Real-Time Systems

Lecture Topic - Accounting for Execution Efficiency

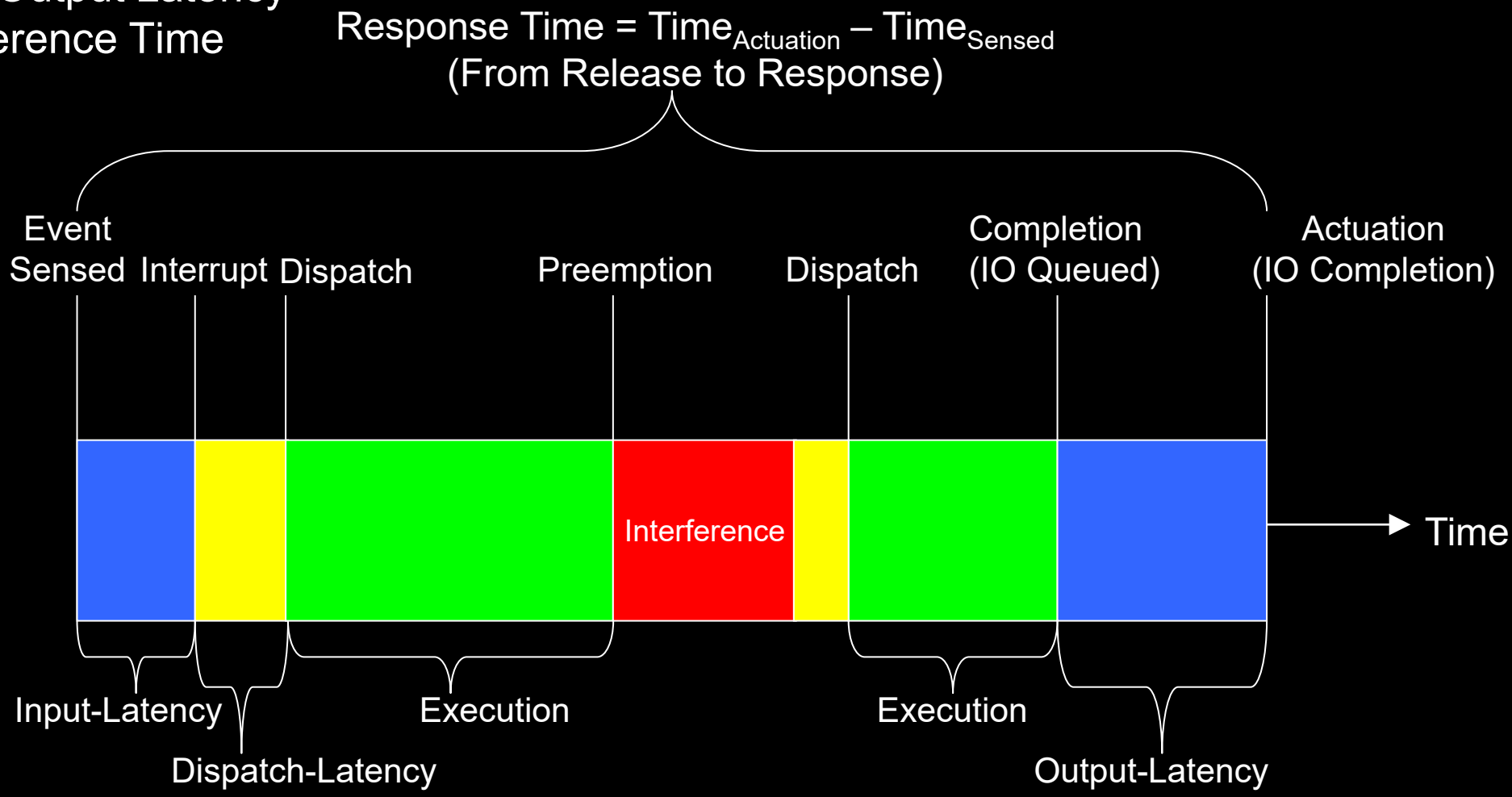
Dr. Sam Siewert

Electrical, Computer and Energy Engineering

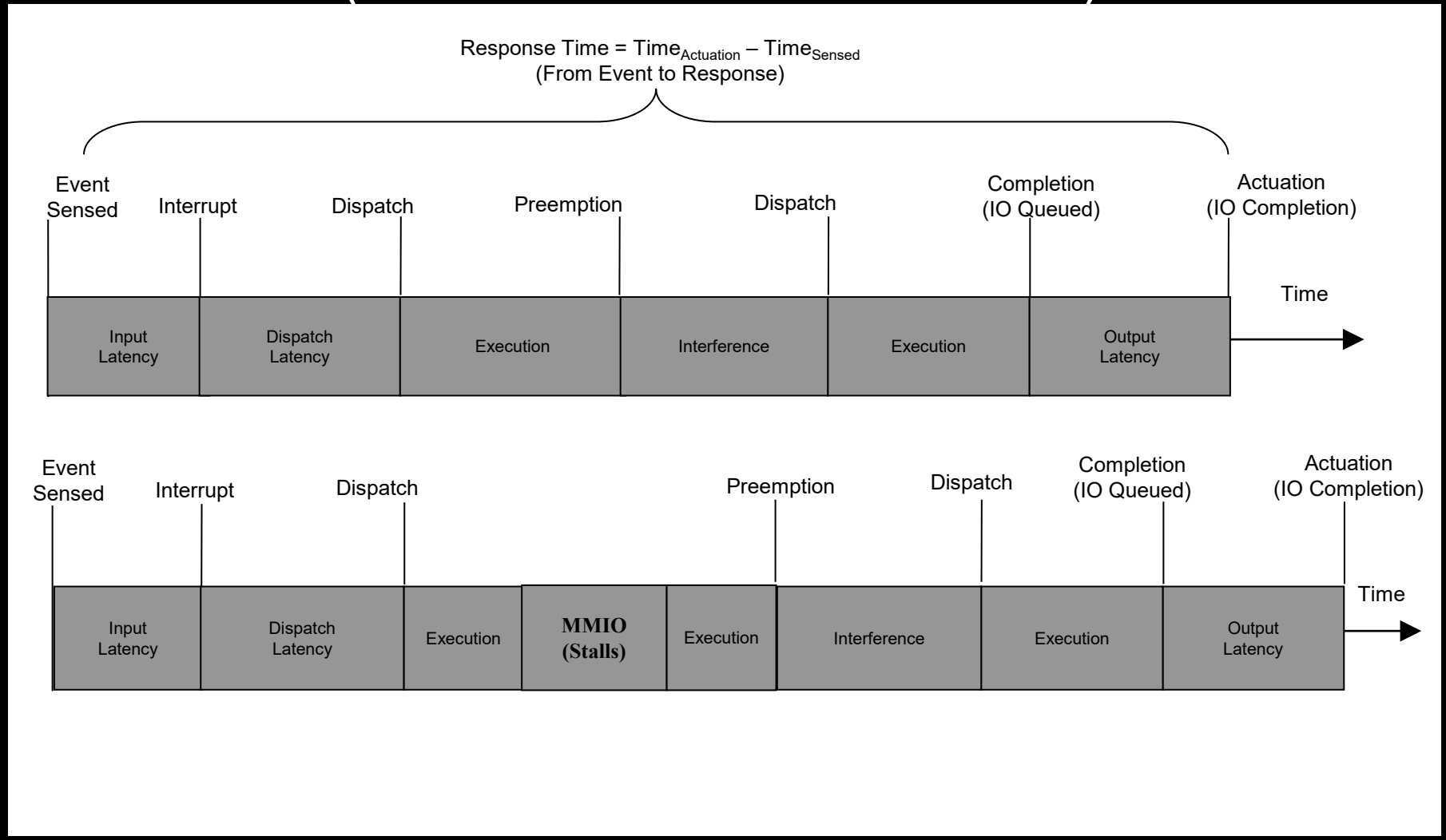
Embedded Systems Engineering Program

A Service Release and Response

- C_i WCET
- Input/Output Latency
- Interference Time



Service Response Timeline - WCET
(With Intermediate I/O – WCET)



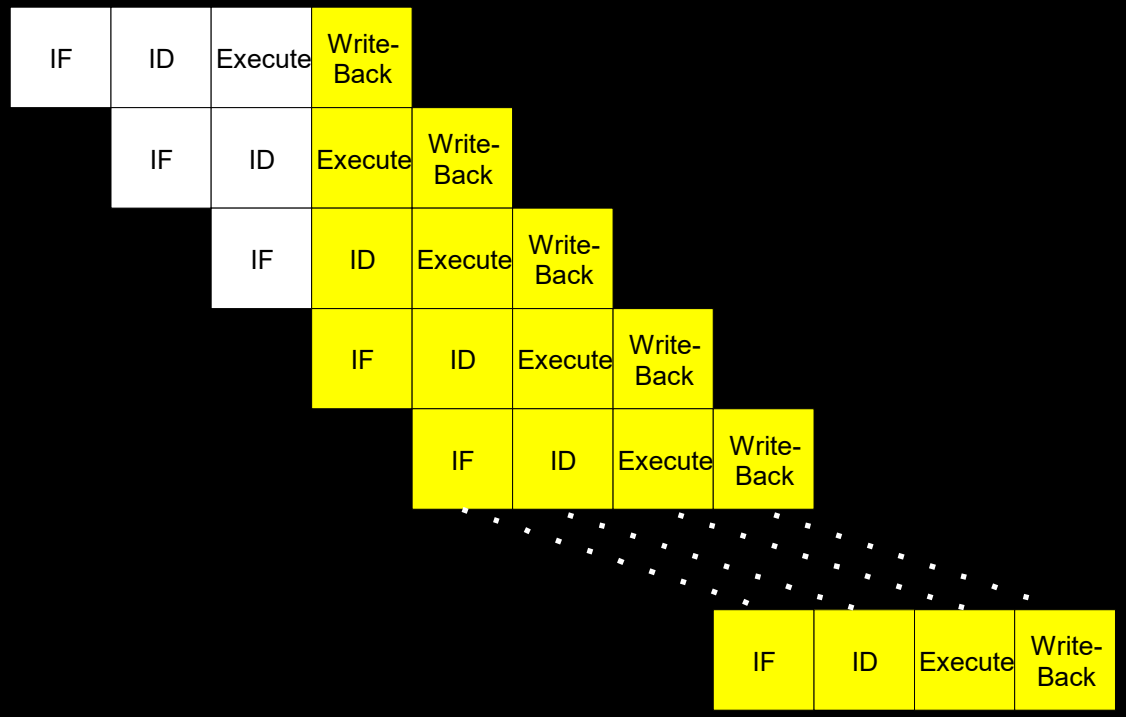
Services and Execution Efficiency

- CPU Pipeline Stalls (Hazards)
  - Cache Miss/Hit
  - Off-Chip MMIO
  - Branch Misprediction
  - Data Dependencies
  - Register Pressure, MMR (Memory Mapped Registers)

- During Service Execution, Memory Mapped Register I/O
- External Memory I/O
- Cache Loads and Write-Backs
- In Memory Input/Output from Service to Service

Pipelined Architecture Review

- Recall that Pipeline Yields CPI of 1 or Less
- Instruction Completed Each CPU Clock
- Unless Pipeline Stalls!



Service Execution

$$WCET = \left[(CPI_{best-case} \times Longest\_Path\_Inst\_Count) + Stall\_Cycles \right] \times Clk\_Period$$

- Efficiency of Execution
- Memory Access for Inter-Service Communication
- Bounded Intermediate I/O
- Ideally Lock Data and Code into Cache or Use Tightly Coupled Memory [Scratch Pad, Zero Wait State]

Service Efficiency

- Path Length for a Release
  - Instruction Count
  - Longest Path Given Algorithm
    - Branches
    - Loop Iterations
    - Data Driven?
- Path Execution
  - Number of Cycles to Complete Path
  - Clocks Per Instruction
  - Number of Stall Cycles for Pipeline
    - Data Dependencies (Intermediate IO)
    - Cache Misses (Intermediate Memory Access)
    - Branch Mis-predictions (Small Penalty)

Hiding Intermediate IO Latency
(Overlapping CPU and Bus I/O)

- ICT = Instruction Count Time
  - (CPI=1) x Inst_Cnt x CPU_Clk_Period
  - Time to execute a block of instructions with no stalls, hence CPI=1
  - CPU Clk Cycles x CPU Clock Period
- IOT = Bus Interface IO Time
  - Bus IO Cycles x Bus Clock Period
  - Bus Clock Period / CPU_Clk_Period = Wait Clocks
- OR = Overlap Required
  - Percentage of CPU cycles that must be concurrent with I/O cycles
- NOA = Non-Overlap Allowable for Si to meet Di
  - Percentage of CPU cycles that can be in addition to IO cycle time without missing service deadline
- Di = Deadline for Service S_i relative to release
  - interrupt or system call initiates S_i request and S_i execution
- CPI = Clocks Per Instruction for a block of instructions
  - IPC is Instructions Per Clock, Also Used, Just the Inverse (Superscalar, Pipelined)

Processing and I/O Time Overlap

- If Di < IOT, S_i is ***IO-Bound***

Hardware Re-design REQUIRED

- If Di < ICT, S_i is ***CPU-Bound***

- if Di < (IOT + ICT) where (Di < IOT AND Di < ICT), deadline **Di can't be met regardless of overlap, both IO-Bound and CPU-Bound**

- Di >= (IOT + ICT) **requires no overlap** of IOT with ICT

No code I/O optimization required

- if Di < (IOT + ICT) where (Di >= IOT and Di >= ICT), **overlap of IOT with ICT is required**

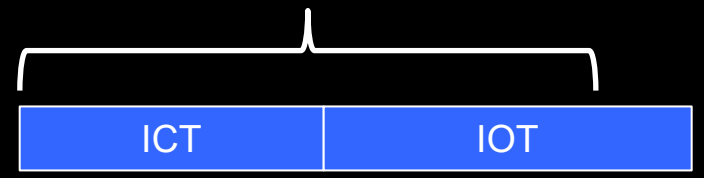
Code I/O optimization required

- **CPI_{ICT}=1 by definition for ICT alone (no stalls, best case efficiency)**

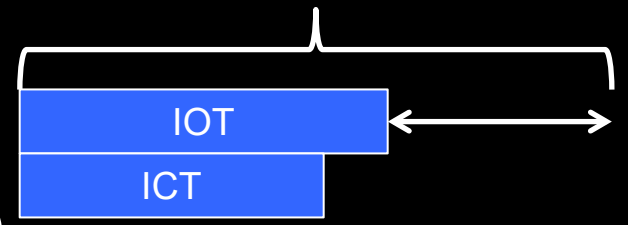
- **$CPI_{actual} = f_{efficiency} \times CPI_{ICT}$**

Service Execution Efficiency – Waiting on Bus and Memory I/O

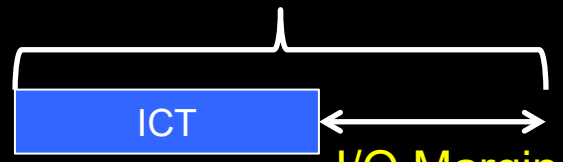
- $CPI_{worst\text{-}case} = (ICT + IOT) / ICT$
- $CPI_{best\text{-}case} = (\max(ICT, IOT)) / ICT$
- $CPI_{required} = D_i / ICT$
- $OR = 1 - [(D_i - IOT) / ICT]$
- $CPI_{required} = [ICT(1\text{-}OR) + IOT] / ICT$
- $NOA = (D_i - IOT) / ICT$; $OR + NOA = 1$ (by definition)



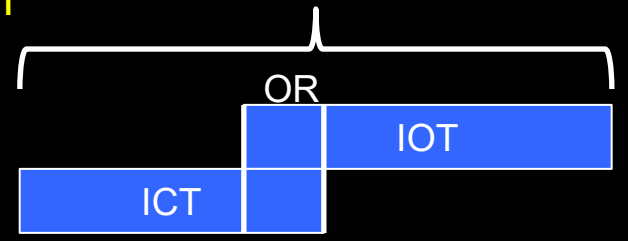
Missed Deadline



Maximum Margin



I/O Margin



Meets Deadline
(with I/O hiding)

CPI Units

- $CPI_{required} = D_i$ / ICT = Time/Time?
  - D_i = Clocks x Clock_T = Time [N x X nanoseconds]
  - ICT = CPI=1 x Num_instructions x Clock_T = Time [N x X nanoseconds]
- D_i / ICT = (Clocks x Clock_T) / (1 x Num_instructions x Clock_T)
  - Clocks / Num_instructions = $CPI_{required}$ over deadline D_i
- IPC = 1/CPI (IPC used for pipelined, superscalar)

Processing Latency Alone

- Write Code with Memory Resident Frames
  - Load Frames in Advance
  - Process In-Memory Frames Over and Over
  - Do No I/O During Processing
  - Provides Baseline Measurement of Processing Latency per Frame Alone
  - Provides Method of Optimizing Processing Without IO Latency