NC STATE UNIVERSITY

ECE 460/560
Embedded Systems Architectures: Introduction (v2)

A.G. Dean
agdean@ncsu.edu
https://sites.google.com/ncsu.edu/agdean/teaching

8/19/2025

NC STATE UNIVERSITY

Embedded Systems High-Level View (1)

- Embedded Computer Systems frequently target **control applications**
  - Get input (read signal, detect event), Compute new output value, Update output
  - Microcontroller = Microprocessor + memory + hardware peripherals to support control
- Embedded Systems have **processes**, different implementation options
  - **Software** can do almost anything (eventually). Timing is slow, very sloppy.
  - **Hardware** is very fast and energy-efficient, uses dedicated circuits. Stable timing. Limited functionality available.
- Typically have **multiple concurrent processes** due to application requirements
- These processes often have **diverse I/O operations**
  - Digital signals, analog signals (must be converted to digital)
  - Bursts of events (e.g. PWM, serialized data, etc.),
  - Sample input periodically vs. receive event notification,
  - Range of I/O operation frequencies

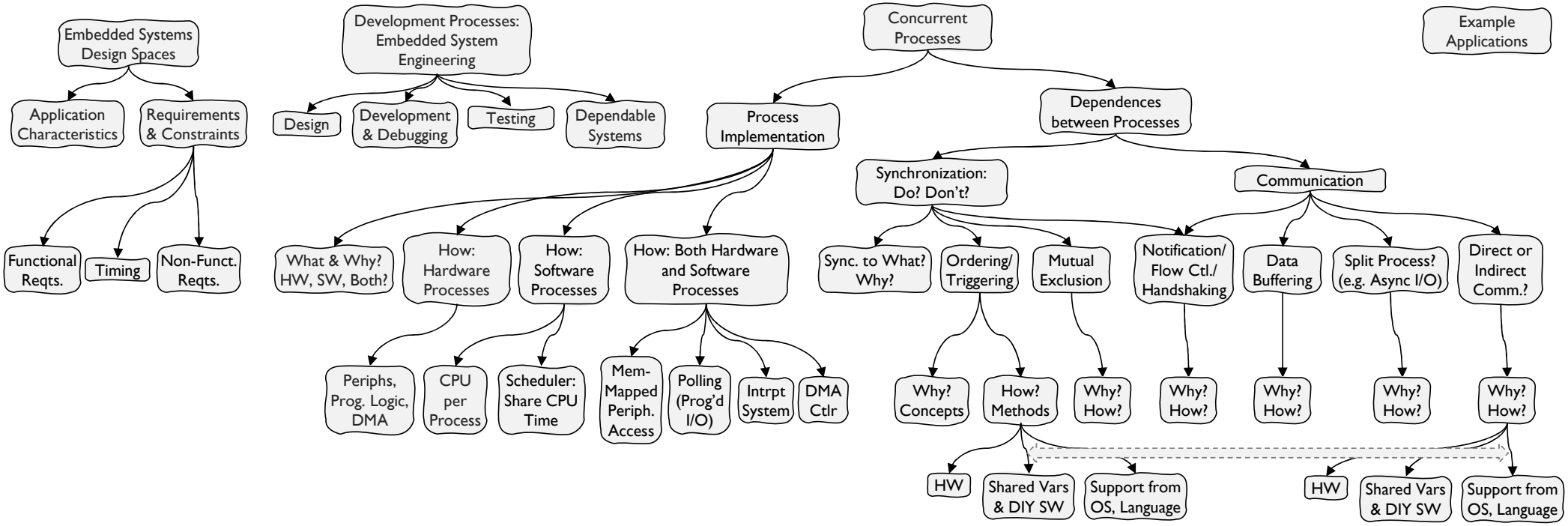
NC STATE UNIVERSITY

Embedded Systems High-Level View (2)

- The **I/O** for a process often has **challenging timing requirements**
  - **Periodic** events**,** events **synched to other/previous events** on this/other signals
- **Decouple the I/O** from compute software (bad timing characteristics) by splitting it into two or more processes to make **input or output** operations **asynchronous** to the **compute** operations**.**
  - We may move some processing to **hardware peripheral circuits**.
- These processes need to **synchronize** and **communicate** (data buffering).
- We use **interrupts**, **HW peripherals** and **DMA** to make a **low-cost** and **feasible** solution with a low-frequency CPU.

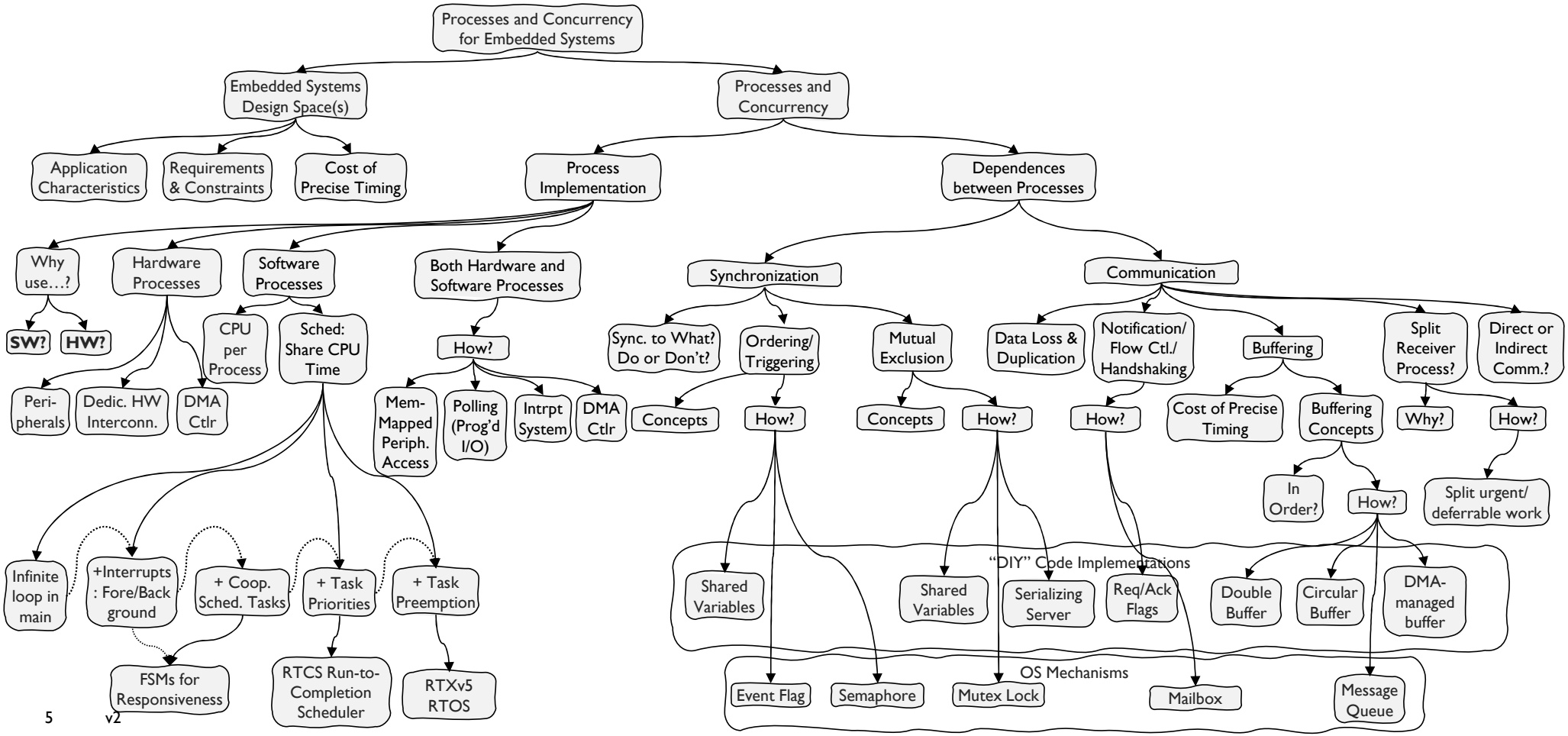
NC STATE UNIVERSITY

High-Level Topic Map

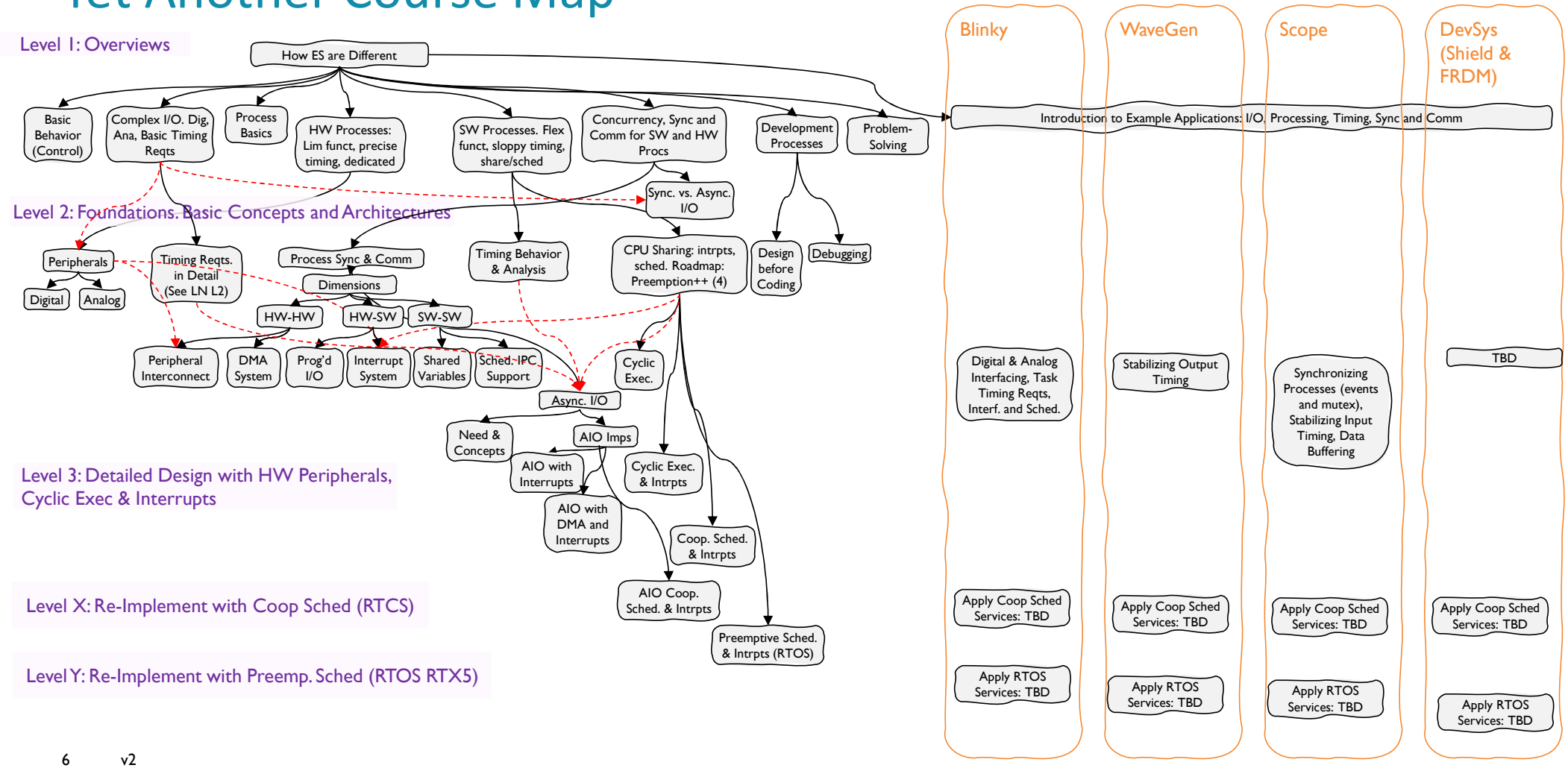


NC STATE UNIVERSITY

Extending the Topic Map



Yet Another Course Map



NC STATE UNIVERSITY

Apply to Examples

Concepts and Methods											Problem-Solving					Examples								
Application Requirements		Development Processes	Process Implementation		Process Scheduling		Process Synchronization and Communication				Correct Functionality	Timing Stability	Responsiveness	Compute Efficiency	Throughput	Blinky Lights	Waveform Generator	Oscilloscope	FRDM		Shield			
Inputs, Outputs, Functionality	Timing, other Non-Functional		HW	SW	HW	SW	HW	SW	SW->HW	HW->SW									Serial Comm.	I²C Comm.	LCD Controller	Touchscreen	SMPS Controller	µSD via SPI

NC STATE UNIVERSITY

Many Interconnected Methods

Concepts and Methods										Problem-Solving				
Application Requirements		Development Processes	Process Implementation		Process Scheduling		Process Synchronization and Communication			Correct Functionality	Timing Stability	Responsiveness	Compute Efficiency	Throughput
Inputs, Outputs, Functionality	Timing, other Non-Functional		HW	SW	HW	SW	HW	SW	SW->HW	HW->SW					
User interface, Control Systems, Media DSP, Data logging, Sensor data processing & fusion, etc. …	I/O event timing, internal timing, power and energy consumption, code size	Defining requirements, Design before coding, Estimation, Design for X, Testing, Dev. Processes for dependable and safety-critical systems	Peripherals, DMA controller Prog. logic,	Source code, build toolchain, object code	Peripheral interconn., DMA	Events vs. polling, While 1 loop, Interrupt system, Cooperative tasks, Preemptive Tasks. Priorities, preemption	Peripheral interconn., DMA	Shared variables with algorithms, OS/Language support	Sync. Output, Async. Output, Data buffering	Sync Input, Interrupts, Async Input, Data buffering	Concurrency bugs, Testing, Debugging, Dependable system architecture	Timing analysis, Time synchronization, Timer peripheral, sched/OS timer, preemption & blocking	SW process Timing Analysis , System Response time analysis, Prioritization, blocking, preemption, Real-Time	Overhead, batch processing, SW->HW	Overhead, batch processing, SW ->HW