TUHH
Hamburg University of Technology

Bus-centric Optimization and Analysis for Multicore Hard Real-Time Systems

Dominic Oehlert Heiko Falk

Institute of Embedded Systems
Hamburg University of Technology

firstname.surname@tuhh.de

11.06.2018

Outline

Static Instruction SPM Allocation

- The Worst-Case Execution Time (WCET) of a program is crucial in hard real-time systems
- Scratchpad Memories (SPMs) offer great opportunities to reduce the WCET
- ILP-based optimizations can be used

$\Rightarrow$ SPM allocation involves many challenges when actually applied

  - Instructions added, referenced blocks, asymmetric jump costs, ...
  - Increased complexity for multicore systems

Static Instruction SPM Allocation

- The Worst-Case Execution Time (WCET) of a program is crucial in hard real-time systems
- Scratchpad Memories (SPMs) offer great opportunities to reduce the WCET
- ILP-based optimizations can be used

$\Rightarrow$ SPM allocation involves many challenges when actually applied

- Instructions added, referenced blocks, asymmetric jump costs, ...
- Increased complexity for multicore systems

Static Instruction SPM Allocation

- The Worst-Case Execution Time (WCET) of a program is crucial in hard real-time systems
- Scratchpad Memories (SPMs) offer great opportunities to reduce the WCET
- ILP-based optimizations can be used

$\Rightarrow$ SPM allocation involves many challenges when actually applied

- Instructions added, referenced blocks, asymmetric jump costs, ...
- Increased complexity for multicore systems

Static Instruction SPM Allocation

- The Worst-Case Execution Time (WCET) of a program is crucial in hard real-time systems
- Scratchpad Memories (SPMs) offer great opportunities to reduce the WCET
- ILP-based optimizations can be used

$\Rightarrow$ SPM allocation involves many challenges when actually applied

- Instructions added, referenced blocks, asymmetric jump costs, ...
- Increased complexity for multicore systems

Static Instruction SPM Allocation

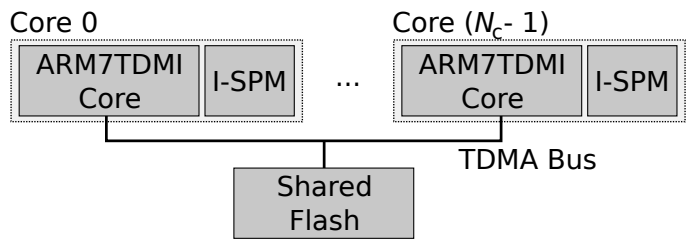
- The Worst-Case Execution Time (WCET) of a program is crucial in hard real-time systems
- Scratchpad Memories (SPMs) offer great opportunities to reduce the WCET
- ILP-based optimizations can be used

$\Rightarrow$ SPM allocation involves many challenges when actually applied

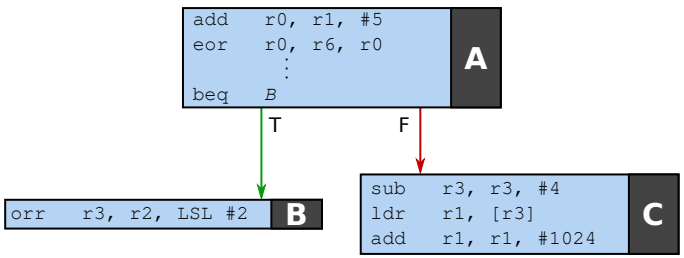
  - Instructions added, referenced blocks, asymmetric jump costs, ...
  - Increased complexity for multicore systems

Multicore Architecture

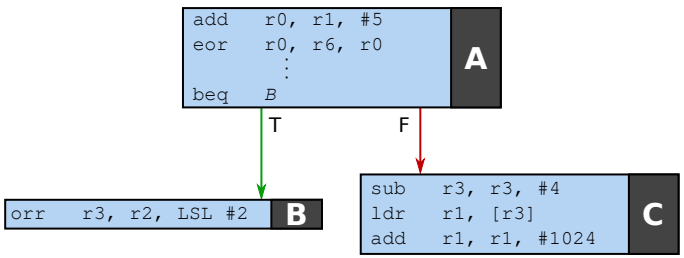
- N_c homogeneous parallel cores
- Private instruction scratchpad memories
- Shared Flash memory
- TDMA scheduled bus



Applying Singlecore SPM Allocation

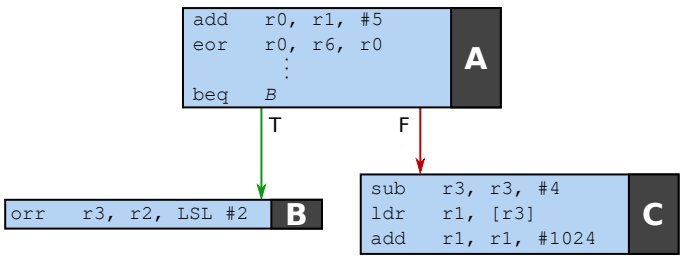


Applying Singlecore SPM Allocation



- 4 cores
- Private SPM size: 20 B

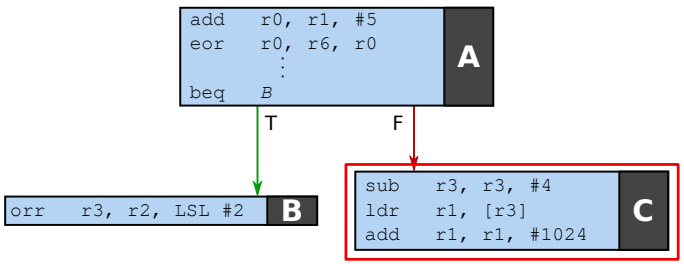
Applying Singlecore SPM Allocation



Basic Block	WCET (Cycles) Flash	WCET (Cycles) SPM	Size (B)
A	390	20	80
B	96	1	4
C	114	9	12

- 4 cores
- Private SPM size: 20 B

Applying Singlecore SPM Allocation

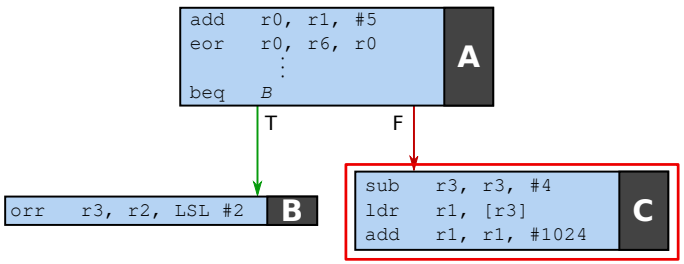


Basic Block	WCET (Cycles) Flash	WCET (Cycles) SPM	Size (B)
A	**411**	20	80
B	96	1	4
C	114	**9**	12

- 4 cores
- Private SPM size: 20 B

$\Rightarrow$ Expected WCET Reduction: 84 Cycles ($\downarrow$ 16%)

Applying Singlecore SPM Allocation

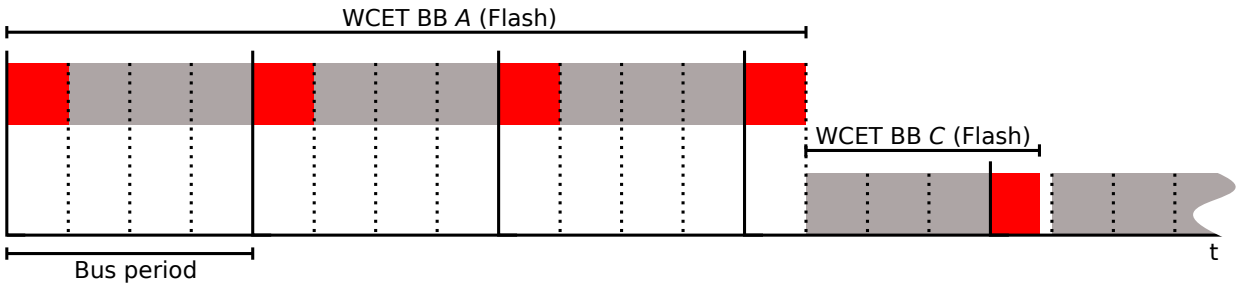


Basic Block	WCET (Cycles) Flash	WCET (Cycles) SPM	Size (B)
A	**411**	20	80
B	96	1	4
C	114	**9**	12

- 4 cores
- Private SPM size: 20 B

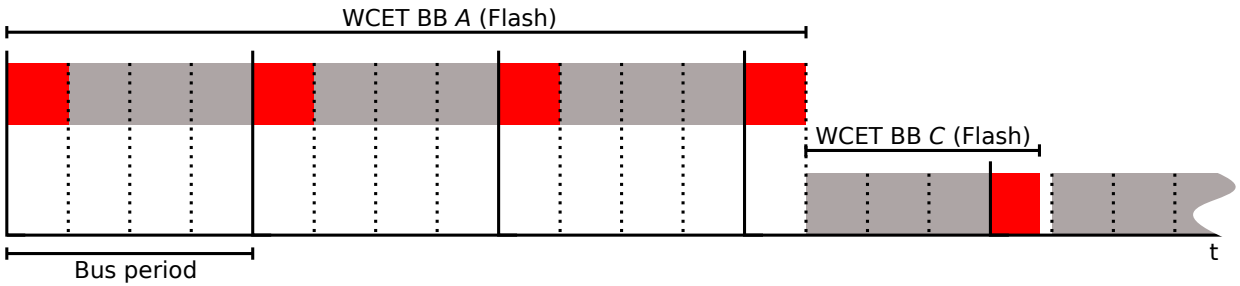
$\Rightarrow$ Expected WCET Reduction: 84 Cycles ($\downarrow$ 16%)
$\Rightarrow$ **Actual WCET Reduction: -7 Cycles ($\uparrow$ 1%)**

Cause of the False Estimation

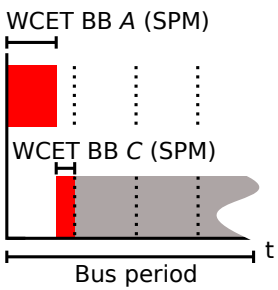


All in Flash

Cause of the False Estimation

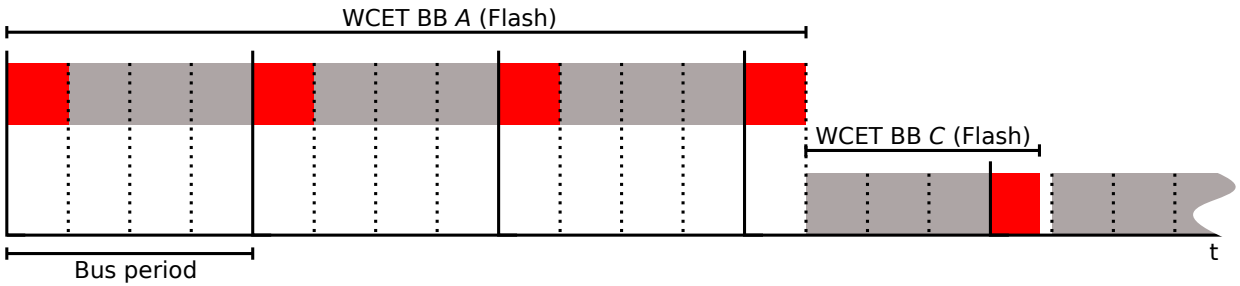


All in Flash

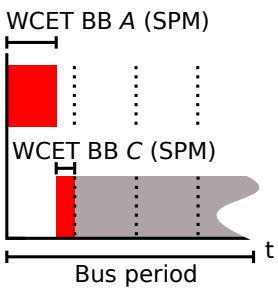


All in SPM

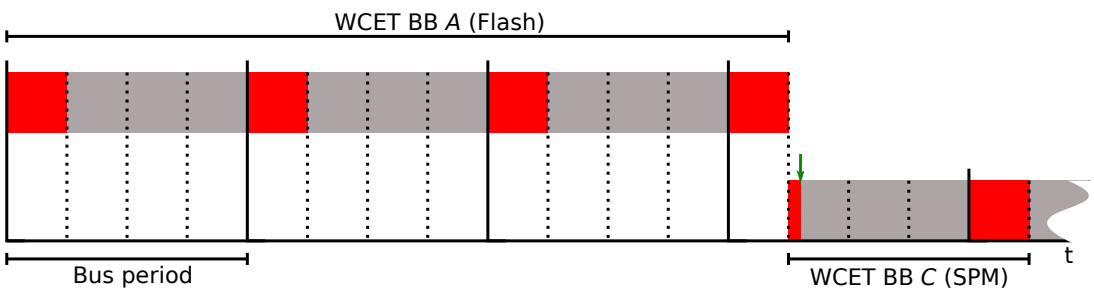
Cause of the False Estimation



All in Flash

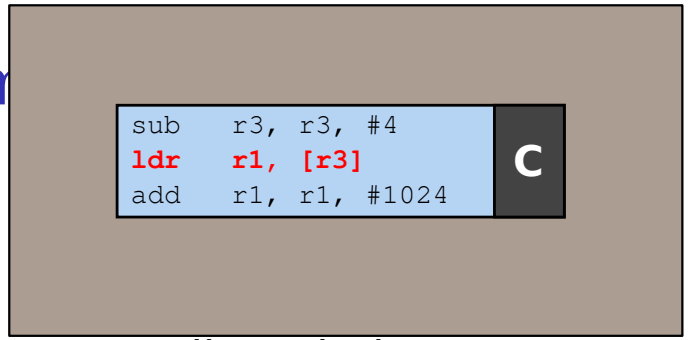


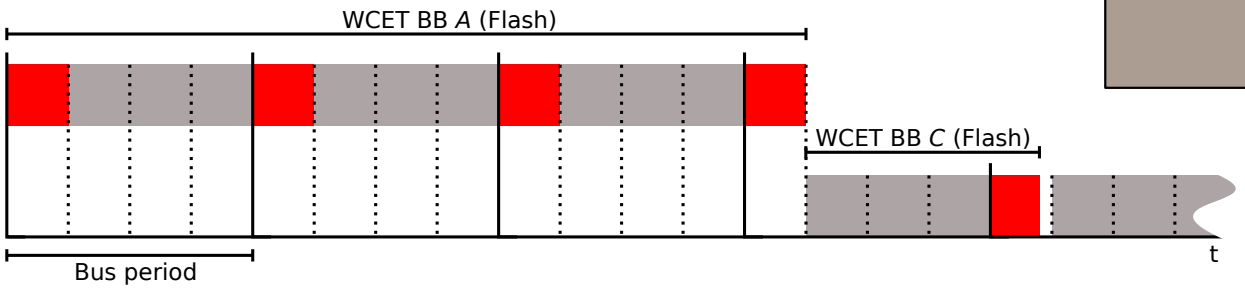
All in SPM



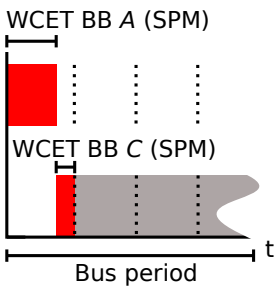
C allocated to SPM

Cause of the False Estim

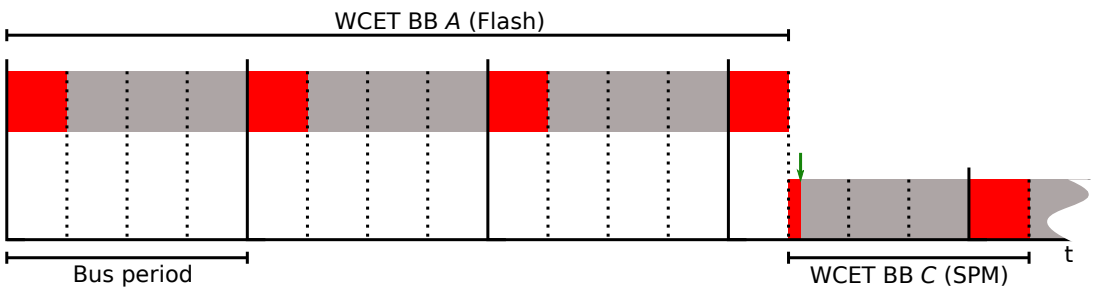




All in Flash



All in SPM



C allocated to SPM

Preliminaries (Slot Length)

- WCETs of basic blocks located in the shared Flash memory depend on the *ingoing* bus offset

- Analyzed WCETs are not ensured to be safe anymore

$\Rightarrow$ Assume all TDMA slots equally-sized and fixed to the length of a Flash access

  - Analyzed WCETs are safe again (per BB in shared memory)

Preliminaries (Slot Length)

- WCETs of basic blocks located in the shared Flash memory depend on the *ingoing* bus offset
- Analyzed WCETs are not ensured to be safe anymore

$\Rightarrow$ Assume all TDMA slots equally-sized and fixed to the length of a Flash access

- Analyzed WCETs are safe again (per BB in shared memory)

Preliminaries (Slot Length)

- WCETs of basic blocks located in the shared Flash memory depend on the *ingoing* bus offset
- Analyzed WCETs are not ensured to be safe anymore

$\Rightarrow$ Assume all TDMA slots equally-sized and fixed to the length of a Flash access

- Analyzed WCETs are safe again (per BB in shared memory)

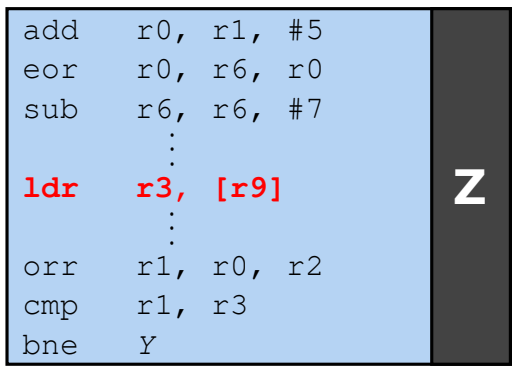
Preliminaries (Slot Length)

- WCETs of basic blocks located in the shared Flash memory depend on the *ingoing* bus offset
- Analyzed WCETs are not ensured to be safe anymore

$\Rightarrow$ Assume all TDMA slots equally-sized and fixed to the length of a Flash access
  - Analyzed WCETs are safe again (per BB in shared memory)

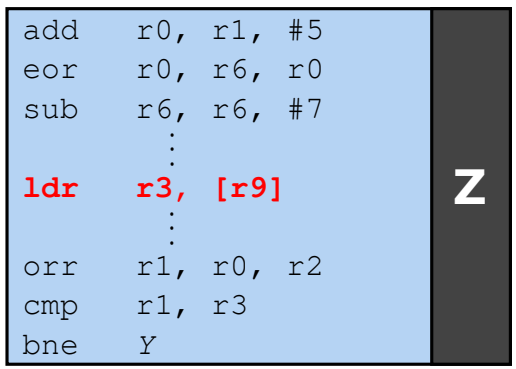
Preliminaries (Sub Basic Blocks)

- Instructions located in the private SPM may access the shared memory



Preliminaries (Sub Basic Blocks)

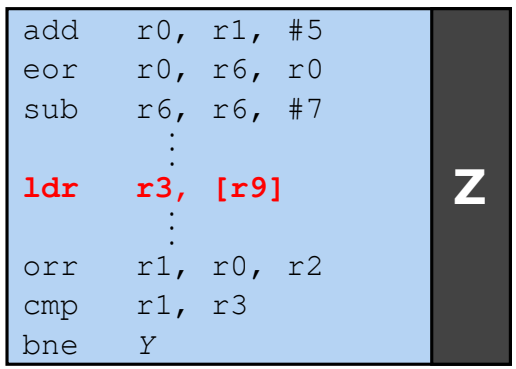
- Instructions located in the private SPM may access the shared memory
- A basic block's WCET may depend on the bus schedule, despite residing in the private SPM



Preliminaries (Sub Basic Blocks)

- Instructions located in the private SPM may access the shared memory
- A basic block's WCET may depend on the bus schedule, despite residing in the private SPM

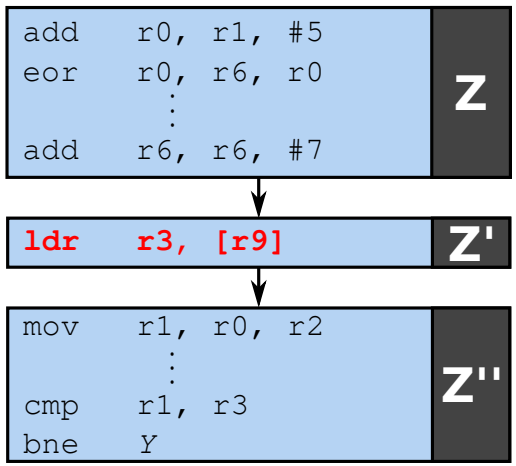
$\Rightarrow$ Assume sub basic blocks, containing at maximum *one* such potential instruction



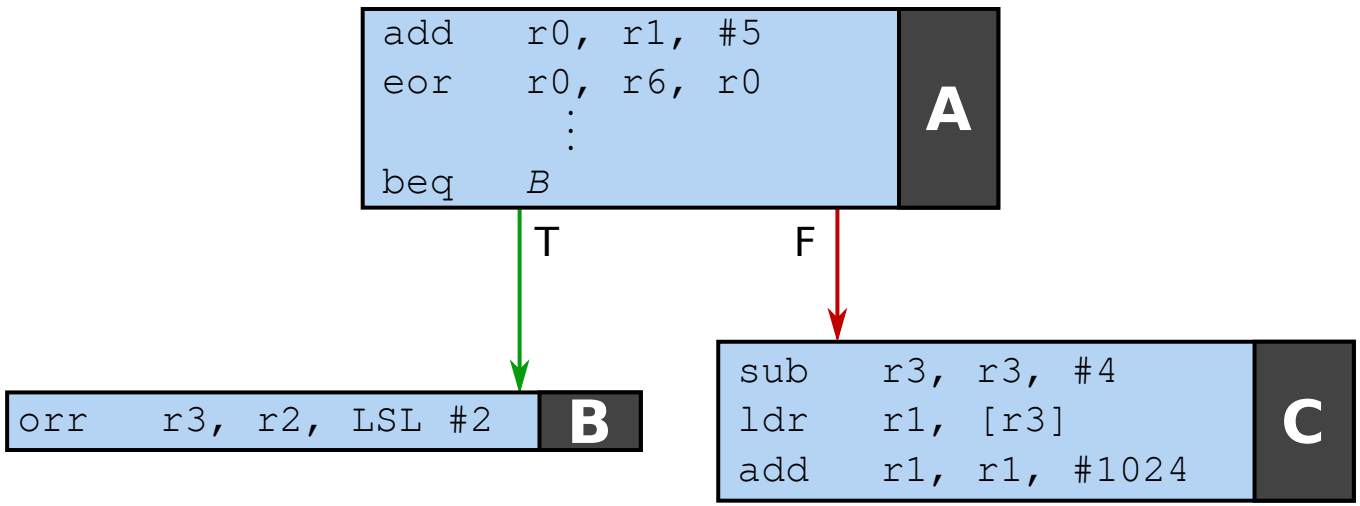
Preliminaries (Sub Basic Blocks)

- Instructions located in the private SPM may access the shared memory
- A basic block's WCET may depend on the bus schedule, despite residing in the private SPM

$\Rightarrow$ Assume sub basic blocks, containing at maximum *one* such potential instruction

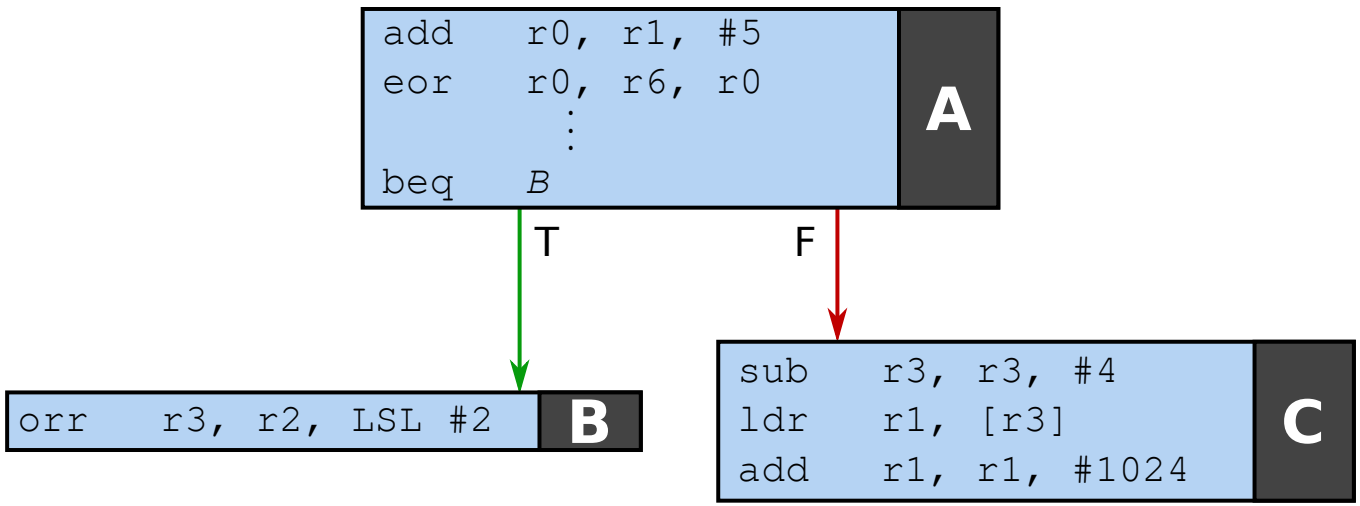


Base ILP Model



Base ILP Model

$C_{A,Flash} = 390$ Cycles



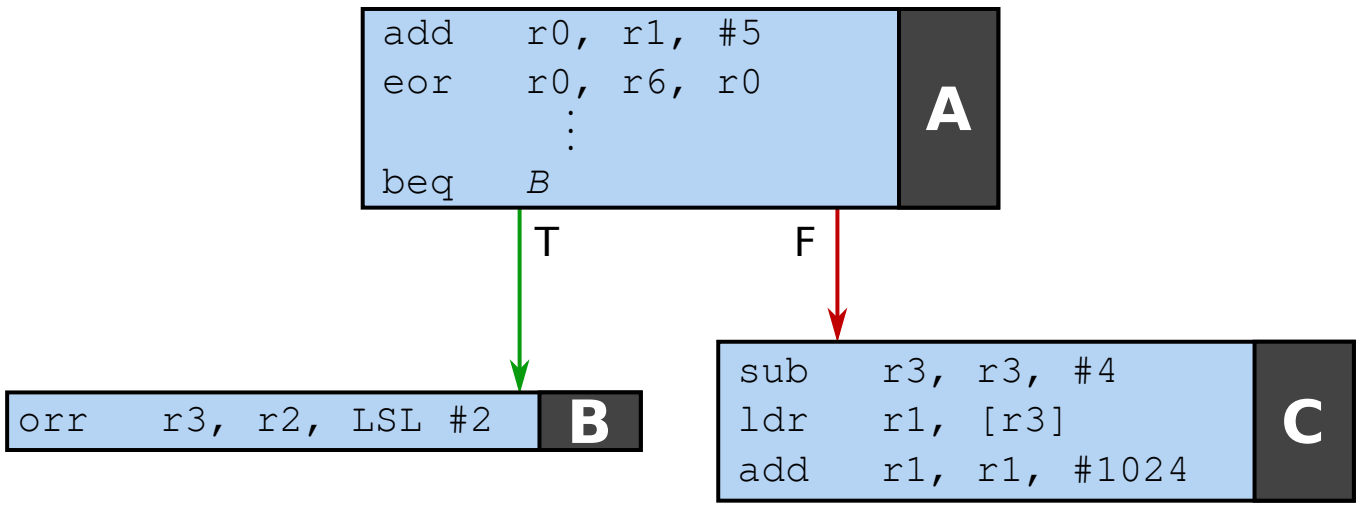
$C_{B,Flash} = 96$ Cycles

$C_{C,Flash} = 114$ Cycles

Base ILP Model

$C_{A,Flash} = 390$ Cycles

$C_{A,SPM} = 20$ Cycles



$C_{B,Flash} = 96$ Cycles

$C_{B,SPM} = 1$ Cycle

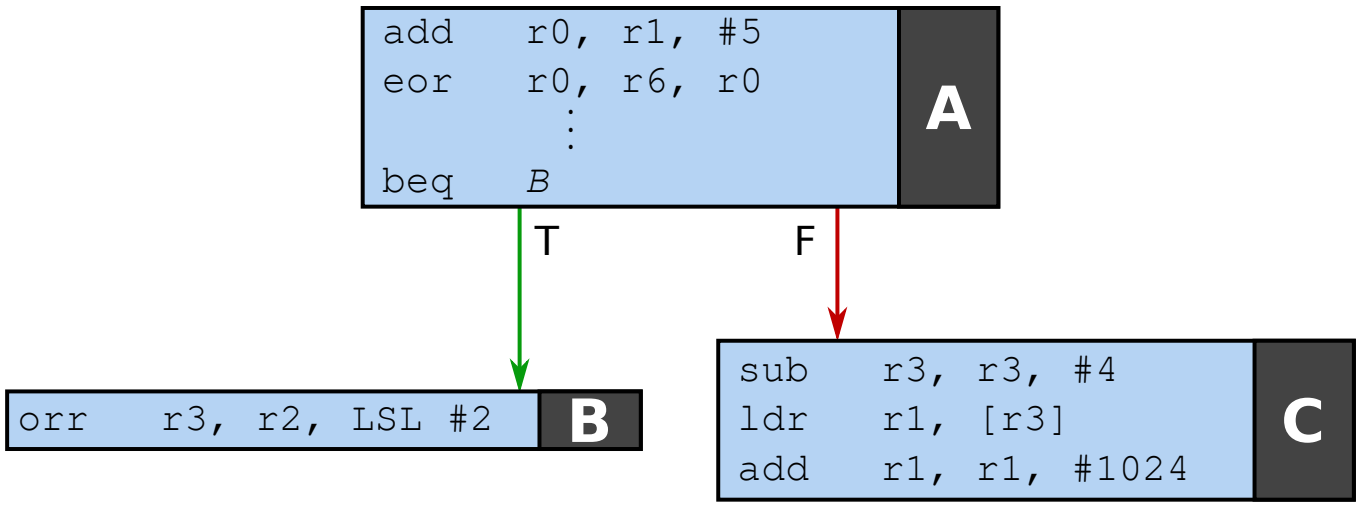
$C_{C,Flash} = 114$ Cycles

$C_{C,SPM} = 9$ Cycles

Base ILP Model

$C_{A,Flash} = 390$ Cycles

$C_{A,SPM} = 20$ Cycles



$C_{B,Flash} = 96$ Cycles

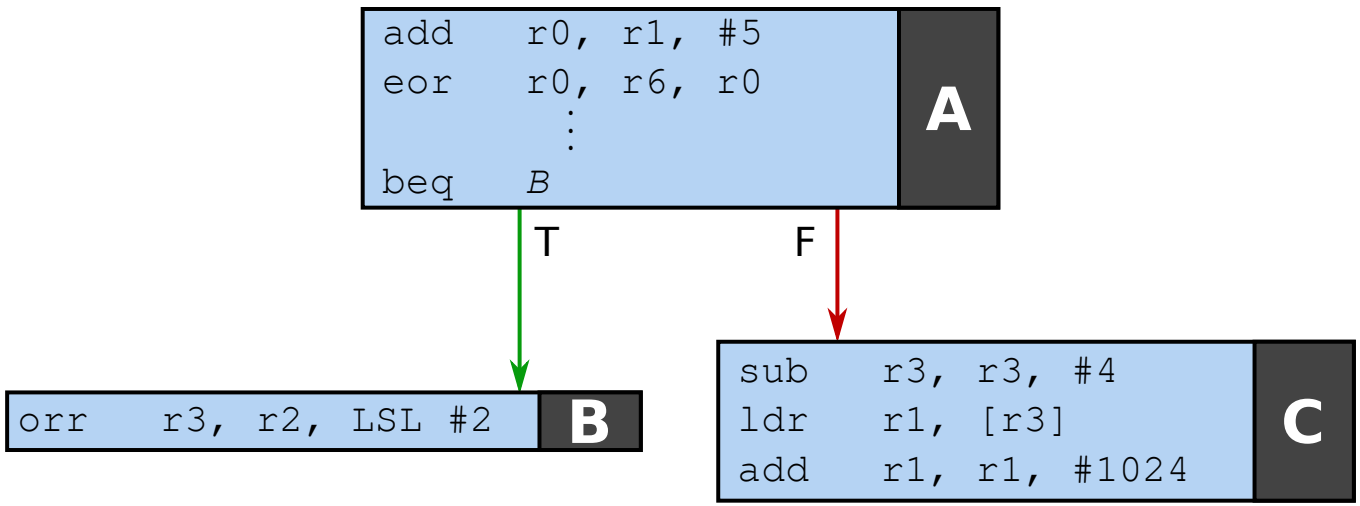
$C_{B,SPM} = 1$ Cycle

$C_{C,Flash} = 114$ Cycles

$C_{C,SPM} = 9$ Cycles

$S_{SPM} = 20$ B

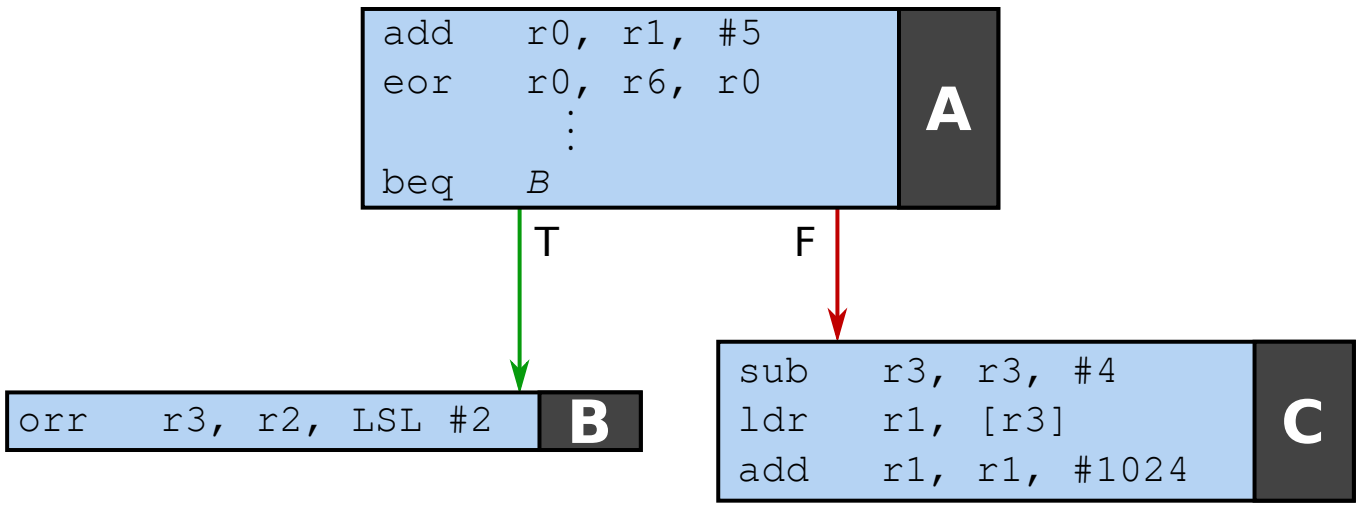
Base ILP Model



$w_B = C_{B,Flash}$ $w_C = C_{C,Flash}$

Base ILP Model

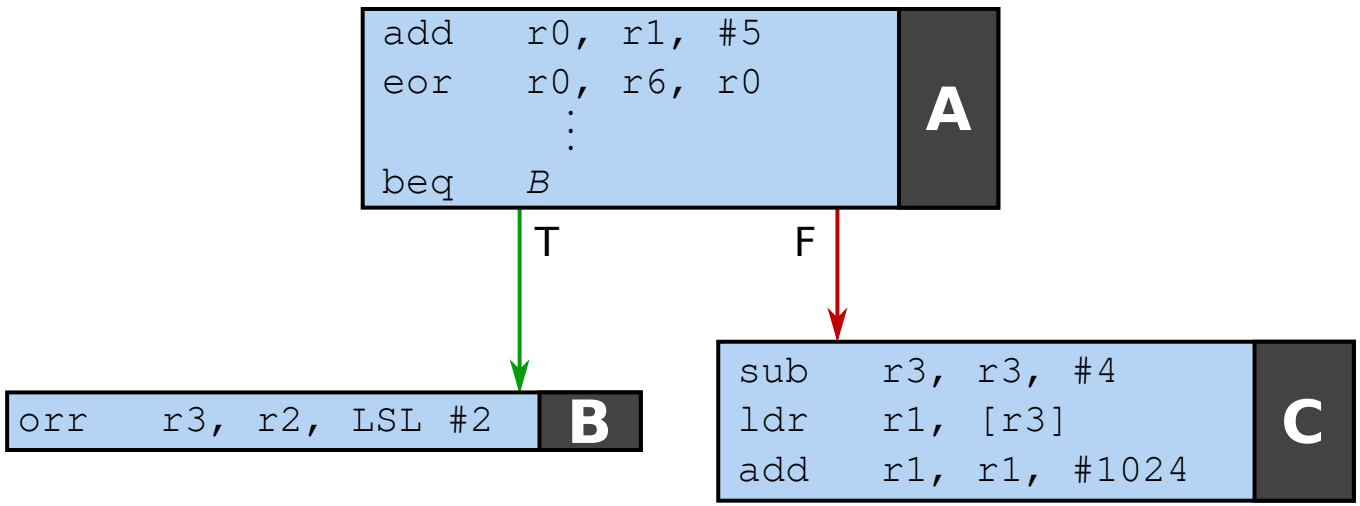
$$w_A \geq C_{A,Flash} + w_B$$
$$w_A \geq C_{A,Flash} + w_C$$



$$w_B = C_{B,Flash} \qquad w_C = C_{C,Flash}$$

Base ILP Model

$$w_A \geq C_{A,Flash} + w_B$$
$$w_A \geq C_{A,Flash} + w_C$$

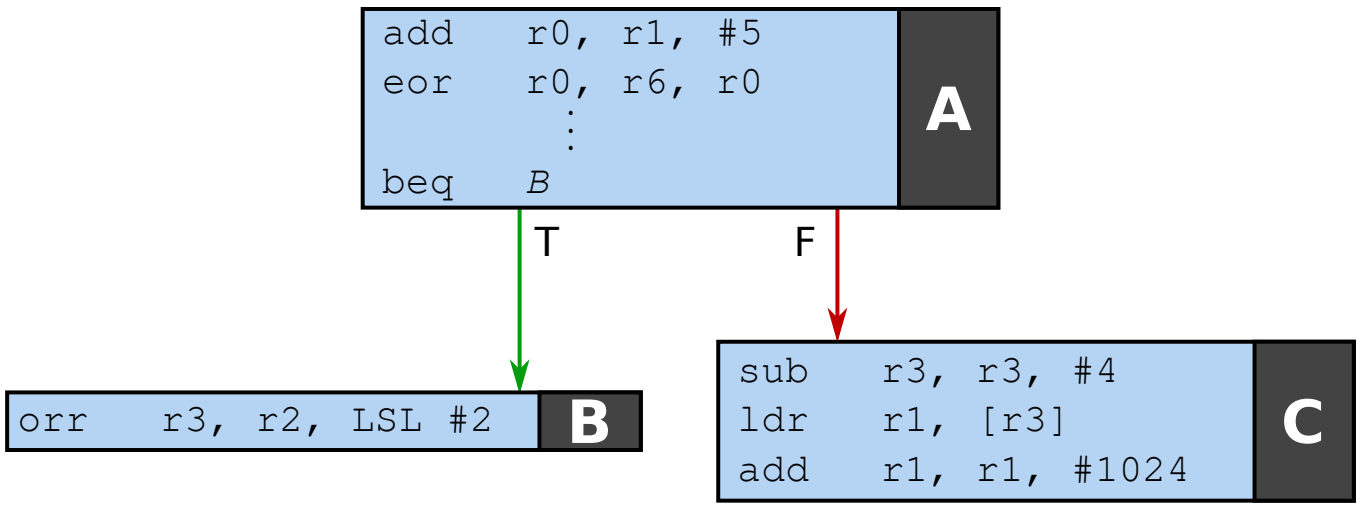


$$w_B = C_{B,Flash} \qquad w_C = C_{C,Flash}$$

$$G_Y = C_{Y,Flash} - C_{Y,SPM}$$

Base ILP Model

$$w_A \geq C_{A,Flash} - x_A \cdot G_A + w_B$$
$$w_A \geq C_{A,Flash} - x_A \cdot G_A + w_C$$

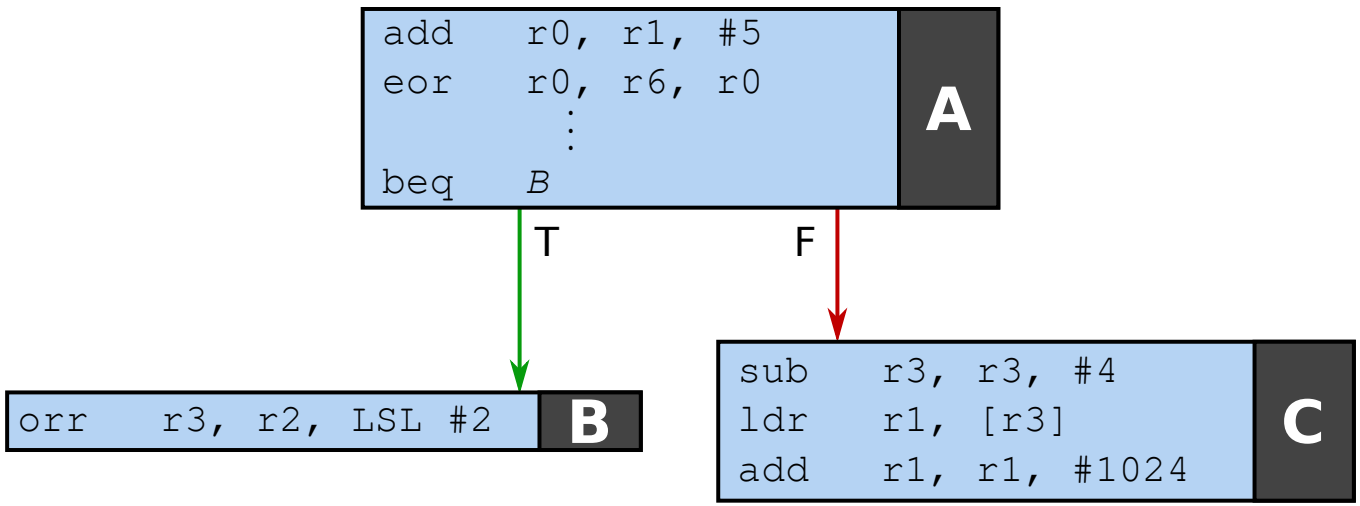


$$w_B = C_{B,Flash} - x_B \cdot G_B \qquad w_C = C_{C,Flash} - x_C \cdot G_C$$

$$G_Y = C_{Y,Flash} - C_{Y,SPM}$$

Base ILP Model

$$w_A \geq C_{A,Flash} - x_A \cdot G_A + w_B$$
$$w_A \geq C_{A,Flash} - x_A \cdot G_A + w_C$$

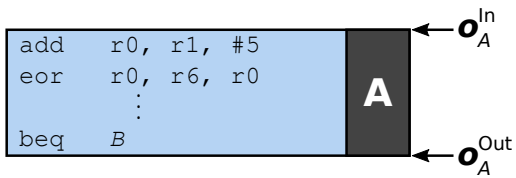


$$w_B = C_{B,Flash} - x_B \cdot G_B \qquad w_C = C_{C,Flash} - x_C \cdot G_C$$

$$S_{SPM} \geq x_A \cdot S_A + x_B \cdot S_B + x_C \cdot S_C$$

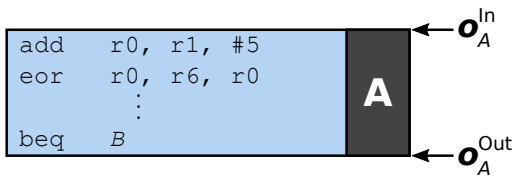
Bus Offset Calculation

- Ingoing o_ν^{In} and outgoing bus offsets o_ν^{Out} are calculated per BB ν



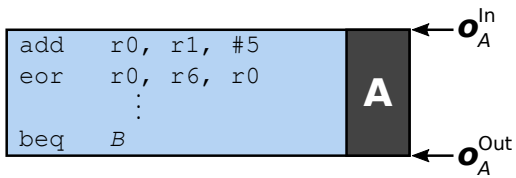
Bus Offset Calculation

- Ingoing $\boldsymbol{o}_{\nu}^{\mathsf{In}}$ and outgoing bus offsets $\boldsymbol{o}_{\nu}^{\mathsf{Out}}$ are calculated per BB ν
- $\boldsymbol{o}_{\nu} = (o_{\mathsf{low}}, o_{\mathsf{high}})$



Bus Offset Calculation

- Ingoing $\boldsymbol{o}_{\nu}^{\mathsf{In}}$ and outgoing bus offsets $\boldsymbol{o}_{\nu}^{\mathsf{Out}}$ are calculated per BB ν
- $\boldsymbol{o}_{\nu} = (o_{\mathsf{low}}, o_{\mathsf{high}})$
- $o \in [0, \text{Bus Period} - 1]$



Bus-related Timings

- The WCET of a BB in the ILP model can be adjusted based on ...
  - bus offsets determined inside the ILP model
  - bus offsets from the WCET analysis
- Timing gain/penalty of a data access is expressed via d_ν
- Bus-related timing of a jump correction is expressed via $l_{\nu,\mu}$

Bus-related Timings

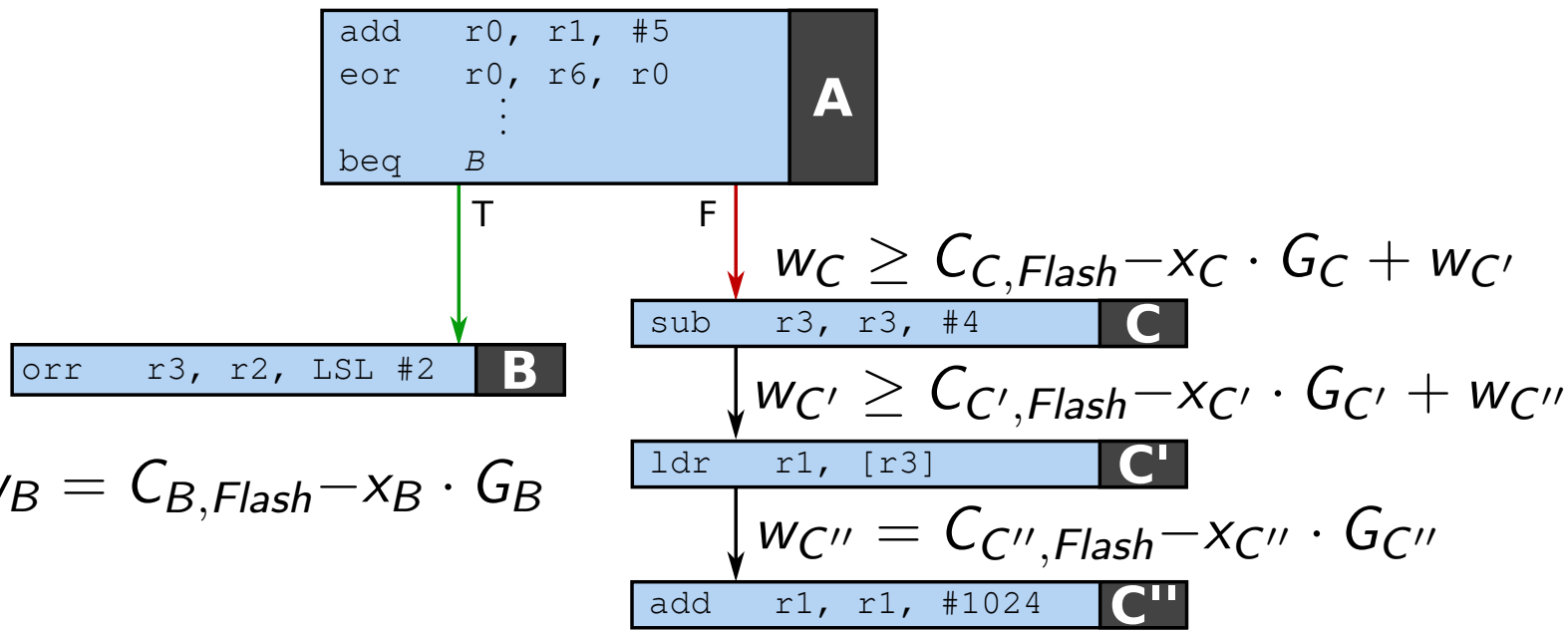
- The WCET of a BB in the ILP model can be adjusted based on ...
  - bus offsets determined inside the ILP model
  - bus offsets from the WCET analysis
- Timing gain/penalty of a data access is expressed via d_ν
- Bus-related timing of a jump correction is expressed via $l_{\nu,\mu}$

Bus-related Timings

- The WCET of a BB in the ILP model can be adjusted based on ...
  - bus offsets determined inside the ILP model
  - bus offsets from the WCET analysis
- Timing gain/penalty of a data access is expressed via d_ν
- Bus-related timing of a jump correction is expressed via $l_{\nu,\mu}$

Final ILP Model

$$w_A \geq C_{A,Flash} - x_A \cdot G_A + w_B$$
$$w_A \geq C_{A,Flash} - x_A \cdot G_A + w_C$$

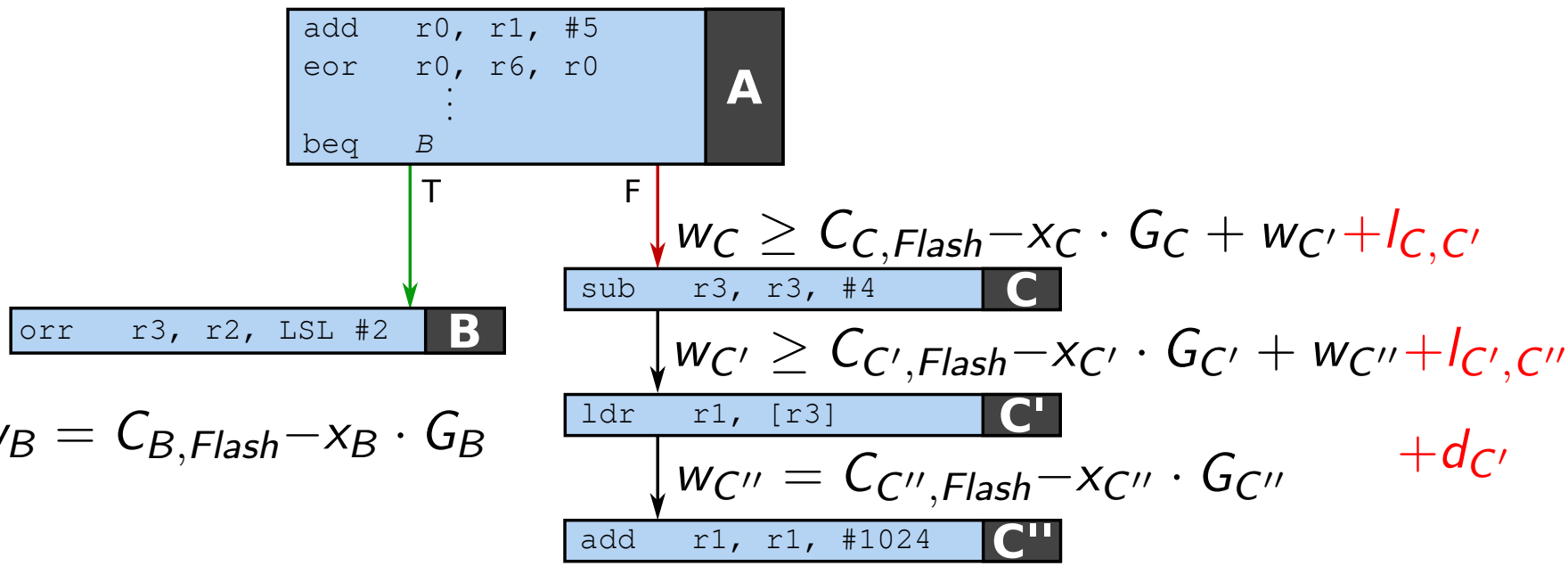


$$w_C \geq C_{C,Flash} - x_C \cdot G_C + w_{C'}$$
$$w_{C'} \geq C_{C',Flash} - x_{C'} \cdot G_{C'} + w_{C''}$$
$$w_{C''} = C_{C'',Flash} - x_{C''} \cdot G_{C''}$$
$$w_B = C_{B,Flash} - x_B \cdot G_B$$

$$S_{SPM} \geq x_A \cdot S_A + x_B \cdot S_B + x_C \cdot S_C$$

Final ILP Model

$$w_A \geq C_{A,Flash} - x_A \cdot G_A + w_B + I_{A,B}$$
$$w_A \geq C_{A,Flash} - x_A \cdot G_A + w_C + I_{A,C}$$



$$w_C \geq C_{C,Flash} - x_C \cdot G_C + w_{C'} + I_{C,C'}$$
$$w_{C'} \geq C_{C',Flash} - x_{C'} \cdot G_{C'} + w_{C''} + I_{C',C''}$$
$$w_{C''} = C_{C'',Flash} - x_{C''} \cdot G_{C''} + d_{C'}$$
$$w_B = C_{B,Flash} - x_B \cdot G_B$$

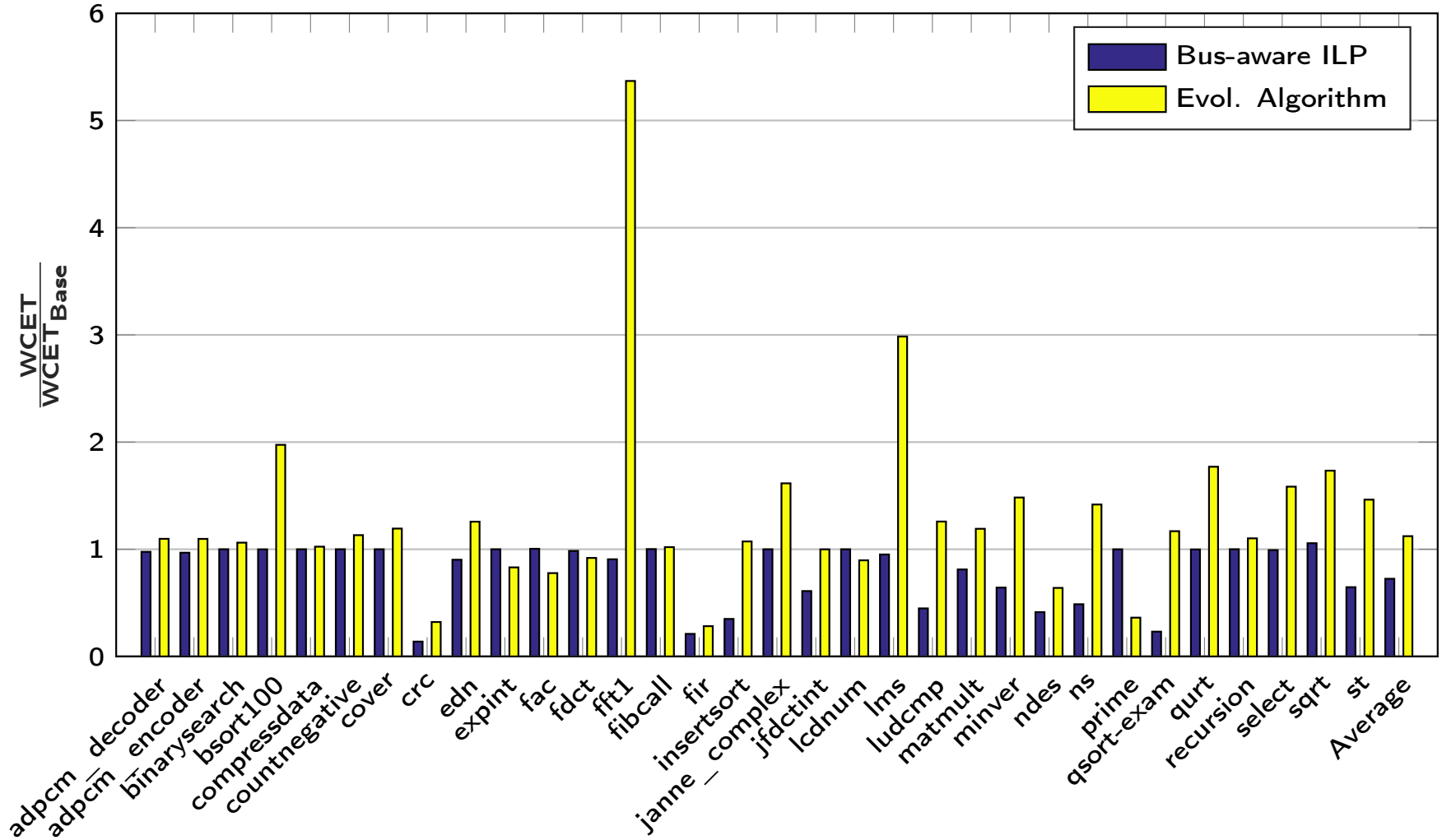
$$S_{SPM} \geq x_A \cdot S_A + x_B \cdot S_B + x_C \cdot S_C$$

Overview

Evaluation Setup

- MRTC benchmark suite
  - `duff` benchmark excluded (irregular loops are currently not supported)
  - `petrinet` & `statemate` excluded due to timeout
- Bus-unaware ILP-based instruction SPM allocation optimization as a baseline
- Evolutionary algorithm used as a reference for the ILP-based optimizations
- Evaluations performed on an Intel Xeon Server
- ILPs solved using Gurobi 7.0.1
- Compiled with the WCET-aware C compiler (WCC) using the -02 flag
- SPM size was set to 50% of a benchmark's size

ARM7TDMI Quadcore System



Overview

Conclusion

- Instruction SPM allocation in a multicore environment needs special care
- Extended ILP model (under given assumptions) is able to:
  - Calculate TDMA bus offsets are within the ILP
  - Calculate bus timings based on the offsets
- 26% WCET reduction in average (in comparison to ILP-based bus-unaware optimization)
- Evolutionary algorithms are also applicable, but need more time in average

Conclusion

- Instruction SPM allocation in a multicore environment needs special care
- Extended ILP model (under given assumptions) is able to:
  - Calculate TDMA bus offsets are within the ILP
  - Calculate bus timings based on the offsets
- 26% WCET reduction in average (in comparison to ILP-based bus-unaware optimization)
- Evolutionary algorithms are also applicable, but need more time in average

Conclusion

- Instruction SPM allocation in a multicore environment needs special care
- Extended ILP model (under given assumptions) is able to:
  - Calculate TDMA bus offsets are within the ILP
  - Calculate bus timings based on the offsets
- 26% WCET reduction in average (in comparison to ILP-based bus-unaware optimization)
- Evolutionary algorithms are also applicable, but need more time in average

Conclusion

- Instruction SPM allocation in a multicore environment needs special care
- Extended ILP model (under given assumptions) is able to:
  - Calculate TDMA bus offsets are within the ILP
  - Calculate bus timings based on the offsets
- 26% WCET reduction in average (in comparison to ILP-based bus-unaware optimization)
- Evolutionary algorithms are also applicable, but need more time in average

Overview

Event Arrival Functions

- Holistic WCET-estimation approaches for multicore architectures are very complex do not scale well

- System level analyses typically compute the worst-case timing using
  - WCET in isolation
  - Abstract notion of interfering events (e.g., task activations, shared memory accesses, ...)

$\Rightarrow$ Event arrival functions

Event Arrival Functions

- Holistic WCET-estimation approaches for multicore architectures are very complex do not scale well
- System level analyses typically compute the worst-case timing using
  - WCET in isolation
  - Abstract notion of interfering events (e.g., task activations, shared memory accesses, ...)

$\Rightarrow$ Event arrival functions

Event Arrival Functions

- Holistic WCET-estimation approaches for multicore architectures are very complex do not scale well
- System level analyses typically compute the worst-case timing using
  - WCET in isolation
  - Abstract notion of interfering events (e.g., task activations, shared memory accesses, ...)

$\Rightarrow$ Event arrival functions

Event Arrival Functions

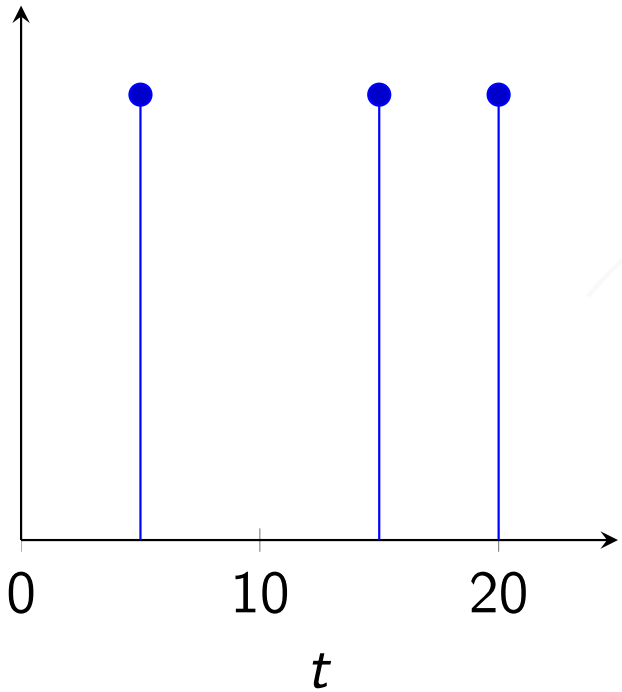
- Holistic WCET-estimation approaches for multicore architectures are very complex do not scale well
- System level analyses typically compute the worst-case timing using
  - WCET in isolation
  - Abstract notion of interfering events (e.g., task activations, shared memory accesses, ...)

$\Rightarrow$ Event arrival functions

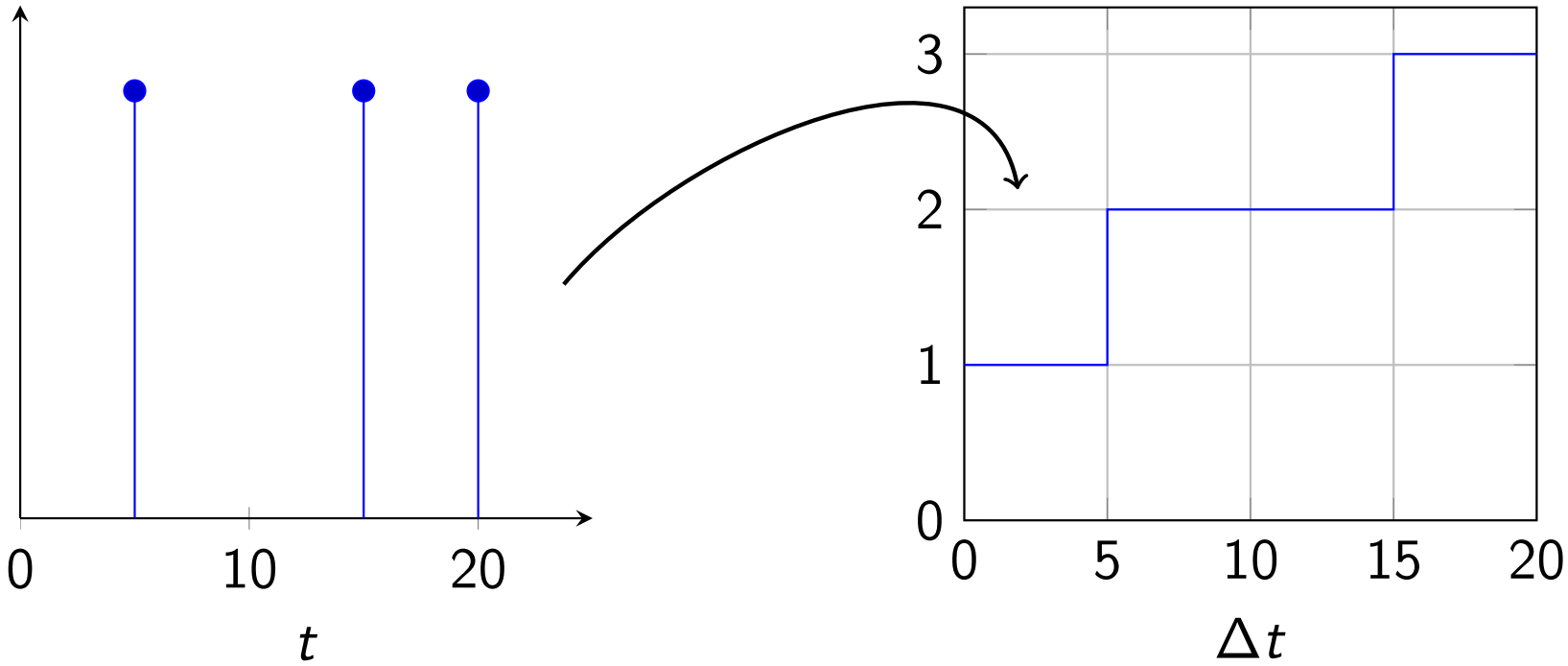
Event Arrival Functions

- Holistic WCET-estimation approaches for multicore architectures are very complex do not scale well
- System level analyses typically compute the worst-case timing using
  - WCET in isolation
  - Abstract notion of interfering events (e.g., task activations, shared memory accesses, ...)

$\Rightarrow$ Event arrival functions


0
10
20
t

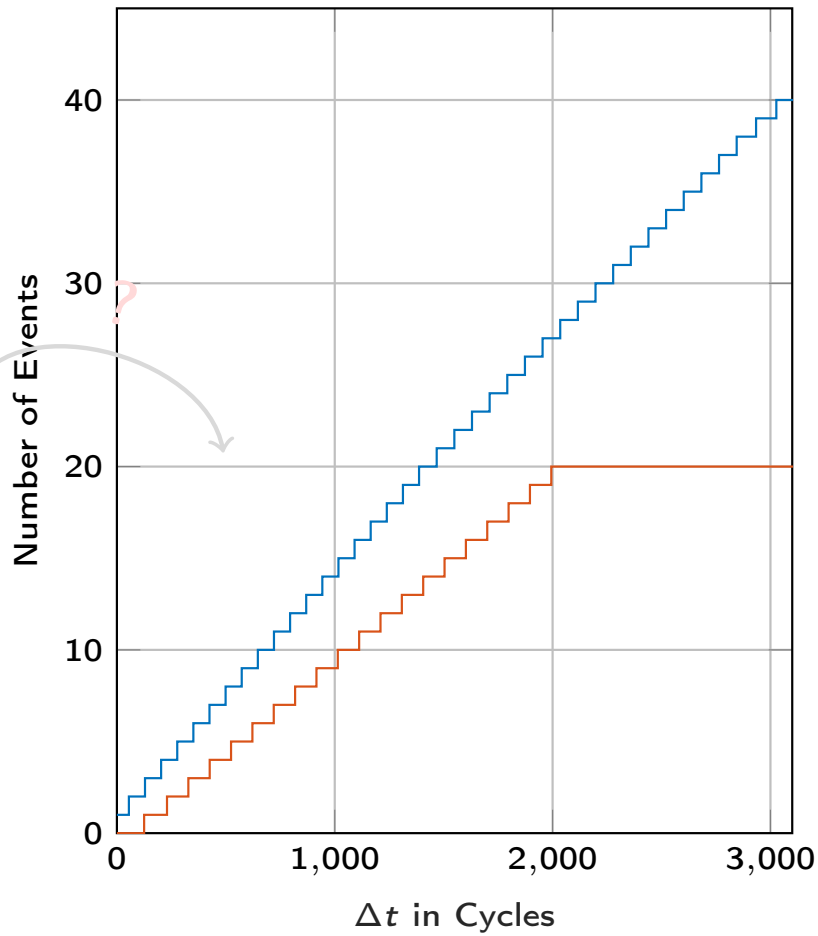
3
2
1
0
0
5
10
15
20
Δt


0
10
20
t
3
2
1
0
0
5
10
15
20
Δt

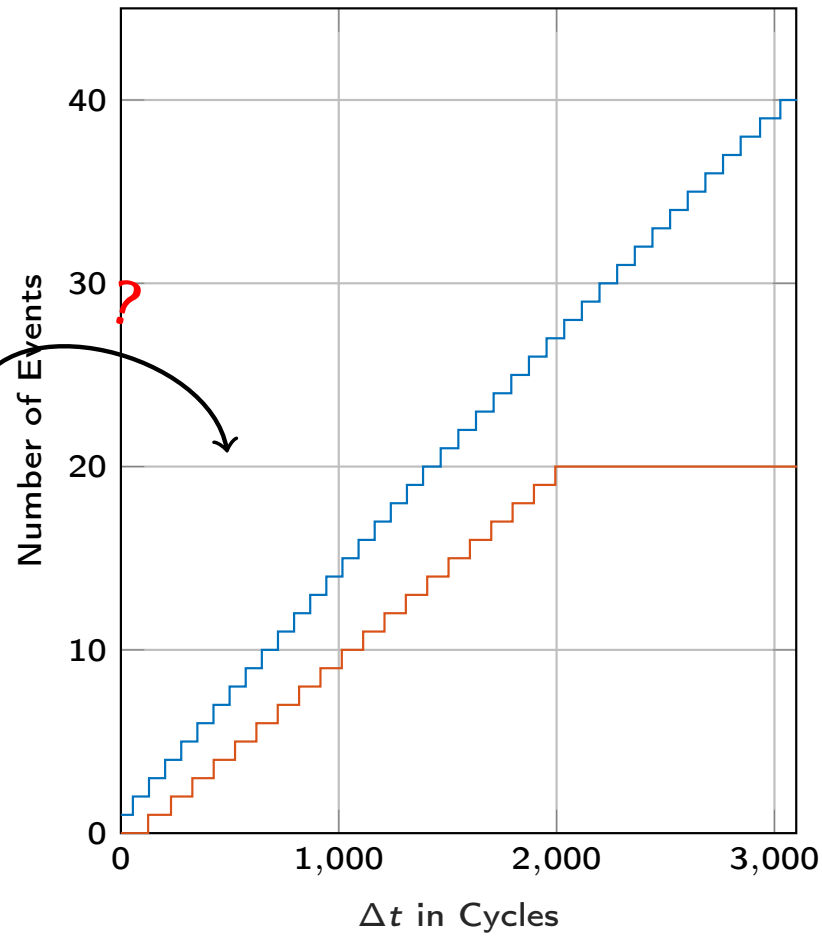
Compiler-based Event Arrival Function Extraction

```
int globalData[ 2 ] = {-1, 1};
volatile int comm;

int main() {
  for ( int i = 0; i < 20; ++i ) {
    if ( comm == 0 ) {
      globalData[ i % 2 ] = -1;
    }
  }

  return 0;
}
```

```
int globalData[ 2 ] = {-1, 1};
volatile int comm;

int main() {
  for ( int i = 0; i < 20; ++i ) {
    if ( comm == 0 ) {
      globalData[ i % 2 ] = -1;
    }
  }

  return 0;
}
```



```
int globalData[ 2 ] = {-1, 1};
volatile int comm;

int main() {
  for ( int i = 0; i < 20; ++i ) {
    if ( comm == 0 ) {
      globalData[ i % 2 ] = -1;
    }
  }

  return 0;
}
```



Extracting Curves?

- Capture traces
  - $\Rightarrow$ Potentially unsafe
- Rely on specifications
  - $\Rightarrow$ Potentially overly pessimistic
- Extraction based on the low-level representation

Extracting Curves?

- Capture traces
  $\Rightarrow$ Potentially unsafe
- Rely on specifications
  $\Rightarrow$ Potentially overly pessimistic
- Extraction based on the low-level representation

Extracting Curves?

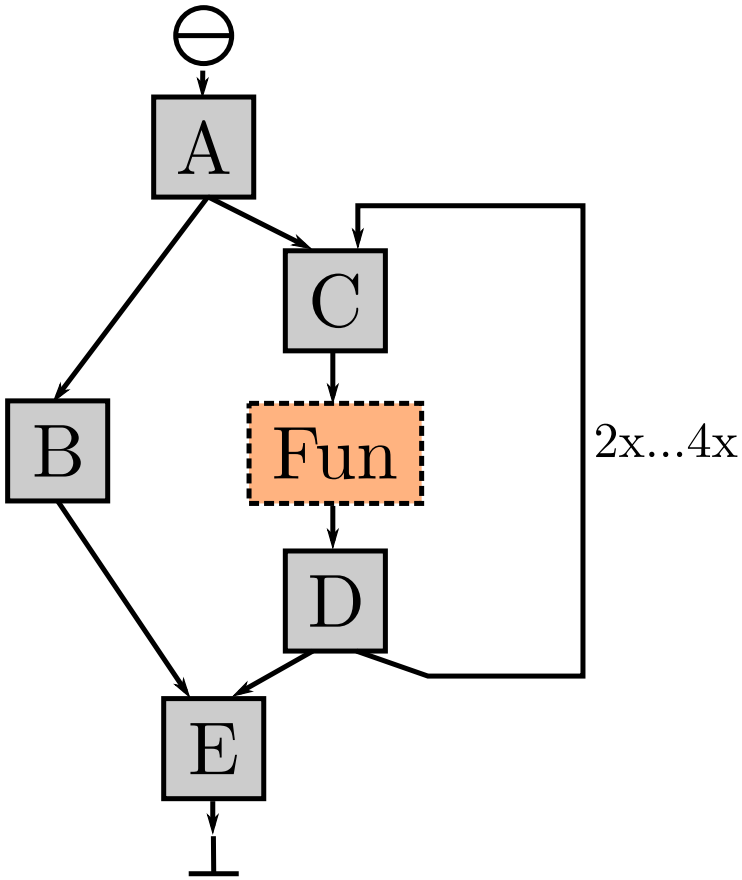
- Capture traces
  $\Rightarrow$ Potentially unsafe
- Rely on specifications
  $\Rightarrow$ Potentially overly pessimistic
- Extraction based on the low-level representation

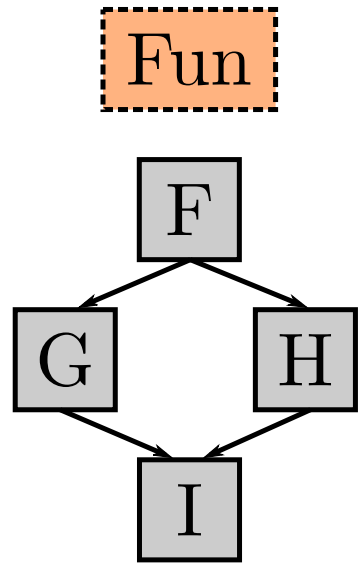
Extracting Curves?

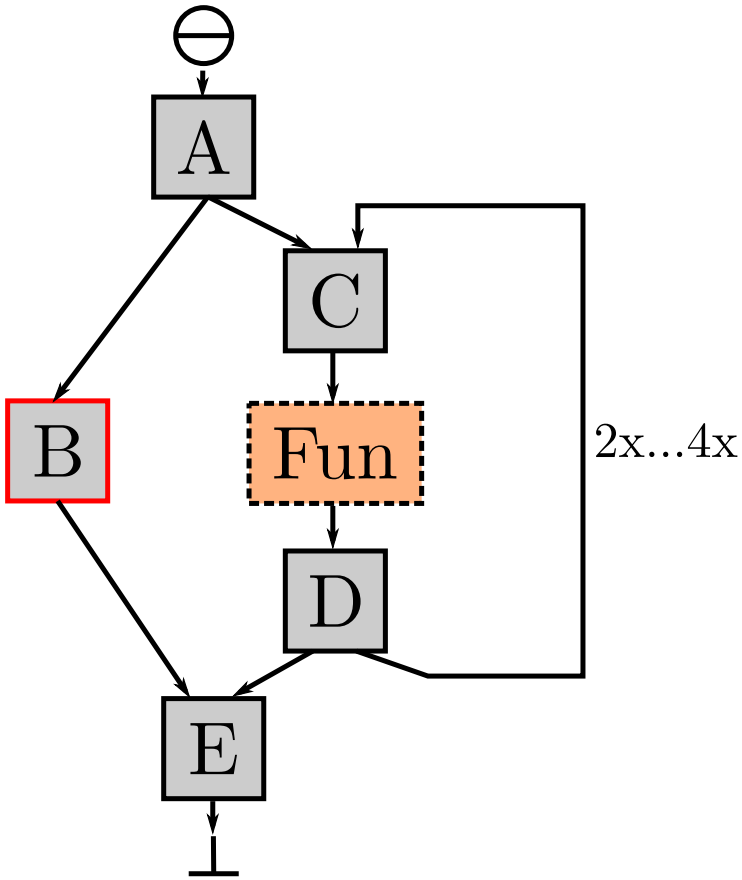
- Capture traces
  $\Rightarrow$ Potentially unsafe
- Rely on specifications
  $\Rightarrow$ Potentially overly pessimistic
- Extraction based on the low-level representation

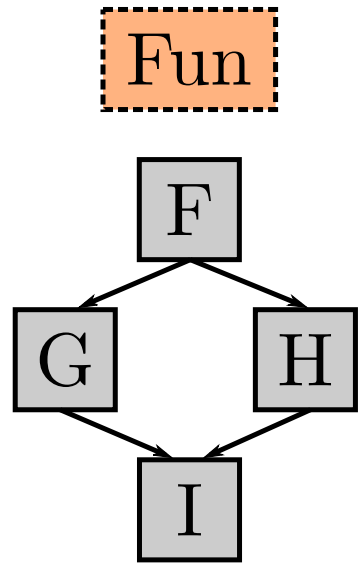
Extracting Curves?

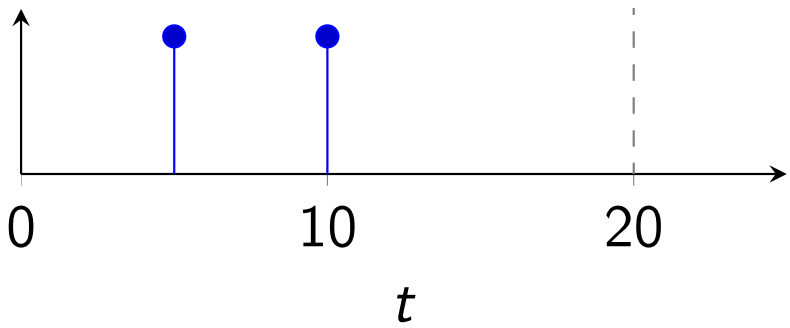
- Capture traces
  $\Rightarrow$ Potentially unsafe
- Rely on specifications
  $\Rightarrow$ Potentially overly pessimistic
- Extraction based on the low-level representation

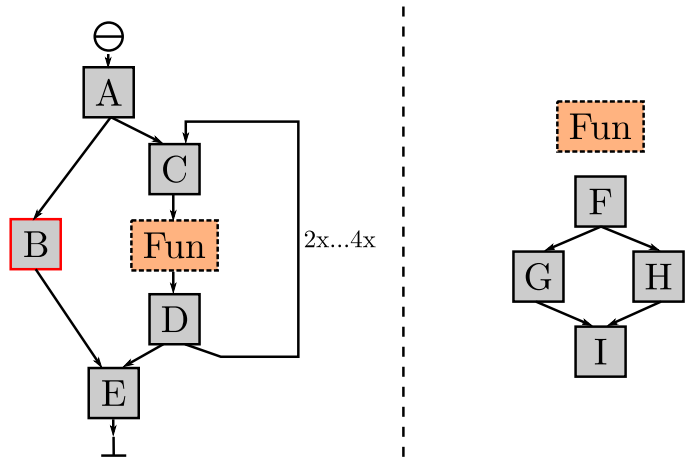

A
B
C
Fun
D
E
2x...4x


Fun
F
G
H
I

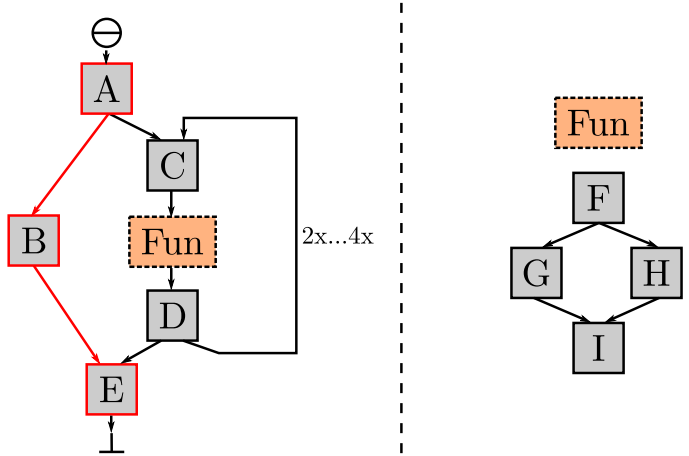

A
B
C
Fun
D
E
2x...4x

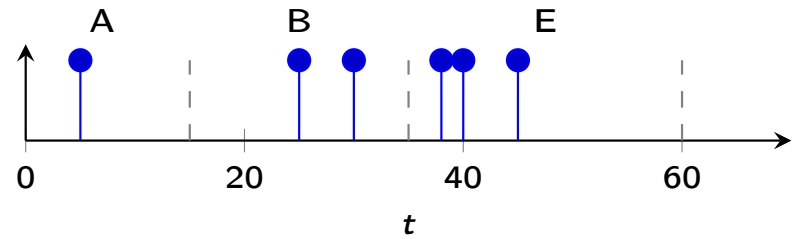

Fun
F
G
H
I


0
10
20
t


A
B
C
Fun
D
E
2x...4x
Fun
F
G
H
I

A
B
E
0
20
40
60
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t
. . .


A
B
C
Fun
D
E
2x...4x
Fun
F
G
H
I

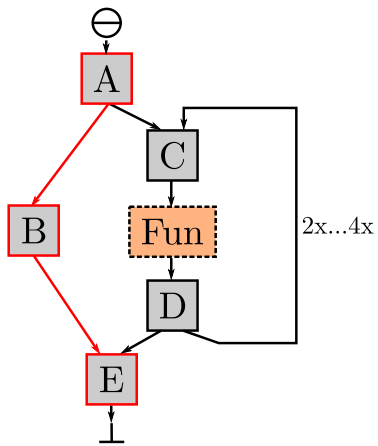

A
B
E
0
20
40
60
t

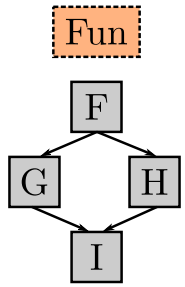
A
C
F
G
0
20
40
60
80
t

A
C
F
H
0
20
40
60
80
t

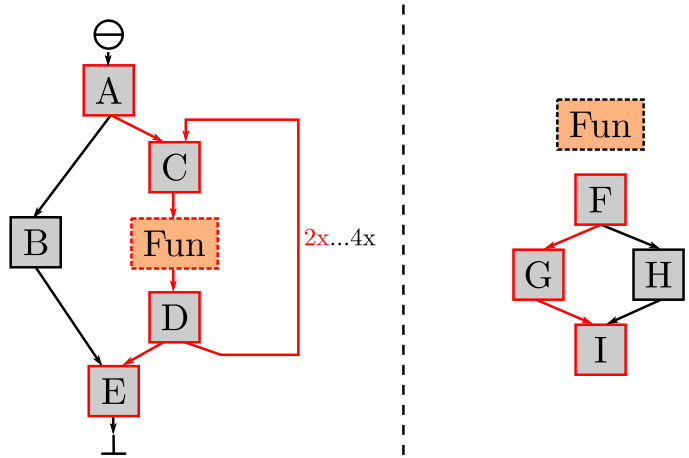
A
C
F
G
0
20
40
60
80
t

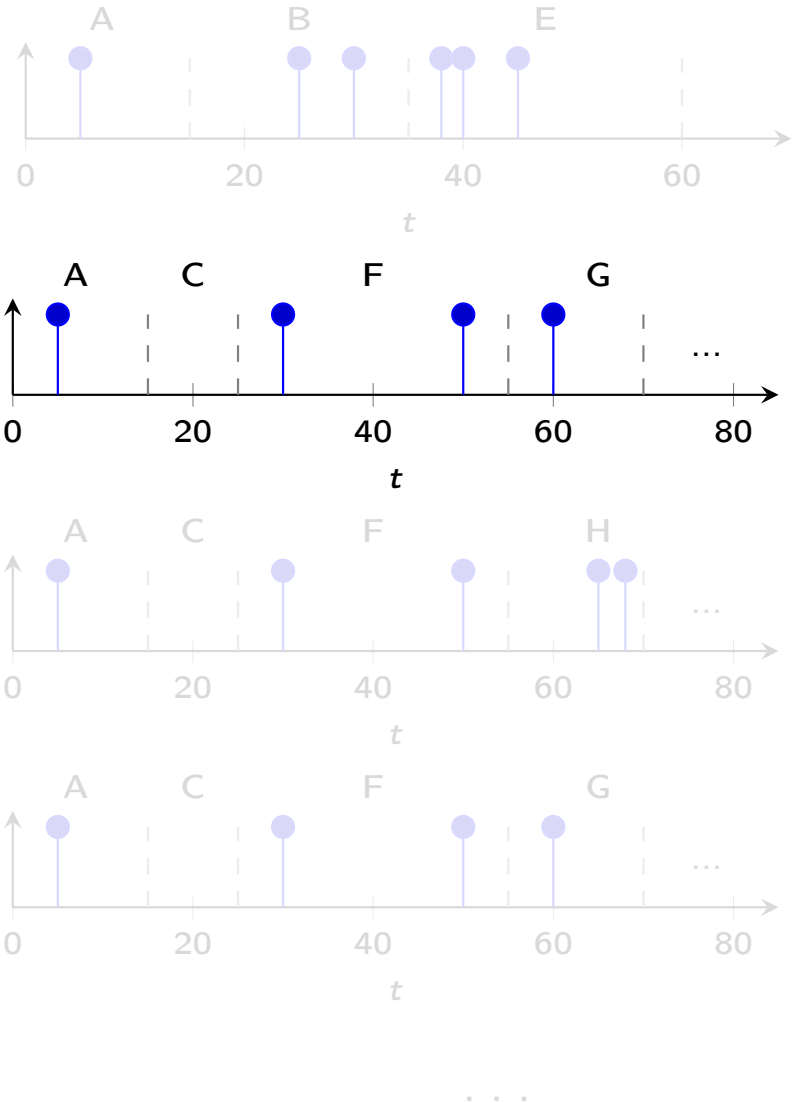
. . .

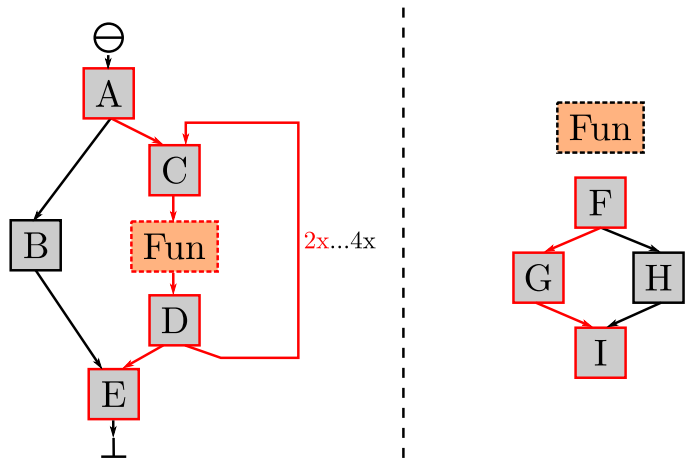

A
B
C
Fun
D
E
2x...4x


Fun
F
G
H
I

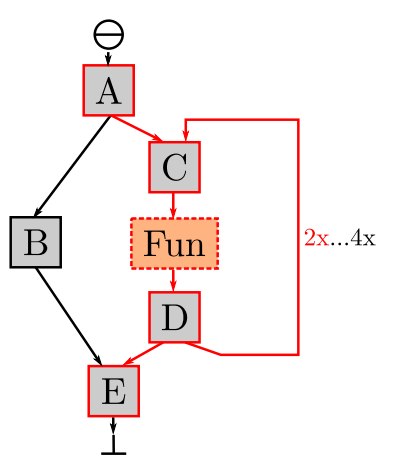
A
B
E
0
20
40
60
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t
. . .

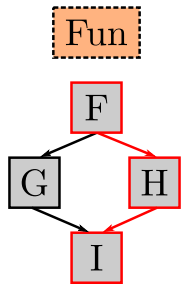

A
B
C
Fun
D
E
2x...4x
Fun
F
G
H
I

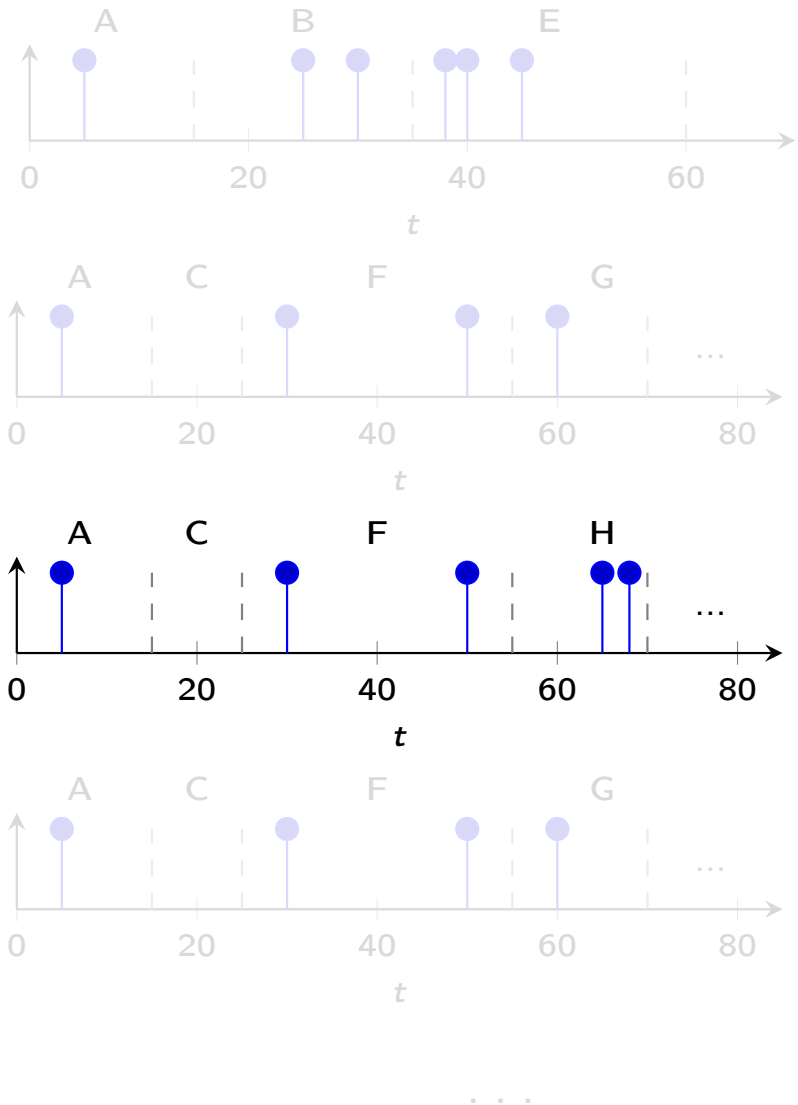

A
B
E
0
20
40
60
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t
. . .

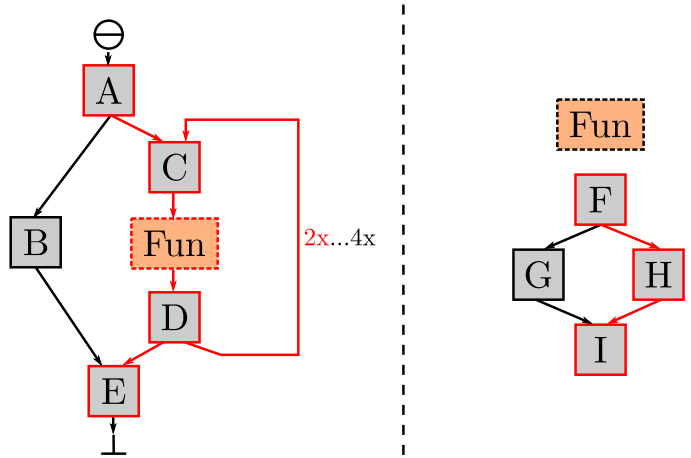

A
B
C
Fun
D
E
2x...4x
Fun
F
G
H
I

A
B
E
0
20
40
60
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t
. . .

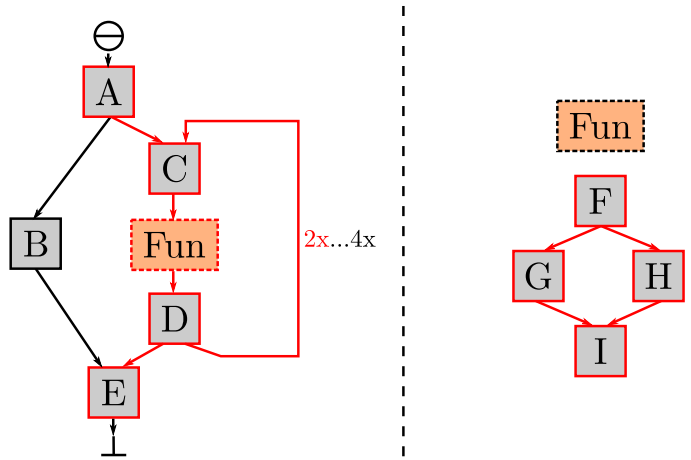

A
B
C
Fun
D
E
2x...4x

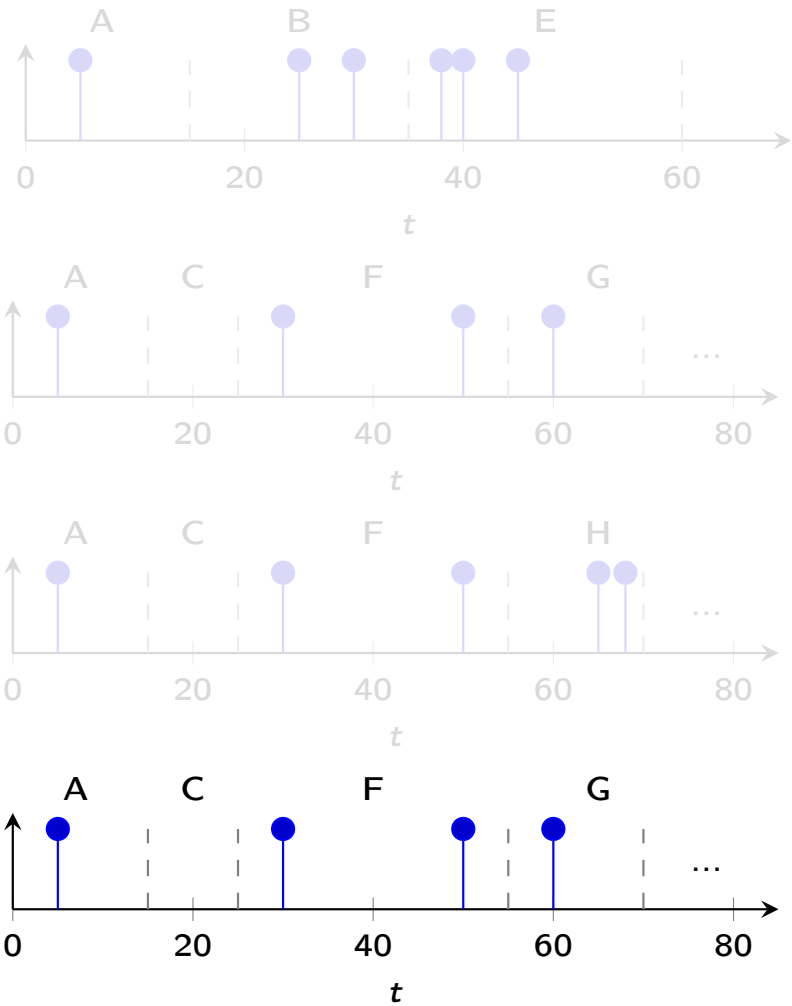

Fun
F
G
H
I

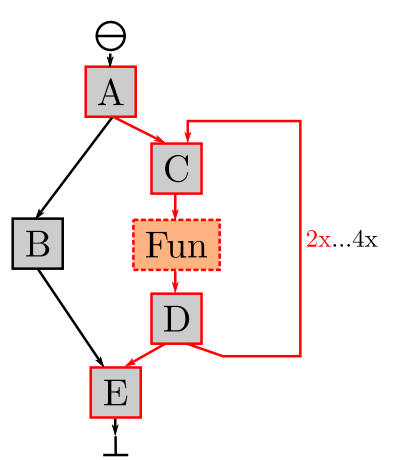

A
B
E
0
20
40
60
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t

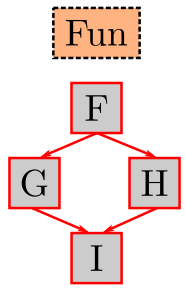

A
B
C
Fun
D
E
2x...4x
Fun
F
G
H
I

A
B
E
0
20
40
60
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t


A
B
C
Fun
D
E
2x...4x
Fun
F
G
H
I

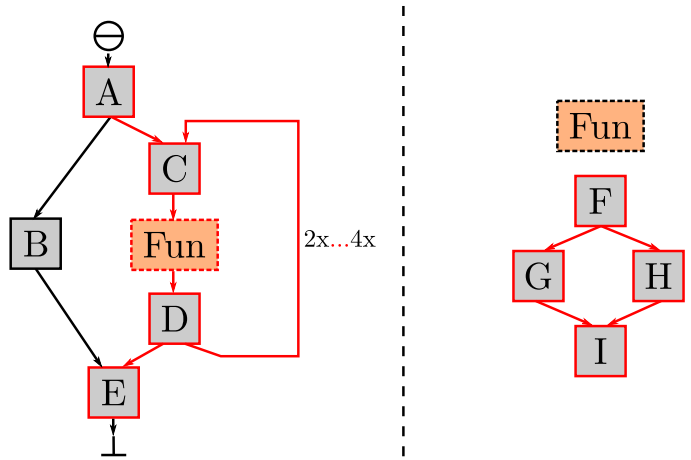

A
B
E
0
20
40
60
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t


A
B
C
Fun
D
E
2x...4x


Fun
F
G
H
I

A
B
E
0
20
40
60
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t

. . .


A
B
C
Fun
D
E
2x...4x
Fun
F
G
H
I

Path Analysis for Arrival Functions

- Explicit path analysis quickly becomes practically infeasible
  - $\Rightarrow$ Sliding window for all traces
- Adapt implicit path enumeration technique (IPET)
  - First introduced by Jacobs et al. [RTNS15]

Path Analysis for Arrival Functions

- Explicit path analysis quickly becomes practically infeasible
  $\Rightarrow$ Sliding window for all traces

- Adapt implicit path enumeration technique (IPET)
  - First introduced by Jacobs et al. [RTNS15]

Path Analysis for Arrival Functions

- Explicit path analysis quickly becomes practically infeasible
  $\Rightarrow$ Sliding window for all traces
- Adapt implicit path enumeration technique (IPET)
  - First introduced by Jacobs et al. [RTNS15]

Path Analysis for Arrival Functions

- Explicit path analysis quickly becomes practically infeasible
  $\Rightarrow$ Sliding window for all traces
- Adapt implicit path enumeration technique (IPET)
  - First introduced by Jacobs et al. [RTNS15]

Implicit Path Enumeration Technique

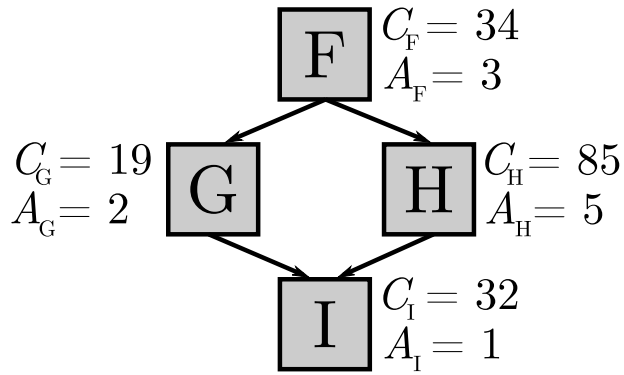
$$p_F = 1$$

$$p_F = p_{F,G} + p_{F,H}$$

$$p_{F,G} = p_{G,I}$$

$$p_{F,H} = p_{H,I}$$

$$p_{G,I} + p_{H,I} = 1$$



Implicit Path Enumeration Technique

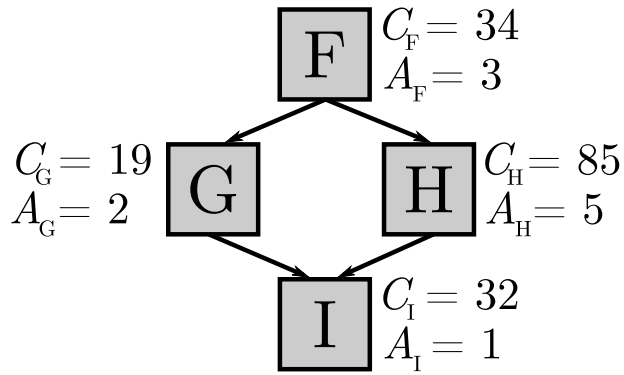
$$p_F = 1$$

$$p_F = p_{F,G} + p_{F,H}$$

$$p_{F,G} = p_{G,I}$$

$$p_{F,H} = p_{H,I}$$

$$p_{G,I} + p_{H,I} = 1$$



Implicit Path Enumeration Technique

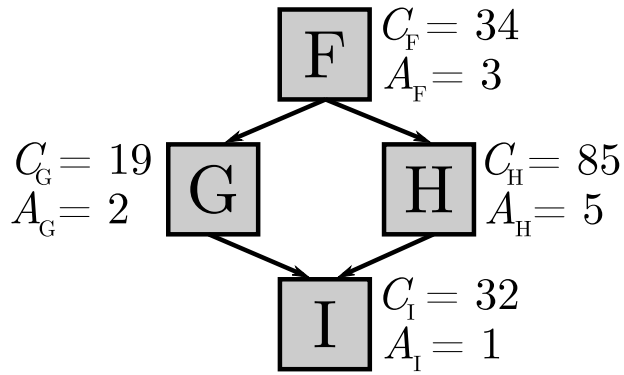
$$p_F = 1$$

$$p_F = p_{F,G} + p_{F,H}$$

$$p_{F,G} = p_{G,I}$$

$$p_{F,H} = p_{H,I}$$

$$p_{G,I} + p_{H,I} = 1$$



Implicit Path Enumeration Technique

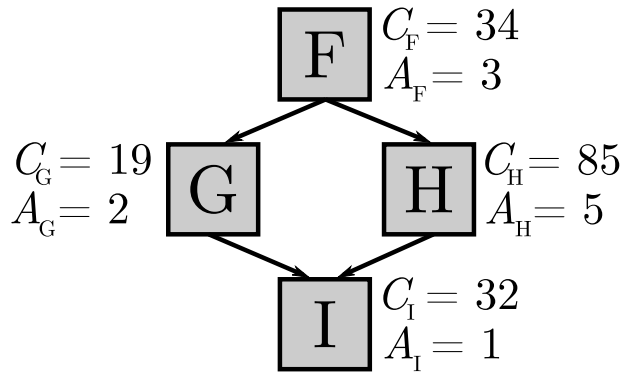
$$p_F = 1$$

$$p_F = p_{F,G} + p_{F,H}$$

$$p_{F,G} = p_{G,I}$$

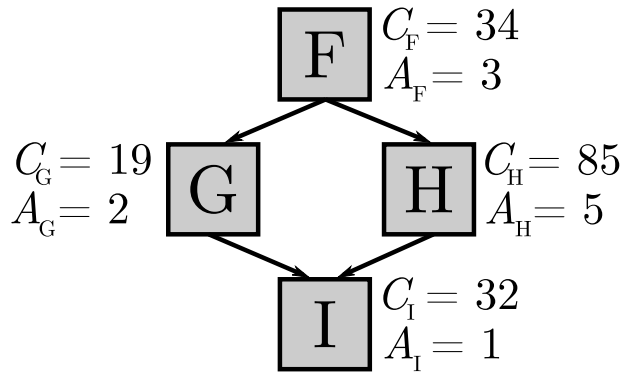
$$p_{F,H} = p_{H,I}$$

$$p_{G,I} + p_{H,I} = 1$$



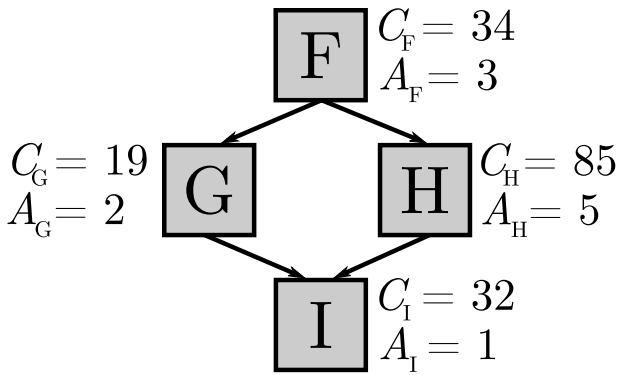
Implicit Path Enumeration Technique

$$p_F = 1$$
$$p_F = p_{F,G} + p_{F,H}$$
$$p_{F,G} = p_{G,I}$$
$$p_{F,H} = p_{H,I}$$
$$p_{G,I} + p_{H,I} = 1$$



Implicit Path Enumeration Technique

$$p_F = 1$$
$$p_F = p_{F,G} + p_{F,H}$$
$$p_{F,G} = p_{G,I}$$
$$p_{F,H} = p_{H,I}$$
$$p_{G,I} + p_{H,I} = 1$$



Enforces a complete path through the program.

Overview

Adapted IPET

Introducing "movable" sources and sinks.

$$p_{F,G} - e_G = p_{G,I} - s_{G,I}$$

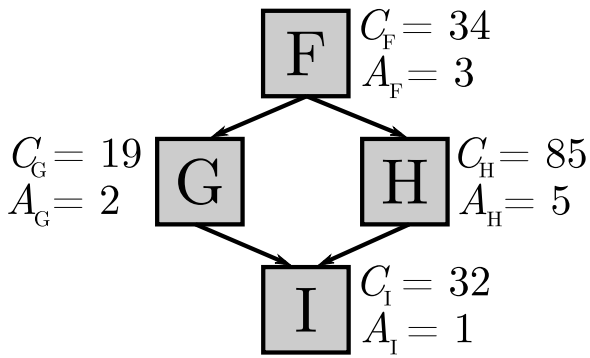
$$p_{G,I} + p_{H,I} - e_I = 0$$

Calculating number of events on path.

$$a_{\text{Tot}} = \sum_i A_i \cdot p_i$$

Maximize (resp. minimize) for a given time interval Δt.

$$\max : a_{\text{Tot}}, \text{ while } \Delta t \geq \sum_i C_i \cdot p_i \quad -(...)$$



Adapted IPET

Introducing "movable" sources and sinks.

$$p_{F,G} - e_G = p_{G,I} - s_{G,I}$$

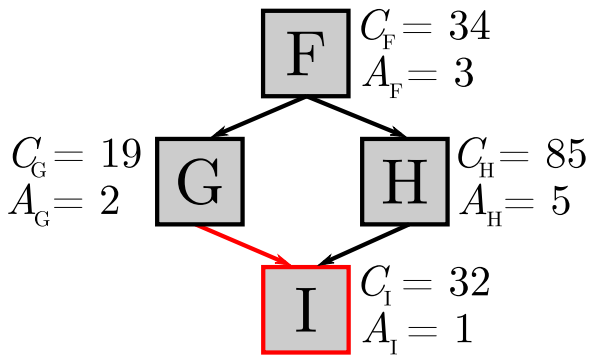
$$p_{G,I} + p_{H,I} - e_I = 0$$

Calculating number of events on path.

$$a_{\text{Tot}} = \sum_i A_i \cdot p_i$$

Maximize (resp. minimize) for a given time interval Δt.

$$\max : a_{\text{Tot}}, \text{ while } \Delta t \geq \sum_i C_i \cdot p_i \quad -(...)$$



Adapted IPET

Introducing "movable" sources and sinks.

$$p_{F,G} - e_G = p_{G,I} - s_{G,I}$$

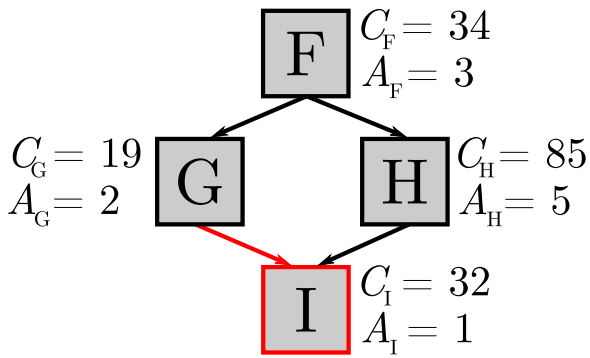
$$p_{G,I} + p_{H,I} - e_I = 0$$

Calculating number of events on path.

$$a_{\text{Tot}} = \sum_i A_i \cdot p_i$$

Maximize (resp. minimize) for a given time interval Δt.

$$\max : a_{\text{Tot}}, \text{ while } \Delta t \geq \sum_i C_i \cdot p_i \quad -(...)$$



Adapted IPET

Introducing "movable" sources and sinks.

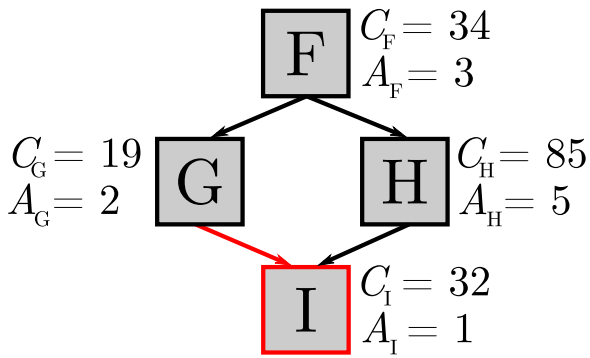
$$p_{F,G} - e_G = p_{G,I} - s_{G,I}$$
$$p_{G,I} + p_{H,I} - e_I = 0$$

Calculating number of events on path.

$$a_{\text{Tot}} = \sum_i A_i \cdot p_i$$

Maximize (resp. minimize) for a given time interval Δt.

$$\max : a_{\text{Tot}}, \text{ while } \Delta t \geq \sum_i C_i \cdot p_i \quad -(...)$$



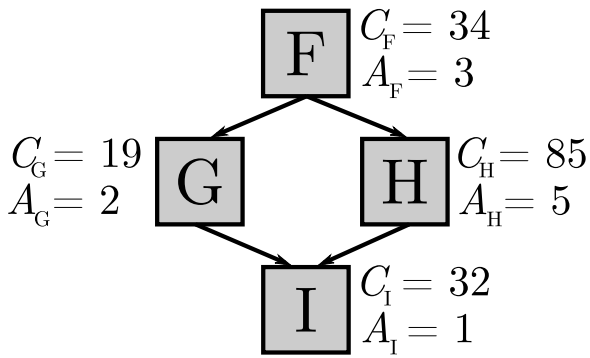
Adapted IPET

Introducing "movable" sources and sinks.

$$p_{F,G} - e_G = p_{G,I} - s_{G,I}$$
$$p_{G,I} + p_{H,I} - e_I = 0$$

Calculating number of events on path.

$$a_{\text{Tot}} = \sum_i A_i \cdot p_i$$



Maximize (resp. minimize) for a given time interval Δt.

$$\max : a_{\text{Tot}}, \text{ while } \Delta t \geq \sum_i C_i \cdot p_i \quad -(\ldots)$$

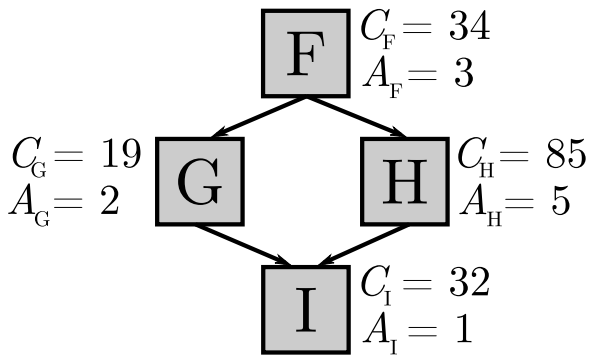
Adapted IPET

Introducing "movable" sources and sinks.

$$p_{F,G} - e_G = p_{G,I} - s_{G,I}$$
$$p_{G,I} + p_{H,I} - e_I = 0$$

Calculating number of events on path.

$$a_{\text{Tot}} = \sum_i A_i \cdot p_i$$



Maximize (resp. minimize) for a given time interval Δt.

$$\max : a_{\text{Tot}}, \text{ while } \Delta t \geq \sum_i C_i \cdot p_i \quad -(\ldots)$$

Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe

Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe

Automated Extraction

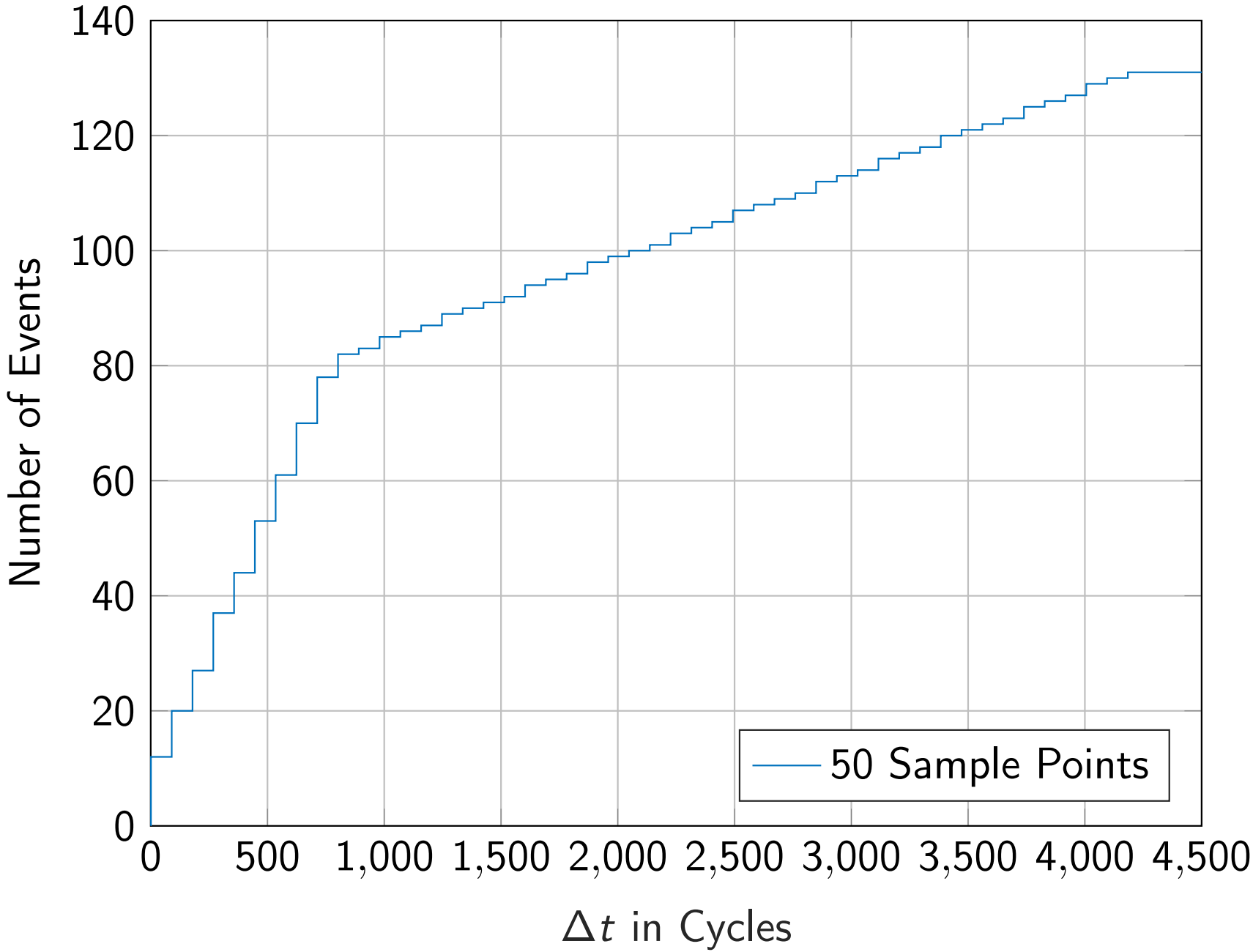
- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe

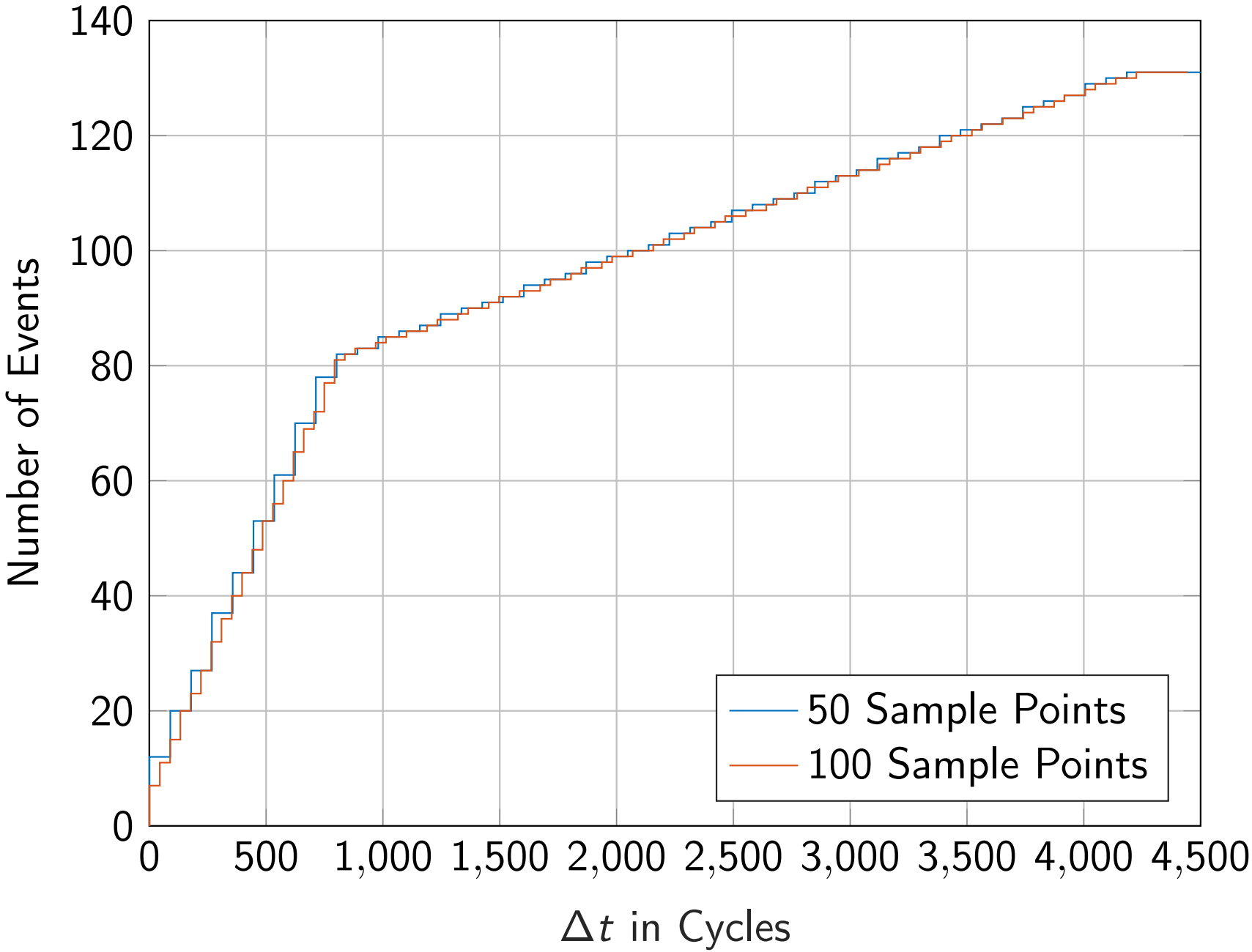
Automated Extraction

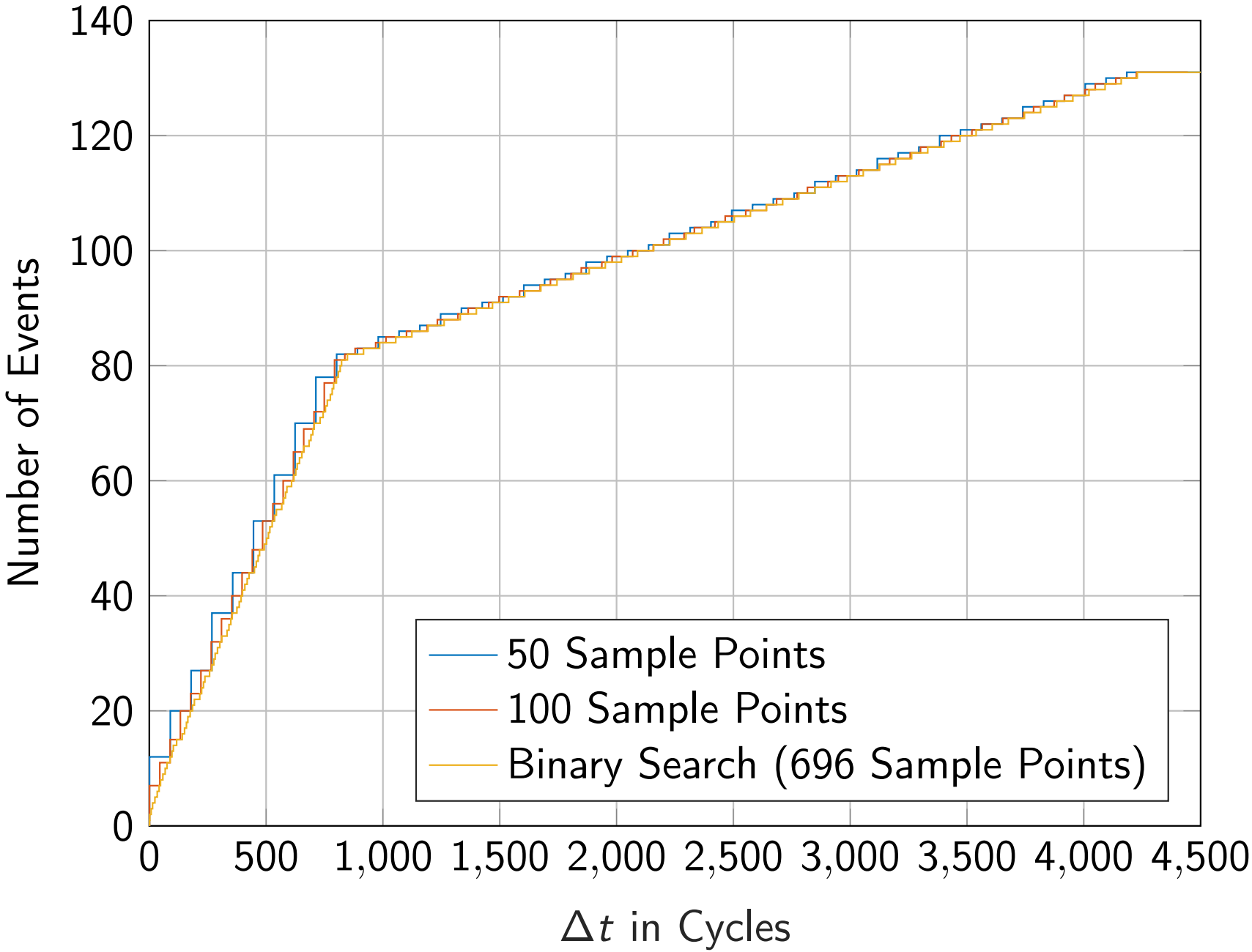
- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe

Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe


140
120
100
80
60
40
20
0
Number of Events
0
500
1,000
1,500
2,000
2,500
3,000
3,500
4,000
4,500
Δt in Cycles
50 Sample Points


140
120
100
80
60
40
20
0
Number of Events
0
500
1,000
1,500
2,000
2,500
3,000
3,500
4,000
4,500
Δt in Cycles
50 Sample Points
100 Sample Points


140
120
100
80
60
40
20
0
Number of Events
0
500
1,000
1,500
2,000
2,500
3,000
3,500
4,000
4,500
Δt in Cycles
50 Sample Points
100 Sample Points
Binary Search (696 Sample Points)

Overview

Future Applications

- Bus-oriented low-level optimizations
- Precise analyses of complex multicore architectures
- Extension to multitask-multicore

Future Applications

- Bus-oriented low-level optimizations
- Precise analyses of complex multicore architectures
- Extension to multitask-multicore

Future Applications

- Bus-oriented low-level optimizations
- Precise analyses of complex multicore architectures
- Extension to multitask-multicore