Power aware WCRT analysis of synchronous programs

Hugh Wang

Partha S Roop

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Outline

- Background
  - Dynamic Voltage and Frequency Scaling
  - Related work on DVFS
- Problem statement
- Methodology
- Results
- Conclusions

Power consumption

- Power consumption
  - $P = P_{static} + P_{dynamic}$
    - Static: current leakage
  - $P_{dynamic} \propto CV^2f$
- Dynamic Voltage and Frequency Scaling(DVFS)
  - Reduce dynamic power consumption by lowering the voltage and frequency.
  - Through inserting DVFS commands at specific program points.

Dynamic Voltage and Frequency Scaling

No	f^{epu} (MHz)	CPU Volt. (V)	f^{int} (MHz)	f^{ext} (MHz)
F_1	100	0.85	50	100
F_2	200	1.0	50	100
F_3	300	1.1	50	100
F_4	200	1.0	100	100
F_5	300	1.1	100	100
F_6	400	1.3	100	100
F_7	400	1.3	200	100
F_8	133	0.85	66	133
F_9	265	1.0	133	133

DVFS in BitsyX system

Kihwan Choi; Wonbok Lee; Soma, R.; Pedram, M., "Dynamic voltage and frequency scaling under a precise energy model considering variable and fixed components of the system power dissipation," ICCAD 2004

Dynamic Voltage and Frequency Scaling

- DVFS influence
  - From 400Mhz to 100Mhz
    - 4 times the execution time
  - Voltage reduce from 1.35 to 0.85
    - 90% less dynamic power (mW)
    - 60% less energy per cycle (Joule/Cycle)

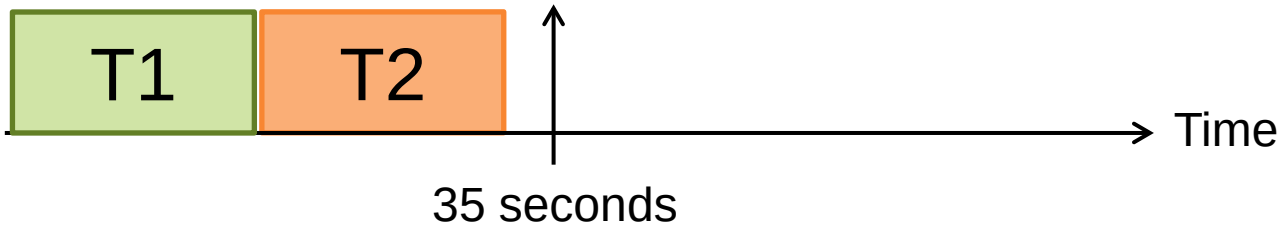
Related work on DVFS

- Plethora of techniques for DVFS combined with schedulability analysis.
- Main idea – to reclaim the slack.
- Assumes that the WCET of tasks are given.

A. Orgerie, M. Assuncao, and L.Lefevre. *A survey on techniques for Improving the energy efficiency of large-scale distributed systems*. *ACM Comput 2014*

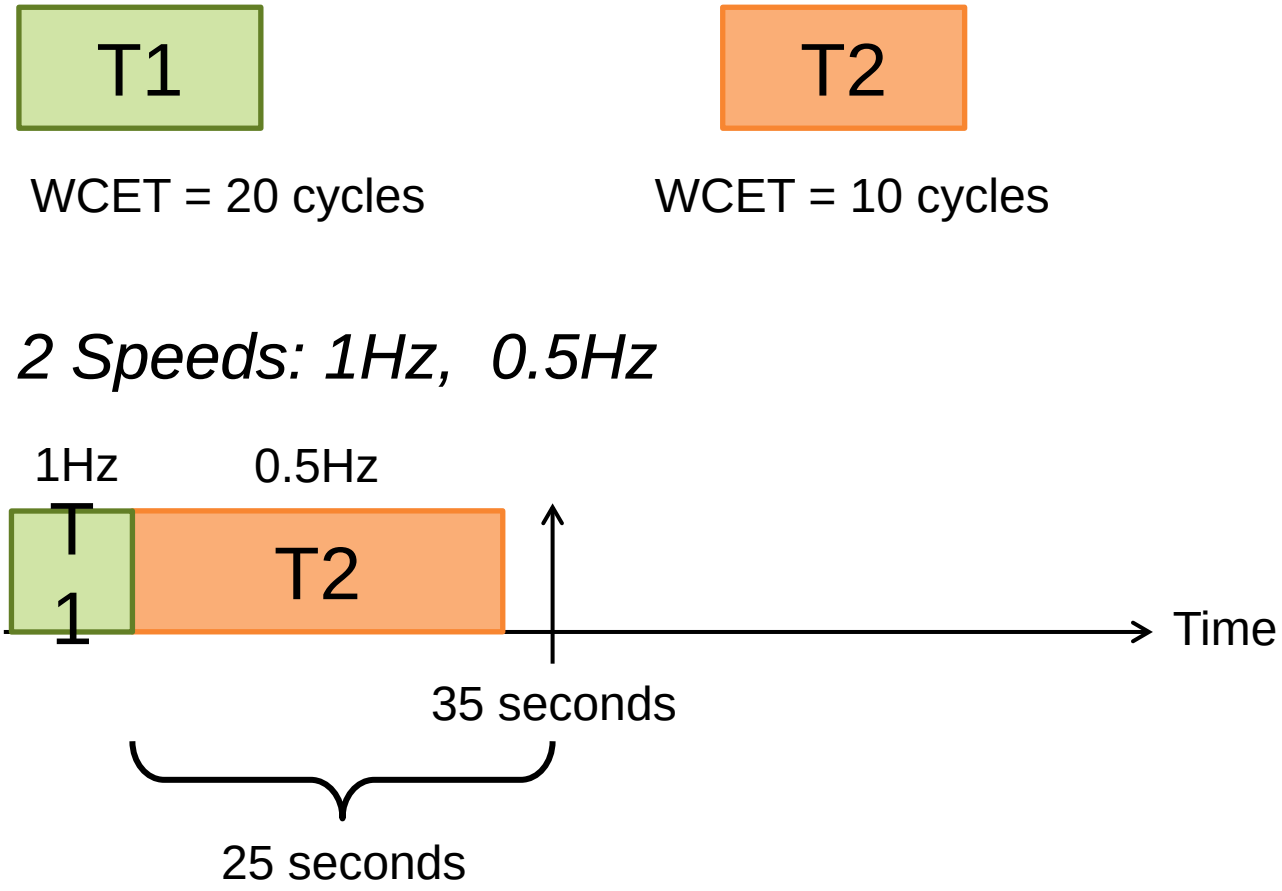
Related work on DVFS

2 Speeds: 1Hz, 0.5Hz



Deadline satisfied at 1Hz.

Related work on DVFS



Outline

- Background
  - Dynamic Voltage and Frequency Scaling
  - Related work on DVFS
- Problem statement
  - Timing analysis and DVFS
  - Proposal
- Methodology
- Results
- Conclusions

Problem statement

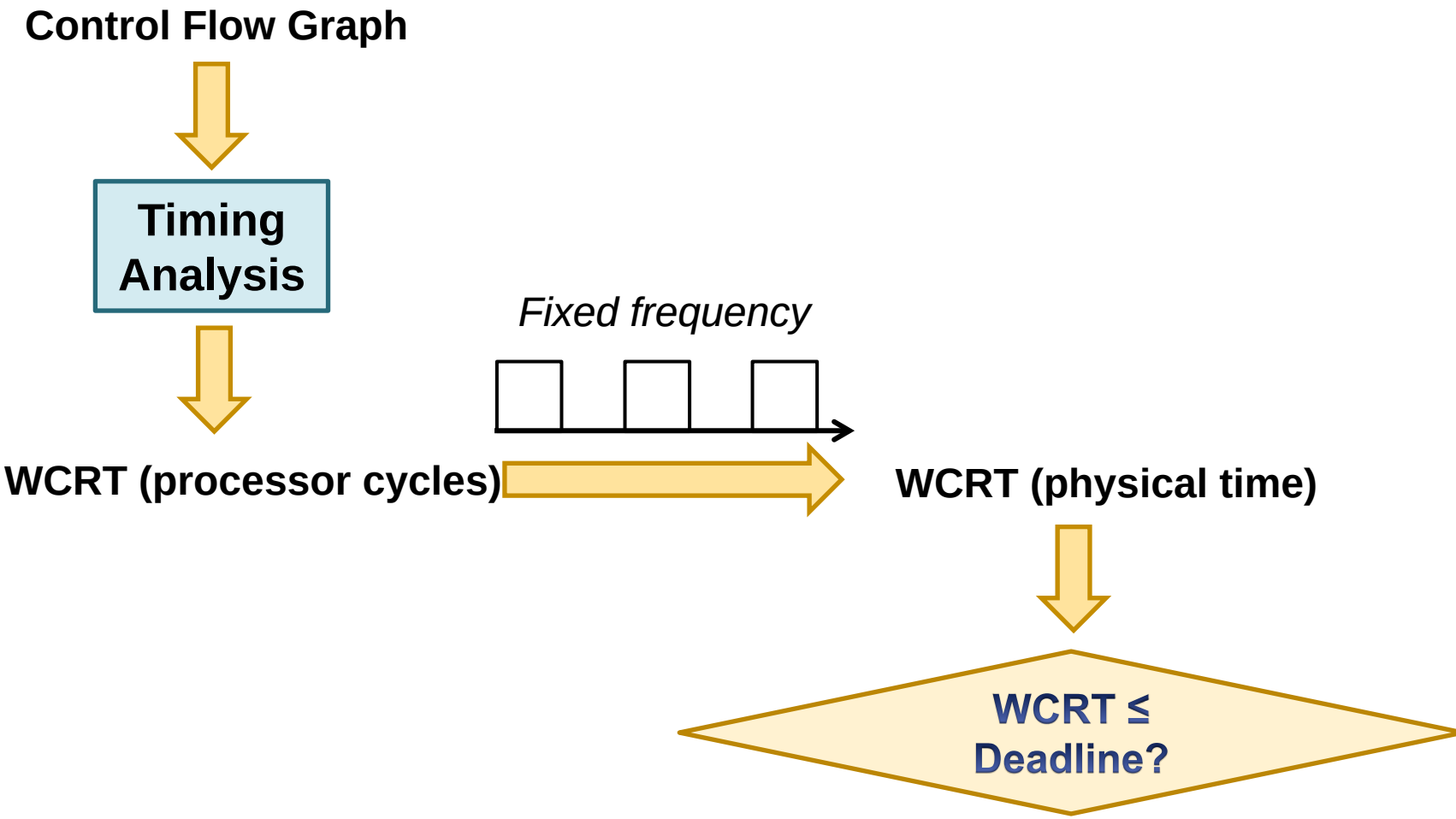
- Minimal work so far on combining DVFS with WCET analysis.
- This problem is also interesting for synchronous programs. This is not yet investigated.
- Energy aware timing analysis for synchronous programs.

Timing analysis

- Changing the frequency of a processor influences the Worst Case Reaction Time(WCRT).

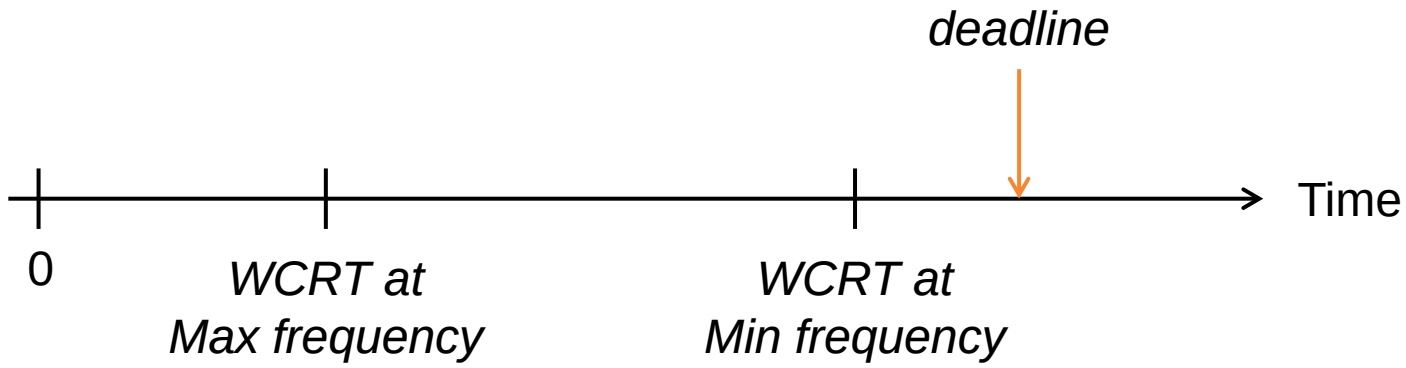
- However, most timing analysis techniques are based on a single frequency.

Timing analysis



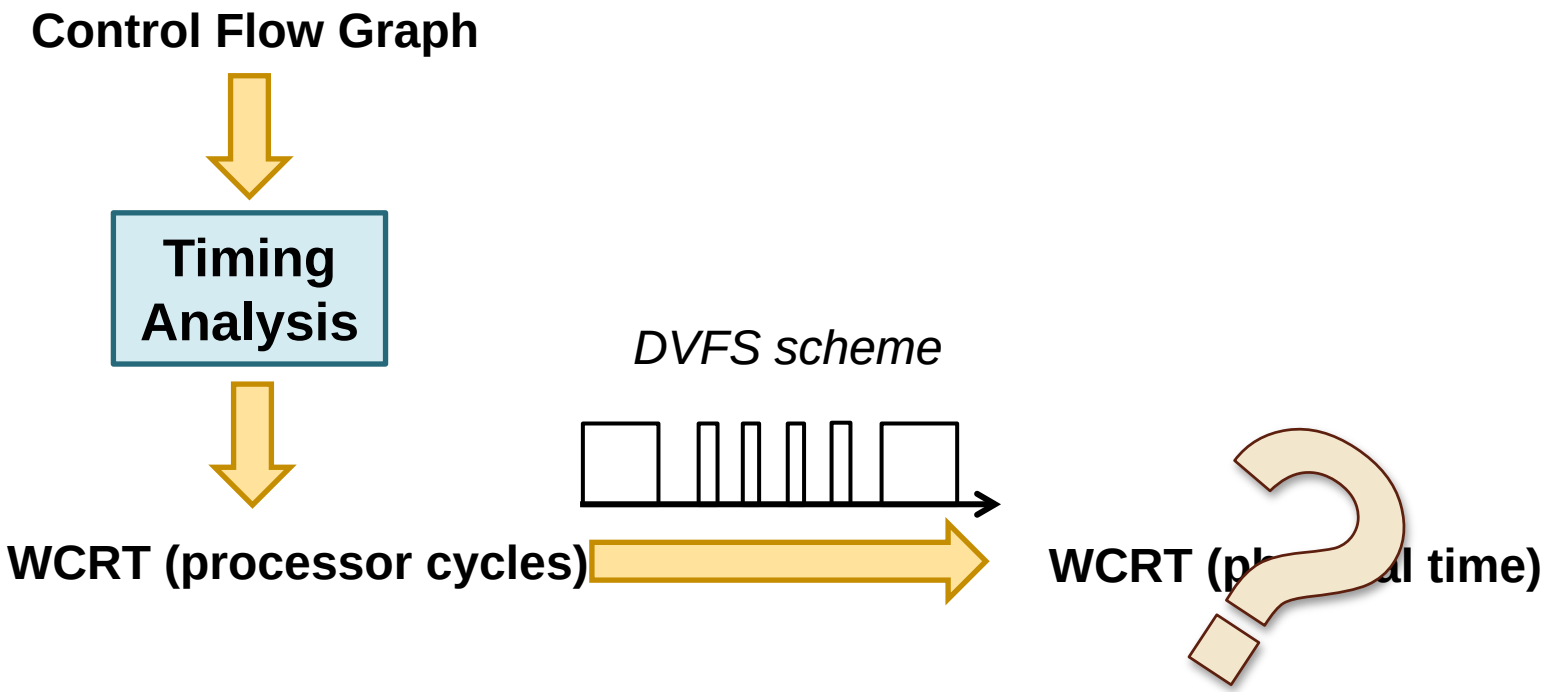
Static safe frequency

Static safe frequency



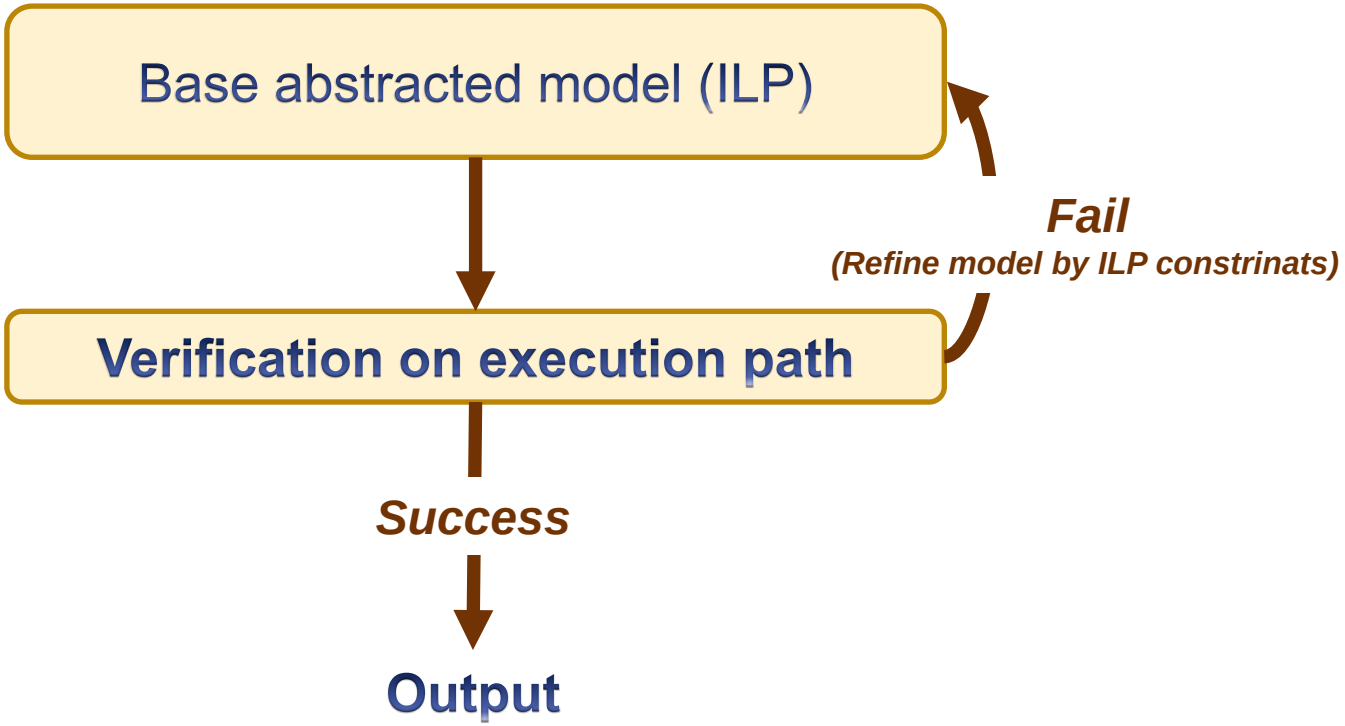
- Maximum frequency is only needed for on the critical path.
- The system can run on lower frequency in other time.

Timing analysis

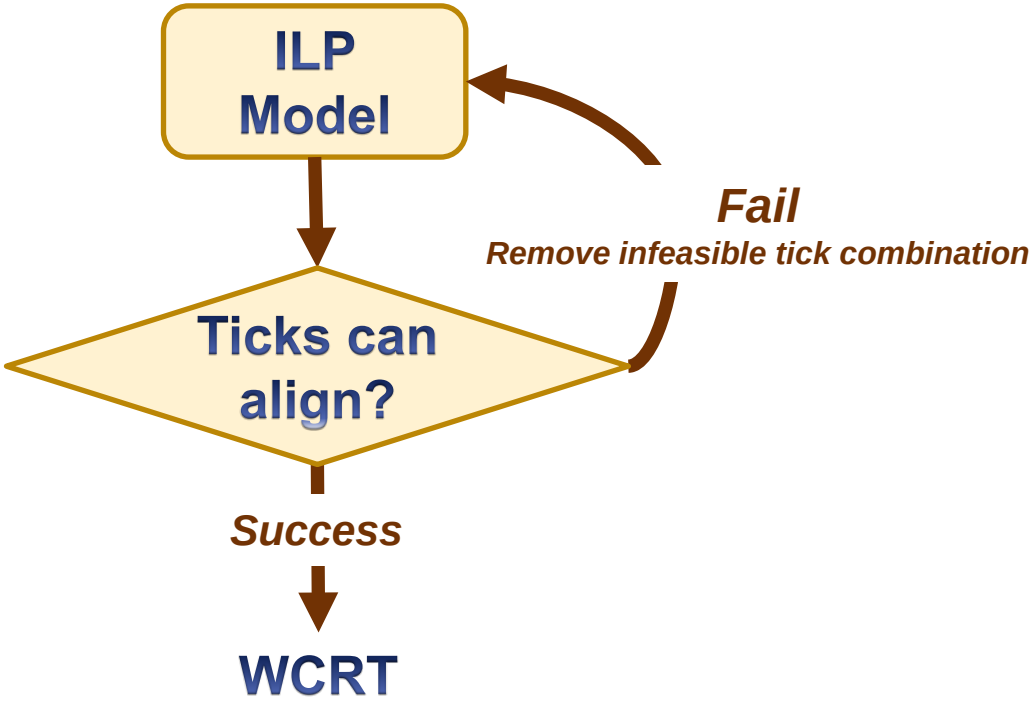


Proposal

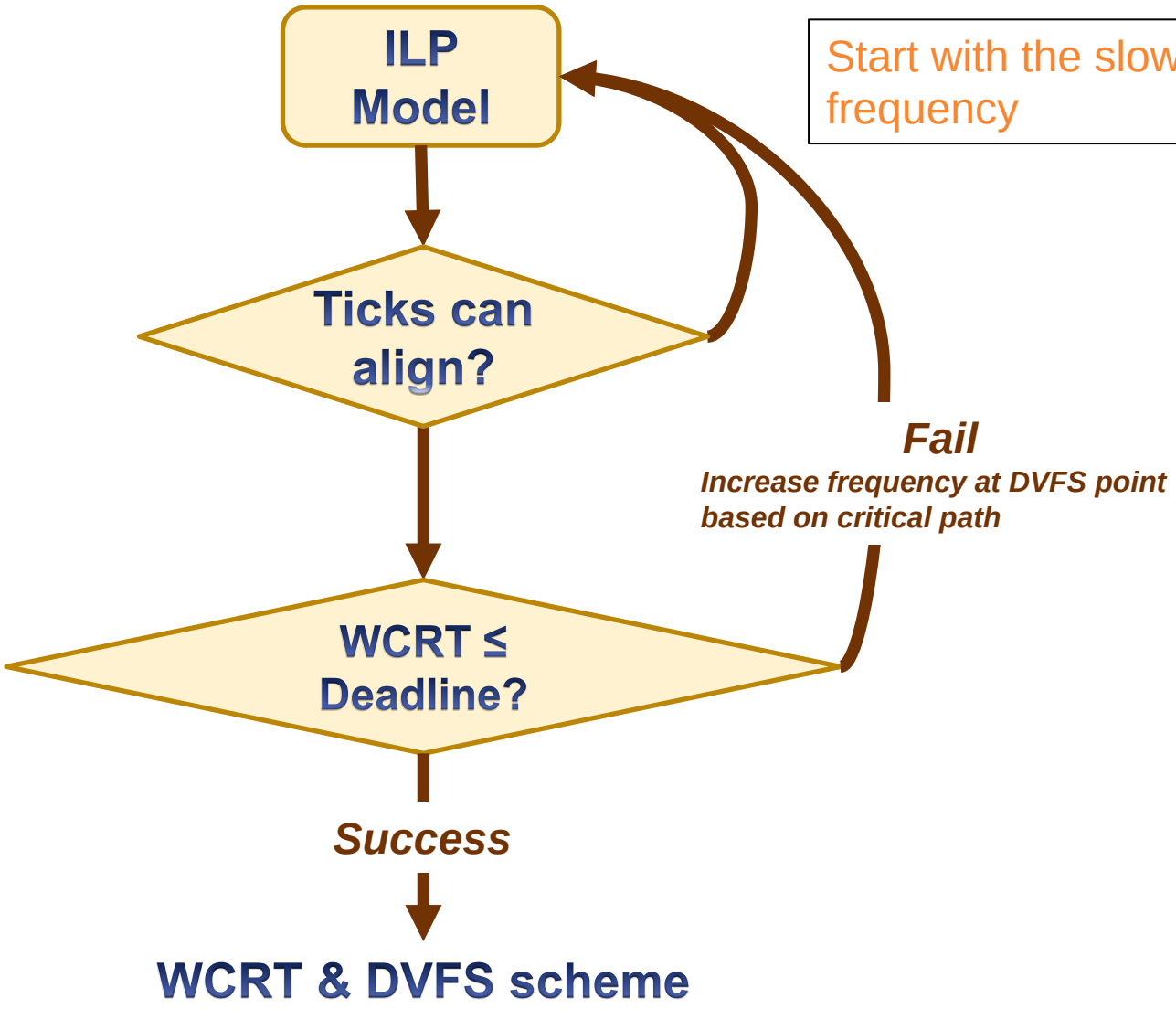
- Timing analysis that considers DVFS schemes.
- For synchronous programs with complex control flow.
- Base on ILPc.



Proposal


ILP Model
Fail
Remove infeasible tick combination
Ticks can align?
Success
WCRT

Proposal


ILP Model
Start with the slowest frequency
Ticks can align?
Fail
Increase frequency at DVFS point based on critical path
WCRT ≤ Deadline?
Success
WCRT & DVFS scheme

Proposal

- Extend the base ILP model to capture DVFS.
- Develop an analysis component to adjust DVFS scheme based on the critical paths.

Outline

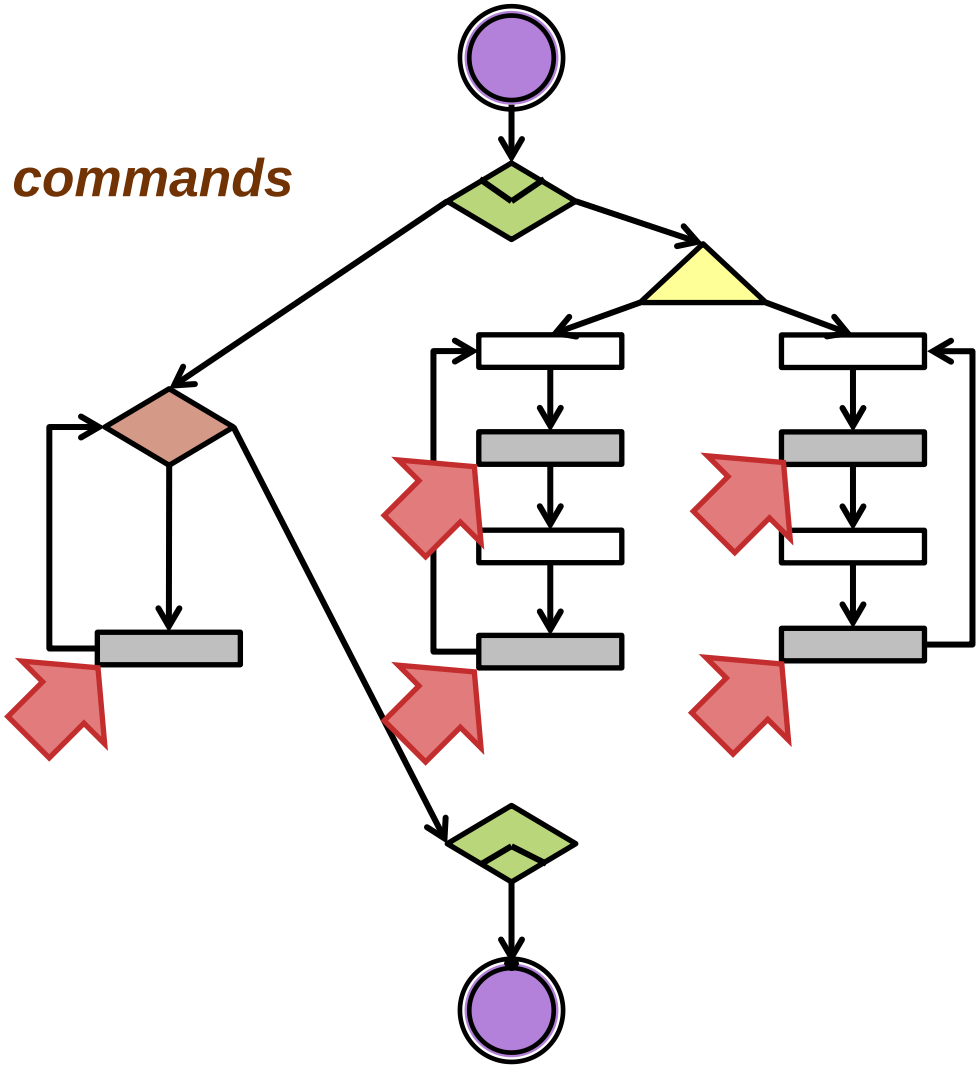
- Background
- Problem statement
  - Timing analysis and DVFS
  - Proposal
- Methodology
  - Extending base ILP model
  - DVFS analysis component
- Results
- Conclusions

DVFS scheme

- Where to place the DVFS commands?
  - At the execution of EOT and join.

DVFS scheme

DVFS commands



DVFS scheme

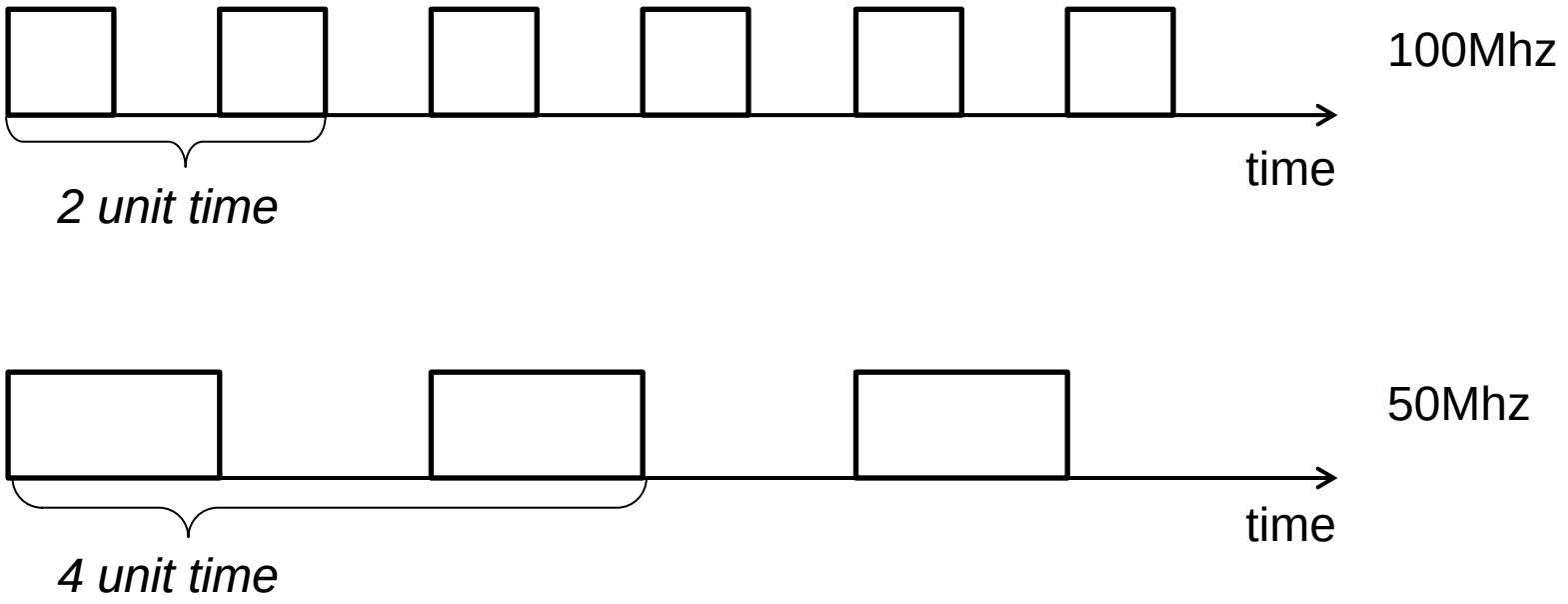
- Where to place the DVS commands?
  - At the execution of EOT and join.
- What is the value (frequency)?
  - Determine through timing analysis.
- For illustration purpose, assume our processor runs at 100Mhz,50Mhz and 25Mhz.

Bridging processor cycles and physical time

- Unit time
  - Reference time
  - The fastest clock of the processor

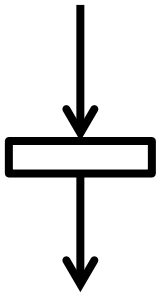
Bridging processor cycles and physical time

- *Instruction of 2 cycles*
- *Same processor cycles, different physical time*



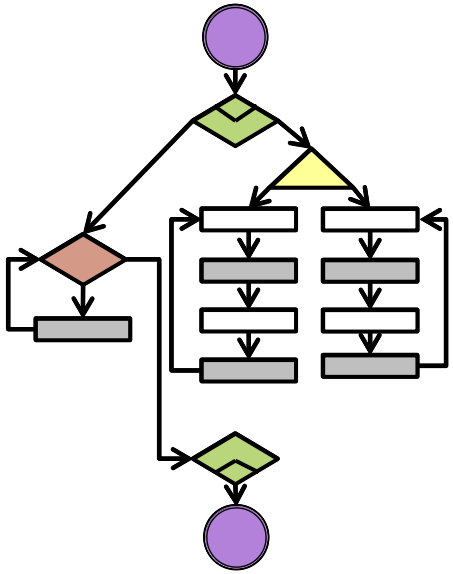
Bridging processor cycles and physical time

- Unit time
  - Reference time
  - The fastest clock of the processor
- Similarly, deadline can be converted into unit time as well.



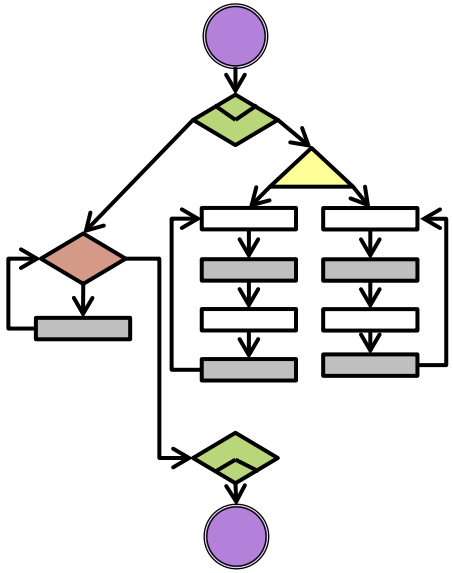
Clock	Processor cycles	Unit time
100MHz	2	2
50MHz	2	4
25MHz	2	8

Extending ILPc

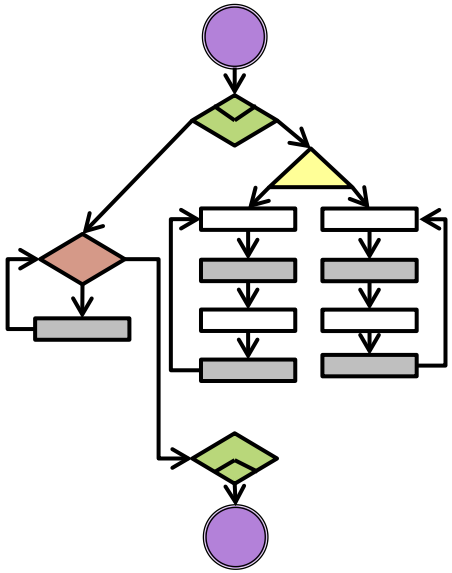


Replicate the ILP model for different frequency.

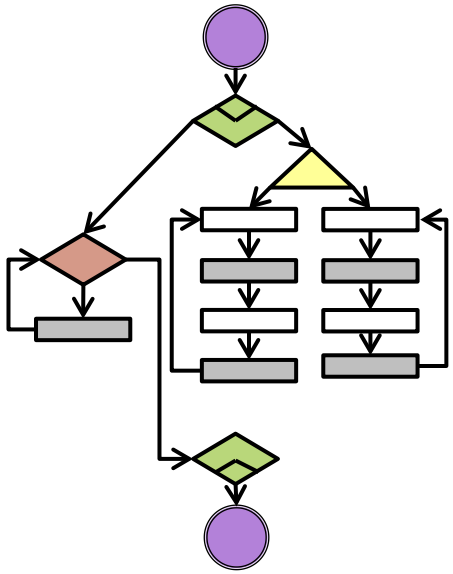
Extending ILPc



Unit time for 100MHz

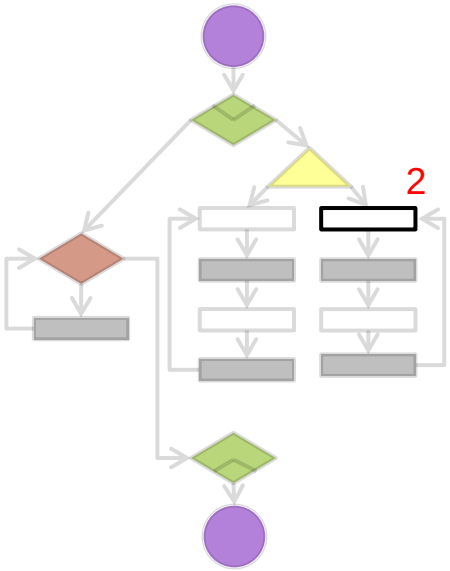


Unit time for 50MHz

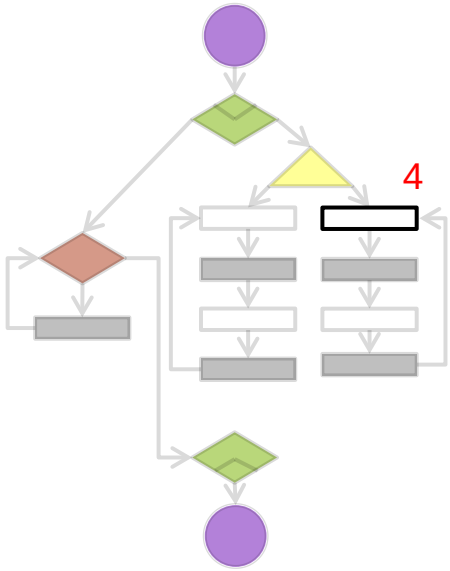


Unit time for 25MHz

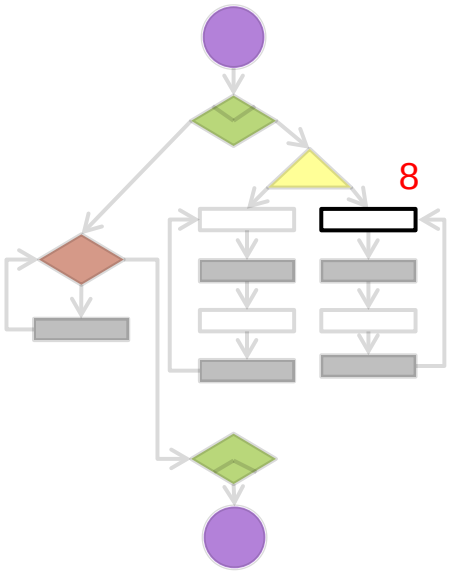
Extending ILPc



Unit time for 100MHz

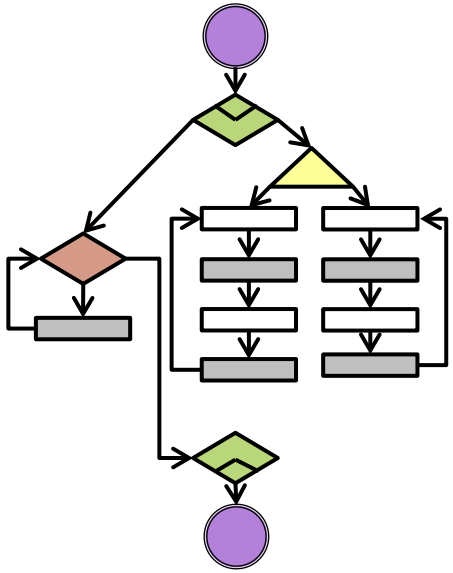


Unit time for 50MHz



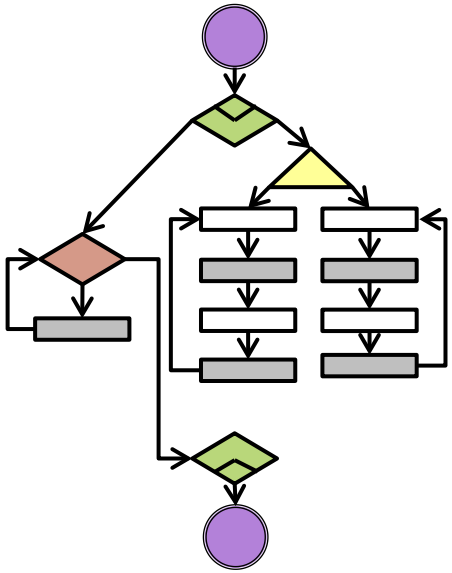
Unit time for 25MHz

Extending ILPc



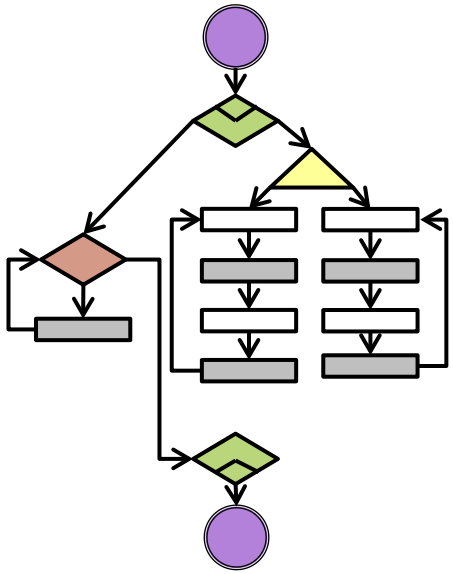
Unit time for 100MHz

a



Unit time for 50MHz

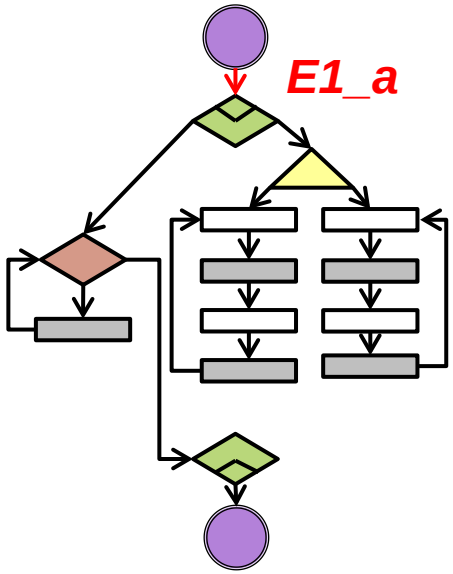
b



Unit time for 25MHz

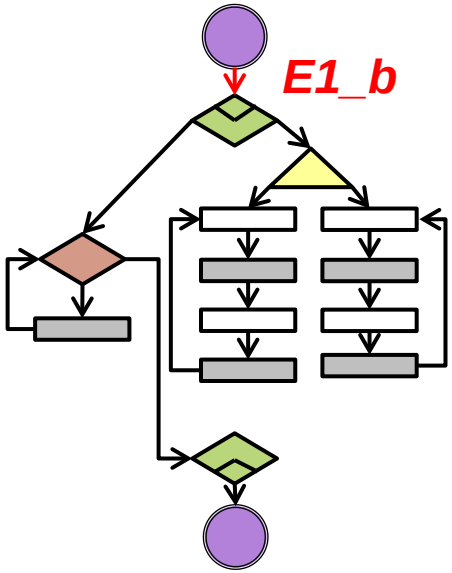
c

Extending ILPc



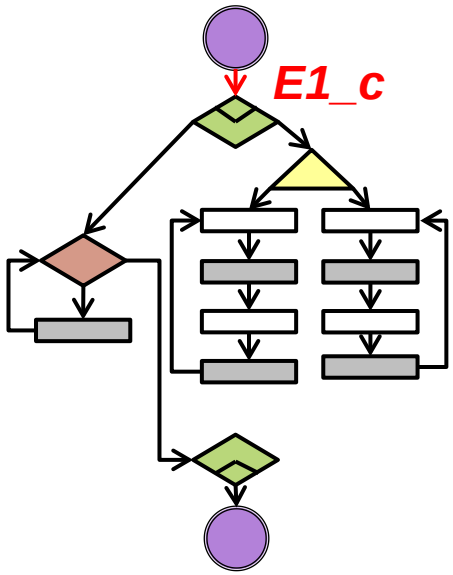
Unit time for 100MHz

a



Unit time for 50MHz

b

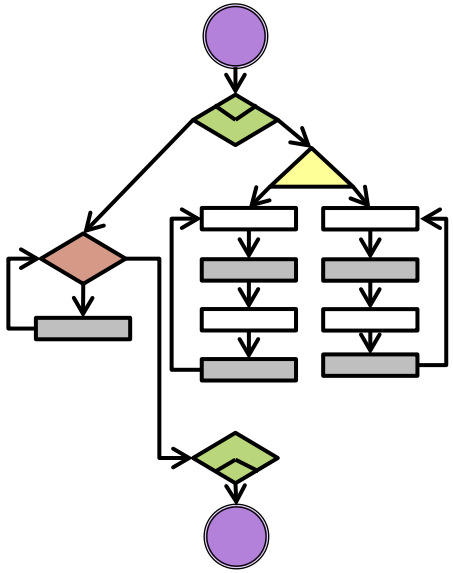


Unit time for 25MHz

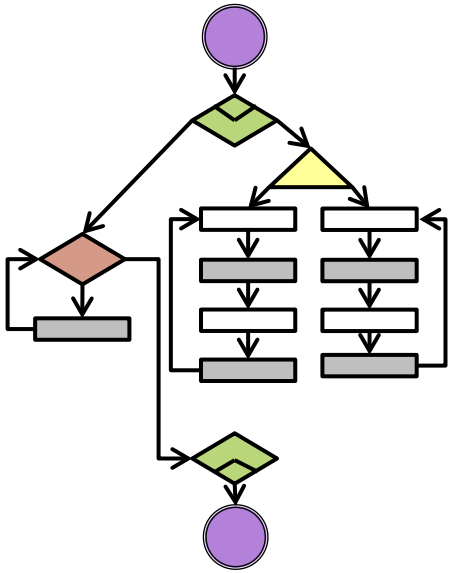
c

E1 -> ***E1_a***, ***E1_b*** and ***E1_c***

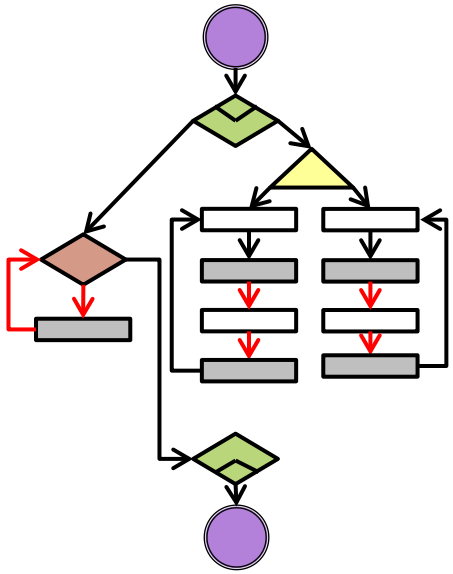
Extending ILPc



Unit time for 100MHz



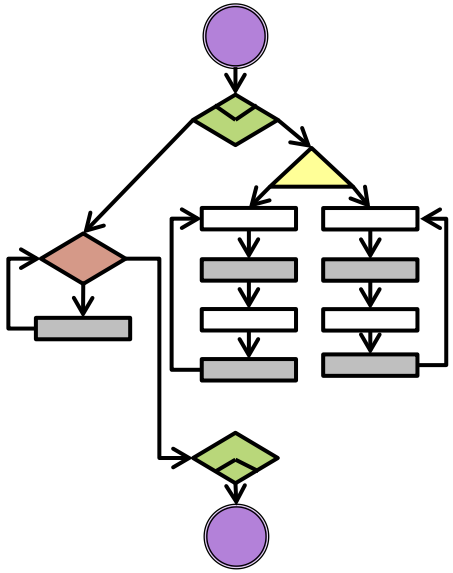
Unit time for 50MHz



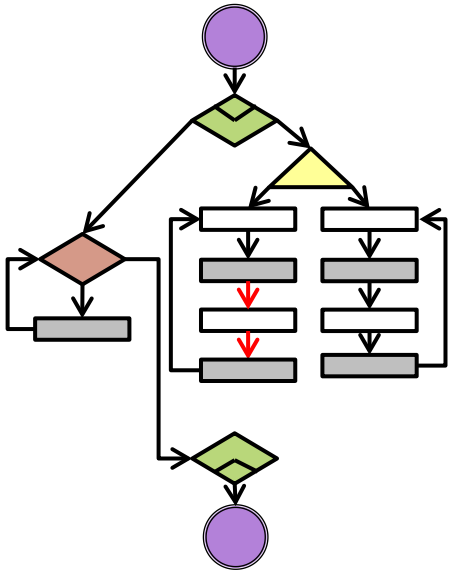
Unit time for 25MHz

An execution path at slowest frequency.

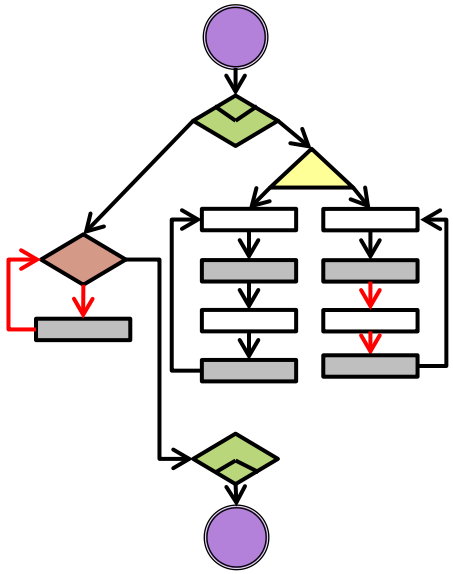
Extending ILPc



Unit time for 100MHz



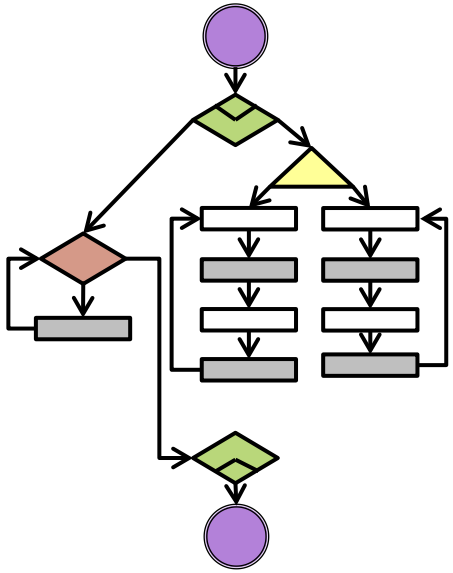
Unit time for 50MHz



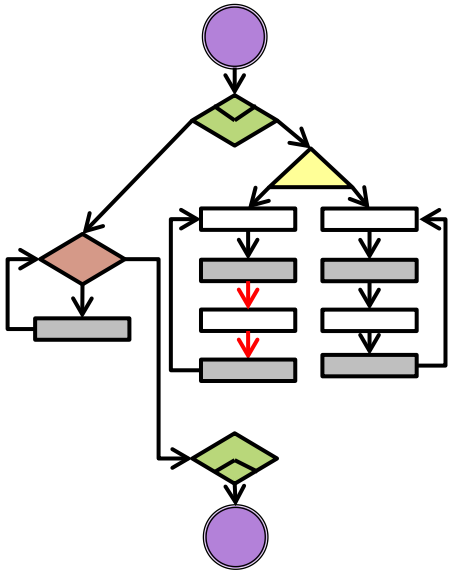
Unit time for 25MHz

An execution path with mix frequencies.

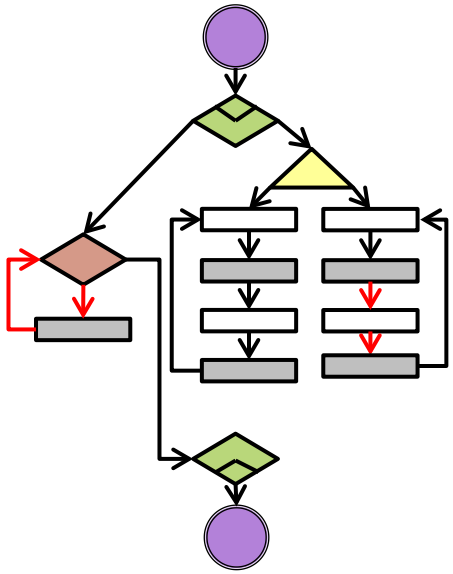
Extending ILPc



Unit time for 100MHz



Unit time for 50MHz

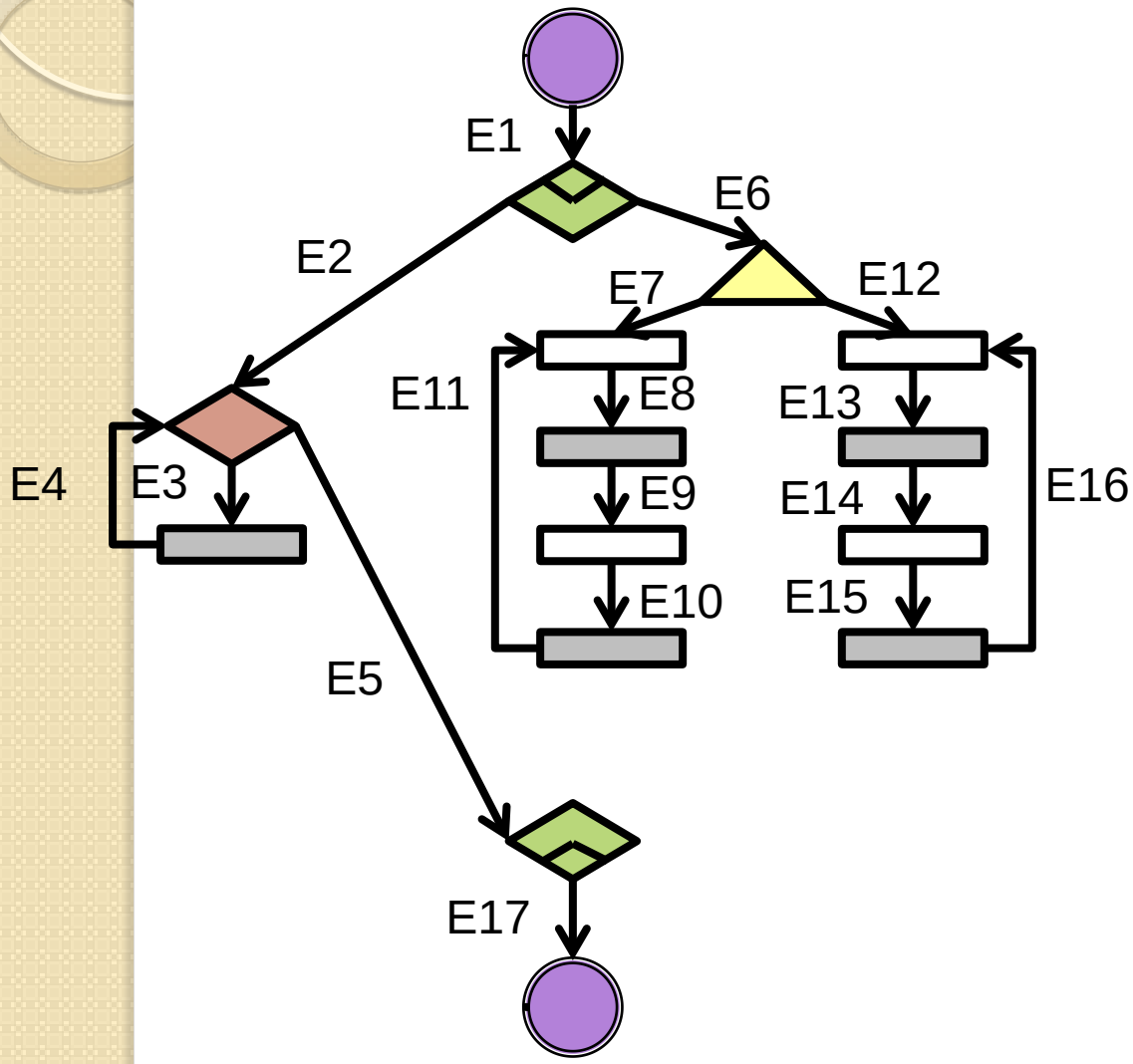


Unit time for 25MHz

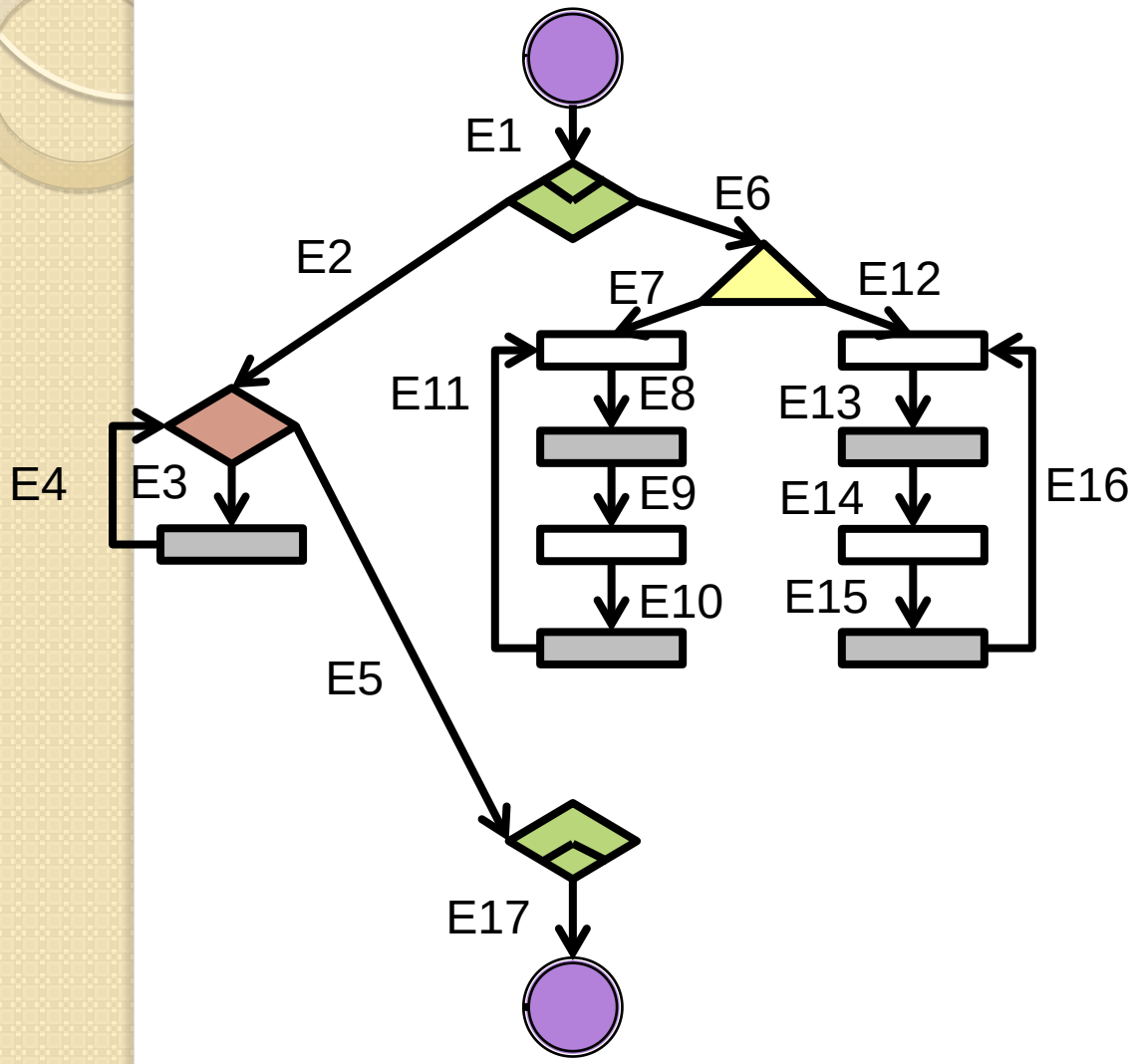
An execution path with mix frequencies.

- WCRT = sum edges in all TCCFG
- Execution paths can only switch at EOT and join nodes

Extending ILPc



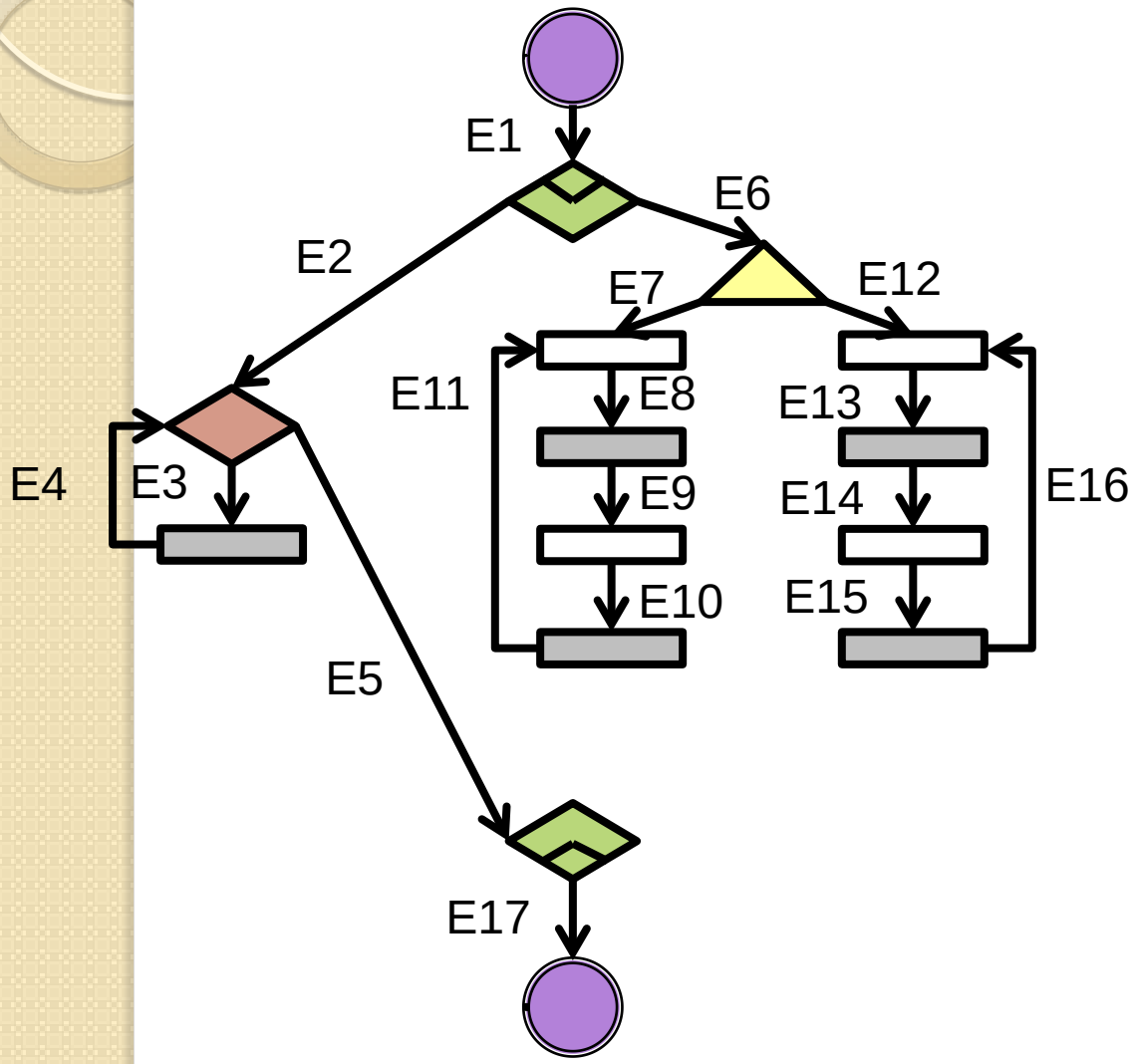
Extending ILPc



Based on ILPc with extra constraints:

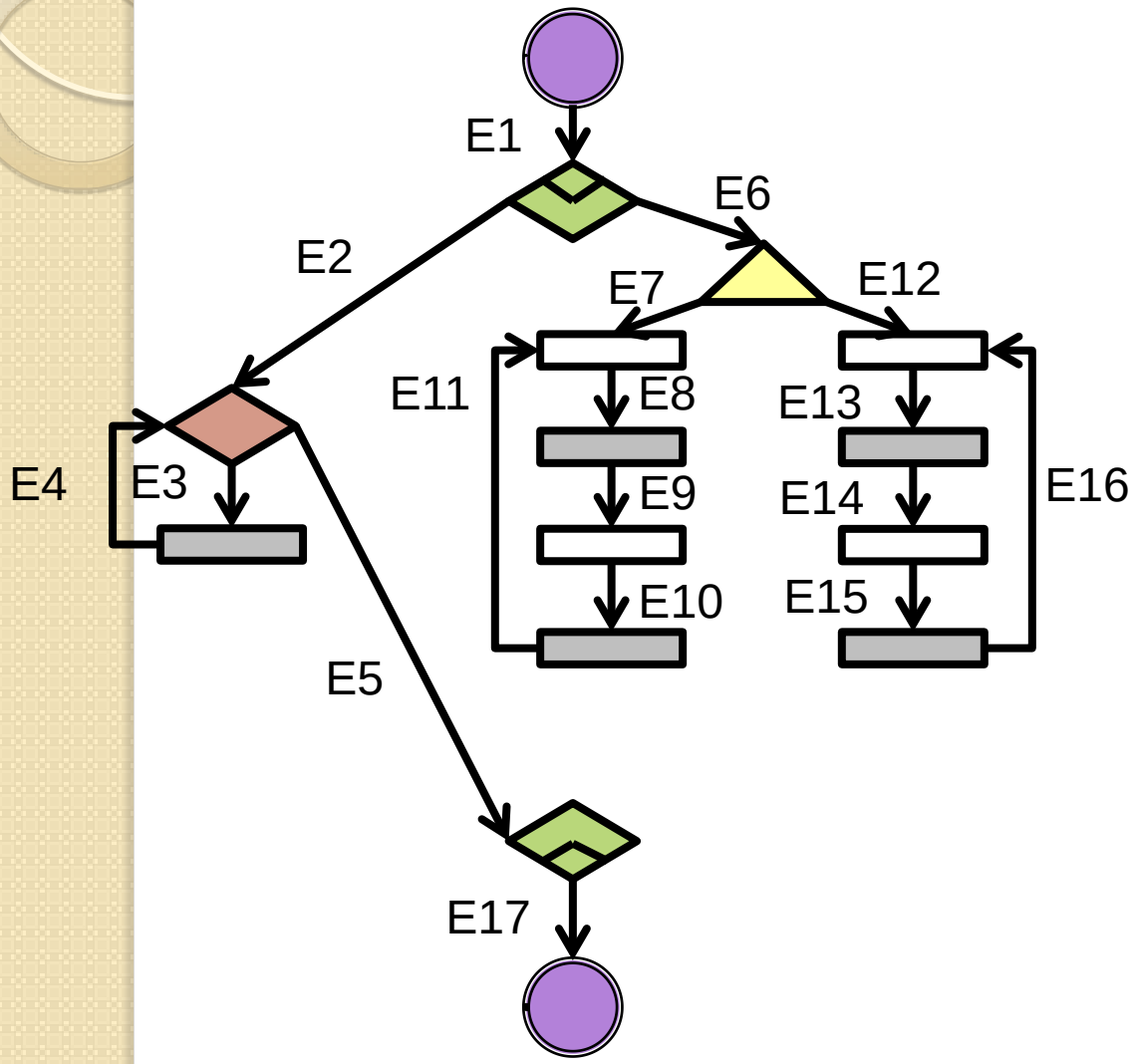
1. Create constraints for each TCCFG individually (ILPc).
2. Additional constraints across TCCFG to correctly capture execution paths.

Extending ILPc



Objective function:

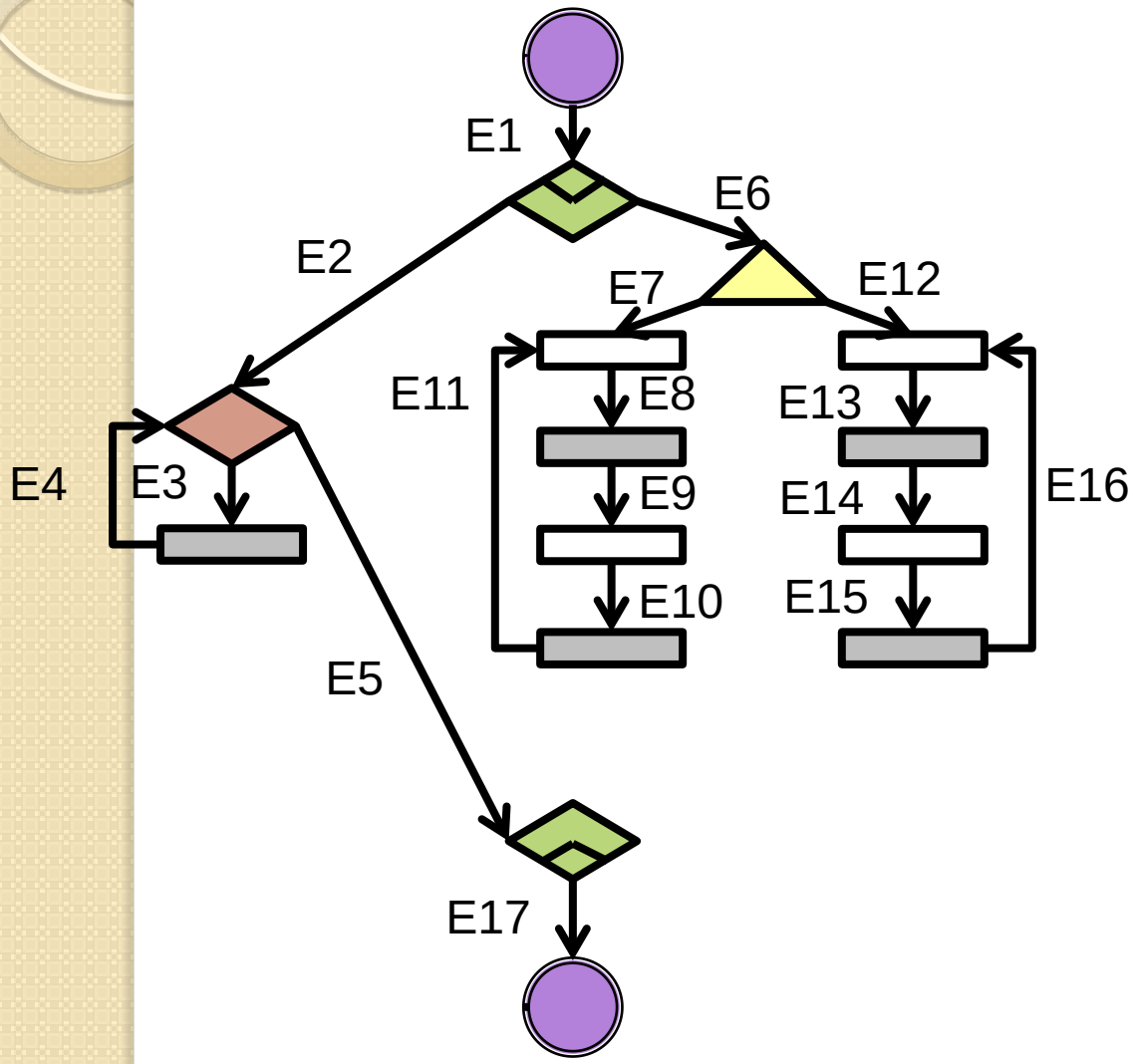
Extending ILPc



Objective function:

$$obj = MAX(\sum C_i \quad E_i)$$

Extending ILPc

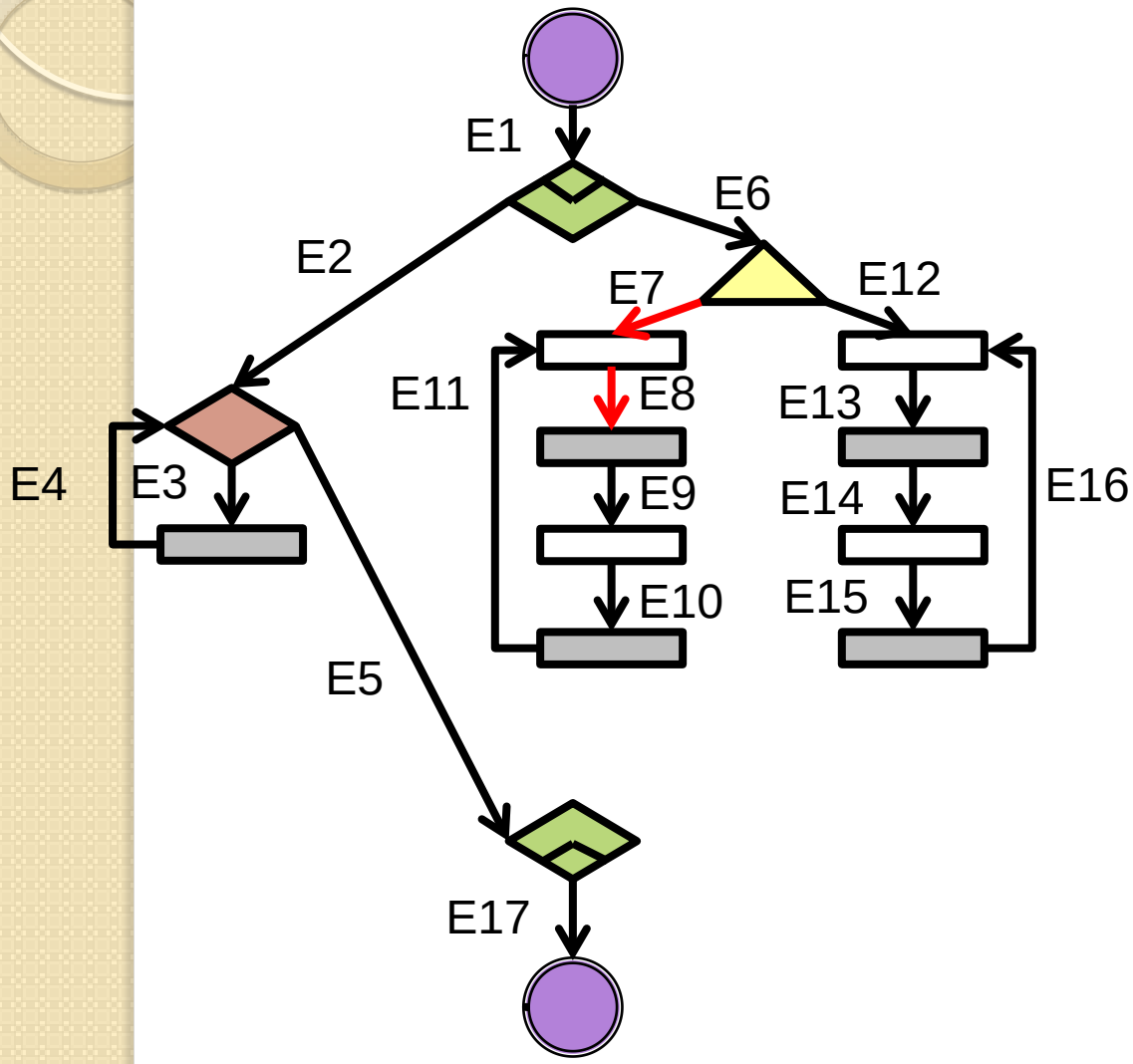


Objective function:

$$obj = MAX(\sum C_i \quad E_i)$$

- Create constraints based on each graph

Extending ILPc



Objective function:

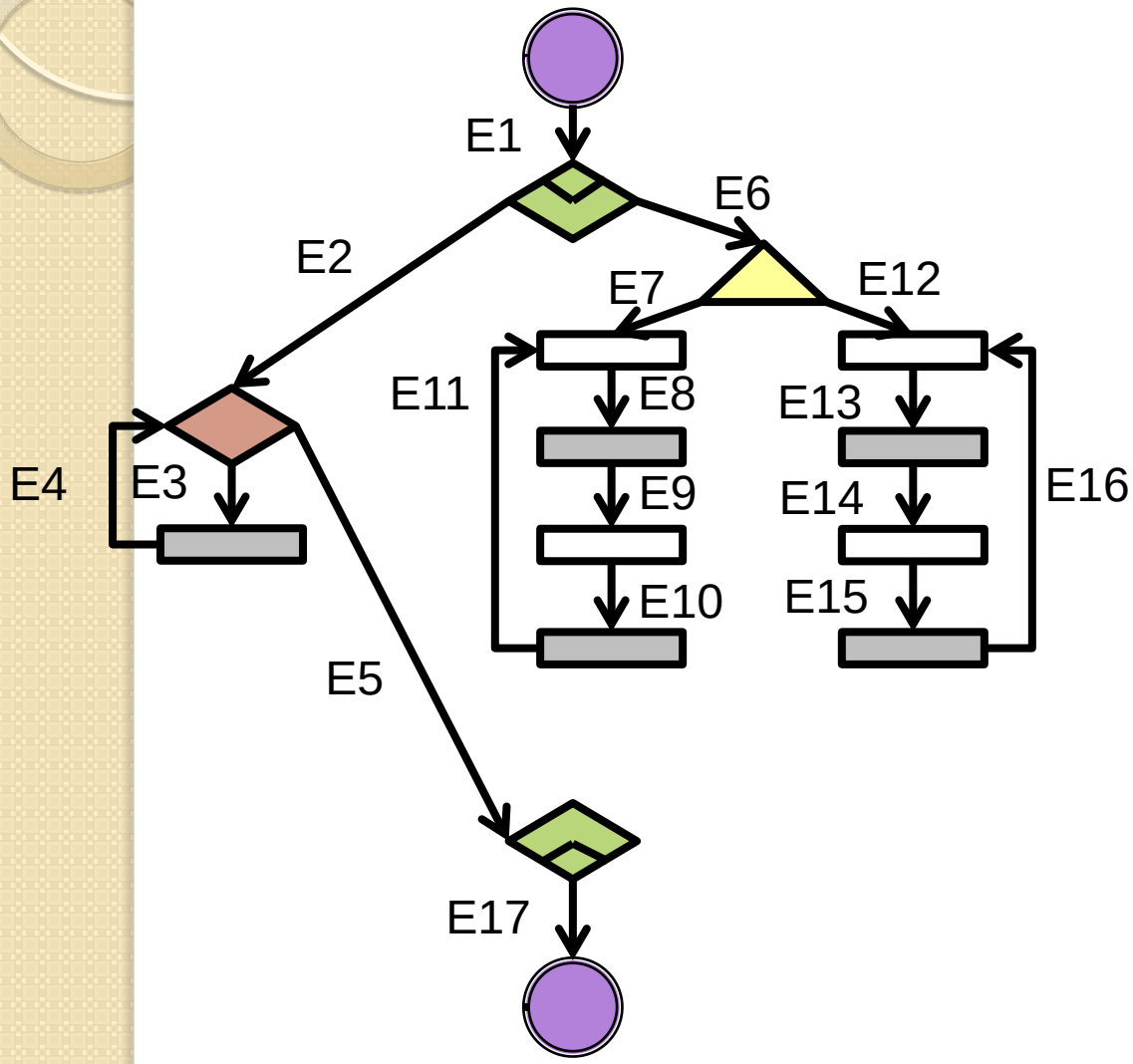
$$obj = MAX(\sum C_i \quad E_i)$$

- Create constraints based on each graph

E.g.,

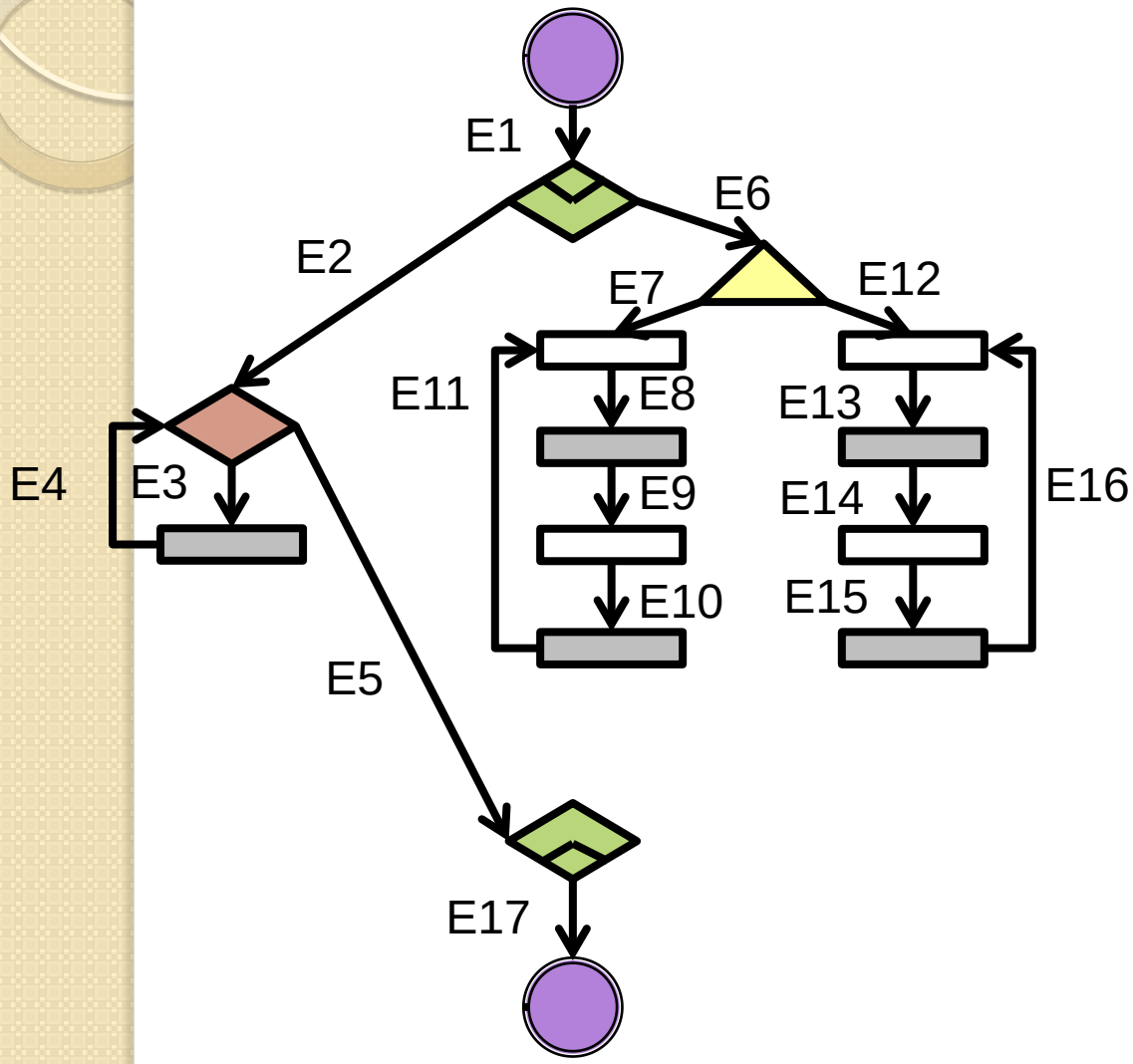
$$E7\_a = E8\_a$$
$$E7\_b = E8\_b$$
$$E7\_c = E8\_c$$

Extending ILPc



Constraints across the graphs:

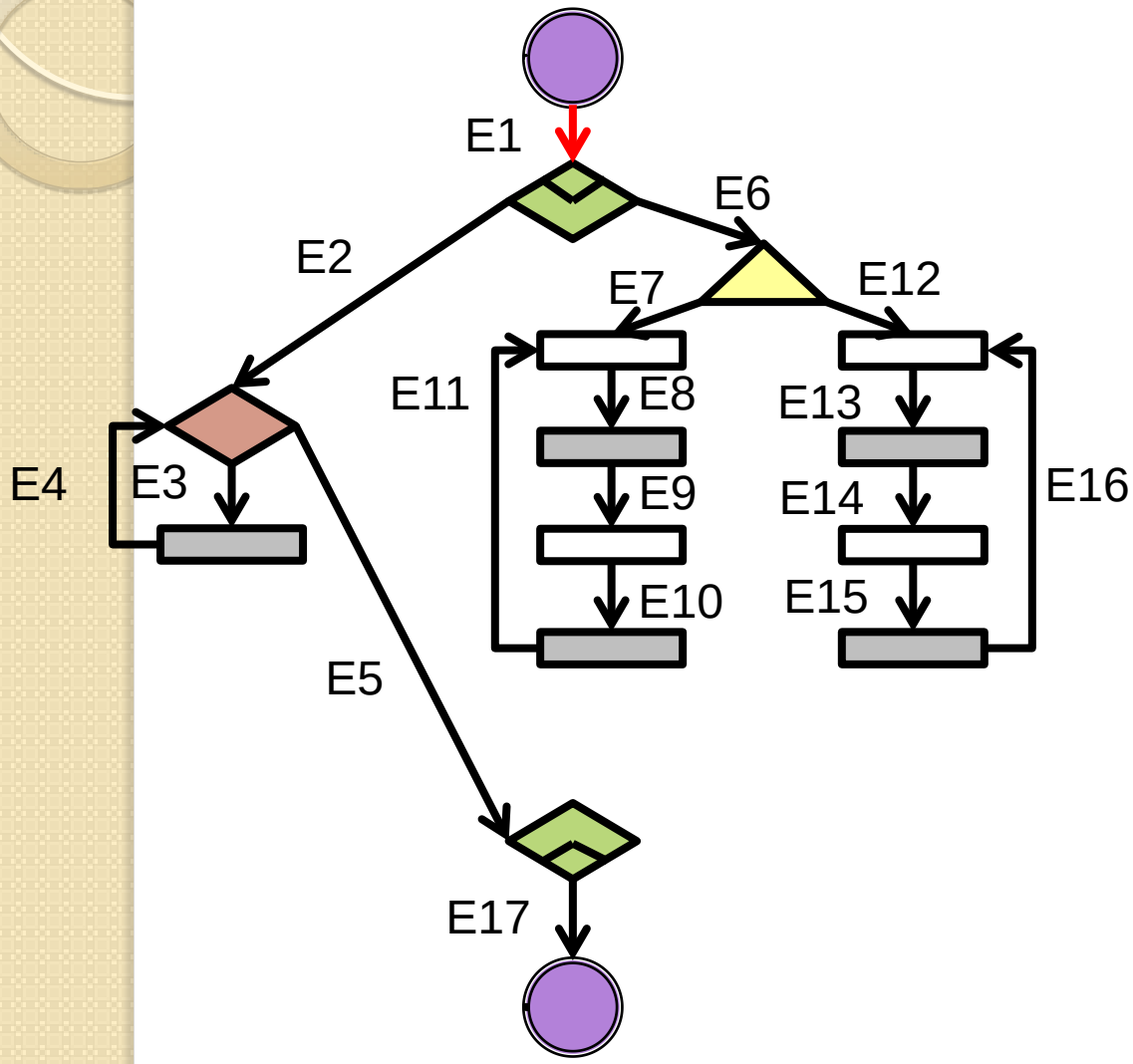
Extending ILPc



Constraints across the graphs:

$$E_{i\_a} + E_{i\_b} + E_{i\_c} \leq 1$$

Extending ILPc



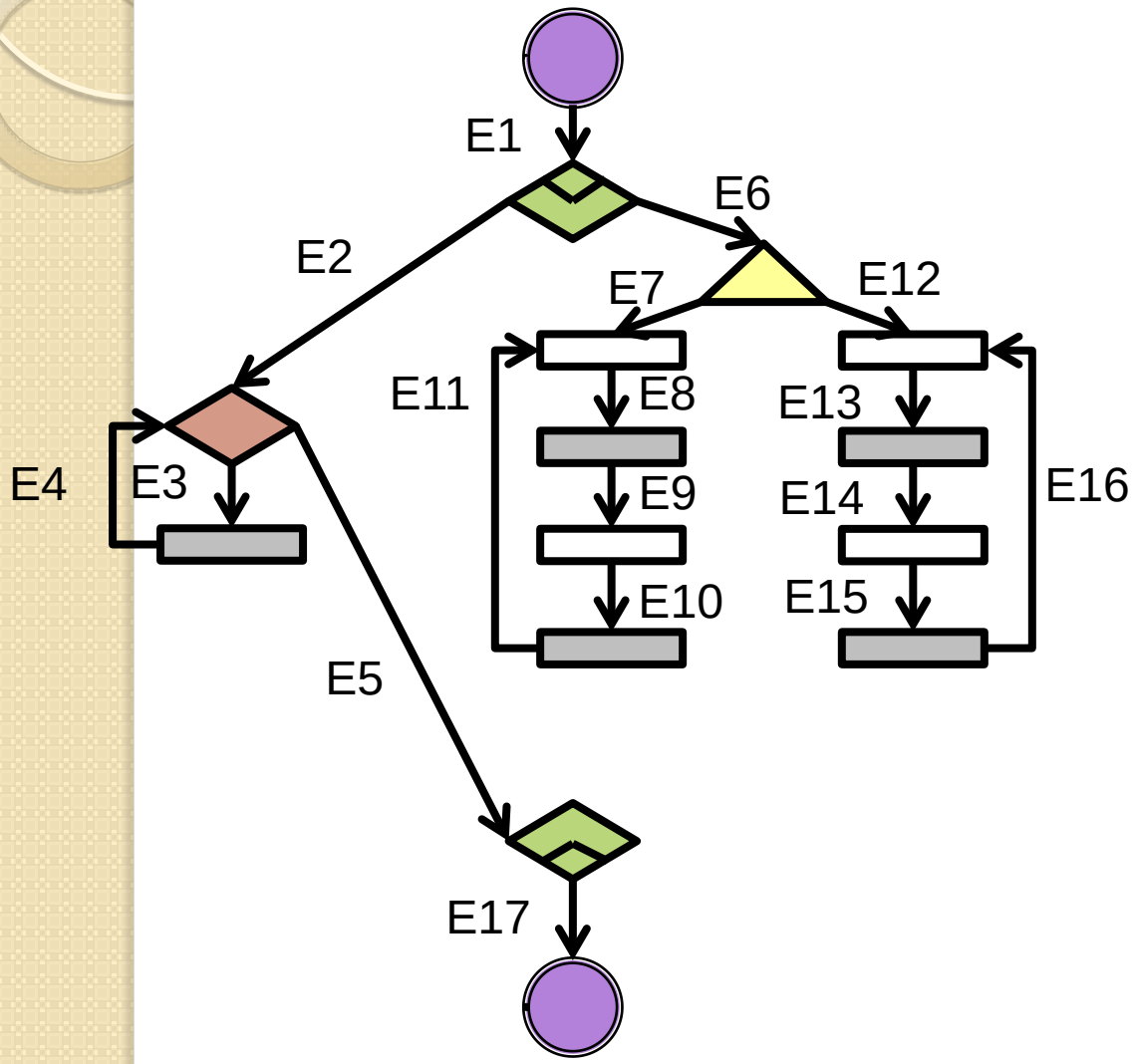
Constraints across the graphs:

$$E_{i\_a} + E_{i\_b} + E_{i\_c} \leq 1$$

E.g.,

$$E1\_a + E1\_b + E1\_c \leq 1$$

Extending ILPc



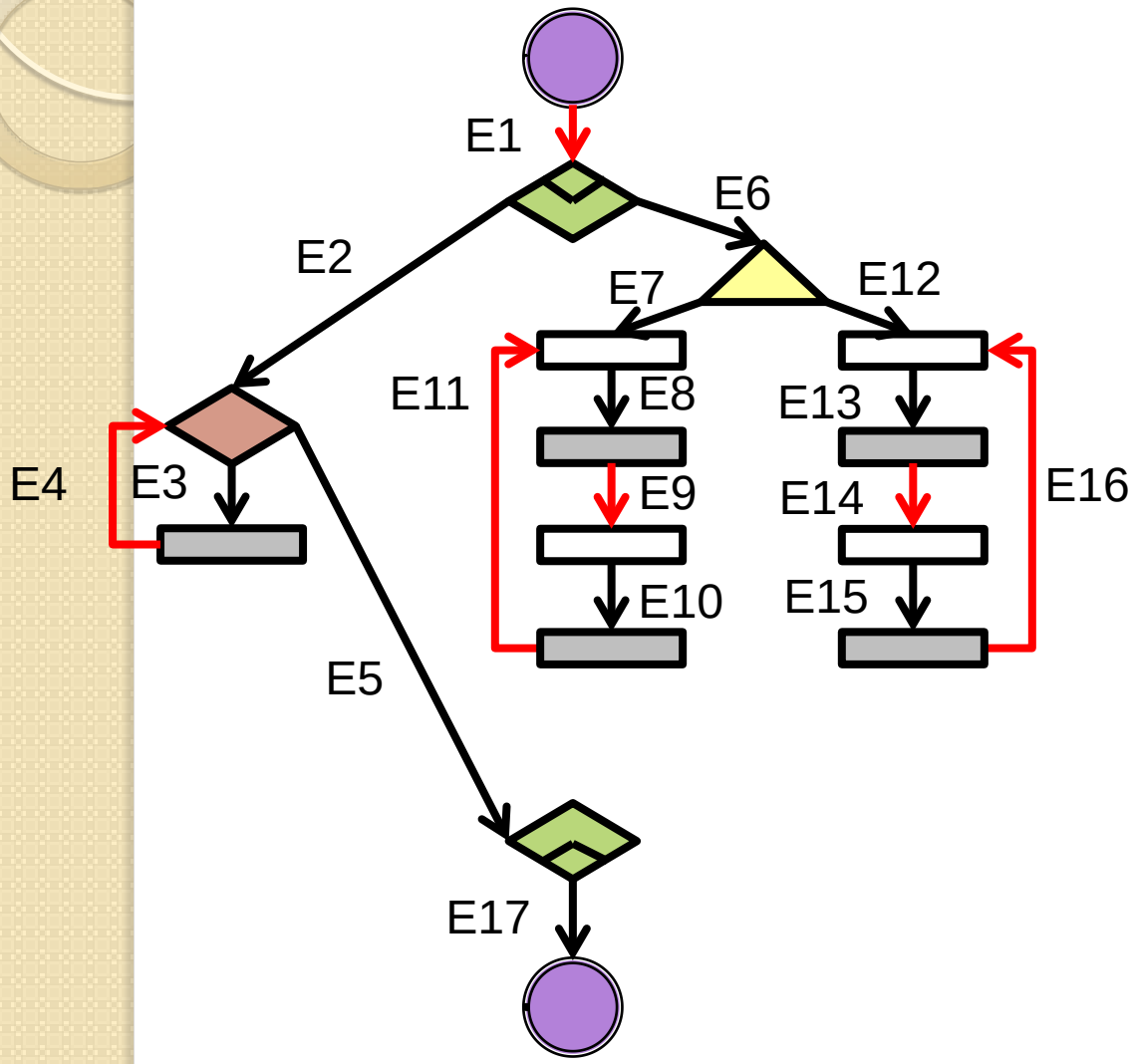
Constraints across the graphs:

$$E_{i\_a} + E_{i\_b} + E_{i\_c} \leq 1$$

E.g.,

$$E1\_a + E1\_b + E1\_c \leq 1$$

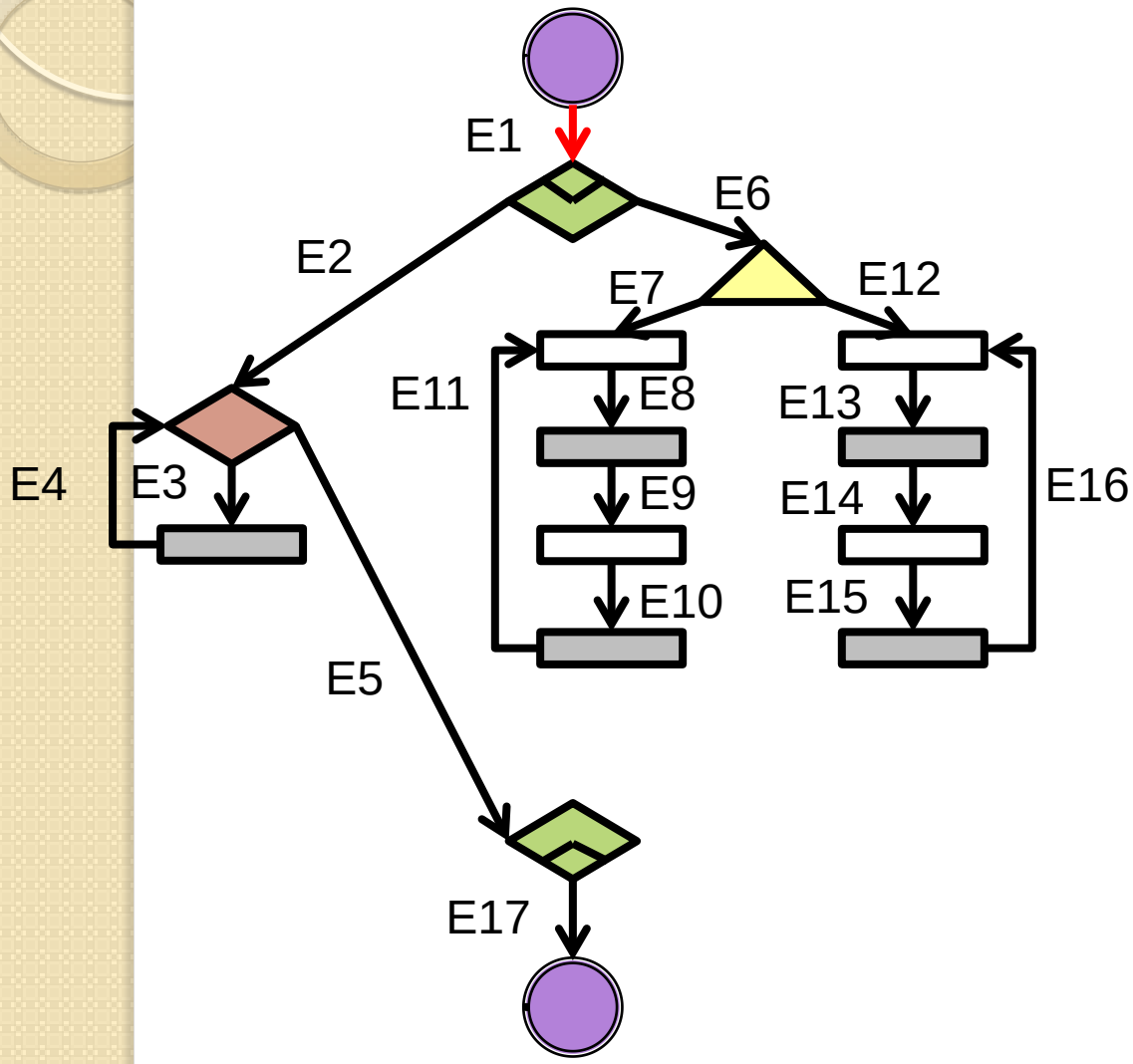
Extending ILPc



Constraints for EOT, join and preemption:

- These constraints uses EOT edges

Extending ILPc



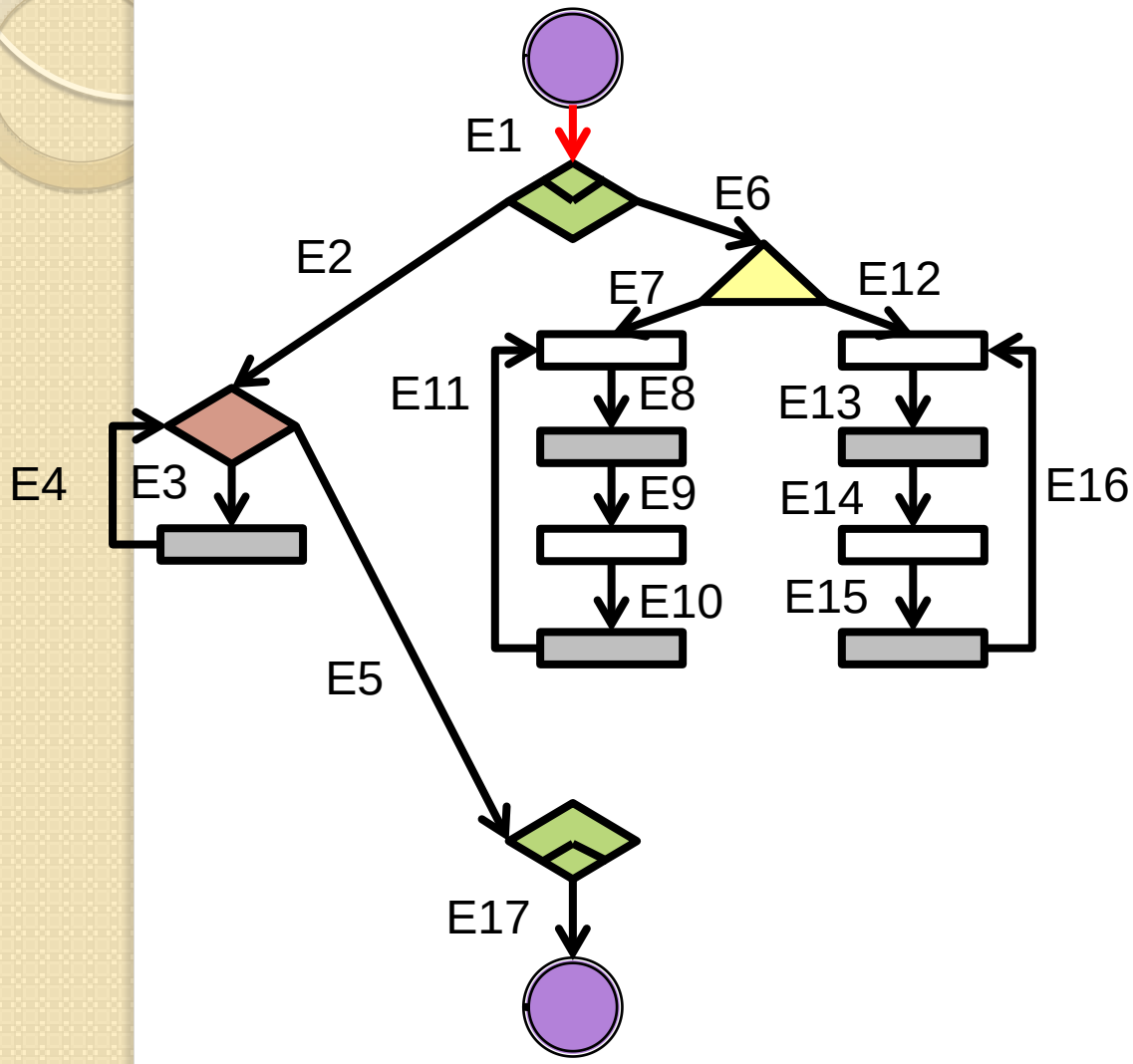
Constraints for EOT, join and preemption:

- These constraints uses EOT edges
- Each edge has multiple corresponding variables

E.g.,

E1_a, E1_b and E1_c

Extending ILPc



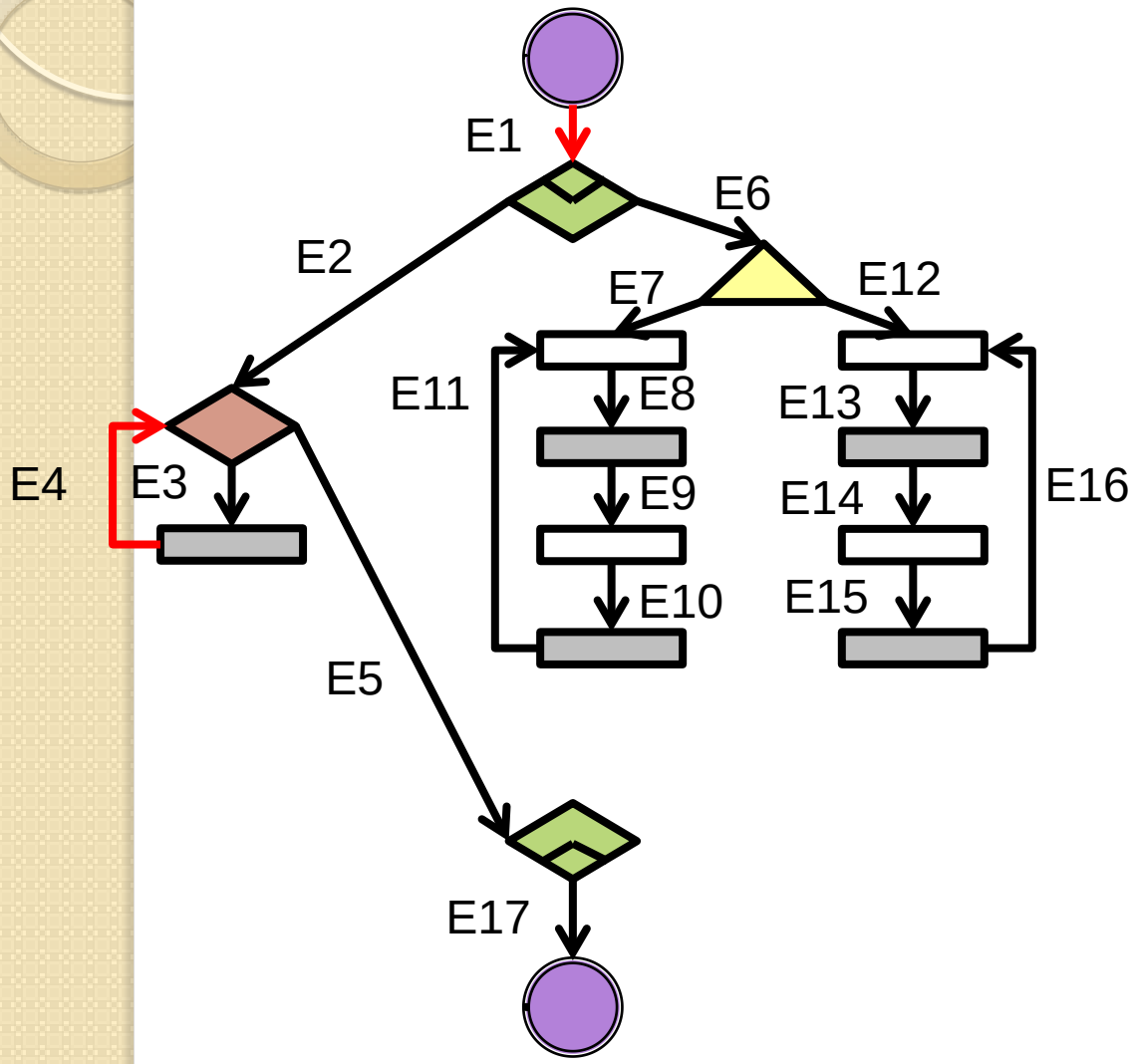
Constraints for EOT, join and preemption:

- These constraints uses EOT edges
- Each edge has multiple corresponding variables

E.g.,
E1_a, E1_b and E1_c

- To mimic the original constraint, we use the sum of these variables in the formulation

Extending ILPc

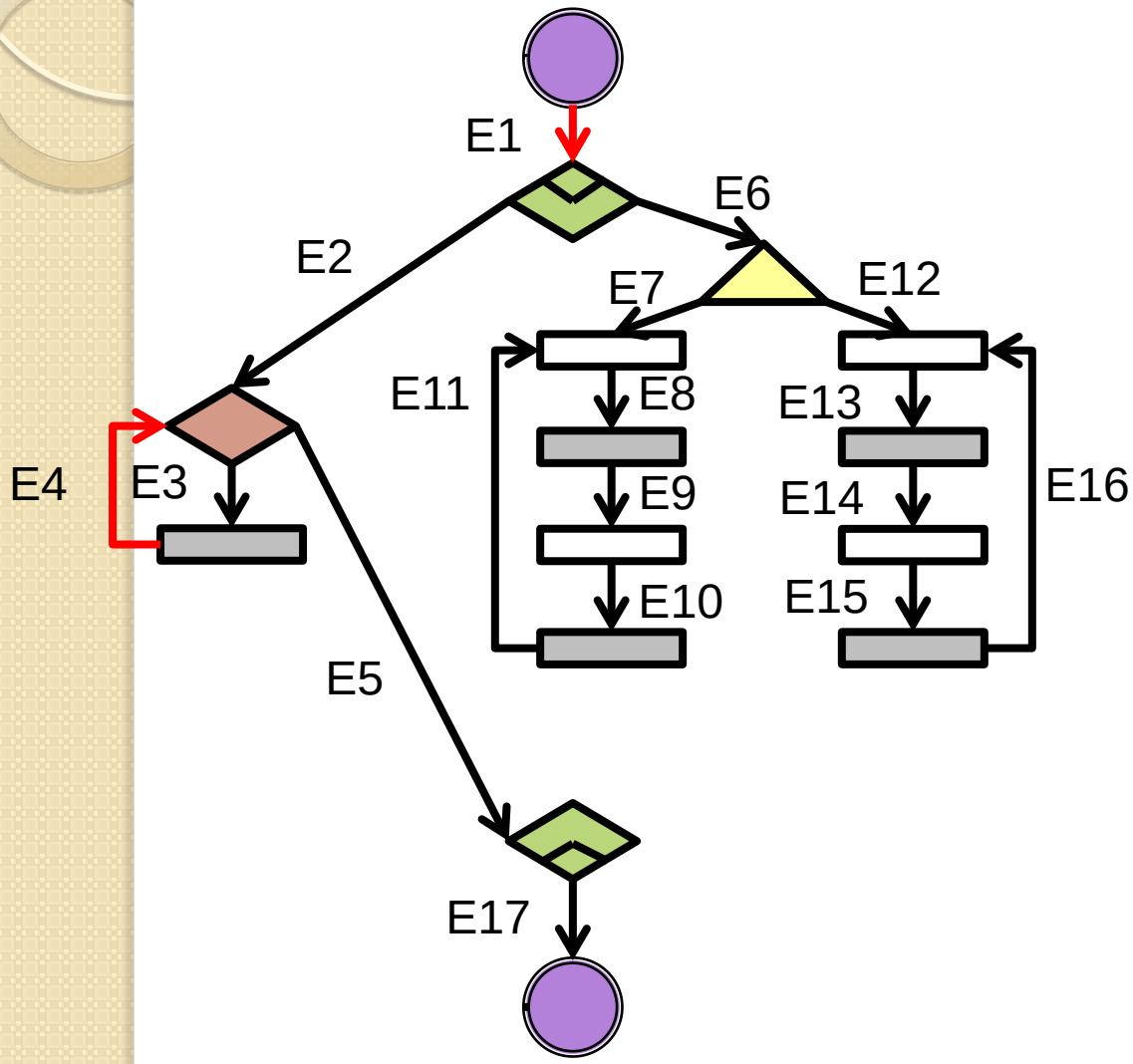


For Example:

Original EOT constraint:
E1 + E4 ≤ 1

- Parent thread and child threads cannot have active EOT edges together.
- Either ***E1*** or ***E4*** can be ‘1’

Extending ILPc



For Example:

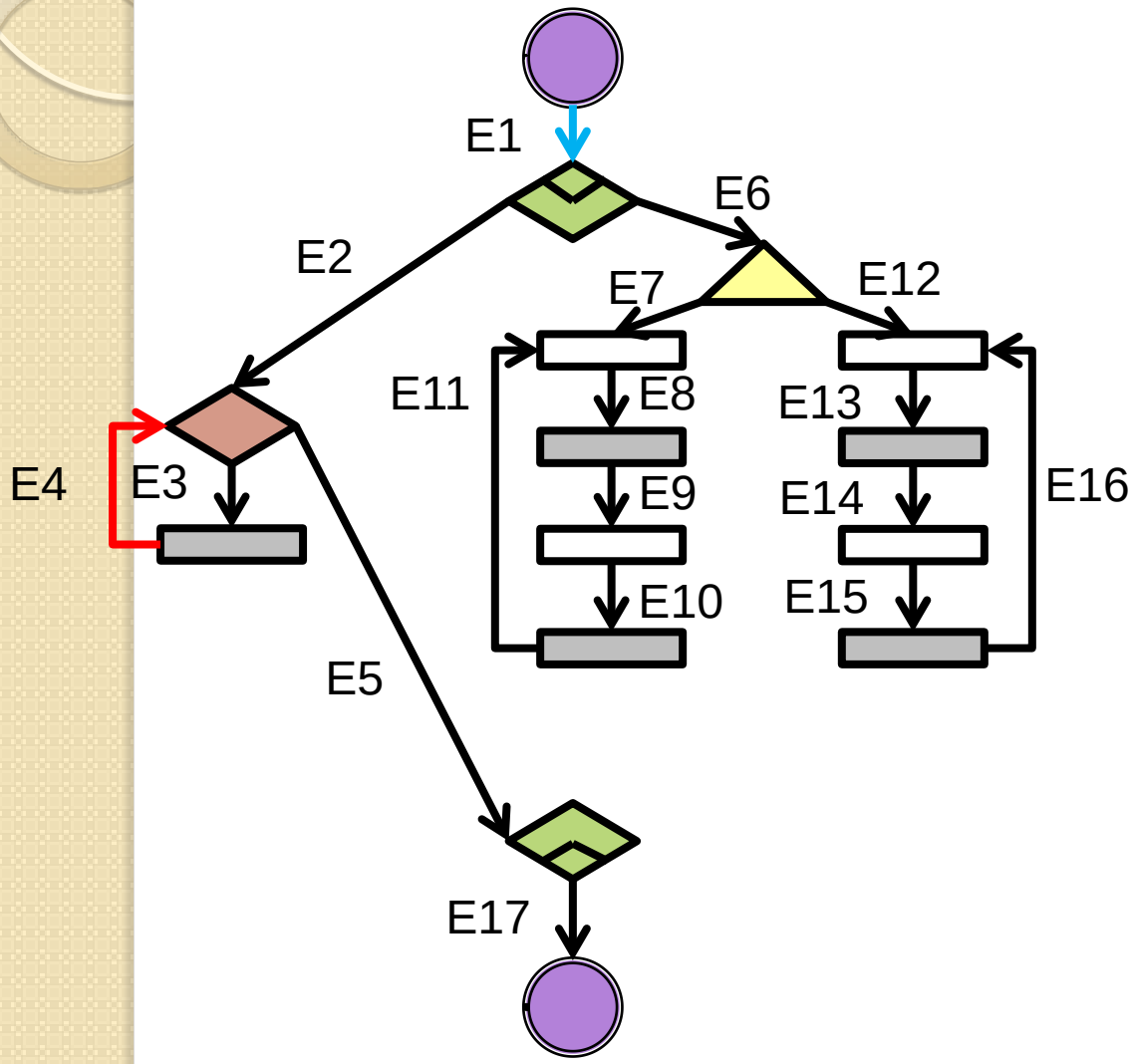
Original EOT constraint:

E1 + E4 ≤ 1

Transformed:

E1_a + E1_b + E1_c +
E4_a + E4_b + E4_c ≤ 1

Extending ILPc



For Example:

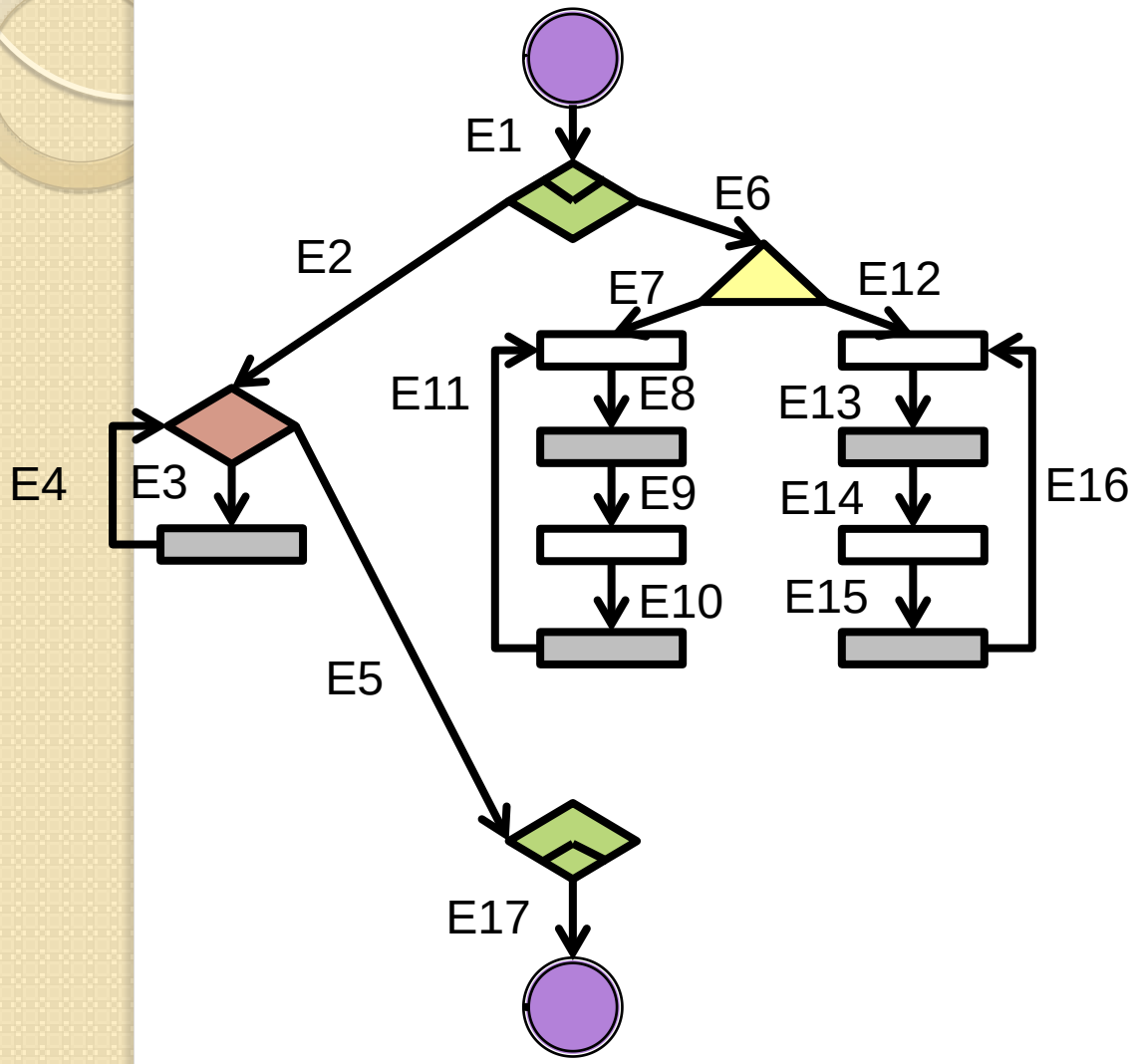
Original EOT constraint:

$E1 + E4 \leq 1$

Transformed:

$E1\_a + E1\_b + E1\_c +$
$E4\_a + E4\_b + E4\_c \leq 1$

Extending ILPc



For Example:

Original EOT constraint:
E1 + E4 ≤ 1

Transformed:
E1_a + E1_b + E1_c +
E4_a + E4_b + E4_c ≤ 1

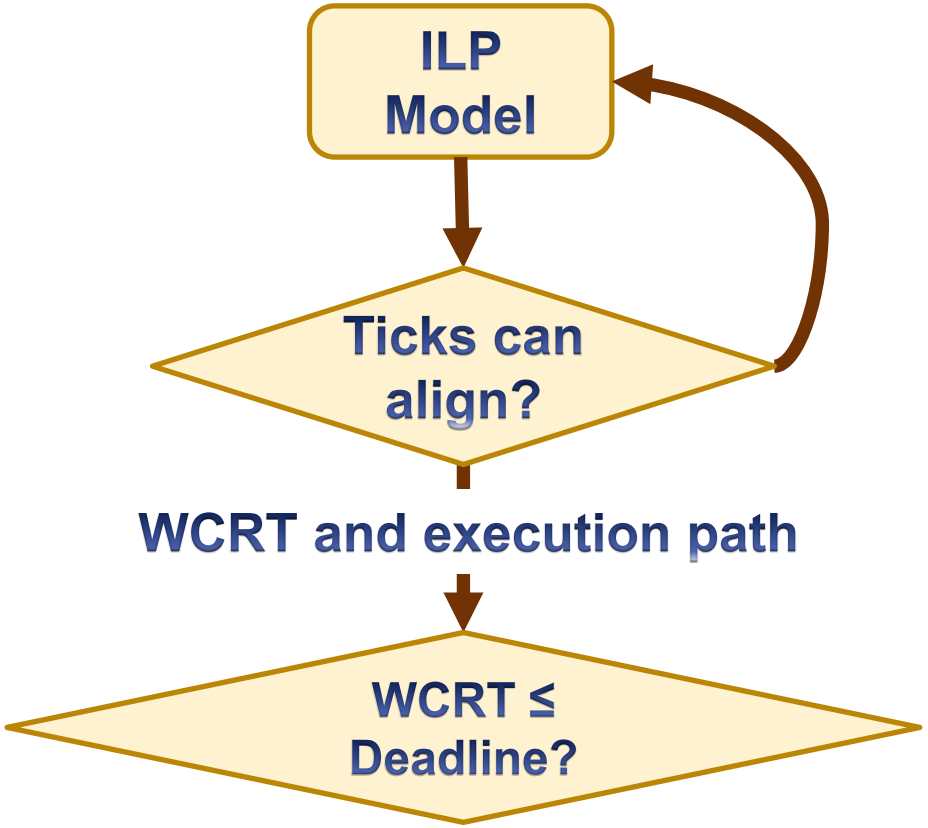
- These will allow the constraint to work across graphics
- Equivalent to the original ILP constraint

E.g.,
If ***E4_c*** is '1', all of ***E1_a***, ***E1_b*** and ***E1_c*** must be '0'

Outline

- Background
- Problem statement
  - Existing works
  - Proposal
- Methodology
  - Extending base ILP model
  - DVFS analysis component
- Results
- Conclusions

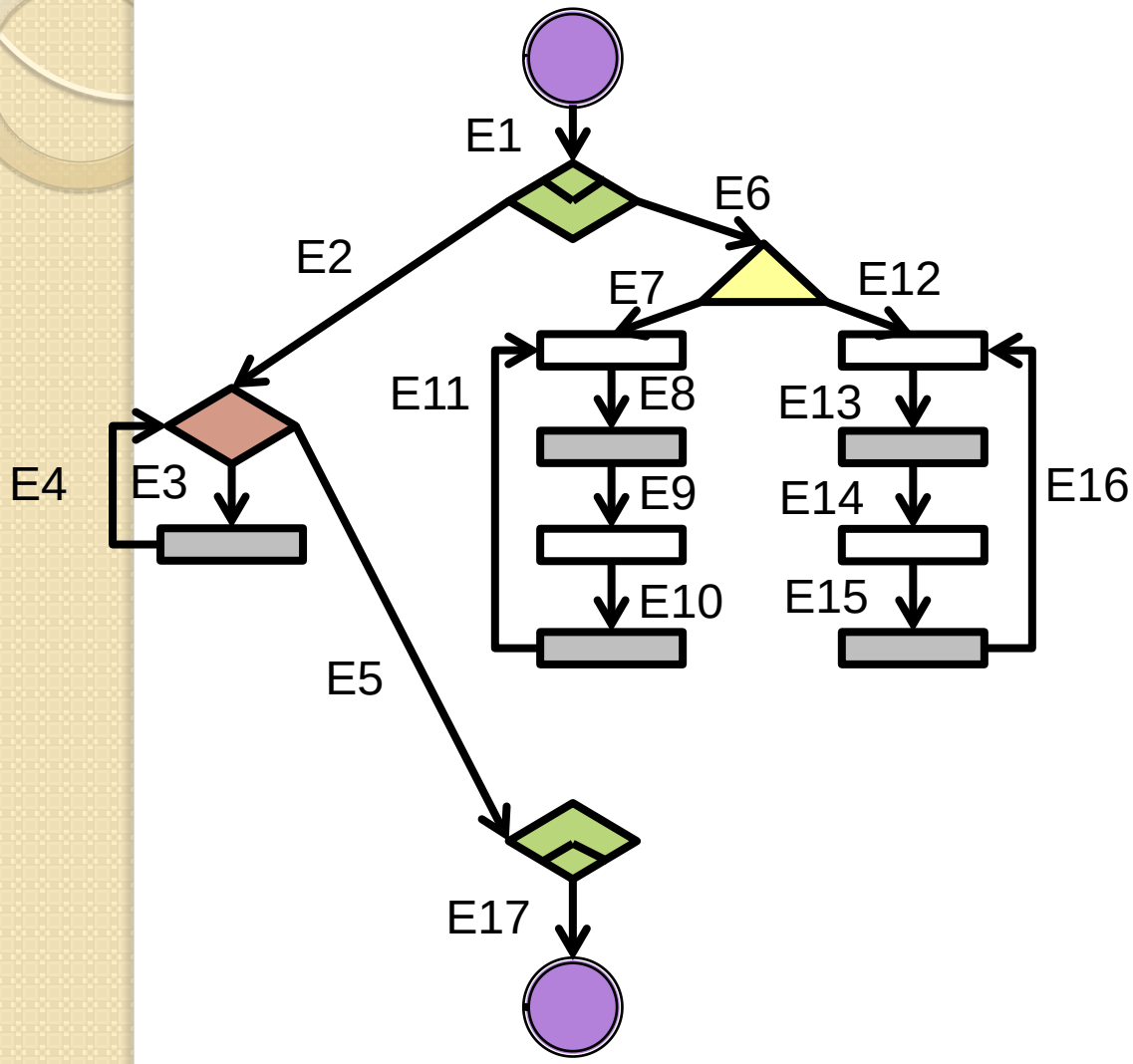
Overview


ILP
Model
Ticks can
align?
WCRT and execution path
WCRT ≤
Deadline?

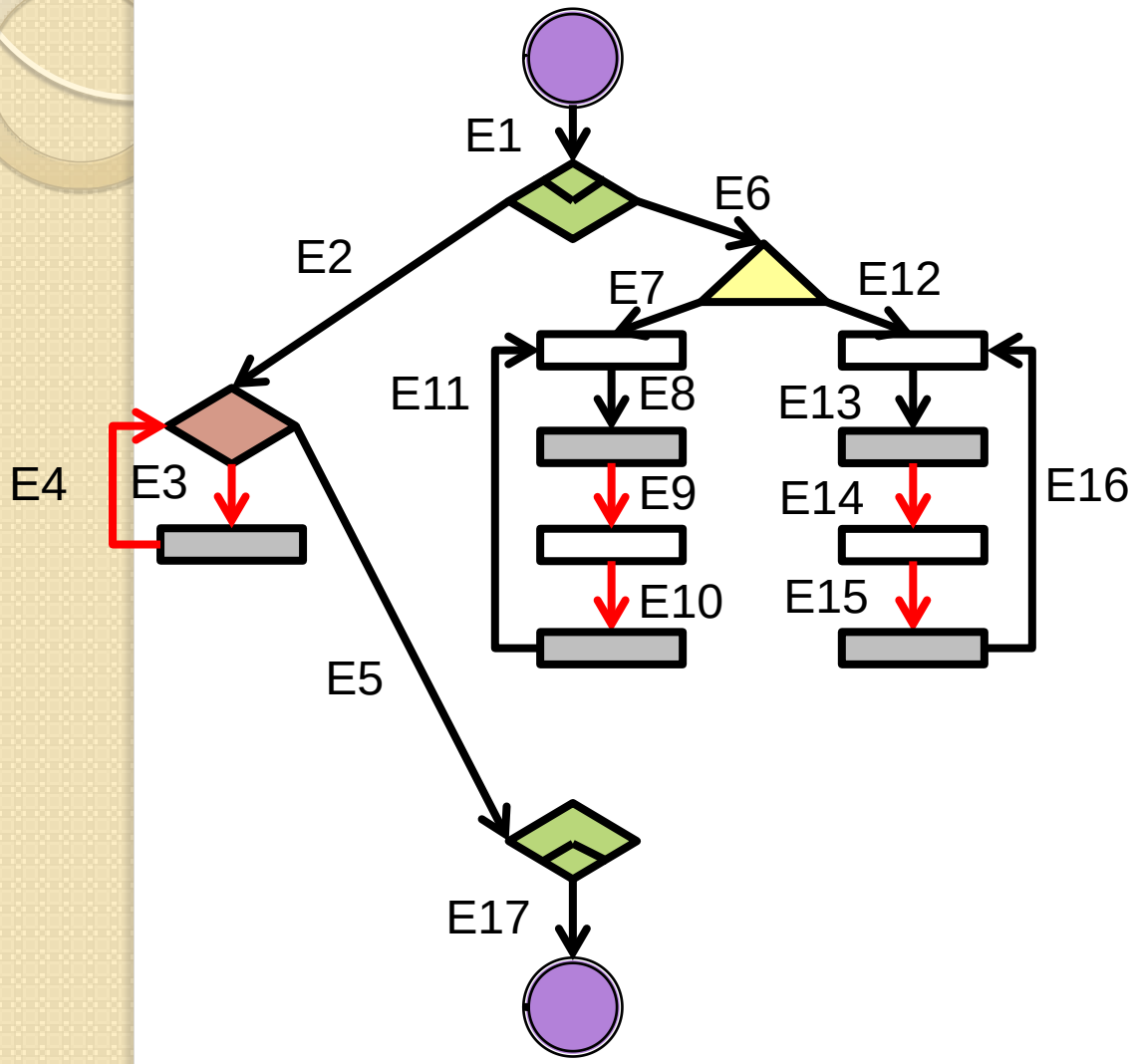
DVFS analysis

- Find the DVSF point that contribute the most toward the WCRT.
  - EOT and join nodes.
- Increase its frequency.

DVFS analysis



DVFS analysis



Execution path:

E4_c, E3_c

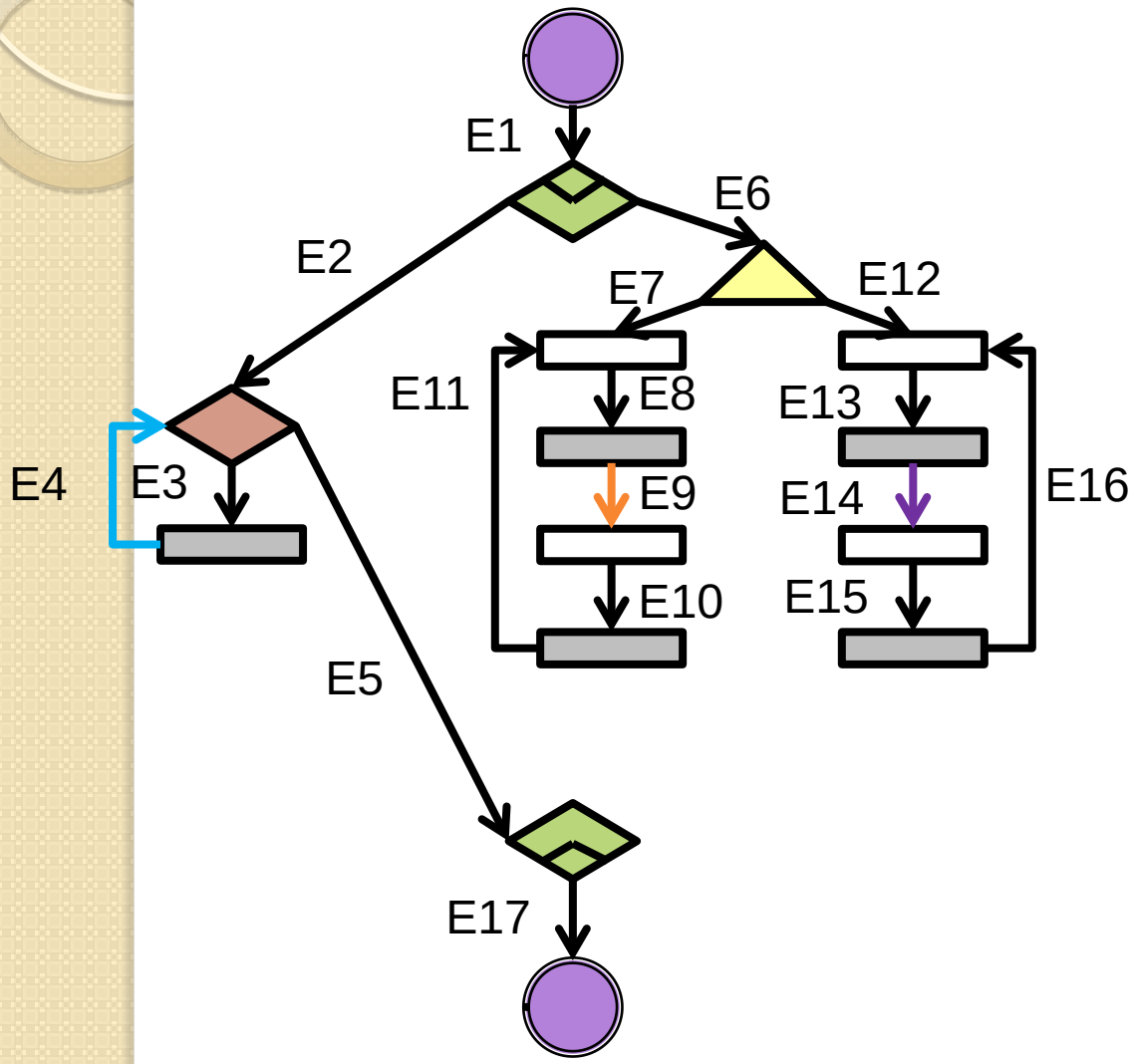
||

E9_c, E10_c

||

E14_c, E15_c

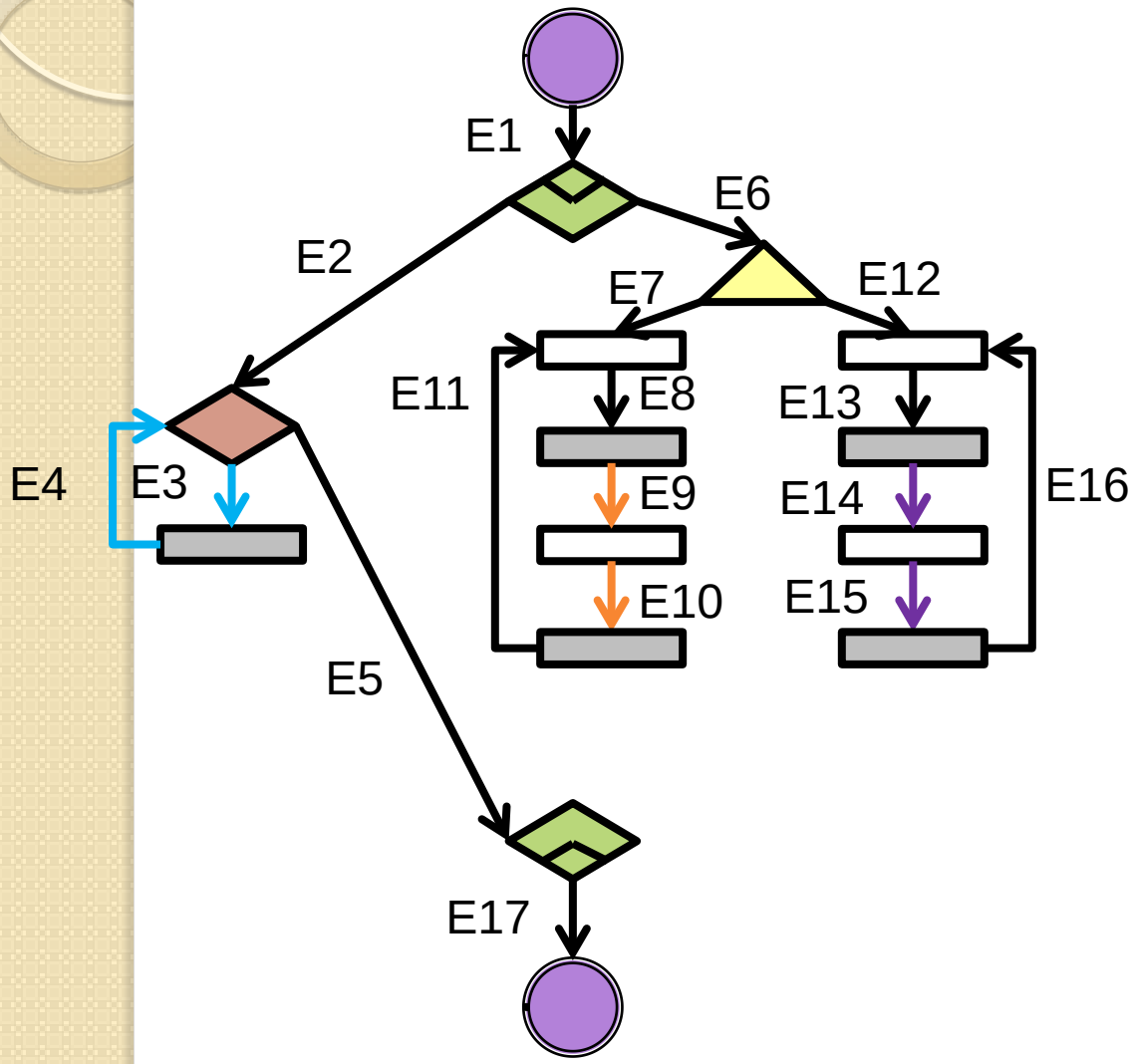
DVFS analysis



Execution path:
E4_c*, *E3_c
||
E9_c*, *E10_c
||
E14_c*, *E15_c

- 3 EOT edges

DVFS analysis



Execution path:

***E4_c*, *E3_c* (10)**

||

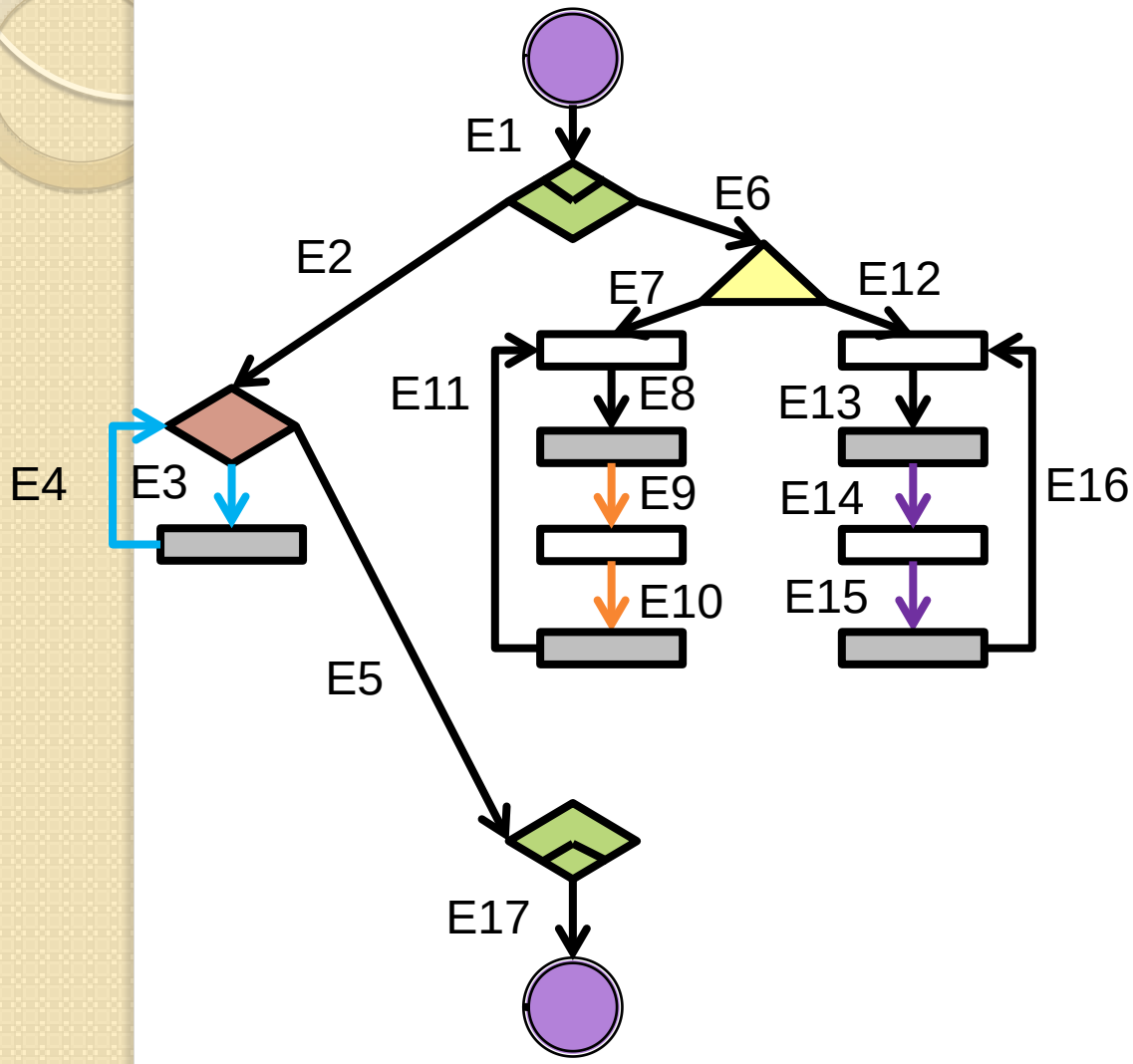
***E9_c*, *E10_c* (16)**

||

***E14_c*, *E15_c* (12)**

- 3 EOT edges
- Each contribute toward the WCRT

DVFS analysis



Execution path:

***E4_c*, *E3_c* (10)**

||

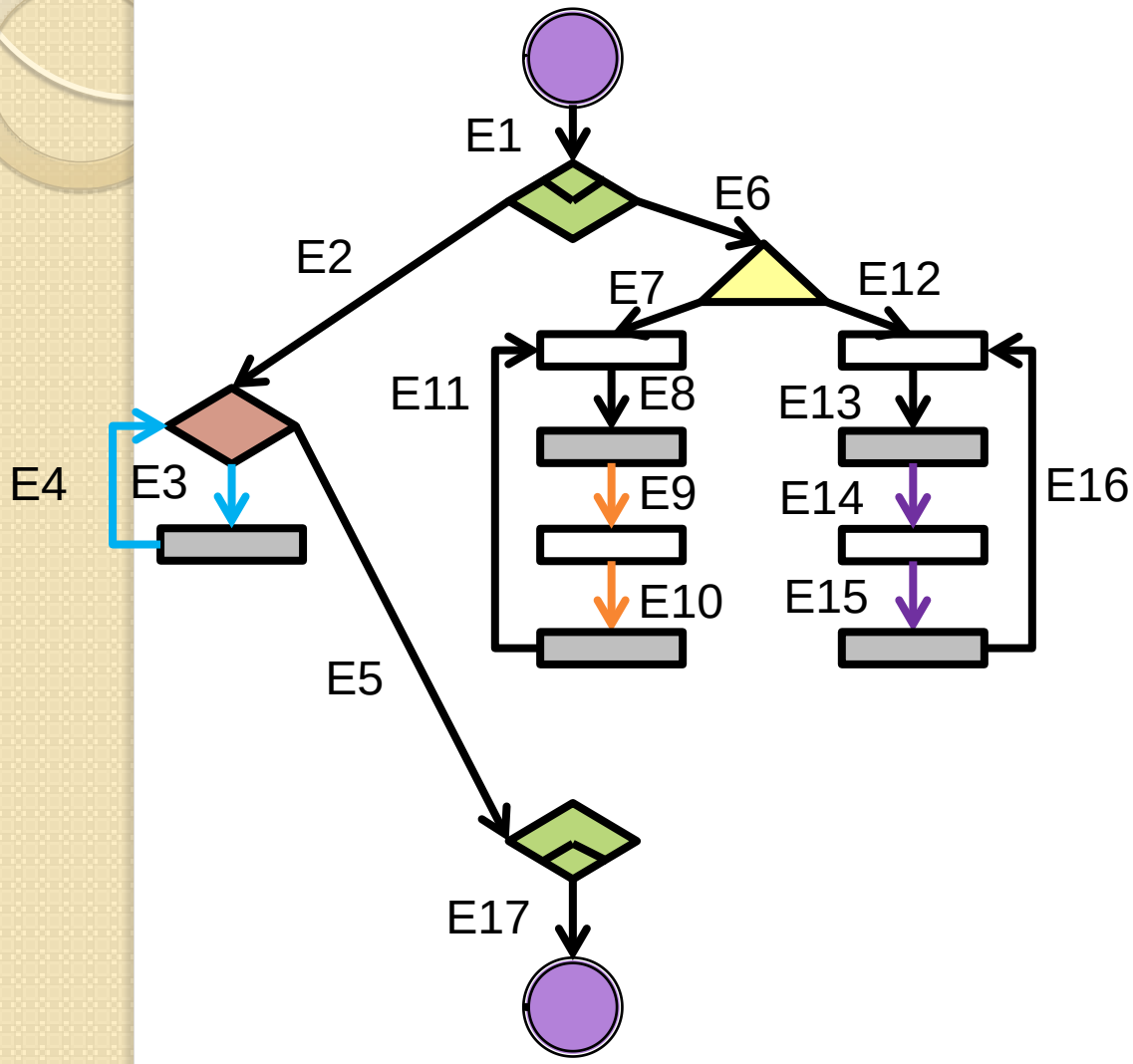
***E9_c*, *E10_c* (16)**

||

***E14_c*, *E15_c* (12)**

- 3 EOT edges
- Each contribute toward the WCRT
- Assume the deadline is 30 unit time

DVFS analysis



Execution path:

***E4_c*, *E3_c* (10)**

||

***E9_c*, *E10_c* (16)**

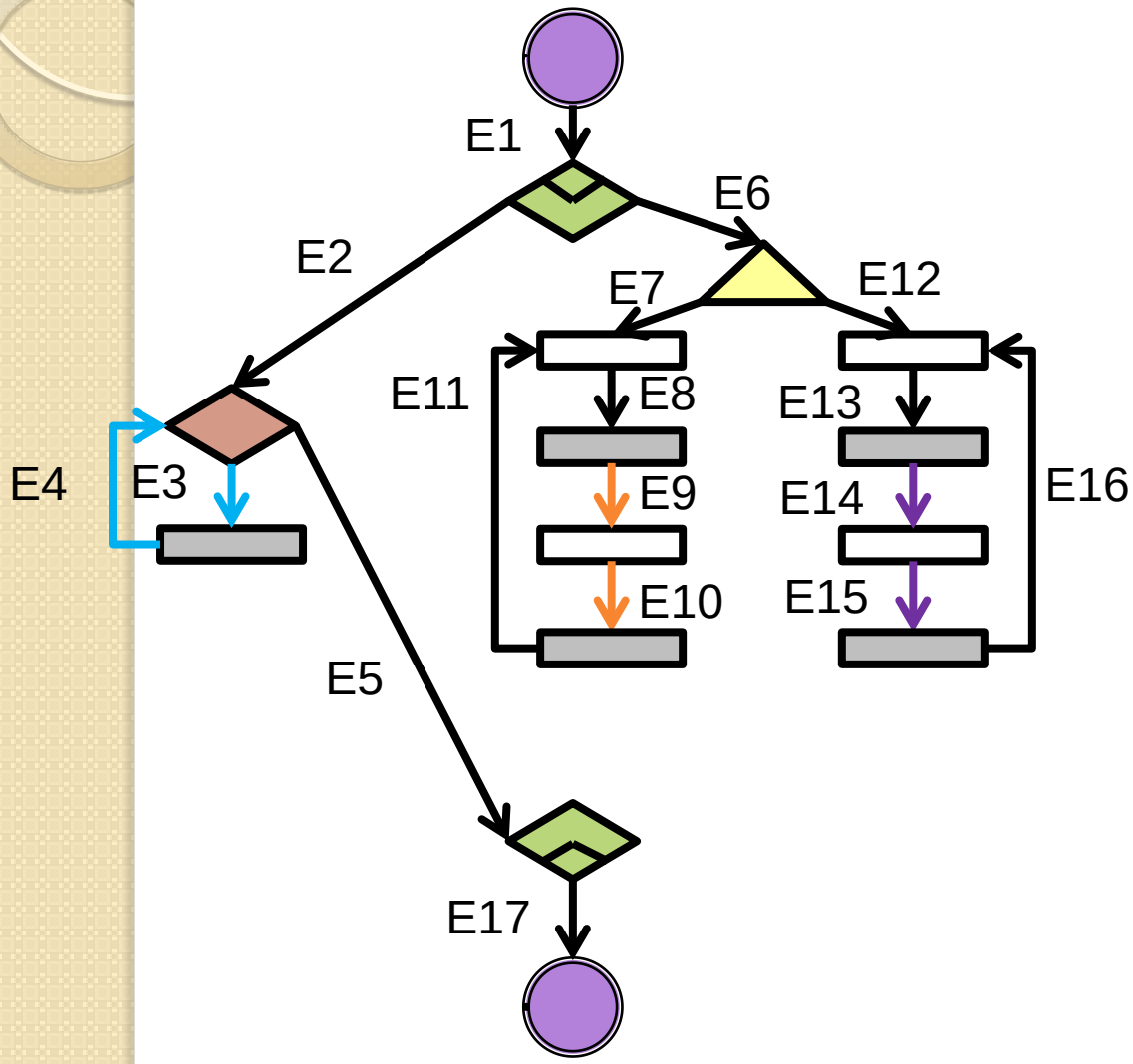
||

***E14_c*, *E15_c* (12)**

- 3 EOT edges
- Each contribute toward the WCRT
- Assume the deadline is 30 unit time

WCRT = 10+16+12 = 38

DVFS analysis



Execution path:

***E4_c*, *E3_c* (10)**

||

***E9_c*, *E10_c* (16)**

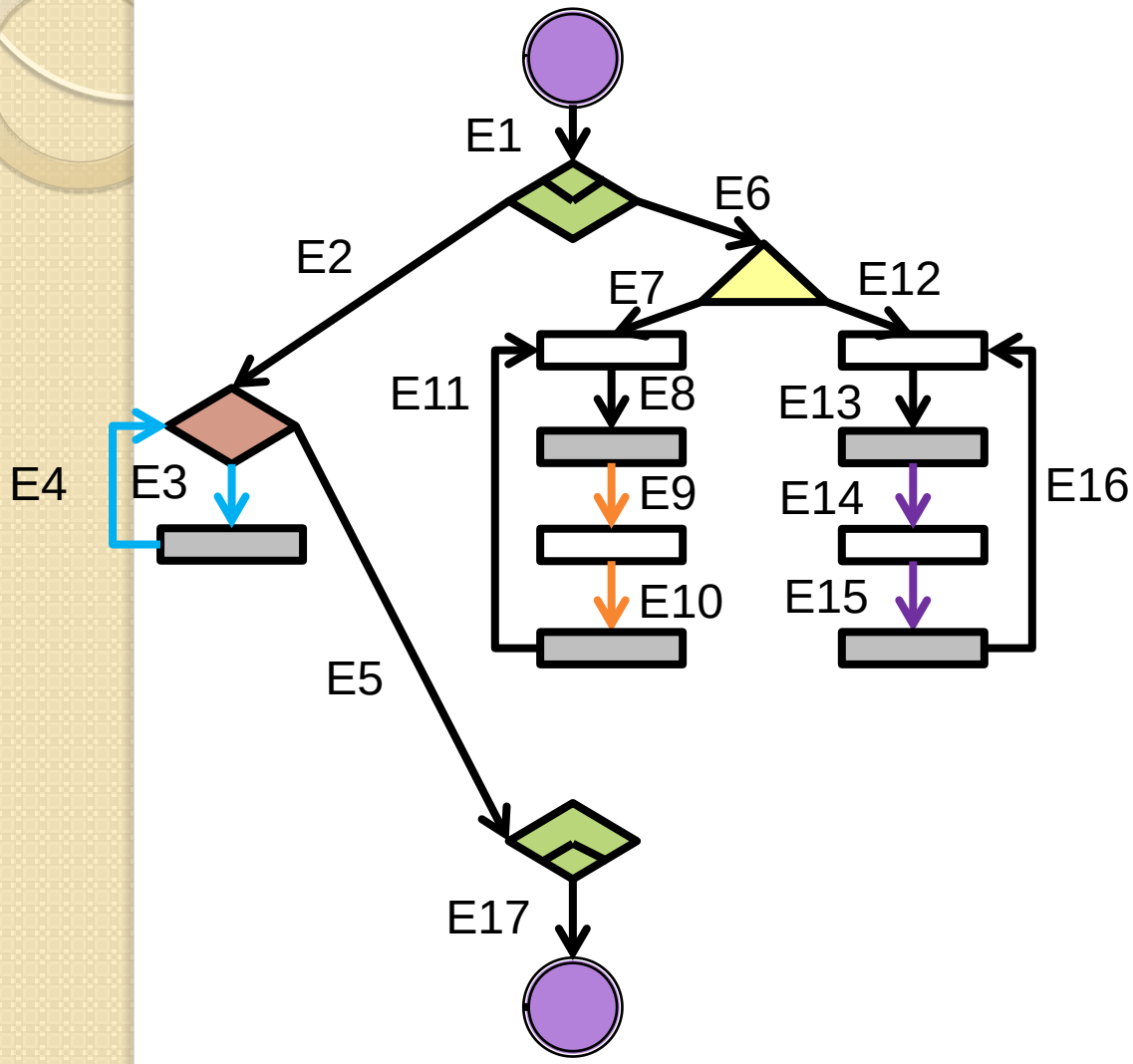
||

***E14_c*, *E15_c* (12)**

- 3 EOT edges
- Each contribute toward the WCRT
- Assume the deadline is 30 unit time
- Increase the frequency of ***E9*** by add the following constraint:

E9_c = 0

DVFS analysis



Execution path:

***E4_c*, *E3_c* (10)**

||

***E9_c*, *E10_c* (16)**

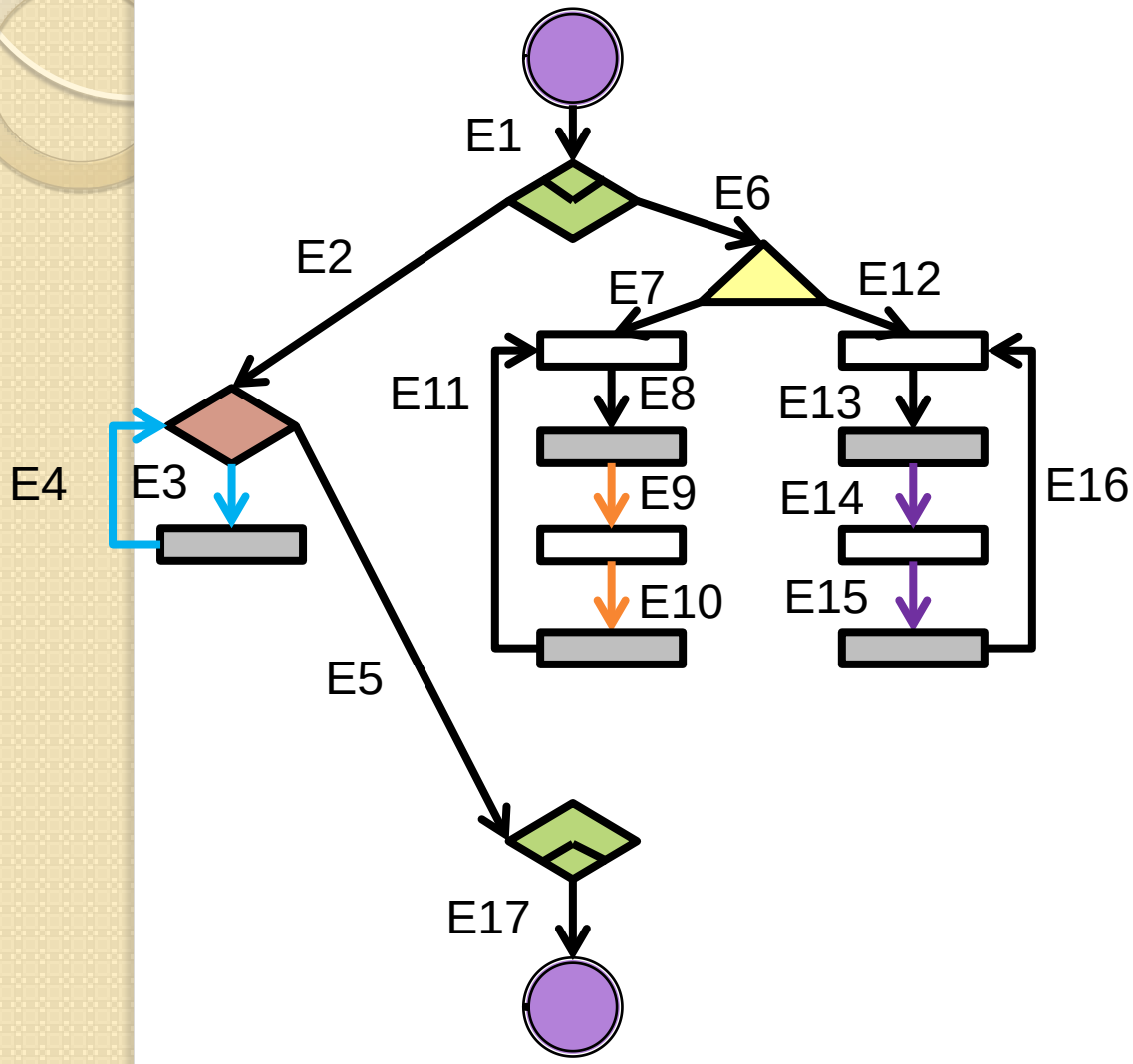
||

***E14_c*, *E15_c* (12)**

- 3 EOT edges
- Each contribute toward the WCRT
- Assume the deadline is 30 unit time
- Increase the frequency of ***E9*** by add the following constraint:

  E9_c = 0
- Force the solver to use a faster frequency

DVFS analysis



Execution path:

***E4_c*, *E3_c* (10)**

||

***E9_c*, *E10_c* (16)**

||

***E14_c*, *E15_c* (12)**

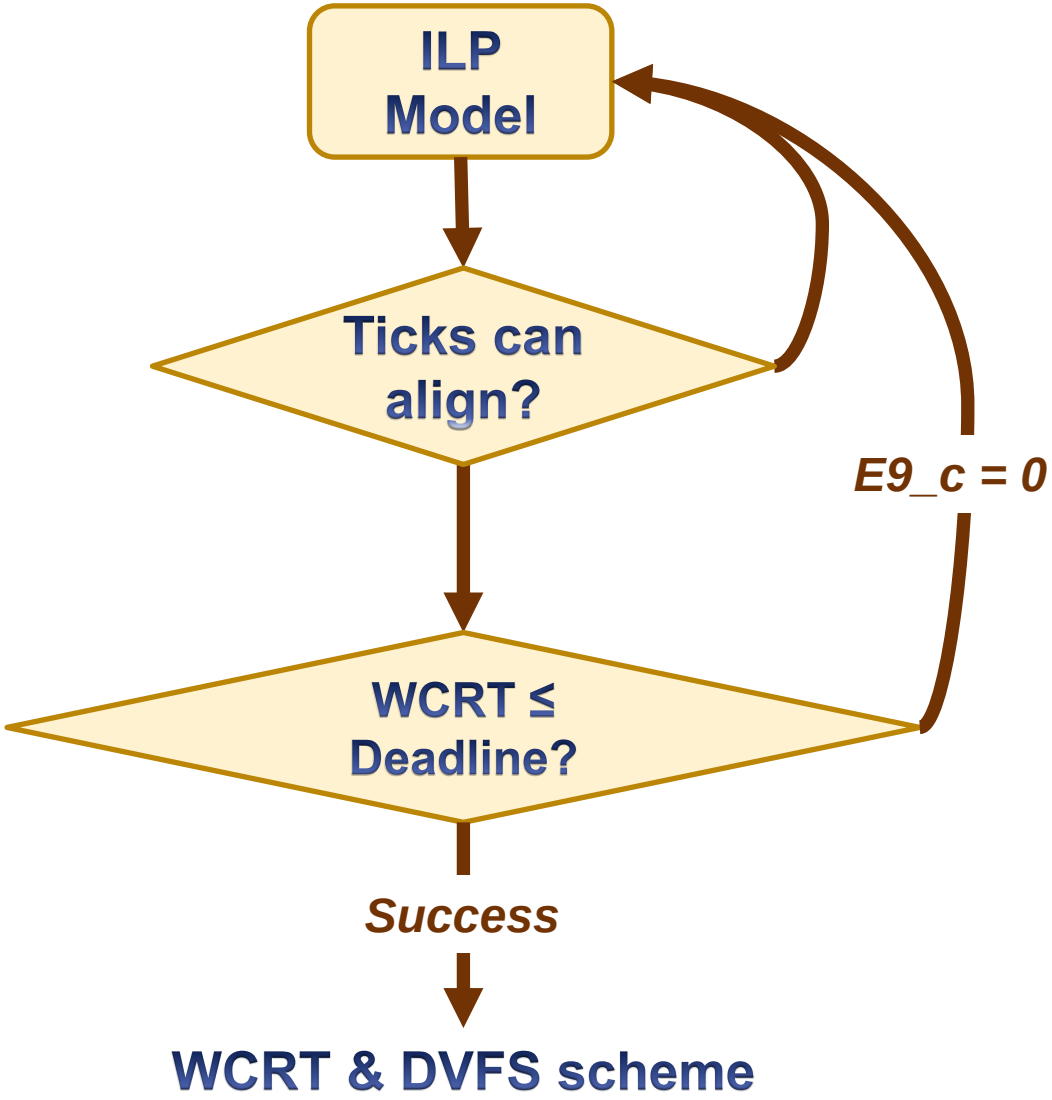
- 3 EOT edges
- Each contribute toward the WCRT
- Assume the deadline is 30 unit time
- Increase the frequency of ***E9*** by add the following constraint:

E9_c = 0

New WCRT with ***E9_b*:**

WCRT = 10+8+12 = 30

DVFS analysis



DVFS analysis

- If a DVFS point is already at its max frequency.
  - E.g., ***E9_c = 0*** and ***E9_b = 0***
  - Adjust the next most influential DVFS point.
- If all DVFS points along an execution path are at max frequency and WCRT > deadline.
  - Impossible to meet the deadline.

Outline

- Background
- Problem statement
- Methodology
  - Extending base ILP model
  - DVFS analysis component
- Results
- Conclusions

Evaluation

- Compare the WCRT and worst case energy consumption per tick.
  - Max frequency
  - With DVFS

Evaluation

- Assumptions:
  - Energy consumption and DVFS switching delay is estimated based on published data.
  - Perfect clock gate
    - Idle consumes no energy ($P_{dynamic} = 0$)
  - Energy consumption is computed using the extended ILPc
    - Using energy per cycle instead of unit time.
    - All un-used DVFS points are set to 0.

Kihwan Choi; Wonbok Lee; Soma, R.; Pedram, M., "Dynamic voltage and frequency scaling under a precise energy model considering variable and fixed components of the system power dissipation," ICCAD 2004

Evaluation

- Assumptions:
  - Programs execute periodically with period equal to deadline.
  - Each execution is one tick.
  - Enable us to compute worst case energy per tick, thereof compute worst case battery life.

Benchmark programs

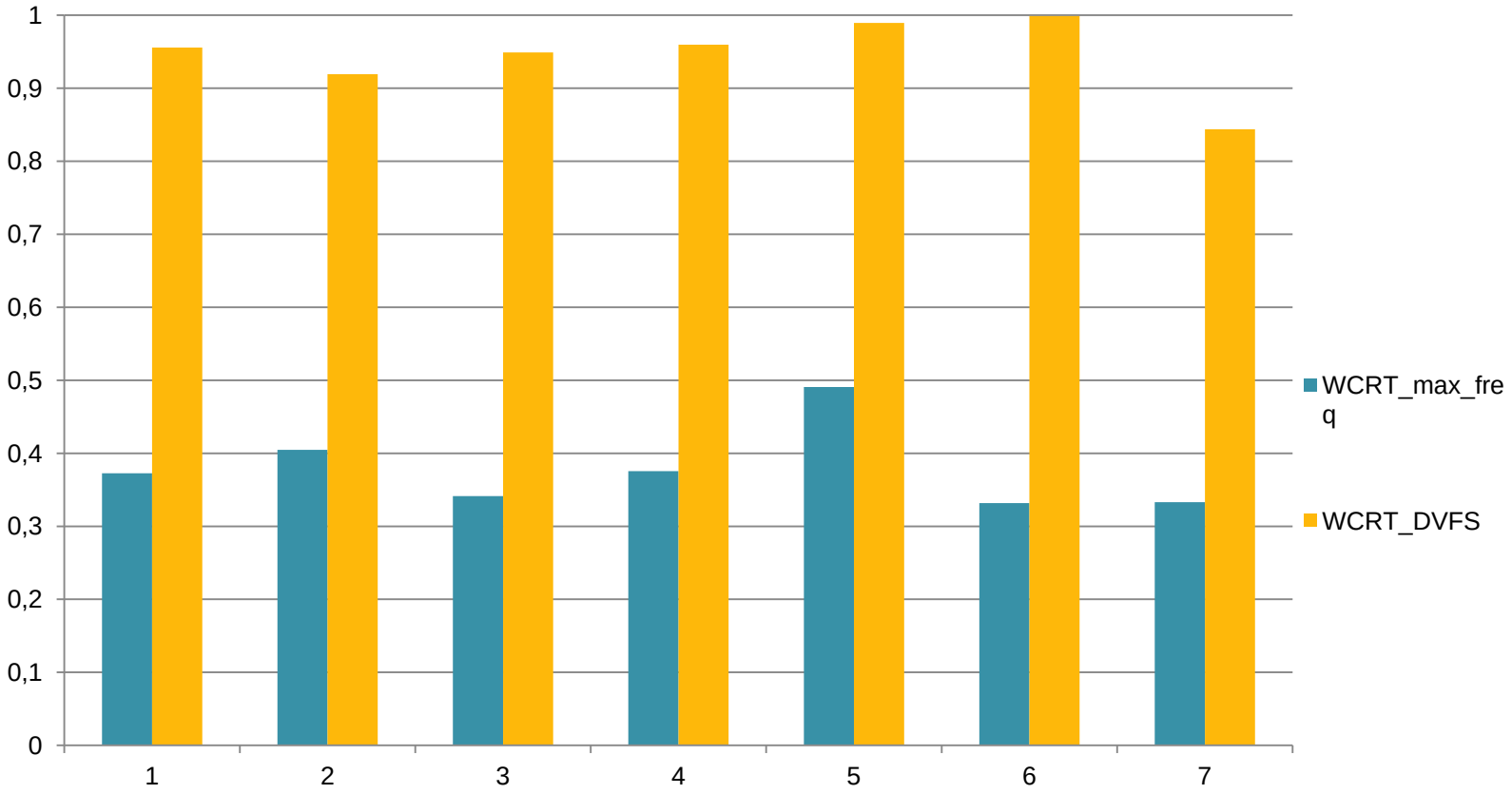
	Name	Threads
1	ChannelProtocol	7
2	Flasher	7
3	RobotSonar	7
4	Synthetic1	7
5	Synthetic1	7
6	CruiseControl	25
7	WaterMonitor	40

L. H. Yoong and G. D. Shaw. *Auckland function block benchmark. University of Auckland*, 2010. www.ece.auckland.ac.nz/~pretzel/Auckland_FB_Benchmark.zip

Results - WCRT

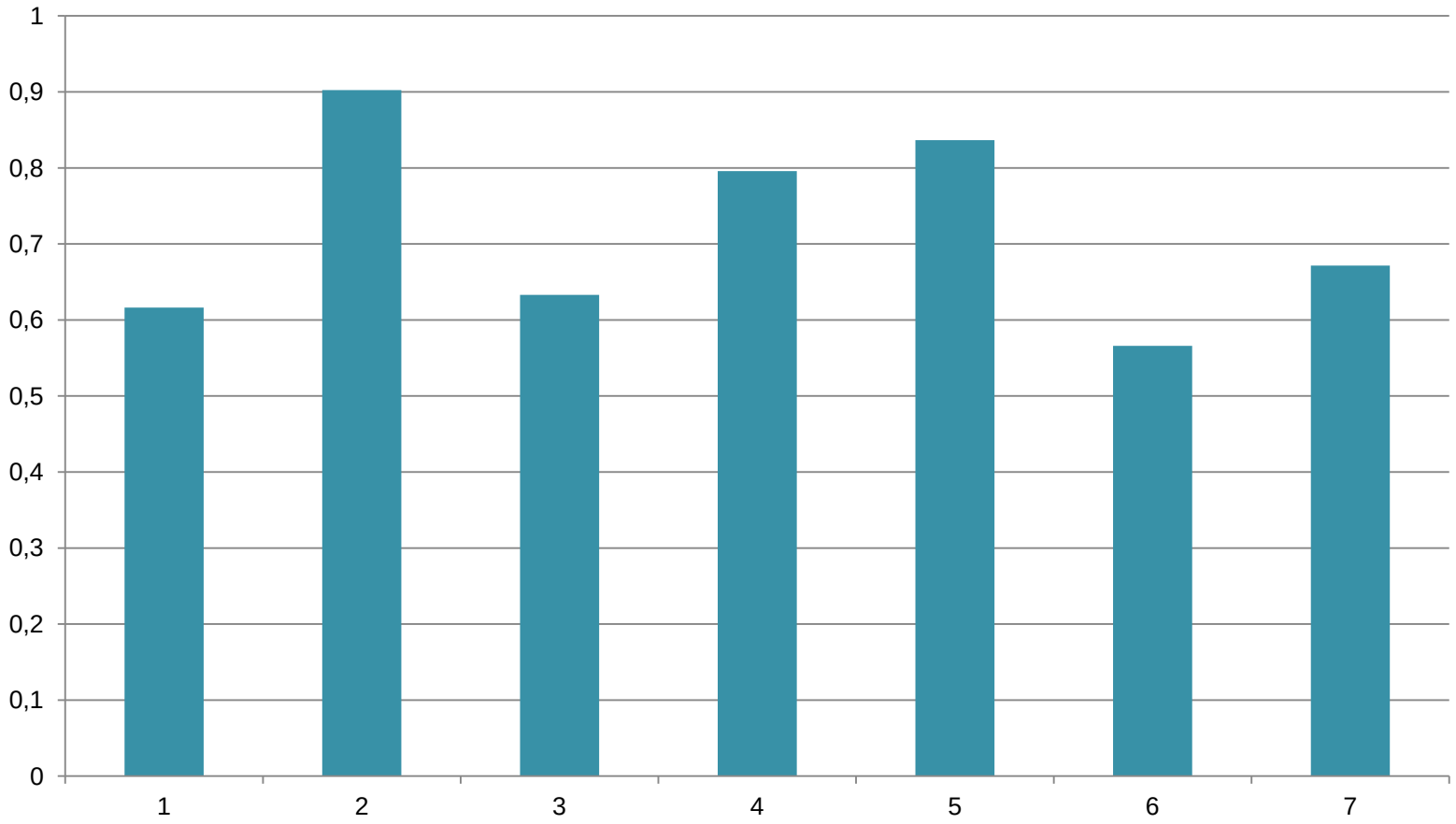
		Deadline	WCRT_max_freq	WCRT_DVFS
1	ChannelProtocol	2773	1033	2650
2	Flasher	1569	635	1442
3	RobotSonar	5577	1904	5292
4	Synthetic1	5971	2242	5729
5	Synthetic1	5203	2554	5148
6	CruiseControl	6110	2027	6102
7	WaterMonitor	14012	4666	11822

Results - WCRT



Normalized WCRT with respect to deadline

Results - Energy per tick

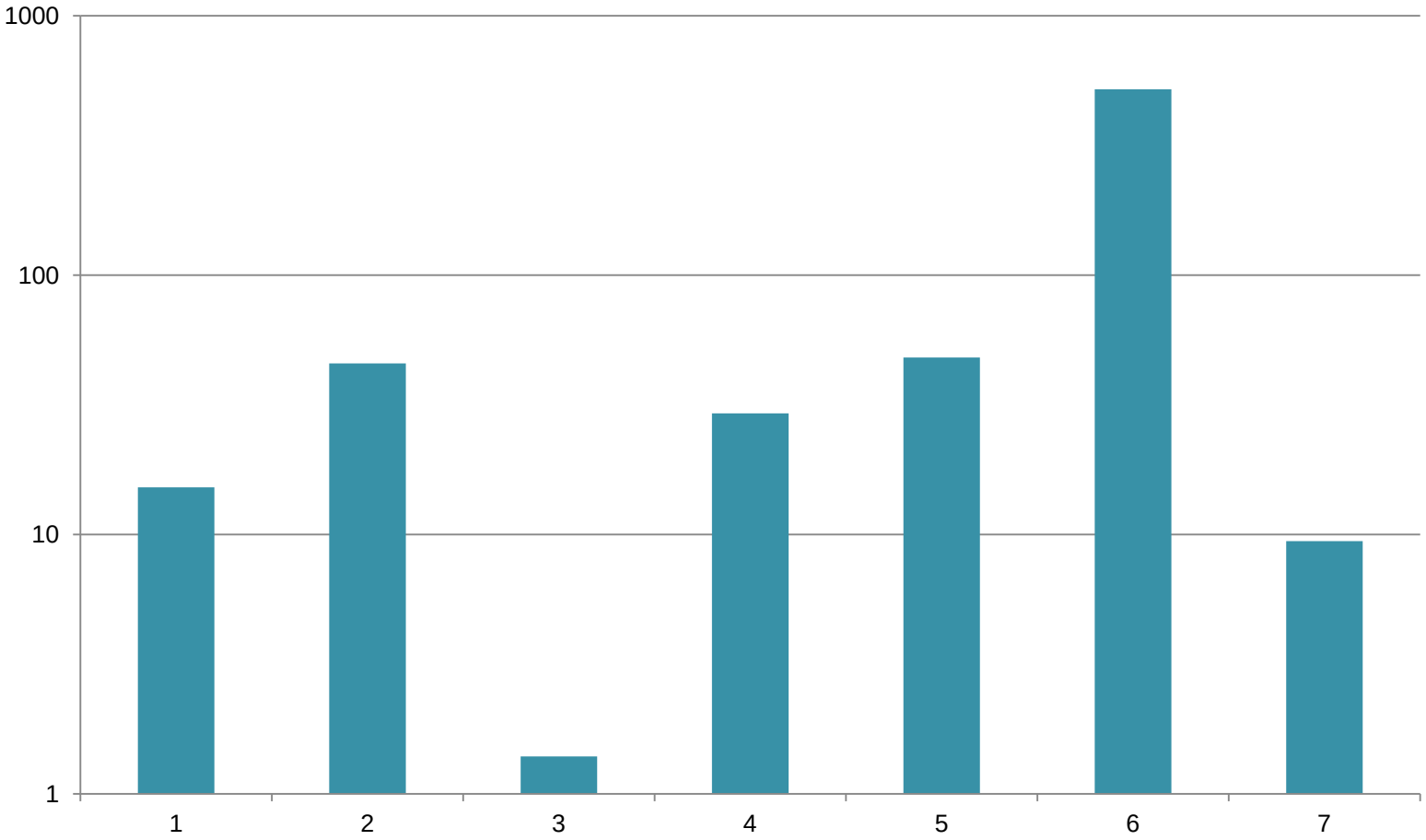


Normalized worst case energy consumption with DVFS with respect to max frequency

Results - Analysis time

		WCRT_max_freq (s)	WCRT_DVFS (s)
1	ChannelProtocol	0.106	1.61
2	Flasher	0.046	2.1
3	RobotSonar	0.28	0.39
4	Synthetic1	2.83	82.91
5	Synthetic1	0.85	40.91
6	CruiseControl	0.32	166.52
7	WaterMonitor	0.07	0.66

Results - Analysis time



Normalized ILPc with DVFS with respect to ILPc

Conclusion

- These results are provisional.
- Analysis time significant longer than ILP due to grow in state space.
  - Still quite scalable
- WCRT with DVFS is closer to deadline.
- 30% lower worst case energy consumption.
  - Should be much lower on average .

Thank you!