Daniel **Fitz**
43961229

University of Queensland

CSSE3010 – Embedded System Design

Lecture Summary

Table of Contents

Analog Interfacing

Accuracy, Precision, Resolution

Accuracy

Proximity of measurement results to the true value

Precision

Repeatability or reproducibility of the measurement

Measurement resolution

The smallest change in the underlying physical quantity that produces a response in the measurement

Sampling

Time Quantization

Signal value read/available only in specific times (usually at the same interval). This can cause aliasing – frequency ambiguity of signal components

Amplitude Quantisation

Amplitude of each sample can only take one of a finite number of different values. This adds **quantisation noise**: an irreversible corruption of the signal

Sampling Theorem

Nyquist Theorem

A signal having no spectral components above f_m Hz can be determined uniquely by values sampled at the rate:

$$f_s > 2f_m$$

$f_s > 2f_m$ is called the Nyquist rate

Aperture time

The time during which ADC is continuously converting the varying analog input

$$\text{Max slope} = \frac{\Delta V}{\Delta t} = \omega \times V_{peak} = 2\pi f V_{peak}$$

Example

$$\begin{aligned}
\frac{\Delta V}{\Delta t} &= 2\pi f V_{peak} && (f = (\Delta V/\Delta t)(1/2\pi V_{peak})) \\
\Delta V &= \frac{1}{4} LSB = \frac{1}{4}\left(\frac{10V}{4096}\right) = 0.6mV \\
\Delta t &= 10us \\
f &= \left(\frac{\Delta V}{\Delta t}\right)\left(\frac{1}{2\pi V_{peak}}\right) = \left(\frac{0.6mV}{10us}\right)\left(\frac{1}{2\pi 5V}\right) = 2Hz
\end{aligned}$$

Signal to Noise Ratio (SNR)

- Ratio of the maximum sine wave level to the noise level
- Maximum sine wave has an amplitude of $\pm 2^{n-1}$ which equals an RMS value of:

$$0.71 \times 2^{n-1} = 0.35 \times 2^n$$

- SNR is:

$$20\log_{10}\left(\frac{0.35 \times 2^n}{0.3}\right) = 20\log_{10}(1.2 \times 2^n) = 1.8 + 6n\,dB$$

Sample and Hold

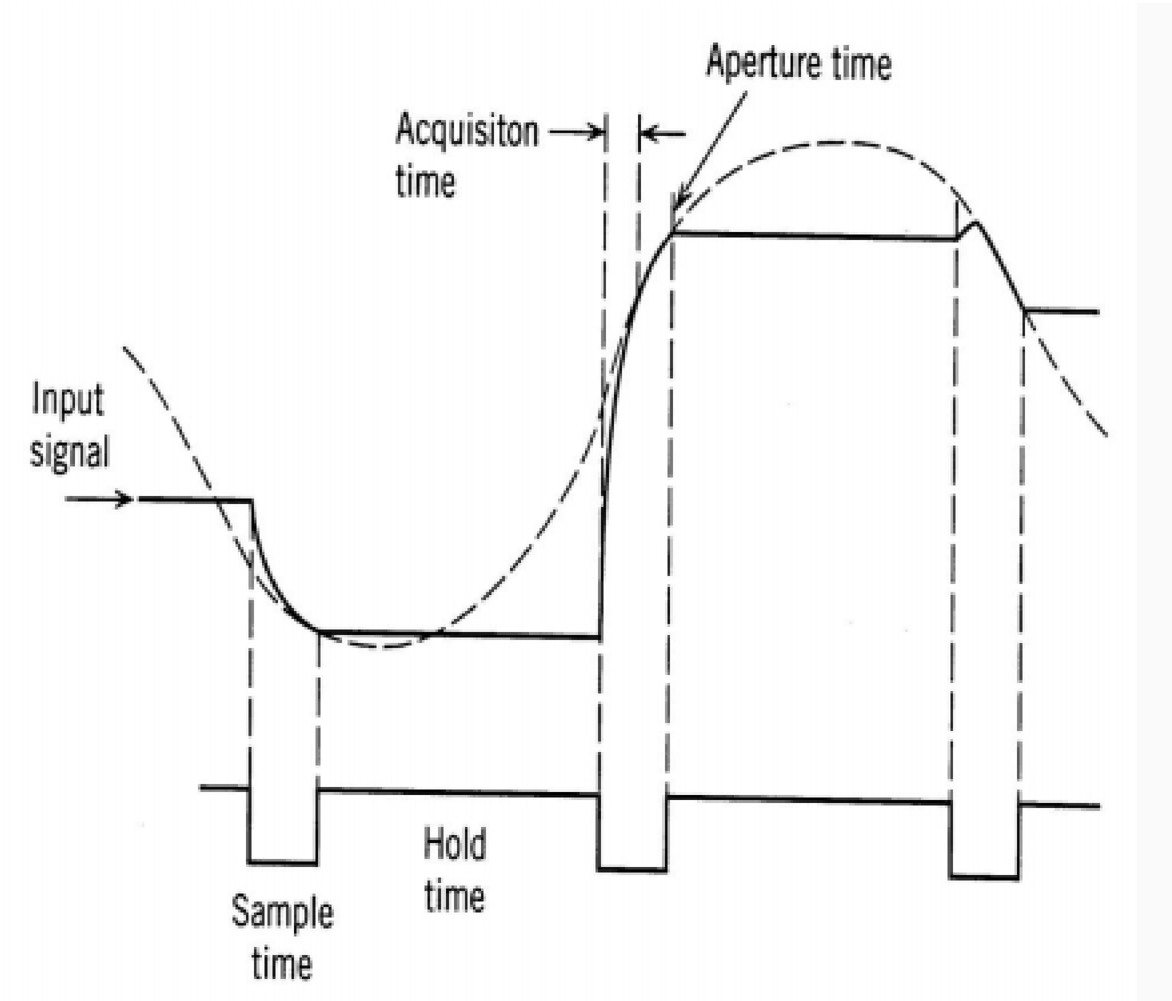

Figure 1: Sample and Hold

Resolution and Dynamic Range

Number of Binary Bits (n)	Full-Scale Decimal Value $(2^n - 1)$	LSB Weight % of Full-Scale Range	LSB Voltage for 1-V Full-Scale Range	Quantization Error Percent of Full-Scale Range	Dynamic Range (From LSB to Full Scale) (dB)
4	15	6.25	60 mV	3.12	24.08
6	63	1.56	16 mV	0.78	36.12
8	255	0.3906	3.9 mV	0.195	48.16
10	1023	0.0977	0.98 mV	0.0488	60.21
12	4095	0.0244	0.24 mV	0.0122	72.25
14	16383	0.00610	61 uV	0.00305	84.29
16	65535	0.00153	15 uV	0.00075	96.33
18	262143	0.000382	4 uV	0.0002	108.37
20	1048575	0.0000954	1 uV	0.00005	120.41

Conclusions

- Interface to analogue world requires thorough understanding and analysis of physical properties this is why it is difficult
- The A/D D/A on-chip converters on microcontrollers are average precision and would require off chip hardware to make conversions more accurate or faster
- Always start interfacing with analysis of the properties and requirements of the analogue side. Digital is always faster

Timing Interfacing

- Use of timing bistate (high or low) waveforms or 'square wave' for interfacing
- A timing waveform can 'mimic' an analog voltage
  - Note Analog voltages an be approximated with specific square waves frequencies and duty cycle
- Commonly used for Pulse Width Modulation, Waveform Frequency or Time Spacing Interfaces
- Timing Interfacing consists of three parameters:
  - Period
  - Frequency
  - Duty Cycle

Waveform Basics

- Period (s) $= T_{high} + T_{low} = T_{period}$
- Frequency (Hz) $= \frac{1}{T_{period}}$
- Duty Cycle (%) $= \frac{100 \times T_{high}}{T_{high} + T_{low}} = 100 \times \frac{T_{high}}{T_{period}}$

Waveform Time Spacing Measurement

- Time spacing of a waveform used to convey information
- Useful for 'irregular' waveforms (high low times are not the same)
  - E.g. time spacing between pulses
- Implemented using Timer Input Capture interrupts
  - A timer counter value is recorded, each time a transition (rising or falling) occurs on the input line ### Frequency Measurement
- The frequency of a waveform can also convey information
- Useful for 'regular' waveforms (High and low times are the same)
- Typically used for optical or mechanical based systems

Example: Wheel Encoder The wheel encoder works by shining light through a pin wheel and detecting the frequency of the light passing through. The frequency of the light passing through is proportional to the speed of the wheel

Waveform Frequency/Period Measurement

- Implemented using Timer Input Capture interrupts
- Can measure using period/frequency by timing transitions.
  Disadvantage: Must rely on accurate timer with enough resolution/precision (e.g. 1ns resolution)
- The number of transitions or zero crossings within a time window, is proportional to the waveform frequency/period.
  Advantage: Does not need high resolution.
  Disadvantage: Only works for regular waveforms

Pulse Width Modulation (PWM)

- Pulse Width Modulation (PWM) uses duty cycle to convey information
- PWM can be used approximate analog (multi-value) waveforms
- Used for controlling mechanical systems such as motors and servo motors

PWM precision/resolution

$$period = N \times \Delta$$

$\Delta =$ resolution
$N =$ PWM precision

Example:

$$Period\ 20ms, \Delta = 20us \rightarrow N = 1000 \rightarrow 10bits$$

Timer

Timers features:

- Update interrupts – cause an update interrupt (periodic or not)
- PWM – pulse width modulation (used for controlling servos)
- Timer Input Capture interrupts – cause an interrupt, when a rising or falling edge is detected on an input signal – captures value of timer
- Timer Output Compare – toggle an output pin high or low, when a compare value matches the timer value

ADC

- 3 ADCs: ADC1 (master), ADC2 and ADC3 (slaves)
- Maximum frequency of the ADC analog clock is 36MHz
- 12-bits, 10-bits, 8-bits or 6-bits configurable resolution
- ADC conversion rate with 12 bit resolution is up to:
  - 2.4 M.samples/s in single ADC mode
  - 4.5 M.samples/s in dual interleaved ADC mode
  - 7.2 M.samples/s in triple interleaved ADC mode
- Conversion range: 0 to 3.6 V
- ADC supply requirement: VDDA = 2.4V to 3.6V at full speed and down to 1.65V at lower speed
- 3 ADC1 internal channels connected to:
  - Temperature sensor
  - Internal voltage reference: Vrefint (1.2V typ)
- External trigger option for both regular and injected conversion
- Single and continuous conversion modes
- Scan mode for automatic conversion of channel 0 to channel 'n'
- Left or right data alignment with in-built data coherency
- Channel by channel programmable sampling time
- Discontinuous mode
- Dual/Triple mode (with ADC1 and ADC2 or all 3 ADCs)
- DMA capability
- Analog Watchdog on high and low thresholds
- Interrupt generation on:
  - End of Conversion
  - End of Injected conversion
  - Analog watchdog
  - Overrun

Embedded Design Methodology

Top Down Design

- Embedded System design methodology
  - A complex system is created to meet specific design attributes
- Top down is a process in which a complex design is first organised as a top or high level view
  - The high level overview of the design is divided into sub-components
  - Each sub-componenet is a distinct section of the top level design

        * The sub-components can be further broken down into elements

Valvano

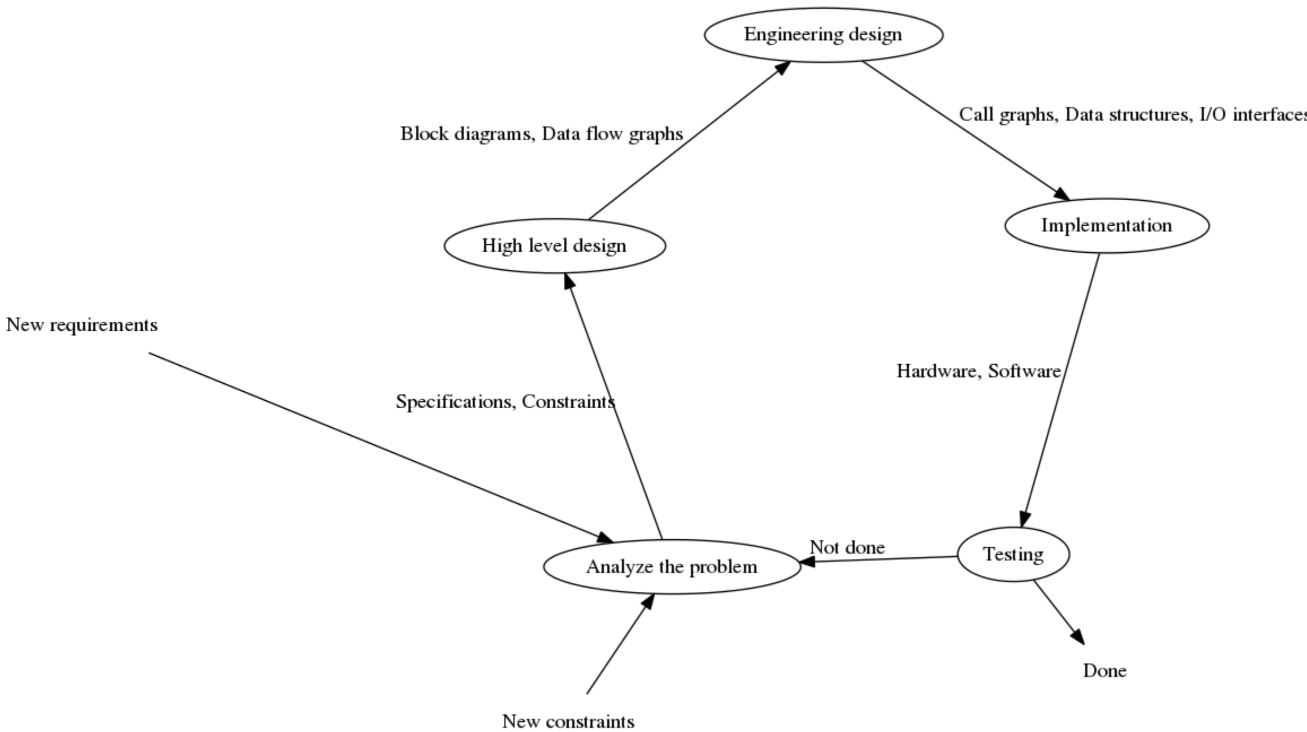

Figure 2: Top Down Valvano

High Level Design Overview

Consists of a number of concepts:

- System Flow:
  - Schematic – shows system inputs and outputs connections
  - Signal/Data Flow diagram
    * Shows the connections of the inputs, all the way to the outputs
      · Shows each stage of connecting the input to the output
- Program Flow:
  - State Diagrams
    * Embedded System Programming main loop can be abstracted as a State Controller
      · A microcontroller program must enter and exit certain states, as it executes
  - Flow Charts
    * Software abstraction of microcontroller program

System Flow – Signal/Data Flow

- Signal/Data Flow diagram
  - Shows the connections of the inputs, all the way to the outputs
- Shows each stage of connecting the input to the output
- Differs to block diagram – is not an overview of the system
- Useful for identifying which software/hardware modules to use
- Useful for debugging and identifying:
  - Break points – where your code will definitely break
  - Weak points – where your code could potentially break
  - Bottle necks – where your system's performance is limited – i.e. 'slow' to respond

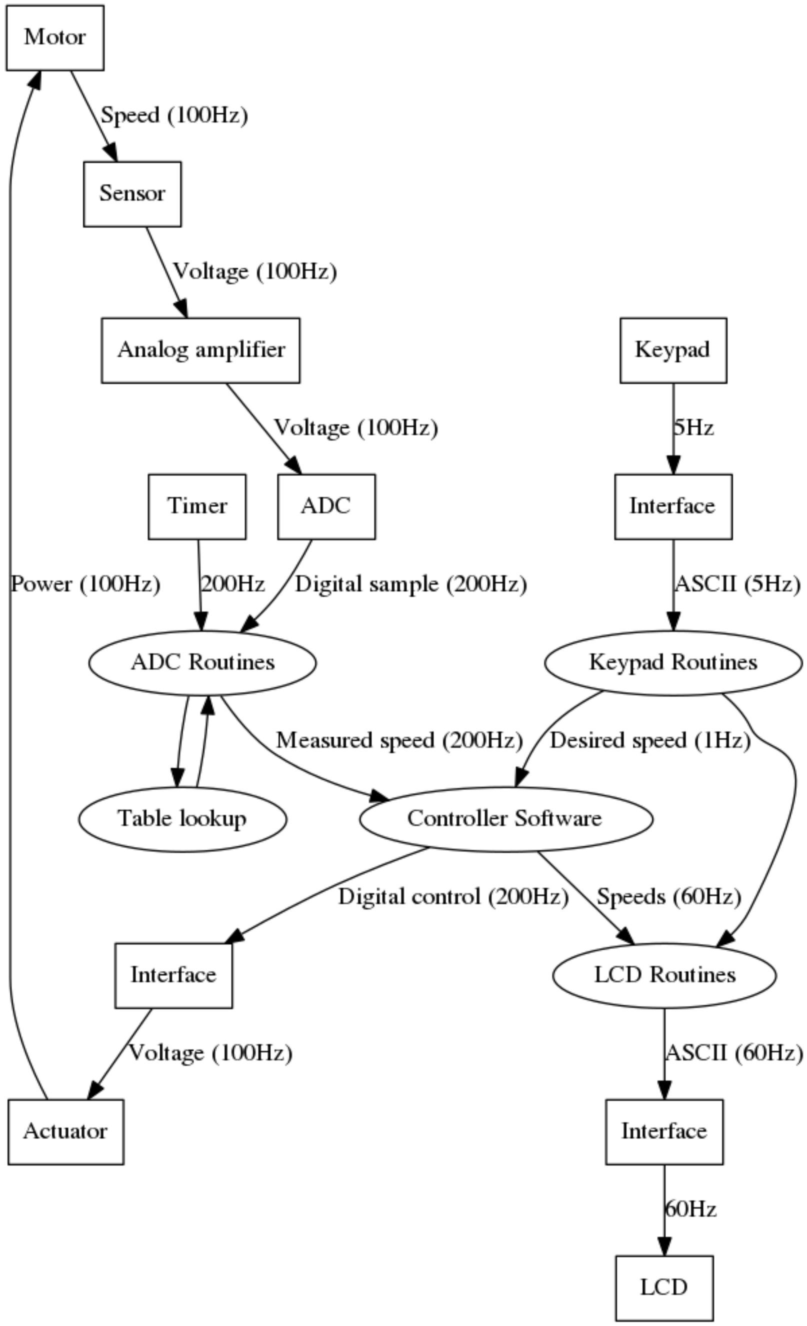

Figure 3: Motor controller Signal/Data Flow diagram

Program Flow

Your program flow should consist of:

- Main loop
- Hardware Initialisation function
- Functions – callable block of code
- Subroutine (not a function but a unit of code)
- Interrupt Service Routines

Program flow is described as:

- State Diagrams
  - main loop

- Flow Charts
  - subroutines

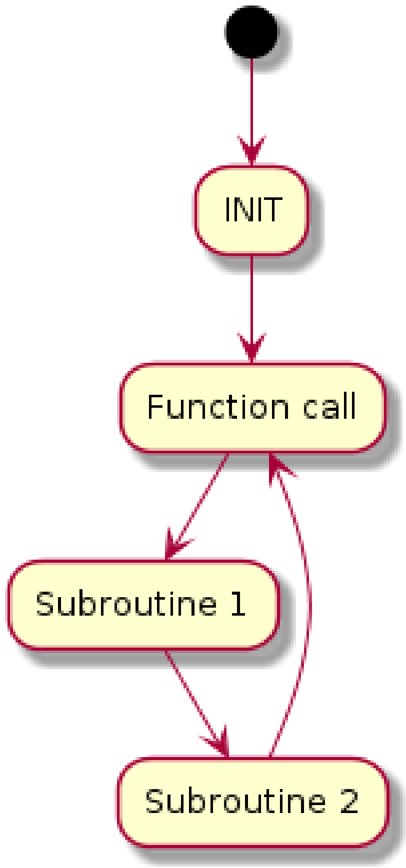

Figure 4: Program Flow – Outline

ISRs have a lightning symbol

Cyclic Executive

- Control loop, using an infinite loop
- Easy to implement
- Has disadvantages
  - unable to prioritising functions/features
  - no realtime control
  - can cause deadlocks, if more than one is used
- Use ONLY one Cyclic Executive to prevent deadlocks

Good Example:

```
while (1) {
    function_a();
    ...
    function_x();
}
```

Bad Example:

```
while (1) {
    while (1) {
        function_a();
```

```
            break;
        }
        ...
        while (1) {
            function_x();
            break;
        }
    }
```

- Appears to support multi-tasking by taking advantage of relatively short processes in a continuous loop:

```
    while (1) {
        function_a();
        ...
        function_x();
    }
```

- Different timing of operations can be achieved by repeating functions in the list:

```
    while (1) {
        function_a();
        function_a();
        function_b();
        function_x();
        function_b();
    }
```

Controller

- For more complex designs – need to implement a controller
- Use Cyclic Executive to implement a controller
- The controller should enter different 'states' of operation
- e.g. initialisation, operating, waiting, etc
- Controllers are typically implemented with Finite State Machines

Basics of Communication

Terminology

Simplex

Communication channel that sends information in one direction only

Half-duplex

Communication in both directions, but only one direction at a time

Full-duplex

Communication in both directions, simultaneously

Serial

One signal path

Parallel

Multiple signal paths

Baseband

Is the signal modulated at (or around) DC, (e.g. Wired transmission)

Bandpass

Or is it modulated onto a higher (carrier) frequency (e.g. Wireless LAN, Radio, TV)

Baseband Modulation

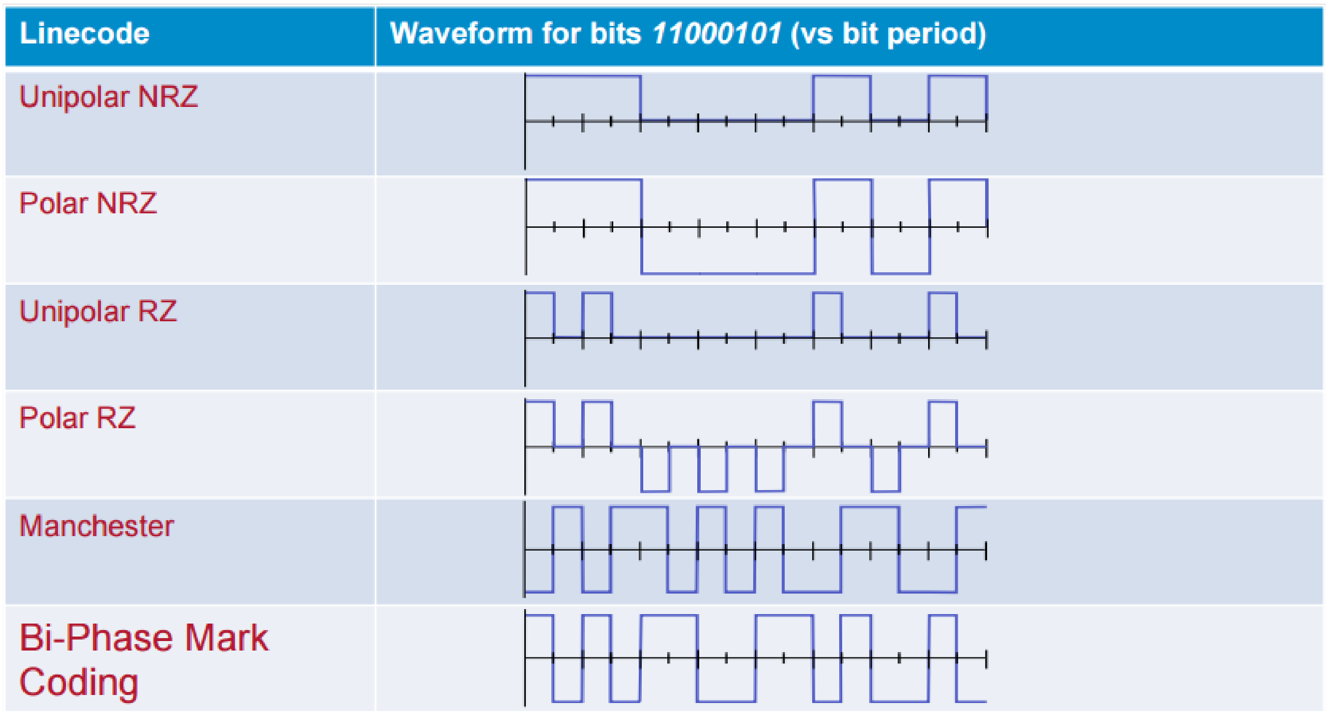

Figure 5: Baseband

Benefits Analysis of Modulation

The previous modulation techniques can be evaluated in terms of:

- Minimal DC component
- BW usage
- Polarity Inversion
- Timing Information
- Frequency Spectrum

Block Coding

- Defined as a (n, m) block code
- ‘n’ is the number of encoded bits
- ‘m’ is the number of data bits
- Implemented in different ways
- Here we will use the Generator matrix (G) and Parity Check matrix (H)
  - $y = x\,G$ (encoding data)
  - $s = H\,y^T$ (calculating the syndrome)
  - y is the code word, x is the data, s is the syndrome

Hamming (7, 4) in Matrix form

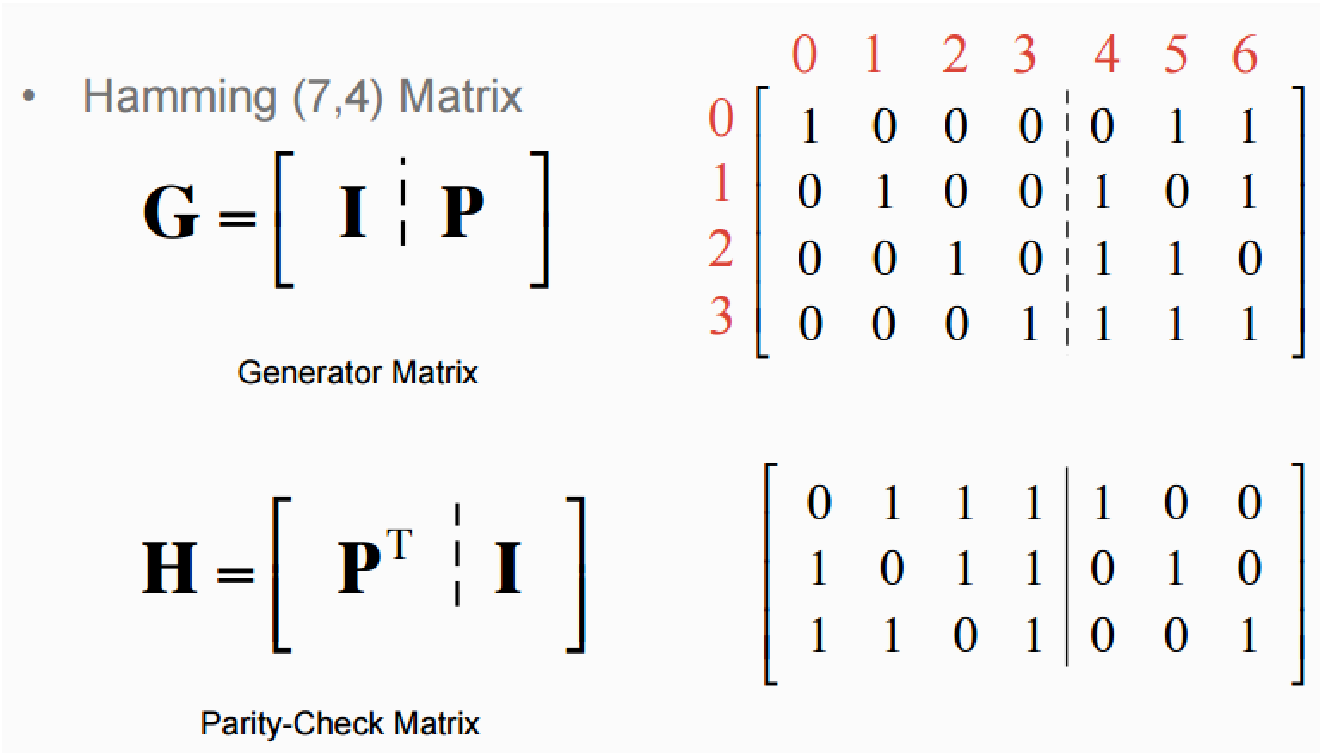

Figure 6: Hamming (7, 4) example matrix

Infrared Communications

- Infrared (IR) communications is a short-range form of wireless communications
- IR communications uses the infrared spectrum for transmitting and receiving information
- IR communications is widely as a remote control interface for entertainment and other interfacing applications

Finite State Machine

- Finite State Machine is an abstraction of a controller
- Commonly used design methodology for controllers
- Implemented with either microcontrollers or logic circuitry
  - Our focus on microcontrollers
  - Logic Implementations
- Consists of three sections:
  - Input Processing
  - Next State Processing
  - Output Processing

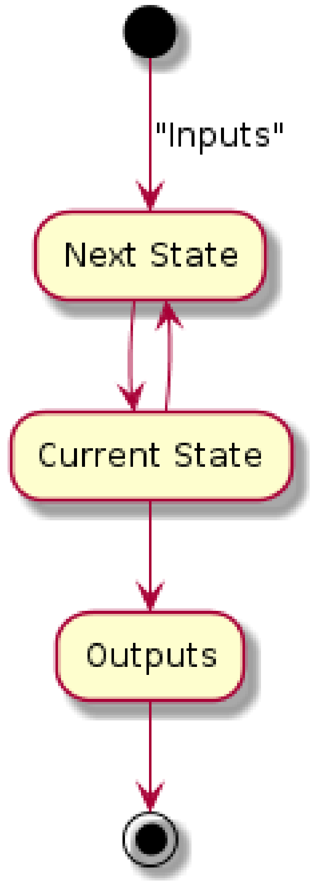

Figure 7: FSM Architecture – Moore Machine

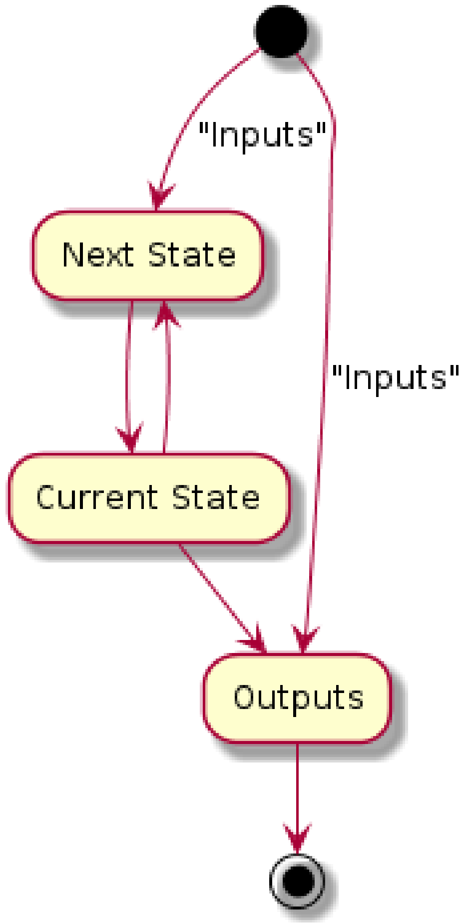

Figure 8: FSM Architecture – Mealy Machine

- Can combine both Mealy and Moore
  - Mealy outputs (depends on input only)
  - Moore outputs (depends on current state only)
- Implemented as cyclic executive

Algorithmic State Machine (ASM)

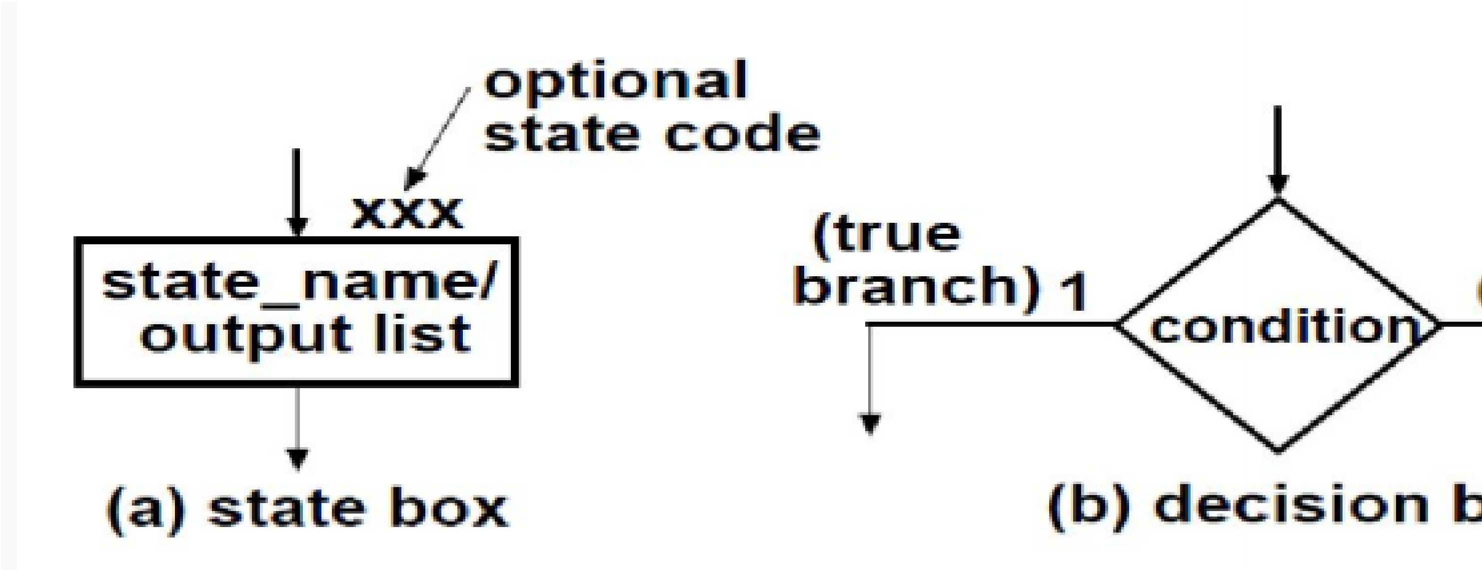

Figure 9: ASM Chart Symbols

- Constructed from ASM Blocks
- Each ASM Block consists of ONE state, together with decision boxes and conditional output boxes
- All the operations in the ASM block happen concurrently when the machine is in the given state
- One entrance, many exits
- A link path is a path from entrance to exit, determined by conditions that are true
- All outputs variables encountered on the active link path are true, all others are false

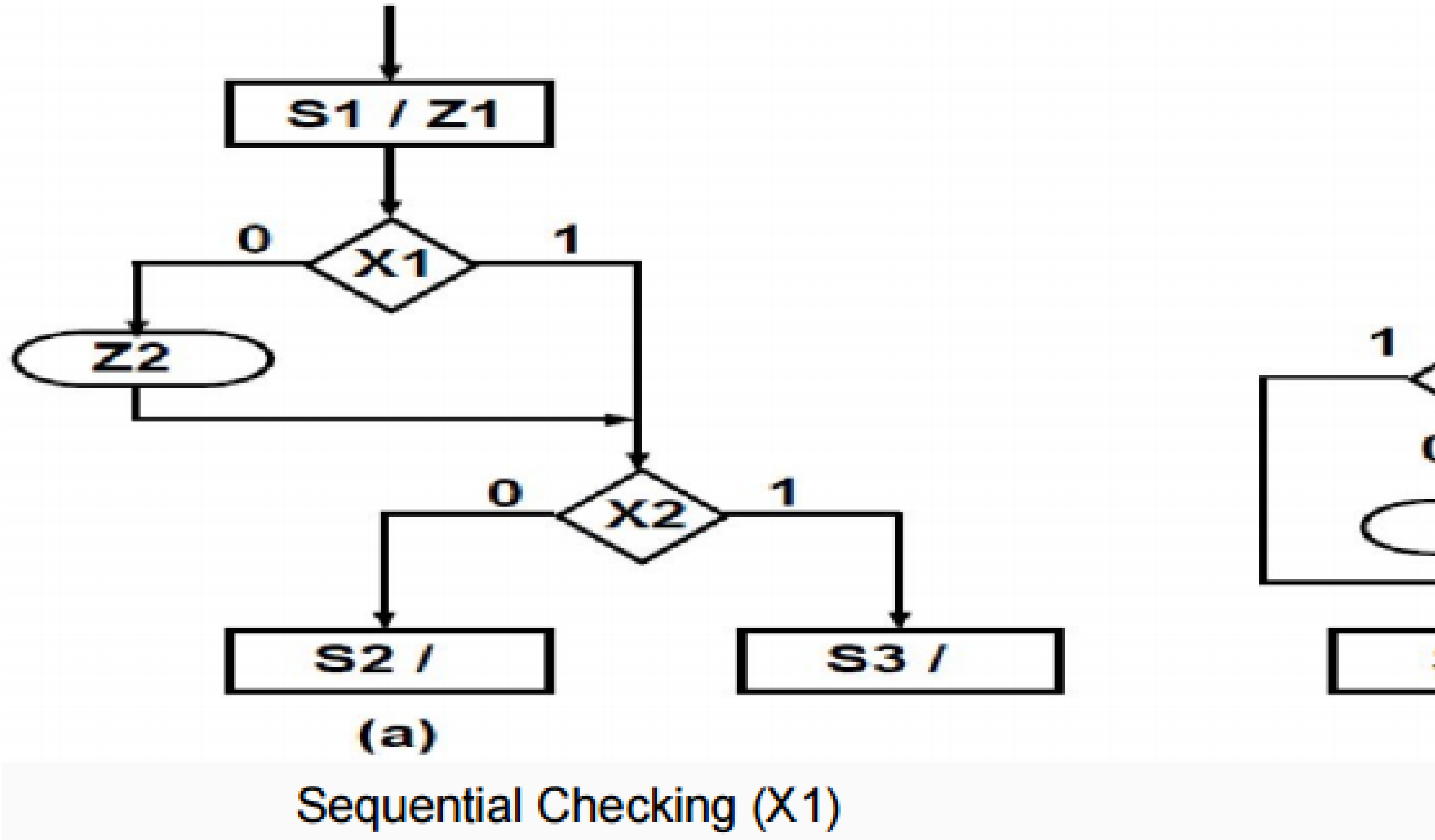

Figure 10: ASM Check conditions in parallel

Parallel Form of ASM

- There may be more than one link path which is true, so that different conditions may be evaluated in parallel
- However, for every unique combination of input variables, they can only have ONE exit path, leading to the next state

Conclusions

- Finite State Machine – FSM defines a sequence of operations that can be implemented in software or hardware
- ASM is a graphical representation of that sequence and easiest to conceptualise
- ASM is a formal specification if it obeys some clear rules
- It can be converted to C code or hardware automatically by appropriate software
- FSM is a very handy concept in defining control sequences and often used in real time embedded systems since it defines very well timing of events and can be checked for correctness by formal methods

Serial Interfacing

- Allow communication between digital devices using a sequential based signalling – e.g. square waves
- Allows for complex and continuous communication, using a few connections (or wires). E.g. to send 8 bits of data – either use 8 wires or a single wire, with 8 low/high transitions
- Variants:
  - Simple Signalling
  - I2C – Inter IC Communications
  - SPI – Serial Peripheral Interface
  - Universal Asynchronous Receive Transmit (UART)

Simple Signalling

- Signalling between digital devices
- Involves signal level transitions:
  - low-high or high-low
- Use to initiate, acknowledge or terminate data transfers
- Also known as 'handshaking'
  - Used for synchronous and asynchronous serial data transfers

I2C

- Only two bus lines are required
- No strict baud rate requirements like for instance with RS232, the master generates a bus clock
- Simple master/slave relationships exist between all components
- Each device connected to the bus is software-addressable by a unique address
- I2C is a true multi-master bus providing arbitration and collision detection
- Relatively slow bus in terms of data throughput

Serial Peripheral Interface (SPI)

- Synchronous Serial Protocol
- Seperate Transmit, Receive and Clock lines
- Select line is used for handshaking between master and slave device

Universal Asynchronous Receive Transmit (UART)

- Serial protocol that is not synchronous (e.g. no share clock signal)
- Relies on:
  - Initial handshake signalling
  - Agreed data transfer rate (baud rate)
  - Oversampling (disadvantage – requires more complex Hardware)
- Separate connections for receive and transmit
- Prone to error at high data rates
- RS232 & RS485 UART protocols are designed for consumer and industrial applications

Conclusions

- I2C bus is a 'proper' serial bus with a protocol for addressing devices and acknowledge signals for both master and slave which are all connected to the same data and clock
- SPI bus implements slave selection through separate enable lines and therefore requires more wires than I2C. From this perspective it is NOT a full-fledged serial bus
- UART bus is an asychronous protocol that requires oversampling (more complex Hardware)

Noise and Synchronisation

Hamming Distance

The number of bits which differ between two words

Cyclic codes

- Easily implemented in hardware
- Represent data bits using a polynomial e.g. message x encodes to $x(p)$, where:
  - $x(p) = x_{n-1}p^{n-1} + \ldots + x_1 p + x_0$
- More concrete example:
  - $x = [1\,0\,1\,1]$ (LSB first)
  - $x(p) = 1p^3 + 1p^2 + 0p + 1 = p^3 + p^2 + 1$
  - NOTE: lowest power always corresponds to LSB
- Cyclic codes are a special type of block code where every cyclic shift of a valid code gives another valid codeward

A cyclic shift, moves bits around from the least significant around.
For LSBit ordered bits: $[1\,0\,1\,1] \to [1\,1\,0\,1]$

In polynomial form:

- $x(p) = x_{n-1}p^{n-1} + \ldots + x_1 p + x_0$ is shifted to
- $x'(p) = x_{n-2}p^{p-1} + \ldots + x_0 p + x_{n-1} = px(p) + x_{n-1}(p^n + 1)$

Encoding Cyclic Codes

- Codes are encoded using a "generator polynomial" which is a factor of $p^n + 1$ of order $q = n - k$
  - NOTE: n = coded bits, k = msg bits
- Transmitted codewords $x(p)$ are in the form:
  - $x(p) = q_m(p)g(p)$
- Or in terms of the message bits $m(p)$
  - $x(p) = p^q m(p) + c(p)$
- $c(p)$ is the important bit and equals $\frac{p^q m(p)}{g(p)}$

Decoding Cyclic Codes

- Syndrome is calculated by division by $g(p)$ and taking the remainder
- If we take the correctly encoded data $[0\,1\,0\ \mid\ 1\,1\,0\,0]$ or $p^6 + p^5 + p$ and divide by $g(p)$, we will get no remainder