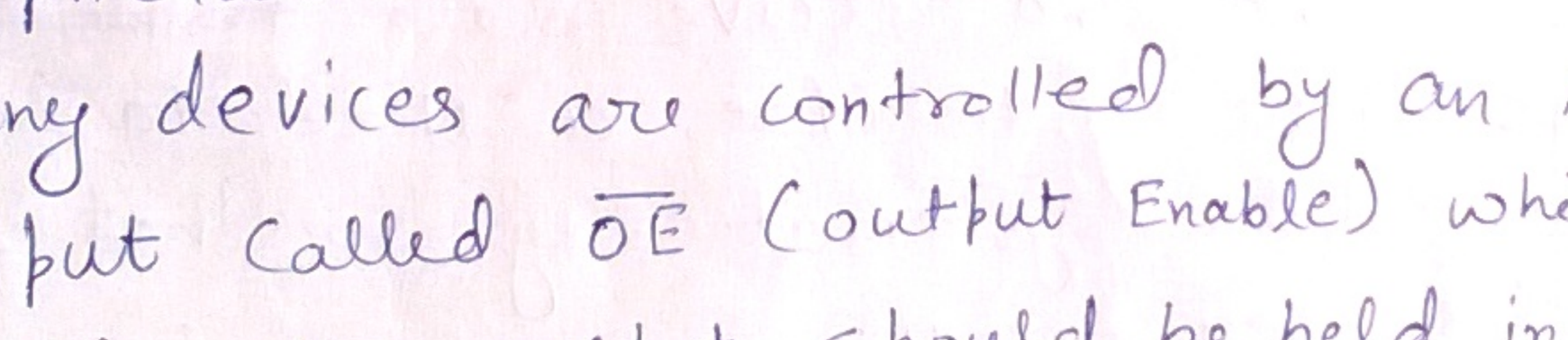
→ open collector can be used to convert 5V logic to 8V or 12V logic or down to 3.3V since the logic high will be determined by the external supply to the Pull-up resistor

Tri state outputs →

In digital electronics three-state, tri-state or 3-state logic allows an output port to assume a high impedance state in addition to the 0 and 1 logic levels, effectively removing the output from the circuit. This Allows Multiple Circuits to share the same output line or lines (such as bus).

Three-state outputs are implemented in many registers, bus drivers, and flip-flops in the 7400 and 4000 series as well as in other types. but also internally in many integrated circuits other typical uses are internal and external buses in microprocessors, memories and peripherals.

many devices are controlled by an Active-low input called $\overline{OE}$ (output Enable) which dictates whether the output should be held in a high-impedance state or drive their respective loads (to either 0 or 1 level)



A tristate buffer can be thought of as a switch. If B is on the switch is closed. if B is off, the switch is open.

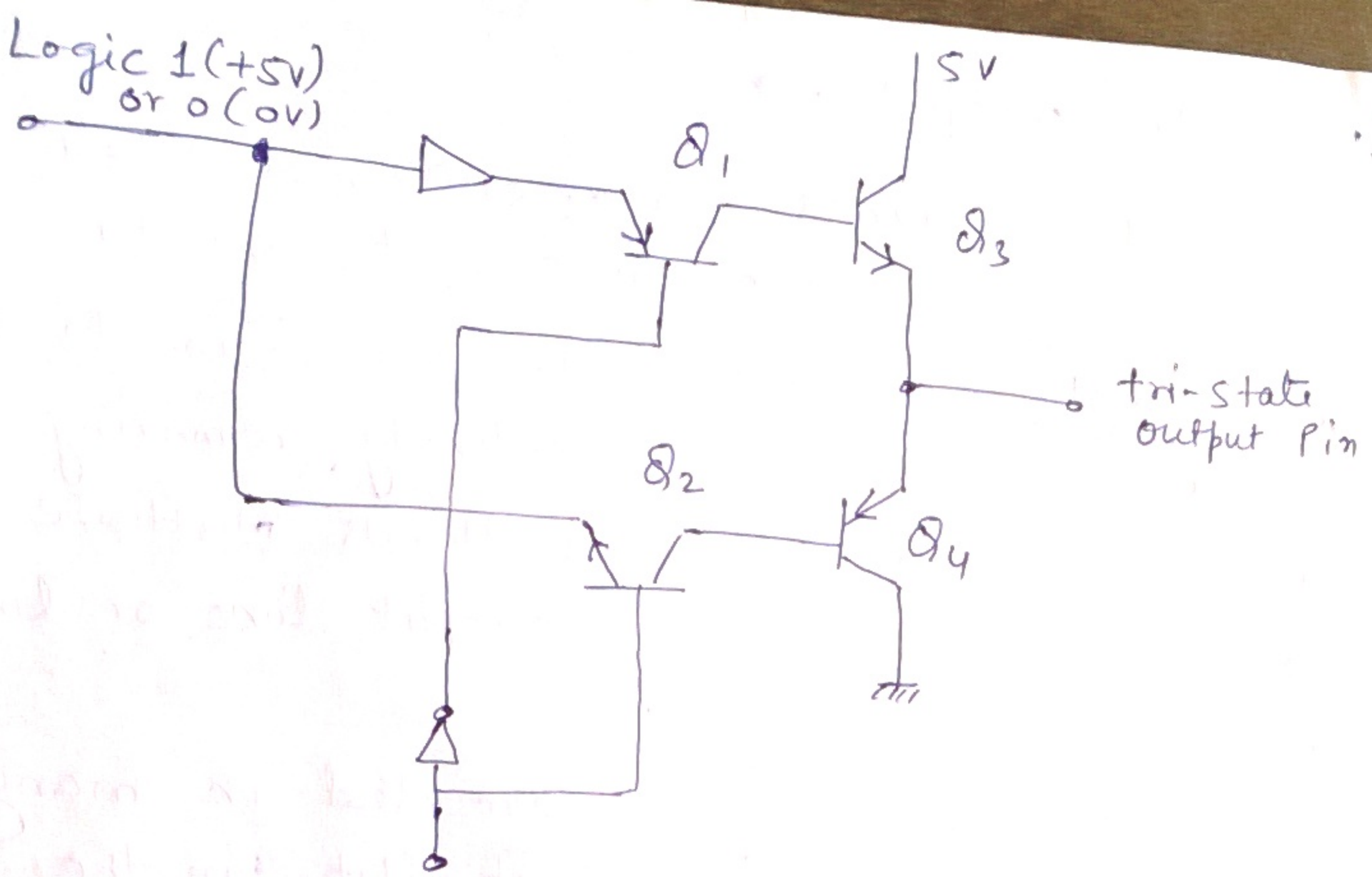


Diagram for a tri-state output

→ Digital circuits often use circuit path and Nodes that are shared by different IC's Connected to them. for execemble memory chips share the same data buses to send and receive data ~~fromoth~~. from other chips. There is, therefore, a Need to let these chips share data systematically so that No data clashes ever occur on any bus. i.e No conflicting data are Put on the same bus at the same time. this is achieved by employing what is known as the tri-state output.

→ ~~A tri state~~ The ckt shows how a digital tri-state output circuit may be implemented if the 'enable' Pin is at logic '0' the base emitter and base-collector junction of both Q_1 and Q_2 are reverse-biased regardless of what the inputs logic state is which ~~pri~~ prevent

PLD's → A programmable logic device is a circuit which can be configured by the user to perform a logic function. Most "standard" PLDs consists of an AND array followed by an OR array either (or both) of which is programmable.

Inputs are fed into the AND array which performs the desired AND functions and generates product terms. The product terms are then fed into the OR ~~Create~~ array. In the OR array, the outputs of the various product terms are combined to produce the desired outputs.

base currents from flowing in both Q_3 and Q_4 this effectively turns off Q_3 and Q_4. Putting the output pin in high-impedance mode.

If the enable pin is at logic 1 and the input is at logic '0' Q_2 conducts and pull Q_1's base voltage to '0' causing Q_4 to conducts. At the same time, Q_1 cuts off and prevents Q_3 from turning 'on' with Q_3 off and Q_4 conducting the outfput is pulled to ground. In short the logic '0' at the input appears at the output.

If the enable pin is at logic '1' and the input is at logic '1' Q_2 turns off which also turns off Q_4. At the same time, Q_1 conducts and causes current to flow into Q_3's base, turning it 'on' with Q_3 'on' and Q_4 'off'. the output is pulled to +5V. In short, the logic '1' at the input appears at the output.

Watchdog timers → A watch dog timer can be thought of as having the inverse functionality that than that of Regular timer. we configure a watchdog timer with a real-time value, just as with a regular timer.

However instead of a timer generating a signal for us every X time units, we must generate a signal for the timer every X time units if we fail to generate this signal in time, than the timer generates a signal indicating that we failed. we often connect this signal to the reset or interrupt signal of a general-purpose processor. thus a watchdog timer provides a Mechanism of ensuring that our software is working properly. every so often in the software we include a statement that generates a signal to the watchdog timer (in particular, that resets the timer). if something undesired happens in the software (eg. we enter an undesired infinite loop, we wait for an input signal that never arrives, a part fails etc). the watchdog generates a signal that we can use to restart or test parts of the system. Using an interrupt service routine we may record information as to the Number of failures and the cause of each, so that a service technician may later evaluate this information to determine if a particular part

Requires Replacement.

freqscale's 6811
Intell 8051
Zilog's z8
PIC 16X

} Unique Instruction set and Register set

8 bit

	8051	8052	8031
ROM	4K bytes	8K bytes	0K bytes
RAM	128	256	128
Timers	2	3	2
I/O Pins	32	32	32
Serial port	1	1	1
Interrupt Sources	6	8	6

---

versions of 8051 from Atmel (All Rom flash)

	ROM	RAM	I/O	Timer	Interrupt	5V
AT89C51	4K	128	32	2	6	5V
AT89LV51	4K	128	32	2	6	3V
AT89C1051	1K	64	15	1	5	5V
AT89C2051	2K	128	15	2	6	5V
AT89C205L	8K	128	32	3	8	5V
AT89LV52	8K	128	32	3	8	3V