Real-Time & Embedded Systems Past, Present, and Future

Dr. Doug Locke

Locke Consulting, LLC

www.doug-locke.com

Outline

- What is a real-time System?
- What is an embedded System?
- Where have we come from?
- What have we achieved?
- Where are we going?

What is a Real-Time System?

1. Correctness is a function of time?
2. Must respond to external device in less than X microseconds?
3. Real-fast?
4. Missed deadline means catastrophic result?
5. System should respond “instantaneously”
6. All of the above?
7. None of the above?

What is an Embedded System?

- Small device, like a cell phone?
- Small processor installed in some other device, like a car?
- Software that controls a consumer device?
- Must have real-time response?

My favorite:

- Any system where the user doesn't want to know that it includes a processor

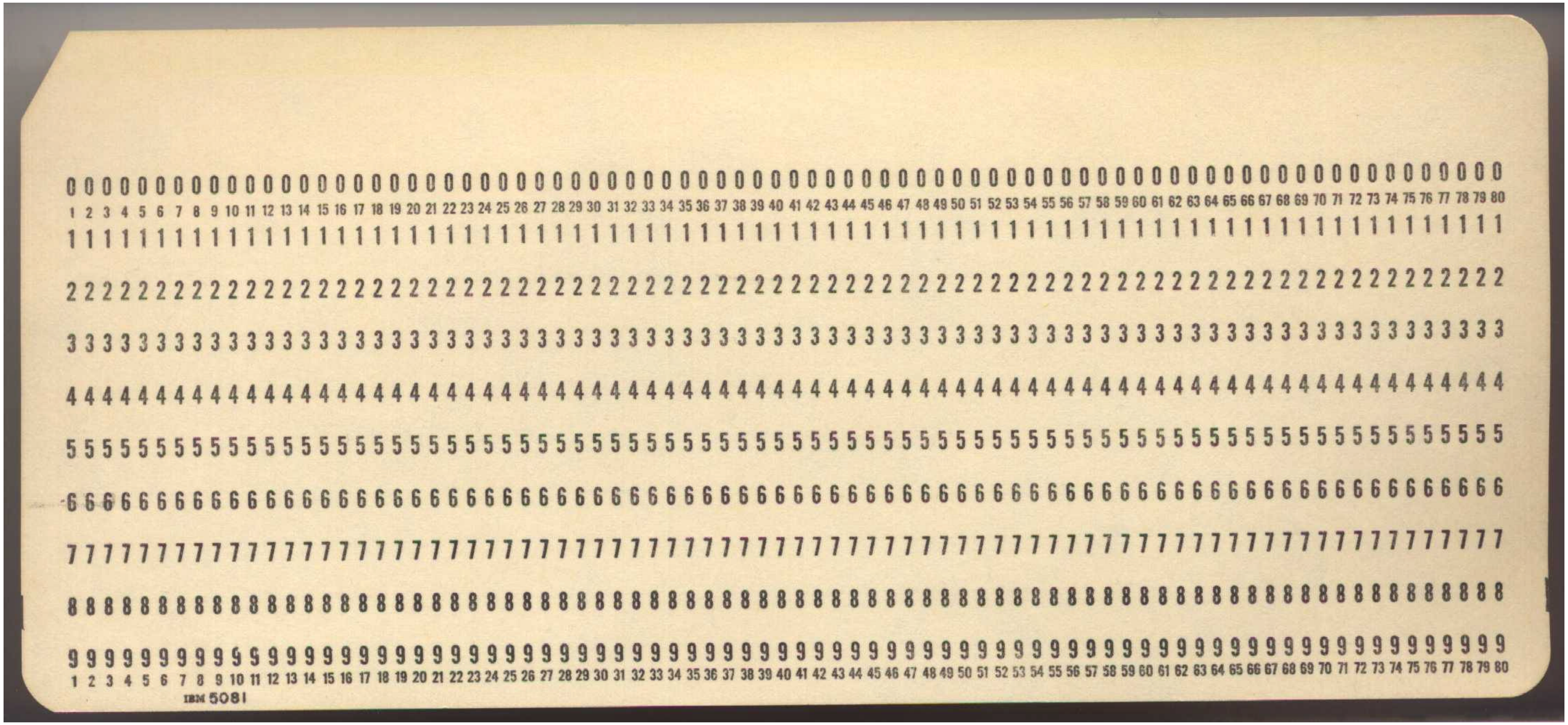
Examples of Real-Time / Embedded Systems

- Car engine
- Cell phone
- Set-top box
- Car navigation
- Industrial control
- Telecom switch
- Global Positioning System
- Air Traffic Management
- Satellite flight manager
- Satellite Ground Control
- TV receiver
- Flight control
- Electric shaver
- Toaster

Outline

- What is a real-time System?
- What is an embedded System?
- ü Where have we come from?
- What have we achieved?
- Where are we going?

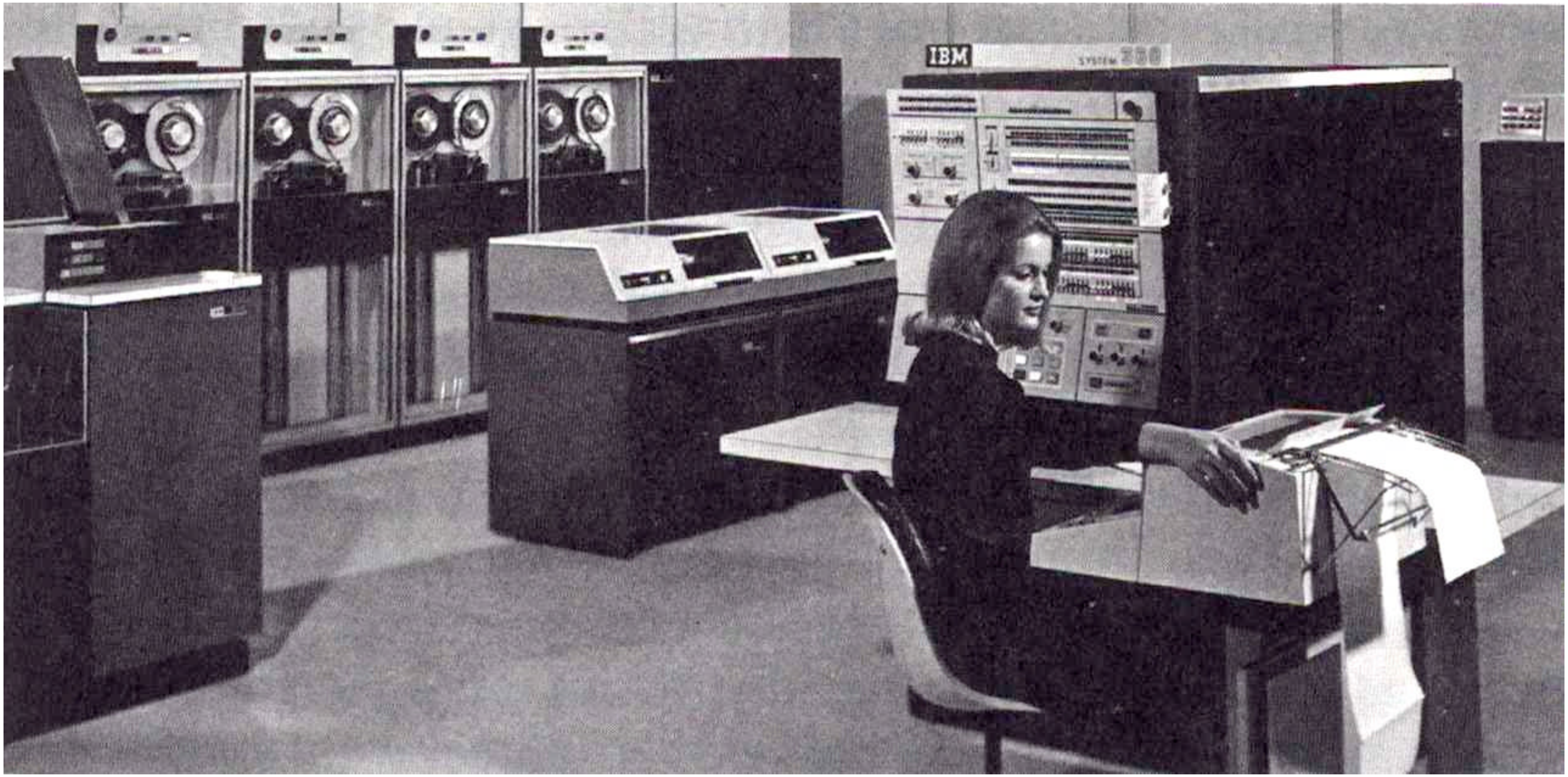
Punch Card



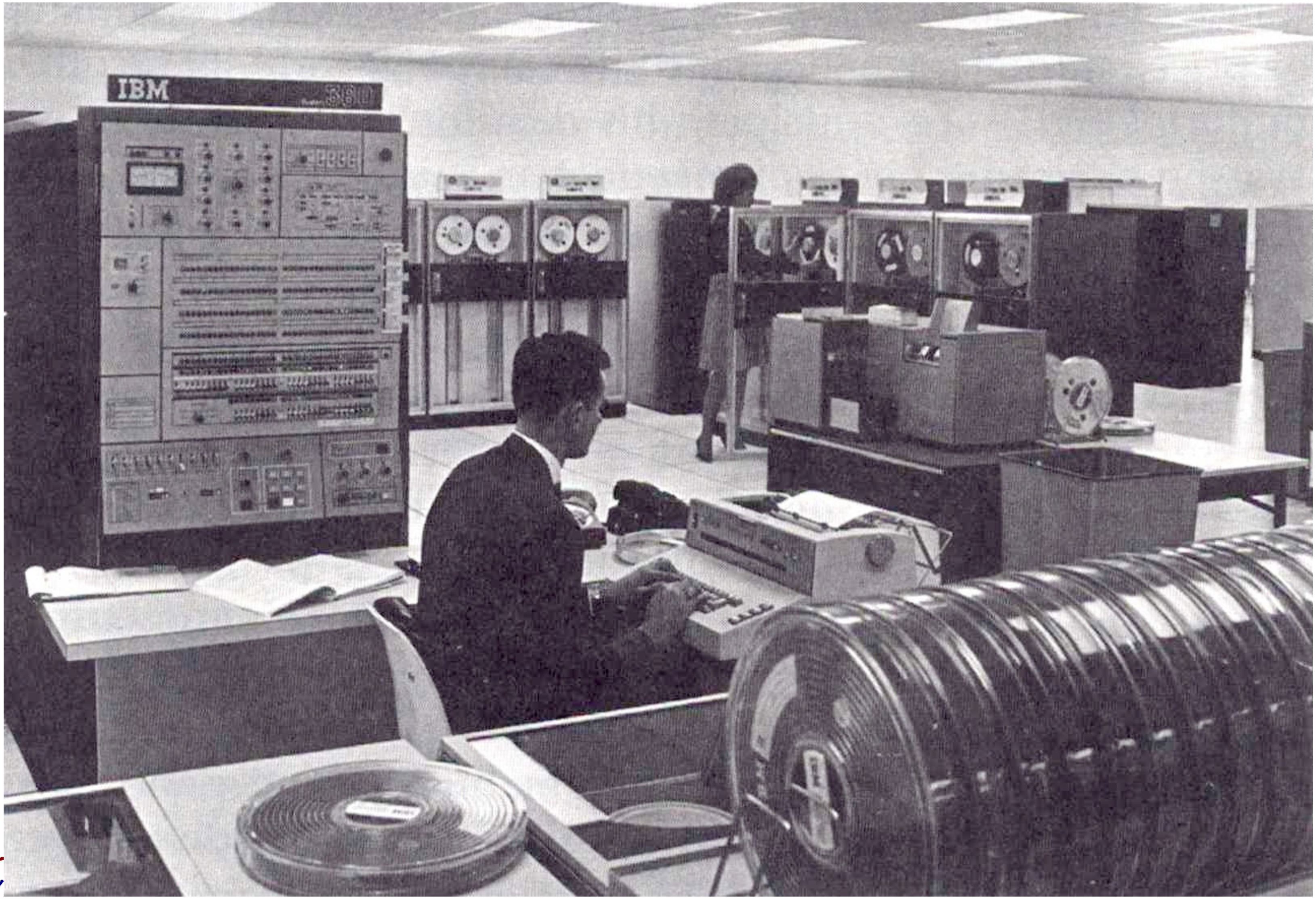
Model 029 Keypunch

IBM 7090

System/360 Model 40



System/360 Model 50



LAMPS Mark I

LAMPS Radar Test

LAMPS Mark III

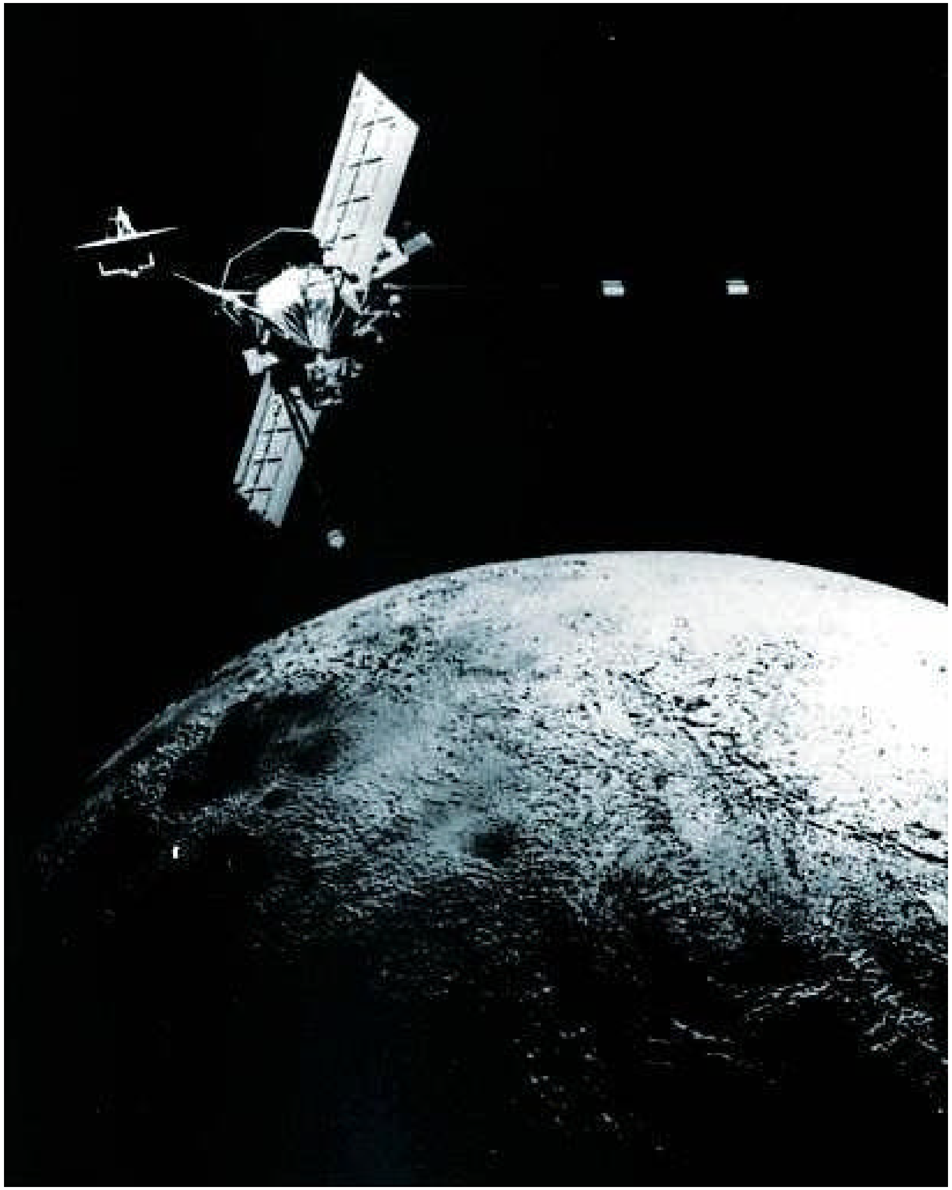

NAVY
P-3C
NAVY

Ticonderoga Class (USS San Jacinto, CG-56)

AWACS

NASA Space Shuttle

NASA Mariner 10



Embedded Computer Capacities

Memory Size:		CPU Speed:
– 1970	8-32KB	128 KIPS
– 1975	16-64KB	1.2 MIPS
– 1980	64-128KB	5 MIPS
– 1985	128-1MB	20 MIPS
– 1990	1-4MB	50 MIPS
– 1995	2-32MB	150 MIPS
– 2000	4-128MB	800 MIPS

- Increasing variability throughout this time

Size of Large Embedded Software

- How large is “large”:
  - 1970 10K SLOC
  - 1975 150K SLOC
  - 1980 1M SLOC
  - 1985 2M SLOC
  - 1990 4M SLOC
  - 1995 4M SLOC (increasing component use)
  - 2000 4M SLOC (increasing component use)
- Increasing variability throughout this time

Time Constraints

- Shortest Time Constraints Reliably Achievable:
  - 1970 50 milliseconds
  - 1975 1 millisecond
  - 1980 500 microseconds
  - 1985 100 microseconds
  - 1990 50 microseconds
  - 1995 10 microseconds
  - 2000 5 microsecond

Embedded Systems Proliferation

- Applications:
  - 1970 Military / Aerospace
  - 1975 Factory Automation / Telecom
  - 1980 Consumer Electronics
  - 1985 Wireless Telecom / Automotive
  - 1990 Games / Toys / Entertainment / Internet
  - 1995 Appliances
  - 2000 RFID

Outline

- What is a real-time System?
- What is an embedded System?
- Where have we come from?

ü
- What have we achieved?
- Where are we going?

What Was (Is Still) the Biggest Challenge?

- Exponentially increasing capacity
- Exponentially increasing software size and complexity
- Linearly increasing pool of developers
- Fixed or decreasing budgets
- The big problem – how to build exponentially more systems, and exponentially more complex systems with linearly increasing labor.

A Major Problem – But Not New

Strange game – the only way to win is not to play! - Joshua in the movie *Wargames*

Only way to produce complex software:

- Avoid writing, testing, documenting code
  - Use Commercial Off-The Shelf (COTS)
  - E.g., RTOS, CORBA, Database, Web-based, Automated tools, reuse existing code

- Unintended consequence
  - Performance problems

Present RT/Embedded Challenges

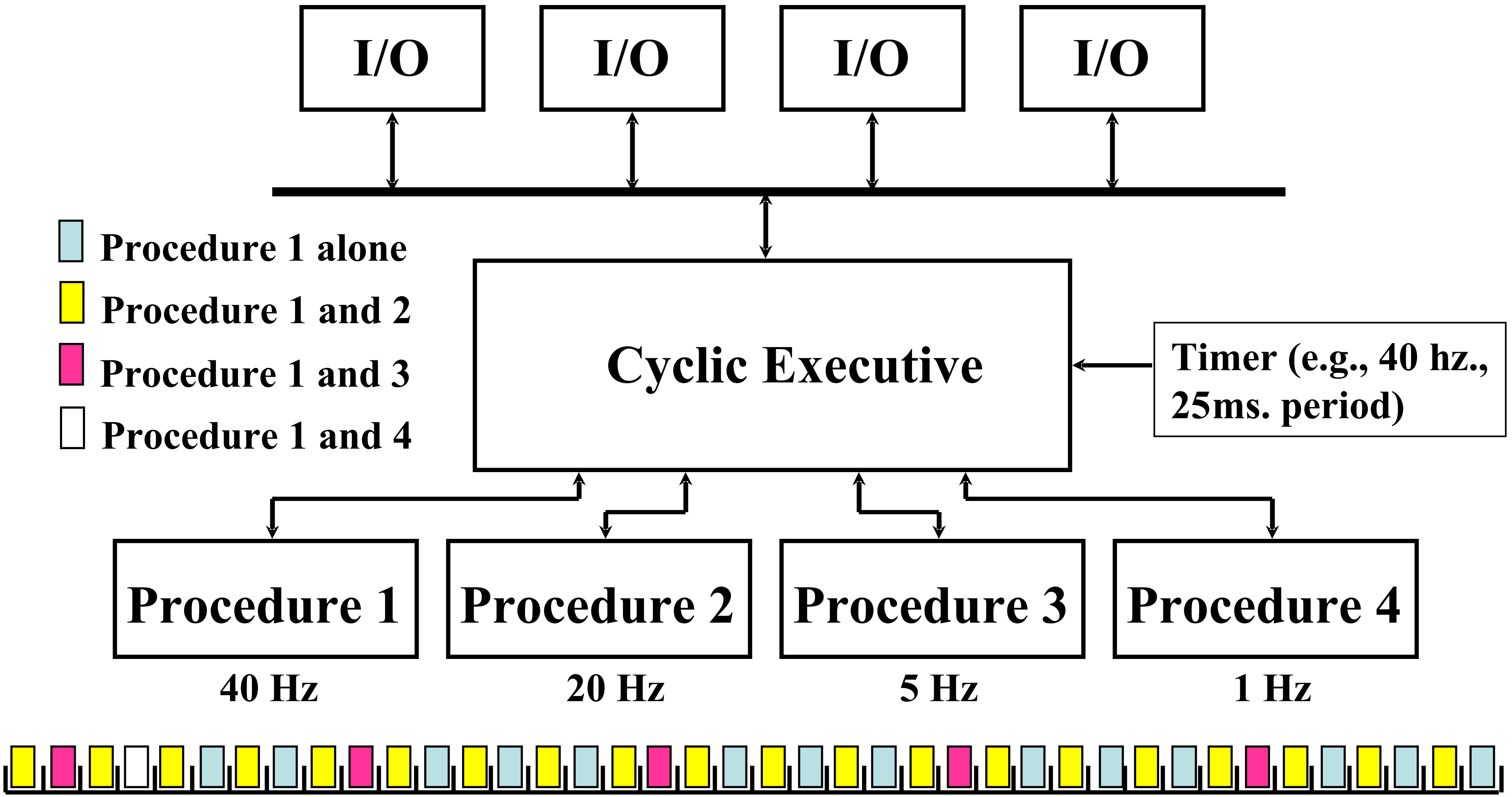
Top Three Problems:

- Managing software engineering organizations
- ü Ensuring development of a performance-relevant architecture
- Finding suitable tools (language, COTS, analysis, simulation)

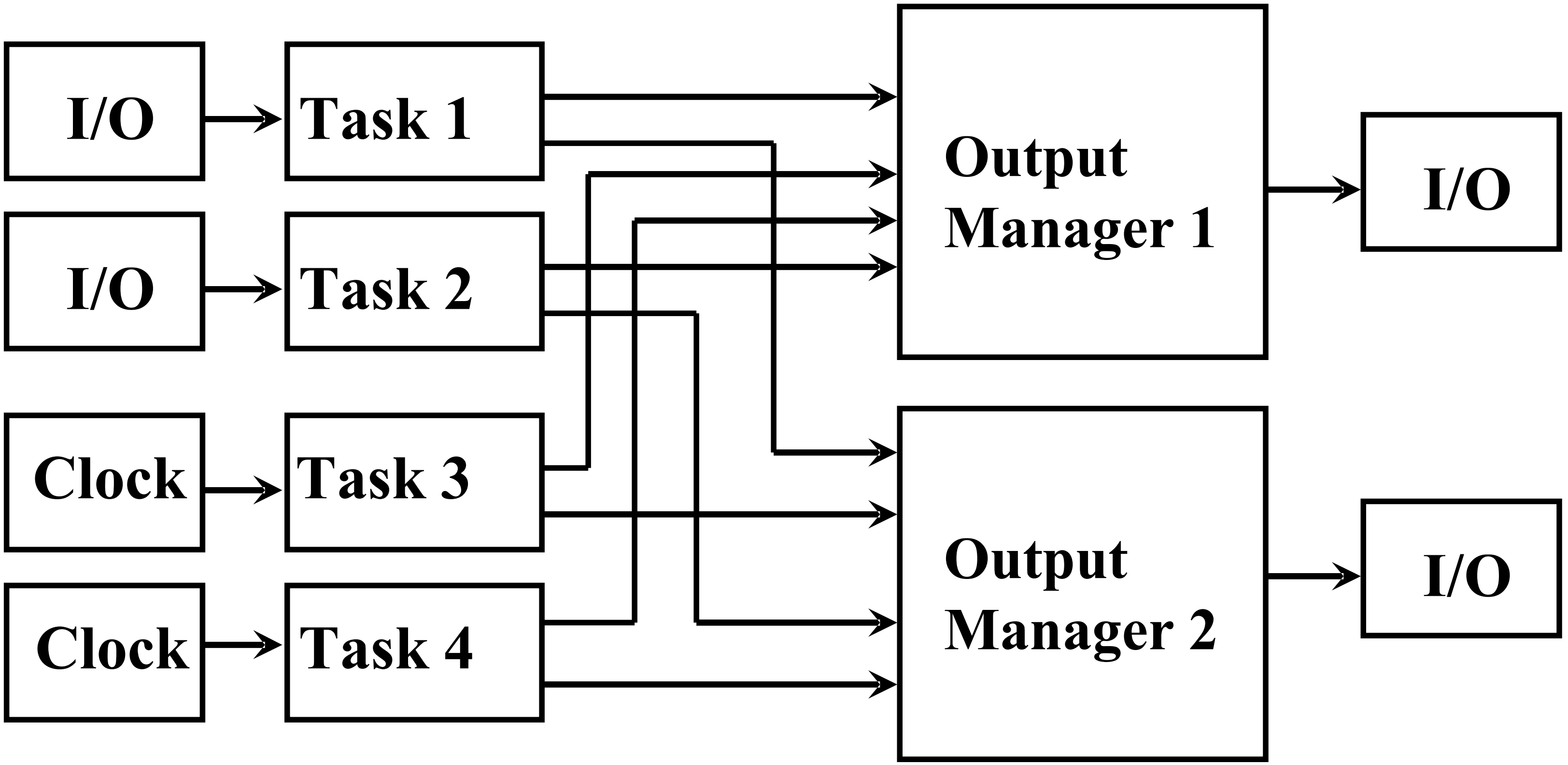
A Taxonomy of Real-Time Architectures

- The vast majority of existing real-time applications use one of four (overlapping) architectural types:
    1. Timeline (a.k.a. cyclic executive or frame-based)
    2. Event-driven (with both periodic and aperiodic activities)
    3. Pipeline
    4. Client-Server

Timeline or Cyclic Executive

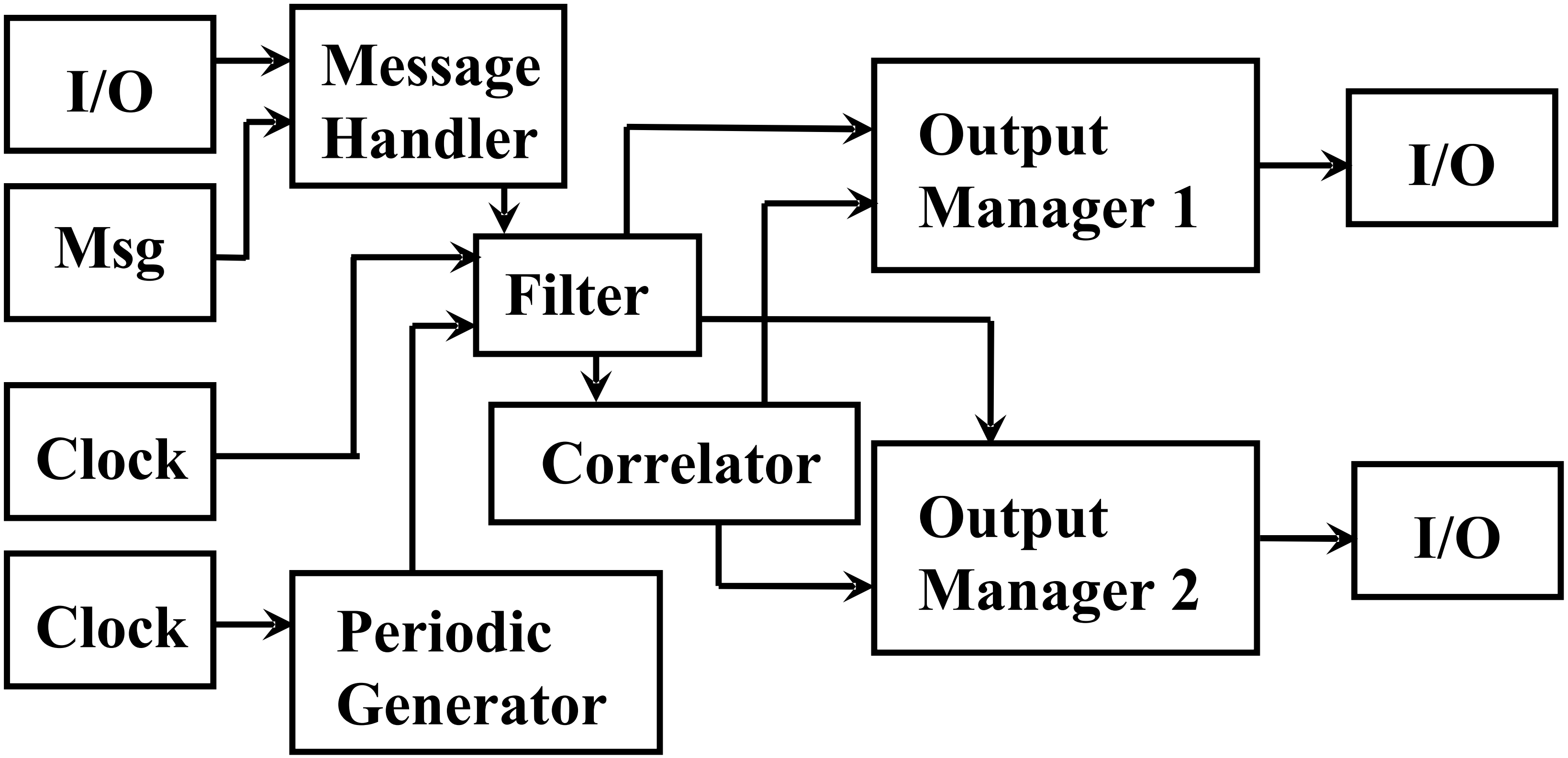


Event-Driven



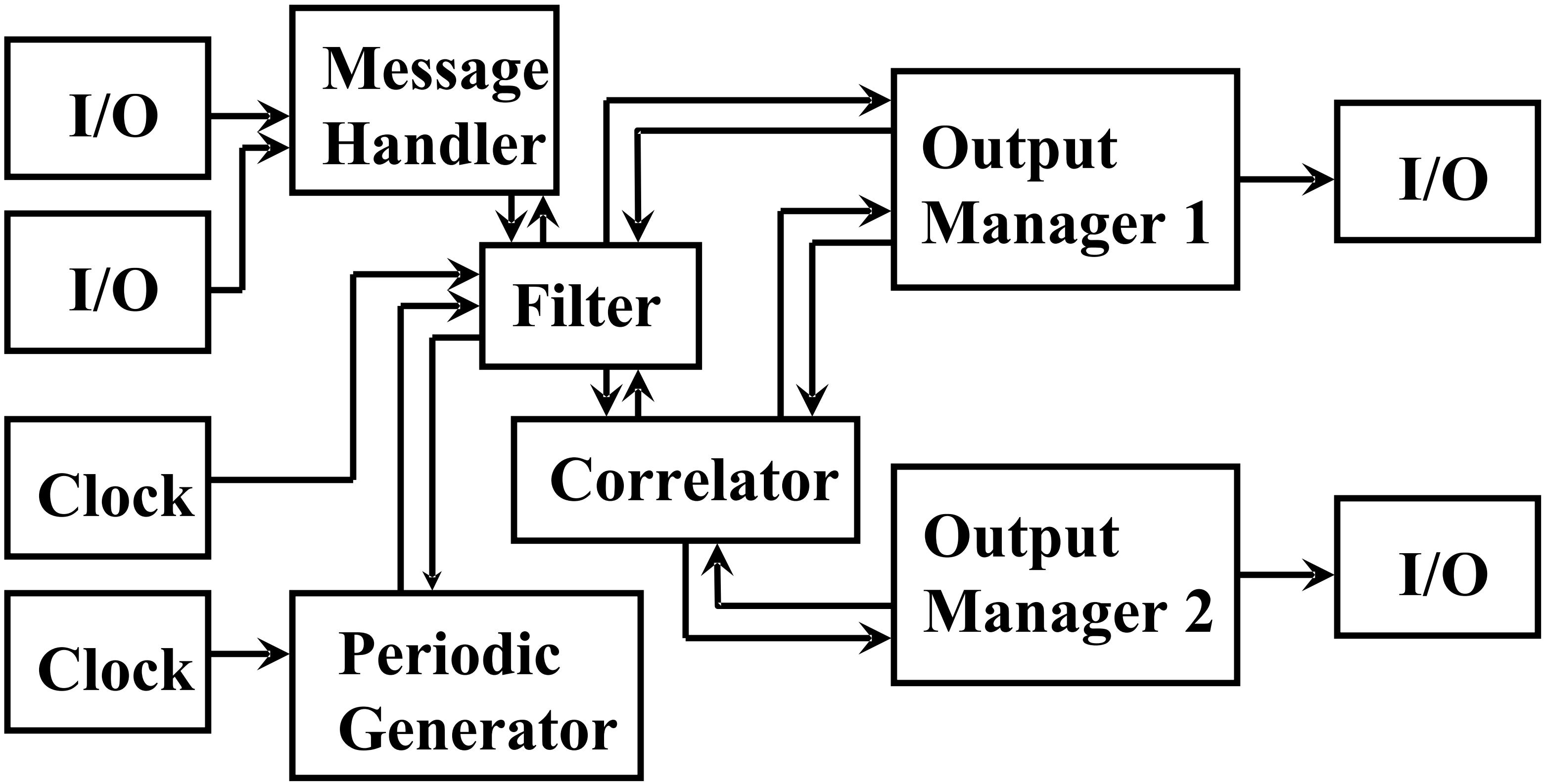
Tasks generally priority scheduled

Pipeline



Tasks usually ad-hoc scheduled

Client-Server



Tasks usually ad-hoc scheduled

Architecture Summary

- None of the architectures described are free of problems
  - Timeline is extremely expensive to integrate and maintain
  - Event-driven model is predictable for relatively static designs
  - Pipelines commonly result in non-preemptive delays (i.e., priority or policy inversion), few tools for predictable response
  - Client Server infrastructures perform similarly to pipelines except concurrency can be much more limited.

- The Bottom Line: Architecture decisions have a major effect on
  - Performance
  - Safety
  - Fault Tolerance
  - Life Cycle Cost

IEEE Computer Society TC-RT

- What has our community produced?
- Quite a lot - a few examples:
  - Rate (Deadline) Monotonic Scheduling
  - Utility (or Value) Function Scheduling
  - Many other scheduling paradigms (e.g., EDF)
  - Imprecise Computations
  - Fault Tolerant Computing (e.g., Simplex)
  - Real-Time Databases
- We have had considerable influence
  - POSIX
  - Real-Time CORBA
  - Real-Time Linux
  - Ada 95, Real-Time Java
- But much of our contribution isn't widely known

Outline

- What is a real-time System?
- What is an embedded System?
- Where have we come from?
- What have we achieved?
- ü Where are we going?

Where Are We Going?

- Resources are still limited
  - Therefore they will still need careful management
  - Scheduling still matters
- "Non-functional" requirements are now the primary focus of most designs
  - Real-time response
  - Fault tolerance
  - Availability
  - Quality of Service
  - Power Management
  - Security
  - Cost (people cost + resource cost)
- This is where we continue to make a difference

Doug Locke

Locke Consulting, LLC

www.doug-locke.com