FMCAD 2011 (Austin, Texas)

Timing Analysis of Concurrent Programs under Context Bounds

Jonathan Kotker, Dorsa Sadigh, Sanjit Seshia
University of California, Berkeley

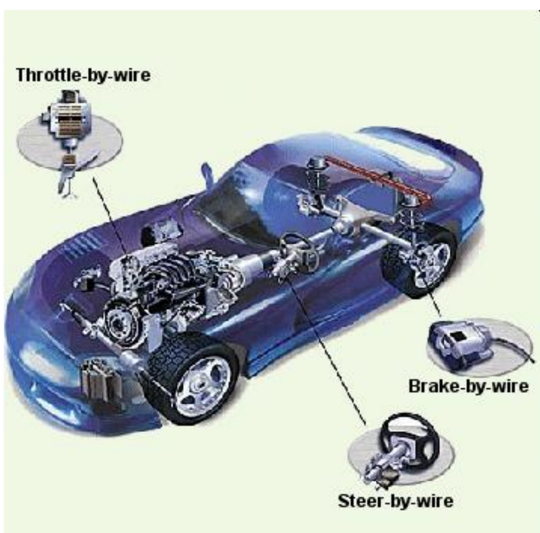
Motivation: Cyber-Physical Systems

Cyber-Physical = Computation + Physical Processes

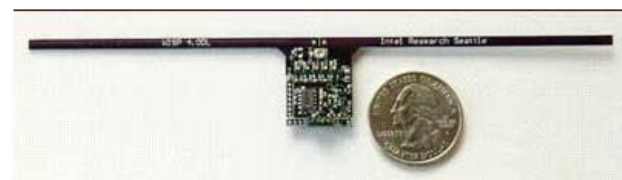
Quantitative analysis of programs is crucial:

How long does it take?

How much energy does it consume?



Safety-critical embedded systems: Does the brake-by-wire software always actuate the brakes within 1 ms?



Energy-limited sensor nets: How much energy must the sensor node harvest for RSA encryption?

Timing Analysis Problems

- Worst-case execution time (WCET) estimation
- Estimating distribution of execution times
- Threshold property: produce test cases that violates program deadline

All three problems can be solved if we could *predict the execution time of arbitrary program paths*.

Timing Analysis with Interrupts

Current code-level analysis techniques assume no interrupts, but practical embedded software is interrupt-driven

NASA Toyota Unintended Acceleration Report

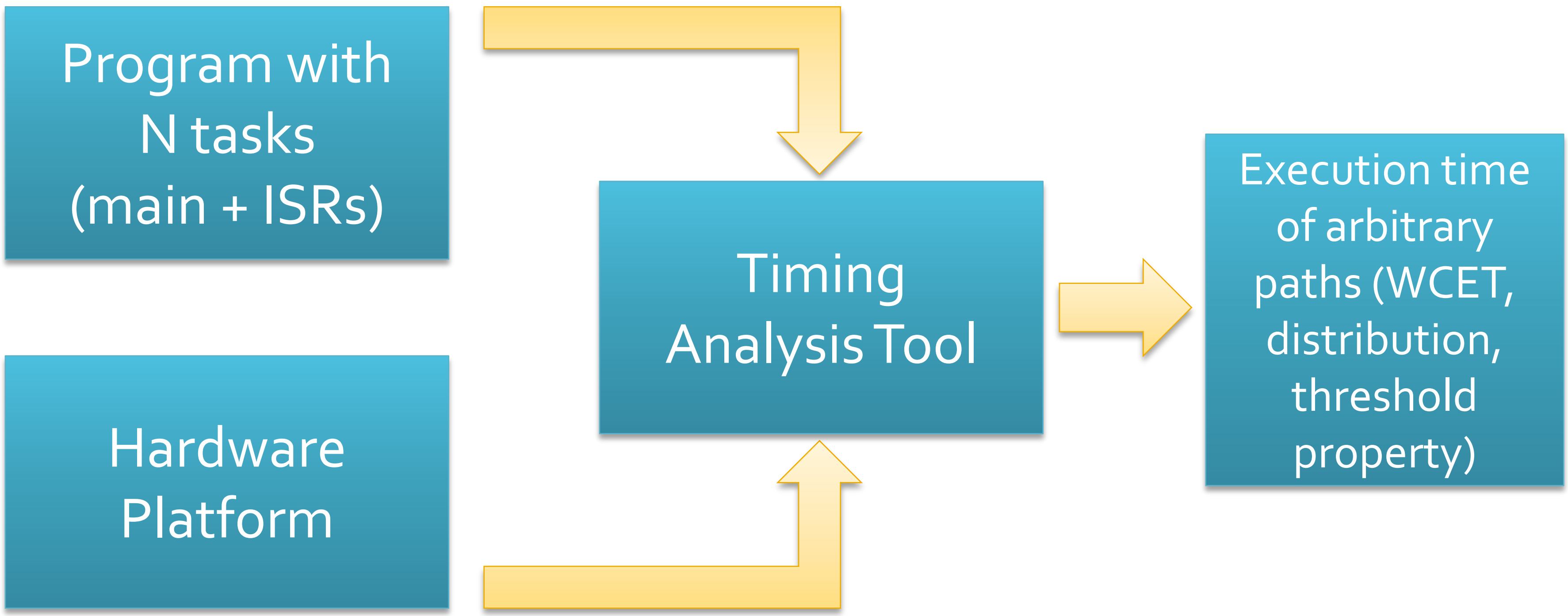
Lack of support in timing analysis tools for interrupt-driven code

Timing Analysis with Interrupts

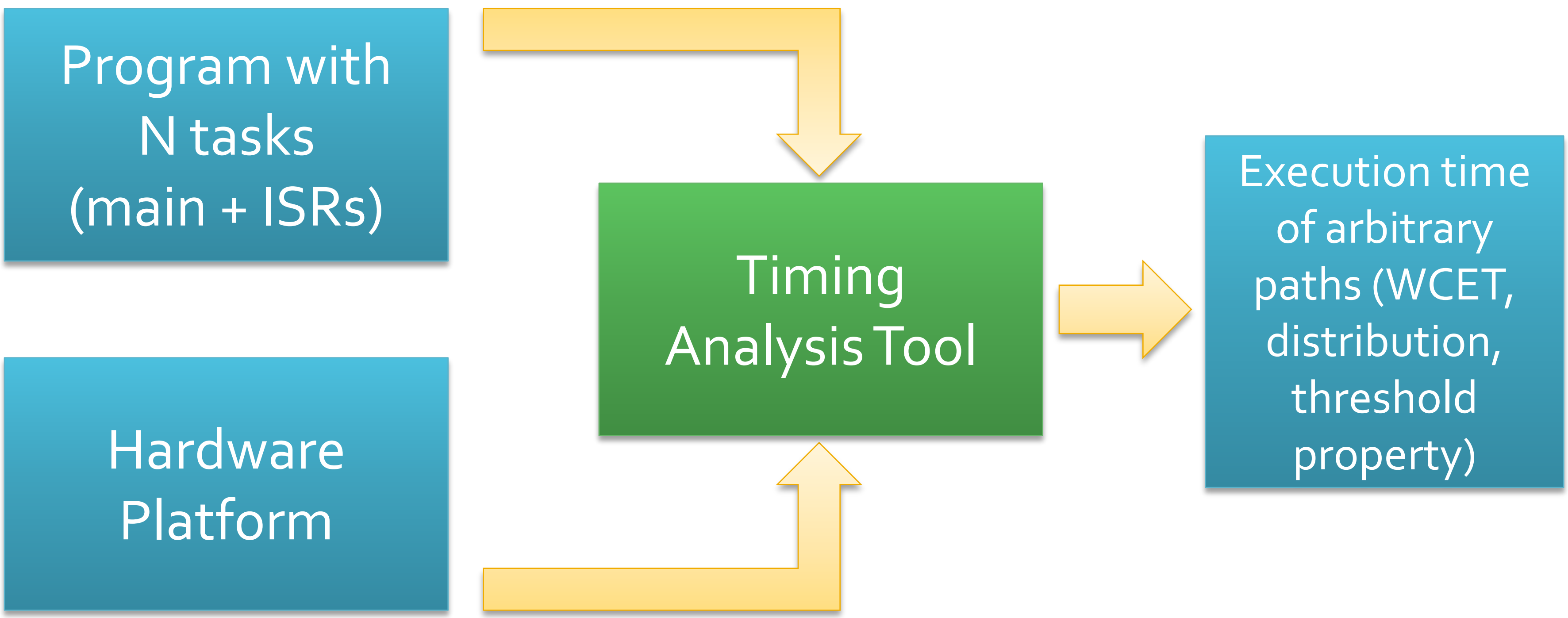
Why is timing analysis of interrupt-driven software a hard problem?

- *Path Explosion:* Unbounded number of interleavings of tasks and interrupt service routines (ISRs)
- *Platform Modeling:* Interrupts impact processor operation

Problem Definition



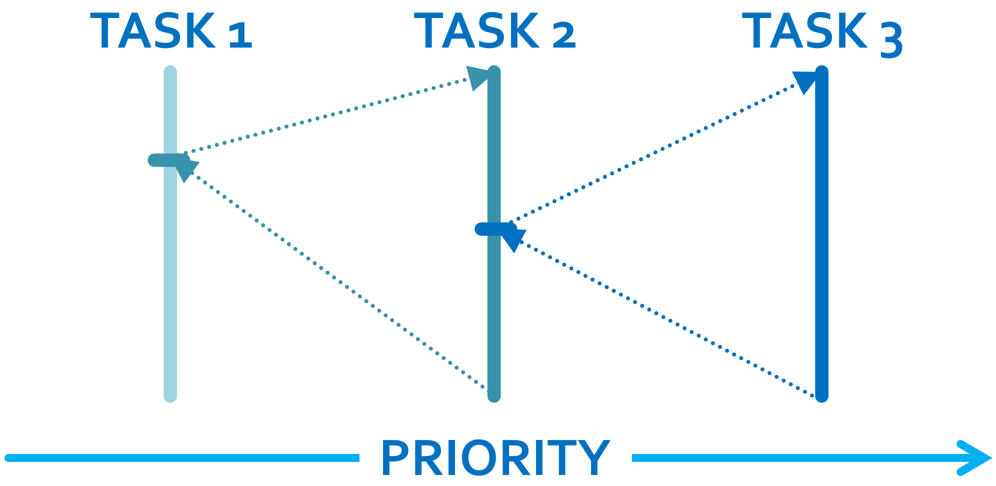
Problem Definition



Assumptions in this work - 1

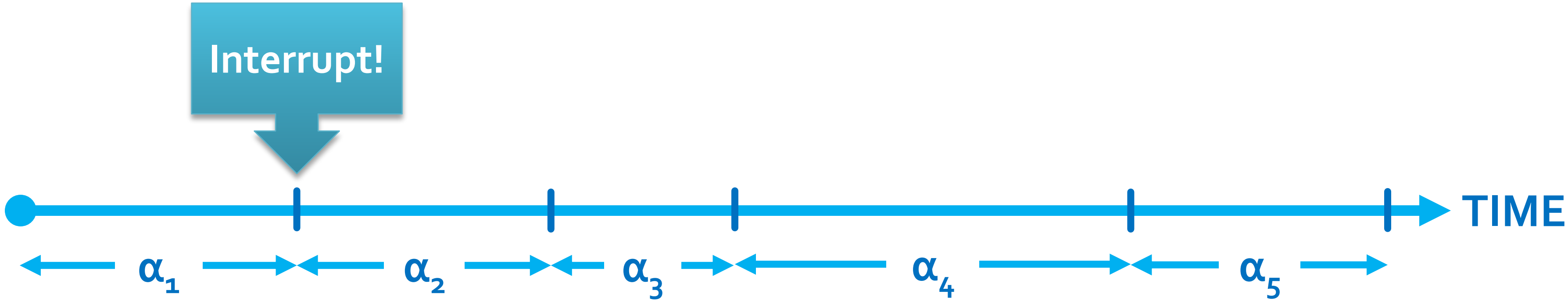
Priority pre-emptive scheduling

- Tasks are ordered by priority
- If a higher-priority task interrupts a lower-priority task, the lower-priority task cannot later interrupt the higher-priority task



Assumptions in this work - 2

Lower-bound on interrupt inter-arrival time



There exists an $\boldsymbol{\alpha > 0}$ such that $\boldsymbol{\alpha < \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5}, \ldots$

“Assumptions” in this work - 3

Atomicity

Code should ideally be structured into atomic sections, perhaps by disabling and re-enabling interrupts*

* Our approach works with any atomicity model.

Contributions

- With these three assumptions, we compute a context bound and perform context-bounded analysis (Qadeer and Rehof, 2005).
- Number of interleaved paths can still be exponential in the context bound
  - Obtaining measurements can be tedious
  - Basis paths drastically reduce number of paths to be measured to be polynomial in size of sequential program
- Experiments on a real embedded platform show that WCET and execution times of arbitrary paths can be predicted accurately

Related Work

Context-bounded analysis

- *Context-Bounded Model Checking of Concurrent Software*
  Shaz Qadeer and Jakob Rehof (2005)
  - Introduces context-bounded analysis
  - Does not address timing analysis
- *One Stack to Run Them All: Reducing Concurrent Analysis to Sequential Analysis under Priority Scheduling*
  N. Kidd, S. Jagannathan, J. Vitek (2010)
  - Transforms a concurrent program with priority pre-emptive scheduling to a sequential program
  - Reduction applies for reachability only

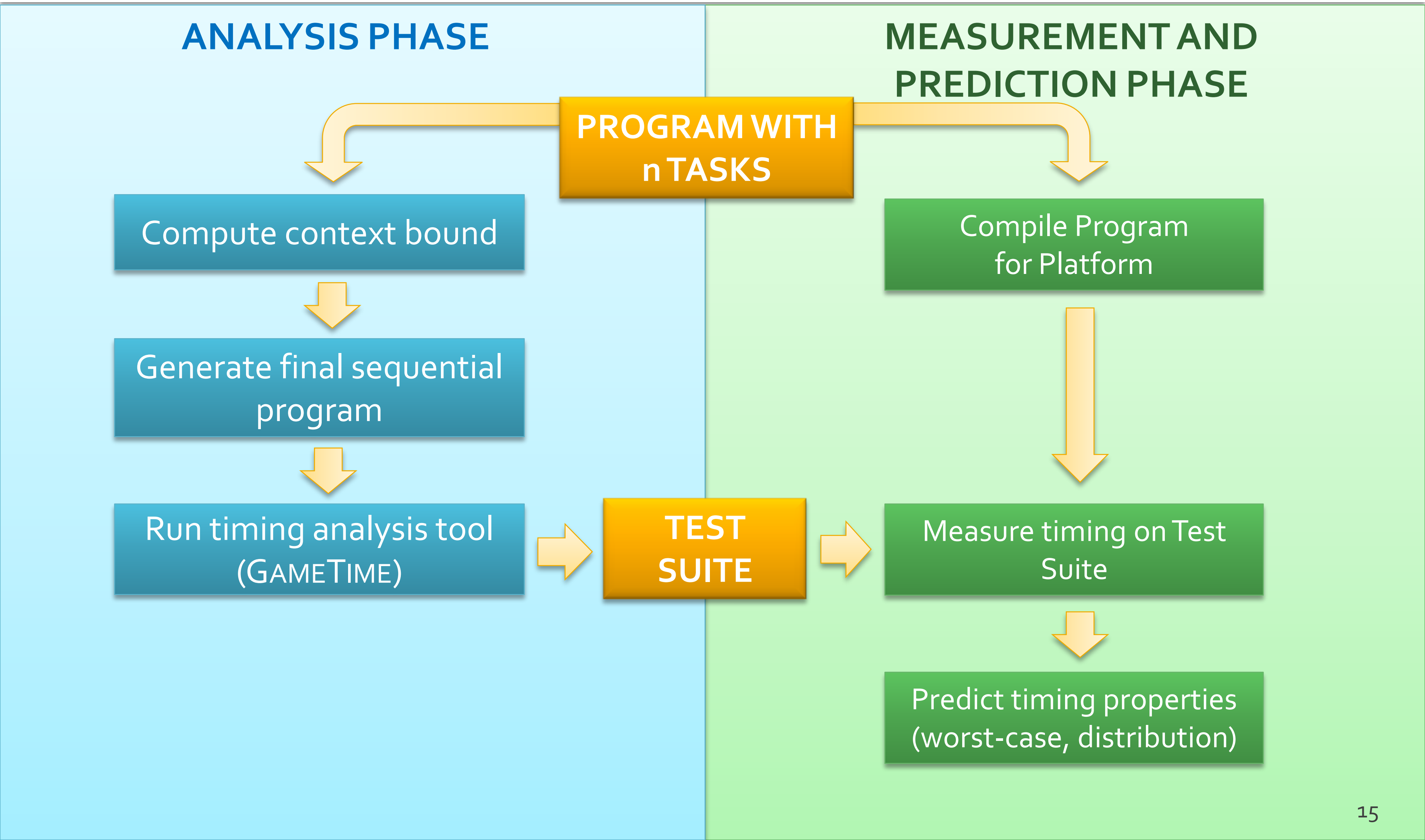
Related Work
Timing Analysis

- Schedulability Analysis
  - Analyzes if a task can meet its deadline despite pre-emption
  - Treats tasks as primitive objects
  - Does not capture code correlation across tasks
- *Deadline Analysis of Interrupt-Driven Software*, Dennis Brylow and Jens Palsberg (2004)
  - Assembly-level
  - Threshold property, not WCET analysis
  - Assumes WCET is already given

Outline

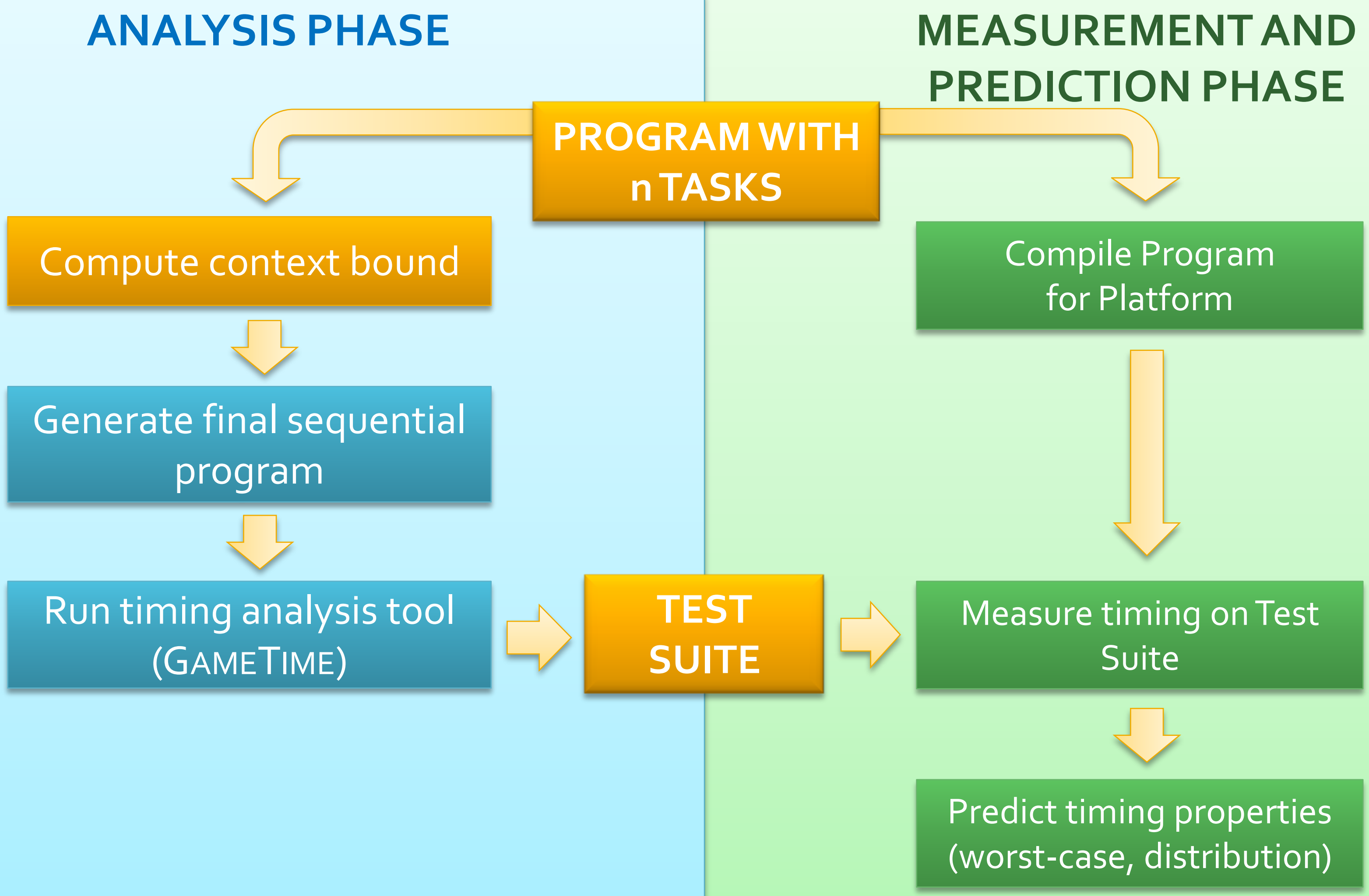
- Approach
- Experimental Setup
- Hardware
- Results
- Summary and Future Work

Approach

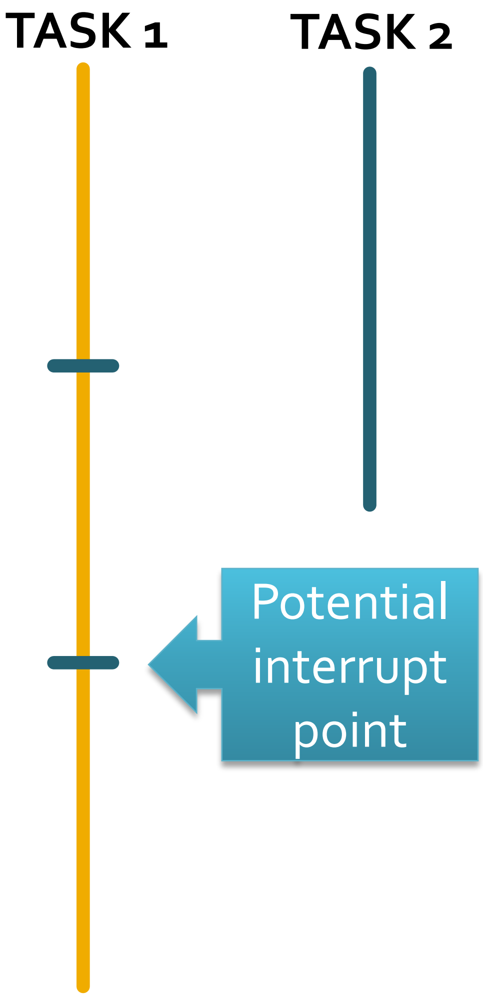


ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

Approach


ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

Context Bound

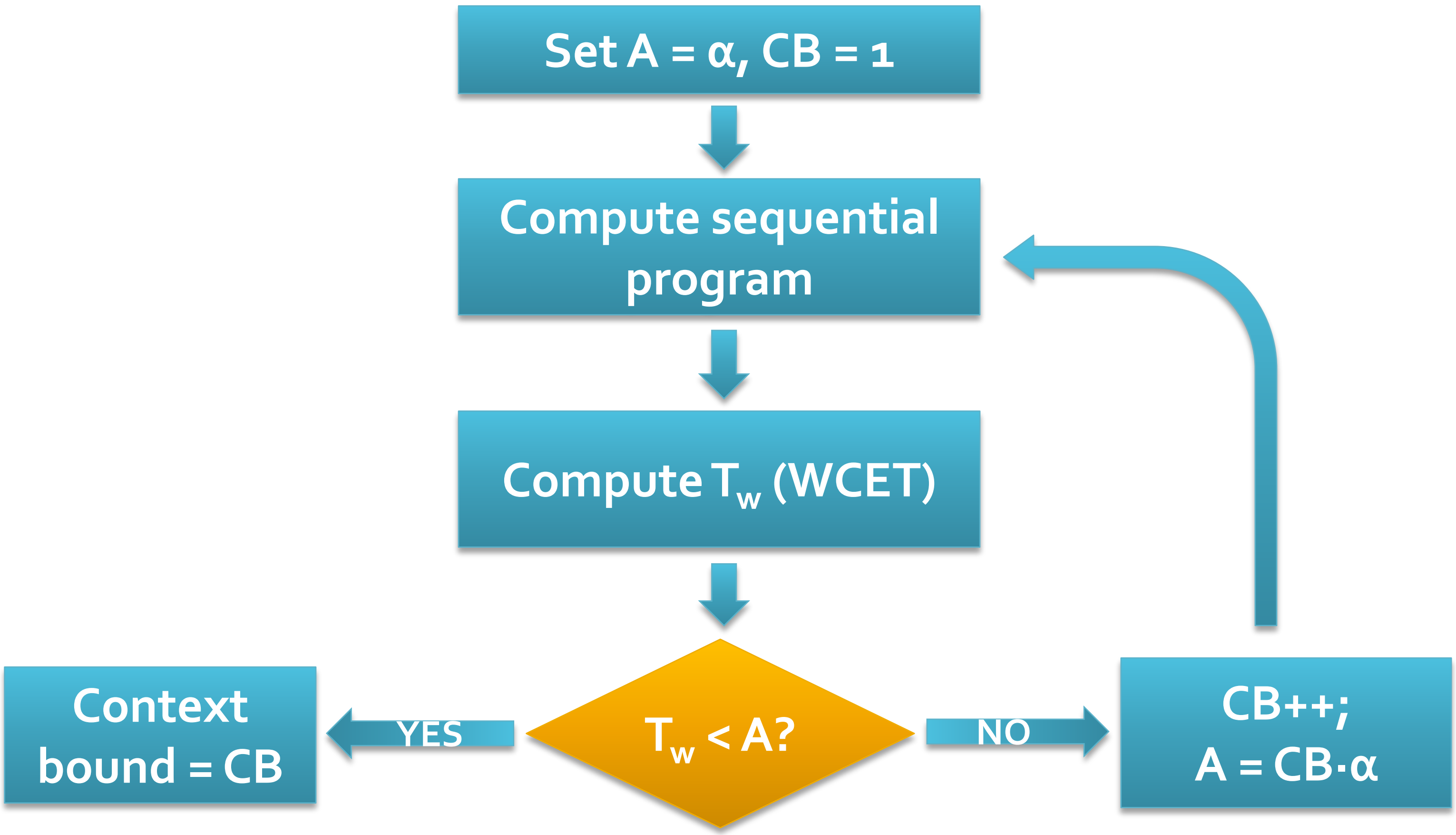


Bound on total number of "context switches" between tasks

For a context bound of 1, the first task can be interrupted at most once, at either of the two interrupt points.

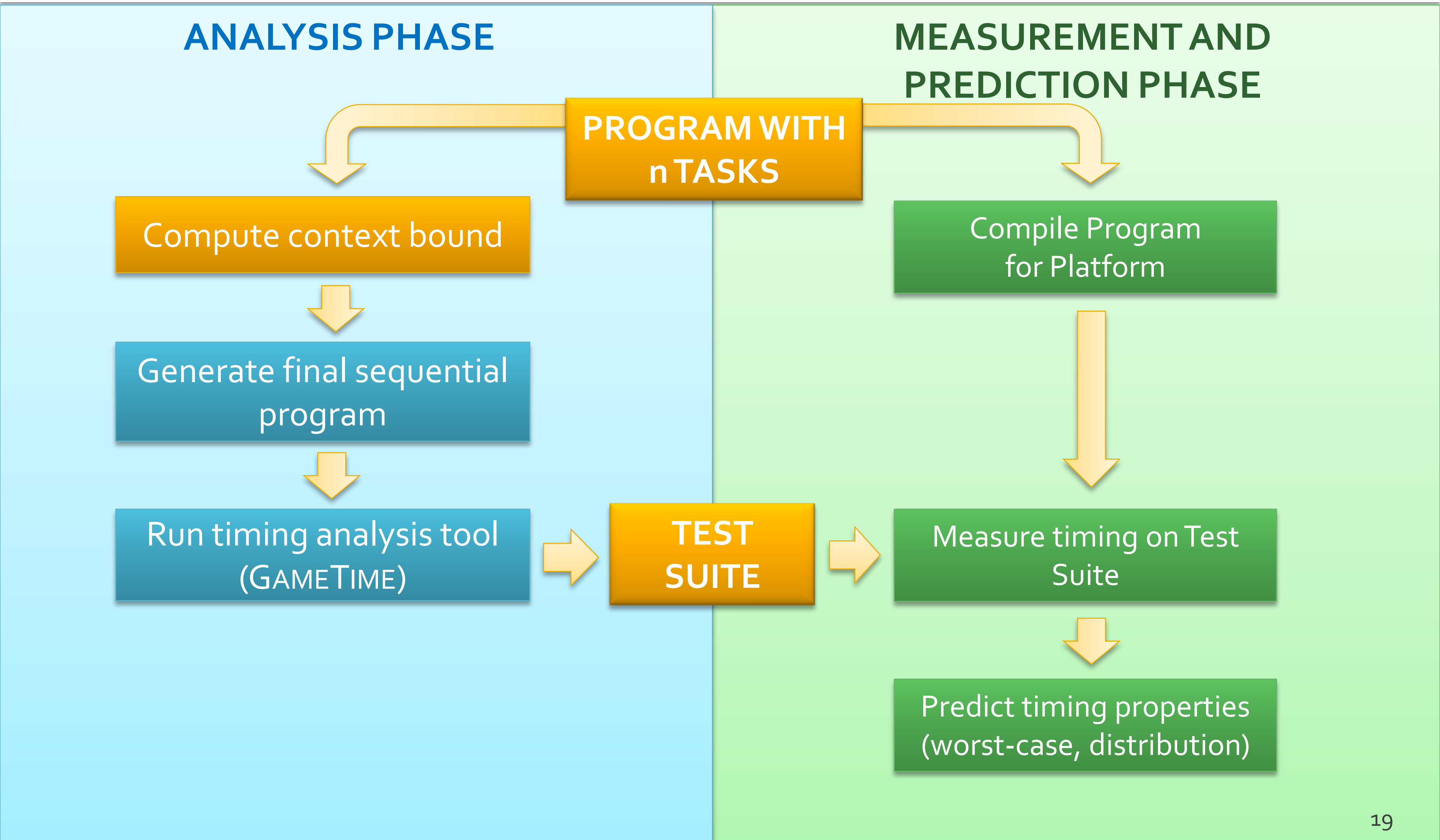
Finding a Context Bound

Lower bound on interrupt inter-arrival time: α



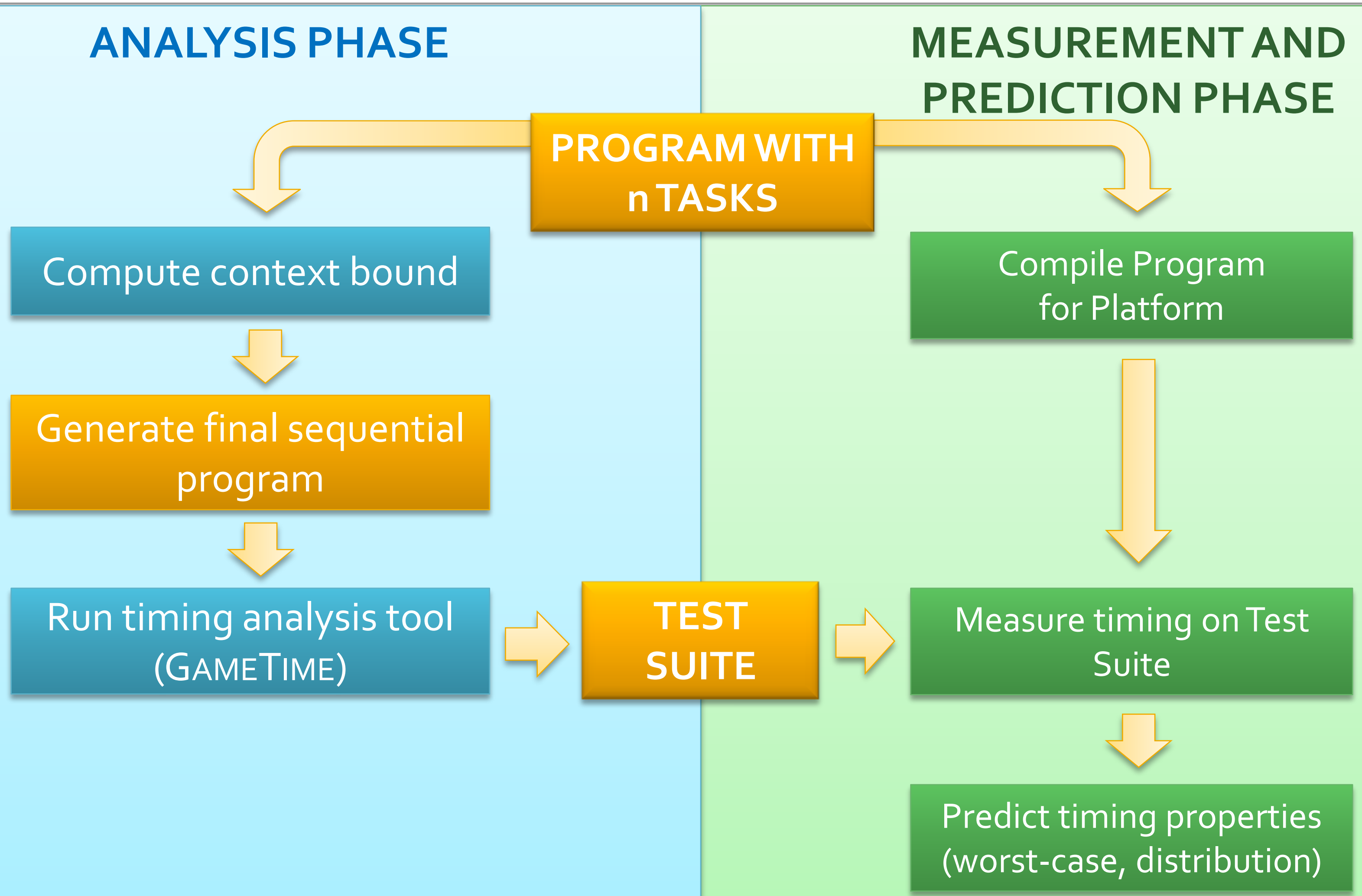
Loop terminates if ISR services the interrupt in time less than α

Approach

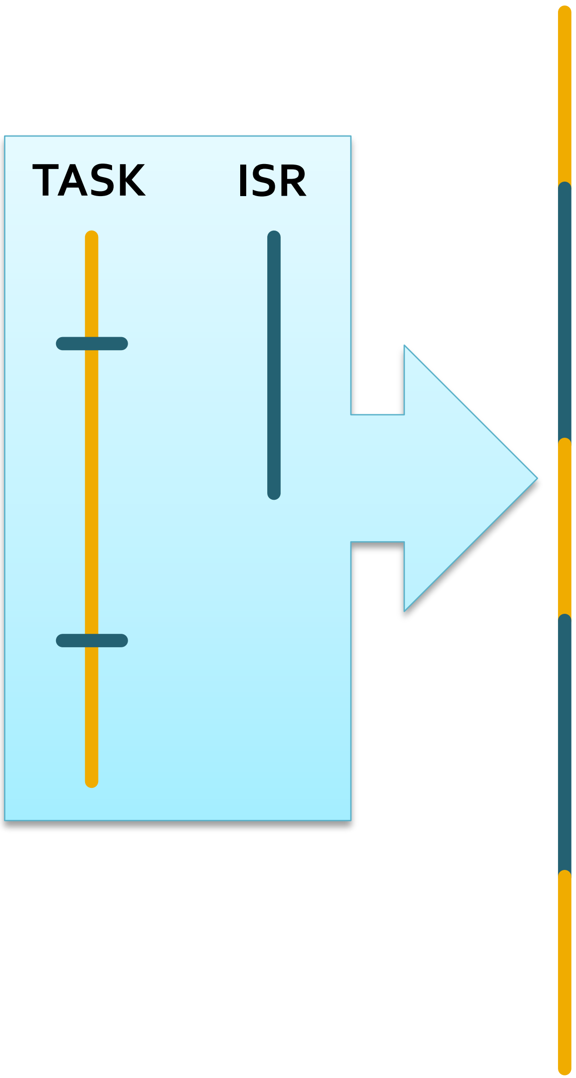


ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

Approach

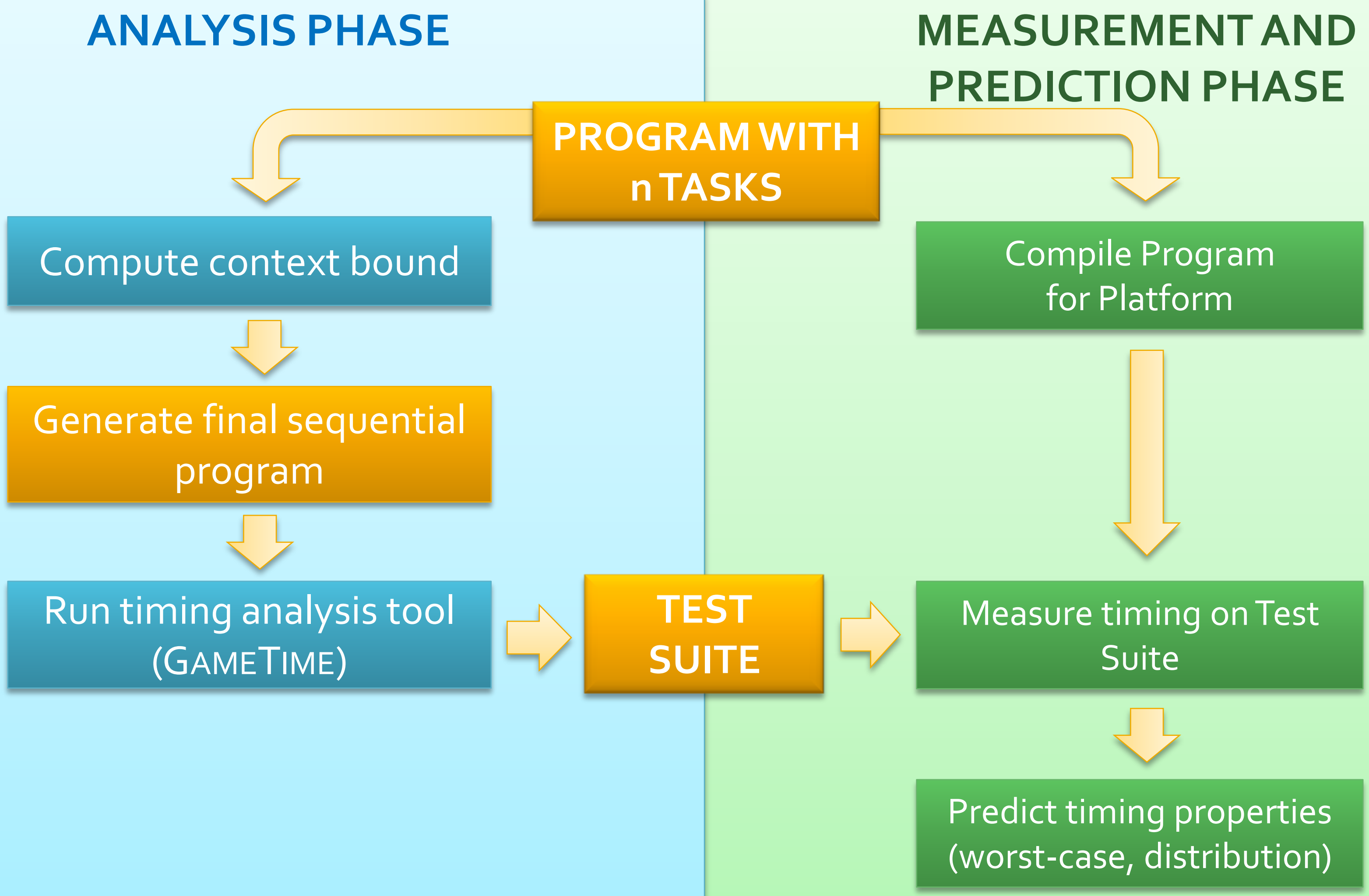

ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

Generating Sequential Programs



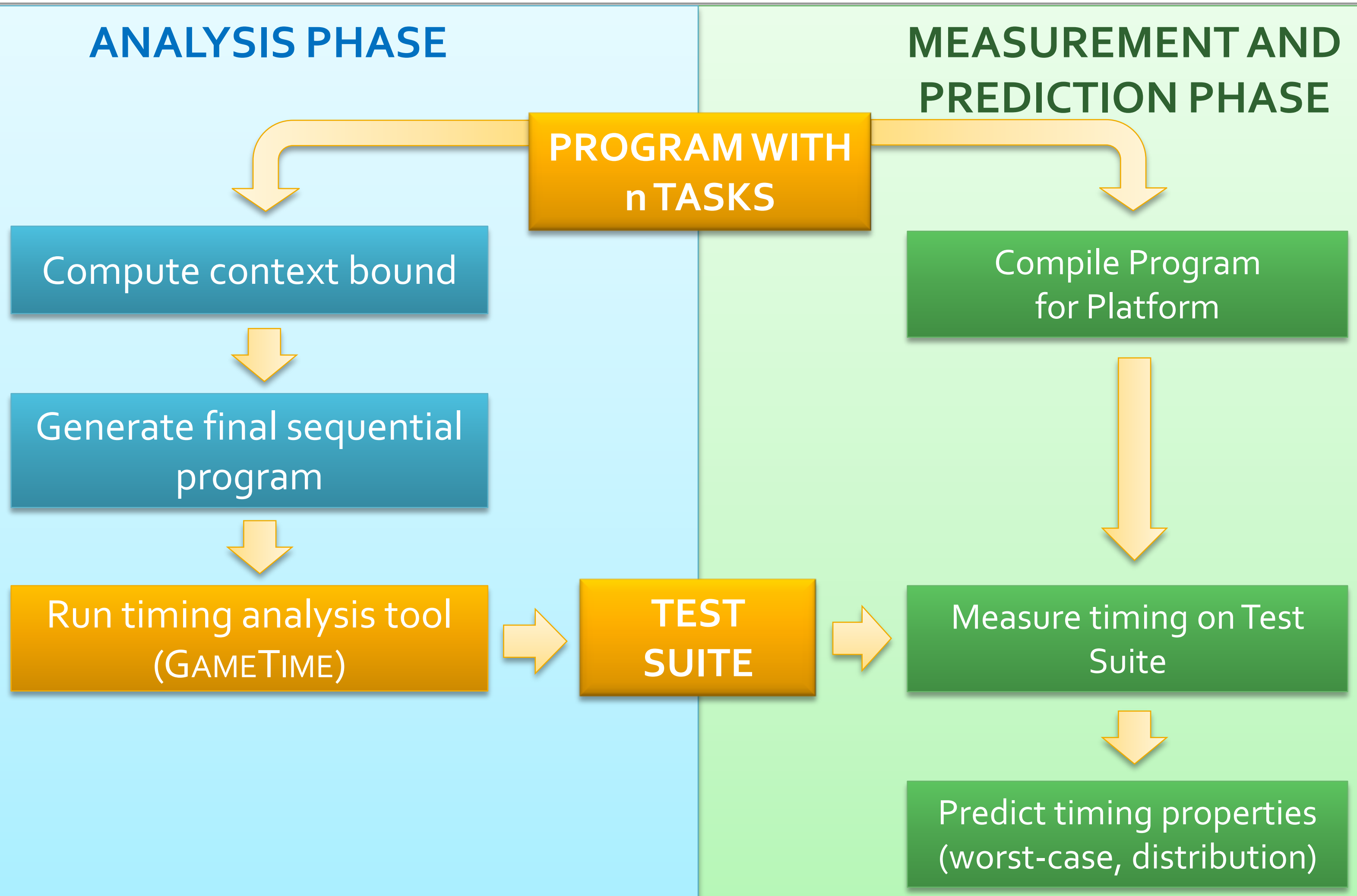
Model occurrence of interrupt points as "function calls" and bound the number of these "function calls" (using a global counter)

Approach



ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

Approach


ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

Basis Paths

Example: Modular Exponentiation

- Common operation in cryptography, used for public-key encryption and decryption.
- "What is `base^exponent % prime`?"
- Exponentiation is performed using ***square-and-multiply***, where the exponent is progressively divided by two, while the base is progressively squared.

Modular Exponentiation

2-bit exponent

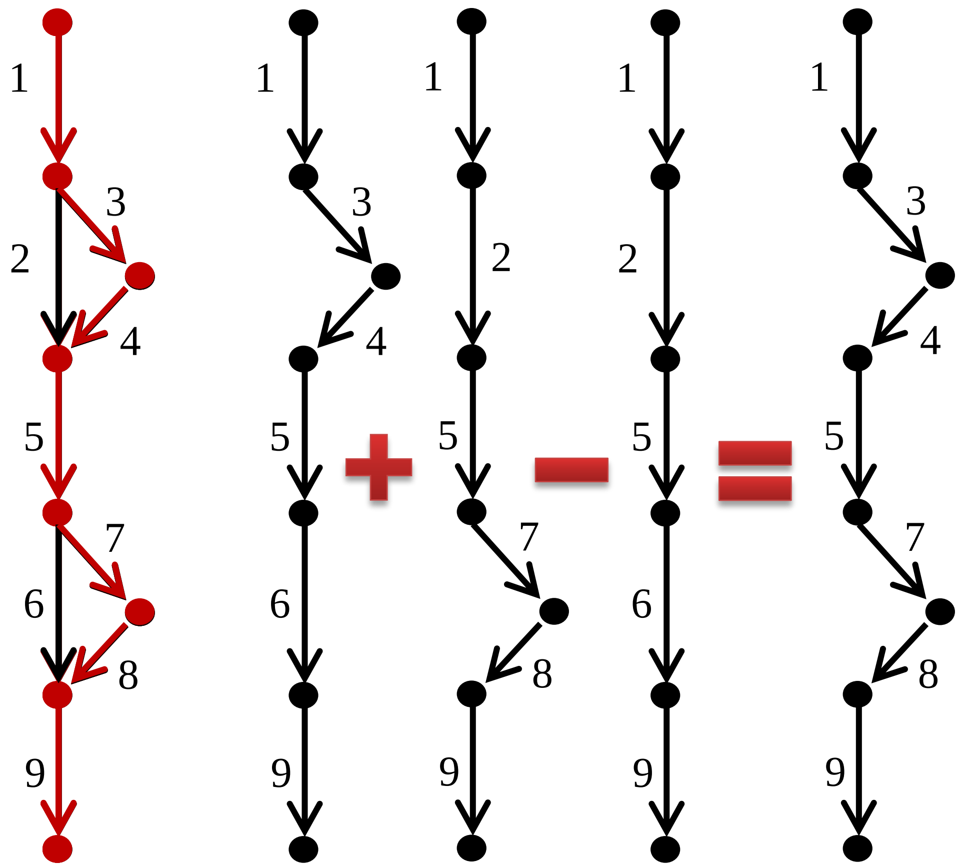
```
modexp(base, exponent) {
    result = 1;

    if ((exponent & 1) == 1) {
        result = (result * base) % p;
    }
    exponent >>= 1;
    base = (base * base) % p;

    if ((exponent & 1) == 1) {
        result = (result * base) % p;
    }
    exponent >>= 1;
    base = (base * base) % p;

    return result;
}
```

Basis Paths



(a) CFG

(b) Basis paths x_1, x_2, x_3

(c) Additional path x_4

Edge labels indicate **Edge IDs** and **positions** in vector representation

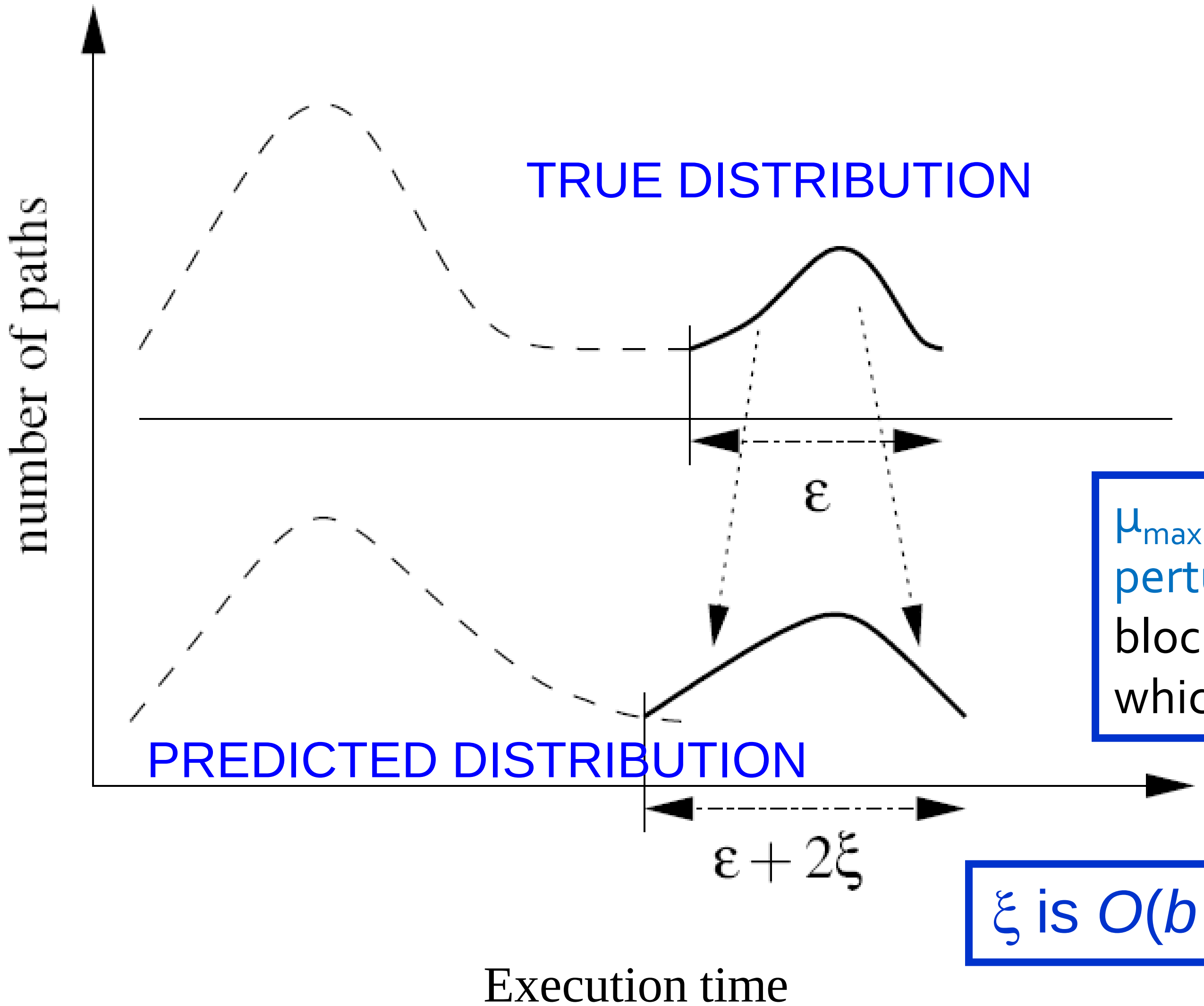
$x_1 = (1, 1, 0, 0, 1, 1, 0, 0, 1)$
$x_2 = (1, 0, 1, 1, 1, 1, 0, 0, 1)$
$x_3 = (1, 1, 0, 0, 1, 0, 1, 1, 1)$

$x_4 = (1, 0, 1, 1, 1, 0, 1, 1, 1)$

$x_4 = x_2 + x_3 - x_1$

(d) Vector representations

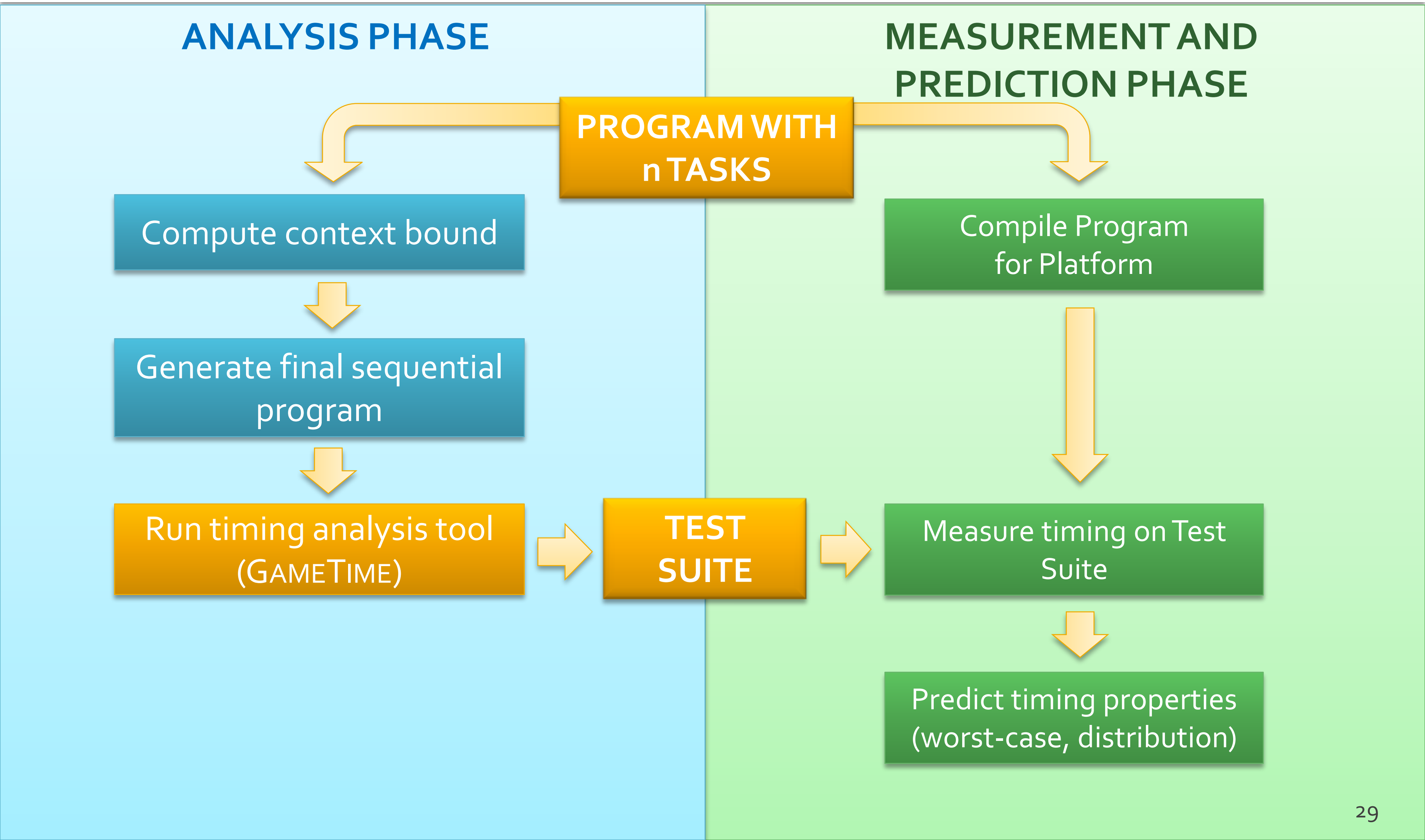
Theorem on Estimating Program Path Timing (Pictorial view)



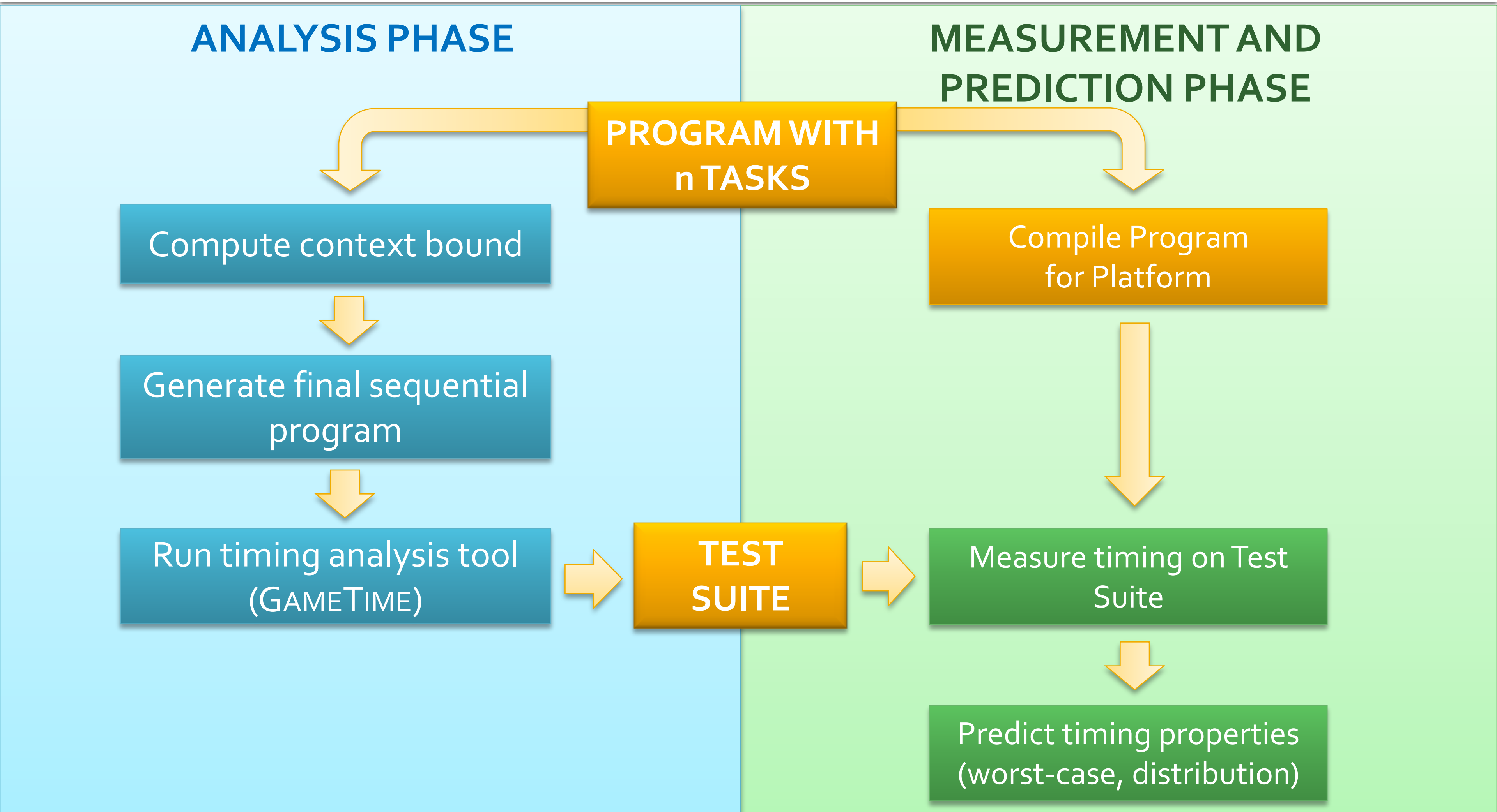
μ_{max} bounds mean perturbation to basic block timing based on which path it lies on

ξ is $O(b\ \mu_{max})$

Approach


ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

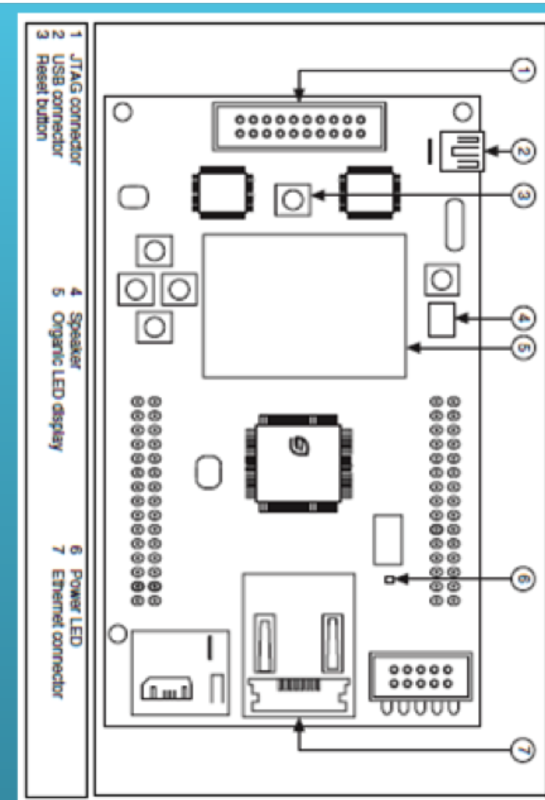
Approach


ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

Platform

Luminary Micro Interface to iRobot Create

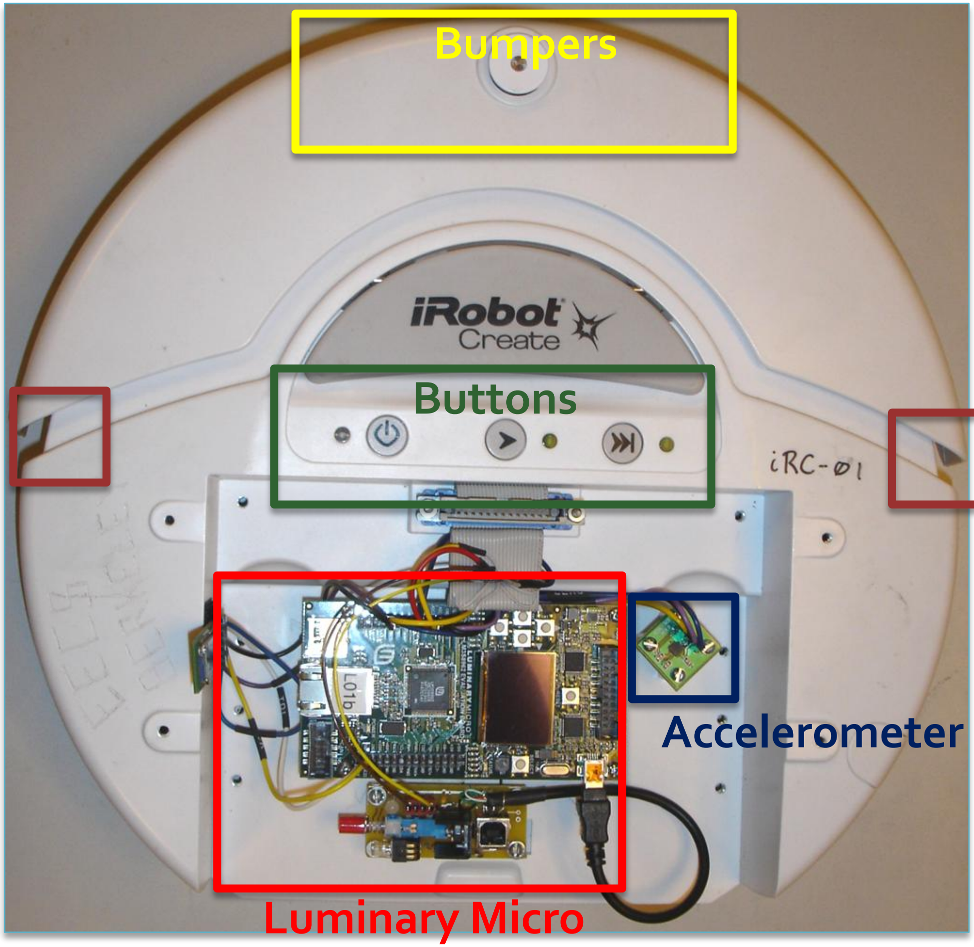
- LM3S8962
- 32 Bit ARM Cortex M3
  - 5 stage pipeline
- UART interface to iRobot Create
- No cache
- No OS



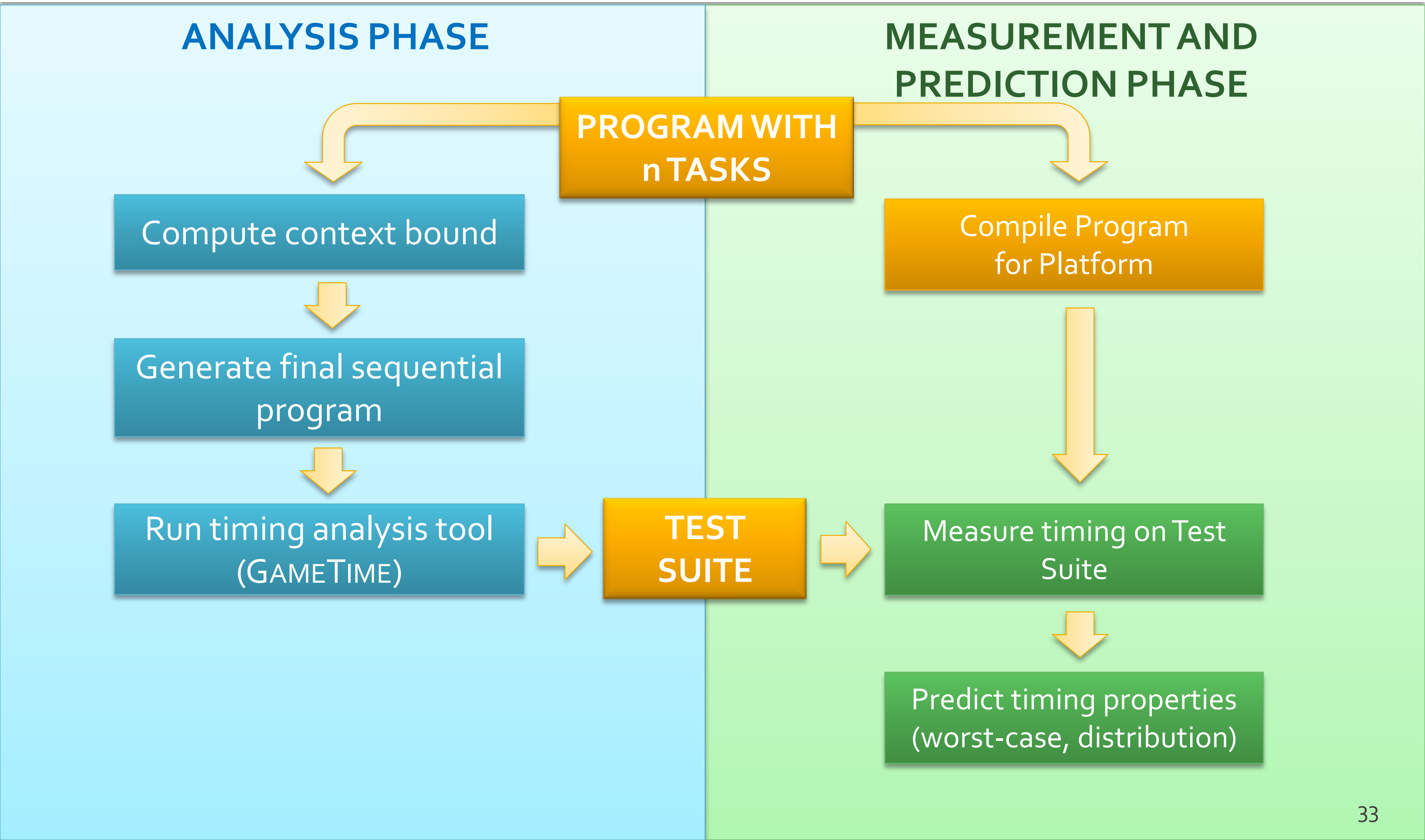
Sensors

iRobot Create

- ADXL-322 accelerometer
- iRobot sensors
  - Buttons
  - Bumpers
  - Cliff sensors
- Use ISRs for accelerometer and sensor

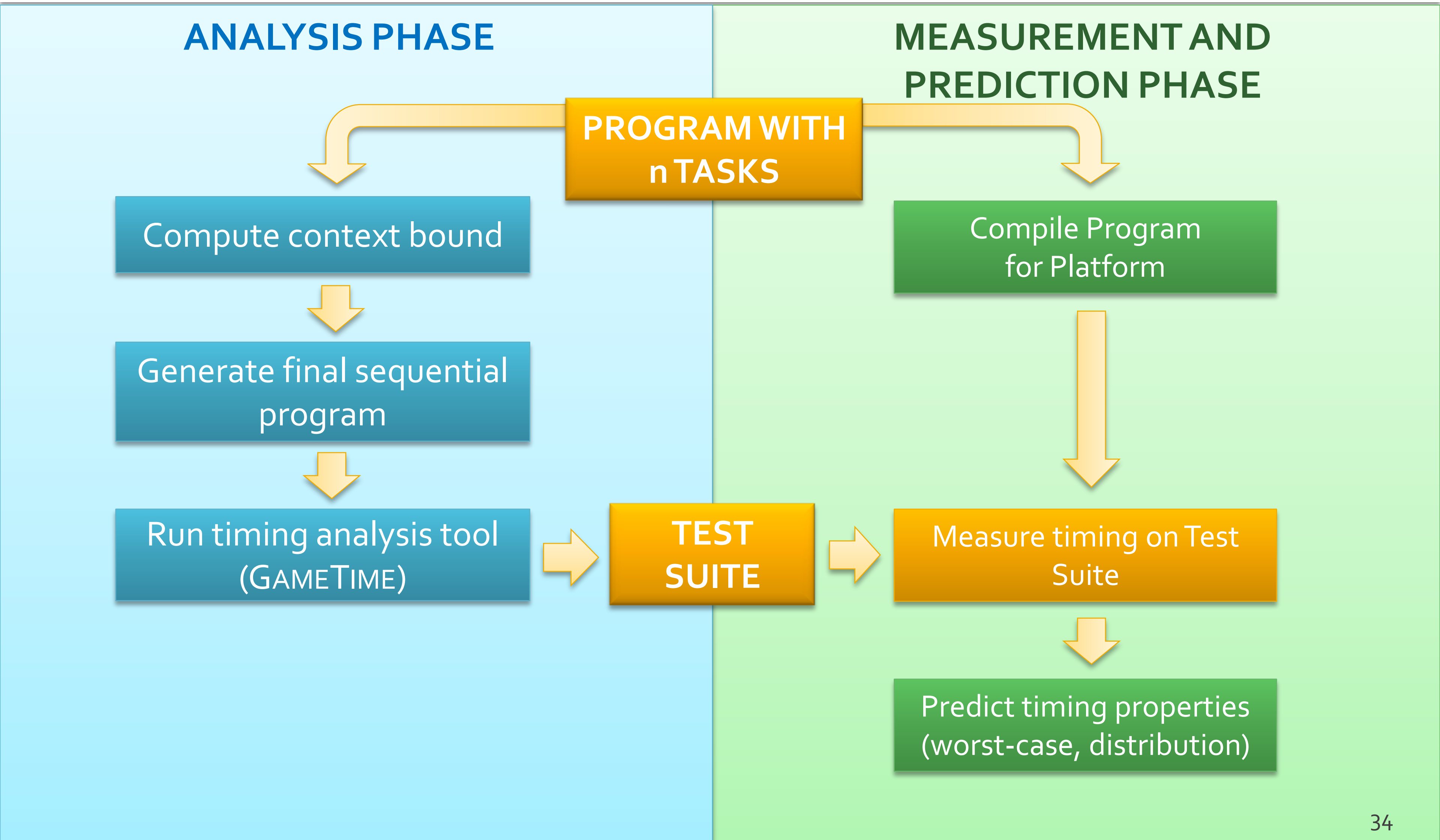


Approach



ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

Approach


ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

Test Suite

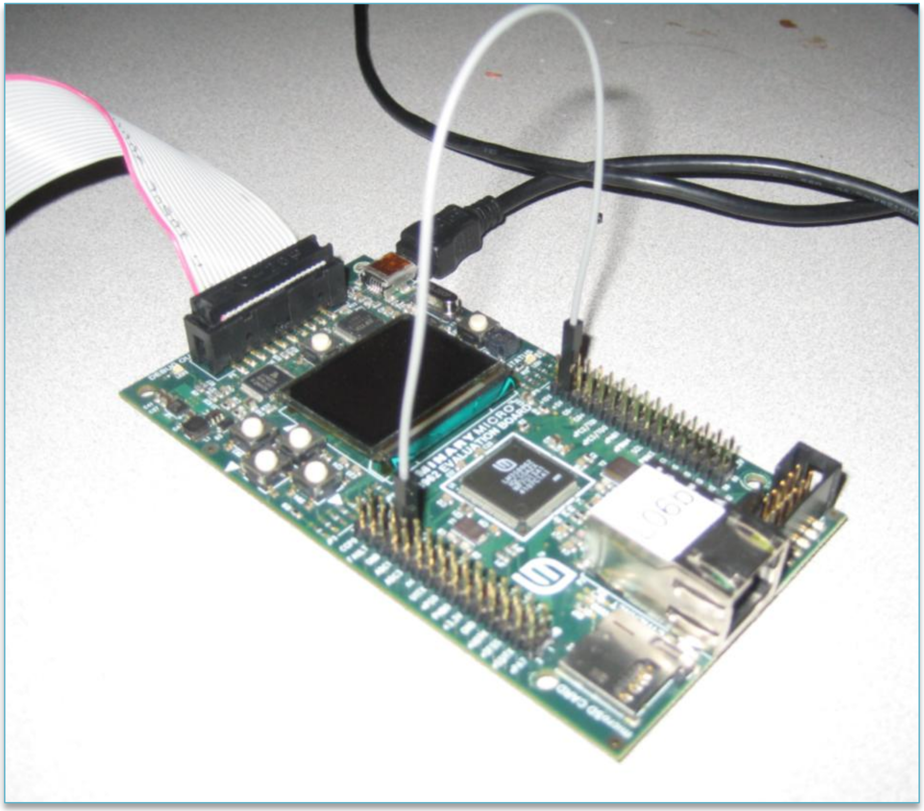
- Test suite are test cases that drive the program along basis paths in sequential code
- Each test case describes initial values for variables and the points where an interrupt should happen

Challenge

How to Force Interrupts

Hardware Interrupt

Can be modeled by setting a GPIO pin to high voltage, and wiring that high voltage to another GPIO pin.



Challenge
How to Force Interrupts

Software Interrupt

- Can be modeled by embedding the ARM assembly instruction, **SVC**, in the code.
- Modify the interrupt vector table to include our interrupt handler.

```
MyHandler       PROC
                EXPORT  MyHandler
                SVC  11
                ENDP
```

Vector Table in `Startup.s`

```
EXPORT   __Vectors
__Vectors        DCD __initial_sp       ; Top of Stack
                 DCD reset              ; Reset Handler
                  :
                  :
                 DCD MyHandler          ; SVCall Handler
```

Forcing Interrupts: Assumptions

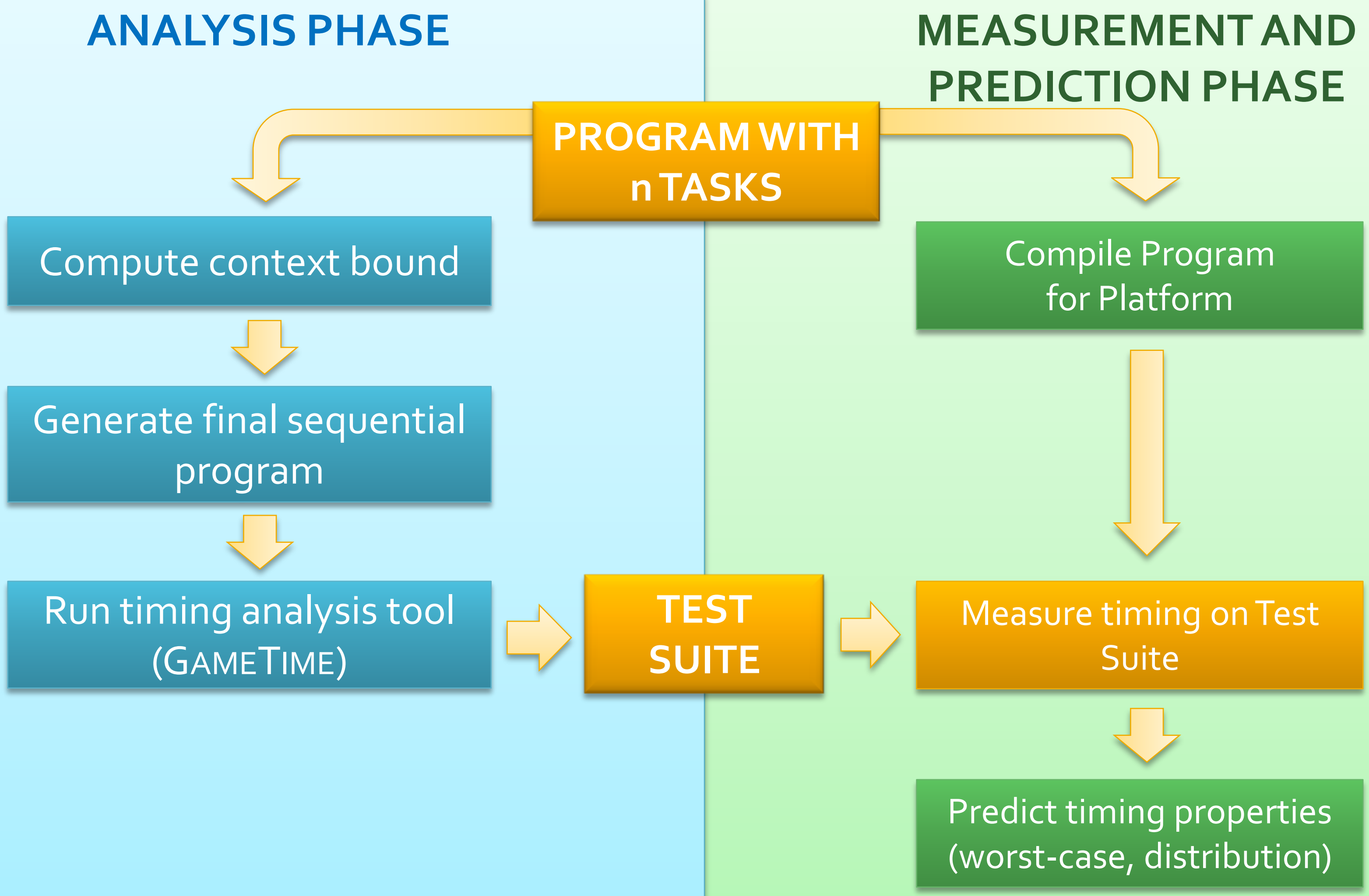
We forced interrupts through software.

- Overhead for the `SVC` call will add to context switch overhead.
- Programs timed with `SysTickTimer`

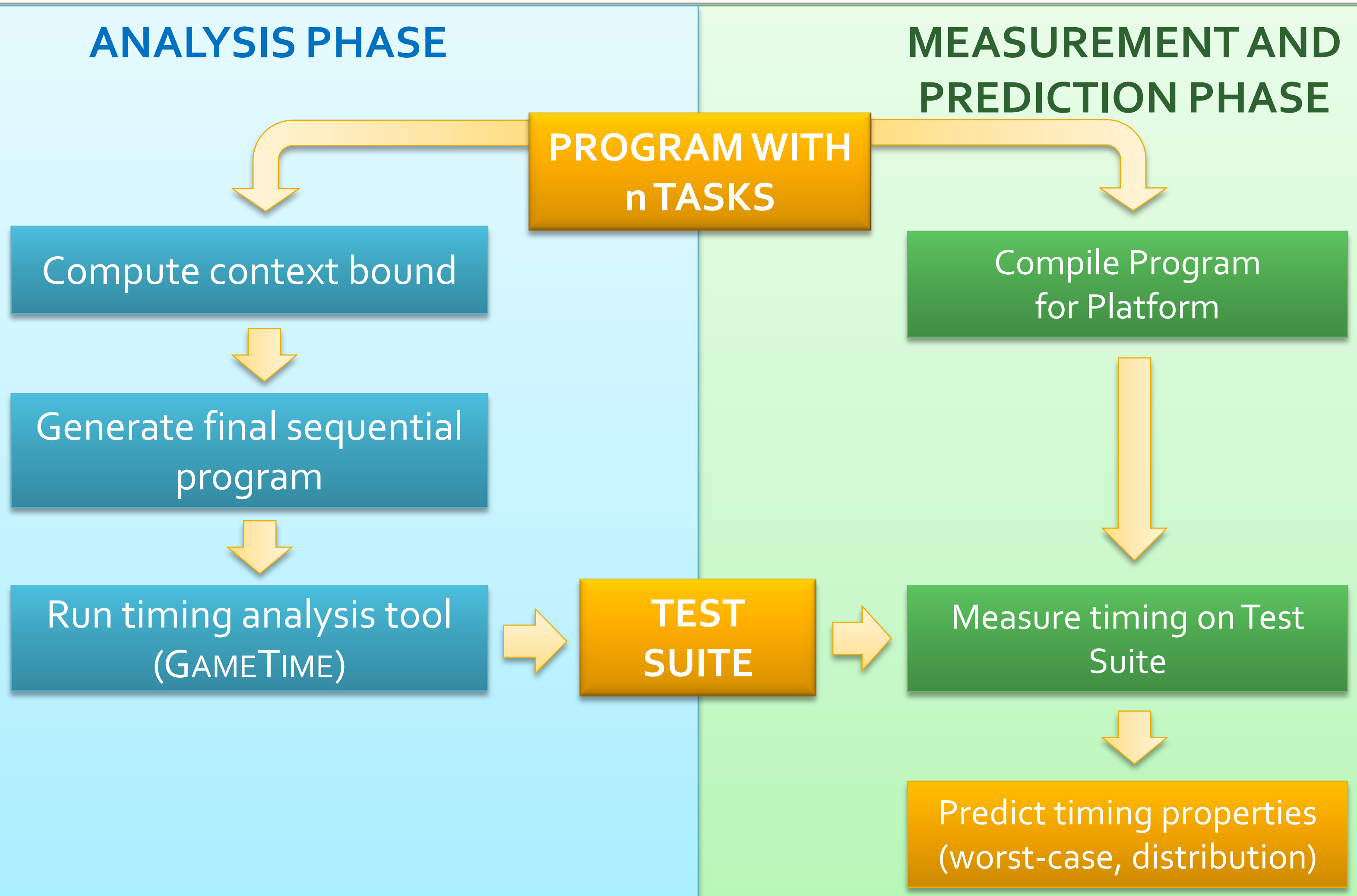
  Timer wraps around after 16,777,261 cycles

Upper bound on program execution time

Approach


ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

Approach


ANALYSIS PHASE
MEASUREMENT AND PREDICTION PHASE
PROGRAM WITH n TASKS
Compute context bound
Generate final sequential program
Run timing analysis tool (GAMETIME)
TEST SUITE
Compile Program for Platform
Measure timing on Test Suite
Predict timing properties (worst-case, distribution)

Timing Prediction

- With measurements, assign weights to edges in control-flow graph of sequential code
- Use weights to predict runtimes for other arbitrary inputs and interleavings

Characteristics of Benchmarks

Name	Lines of Code	Nodes in CFG	Edges in CFG	Total number of paths	Number of basis paths	Context Bound	Interrupt Inter-arrival Time
modexp	60	60	70	500	12	1	1ms
iRobot-1	210	55	60	33	5	1	1ms
iRobot-2	230	141	160	3362	17	1	1ms
iRobot-3	230	97	108	1281	10	2	50μs
iRobot-4	280	213	244	33728	30	1	1ms
iRobot-5	250	179	206	65088	27	1	1ms

iRobot Code with Interrupt Points

Interrupt point 1

```
if(irobot_sensors_valid && irobot_sensors_buttons &&
   ButtonAdvance) { ... }
```

Interrupt point 2

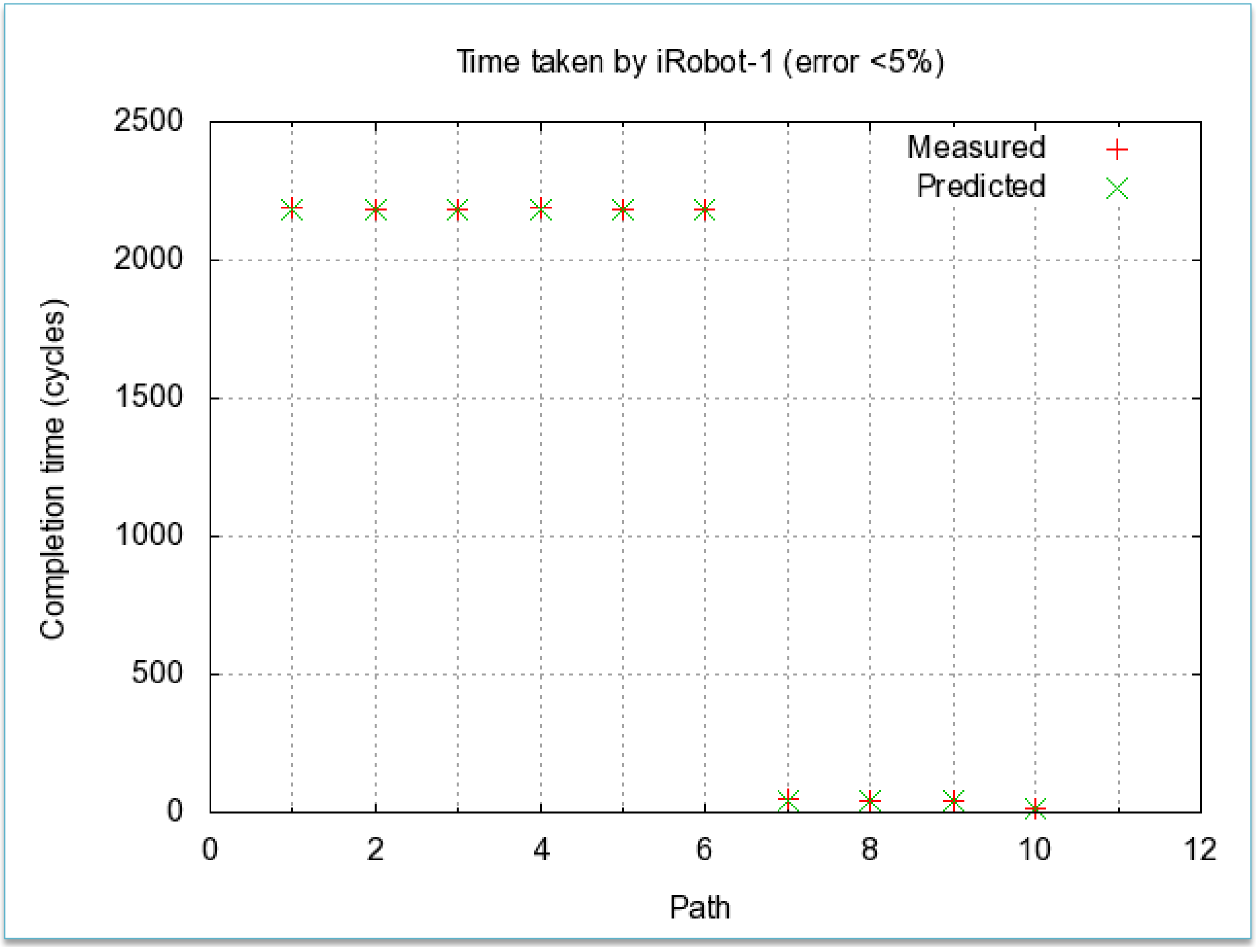
```
// Bumps, cliffs, or wheel drops?
if(irobot_sensors_valid &&
   (irobot_sensors_bumps_drops & BumpEither) ||
   (irobot_sensors_bumps_drops & WheelDropAll) ||
   irobot_sensors_cliffL || ...) { ... }
```

Interrupt point 3

```
// State machine
switch(state){
    case RESET: ...
    case BACKUP: ...
    case DRIVE: ...
}
```

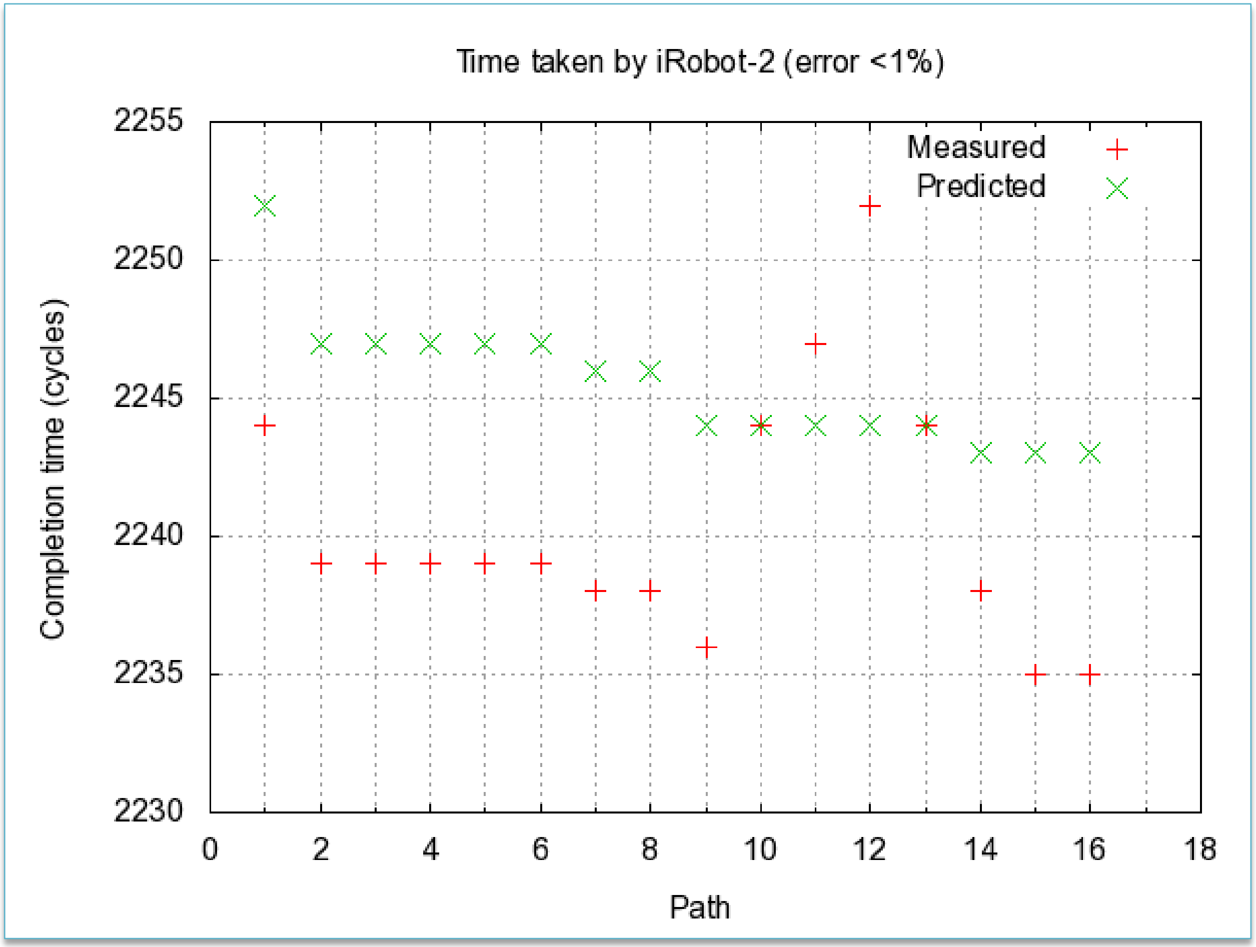
iRobot Code with Interrupt Points

(iRobot-1: Fewest states)



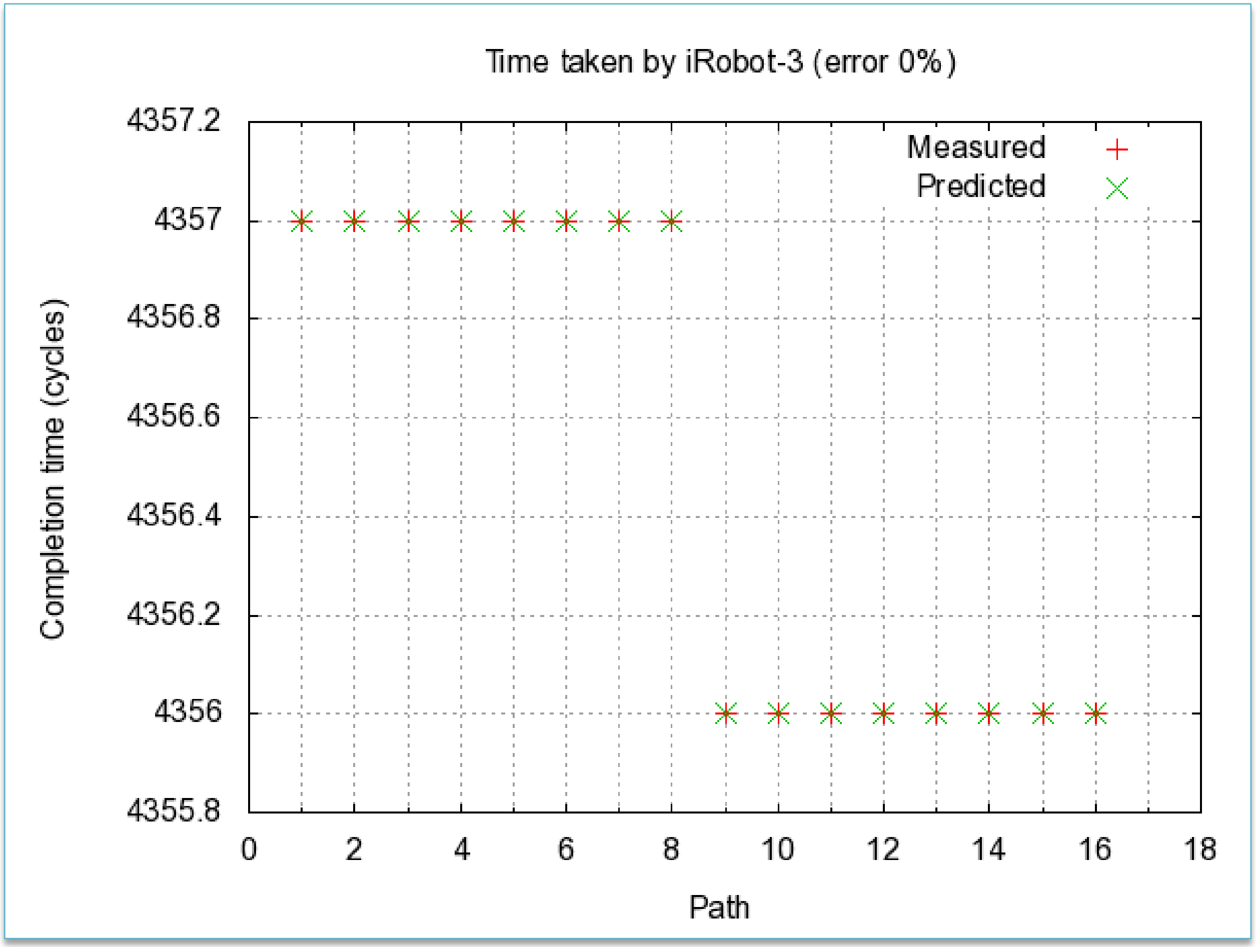
iRobot Code with Interrupt Points

(iRobot-2: One more state)



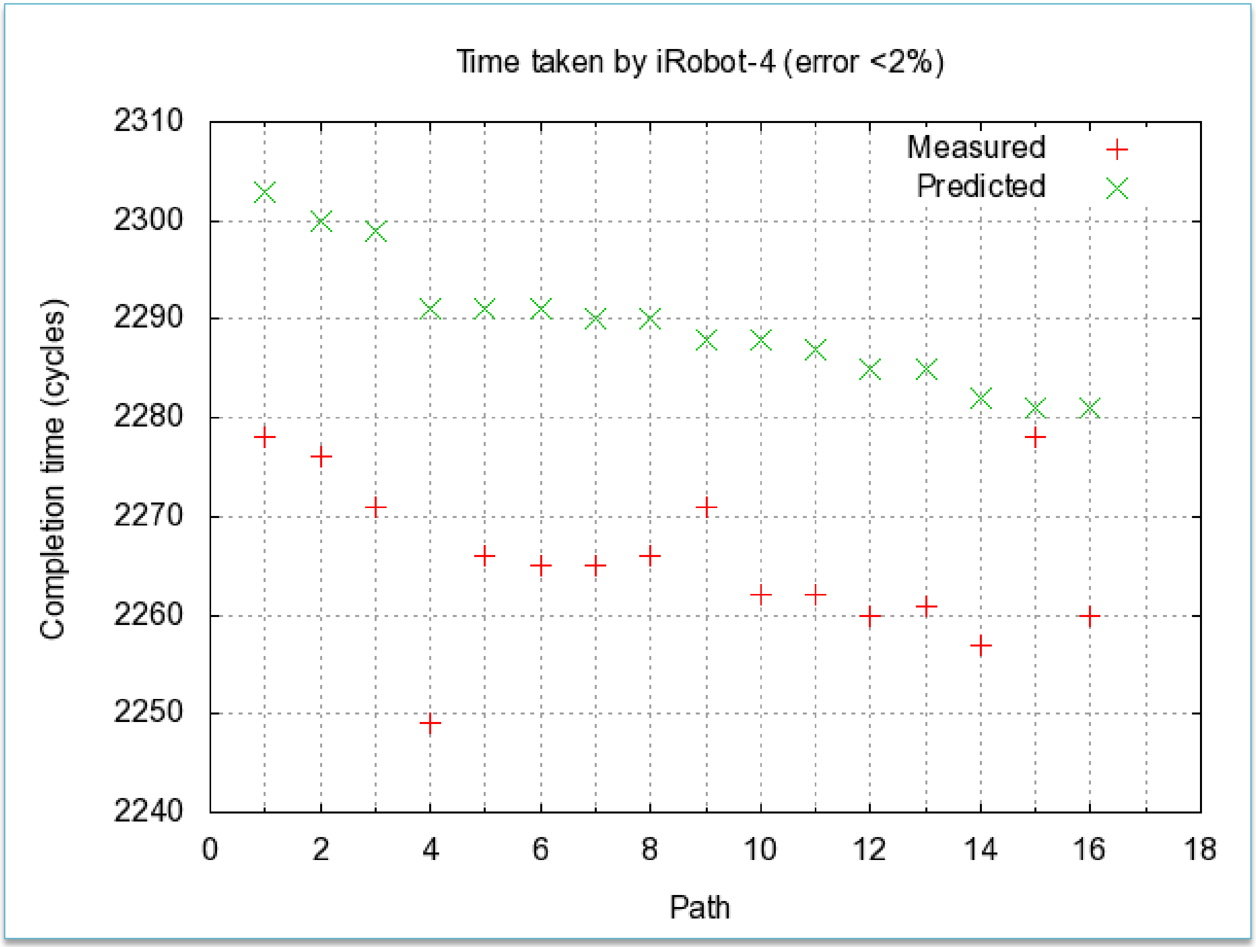
iRobot Code with Interrupt Points

(iRobot-3: Context bound of two)

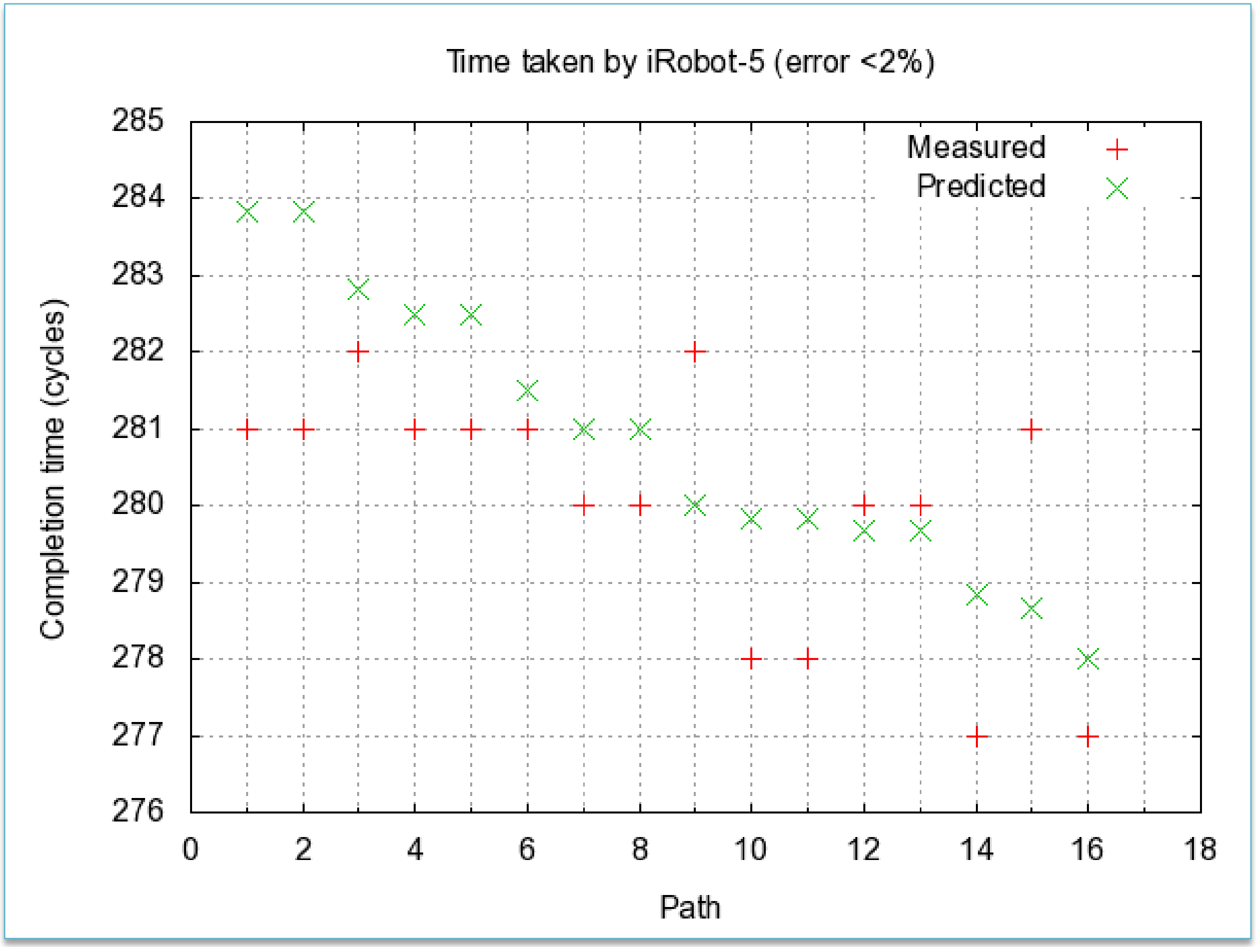


iRobot Code with Interrupt Points

(iRobot-4: More states)



iRobot Code with Interrupt Points
(iRobot-5: States that use the accelerometer)



Summary and Future Work

- Under a certain set of reasonable assumptions, GAMETIME can be used to predict times for interrupt-driven programs.
- *Ongoing/Future work*
  - Extend to other scheduling strategies.
  - Expand evaluation to larger benchmarks with several ISRs.
  - Analysis of energy consumption.

Thank you!