WCET Analysis

CPEN 400P

Arpan Gujarati
University of British Columbia

Who am I?

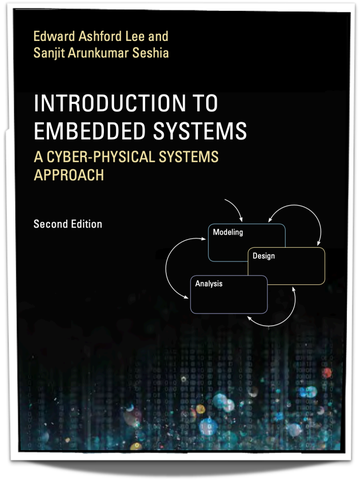
- Arpan Gujarati
  - Research Associate in CS
  - https://arpangujarati.github.io/

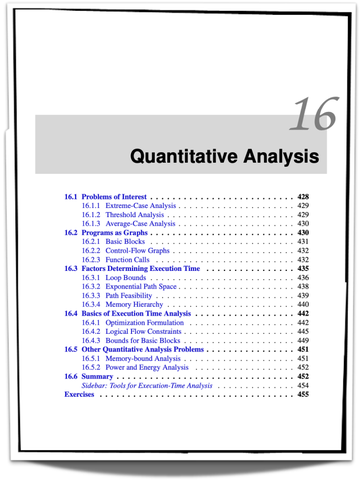
- Education
  - BE from BITS Pilani, India
  - PhD from MPI-SWS, Germany
    - Saarland University
    - TU Kaiserslautern

- Research Interests
  - Real-time systems, distributed systems, fault tolerance, reliability analysis, and scheduling
  - Domains: Cyber-physical systems, datacenter systems

In this lecture, you will learn …

- Introduction to **Cyber-Physical Systems (CPS)**
- **Real-time systems (RTS)** and the problem of **timing analysis**
- **Program analysis** for estimating the Worst-case Execution Time (WCET)
- Reference
  - Ch. 16, Introduction to Embedded Systems, 2nd Ed.
  - Available online at LeeSeshia.org
- Acknowledgements for slides
  - Max Planck Institute for Software Systems (© MPI-SWS)
  - Real-time System Design (© CPEN 432)




16
Quantitative Analysis

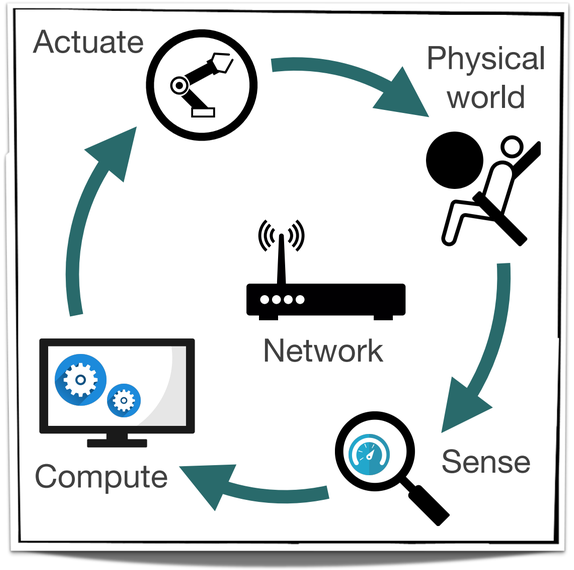
Cyber-Physical Systems

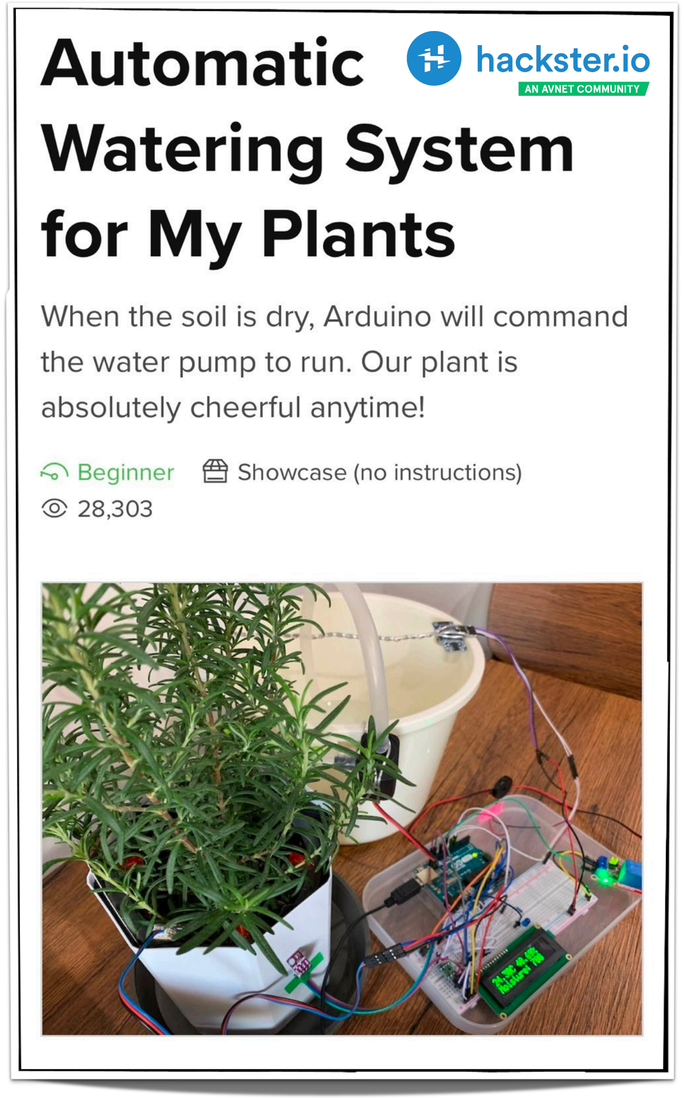
Cyber-Physical Systems

Tight and seamless integration

- Computation
- Networking
- Actuation and control
- Sensing of the physical world

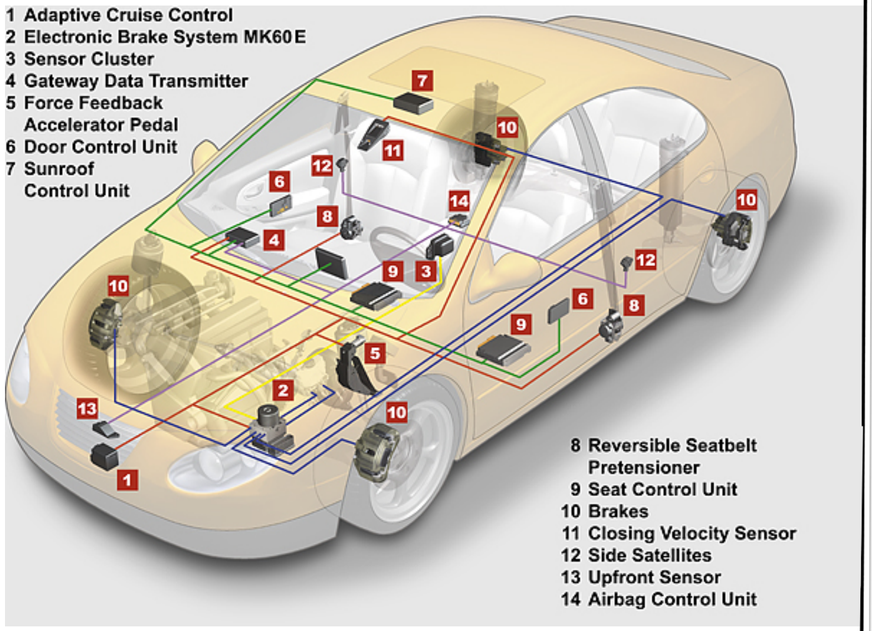
Feedback control loops





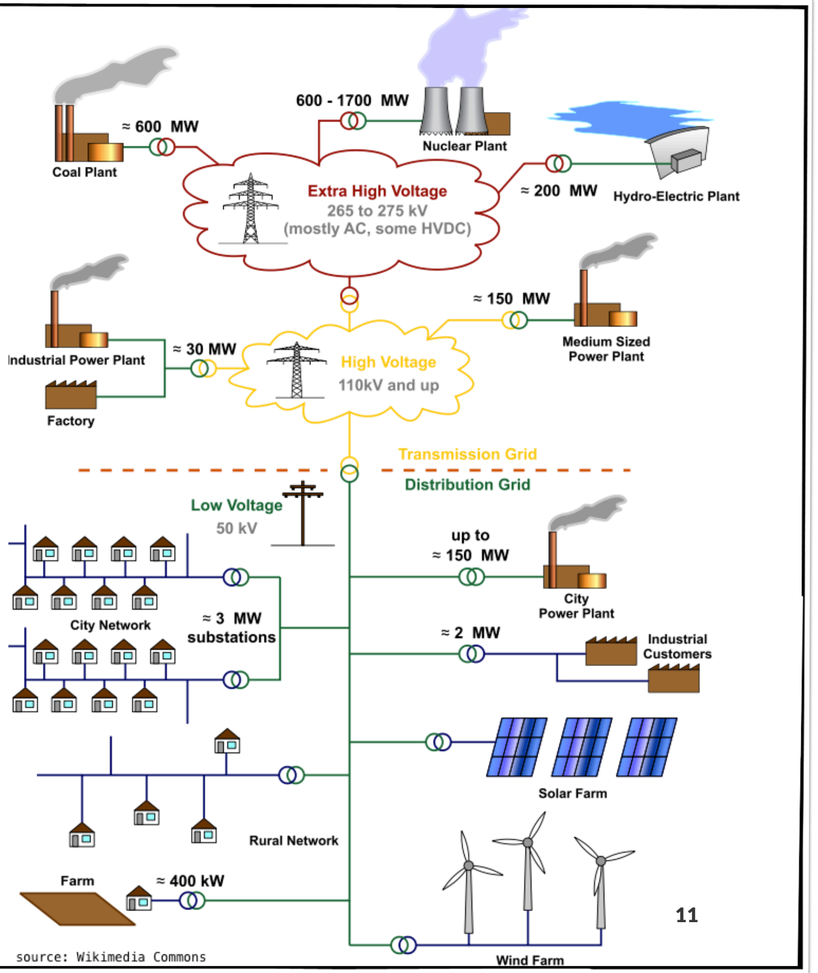
Example CPS (1/3): Automotive Systems

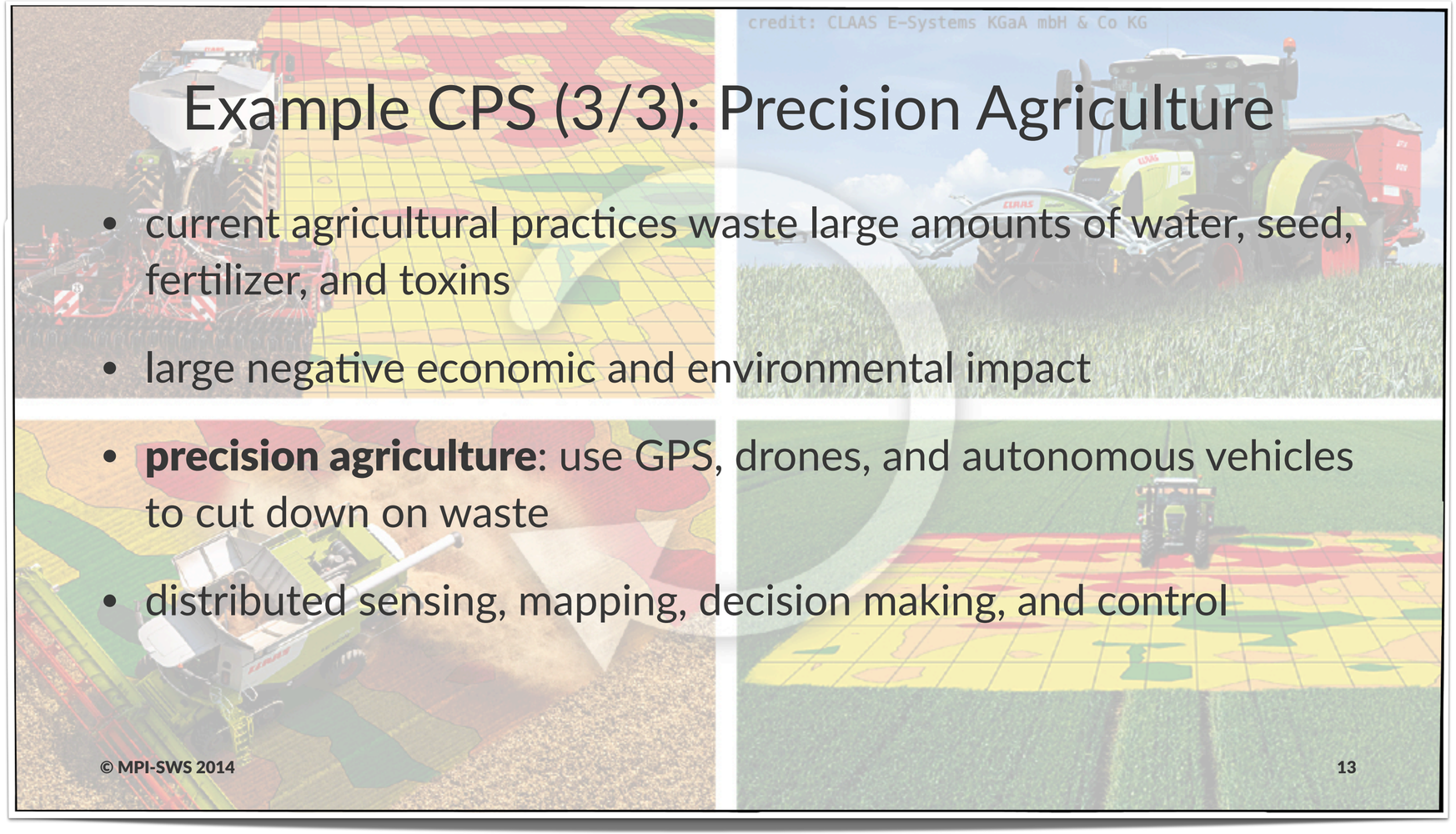
- distributed sensors and actuators
- many driver assistance systems: anti-lock breaking system (ABS), electronic stability control (ESC), adaptive cruise control, adaptive light control, lane departure warning, X-by-wire … → **complex interactions**
- Safety determined by physics!



Example CPS (2/3): Power Grid

- constantly changing demand and supply
- limited power line capacity
- supply must be controlled and adjusted
- inherently distributed and time-critical
- possibility of cascading failures
  (→ 2003 US & Canada power outage)
- The physics of power transmission determine and limit safe operation.





Example CPS (3/3): Precision Agriculture

- current agricultural practices waste large amounts of water, seed, fertilizer, and toxins
- large negative economic and environmental impact
- **precision agriculture**: use GPS, drones, and autonomous vehicles to cut down on waste
- distributed sensing, mapping, decision making, and control

Cyber-Physical Systems — Challenges (1/3)

- physical processes are **continuous** (typically modeled as differential equations), whereas software and computers are **discrete** (typically modeled as automata)
- nature is concurrent — **concurrency** is inherent in CPSs
- nature doesn't stop or slow down — **timeliness** is critical
- communication and computation takes time — **implementation performance** matters and cannot be "abstracted away"

Cyber-Physical Systems — Challenges (2/3)

- stuff happens — must deal with noise, delays, breakdowns, …
- the digital components are often **distributed** — partial failures
- limited energy, size, weight, cost budgets — **resource constraints**
- CPS are often part of a larger system — **open systems** interfacing with other, independent systems
- attackers might target critical infrastructure — **security** concerns

Cyber-Physical Systems — Challenges (3/3)

- often too large and **too important for trial and error** — cannot build a copy of the power grid, cannot afford autonomous vehicle failures
- by definition CPSs, interact with and control the real world — many CPSs are thus **safety-critical**
- **certification** requirements — must meet stringent correctness and documentation criteria

Real-Time Systems and Timing Analysis

CPS vs. General-Purpose Computing

"When studying CPS, certain key problems emerge that are rare in so-called general-purpose computing. For example, in general-purpose software, **the time it takes to perform a task** is an issue of **performance**, not correctness. It is not incorrect to take longer to perform a task. It is merely less convenient and therefore less valuable. In CPS, the time it takes to perform a task may be **critical** to [the] correct functioning of the system."[Der11]

— Patrica Derler et al., 2011 (emphasis added)

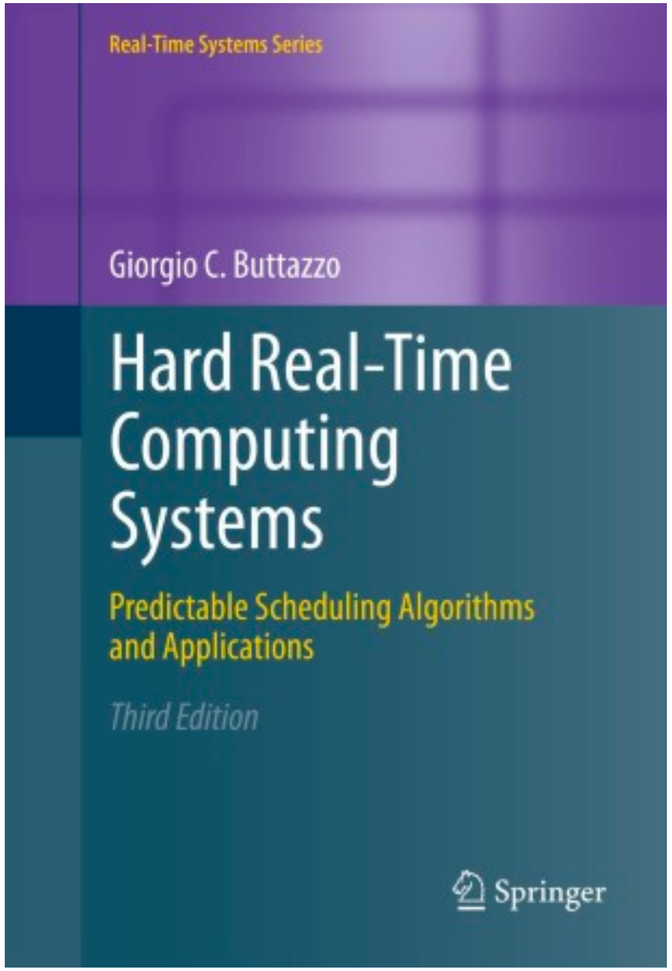
[Der11] P. Derler et al. (2011). Modeling Cyber–Physical Systems.

What are Real-Time Systems? [1/3]

1.1 INTRODUCTION

Real-time systems are computing systems that must react within precise time constraints to events in the environment. As a consequence, the correct behavior of these systems depends not only on the value of the computation but also on the time at which the results are produced [SR88]. A reaction that occurs too late could be useless or even dangerous. Today, real-time computing plays a crucial role in our society, since an increasing number of complex systems rely, in part or completely, on computer control. Examples of applications that require real-time computing include the following:

- Chemical and nuclear plant control,
- control of complex production processes,
- railway switching systems,
- automotive applications,



What are Real-Time Systems? [2/3]

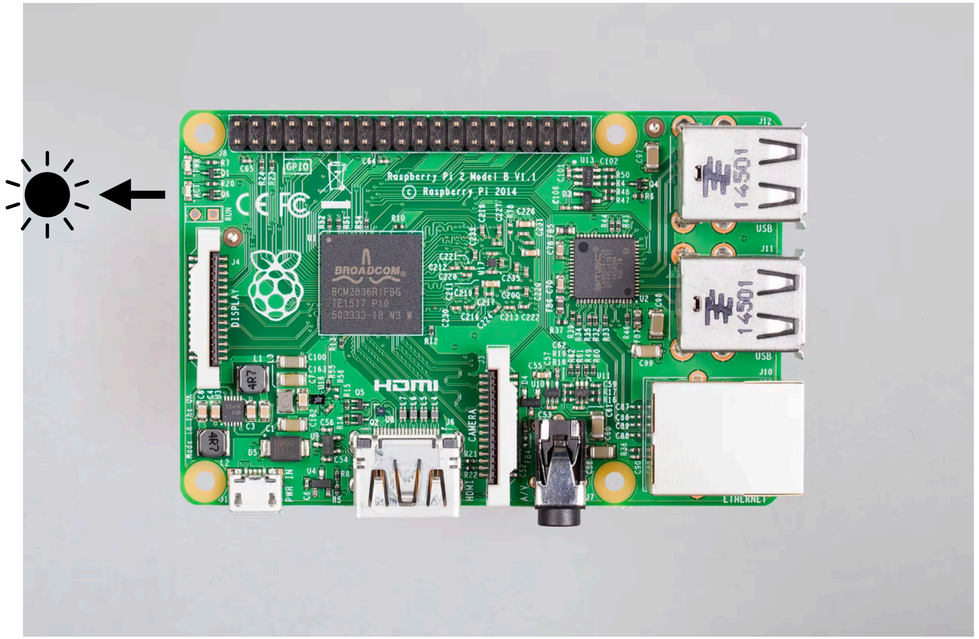
- The **time** it takes to perform a task is
  - not just an issue of performance
  - but **critical to the correct functioning** of the system
- Examples
  - Airbag deployment in cars, processing of sensor data in drones, etc.
- Challenges
  1. Can we **engineer** the system such that it always satisfies “timing constraints”?
  2. Can we **prove in advance** that the system **will always satisfy** “timing constraints”?

What are Real-Time Systems? [3/3]

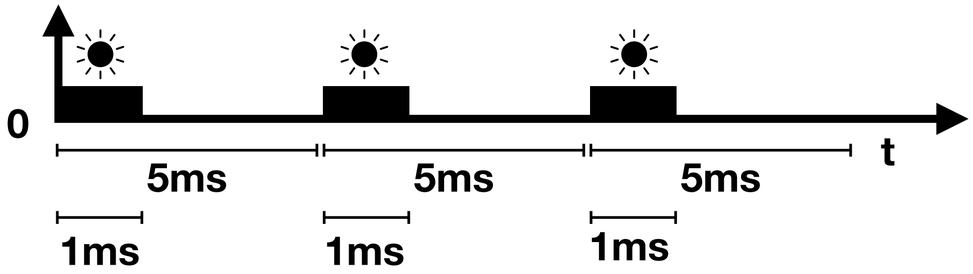
Model the system and the workload

- # instructions in the blinking code?
- Is there an OS? # instructions between calls to the blinking code?
- Processor speed? Time to execute a single instruction? Caching effects?
- Ignore unnecessary details …
  - Can we ignore the GPU?
  - Disable interrupts and ignore?

≈



Timing constraint: The status LED blinks every 5ms, and continues blinking for precisely 1ms



For the given model

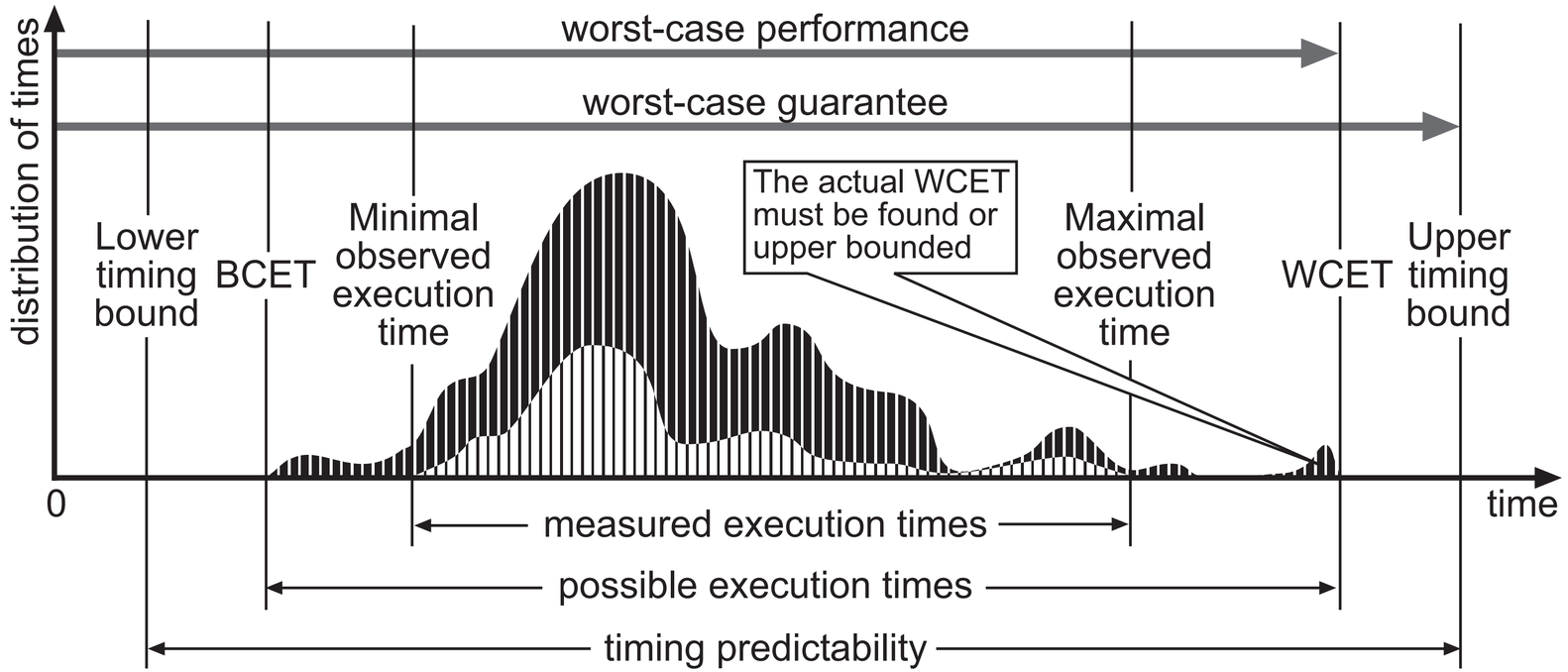
- **Prove** that the specified timing constraint is always satisfied

Timing Analysis

- **WCET analysis**
  - Given a hardware platform and the implementation of a task, for at most how long will a single job execute (in isolation)?
- **Schedulability analysis**
  - Given multiple tasks and the WCET for each task, is it possible to host them on the same hardware platform?
- Ambiguous terminology
  - “Timing Analysis” can refer to either or both types of analyses

WCET Analysis

Execution Time Histogram[Wil08]



[Wil08] R. Wilhelm et al. (2008). The Worst-Case Execution Time Problem — Overview of Methods and Survey of Tools.

Terminology

WCET = maximum ever observed (on target platform)

BCET = minimum ever observed

ACET = average, dependent on input, BCET ≤ ACET ≤ WCET

It's important to distinguish between *bounds* (or *estimates*) and the *actual* WCET/BCET.
→ **Safety**: bound ≥ actual.
→ **Tightness**: bound close to actual value.

WCET Analysis Challenges

Two issues must be considered:
→ **software behavior** (control flow)
→ **hardware timing** (basic block bounds).

Processor caches, **out-of-order** processor pipelines, **speculative execution**
→ move the ACET closer to the BCET (and may even reduce the BCET)
→ typically make the WCET worse
→ increase the span between ACET and WCET.

Caches and speculation make the precise timing more dependent on the execution history, which is difficult to predict precisely.

Typical Software Restrictions

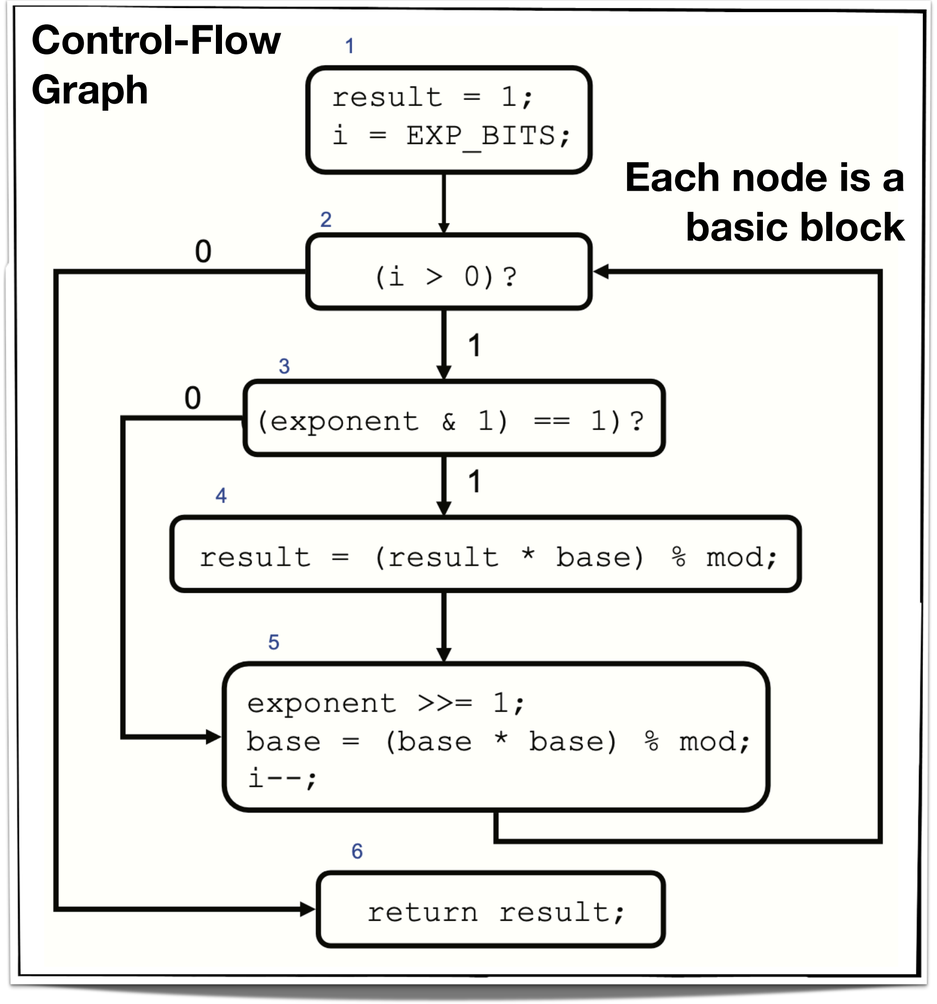
- no recursion
- no unbounded loops
- no function pointers / virtual method dispatch
- no/restricted pointer aliasing
- no dynamic linking
- no dynamic memory management

Programs as Graphs

```
1  #define EXP_BITS 32
2
3  typedef unsigned int UI;
4
5  UI modexp(UI base, UI exponent, UI mod) {
6    int i;
7    UI result = 1;
8
9    i = EXP_BITS;
10   while(i > 0) {
11     if ((exponent & 1) == 1) {
12       result = (result * base) % mod;
13     }
14     exponent >>= 1;
15     base = (base * base) % mod;
16     i--;
17   }
18   return result;
19 }
```

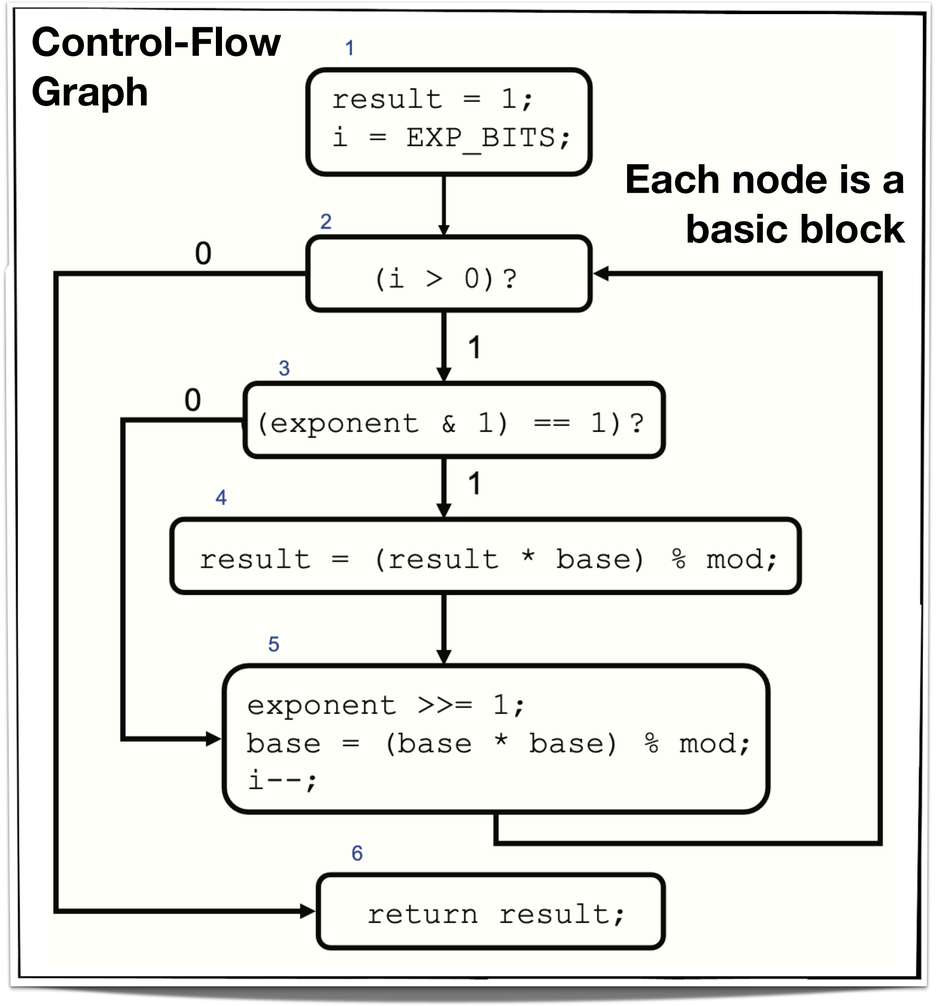
If e is even: $b^e = (b^2)^{\frac{e}{2}}$

If e is odd: $b^e = b \times (b^2)^{\frac{e-1}{2}}$



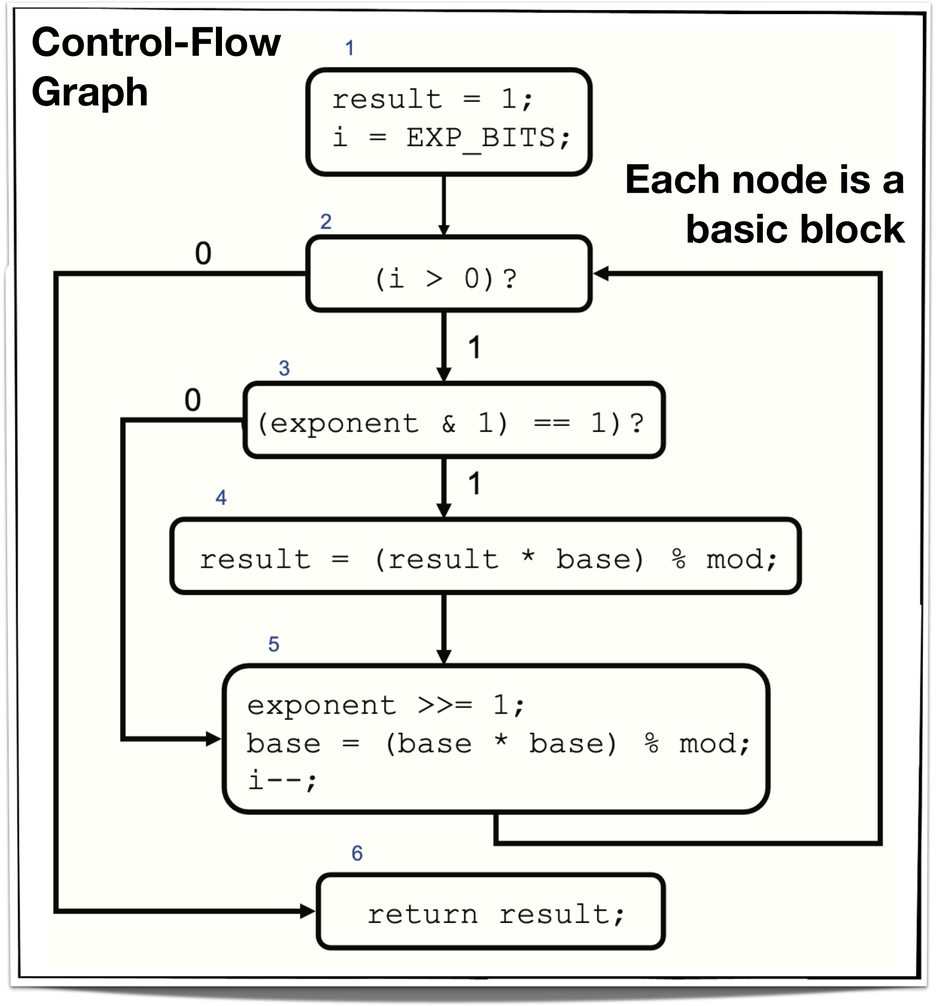
Optimization Formulation [1/3]

- Let $G = (V, E)$ denote the CFG
  - $n = |V|$ and $m = |E|$
- Let $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ be a vector of variables recording execution counts
  - x_i = no. of times basic block i is executed
- **X** is **valid** if its elements correspond to a feasible execution of the program
  - E.g., in the CFG on the right, for a valid **X**
    - $x_1 = x_6 = 1, x_2 = x_3 + 1, x_3 = x_5$



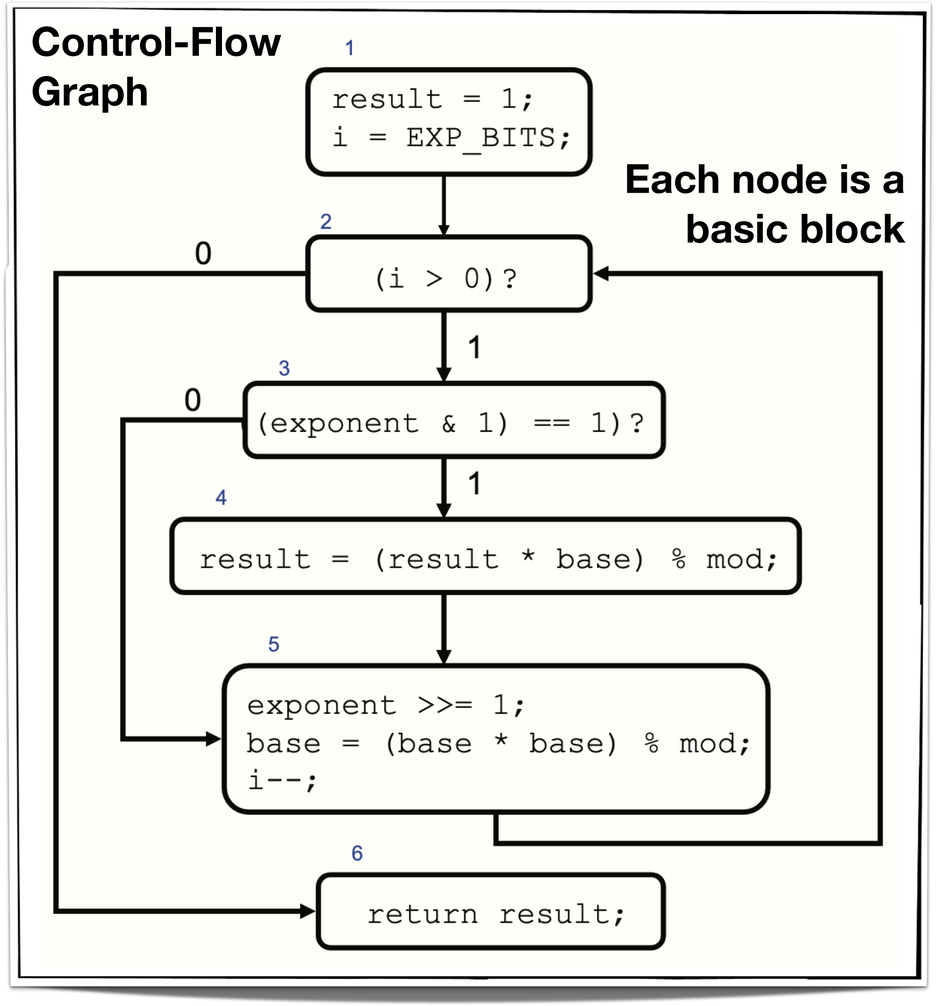
Optimization Formulation [2/3]

- Flow constraints
  - Unit flow at source: $x_1 = 1$ and $x_n = 1$
  - Conservation of flow: $x_i = \sum_{j \in P_i} d_{ji} = \sum_{k \in S_i} d_{ik}$
    - $d_{i,j}$ = no. of times the edge from node i to j is executed
    - $P_1 = \emptyset$ and $S_n = \emptyset$
- E.g., in the CFG on the right
  - $x_1 = 1$ and $x_6 = 1$
  - $x_1 = d_{12}$ and $x_2 = d_{12} + d_{52} = d_{23} + d_{26}$
  - $x_3 = d_{23} = d_{34} + d_{35}$ and $x_4 = d_{34} = d_{45}$
  - $x_5 = d_{35} + d_{45} = d_{52}$ and $x_6 = d_{26}$
- One valid solution: $\mathbf{X} = (1, 2, 3, 0, 1, 1)$



Optimization Formulation [3/3]

- Let w_i be an upper bound on the execution time of basic block i
  - WCET = maximum possible $\sum_{i=1}^{n} w_i x_i$ over all valid **X**
- Linear programming (LP) formulation
  - Find **X** that gives $\max_{x_i,\ 1 \le i \le n} \sum_{i=1}^{n} w_i x_i$
  - Subject to $x_1 = x_n = 1$ and $x_i = \sum_{j \in P_i} d_{ji} = \sum_{k \in S_i} d_{ik}$
- Drawbacks?



Logical Flow Constraints [1/2]

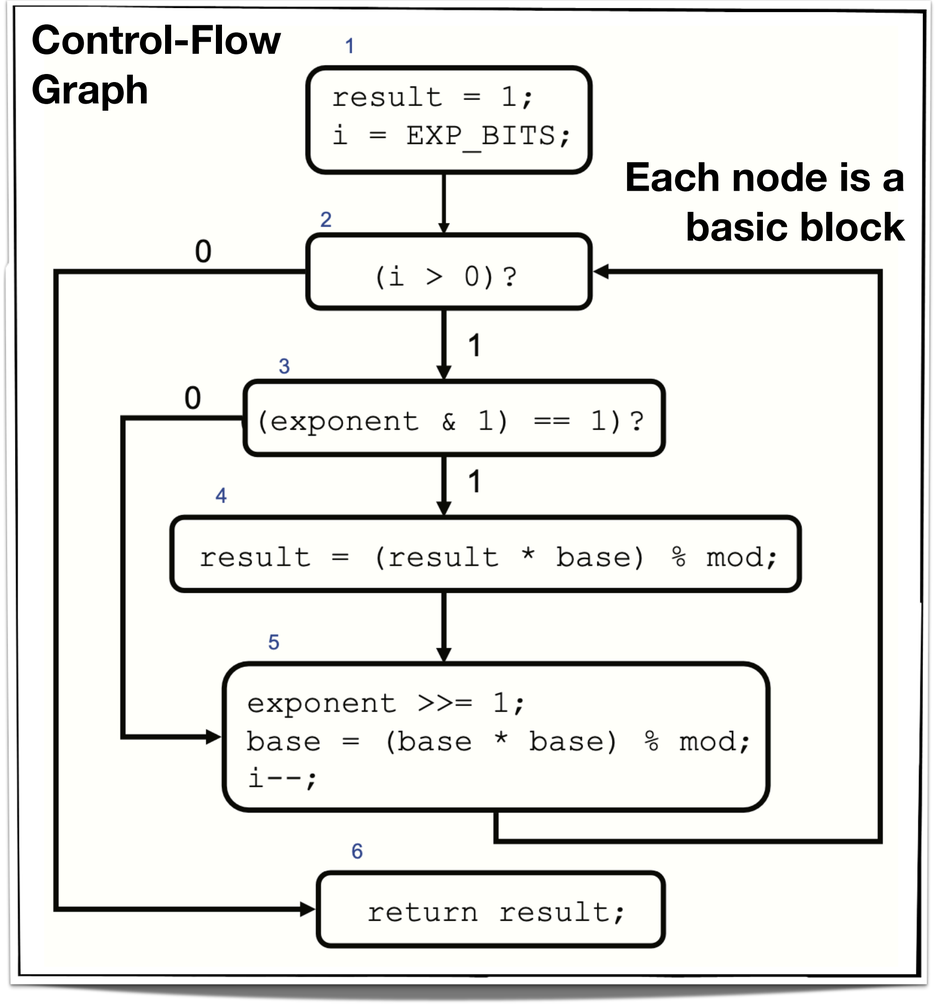
```
1  #define EXP_BITS 32
2
3  typedef unsigned int UI;
4
5  UI modexp(UI base, UI exponent, UI mod) {
6    int i;
7    UI result = 1;
8
9    i = EXP_BITS;
10   while(i > 0) {
11     if ((exponent & 1) == 1) {
12       result = (result * base) % mod;
13     }
14     exponent >>= 1;
15     base = (base * base) % mod;
16     i--;
17   }
18   return result;
19 }
```

If e is even: $b^e = (b^2)^{\frac{e}{2}}$

If e is odd: $b^e = b \times (b^2)^{\frac{e-1}{2}}$

How many times around the while loop?

$x_3 \leq 32$



Logical Flow Constraints [2/2]

```
1  #define CLIMB_MAX 1.0
```

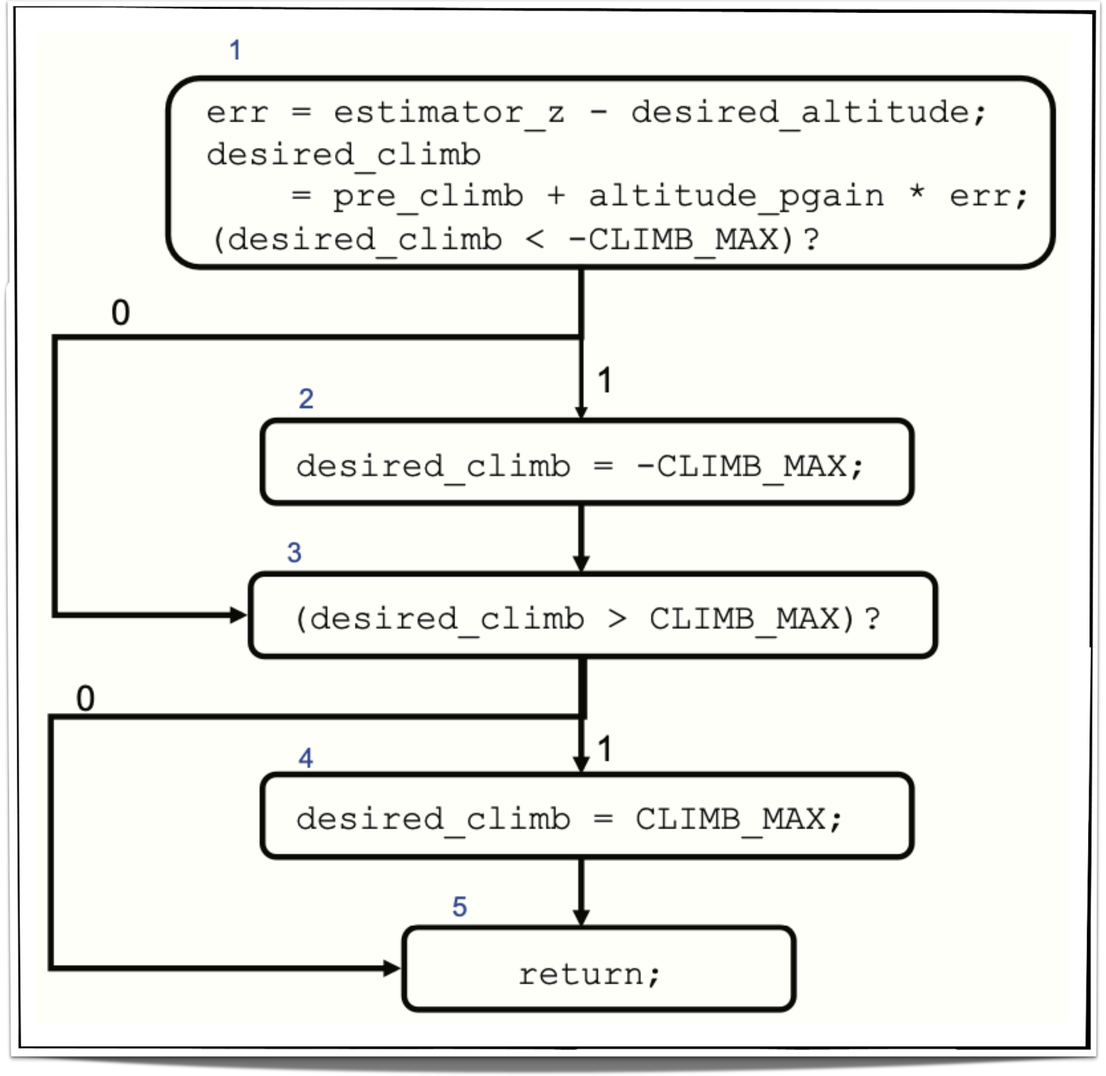
```
3  void altitude_control_task(void) {
4      ...
5      err = estimator_z - desired_altitude;
6      desired_climb
7            = pre_climb + altitude_pgain * err;
8      if (desired_climb < -CLIMB_MAX) {
9        desired_climb = -CLIMB_MAX;
10     }
11     if (desired_climb > CLIMB_MAX) {
12       desired_climb = CLIMB_MAX;
13     }
14     return;
15 }
```

Are there any infeasible paths?

$$
\begin{aligned}
x_1 &= 1 \\
x_5 &= 1 \\
x_1 &= d_{12} + d_{13} \\
x_2 &= d_{12} = d_{23} \\
x_3 &= d_{13} + d_{23} = d_{34} + d_{35} \\
x_4 &= d_{34} = d_{45} \\
x_5 &= d_{35} + d_{45}
\end{aligned}
$$

A solution for the above system of equations is

$$x_1 = x_2 = x_3 = x_4 = x_5 = 1,$$



Logical Flow Constraints [2/2]

```
1  #define CLIMB_MAX 1.0

3  void altitude_control_task(void) {
4      ...
5      err = estimator_z - desired_altitude;
6      desired_climb
7            = pre_climb + altitude_pgain * err;
8      if (desired_climb < -CLIMB_MAX) {
9        desired_climb = -CLIMB_MAX;
10     }
11     if (desired_climb > CLIMB_MAX) {
12       desired_climb = CLIMB_MAX;
13     }
14     return;
15 }
```

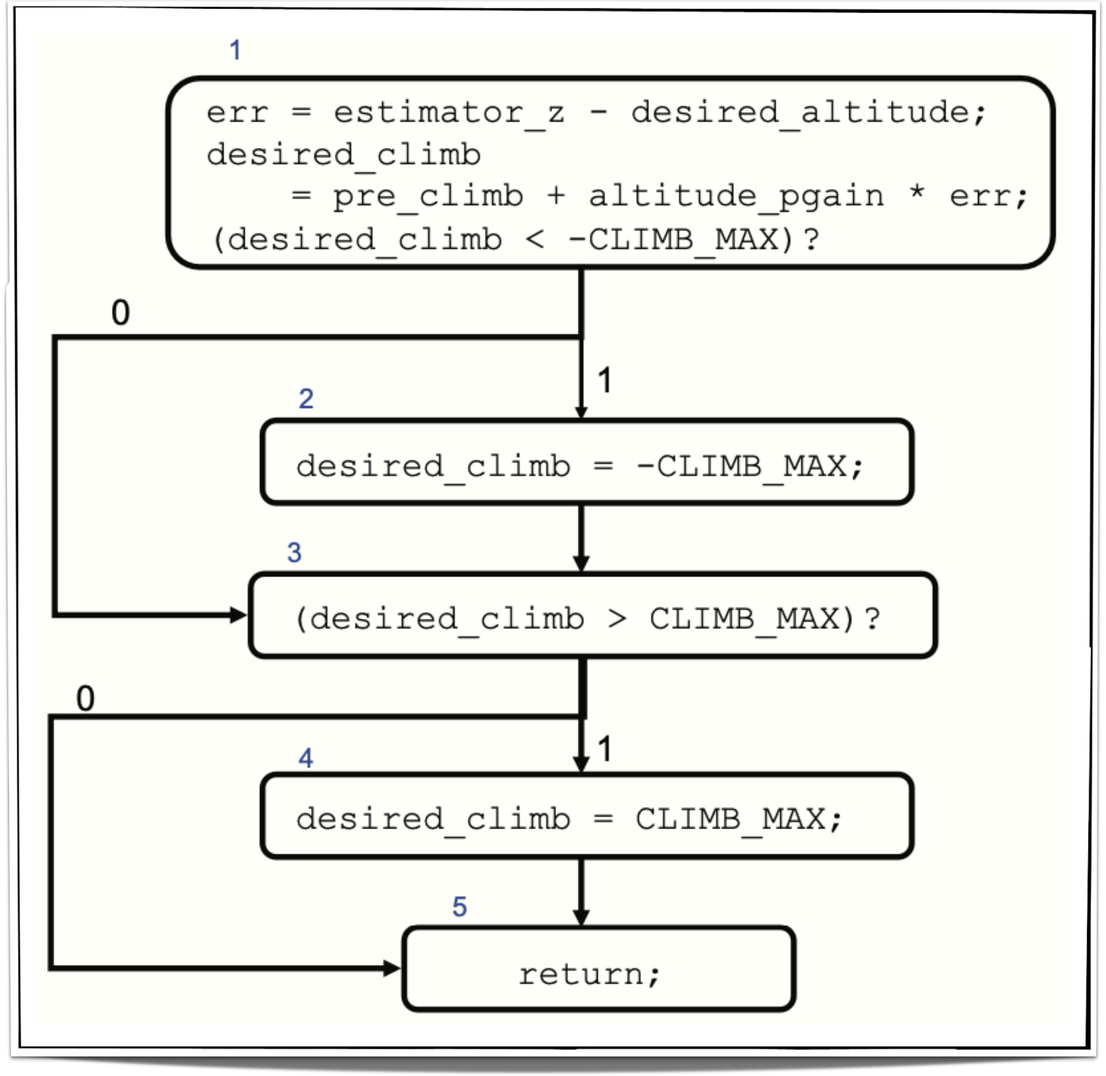
Are there any infeasible paths?

$d_{12} + d_{34} \leq 1$

$$
\begin{aligned}
x_1 &= 1 \\
x_5 &= 1 \\
x_1 &= d_{12} + d_{13} \\
x_2 &= d_{12} = d_{23} \\
x_3 &= d_{13} + d_{23} = d_{34} + d_{35} \\
x_4 &= d_{34} = d_{45} \\
x_5 &= d_{35} + d_{45}
\end{aligned}
$$

A solution for the above system of equations is

$$x_1 = x_2 = x_3 = x_4 = x_5 = 1,$$



Bounds for Basic Blocks

- How to estimate upper bound w_i on the execution time of basic block i?

- Challenges
  - Requires detailed micro-architectural modelling
  - Cache miss versus a hit can change latency by a factor of 100
    - If the analysis does not differentiate between cache hits and misses, the computed bound may be a hundred times larger than the actual execution time

Examples

```
1  void testFn(int *x, int flag) {
2    while (flag != 1) {
3      flag = 1;
4      *x = flag;
5    }
6    if (*x > 0)
7      *x += 2;
8  }
```

Assuming x is not NULL

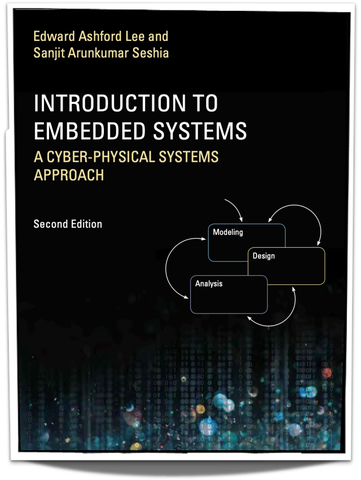
- Draw the Control Flow Graph for this program.
- Is there a bound on the number of iterations of the while loop? Justify your answer?
- How many total paths does this program have? How many of them are feasible, and why?
- Write down the system of flow constraints, including any logical flow constraints, for the control-flow graph of this program?

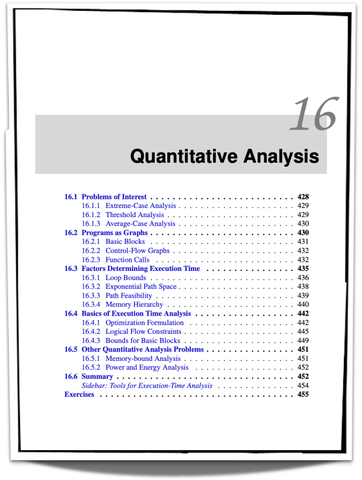
Summary

Contact

- **Name: Arpan B. Gujarati**
- **Email: arpanbg@cs.ubc.ca**
- **Web: https://arpangujarati.github.io/**

- Introduction to **Cyber-Physical Systems (CPS)**
- **Real-time systems (RTS)** and the problem of **timing analysis**
- **Program analysis** for estimating the Worst-case Execution Time (WCET)
- Reference
  - Ch. 16, Introduction to Embedded Systems, 2nd Ed.
  - Available online at LeeSeshia.org
- Acknowledgements for slides
  - Max Planck Institute for Software Systems (© MPI-SWS)
  - Real-time System Design (© CPEN 432)




16
Quantitative Analysis