SOC (System On Chip)

Is an Integrated circuit, integrates all or most components of a electronic system on a single chip.
Contains analog, digital, mixed signal, radio frequency functions lies on a single substrate.
Consists of,
Control unit : The major control units are microprocessors, micro-controllers, digital signal processors etc.
Memory Blocks : ROM, RAM, Flash memory and EERPROM are the basic memory units.
Timing Units : Oscillators and PLLs are the timing units.
Other peripherals of the SOCs are counter times, real timers and power on reset generators.
Analog interfaces, external interfaces, voltage regulators and power management units form the basic interfaces

Data Alignment

The CPU in a computer performs reads and writes to memory most efficiently when data is naturally aligned.

What is the meaning of aligned?

The data's memory address is a multiple of the data size.

Example:

In a 32-bit architecture, the data may be aligned if the data is stored in four consecutive bytes and the first byte lies on a 4-byte boundary.

Direct Memory Access(DMA)

DMA refers to the ability of an I/O device to transfer data directly to an from memory without going through the CPU.
Contrast DMA with programmed I/O where the CPU explicitly copies data using loads and stores.

Why DMA?
i) Higher transfer bandwidth
ii) CPU can do other things during transfer
iii) Faster, more consistent response times(no interrupt or polling overhead)

Basic DMA Example
DMA-capable I/O device will have control registers for:
i) starting address of memory buffer
ii) number of bytes of transfer
iii) transfer direction (to/from device, read/write on bus)
iv) device-specific control (e.g., head, cylinder, sector for disk access)

DMA transfer algorithm:
1. Program makes I/O request to device
2. CPU does initiation routine (ex: part of device driver)
i) use programmed I/O to set up control regs
ii) device-specific parameters
iii) DMA parameters (buffer address/length)
iv) last write: enable bit
3. I/O device interface does transfer
i) arbitrate for bus mastership
ii) put address on bus, assert control signals to do read/write(looks just like CPU to memory)

a) likely use burst transfer if available
b) size/type may be programmable via control reg
iii) I/O device supplies/consumes data
iv) increment address, decrement byte count
v) if byte count > 0, repeat
a) release bus to give other devices a chance
vi) if byte count =0, set completion bit in status reg, generate interrupt
4. CPU ISR runs completion routine
i) check for errors, retry if necessary
ii) notify program that transfer is done (e.g. set flag variable in memory)
iii) set up next transfer if appropriate(call initiation routine)
5. Program notices that request is complete
i) main program may do other things during steps 2,3,4
ii) on multitasking OS, another program may be run
iii) on DMA write to I/O device, program must be careful not to re-use buffer until it's notified that the write is complete

DMA Controllers
In modern systems, devices that need DMA typically have their own DMA control engine built in. In older/low-end systems (like the IBM PC), the logic required was/is a significant burden, so often a stand-alone, shareable "DMA controller" device is provided that lets several simpler I/O devices to DMA transfers.
The MPC823 has a couple of DMA controllers that are shared among the on-chip peripherals and can also be used by simple off-chip peripherals.

Basic idea (single I/O device):
i) DMAC has address, direction, count registers
ii) A pair of request/acknowledge signals go directly to & from actual device
iii) Initiation routine initializes both DMAC and device
iv) For each data transfer on bus:
a) Device uses DMA req line to tell DMAC it's ready for transfer
b) DMAC arbitrates for bus, puts out address, uses ack line to tell device when to put data on/take data from bus
c) note that device is doing a read or write under control of the DMAC without looking at bus address (which actually corresponds to memory location)
d) DMAC interrupts CPU when done

Multiple devices:
1. DMAC has a pair of req/ack lines for each device (say 4)
2. Also has a set of control regs (address, dir, count) for each device
3. Software can treat like 4 separate DMACs, initialize devices & DMAC reg sets independently
4. DMAC looks at all req lines, on mult requests chooses one device to do transfer (prioritized or round-robin)
5. Each of these sets of registers, req/ack lines is called a DMA "channel"

IBM PC had 4, used 0 for DRAM refresh (timer, highest pri.) PC/AT extended to 7.

This is called a “one-cycle” or “fly-by” DMA transfer (same as device w/built-in). Stand-alone DMACs are usually capable of two-cycle transfers as well, where initialization routine specifies two addresses, etc. (for devices too dumb to deal with DMA req/ack lines).
Note that this is not much faster than programmed I/O (but still may have a faster response time and frees the CPU).
This feature can also be used for mem-mem copies (but rarely is).

Master and Slave
Master : Has ability to control the bus, initiates the transaction
Slave : Module activated by the transaction

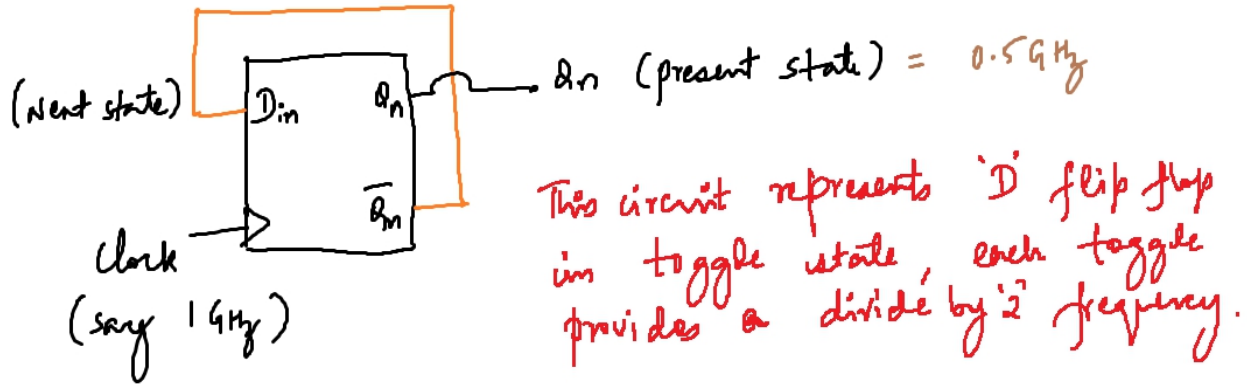
Asynchronous Bus Transfers
Control lines (req, ack) serve to orchestrate sequencing
Synchronous Bus Transfers
Sequence relative to common clock

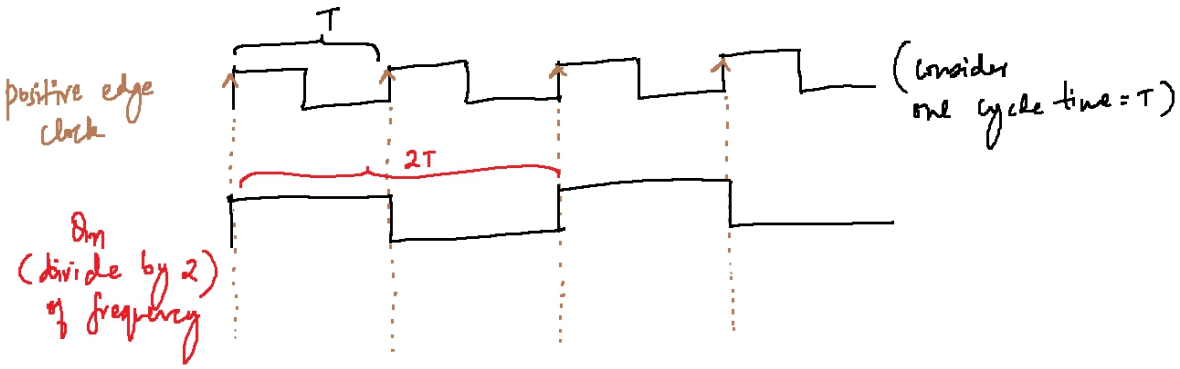
Big Endian Byte Order
The most significant byte (the “big end”) of the data is placed at the byte with the lowest address. The rest of the data is placed in order in the next three bytes in memory.
Little Endian Byte Order
The least significant byte (the “little end”) of the data is placed at the byte with the lowest address. The rest of the data is placed in order in the next three bytes in memory.

Clock Divider Circuit
Purpose: To produce an output frequency which is a divide by frequency of the input frequency.

Why?
Most of the circuit applications which are digital and analog in nature are controlled by clock signals. Many of these circuits require not just one clock frequency but more than one. Most of these clock frequencies are related to each other with a divide by factor.

How to make a clock divider?
The clock divider circuit is usually divided by 2 counters. This circuit employs a Flip-flop in toggle mode. Below circuit with waveforms provides a glimpse of this circuit and its output.




T
positive edge clock
(consider one cycle time = T)
2T
Qm (divide by 2) of frequency