Integrated Modular Avionics (IMA) Requirements and Development

Kevin Driscoll

November 13, 2007

What is Architecture versus Design?

Architecture is to Design

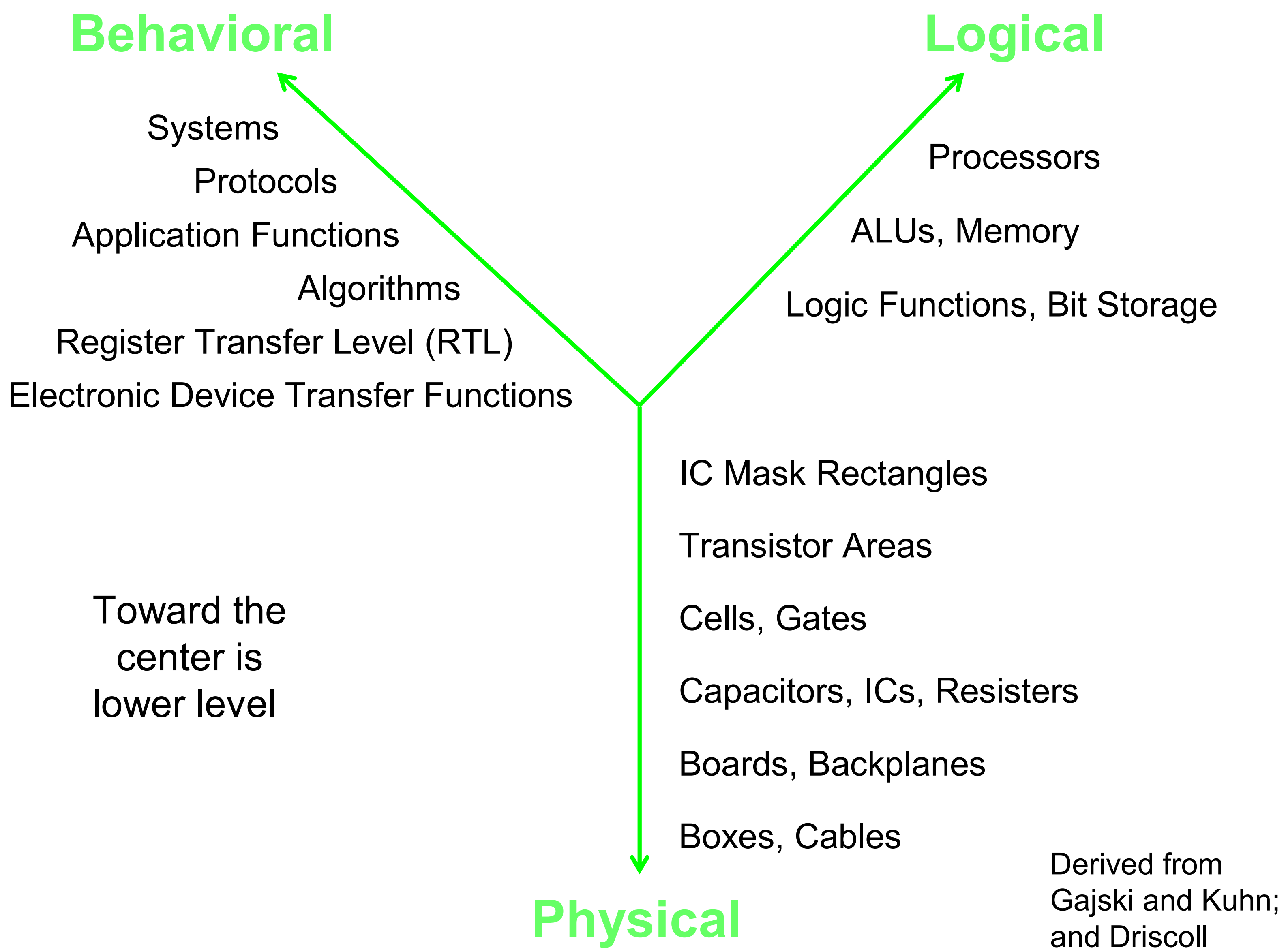
as

Constitution is to Law

➔ a set of general rules, principles, and constraints used as guidance for making design decisions

Design has three orthogonal (independent) aspects.

The Three Orthogonal Aspects of Design



Derived from Gajski and Kuhn; and Driscoll

Key features of a federated architecture

- Function has guaranteed access to the processor
- Deterministic and guaranteed access to I/O
  - Controlled Latency and Jitter
- Critical functional separation
  - Low-criticality functions cannot corrupt critical functions (loosely coupled separate LRUs)

Drawbacks with a federated architecture

- All functions that share a processor must certify to the highest criticality level of those functions
  - Encourages many processors with lower utilization
- Inefficiencies in use of resources
  - Modern processors and memory devices have far more capability than a single critical function usually needs (However, we are reaching the “Hubble Radius”.)
  - Multiple power supplies, networks, I/O systems
  - Size, weigh, and of individual “boxes” (LRUs)
- Proliferation of part types
  - Each subsystem tends to design a point solution optimized for its performance and I/O needs
- Increased weight, power, and wiring

Why Integrate

- Goal: An architecture that provides all of the benefits of a federated approach, and solves the problems
  - Partitioned software ensuring that only the critical software is developed and validated to critical requirements
    - Approach lowers risk by reducing the amount of critical software
  - Resources are shared
    - Better approach for future growth - spare capacity can be used for function extensions, or for new applications of any criticality level
  - Hardware part types are reduced

Requirements Beyond Designers' Experience

- A typical design engineer has << 10,000 (10^4) hours of real hands-on hardware experience over an entire career
- Typical safety critical avionics requirements are equivalent to 1,000,000 (10^6) to 10,000,000,000 (10^{10}) hours MTBF (100 to 1,000,000 times smaller than human experience)
- When a designer feels that something can't happen, that means a probability << 10^{-4} / hour.
- This is so far from the requirements as to be useless.
- It is not safe to rely on our intuition for reasoning about unfathomably small probabilities.

How Systems Fail

- **Assumed importance order**
  1. Exhaustion of resources (triplex versus quad)
  2. Single point of failure
     – unknown failure mode
     – forgotten failure mode
     – underestimated probability of occurrence
  3. Chain or domino effect (fault containment)

- **Real occurrence frequency order**
  1. Chain or domino effect (fault containment)
  2. Single point of failure
  3. Exhaustion of resources

Determinism

- Determinism is the characteristic of a system which allows the correct prediction of its future behavior given its current state and knowledge of future changes to its environment (inputs).
- Non-determinism means that a system's future behavior cannot be correctly predicted. (An unpredictable system cannot be called "safe.")
- How precisely must the behavior be specified?
- Characteristics of behavior:
  - –Value
  - –Ordinal timing
  - –Cardinal timing

Dangers of mixed criticality

- What could a non-critical function do to a critical one?
  - Erroneously write data into wrong areas
  - Steal time / Interrupt the processor
  - Crash the processor
    - How do we deal with “Halt and Smoke”?
    - How do we handle thermal slow down??
      - Missed deadlines
      - Jitter (analysis explosion)
    - How do we handle thermally increased errors???
  - Corrupt I/O
    - Falsely send output data appearing to come from the critical function
    - Corrupt input data before the critical function uses it
  - Hog internal communications paths (e.g. backplane)

IMA Partitioning Requirements
Systems Integration Challenge

- Integrating multiple functions onto shared hardware opens up the potential that a failure in one function can lead to failures in others
  - Such failure paths would force all co-resident software to be certified to the highest integrity level
- ARINC 651 specifies a requirement called **robust partitioning**
  - No failure in a function (or in hardware unique to a single function) can cause another function to fail
- Motivations for Robust Partitioning
  - Can support mixed levels of criticality on shared hardware
    - Lower certification cost
    - Lower maintenance cost
  - Fault containment
    - Reduce cascading failures
- Partitioning is required in two ‘dimensions’
  - Space partitioning - protection of program, data, and dedicated I/O and registers
  - Time partitioning - protection of the processor and communications bandwidth assigned to a function

Space Partitioning

What constitutes space partitioning?

- Any persistent storage location (eg. data memory) must only be writeable by one function
- Any temporary storage location (eg. processor registers) used by a function must be saved when control is transferred

Typical ways to support space partitioning

- Memory management units (write protection, separate virtual memory spaces)
- High-integrity operating system

Space partitioning 'holes'

- I/O device registers
  - I/O device must either dedicated to one function, or time partitioned
- Backplane data bus
  - If bus can remotely address memory, software failure can lead to corruption of another function's resources

Time Partitioning

What constitutes time partitioning?

- A function's access to a prescribed set of hardware resources for a prescribed period of time is guaranteed
- The order of execution between communicating functions is consistent each execution cycle

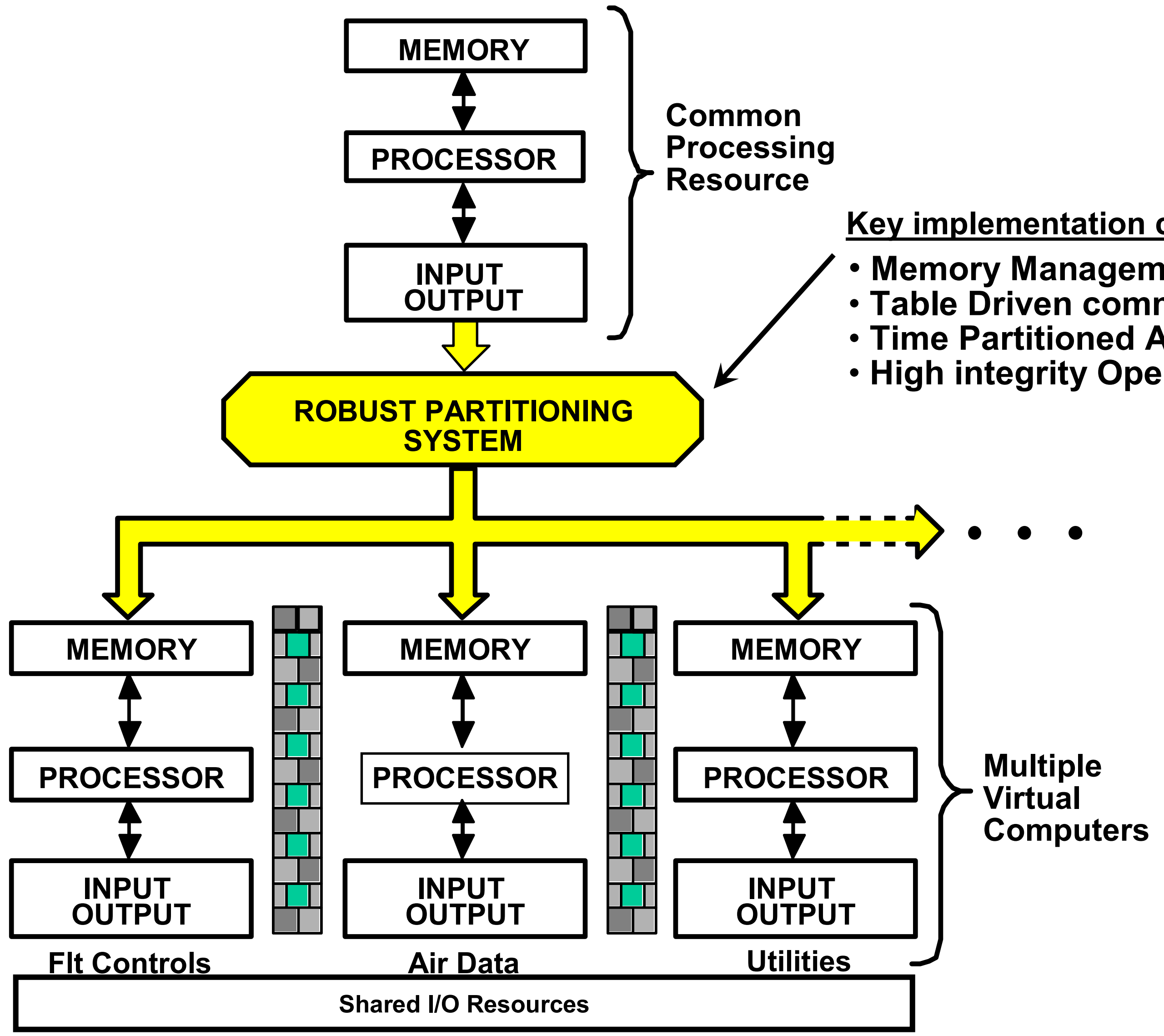
Typical ways to support time partitioning

- Deterministic scheduling (processor and communications)

Time partitioning 'holes'

- Arbitration for resources
  - Cannot guarantee that the order of events will be the same
  - Failed function stops arbitrating, or always arbitrates and can affect the timing of other functions
- Implementing deterministic scheduling across loosely coupled multiple processors

Virtual LRU Resource Partitioning Solution



Key implementation concepts

- Memory Management Unit
- Table Driven communications
- Time Partitioned Access to Shared Resources
- High integrity Operating System

Integrated Modular Avionics Requirements

Robust Partitioning

- No failure in a function can cause another function to fail

Reliability

- Aggressive no-maintenance policy requiring high dispatch probability forces high-reliability on all components

Fault Tolerance

- No single failure can cause loss of critical or essential avionics functions
- Dispatch with failed components to meet dispatch probability requirements

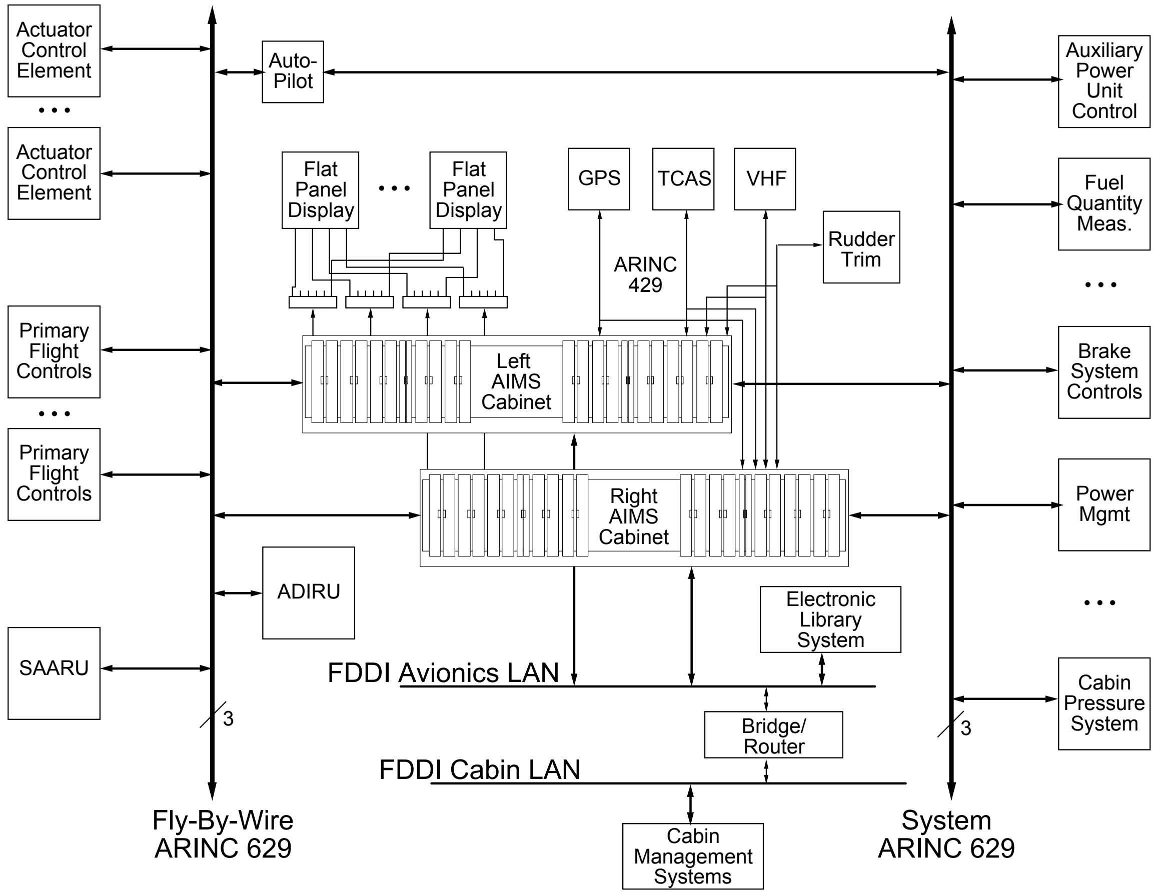
Flexibility

- Must support many module types of differing capability
- Must allow for addition/modification of cabinet functions **without** forcing recertification of unmodified functions

Easy to Debug and Certify

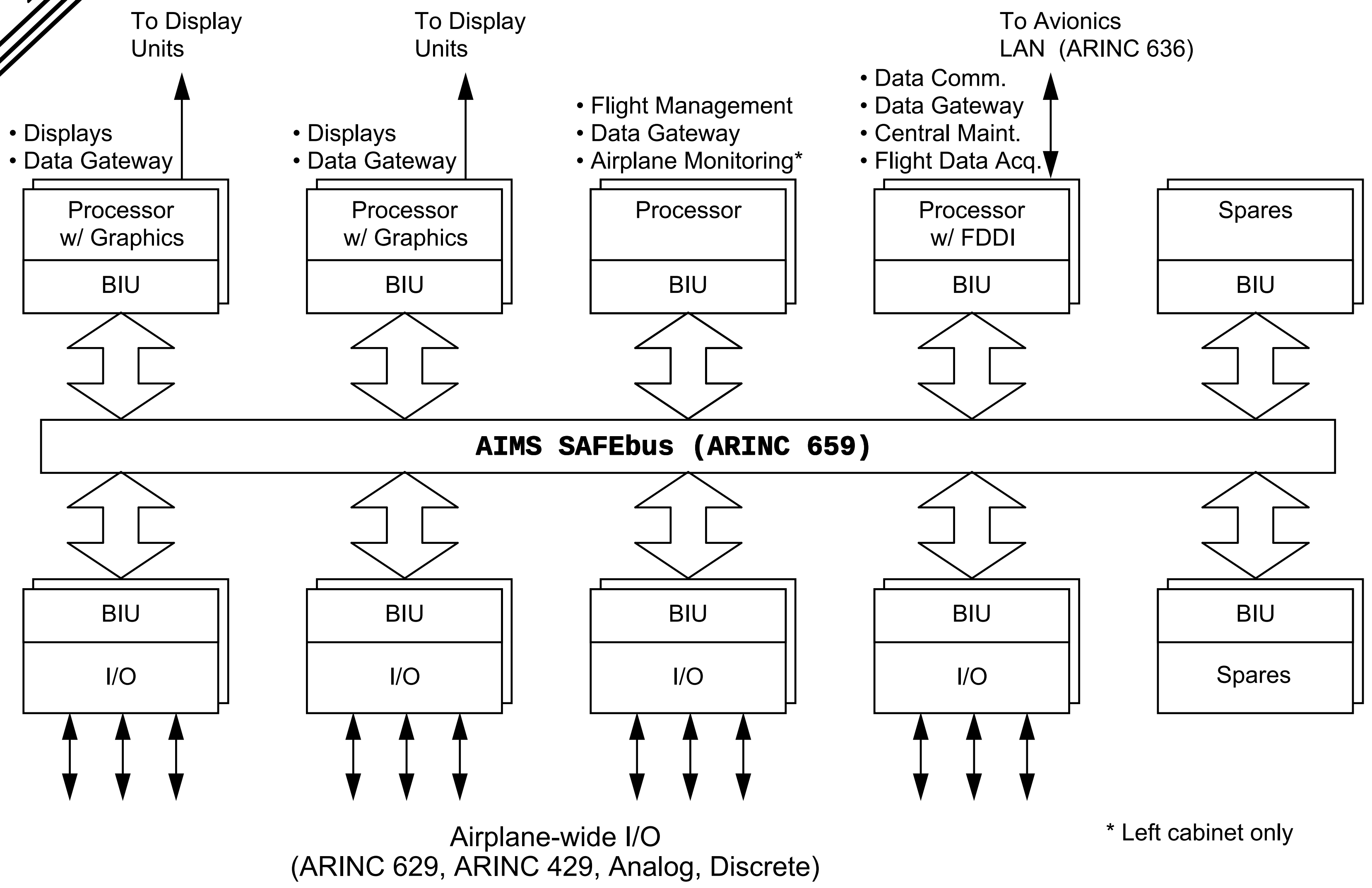
- Control of hardware/software integration
- Predictable behavior over all possible operating conditions

Boeing 777 Avionics Architecture



- A hybrid between a fully integrated and fully federated architecture
- A number of formerly federated functions integrated into AIMS
- Remaining units are assigned to one of three federated subsystems
  - Fly-by-wire (triple ARINC 629 bus connection)
  - System (triple ARINC 629 bus connection)
  - OLAN (ARINC 636, FDDI ring connection)
- AIMS acts as a gateway among these federated subsystems and other I/O
- Units replicated for safety and deferred maintenance

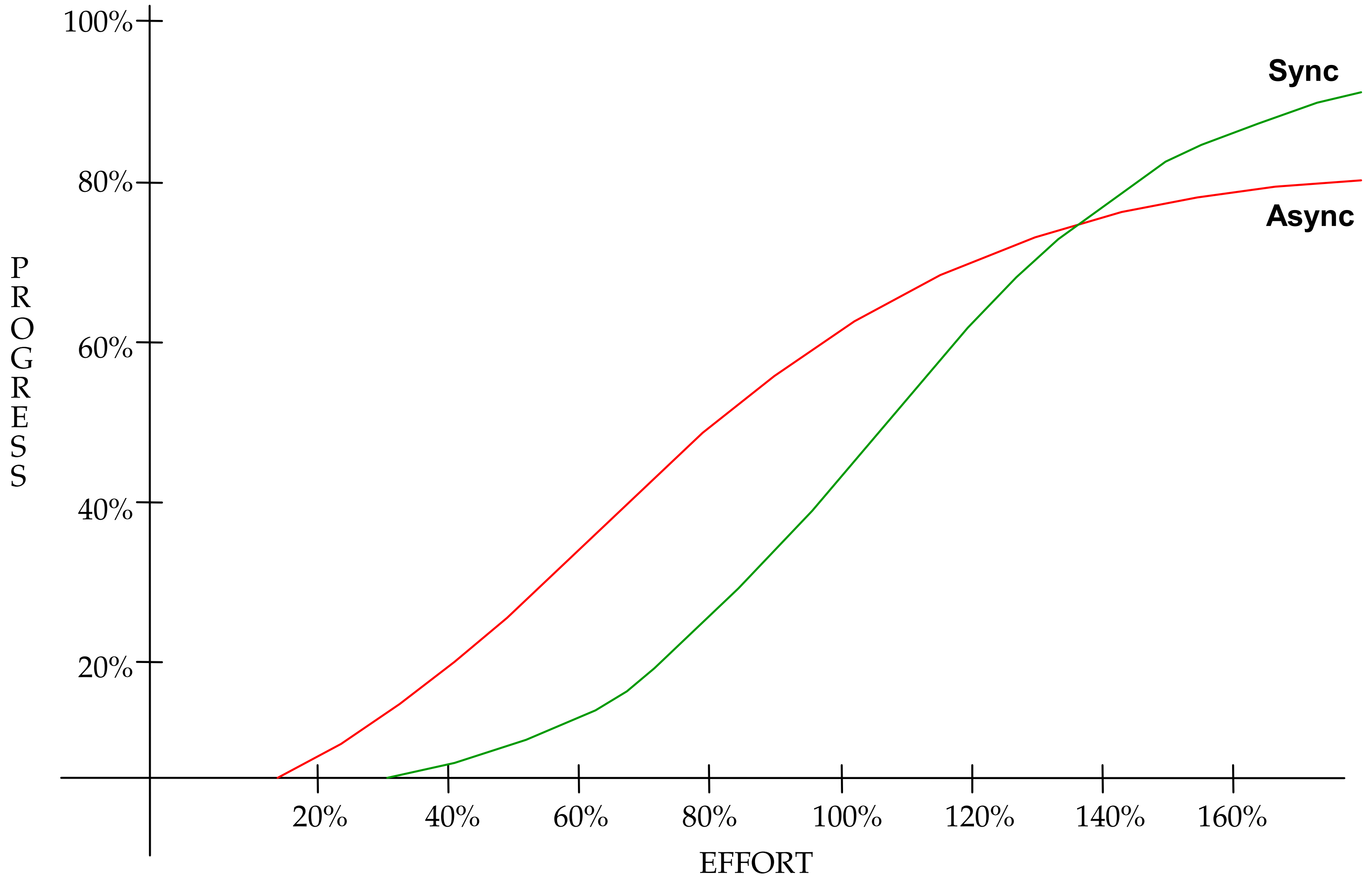
AIMS Function Mapping



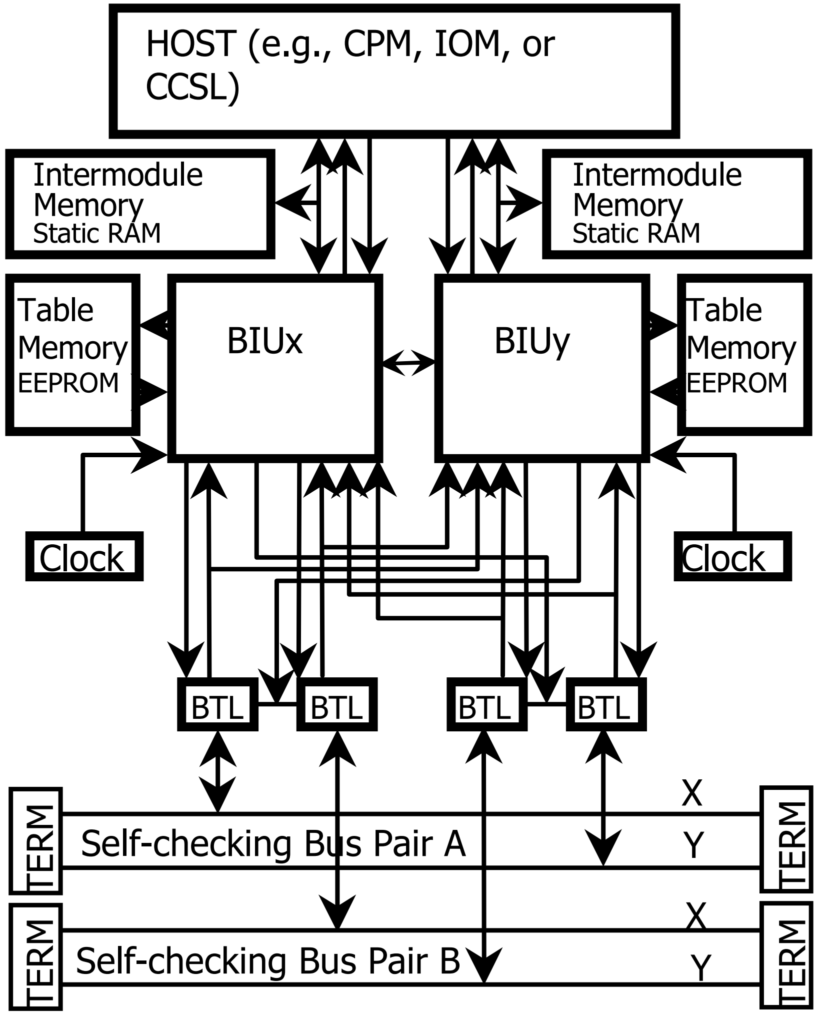
The Key Implementation Concepts

- SAFEbus® backplane
- Dual lock-step processor modules
- All modules synchronized to SAFEbus
- Memory management
- Operating system
- I/O architecture

Apparent Progress Versus Effort



SAFEbus

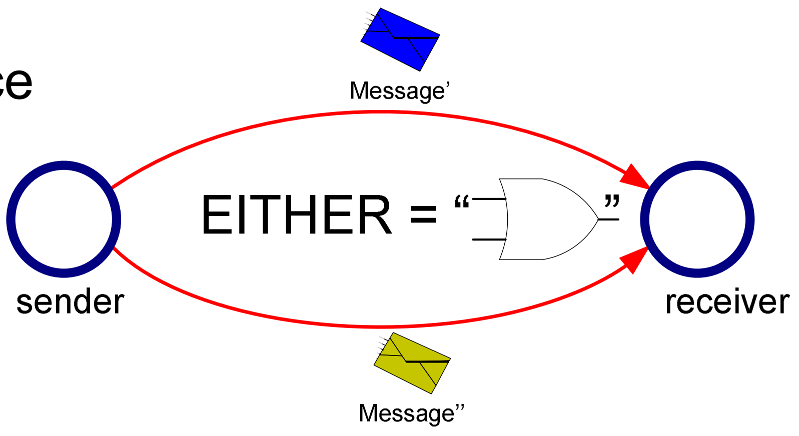


- Space partitioning
  - Shared RAM protected by Host MMU and BIU memory map in Table Memory that is writable only by “ground” IEEE-1149 bus
- Time partitioning
  - Enforced by commands in Table
  - Synchronization protocol
- Fault effects containment
  - Fail passive for BIU pair and related component failures
  - Four bus sets (Ax, Ay, Bx, By)
    - Correct all single bus failures
    - Detect all double bus failures
    - Correct some double failures
  - Each bus set has separate power
  - BIU is independently monitored

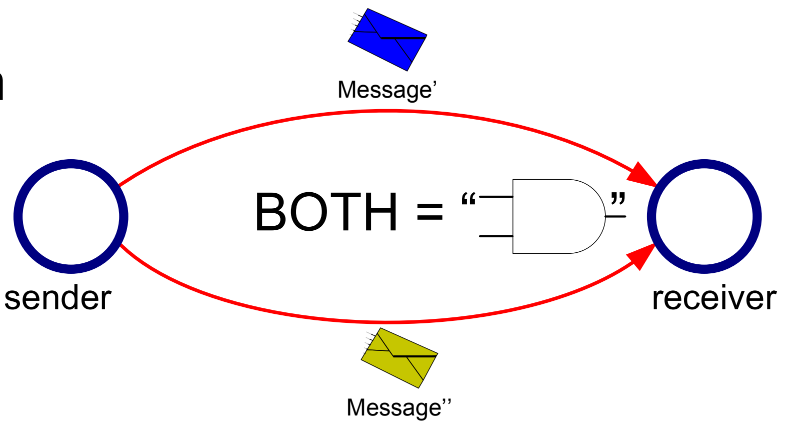
What Does Dual Redundancy Provide?

Example: communication integrity

- **Availability**
  - Readiness for correct service
  - If miscompare, arbitrarily select one
  - The "Availability OR"

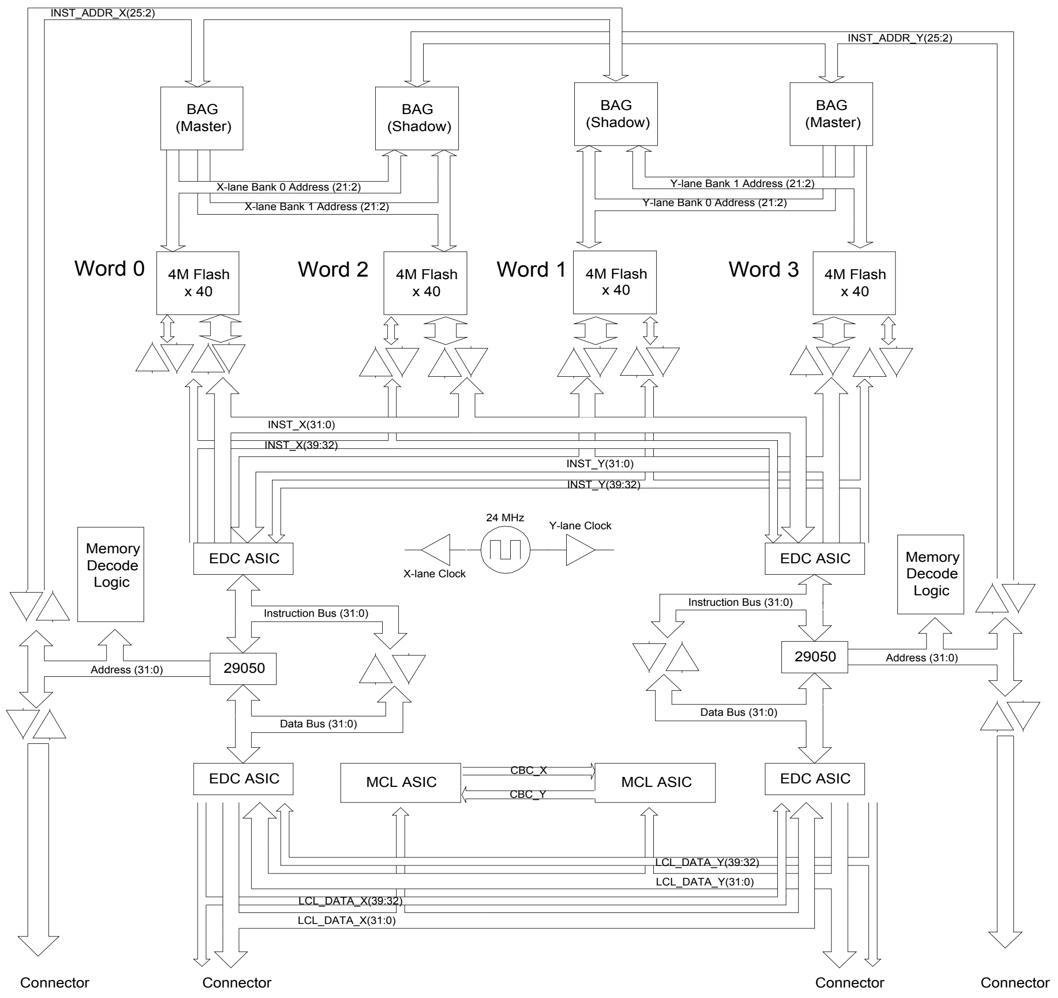


- **Integrity**
  - Absence of improper system alterations
  - If miscompare, reject both
  - The "Integrity AND"



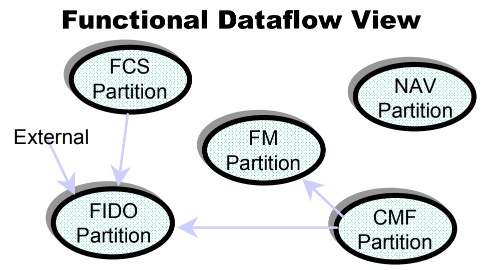
Simple replication gives you availability or integrity but not both!

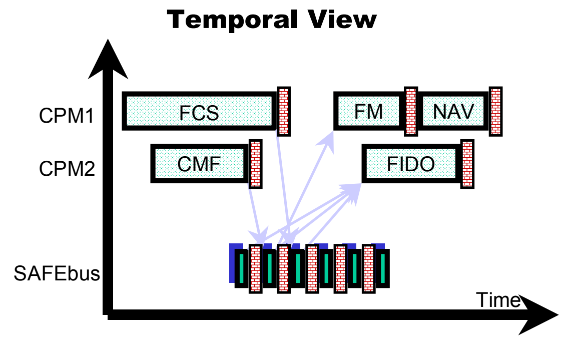
Lock-Step Processor Architecture

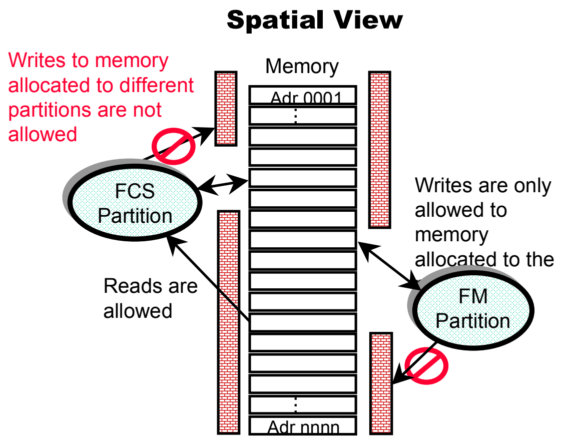


- Space partitioning
  - Protected system page tables
    - Constructed at build time
  - Validated MMU
- Time partitioning
  - Non-user maskable SAFEbus interrupt drives OS schedule
  - No other interrupts allowed
- Fault effects containment
  - Lock-step checking of all memory accesses (I and D)
  - EDC on all memory R/W
  - Monitored clock, power
  - Power up BIT, CBIT

Memory Management

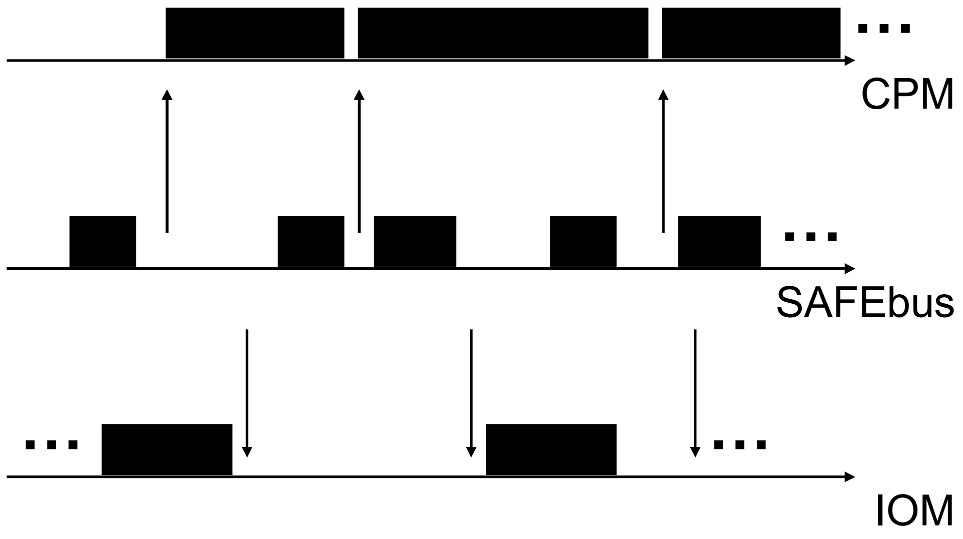





Spatial View
Writes to memory allocated to different partitions are not allowed
Memory
Adr 0001
Adr nnnn
FCS Partition
Reads are allowed
Writes are only allowed to memory allocated to the
FM Partition

- Memory Management Unit
  - Build-time tools define memory requirements of all partitions
    - System page tables for each partition
      - Statically checked
      - Integrity checked at run-time
    - Read access to IMM and static constants
  - SPT set is switched on partition context switch
    - Protected function
  - MMU enforces R/W permission on every memory access

System Scheduling

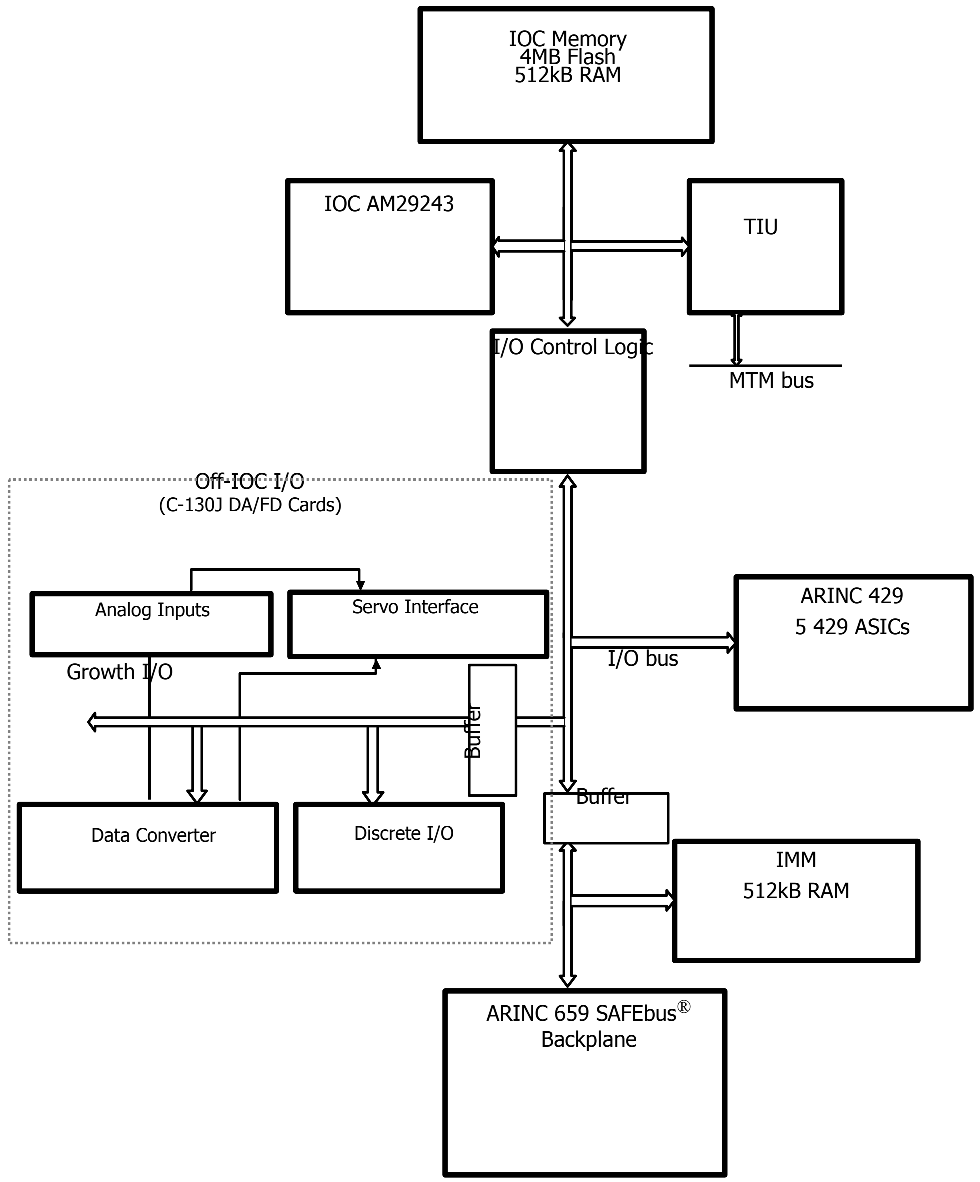


- Space partitioning
  - Manages MMU and page tables
- Time partitioning
  - Responds to SAFEbus time interrupts
- Fault effects containment
  - Detects deadline failures
  - Responds to partition attempts to write to illegal location
  - Rapid restart after power failure
  - Executes PBIT/CBIT
  - Manages fault recovery

Processor Fault Recovery

- Progressive Fault Recovery Design
  - Transparent Re-Try (If Possible)
  - Restart Partition or CPM
  - Shutdown Partition or CPM
- Time based strike counters used to determine recovery action
- Fault Recovery is dependent upon software executing during fault occurrence
  - CPM level recovery during O/S execution, Initialization or Loading
  - Partition level recovery during Partition execution and APEX service calls

IOM



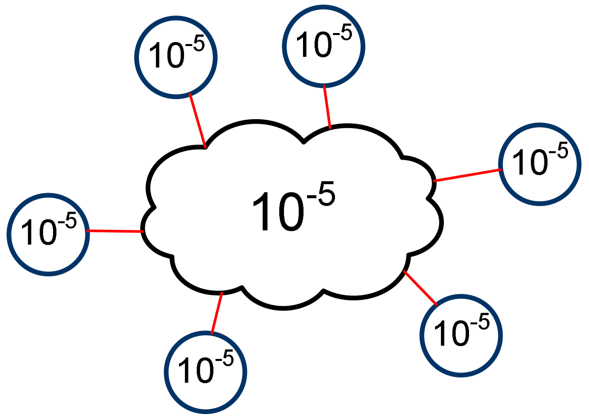
- I/O partitioning
  - Table driven protocols ensure that inputs are sampled and outputs are sent at specific times w.r.t. the SAFEbus timeline
  - Memory mapped transfer from processor to I/O module ensures user software cannot corrupt another function’s I/O
  - All I/O SW is level A
- Fault effects containment
  - Architecture supports dedicated wraps for high-integrity outputs

Summary

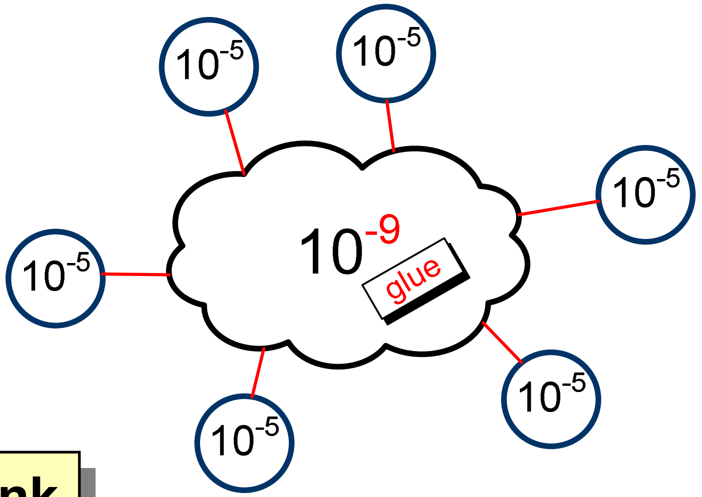
	Space partitioning	Time partitioning	I/O partitioning	Fault Containment
SAFEbus	• Addresses in tables • Split IMMs	• Sync protocol • Table driven	• Supports memory mapped transfer	• Self-checking • Error correcting
Processor	• Protected page tables • Validated MMU	• Only one interrupt (from SAFEbus)	• Through memory mapped IMM	• Lock-step • EDC • Monitors • BIT
IOM	• N/A	• Table driven ensures I/O timing	• Memory mapped transfer ensures routing • Level A SW	• Support for dedicated wraps

Why Is The Network So Important?

- Distribution needed to achieve independence
  - E.g. against
    - Impact of spatial proximity faults
    - Component failures
    - Power supply failures
- The network is the "Glue" of a distributed architecture
  - Connects replicated components
  - Information only as good as provider (component) and communication path
- Key for single source architectures



"Blurring" communication → replication becomes useless



Replication is useful

A chain is only as strong as its weakest link

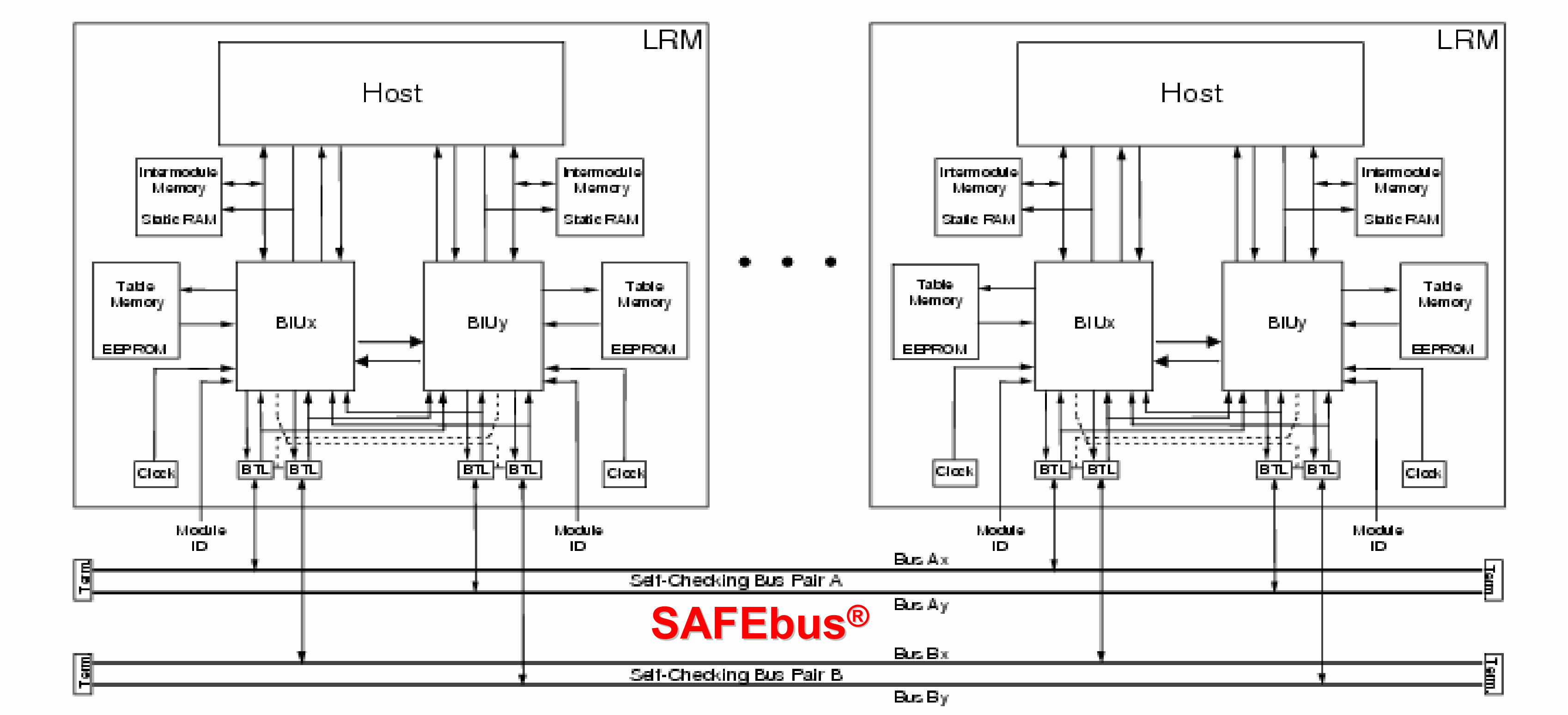
How can we make a reliable system out of notoriously unreliable computers and software?

How does one make a strong building out of sand?

Sand is a very weak construction material. One cannot make a roof or even vertical walls with it.

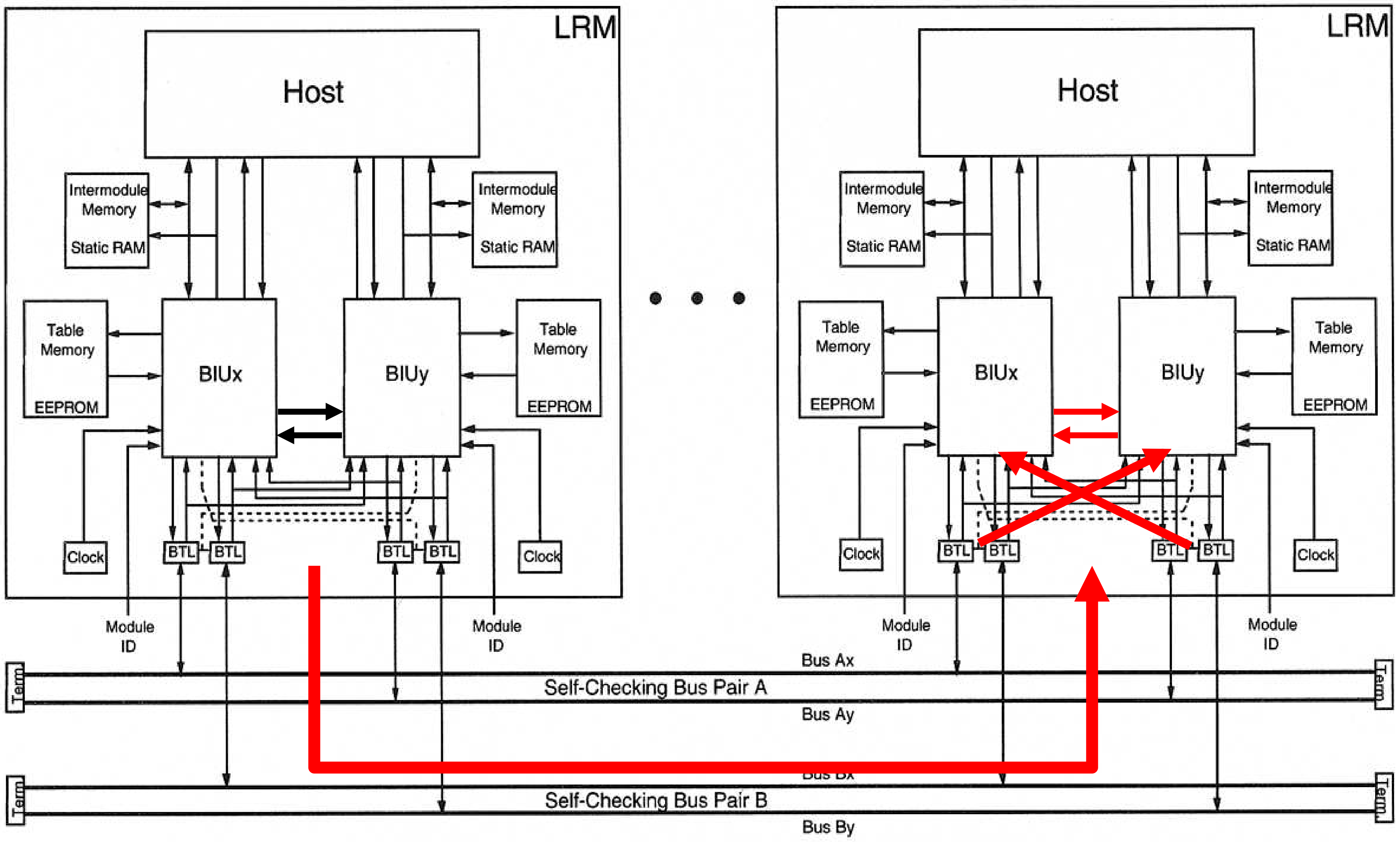
But, add some “glue” (cement), and you get one of the strongest and most commonly used construction materials in the world – concrete.

SAFEbus Fault Coverage and Containment



- Full-Coverage (near 100%), including Byzantine failure
- (C)Lock-stepping host processing – Self-Checking Buses
  - Medium availability via cross-coupled TX bus enable
  - Data integrity via bit-for-bit comparison at RX
    (local exchange for Byzantine fault containment)
  - BIU is firewall for all host failures (including SW)
  - Validated Fail-Silence Model

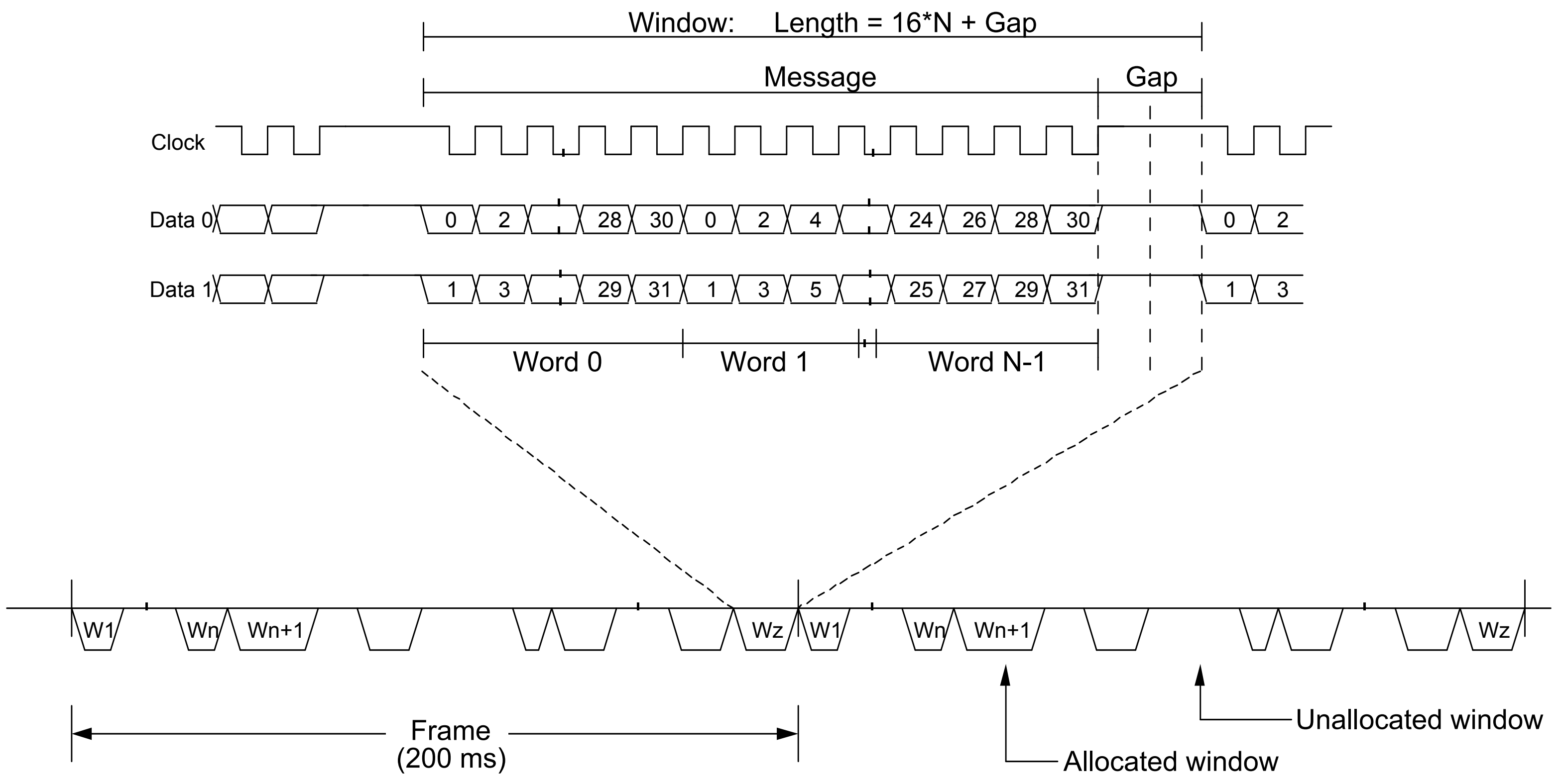
SAFEbus® Byzantine Fault Tolerance Passes



Three exchanges are done simultaneously: on bus, from BTLs to BIUs, between BIUs.

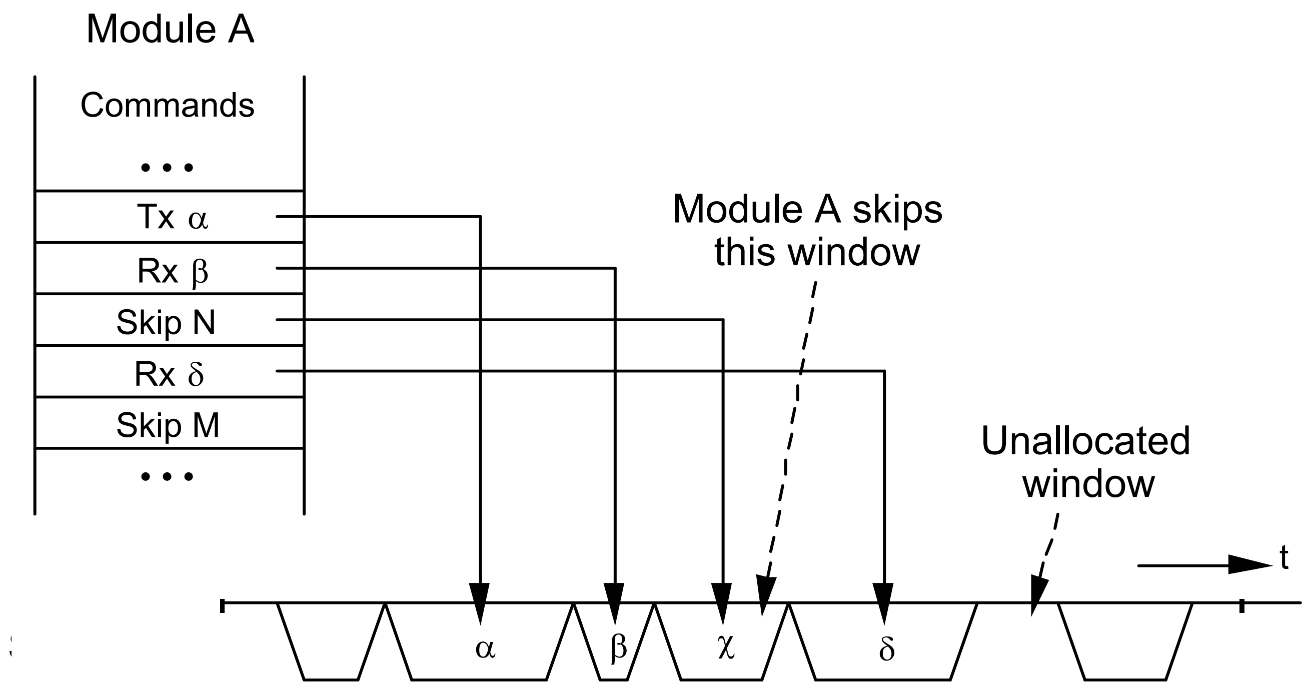
ARINC 659 Messages

Messages are pure data payload. There are no protocol bits in the message to waste bandwidth (SAFEbus is 8 times more efficient than Ethernet for average sized avionics messages). More importantly, there are no protocol bits in the message to get corrupted at the source or in transit, e.g. messages can't get misrouted.



ARINC 659 Table Memory

Table Memory contains commands that tell the BIU when messages are to be transferred, their source, and their destinations. The table memory can only be written by an IEEE 1149 maintenance network that is only active "on the ground" (system is safed).



ARINC 659 - Partition Synchronization

- ARINC 659 provides the mechanism to synchronize partition execution to backplane activity
  - Host interrupt command in the Table Memories
  - Interrupt code is provided to uniquely identify the event
  - Interrupt pattern is unique to each module
- This mechanism can be used to guarantee that data transmission is non-overlapping with the tasks that use the data

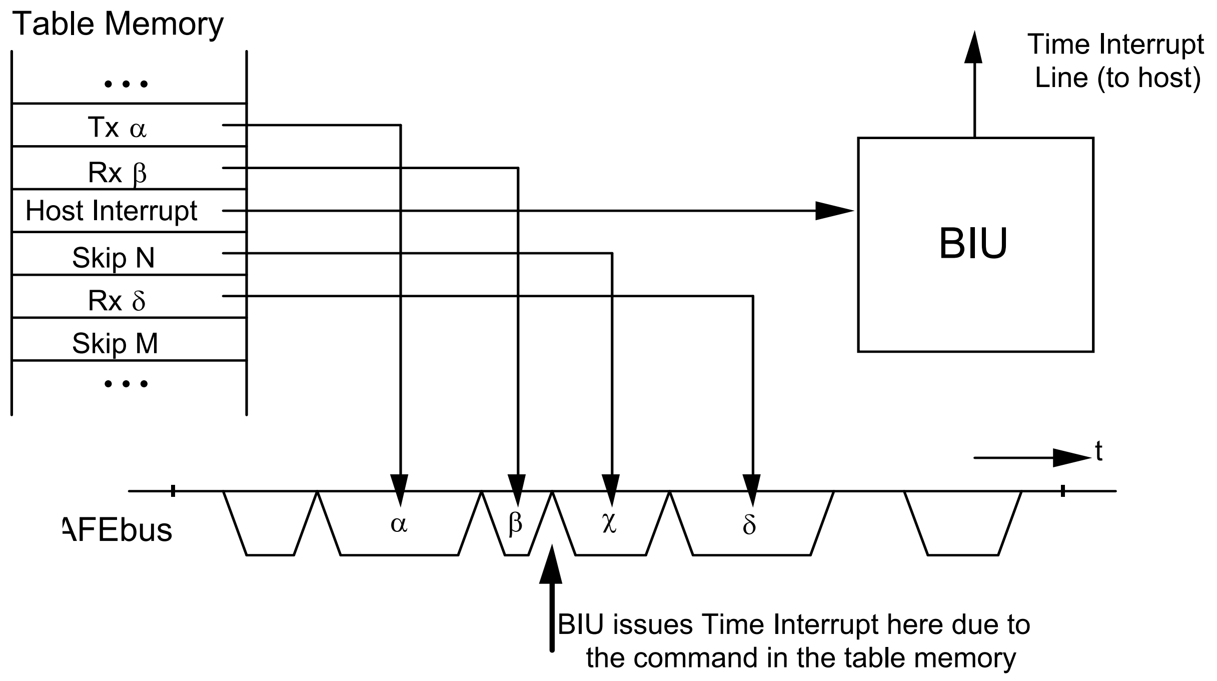
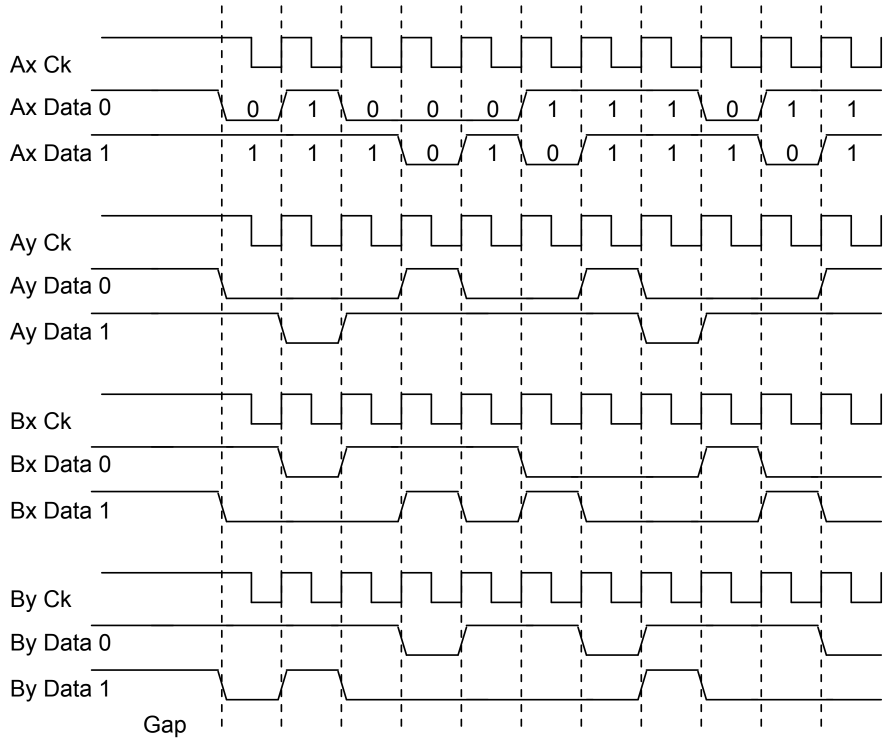


Table Memory
...
Tx α
Rx β
Host Interrupt
Skip N
Rx δ
Skip M
...
Time Interrupt Line (to host)
BIU
t
AFEbus
α
β
χ
δ
BIU issues Time Interrupt here due to the command in the table memory

Bus Encoding

- The bus lines are encoded as follows:
  - Ax - normal data (bus high = '1')
  - Ay = Ax XOR '010101...'
  - Bx = NOT Ax
  - By = NOT Ay
- Provides several key benefits:
  - Rapid detection of bus shorts or transient errors
  - Rapid detection of bus collisions (caused by faults)
  - Evens out power dissipation (both AC and DC)

Example:



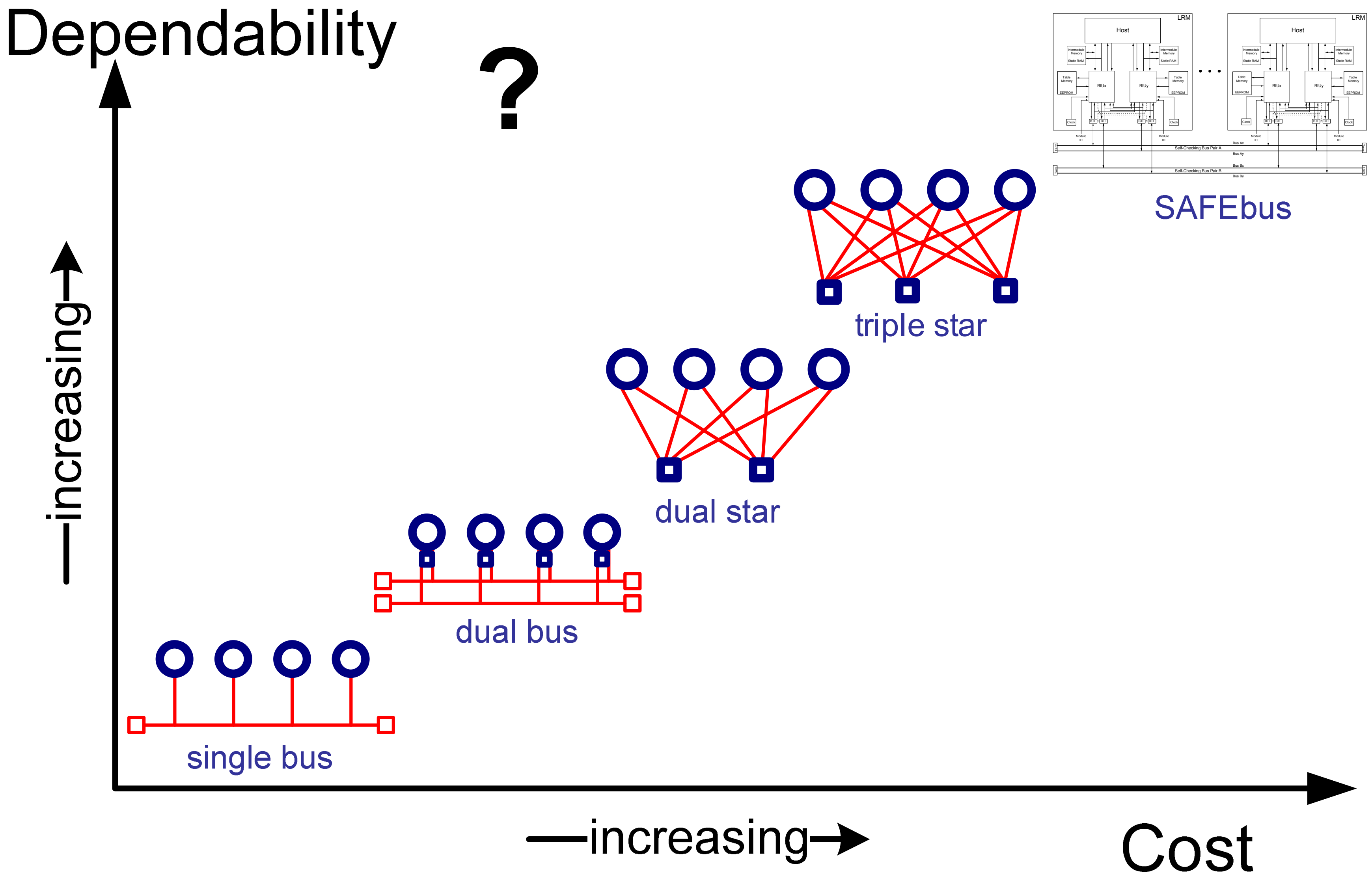
SAFEbus Scheduling

- Table generation supported through an off-line scheduler
  - Schedules both message transfer times and process execution times
- Software developers enter process execution, input and output requirements into database
  - Includes period, jitter, latency, etc.
- Off-line tool attempts to create efficient schedule which meets the user’s requirements
  - Essentially, resource contention is handled off-line
- Post-processing step verifies integrity of tables produced
  - Eliminates the need to verify heuristic search tool to critical status

SAFEbus Scheduling Statistics for proto-AIMS

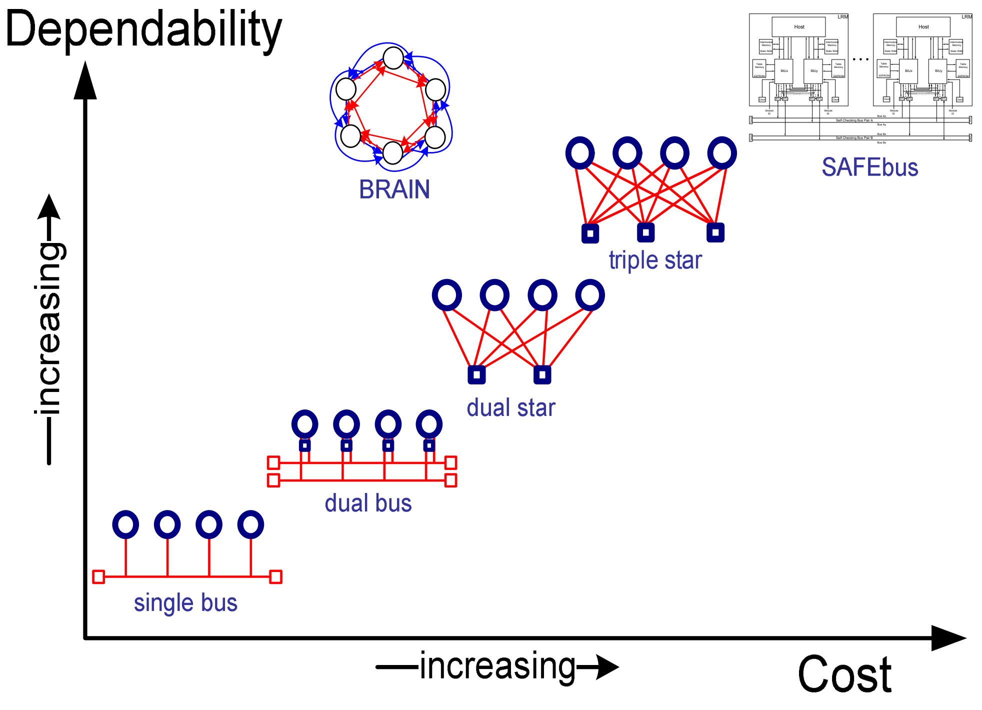
Partitions	18
Tasks	63
Data Items	6,000
Messages	17,000
Constraints	100,000
Decisions	100,000,000,000,000,000,000,000 (= 10^{23})

Can We Achieve Aerospace Dependability At Low Cost?



Braided Ring Integrity Availability Network (BRAIN)

- Cost-effective solution to address high dependability
- Decentralized guardian strategies
- Alternative to centralized stars and busses
- Applicable to existing network standards
- Response to various failures is flexible (availability vs. integrity)
- Simple protocol strategies can decrease guardian design complexity
- Self-checking pairs enable high data integrity and application software simplification
- Adopting aerospace rationale needn't drive increased costs



Detailed description and information in our paper at Dependable Systems and Network (DSN) Conference 2005, Japan