Lecture 3: Buses

Computer Architecture

Roy Vellaisamy
Professor of Intelligent systems

https://www.gla.ac.uk/schools/engineering/staff/royvellaisamy/

BUSES?

1956,first domestic trolleybus

1957, first domestic combustion engine bus

Buses in 1970s

1975, articulated (extended) buses

1990s, articulated (extended) buses

2001, natural gas buses with air conditioning

2008. battery-electric buses

2014, electric buses

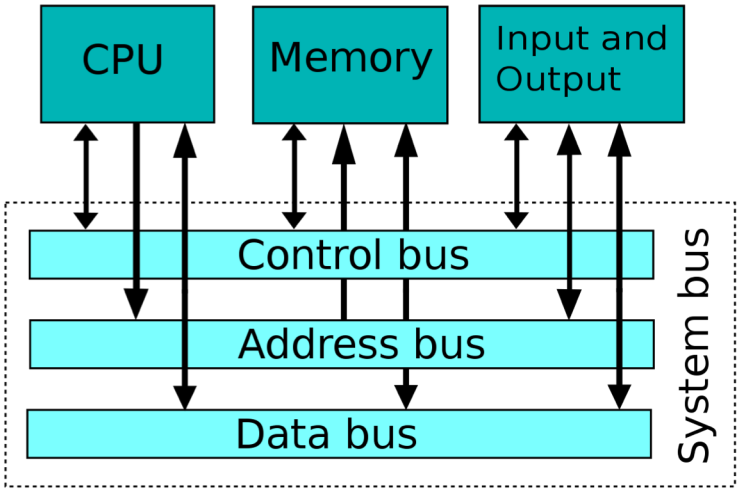
What is a "Bus"?

Bus is a communication system that transfer data between components inside a computer, or between computers / devices. This cover all related hardware and software, including communication protocols.

Address bus - carries memory addresses from the processor to other components such as primary storage and input/output devices.

Data bus - carries the data between the processor and other components.

Control bus - carries control signals from the processor to other components, including the clock's pulses.



Some well-known buses:
USB, CAN, Thunderbolt, HDMI, VGA, GPIB, RS-232, SCSI….
, I2C, SPI, PCI, VESA, PCI….

Note:
In Power system, the "bus" is the power cable which connected to generators, loads, … etc

Bus structure

- Address bus and Data bus are mainly parallel buses while control bus can be considered as a bunch of discrete wires
- Number of address lines indicates the capacity of memory the CPU can address directly
- Number of data lines normally in multiple of 8 (1byte has 8bit). Larger number means higher computation power
- Control bus include all functions other than address and data. E,g. Enable, acknowledge, chip select…
- Inside CPU is designed at semiconductor level, cannot alter once the chip has been made
- PCB is designed by electronic engineer who need good hardware design knowledge and possible to modify the configuration for specific purposes

Bus Arbitration

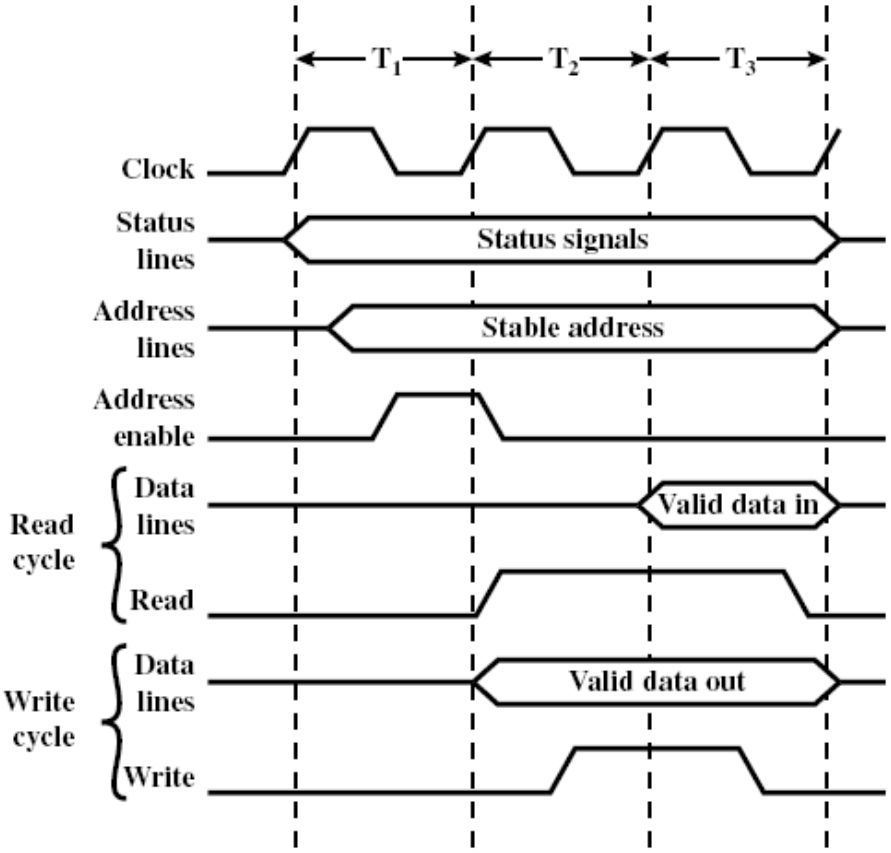
- More than one module can control the bus. E.g. CPU and DMA (Direct Memory Access) controller
- However, only one module can control the bus at one time
- Arbitration may be centralized or distributed

- Method of Arbitration
  - Centralized
    - Single hardware device controlling bus access
    - May be part of CPU or separated
  - Distributed
    - Each module may claim the bus
    - Control logic regulates the operation on all modules

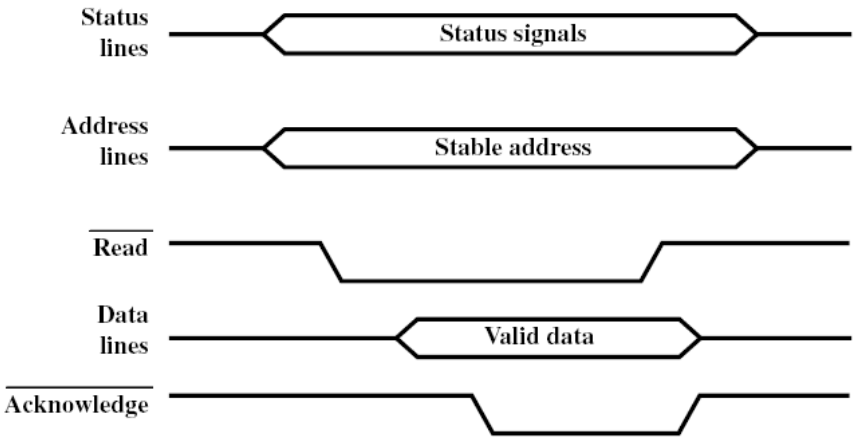
Synchronous Vs Asynchronous - Timing

- **Synchronous**
  - Events determined by clock signals
  - Control Bus includes clock line
  - A single 1-0 cycle is a clock cycle
  - All devices can read clock line
  - Usually synchronization on leading or raising edge
  - Usually a single cycle for an event
  - Usually stricter in terms of its timing requirements
- **Asynchronous**
  - Occurrence of one event on a bus depends on previous events

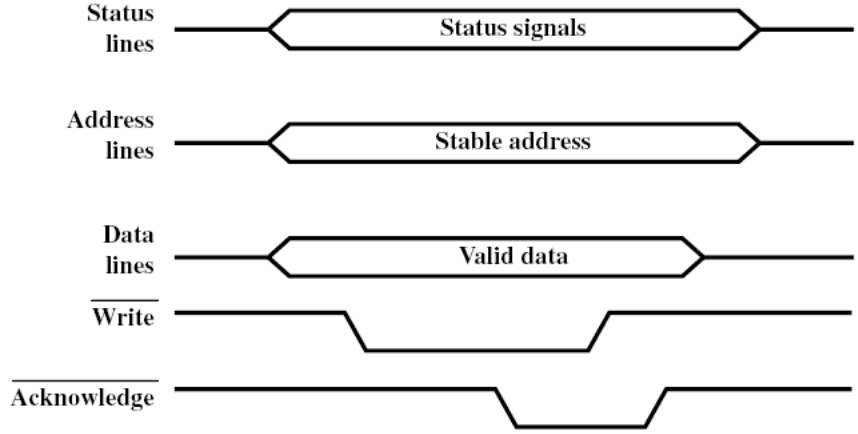
Synchronous Vs Asynchronous - Timing



Synchronous



Asynchronous read

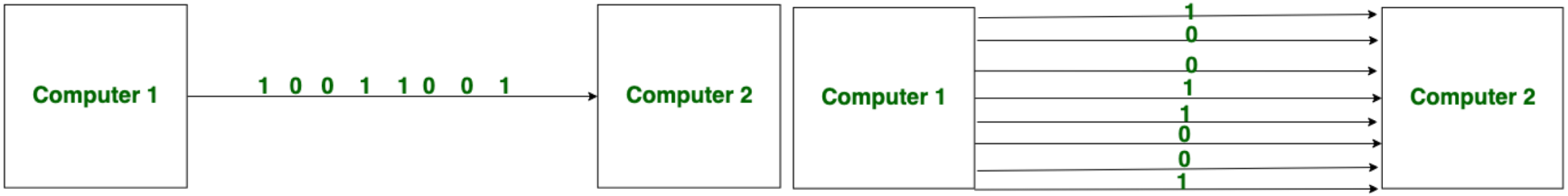


Asynchronous write

Bus Structure – Control lines

- Because multiple devices communicate on a line, control is necessary
  -Timing
- Typical control lines include:
  - Memory Read or Write
  - I/O Read or Write
  - Transfer ACK
  - Bus request
  - Bus grant
  - Interrupt request
  - Interrupt acknowledgement
  - Clock
  - Reset

Parallel Vs Serial



S.NO	Serial Transmission	Parallel Transmission
1	In serial transmission, data(bit) flows in one device to another	In Parallel Transmission, data flows in multiple lines.
2	Serial Transmission is cost-efficient.	Parallel Transmission is not cost-efficient.
3	In serial transmission, one bit transferred at one clock pulse.	In Parallel Transmission, eight bits transferred at one clock pulse.
4	Serial Transmission is slow in comparison with Parallel Transmission.	Parallel Transmission is fast in comparison of Serial Transmission at same clock rate.
5	Generally, Serial Transmission is used for long-distance.	Generally, Parallel Transmission is used for short distance.
6	The circuit used in Serial Transmission is simple.	The circuit used in Parallel Transmission is relatively complex.

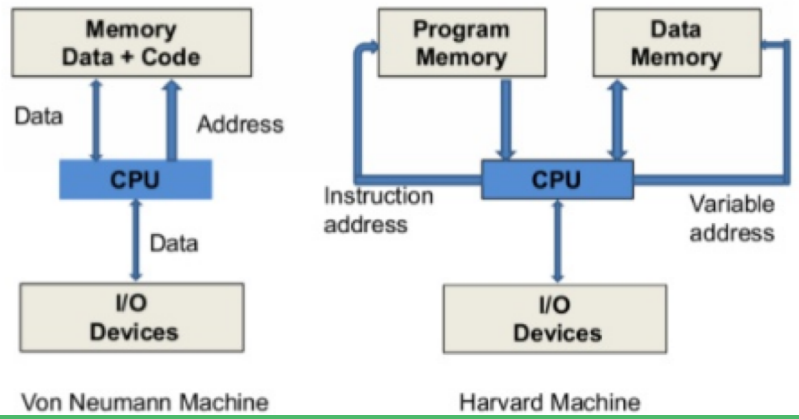
Why serial ?

As the parallel bus can be multiple times of speed of serial bus, why still need a serial bus in high data rate application?

- Propagation delays between wires is one of main issues. The data pattern of data lines is different, then the delay also different. After some length of cable, the "data" can be fast or slow which would cause error
- Cross talk is another main issues. The signal in wires couples to other wire and create a small voltage. Once this small voltage is built up to certain level, creates error
- Reflection also one of the main issues. "Bus" is a transmission line. In high frequency operation the impedance miss-match causes reflection. The only way is to limit the cable length.

High speed serial bus normally be a pair of wire which is built for good impedance matching and operate at very high frequency and long distance. An example - Low-Voltage Differential Signalling (LVDS), it can be as high as 4Gbps in short range and still has 100Mbps at 40 meters long

VON NEUMANN Vs HARVARD architechure

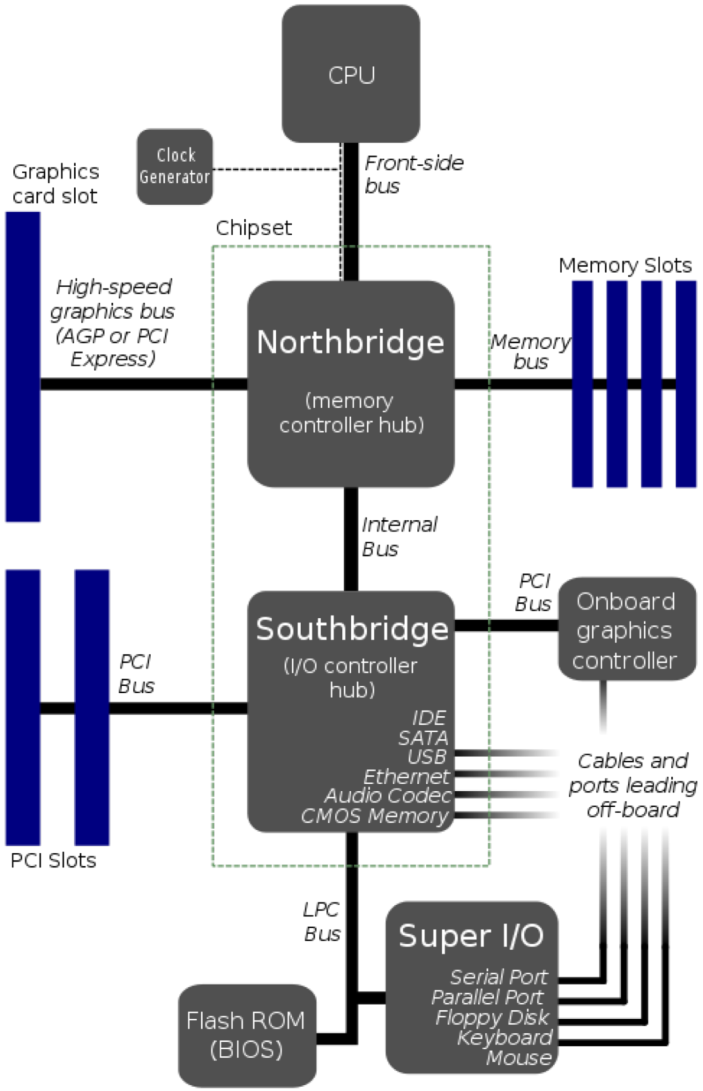


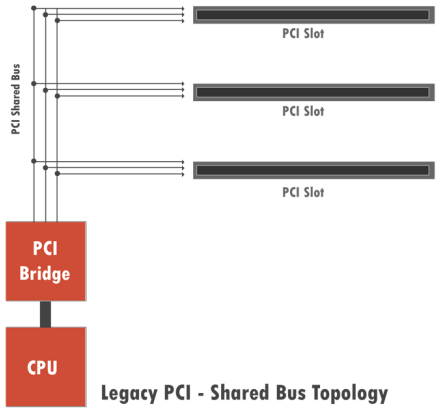
VON NEUMANN ARCHITECTURE	HARVARD ARCHITECTURE
It is ancient computer architecture based on stored program computer concept.	It is modern computer architecture based on Harvard Mark 1 relay based model.
Same physical memory address is used for instructions and data.	Separate physical memory address is used for instructions and data.
There is common bus for data and instruction transfer.	Separate buses are used for transferring data and instruction.
Two clock cycles are required to execute single instruction.	An instruction is executed in a single cycle.
It is cheaper in cost.	It is costly than van neumann architecture.
CPU can not access instructions and read/write at the same time.	CPU can access instructions and read/write at the same time.
It is used in personal computers and small computers.	It is used in micro controllers and signal processing.

Multiple Buses – Benefits

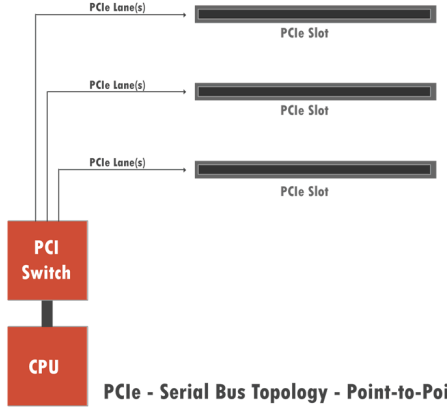
- Isolate processor-to-memory traffic from I/O traffic
- Support wide variety of interfaces
- Processor has bus that connects as direct interface to chip, then an expansion bus interfaces it to external devices (ISA – industry standard architecture -IBM)
- Cache (if it exists) may act as the interface to system bus

PC architecture

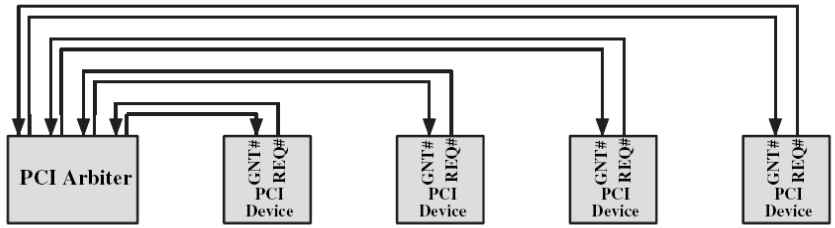




Legacy PCI - Shared Bus Topology



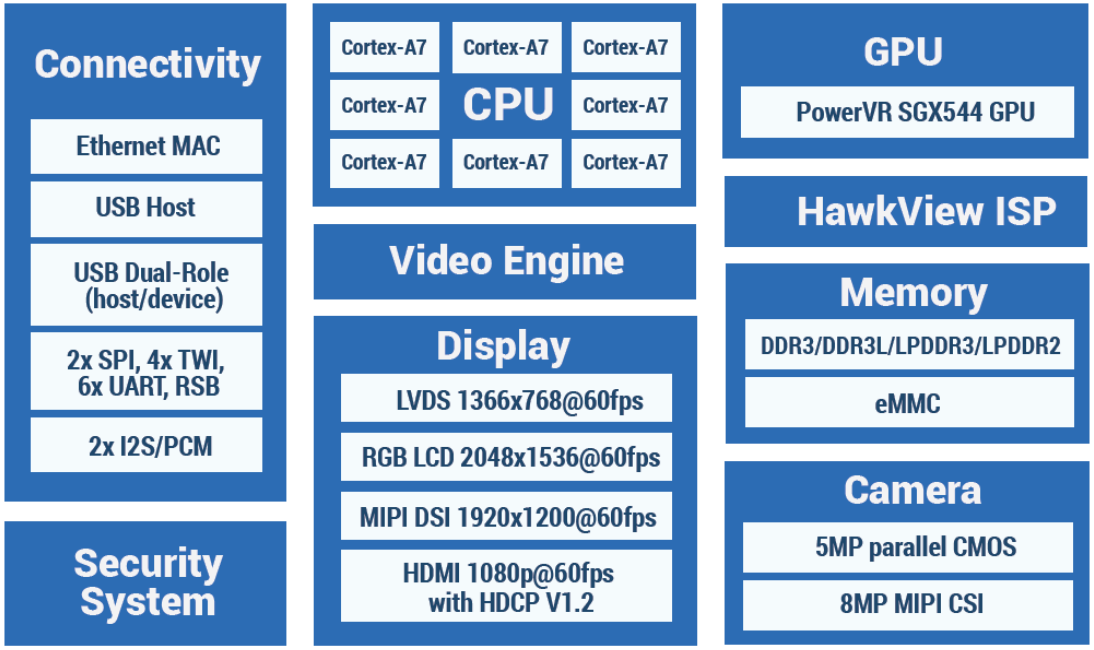
PCIe - Serial Bus Topology - Point-to-Point



Example of bus expansion

Embedded system MCU

Embedded system is a device specifically integrated CPU core, various I/O buses, and functional module for specific application. Below example is a market available product which has 8 pcs A7 cpu (Arm product), GPU, DDR and Flash memory interface, different types of interface port for equipment connection – camera, display, ethernet, etc.



Microprocessors vs Microcontrollers

- The main difference between Microprocessors and Microcontrollers:

Peripherals

- Microprocessors are basically electronic devices that execute our code. It is made up of integrated circuits and its abilities include doing mathematical and logical computations and controlling the devices connected to it. These days they are commonly referred to using the name 'cores'. The core by itself cannot do much, and it needs some other devices to serve a purpose. These other devices are called peripherals.
- In early days these peripherals were bought separately and soldered onto a PCB along with a microprocessor chip to make a useful system that can accomplish a given job. This approach had several disadvantages like

- 1. so much hardware design effort was needed to design a PCB that makes them all go together,
- 2. PCB area/size was too large and it prevented miniaturization and
- 3. lots of software design efforts were needed to integrate the system

What are peripherals?

Peripherals are devices that aid the microprocessor to accomplish a given job. In other words, they serve as accessories to the microprocessor. Depending on their location they can be classified into 2 types, if they are located inside the SoC (expands to System on Chip, in other words, its just the IC containing the microprocessor) of a micro-controller they are called as on-chip peripherals and if they are located outside the SoC but on the same PCB they are called off-chip peripherals.

There are three types of peripherals:

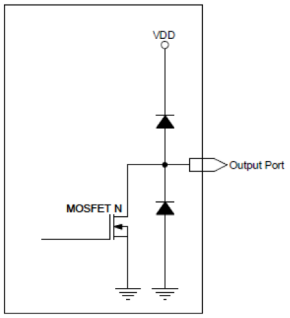
INPUT – Mouse, Keyboard, Camera, Game pad

OUTPUT – Printer, Speaker, Microphone, Display

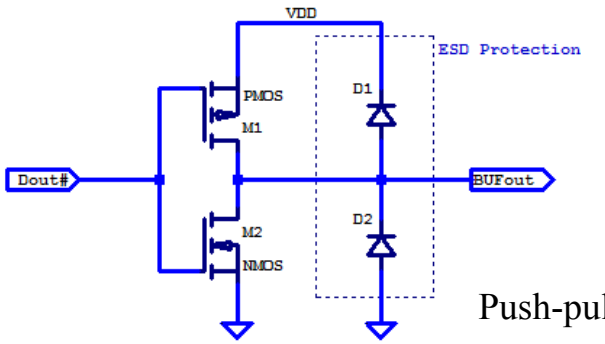
STORAGE – eMMC, Hard disk, CD rom

GPIO

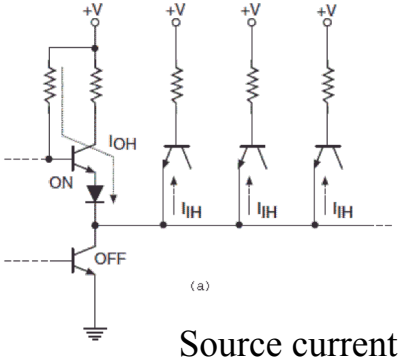
- GPIO stands for General Purpose Input and Output. These refer to the pins that are coming out of the microcontroller SoC.



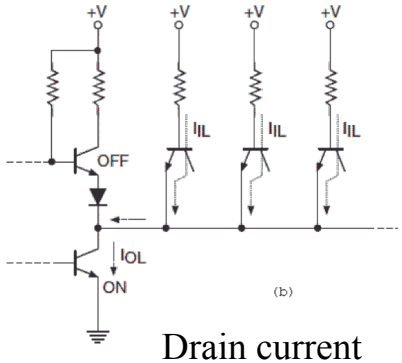
Open drain



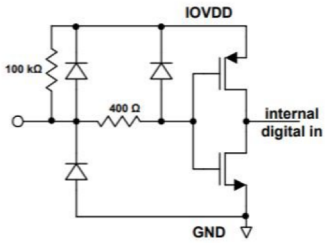
Push-pull



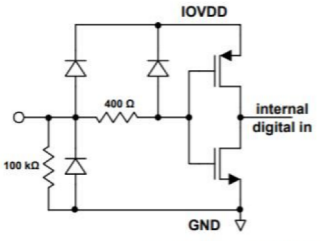
Source current



Drain current



Internal pull up

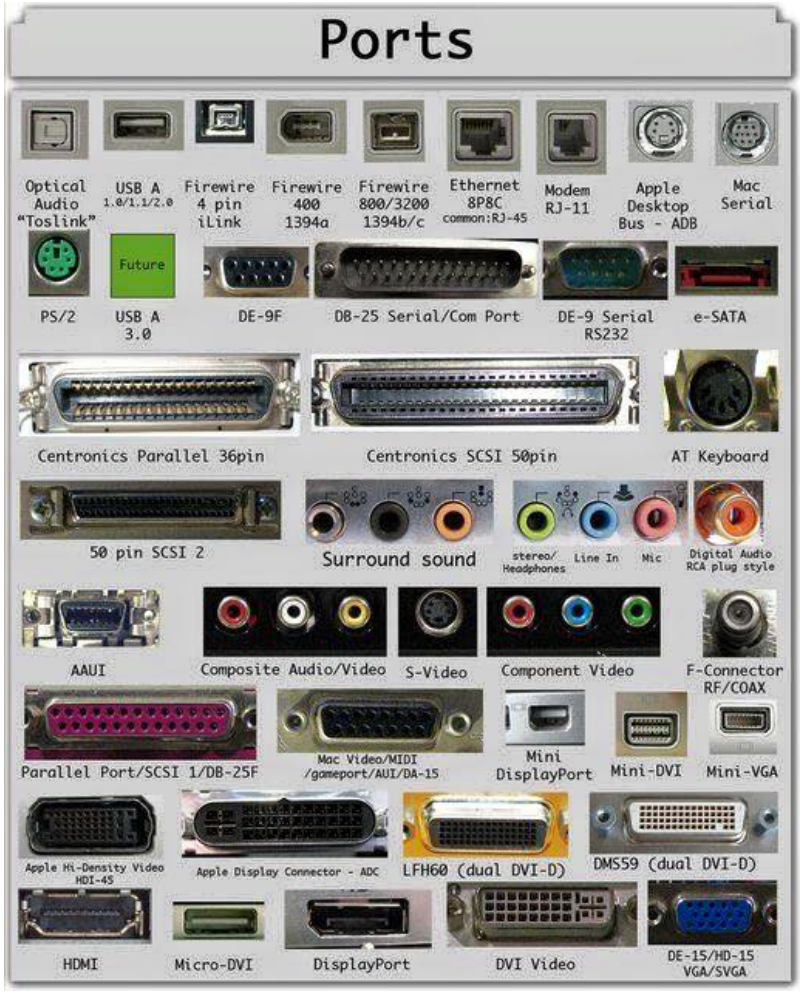


Internal pull down

Connection between MCU and peripherals

- Peripherals connection inside SoC is well designed and not able to be changed afterward. The only things can do is "select" the options through firmware.

- Outside SoC, the requirement and consideration is similar:
  - Software/firmware level is relatively simple, follow the related specification with design margin (mainly speed and protocol limitation) is good enough



Connection between MCU and peripherals (cont'd)

- Hardware
  - Voltage
    - 1.5V, 1.8V, 2.5V, 3.3V, +/-5V, +/-12V, 24V… There are many voltages used in same system. If wrongly connect a low voltage device to high voltage, the result very likely is smoke or "crack" sound
  - Current
    - The output port should have enough power to drive the input port(s) . The threshold voltage determines "1" or "0" for proper functioning. If the current is not enough, lower the voltage, and, "1" is no longer "1" but not "0". On the other hand if the current drain capability is not large enough, the voltage of "0" goes up then "0" no longer be "0" but not "1".
  - Impedance
    - Bus itself is a transmission line. The impedance matching in high frequency is very important. Miss match means reflection is high and the waveform will be distorted seriously. Once the "ripple" exceeds the allowable limit, shows an error

Connection between MCU and peripherals (cont'd)

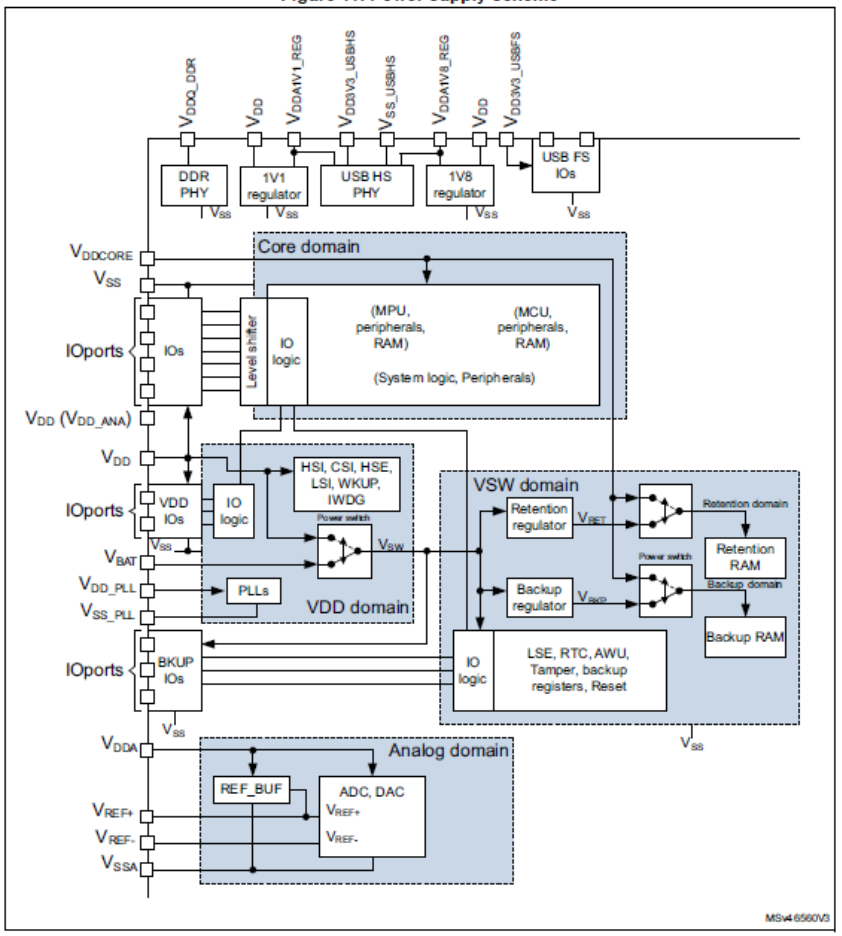
- Noise
  - There are huge noise source: transistor switching noise, power line current loop… Once the noise voltage superimpose on the power line and exceed the threshold, error occurs
- Timing
  - Power up sequence and reset activate/inactivate, needs attention. Otherwise, the system may not work properly
  - Clock accuracy and stability need to be well calculated. If some operation condition exceeds the working window, the system fails.
  - In discrete logic design, the propagation delay, raising / falling edge speed should be well considered. Otherwise, there may be "glitches" occasionally which will cause system failure.

Supply voltage Examples

Table 13. General operating conditions (continued)

Symbol	Parameter	Operating conditions	Min.	Typ	Max.	Unit
V_{DDCORE}	Digital core domain supply voltage	Run mode (F_{mpuss_ck} above 650 MHz)(5)	1.30	1.34	1.38	V
		Run mode (F_{mpuss_ck} up to 650 MHz)	1.18	1.20	1.25	
		Stop, LP-Stop mode	1.10	1.20	1.25	
		LPLV-Stop mode	0.85	0.90	1.25(6)	
		Standby mode	0	0	0.75	
V_{DDA}	Analog operating voltage	ADC used with V_{REF} < 2 V	1.62	-	2	V
		ADC used with V_{REF} > 2 V	2	-	3.6	
		DAC used	1.8	-	3.6	
		VREFBUF with V_{REF} = 1.5 V(7)	1.8	-	3.6	
		VREFBUF with V_{REF} = 1.5 V and ADC used		-	2	
		VREFBUF with V_{REF} = 1.8 V(8)	2.1	-	3.6	
		VREFBUF with V_{REF} = 2.048 V	2.35	-	3.6	
		VREFBUF with V_{REF} = 2.5 V	2.8	-	3.6	
		ADC, DAC, V_{REF} not used	0	-	3.6	
V_{BAT}	Backup operating voltage	64 KB retention SRAM not used	1.2	-	3.6	V
		64 KB retention SRAM used	1.4			
V_{DD3V3_USBFS}(9)	USB FS I/O supply voltage	USB OTG FS used	3	3.3	3.6	V
		USB OTG FS not used	0	-	3.6	
V_{DD3V3_USBHS}(9)(10)	USB HS I/O supply voltage	USBH or USB OTG HS used	3.07	3.3	3.6	V
		USBH and USB OTG HS not used	0	-	3.6	
V_{DD3V3_USB}(9)	USB I/O supply voltage	USB used	3.07	3.3	3.6	V
		USB not used	0	-	3.6	
V_{DDQ_DDR}(11)	DDR PHY supply voltage	DDR3 memory	1.425	1.5	1.575	V
		DDR3L memory	1.283	1.35	1.45	
		LPDDR2 or LPDDR3	1.14	1.2	1.3	
V_{DDA1V8_REG}	USB HS PHY voltage supply with 1.8 V regulator in bypass mode	BYPASS_REG1V8 = V_{DD}	1.65	1.8	1.95	V

Figure 11. Power supply scheme



Caution: Each power supply pair (V_{DD}/V_{SS}, V_{DDCORE}/V_{SS}, V_{DDA}/V_{SSA} ...) must be decoupled with filtering ceramic capacitors. These capacitors must be placed as close as possible to, or below, the appropriate pins on the underside of the PCB to ensure good operation of the device. It is not recommended to remove filtering capacitors to reduce PCB size or cost. This might cause incorrect operation of the device.

Current example

USB power standards

Specification	Current	Voltage	Power (max.)
Low-power device	100 mA	5 V[a]	0.50 W
Low-power SuperSpeed (USB 3.0) device	150 mA	5 V[a]	0.75 W
High-power device	500 mA[b]	5 V	2.5 W
High-power SuperSpeed (USB 3.0) device	900 mA[c]	5 V	4.5 W
Multi-lane SuperSpeed (USB 3.2 Gen 2) device	1.5 A[d]	5 V	7.5 W
Battery Charging (BC) 1.1	1.5 A	5 V	7.5 W
Battery Charging (BC) 1.2	5 A	5 V	25 W
USB-C	1.5 A	5 V	7.5 W
	3 A	5 V	15 W
Power Delivery 1.0 Micro-USB	3 A	20 V	60 W
Power Delivery 1.0 Type-A/B	5 A	20 V	100 W
Power Delivery 2.0/3.0 Type-C	5 A[e]	20 V	100 W

a. ^ a b The V_{BUS} supply from a low-powered hub port may drop to 4.40 V.

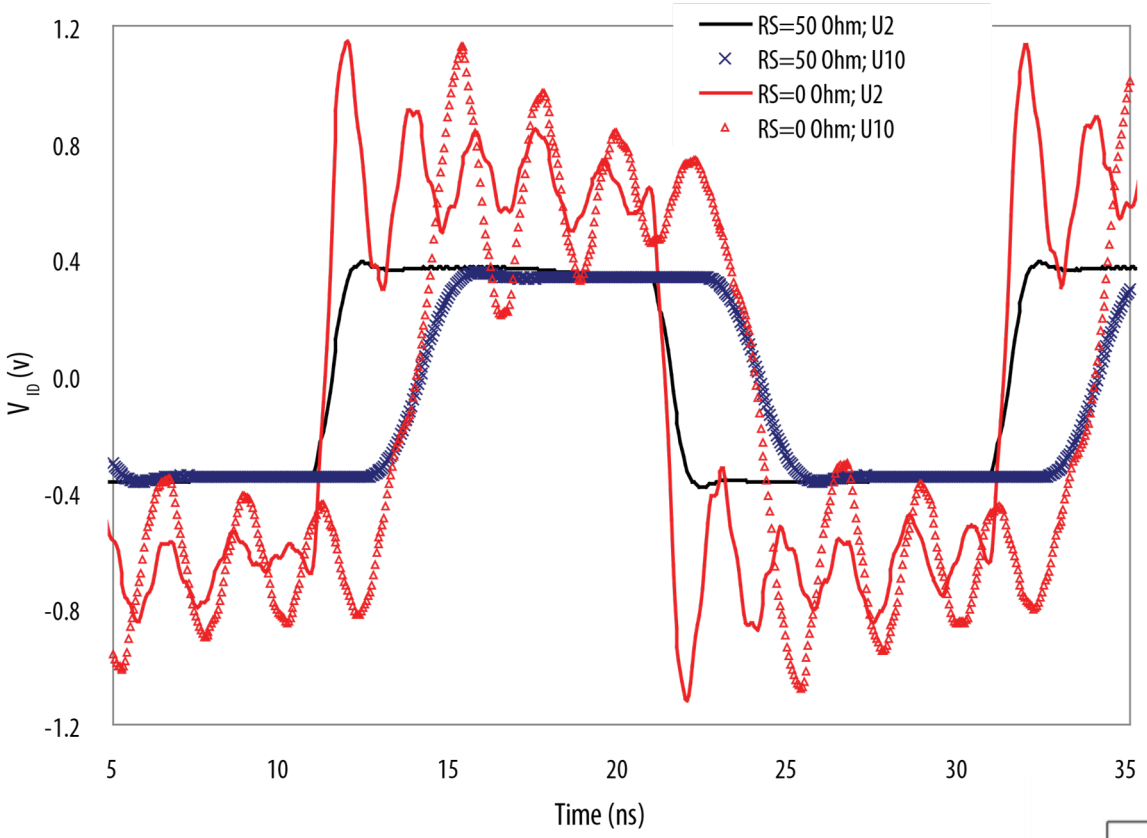
b. ^ Up to five unit loads; with non-SuperSpeed devices, one unit load is 100 mA.

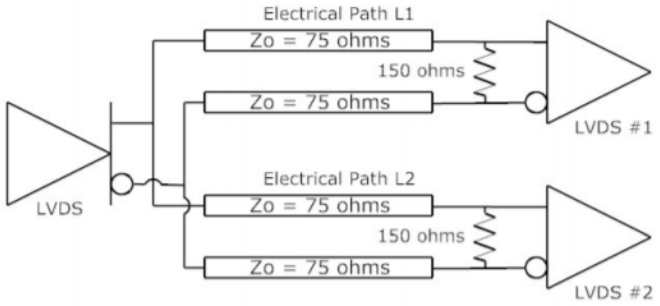
c. ^ Up to six unit loads; with SuperSpeed devices, one unit load is 150 mA.

d. ^ Up to six unit loads; with multi-lane devices, one unit load is 250 mA.

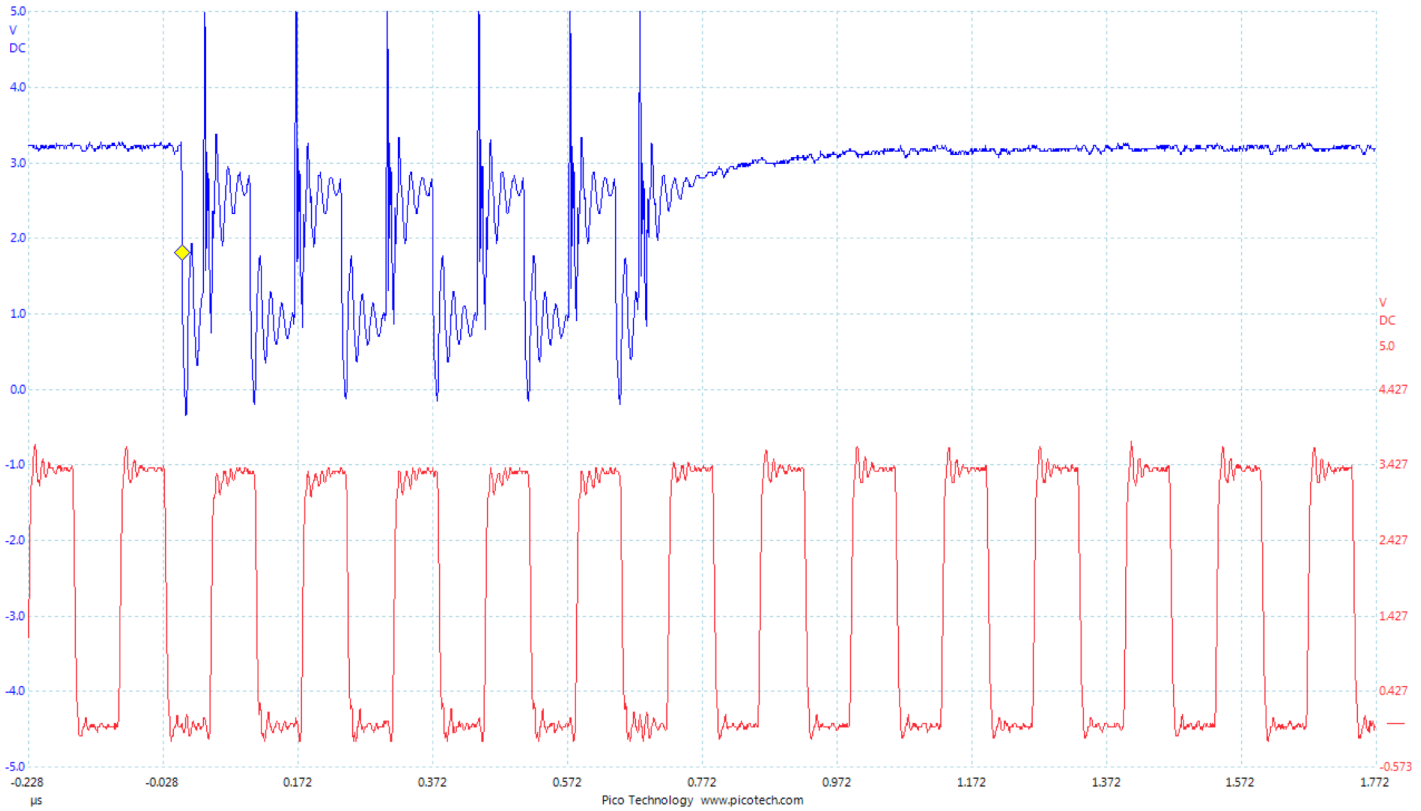
e. ^ > 3 A (60 W) operation requires an electronically marked cable rated at 5 A.

Impedance example

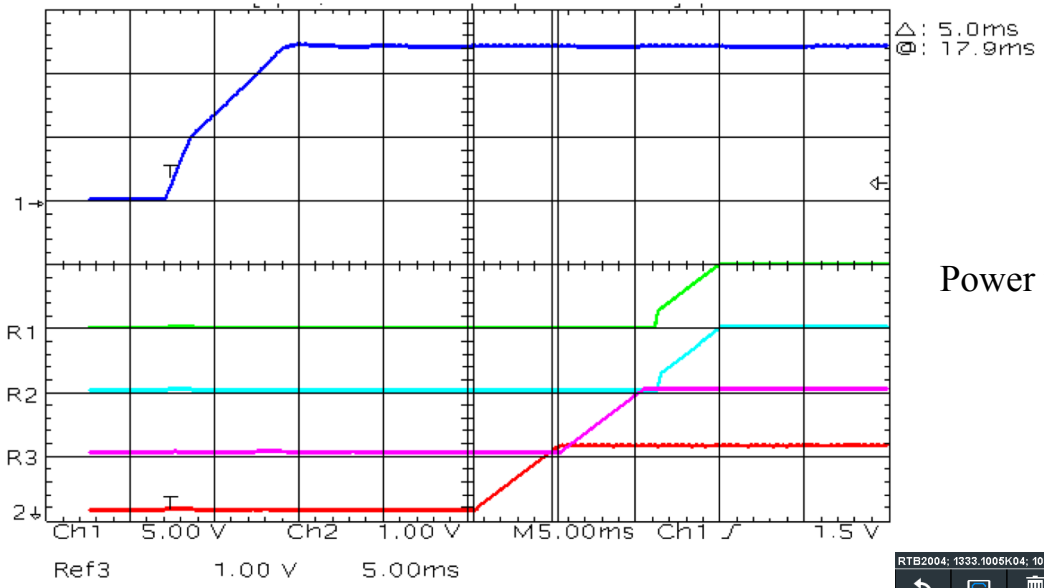




Noise Examples



Timing example



Power up Reset

SEK112 POR problem. blue: input power 12V.
green: VCCPINT, no POR problem; cyan: VCCPINT with POR fail
magenta: VCCINT, no POR problem; red: VCCINT with POR fail