The Design Process

Introduction
As we’ve learned
Embedded systems are pandemic
From handful of computers few years ago
Now literally count in billions
Days of ubiquitous computing not too far in future
Encountering embedded computers in daily lives
Increasing rate
Found in systems ranging from complex to mundane
Complex
- Aircraft flight control systems
- Sophisticated medical equipment
Mundane
- Washing machine
- Children’s toys
Failures in such systems
Can have significant and resounding impact

As embedded computing becomes more pandemic
Associated software
Safety in such systems
Becomes correspondingly more important

Safety
Because of growing importance of safety
Safety related issues
Want to spend some time to talk about subject

Safety and Reliability
People often confuse safety with reliability
Both are important issues
But are distinct concepts
Popular press confuses issue
By using terms in imprecise ways
Reliability

- Measure of up time or availability of system
- A reliable product
  - Begins at the specification and design stage
  - Not at the pre-production stage

Quality cannot be tested into a product
- No matter how good the test suite is

Reliability like performance
- Somewhat of soft measure
  - Not soft in sense it cannot be measured
  - Soft in sense of its meaning

In the general case
- Let’s consider several common sense descriptors
  - ✔ Product consistently performs in a manner in which the customer expects
    - For real time systems
      - Meeting all timing constraints
  - ✔ Mean time between failure is long
    - All physical things fail eventually
    - Want time before / between failure to be as long as reasonable
      - Testing nuclear explosion or atomic accelerator
      - Long time measured in fractions of second
      - Automobile
        - Quoting 100 k miles or more on certain parts
    - What we really want to say is we want MTBF to be long
      - In context of useful life of product
  - ✔ When fails product fails gracefully and safely
  - ✔ Product responds in deterministic way
  - ✔ Product responds or fails gracefully
    - For out of bounds or unexpected inputs
    - Recovers if possible

One formal definition of reliability
Probability that failure of system less than some threshold

Let’s amend this definition
With intelligent and self diagnosing / correcting systems
They can detect anomalies conditions
Vote out
Reconfigure
Work around problem
Failures can occur but not be detected
Internet or other communication networks fail continuously
Recovery through retransmission
Occurs on casual basis
Let’s define reliability as
Probability of detected failure is less than some threshold

Safety
As noted
Safety distinct from reliability
Safe system is one
Does not incur too much risk to
People
Equipment
Risk
Any event or condition deemed undesirable
Can propose simple relationship

\[
\text{Risk} = \text{Probability of Failure} \times \text{Severity}
\]

Such undesirable conditions include
- Release of energy
- Release of toxins
- Interference with life support functions
- Supplying misleading information to
  Safety personnel
  Control systems
• Failure to alarm when hazardous conditions arise

Safety hazards occur in a system through
  Improper or unsafe specifications
  Failures in some component of the system

Faults and Failures
  Faults
    Failure – nature
    Error – us

*Failure* is an event occurring at a specific time
*Error* is static
  Inherent characteristic of system
  Result of design error for example

*Fault* is unsatisfactory system condition or state
Failures and errors different kinds of faults

Fault can affect our system in different ways
  • Action – inappropriate action taken or appropriate action not taken
  • Timing – actions taken at inappropriate time
    Too early
    Too late
  • Sequence – skipping action or executing out of sequence
  • Amount – inappropriate amount of energy or reagent used

Fault may be random or systematic
  These are different
When failure occurs in system
  System that once functioned no longer does so
    Usually occur in field when
    Hardware part breaks or wears out

Random faults can only occur in physical entity
  Electrical or mechanical component
  Cannot be designed away
Can attack through
  • Redundant components
  • Sound design practices

Software fault always systematic
Software does not break or wear out

Single Point and Common Mode Failures
When we design a system we need a metric
Since many possibilities of fault

Chose to utilize / design against what is called single point failure
Must ensure that
  • Failure of single component
  • Failure of multiple components
    Due to single failure event
    Does not lead to unsafe condition

Consider following situations
  1. Same CPU used for
     Control
     Alarming or shut down
  2. Watch dog timer
     Same clock as CPU
  3. Common power supply
     Control
  4. Common Bus
     Control
     Alarming or shut down

In all these cases
Single point failure can lead to hazardous situation
Case 1
If CPU fails in primary task
Will be unable to invoke alarms or shut down safety components. Should be designed as separate system. Failure of either main or safety CPUs will not create single point failure. Failure of both units is independent and implies multiple failures. Still avoids single point failure.

Case 2
Watch dog timer like dead man’s throttle. If not reset on regular basis, forces system into fail-safe state. If shared clock, failure of clock prevents CPU from resetting time. At same time, prevents timer from advancing. Independent clocks are similar to independent CPUs. Simultaneous failure of two clocks implies multiple failures. Not considered single point failure.

Case 3
Often times in designing safety critical systems, safety components powered by what is called an UPS, uninterruptible power supply. If common power system, single point failure can be significant problem.

Some Solutions
We typically attack hardware and software safety independently. No matter which approach we use – independent of hardware or software – must begin as one of original system requirements. Cannot incorporate just before release to production.
Software
Boundaries
Will incorporate boundary tests
Confirm data remains within upper and lower bounds
Need to decide what to do if such bounds exceeded
- Hold at max or min value
- Alarm
- Combination

Points A, B, and C in figure
At low and high boundaries
A and B remain within range
C exceeds upper limit and has value locked at max

Damaged Data
Damaged data
Assumes data has been corrupted somehow
Noise
Electro magnetic interference (EMI)
Crosstalk within system

Questions
There are several questions we can ask about errors

Detectability
Detectable
Those errors that can be distinguished from legitimate data
Undetectable
Those errors that masquerade as legitimate data

Extent
Single bit
Several bits
Group of bits
Substantial

Detectability
Faulty data is said to contain a detectable error if it can be
Distinguished from legitimate data
One measure of our ability to detect corrupted data
Related to the distance between legitimate words in the language
The number of bits that must be changed to map
One data word into another
Denoted the Hamming distance
The larger the Hamming distance between legitimate words
The greater the extent of the damage before an error is
Incorrectly interpreted as correct
Consider accompanying diagram

Extent
From the outset
Data is subject to the vagaries of the environment
We see such errors may comprise
Single / Several Bit Errors
Occur when a bit is
Dropped / added
Transformed from one state to another

Group Bit Errors
Occur when a group of spatially close bits are corrupted
Referred to as burst errors

Substantial Bit Errors
Occur when there is a major loss of integrity in the physical link
We can address errors in the first two categories in a number of ways
Little can be done for those in the third

Response
When an error occurs
Detect
We can detect it signal an error
Detect and Correct

Depending upon

- The extent of the error
- The cost of the correction algorithm
- Time
- Added bits

**Detection**

**Parity**

The simplest form of detection
For each data word or block

Single bit is appended to the

Data word or block

**Odd Parity**

- Total number of bits is odd
- Data word has odd number of bits - add a 0
- Data word has an even number of bits - add a 1

**Even Parity**

- Total number of bits is even
- Data word has odd number of bits - add a 1
- Data word has an even number of bits - add a 0

When data used
Parity recomputed and compared with the expected
Mismatch indicates error

**Advantage**
- Simple
- Detects all odd numbers of errors

**Problem**
- Even number of errors undetectable
- Does not work reliably for burst
- Detection only

**Block Check Sum**

Simple extension of the single parity bit idea
Applies to complete block or frame
Process
Each individual word is checked as before
Referred to a lateral or row parity bit
Parity word computed for the block
Bit 0 - Parity of bit 0 of all words in block
Bit 1 - Parity of bit 1 of all words in block
Bit n - Parity of all lateral parity bits in block
Referred to as longitudinal or column parity
Variations
May use odd parity in one direction and even in the other

Advantage
Simple
Detects
All single bit errors
Many double bit errors

Problem
Simple errors can still escape detection
Does not work reliably for bursts
Detection only

Cyclic Redundancy Check - CRC Codes
At root of the approach
Notion of a polynomial
Associated with binary sequence of bits.

Polynomial over a Galois Field of two elements - GF(2)
Written in terms of a dummy variable usually x
\[ f(x) = \sum_{i=0}^{n-1} a_i x^i \]

For the set of bits [1011]
\[ a_3 = 1, a_2 = 0, a_1 = 1, \text{ and } a_0 = 1 \]
Corresponding polynomial
\[ f(x) = x^3 + x + 1 \]

According to Galois field arithmetic
Can define addition and multiplication as

<table>
<thead>
<tr>
<th>Addition</th>
<th>Multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 + 0 = 0</td>
<td>0 • 0 = 0</td>
</tr>
<tr>
<td>0 + 1 = 1</td>
<td>0 • 1 = 0</td>
</tr>
<tr>
<td>1 + 0 = 1</td>
<td>1 • 0 = 0</td>
</tr>
<tr>
<td>1 + 1 = 0</td>
<td>1 • 1 = 1</td>
</tr>
</tbody>
</table>

Goal of CRC codes
Add sufficient redundant information
To an encoded transmitted word
That can detect or detect and correct a specified number of bit errors
In received word

Approach
- Start with a binary sequence of length k
- Append n-k additional parity bits

Redundancy

Result
\[[d_{k-1}, d_{k-2}, \ldots d_1, d_0, r_{n-k-1}, r_{n-k-2}, \ldots r_1, r_0]\]
Expresses payload plus redundant bit

Can be written in polynomial form
\[m(x) = \sum_{i=n-k}^{n-1} d_i x^i + \sum_{i=0}^{n-k-1} r_i x^i\]

Equivalently
\[M(x) = x^n D(x) + R(x)\]

Where M(x), D(x), and R(x) represent the three polynomials.

In equations as observed
d_i contains message and r_j contain redundant bits
Can take advantage of such an equation as follows

Let
- \( n \) and \( k \) be integers, \( k > n \)
- \( D(x) \) be a \( k \) bit binary number expressed in polynomial form
•\ G(\text{x}) be an (n-k) bit number expressed in polynomial form
•\ R(\text{x}) be an (n-k) bit number expressed in polynomial form

We observe

1. D(\text{x}) is shifted to the left by n places
   \[ D(\text{x}) \cdot 2^n \]

2. Then divided by G(\text{x}) to give a quotient and a remainder
   \[ \frac{D(\text{x}) \cdot 2^n}{G(\text{x})} = Q(\text{x}) + \frac{R(\text{x})}{G(\text{x})} \]

3. Remainder portion is added to both sides of equation
   \[ \frac{D(\text{x}) \cdot 2^n + R(\text{x})}{G(\text{x})} = Q(\text{x}) + \frac{R(\text{x})}{G(\text{x})} + \frac{R(\text{x})}{G(\text{x})} \]

4. Simplify
   \[ \frac{D(\text{x}) \cdot 2^n + R(\text{x})}{G(\text{x})} = Q(\text{x}) \]

The final equation holds

Assuming modulo 2 arithmetic

That is we form a managed outgoing message by

1. Taking a binary number D(\text{x}) and shift it to the left by n places
2. Dividing that number by G(\text{x})
3. Adding the remainder to the original left shifted number

The remainder R(\text{x}) / G(\text{x})

Expresses the CRC

The redundant bits

The number computed in step 3

Represents the CRC encoded data word that is transmitted

On receipt
The received word is divided by $G(x)$
If remainder is zero there was no error

The number $G(x)$
Called a generator
Is not arbitrary a lot of care must be taken in its selection
Choice determines the number and type of errors
That can be detected

It can be shown with the proper choice of generator one can detect
All single bit errors
All double bit errors
All odd number of bit errors
All error bursts $< n+1$
Most error bursts $\geq n+1$

Similar techniques can be used to detect and correct
Multiple bit as well as certain families of burst errors
Typically error correction relies on re-transmission
Rather than computation by receiver

Encoding to develop redundant bits implemented using a simple circuit
Such as the one given in following block diagram

Illustrate process with following example
Design will utilize four bit data word
Sets the value of $k$ to 4
If we wish to correct all single bit errors
Hamming distance must be 3
Three additional bits will be sufficient
Gives encoded word length $n$ will be 7

From a table of generator polynomials
Select the following polynomial
$G(x) = 1 + x + x^3$

\[
G(x) = g_0 + g_1x + g_3x^3
\]
\[
' g_0 == 1, g_1 == 1, g_2 == 0, g_3 == 1
\]
Generator implemented as

![Hardware diagram]

**Hardware**

Major components on hardware side
- Control
- Alarms
- Power
- Interconnect

We can build safety into system
- Large or small scale
  Typically combination of both

Large Scale
- Large scale approach entails working with hardware architecture

Highly reliable systems fall into three classes
Based upon their capabilities following failure

- First are called *fail operational*\(^2\)
  Read as *fail operational squared*
  Such systems can tolerate two single point failures
  Continue to operate with full capability

- Second group are called *fail operational*
  Tolerate one single point failure
  Continue to operate with full capability

- Third group can continue to operate
  At diminished capability

Approach to designing such systems actually rather simple

1. Provide redundant paths of control
2. Separate critical elements

*Fail Operational*\(^2\) / *Fail Operational Capability*

At the component level *fail operational*\(^2\) capability

Commonly involves some form of voting scheme

Implementation comprises

An odd number channels of control

Voting protocol

To decide on validity of incoming signals and outgoing controls.

The basic case denoted *triple module redundancy*, TMR

Shown in following diagram

Figure focuses on

Potential failures in the three channels

Not in the voter or alarm

Clearly both the voter and alarm

Vulnerable to single point failures

Implementation comprises

Odd number of channels of control

Voting scheme to decide on validity of

Incoming signals
Outgoing controls
Uses m out of n vote to decide
Odd number of control channels ensures
Cannot have a tie
Approach may implement channels with

Same Design
All designs physically identical
Make sure don’t have single point failure
That can take out all channels
Advantage
Improved robustness of system
Minimal design cost
Additional systems
Copies of original design

Disadvantages
Common failure mode
For example
Design flaw or error
Susceptibility to certain kinds of input errors
Thus cannot avoid such errors
One channel affected – all affected
Affects all systems

Alternate Designs
Implements redundancy by
Replicating systems
Each design is different
Advantage
Common failure or design flaw
Not replicated in all channels of control
Can detect / respond to certain kinds of input errors

Disadvantage
Cost
Implemented in several different ways

Functionally systems the same
  All channels implement same behaviour
  Done in different ways
    Generally different hardware and software
  Most expensive of 3 alternatives

From high level
  Same as we saw earlier
  Input sensors may be duplicated as well
  Potentially different also

**Reduced Capability**
Implementation with reduced capability under failure
  Support critical subset of the output signals
  Adds alarm capability

- Lightweight redundancy
  Subsidiary channel monitors primary
    Intent is to identify gross failures
  Focus is on fault detection rather than tolerance
  Lightweight channel (of control)
    Cannot completely fill in for main channel
    In event of failure
  Block diagram appears as

- Separation of monitoring and actuation
  Function to be performed
  Subdivided into two major kinds of operations
    1. Actuation / Control
    2. Monitoring
  Control channels charged with
    Implementing actions
  Monitor channels responsible for
Keeps track of what actions to be taken
Monitors physical environment
To ensure results of actions
Appropriate and correct
Sensing utilized by monitor channel
Separate from that of control channel
Control channel sensors
Dedicated to that job

Objective
Monitor channel identifies control channel failures
Informs appropriate fault-handling mechanisms
Failure in monitor channel
Doesn’t affect control channel
Still must identify and eliminate
Points of single failure
Two channels
Exchange messages
Ensure operation of each
We now have a block diagram such as

Steps to a Safe design
Building safe systems
Complex task
Supporting mechanisms
Can account for most of complexity of a system
Designing and implementing such systems
Most important task in development
Contemporary embedded and embedded real-time systems
Can capture significant elements of the process as follows
1. Identify the hazards
2. Determine the risks
3. Determine the safety measures
   Additions to product to handle hazard
4. Create safe requirements
   Done through requirements specification
5. Create safe designs
6. Implement safety
   - For example strong vs. weak type checking
   - Compile time vs. runtime errors
   - Properly handle exceptions and errors
7. Assure the safety process
   - Here we have agencies and standards
     - ISO 9000
     - FDA Good Manufacturing Process (GMP)
     - German agencies
8. Test the system

First 3
   - Must be completed prior to finalization of system requirements
   - These lead to hazard analysis of system
     - Feeds into system specification
Steps 4 through 6
   - Design safety into system
Step 7
   - Assures standards met
Step 8
   - Test to ensure that hazards handled safely

As we design our systems
   - Must remember we can’t stress safety too much
We’ll move on next to the design process
   - Look at steps involved