Timing Anomaly through Branch Prediction

Andrei Ilin

Université Grenoble Alpes

23 June 2025

Supervised by Lionel Rieg, Florian Brandner and Mihail Asavoae

Critical Real-Time Systems

Can be found in:

- Cars
- Planes
- Life-supporting equipment

Strict timing requirements for the programs.

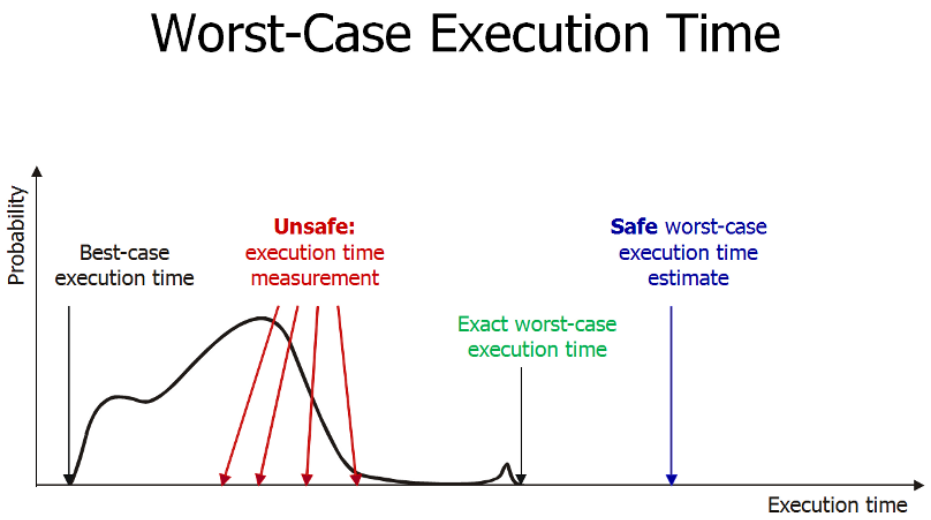
Example

Car break system that must respond in 5 ms.

WCET Analysis

Worst **C**ase **E**xecution **T**ime Analysis:

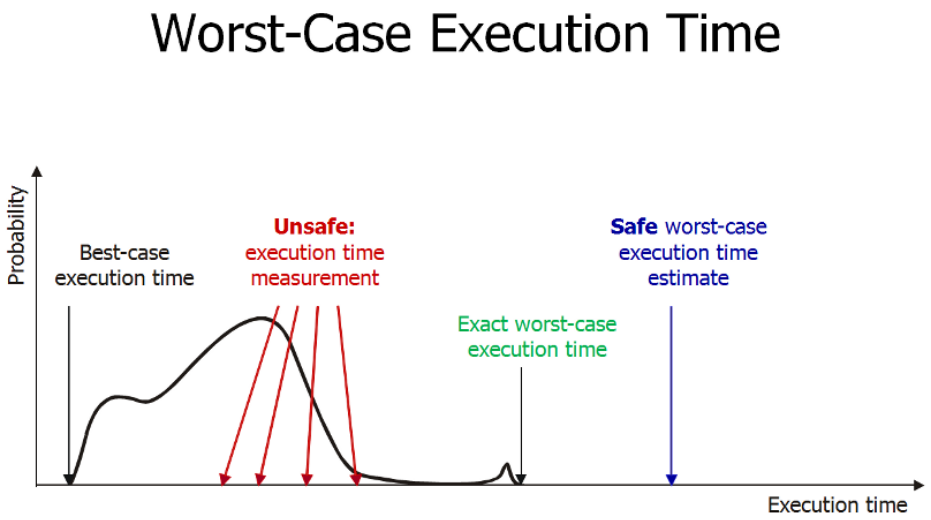
- Hardware + Software
- Upper bound for execution time?

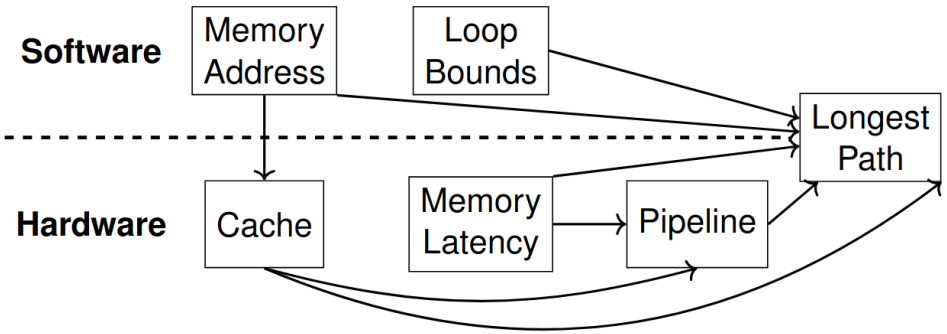


WCET Analysis

Worst **C**ase **E**xecution **T**ime Analysis:

- Hardware + Software
- Upper bound for execution time?





Analysis is split into multiple stages and uses abstractions.

Timing Anomalies

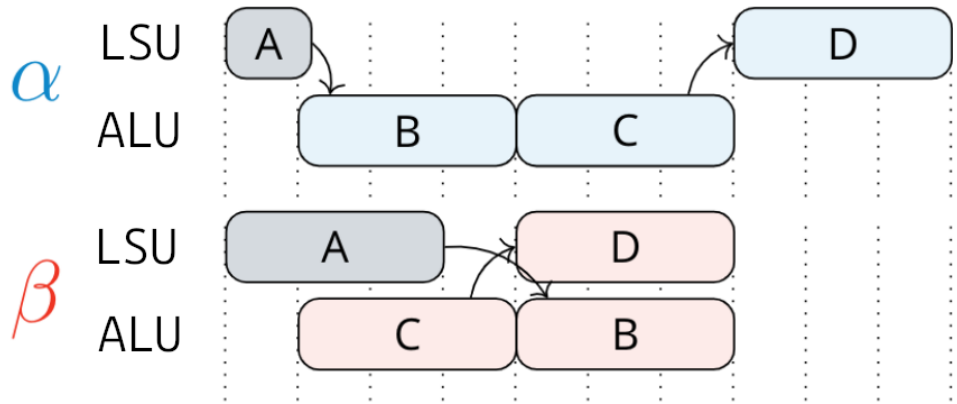
Timing Anomaly (TA)

When a local speedup leads to a global slowdown.

Example

Faster completion of A leads to a slowdown of the whole trace.

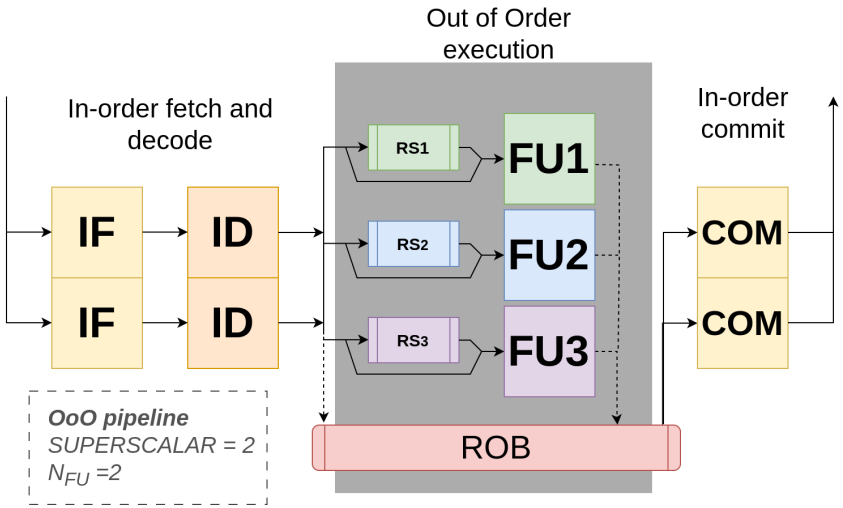
```
LD    r1, 0(r2)    ; A
ADD   r3, r1, r4   ; B
ADD   r5, r6, r7   ; C
LD    r8, (0)r5    ; D
```



OoO Multiscalar Pipeline

Processor:

- **Pipelined:** Divided into consecutive stages
- **Superscalar:** Fetch multiple instruction in the same time
- **Out-of-Order:** Execution order dictated by scheduling policy

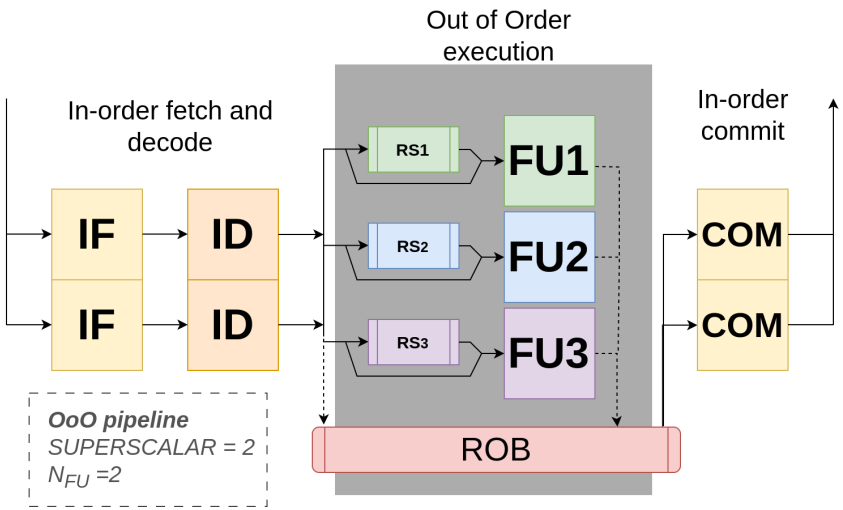


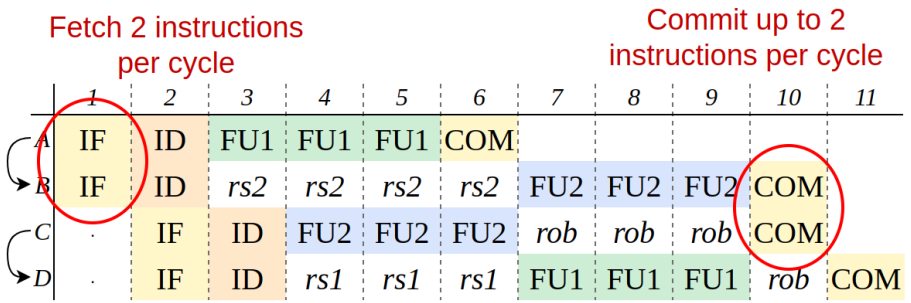
	1	2	3	4	5	6	7	8	9	10	11
A	IF	ID	FU1	FU1	FU1	COM					
B	IF	ID	*rs2*	*rs2*	*rs2*	*rs2*	FU2	FU2	FU2	COM	
C	.	IF	ID	FU2	FU2	FU2	*rob*	*rob*	*rob*	COM	
D	.	IF	ID	*rs1*	*rs1*	*rs1*	FU1	FU1	FU1	*rob*	COM

OoO Multiscalar Pipeline

Processor:

- **Pipelined:** Divided into consecutive stages
- **Superscalar:** Fetch multiple instruction in the same time
- **Out-of-Order:** Execution order dictated by scheduling policy

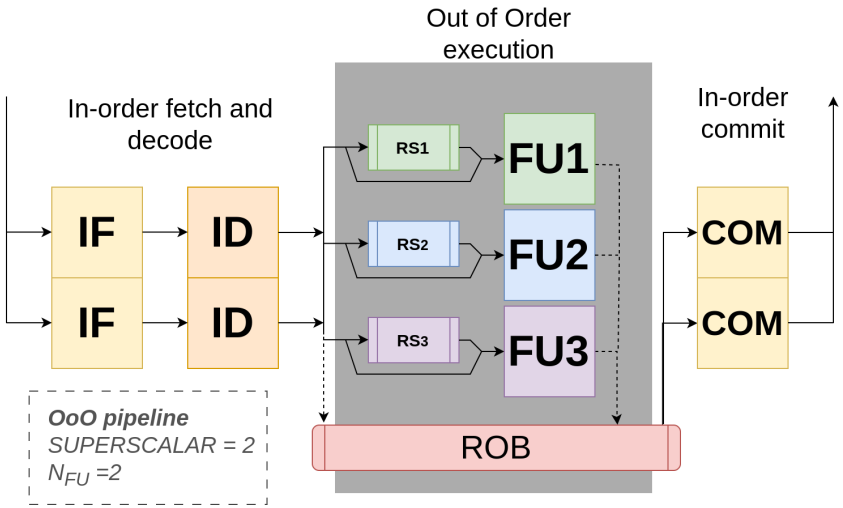




OoO Multiscalar Pipeline

Processor:

- **Pipelined:** Divided into consecutive stages
- **Superscalar:** Fetch multiple instruction in the same time
- **Out-of-Order:** Execution order dictated by scheduling policy



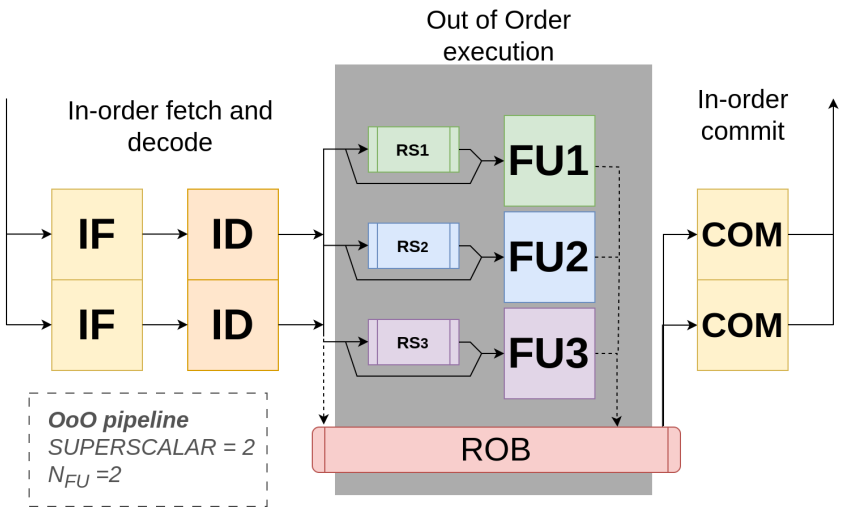
Execute out-of-order

	1	2	3	4	5	6	7	8	9	10	11
A	IF	ID	FU1	FU1	FU1	COM					
B	IF	ID	*rs2*	*rs2*	*rs2*	*rs2*	FU2	FU2	FU2	COM	
C	.	IF	ID	FU2	FU2	FU2	*rob*	*rob*	*rob*	COM	
D	.	IF	ID	*rs1*	*rs1*	*rs1*	FU1	FU1	FU1	*rob*	COM

OoO Multiscalar Pipeline

Processor:

- **Pipelined:** Divided into consecutive stages
- **Superscalar:** Fetch multiple instruction in the same time
- **Out-of-Order:** Execution order dictated by scheduling policy

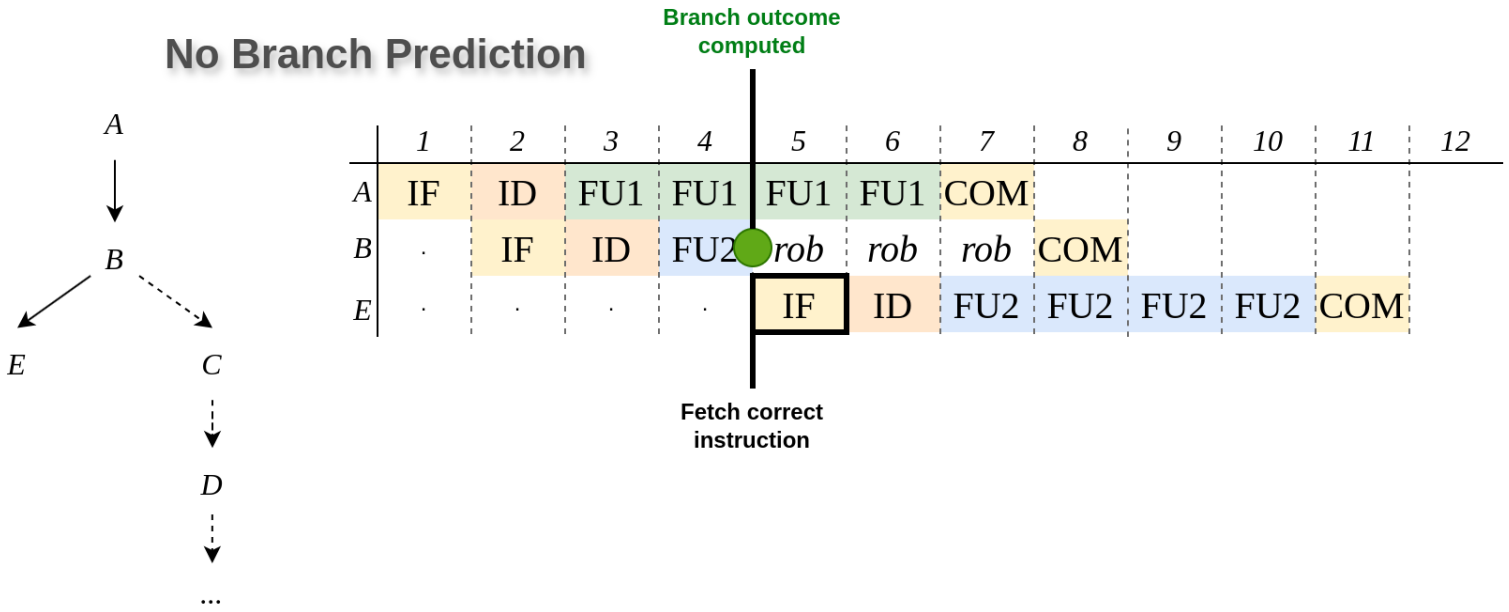


Commit in-order

	1	2	3	4	5	6	7	8	9	10	11
A	IF	ID	FU1	FU1	FU1	COM					
B	IF	ID	*rs2*	*rs2*	*rs2*	*rs2*	FU2	FU2	FU2	COM	
C	.	IF	ID	FU2	FU2	FU2	*rob*	*rob*	*rob*	COM	
D	.	IF	ID	*rs1*	*rs1*	*rs1*	FU1	FU1	FU1	*rob*	COM

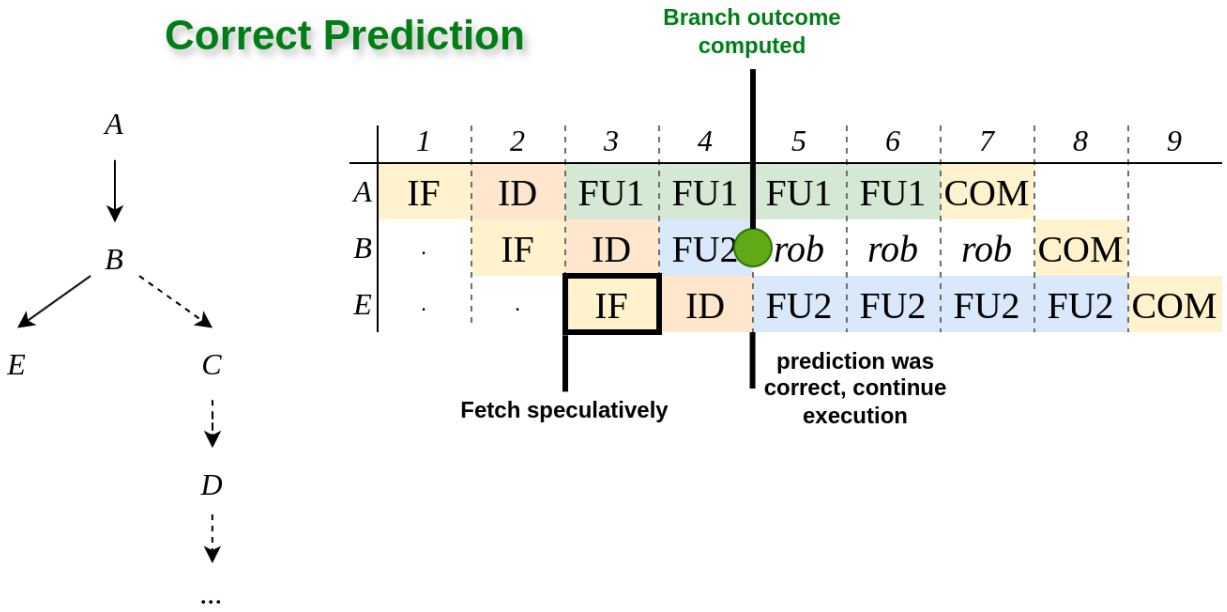
Branch Prediction

- Branch target unknown until branch is resolved.



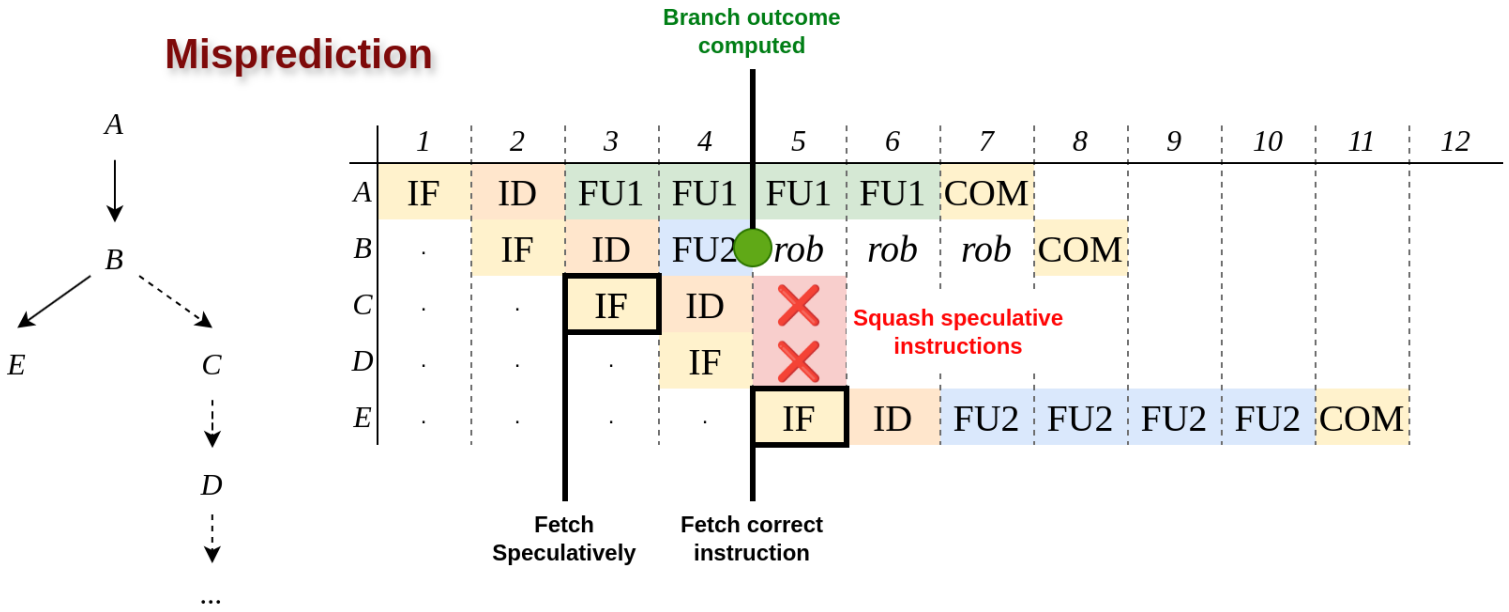
Branch Prediction

- Branch target unknown until branch is resolved.
- Predict next address and execute speculatively.



Branch Prediction

- Branch target unknown until branch is resolved.
- Predict next address and execute speculatively.
- Misprediction detected → start fetching the correct branch



Formal Definition?

How do we formally define a TA?

Many existing definitions

- Step Heights (by Gebhard et al.)
- Step-functions Intersections (by Cassez et al.)
- Component Occupation (by Kirner et al.)
- Instruction Locality (by Reineke et al.)
- Causality Dependency Graph (by Binder et al.)

Formal Definition?

How do we formally define a TA?

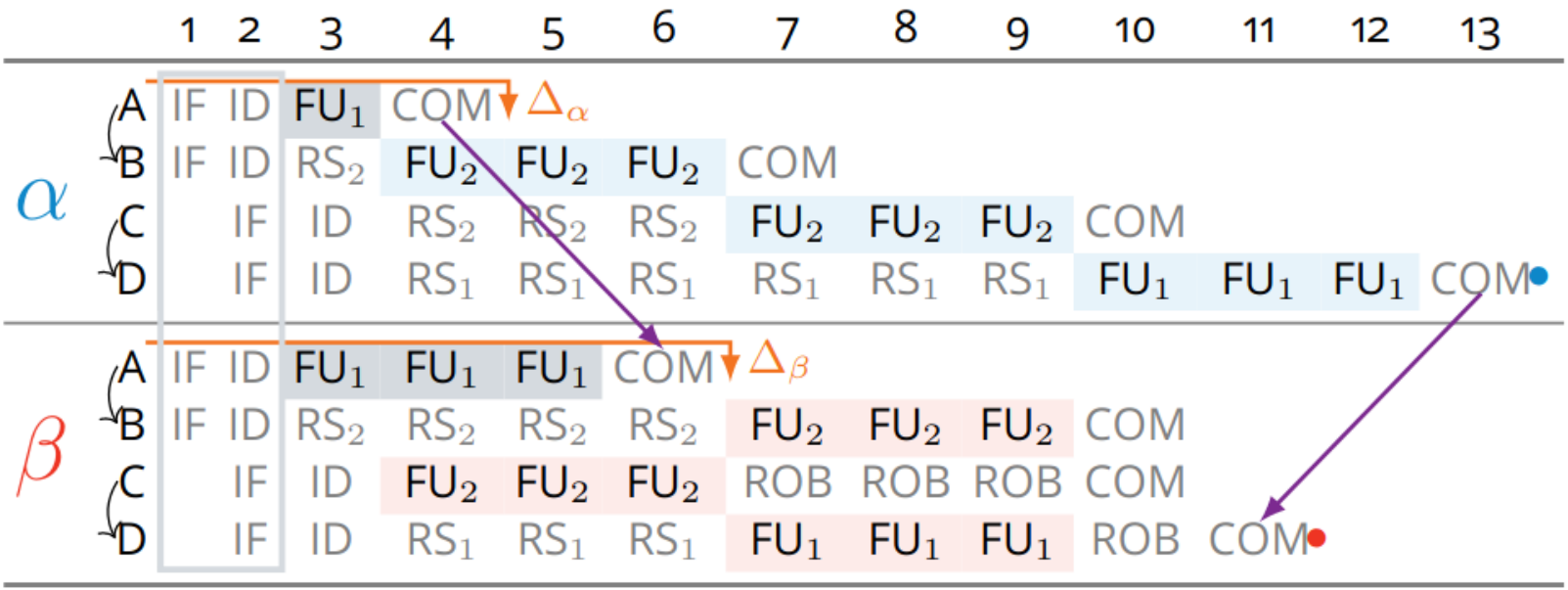
Many existing definitions

- **Step Heights (by Gebhard et al.)**
- **Step-functions Intersections (by Cassez et al.)**
- Component Occupation (by Kirner et al.)
- Instruction Locality (by Reineke et al.)
- **Causality Dependency Graph (by Binder et al.)**

Step Heights by Gebhard et al.

Definition

TA = instruction latency (compared to previous instr.) smaller, global time greater

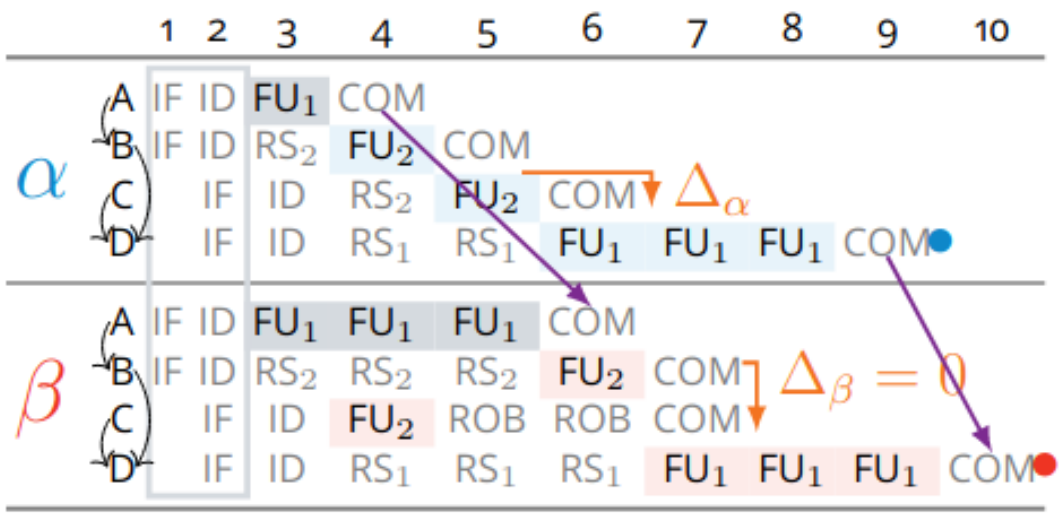


Step Heights by Gebhard et al.

Definition

TA = instruction latency (compared to previous instr.) smaller, global time greater

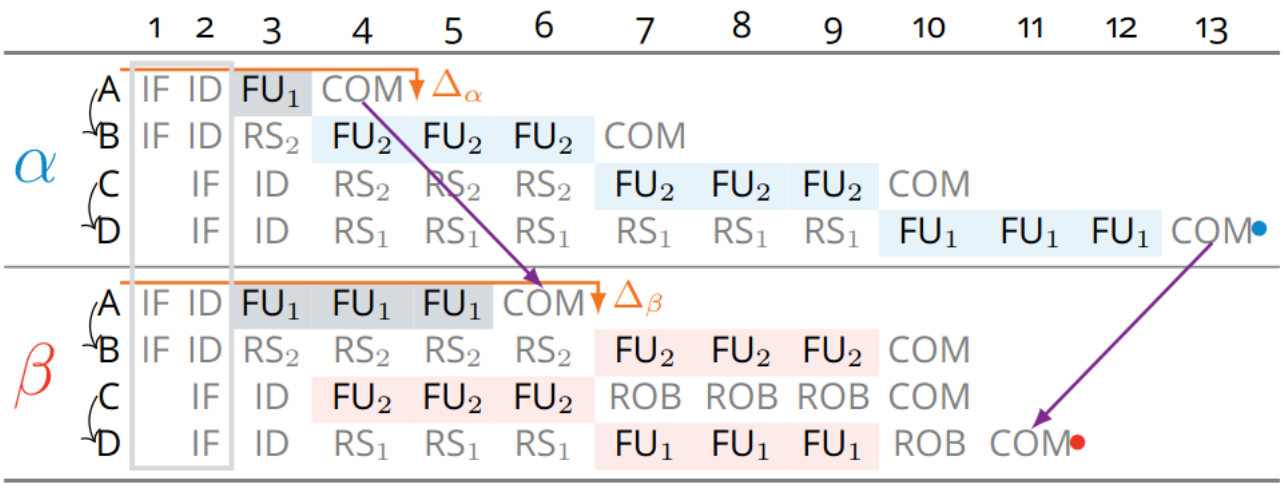
Counterexample!



Step Function Intersections by Cassez et al.

Definition

TA = some instruction finishes earlier in trace α than in β, subsequent instruction finishes later in α than in β

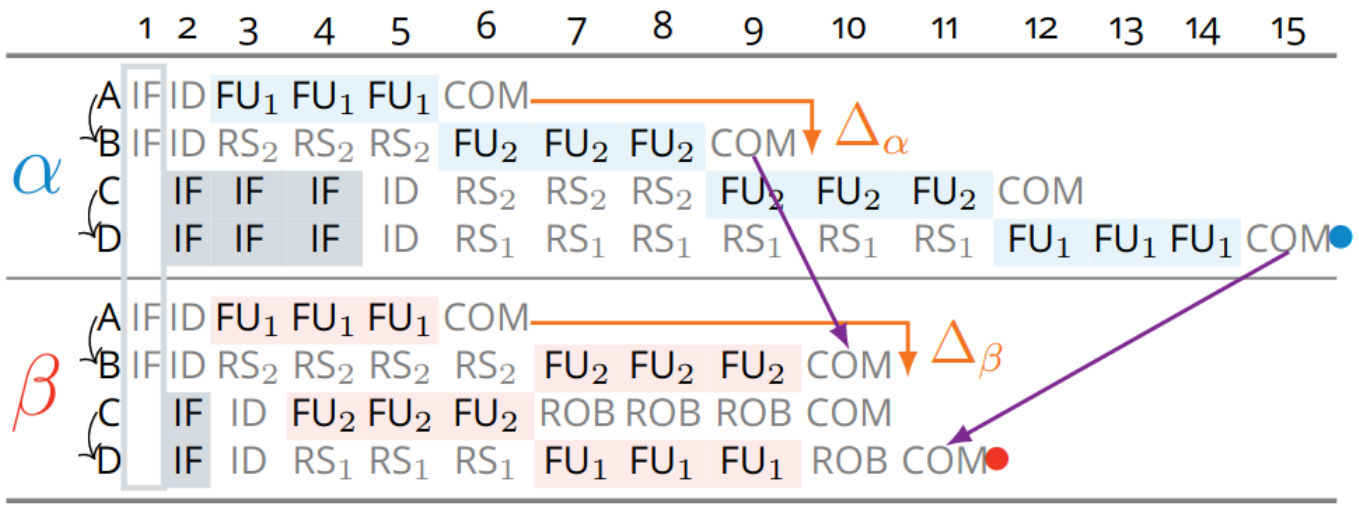


Step Function Intersections by Cassez et al.

Definition

TA = some instruction finishes earlier in trace α than in β, subsequent instruction finishes later in α than in β

Counterexample!



Causality Graph by Binder et al.

3 Components

1. Variation
2. Slowdown
3. Causality

α

	1	2	3	4	5	6	7	8	9	10	11	12	13
A	IF	ID	FU1	COM									
B	IF	ID	*rs*	FU2	FU2	FU2	COM						
C	·	IF	ID	*rs*	*rs*	*rs*	FU2	FU2	FU2	COM			
D	·	IF	ID	*rs*	*rs*	*rs*	*rs*	*rs*	*rs*	FU1	FU1	FU1	COM

β

	1	2	3	4	5	6	7	8	9	10	11
A	IF	ID	FU1	FU1	FU1	COM					
B	IF	ID	*rs*	*rs*	*rs*	*rs*	FU2	FU2	FU2	COM	
C	·	IF	ID	FU2	FU2	FU2	*rob*	*rob*	*rob*	COM	
D	·	IF	ID	*rs*	*rs*	*rs*	FU1	FU1	FU1	*rob*	COM

Causality Graph by Binder et al.

3 Components

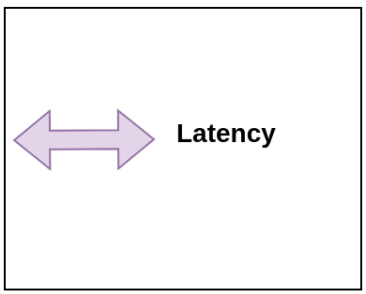
1. **Variation** in latency
2. Slowdown
3. Causality

α

	1	2	3	4	5	6	7	8	9	10	11	12	13
A	IF	ID	FU1	COM									
B	IF	ID	*rs*	FU2	FU2	FU2	COM						
C	·	IF	ID	*rs*	*rs*	*rs*	FU2	FU2	FU2	COM			
D	·	IF	ID	*rs*	*rs*	*rs*	*rs*	*rs*	*rs*	FU1	FU1	FU1	COM

β

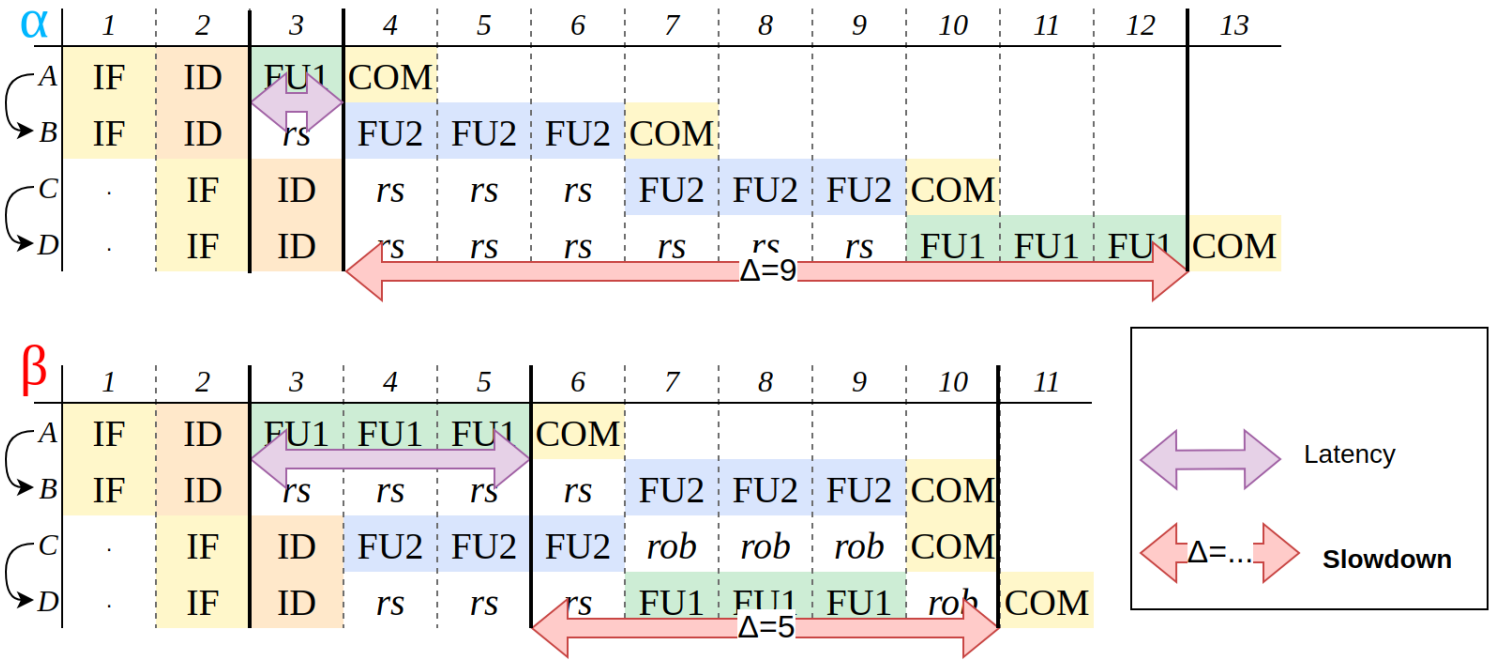
	1	2	3	4	5	6	7	8	9	10	11
A	IF	ID	FU1	FU1	FU1	COM					
B	IF	ID	*rs*	*rs*	*rs*	*rs*	FU2	FU2	FU2	COM	
C	·	IF	ID	FU2	FU2	FU2	*rob*	*rob*	*rob*	COM	
D	·	IF	ID	*rs*	*rs*	*rs*	FU1	FU1	FU1	*rob*	COM



Causality Graph by Binder et al.

3 Components

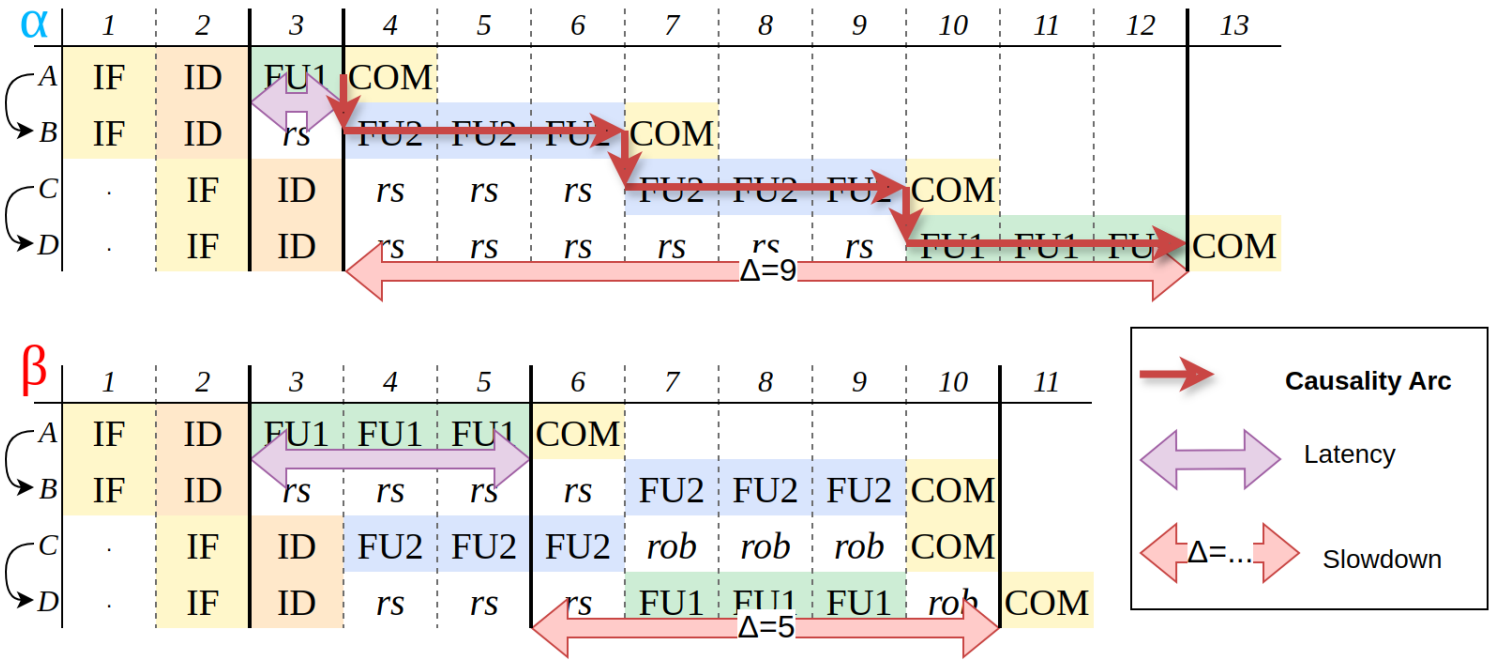
1. Variation in latency
2. **Slowdown** between end of the variation and a later event
3. Causality



Causality Graph by Binder et al.

3 Components

1. Variation in latency
2. Slowdown between end of the variation and a later event
3. **Causality** link = event X cannot be earlier because of Y



Goal

Develop a consistent TA definition applicable for branch prediction.

- **Question 1** Can timing anomaly be caused by branch prediction?
- **Question 2** Can we extend the existing Binder's definition?
- **Question 3** Do we cover all the aspects of branch prediction?

Research Plan

1. Create a tool for automatic example generation.
2. Generate some candidate scenarios.
3. Try to adapt definition on them.
4. Find new scenarios to verify the definition.

Tool Implementation

Existing TLA$^+$ framework

- ✗ Slow: 2-3 seconds for single example
- ✗ No branch and speculation support
- ✗ Lengthy input format
- ✓ Expressive property specification using temoral logic

Tool Implementation

Existing TLA$^+$ framework

- ✗ Slow: 2-3 seconds for single example
- ✓ ~~No branch and speculation support~~ **Added**
- ✓ ~~Lengthy input format~~ **Generate TLA$^+$ from pretty input**
- ✓ Expressive property specification using temoral logic

Tool Implementation

Existing TLA$^+$ framework

- ✗ Slow: 2-3 seconds for single example
- ✓ Added branch and speculation support
- ✓ Generate TLA$^+$ from pretty input
- ✓ Expressive property specification using temoral logic

Our C++ framework

- ✓ Fast: a few ms per example
- ✓ Traces with speculation
- ✓ Consize input format
- ✗ Less expressive properties
- ✓ 3 operation modes:
  1. Interactive
  2. Randomized
  3. State Exploration

Input Format

- Each branch instruction is followed by **misprediction region** – sequence of instruction representing the wrong branch.
- A pair of traces is generated from a single input.

		Res	Dep.	FU Lat
A		FU1		4
$*B$		FU2	$\{A\}$	1
	C	FU2		4
	D	FU2		4
E		FU2		4

α

	1	2	3	4	5	6	7	8	9	10
A	IF	ID	FU1	FU1	FU1	FU1	COM			
B	·	IF	ID	FU2	*rob*	*rob*	*rob*	COM		
C										
D										
E	·	·	IF	ID	FU2	FU2	FU2	FU2	COM	

β

	1	2	3	4	5	6	7	8	9	10	11	12
A	IF	ID	FU1	FU1	FU1	FU1	COM					
B	·	IF	ID	FU2	*rob*	*rob*	*rob*	COM				
C	·	·	IF	ID	✗							
D	·	·	·	IF	✗							
E	·	·	·	·	IF	ID	FU2	FU2	FU2	FU2	COM	

Applying the definition of Binder et al.

TA example: Correct prediction leads to a longer execution time.

α

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	IF	ID	FU1	FU1	FU1	FU1	COM									
B	.	IF	ID	*rs*	*rs*	*rs*	*rs*	*rs*	*rs*	FU2	FU2	FU2	FU2	COM		
C	.	.	IF	ID	FU2	*rob*	*rob*	*rob*	*rob*	*rob*	*rob*	*rob*	*rob*	*rob*	COM	
D																
E																
F	.	.	.	IF	ID	FU2	FU2	FU2	FU2	*rob*	*rob*	*rob*	*rob*	*rob*	*rob*	COM

β

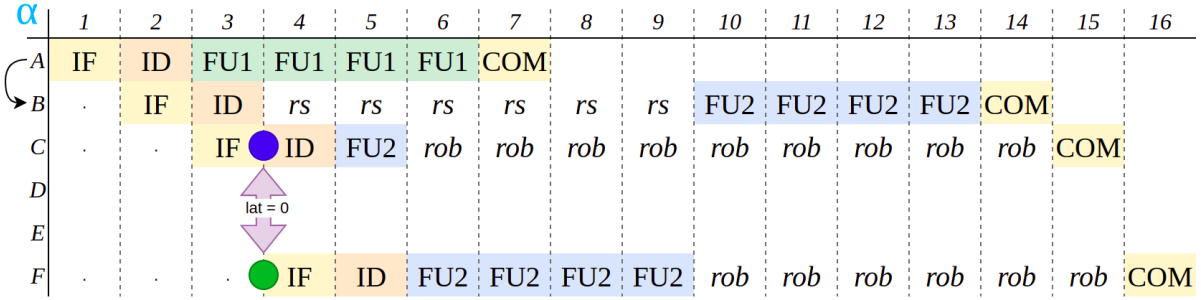
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	IF	ID	FU1	FU1	FU1	FU1	COM								
B	.	IF	ID	*rs*	*rs*	*rs*	FU2	FU2	FU2	FU2	COM				
C	.	.	IF	ID	FU2	*rob*	*rob*	*rob*	*rob*	*rob*	*rob*	COM			
D	.	.	.	IF	ID	❌									
E	.	.	.	.	IF	❌									
F	.	.	.	.	.	IF	ID	*rs*	*rs*	*rs*	FU2	FU2	FU2	FU2	COM

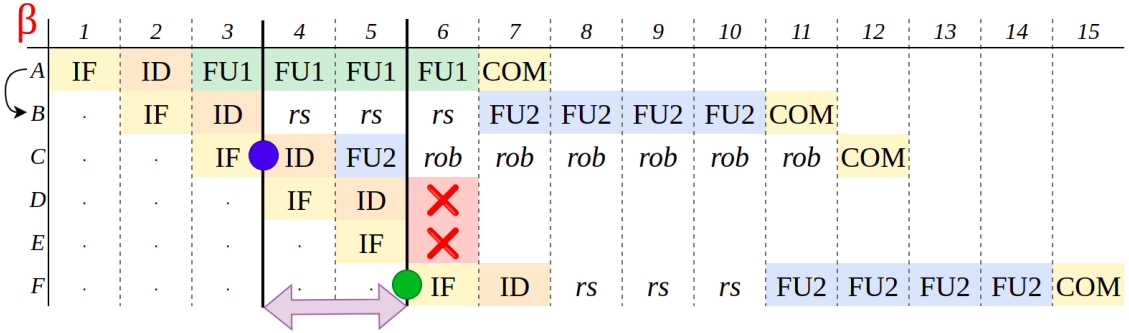
		Res.	Dep.	Lat.
A		FU1		4
B		FU2	{*A*}	4
**C*		**FU2**		1
	D	FU1		4
	E	FU1		4
F		FU2		4

1. **Variation in latency?**
2. **Slowdown?**
3. **Causality?**

Applying the definition of Binder et al.

TA example: Correct prediction leads to a longer execution time.



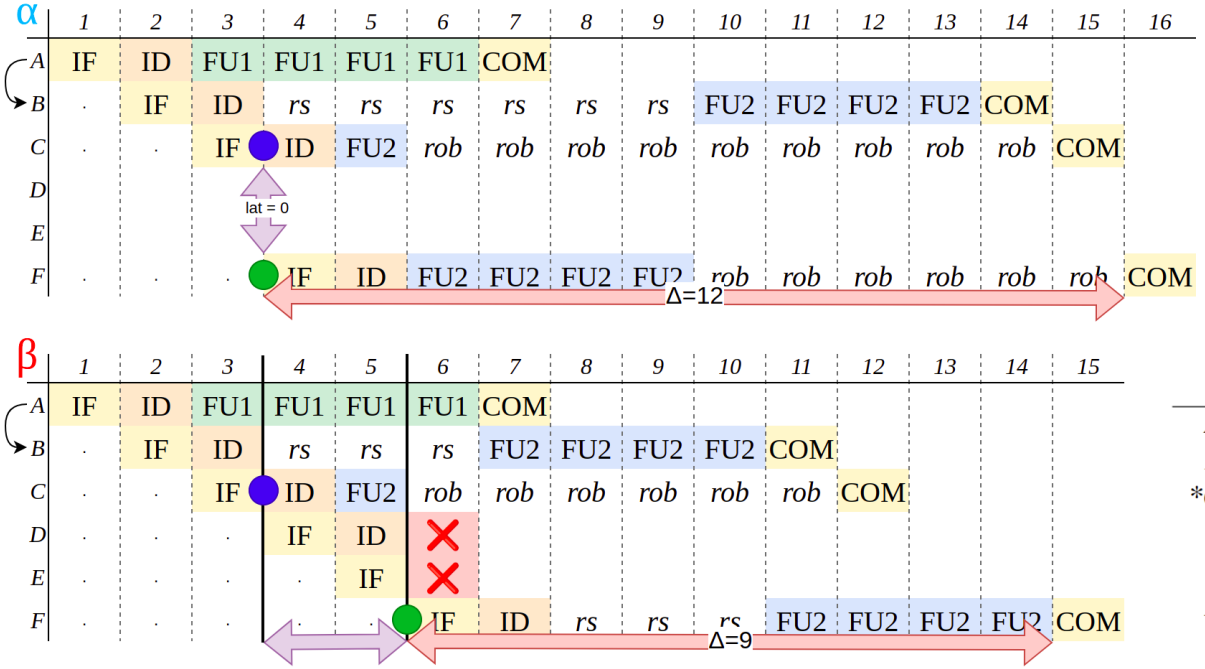


		Res.	Dep.	Lat.
A		FU1		4
B		FU2	{*A*}	4
**C*		**FU2**		1
	D	FU1		4
	E	FU1		4
F		FU2		4

1. **Variation in latency** between "branch prediction" and "correct branch taken" events.
2. **Slowdown?**
3. **Causality?**

Applying the definition of Binder et al.

TA example: Correct prediction leads to a longer execution time.

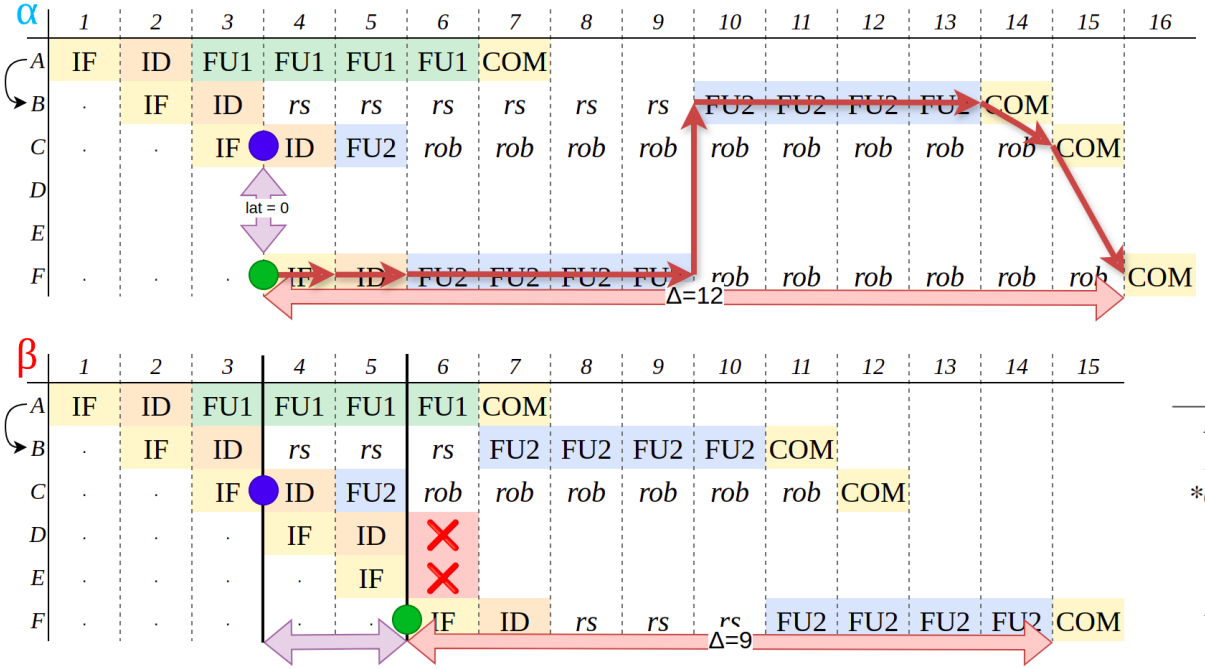


		Res.	Dep.	Lat.
A		FU1		4
B		FU2	{*A*}	4
***C*		**FU2**		1
	D	FU1		4
	E	FU1		4
F		FU2		4

1. **Variation in latency** between "branch prediction" and "correct branch taken" events.
2. **Slowdown** as in the original definition.
3. **Causality?**

Applying the definition of Binder et al.

TA example: Correct prediction leads to a longer execution time.



		Res.	Dep.	Lat.
A		FU1		4
B		FU2	{*A*}	4
**C*		**FU2**		1
	D	FU1		4
	E	FU1		4
F		FU2		4

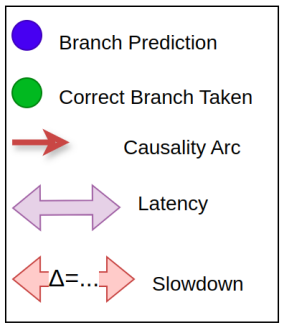
1. **Variation in latency** between "branch prediction" and "correct branch taken" events.
2. **Slowdown** as in the original definition.
3. **Causality** as in the original definition.

FU release by FU acquisition

Example: when a new rule is required.

α

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	IF	ID	FU1	FU1	FU1	FU1	FU1	COM						
B	.	IF	ID	*rs2*	*rs2*	*rs2*	*rs2*	*rs2*	FU2	FU2	FU2	FU2	COM	
C	.	.	IF	ID	*rs1*	*rs1*	*rs1*	FU1	*rob*	*rob*	*rob*	*rob*	*rob*	COM
D	.	.	.	IF	ID	FU2	*rob*	*rob*	✗					
E														
F														
G	.	.	.	.	IF	ID	FU2	FU2	✗					



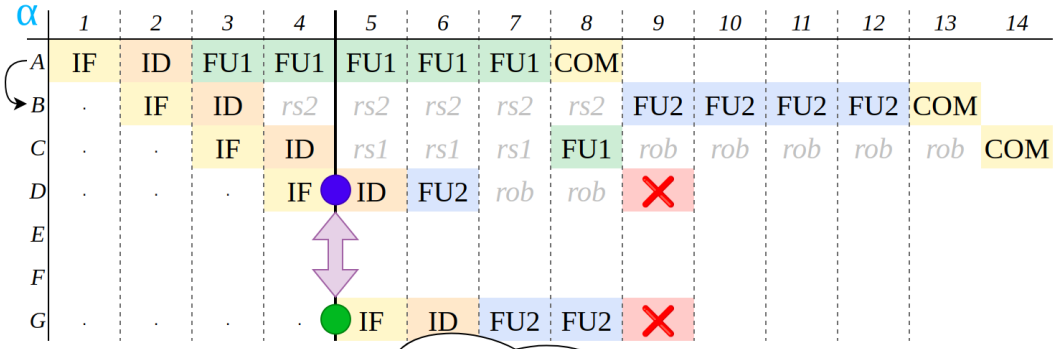
β

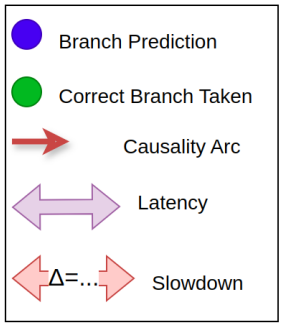
	1	2	3	4	5	6	7	8	9	10	11	12	13
A	IF	ID	FU1	FU1	FU1	FU1	FU1	COM					
B	.	IF	ID	*rs2*	*rs2*	*rs2*	*rs2*	FU2	FU2	FU2	FU2	COM	
C	.	.	IF	ID	*rs1*	*rs1*	*rs1*	FU1	*rob*	*rob*	*rob*	*rob*	COM
D	.	.	.	IF	ID	FU2	*rob*	*rob*	✗				
E	.	.	.	.	IF	ID	✗						
F	.	.	.	.	.	IF	✗						
G	.	.	.	.	.	.	IF	ID	✗				

			Res.	Dep.	Lat.
A			FU1		5
B			FU2	{*A*}	4
C			FU1		1
	**D*		FU2		1
		E	FU2		4
		F	FU2		4
	G		FU2		4

FU release by FU acquisition

Example: when a new rule is required.

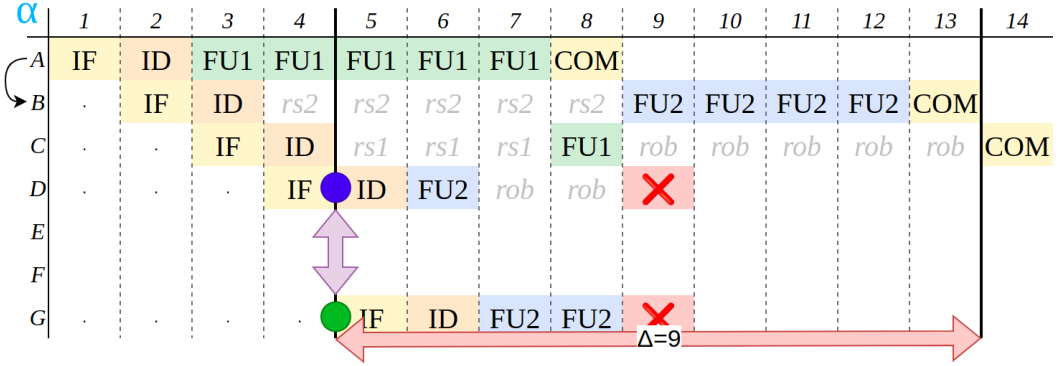


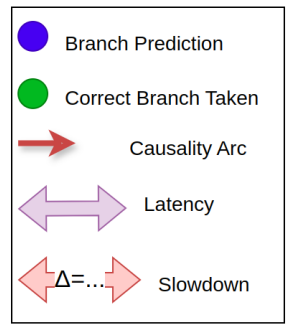


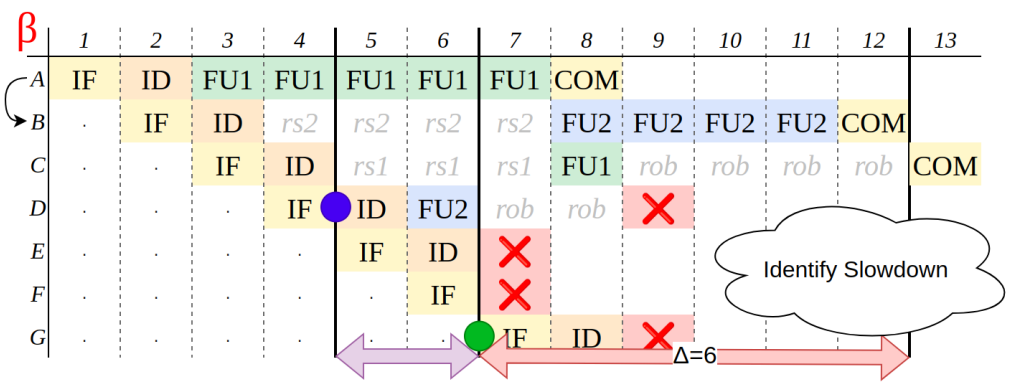
			Res.	Dep.	Lat.
A			FU1		5
B			FU2	{*A*}	4
C			FU1		1
	**D*		FU2		1
		E	FU2		4
		F	FU2		4
	G		FU2		4

FU release by FU acquisition

Example: when a new rule is required.



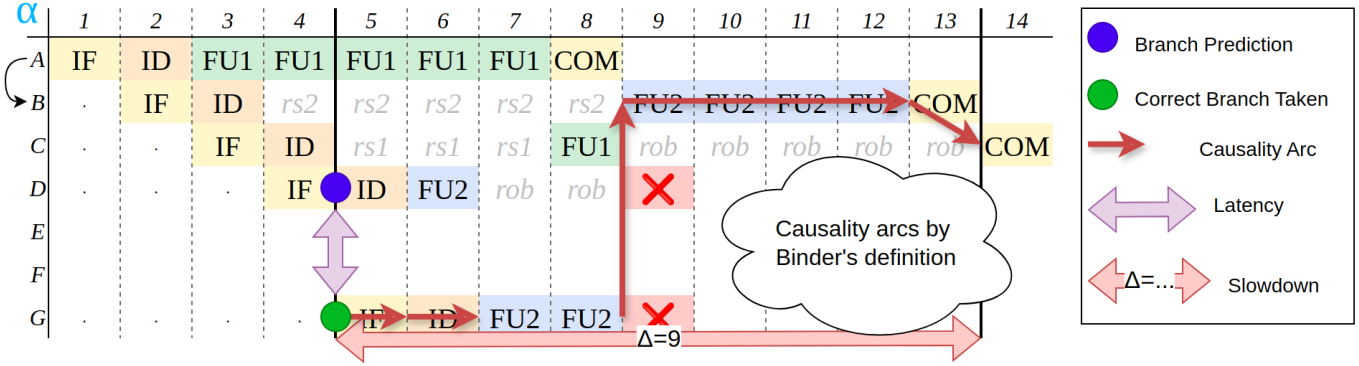


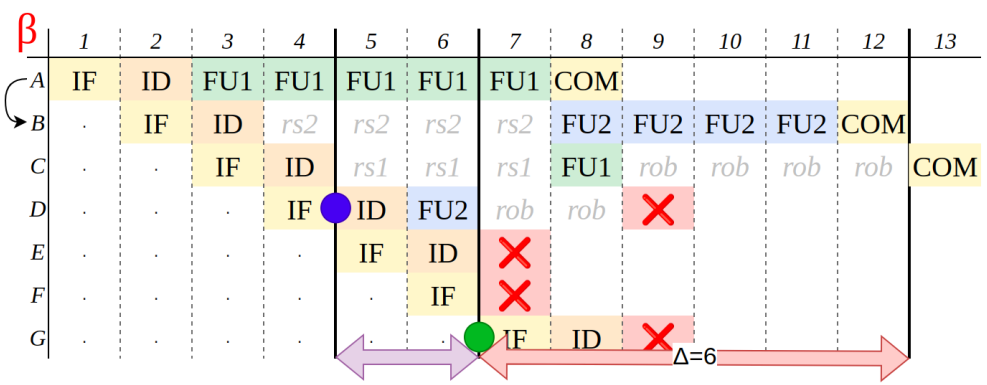


			Res.	Dep.	Lat.
A			FU1		5
B			FU2	{*A*}	4
C			FU1		1
	**D*		FU2		1
		E	FU2		4
		F	FU2		4
	G		FU2		4

FU release by FU acquisition

Example: when a new rule is required.

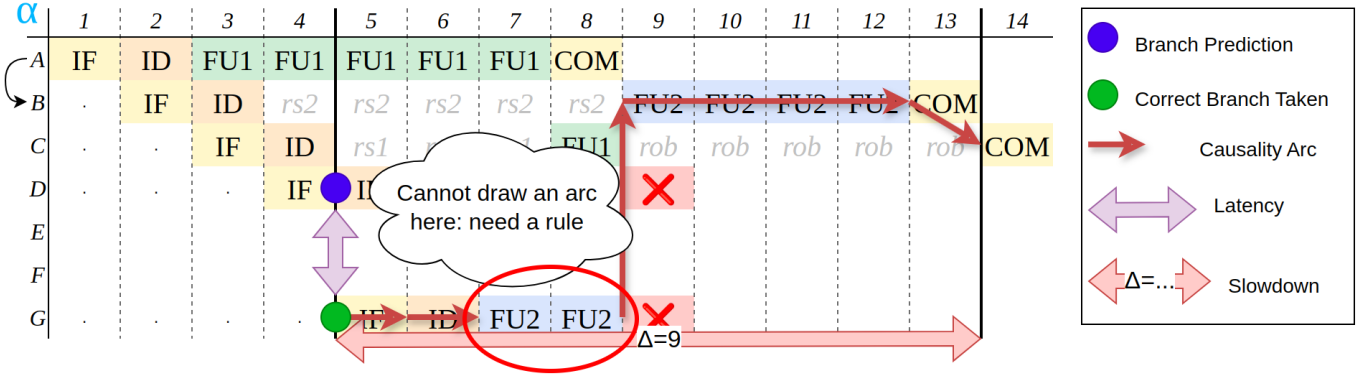


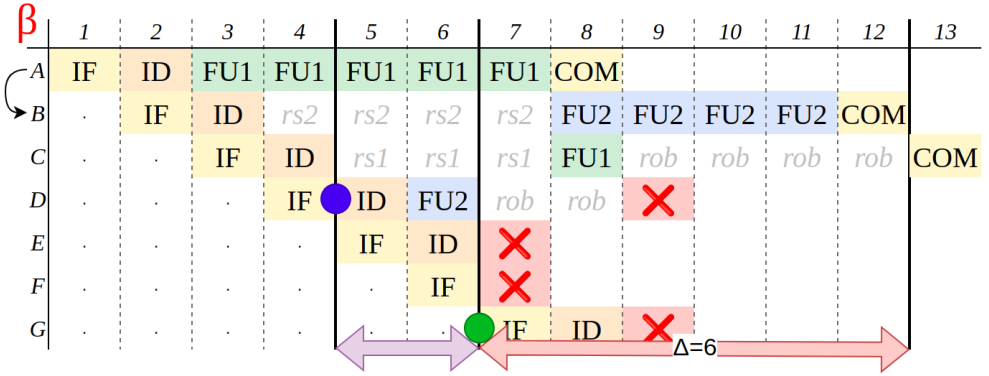


			Res.	Dep.	Lat.
A			FU1		5
B			FU2	{*A*}	4
C			FU1		1
	**D*		FU2		1
		E	FU2		4
		F	FU2		4
	G		FU2		4

FU release by FU acquisition

Example: when a new rule is required.





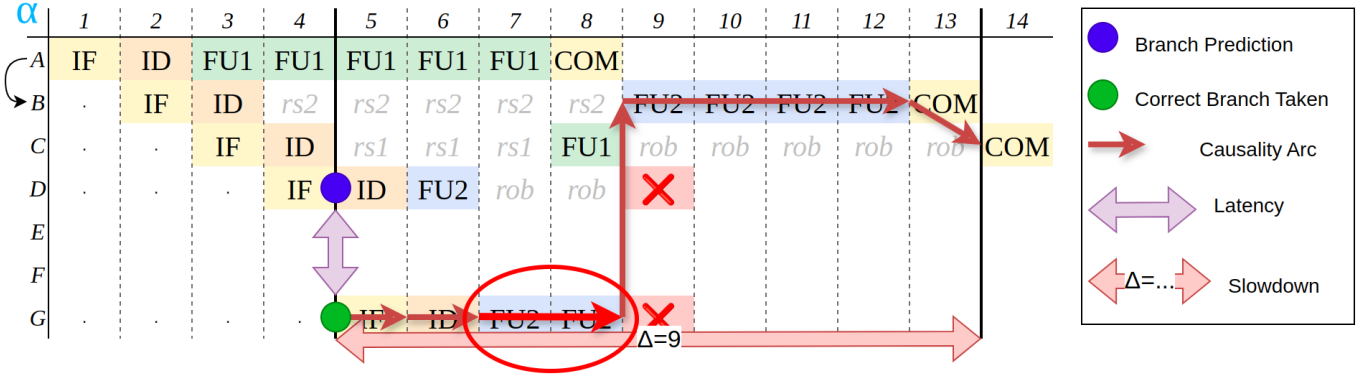
			Res.	Dep.	Lat.
A			FU1		5
B			FU2	{*A*}	4
C			FU1		1
	**D*		FU2		1
		E	FU2		4
		F	FU2		4
	G		FU2		4

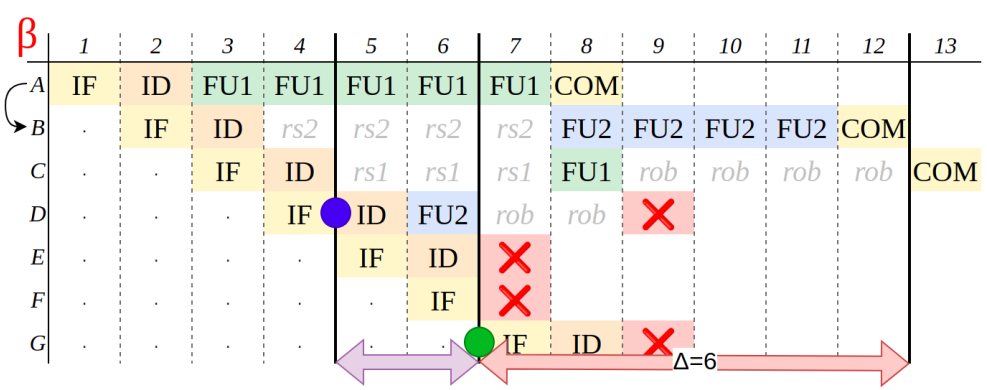
Binder's rule does not work

"4 cycles must be spent inside FU to establish causality."

FU release by FU acquisition

Example: when a new rule is required.





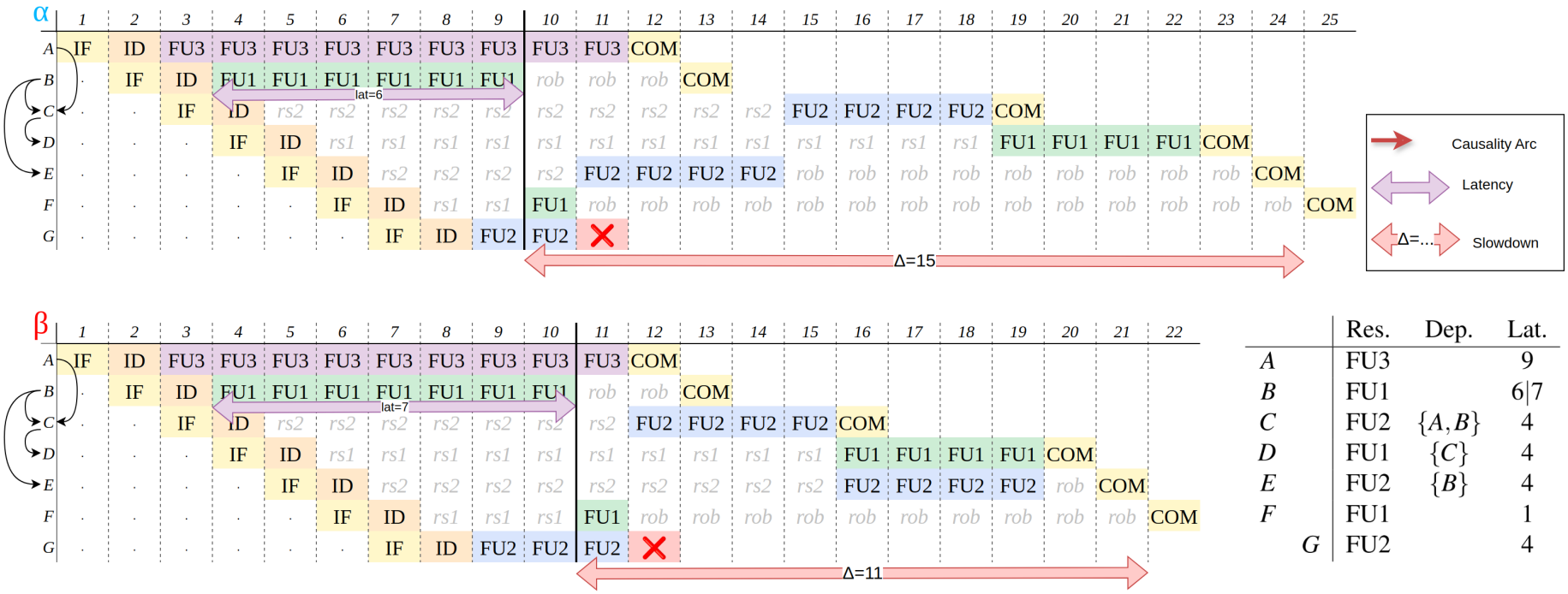
			Res.	Dep.	Lat.
A			FU1		5
B			FU2	{*A*}	4
C			FU1		1
	**D*		FU2		1
		E	FU2		4
		F	FU2		4
	G		FU2		4

"Acquisition" Rule

The FU acquisition is always causal to the respective FU release.

FU release by squashing

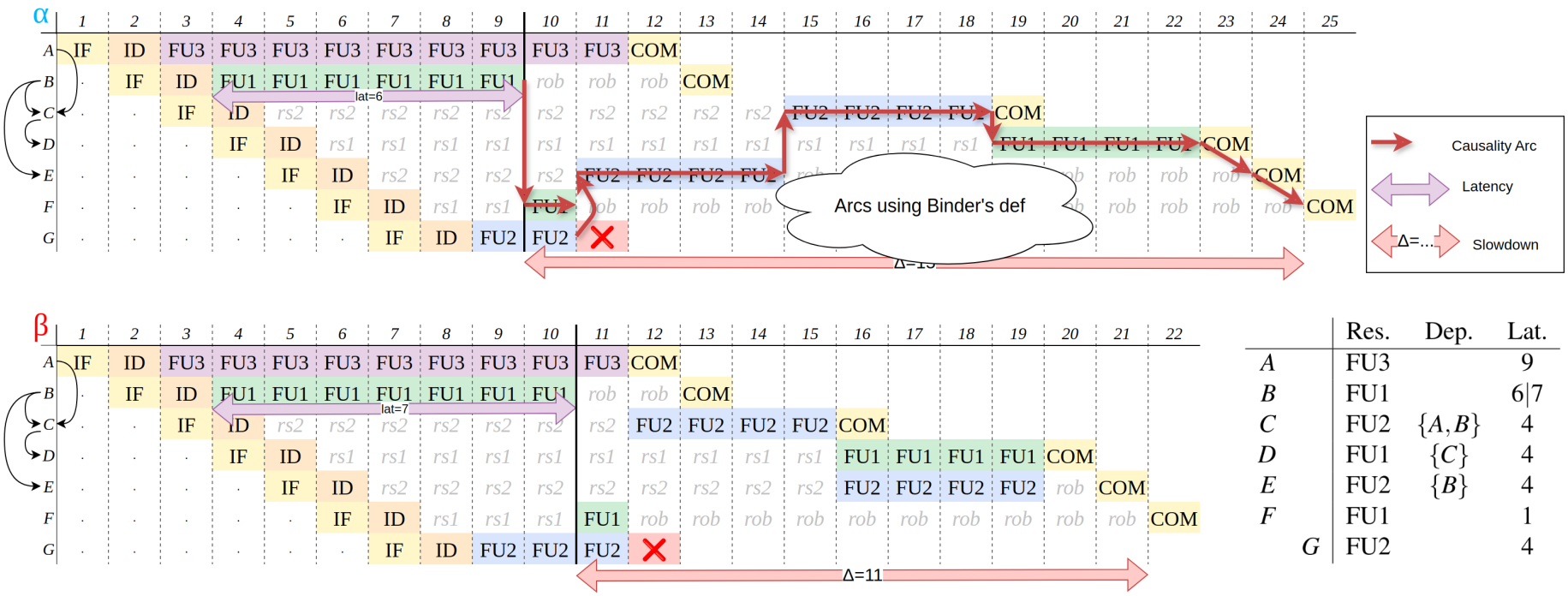
The problem with the "Acquisition" rule.



	Res.	Dep.	Lat.
A	FU3		9
B	FU1		6|7
C	FU2	$\{A,B\}$	4
D	FU1	$\{C\}$	4
E	FU2	$\{B\}$	4
F	FU1		1
G	FU2		4

FU release by squashing

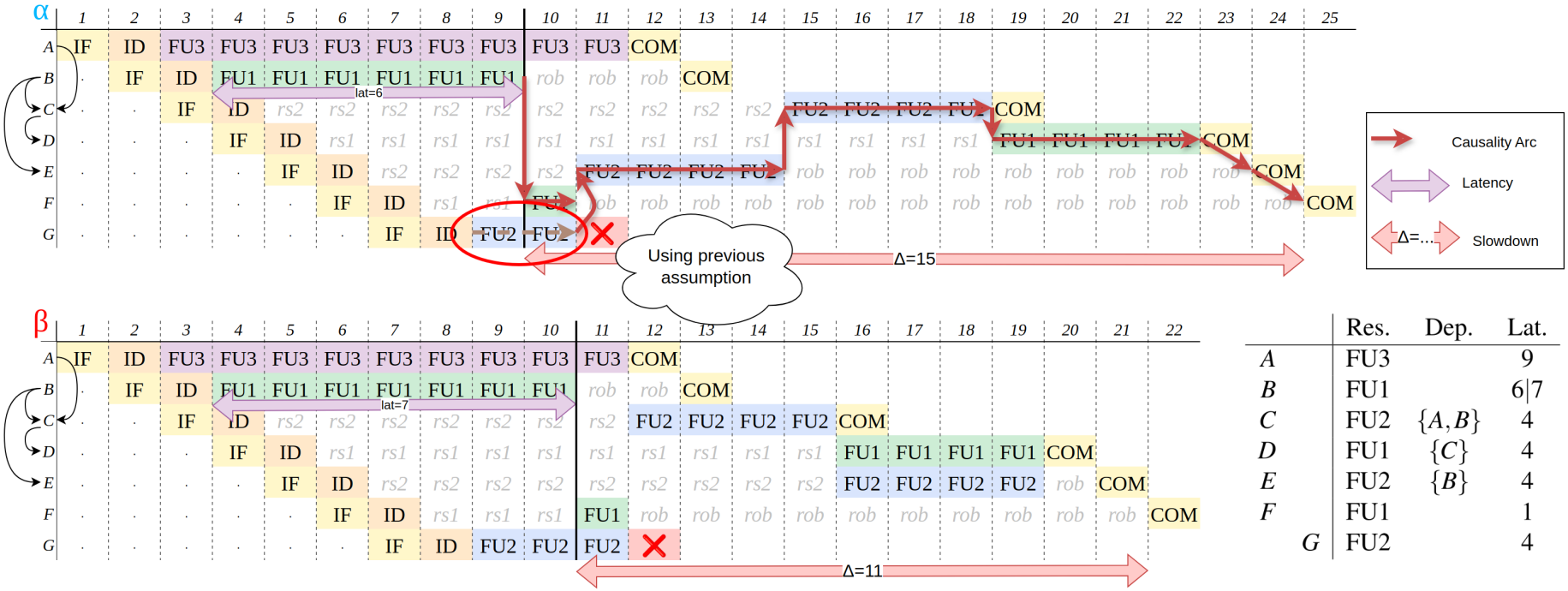
The problem with the "Acquisition" rule.



	Res.	Dep.	Lat.
A	FU3		9
B	FU1		6|7
C	FU2	$\{A,B\}$	4
D	FU1	$\{C\}$	4
E	FU2	$\{B\}$	4
F	FU1		1
G	FU2		4

FU release by squashing

The problem with the "Acquisition" rule.



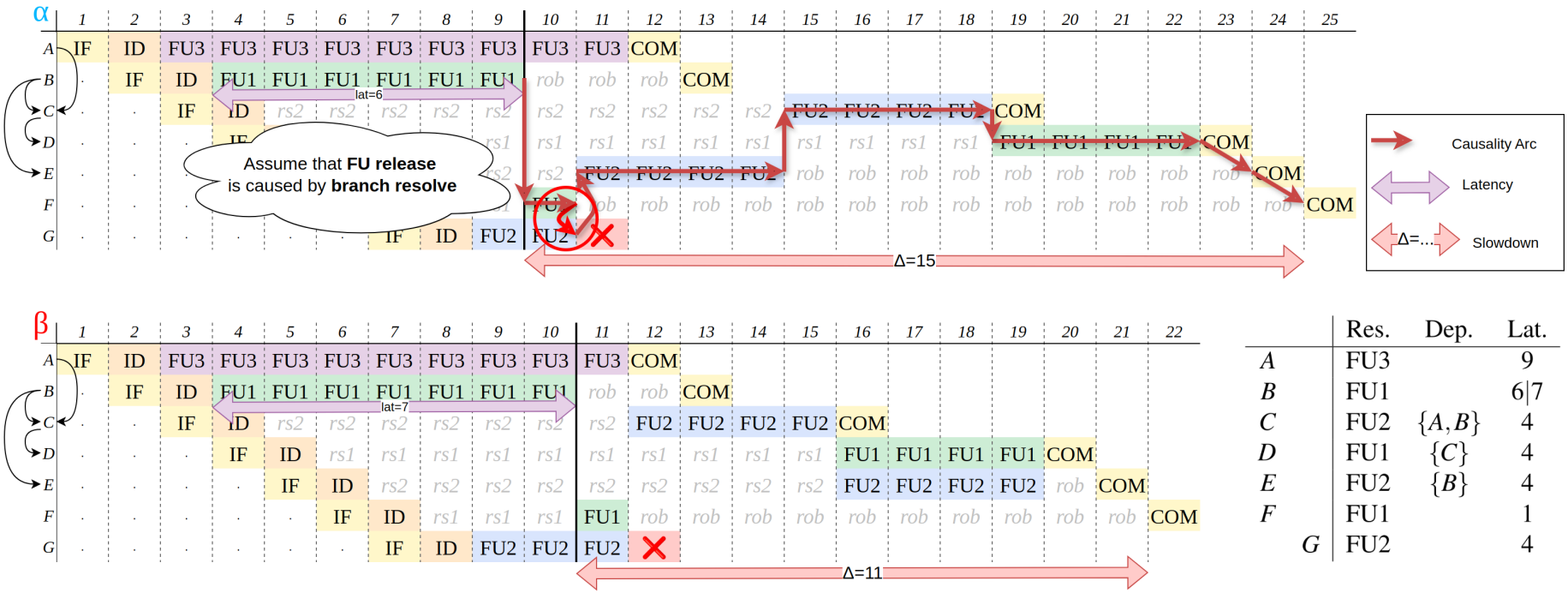
	Res.	Dep.	Lat.
A	FU3		9
B	FU1		6|7
C	FU2	{A,B}	4
D	FU1	{C}	4
E	FU2	{B}	4
F	FU1		1
G	FU2		4

"Acquisition" rule does not work

"The FU acquisition is always causal to the respective FU release."

FU release by squashing

The problem with the "Acquisition" rule.



	Res.	Dep.	Lat.
A	FU3		9
B	FU1		6|7
C	FU2	{A,B}	4
D	FU1	{C}	4
E	FU2	{B}	4
F	FU1		1
G	FU2		4

"Squashing" Rule

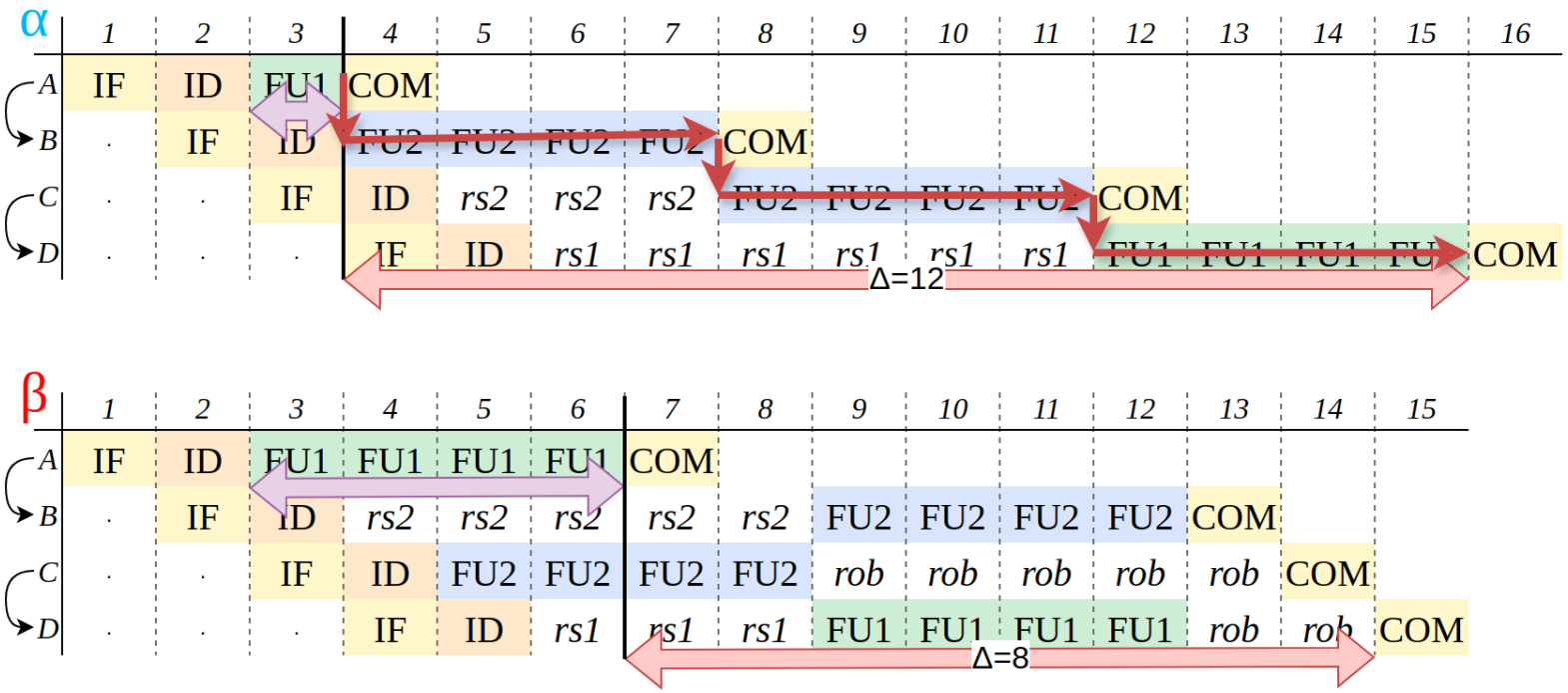
If FU is released as the result of squashing, the FU release is causal to branch resolve.

Results

- The definition of Binder et al. can be applied in branch prediction context with minimal adjustments.
- It is not clear how to modify the rule set to stay consistent.

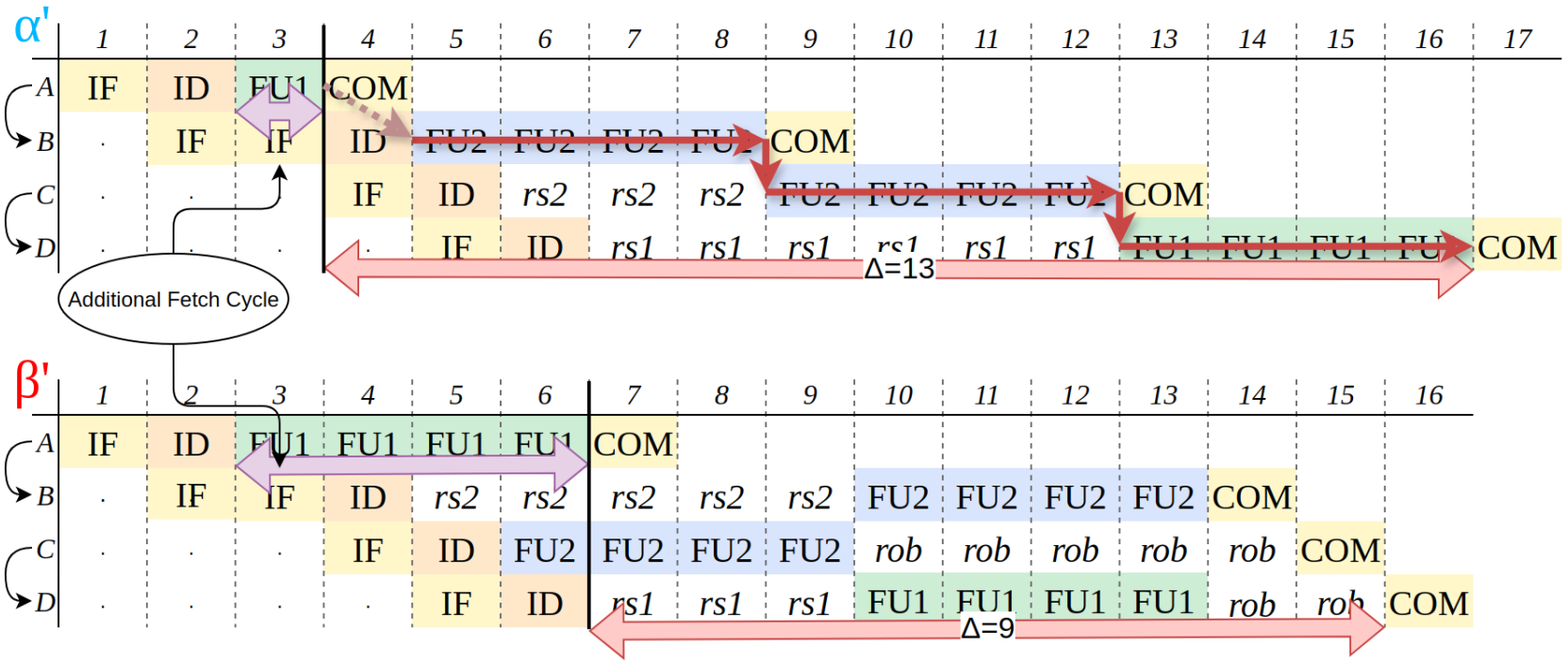
Gap Problem

Even the original definition by Binder et al. has a problem.



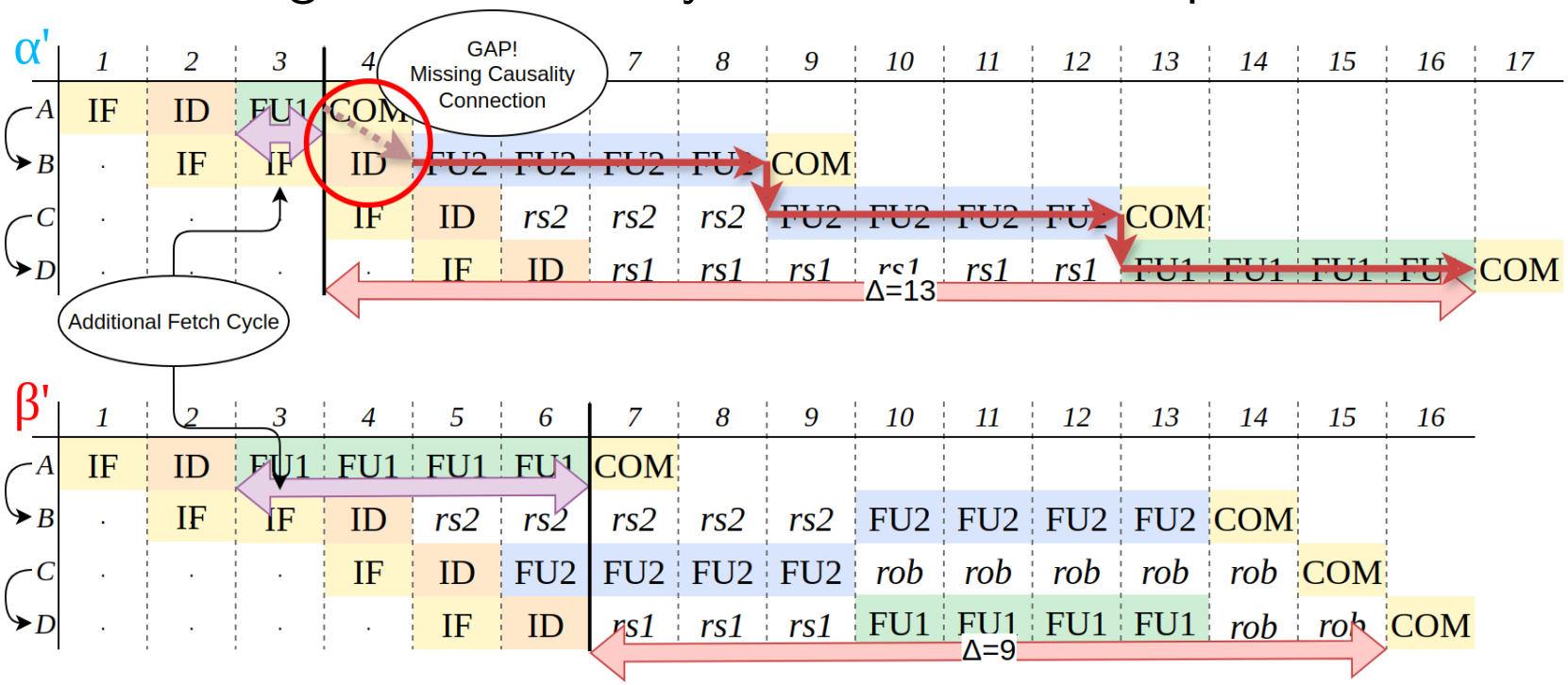
Gap Problem

Even the original definition by Binder et al. has a problem.



Gap Problem

Even the original definition by Binder et al. has a problem.



To conclude:

The problem is in the causality notion itself.

Conclusion

What was Done:

- Analyzed existing timing anomaly (TA) definitions; adopted Binder's causality-based approach.
- Developed a tool to systematically generate and analyze branch prediction-induced TAs.

Results:

- Binder's definition is adaptable, but controversial cases and a "gap problem" remain.

Future Work:

- A new causality definition based on event constraints to address these issues (work in progress).
- Study the impact of branch predictor state.

6 Backup Slides

Branch Prediction: possible implementations

Additional Hardware

- Pattern History Table (PHT)
- Branch Target Buffer (BTB)

Example: 2-bit counter

For each branch store a 4-state automaton in PHT

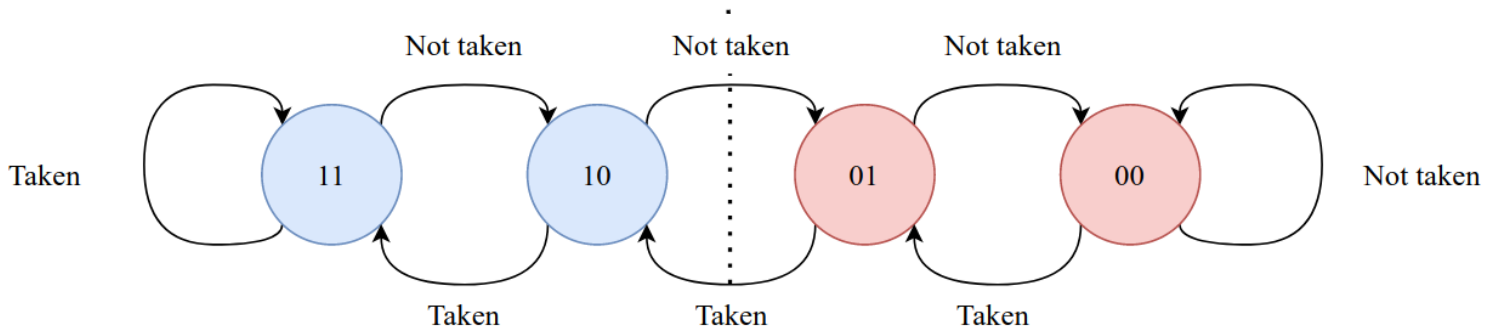
Strategies

Static: always take, never take, take backwards.

Dynamic:

- 1 or 2-Bit Counter
- Global or Local History
- Global share

and more...



Binder's definition: details

7 events for each instruction

1. $\uparrow$ IF
2. $\downarrow$ IF
3. $\uparrow$ ID
4. $\downarrow$ ID
5. $\uparrow$ FU
6. $\downarrow$ FU
7. COM

Event Time Dependency Graph

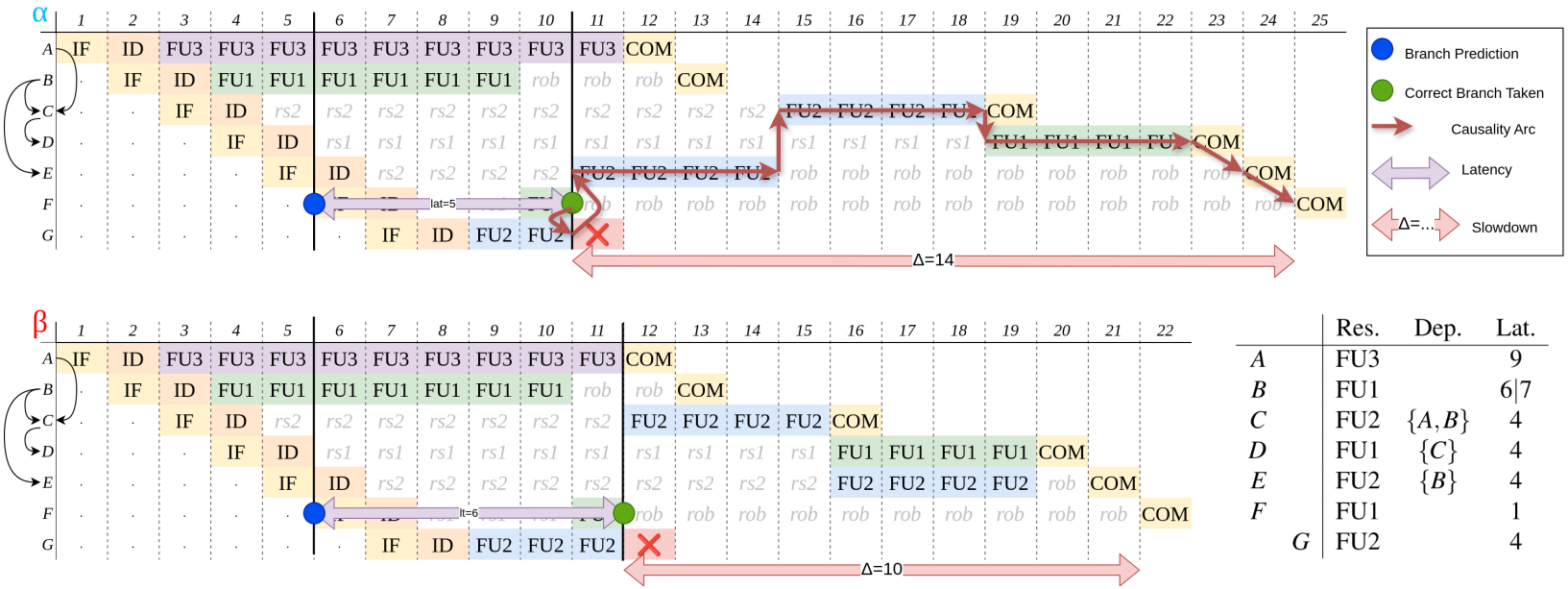
Each $(X \xrightarrow{t} Y)$ arc means at least t cycles must pass between event X and Y

5 Rules for ETDG arcs

1. **Order of Pipeline Stages** (acq. of next stage after rel. of previous)
2. **Resource Use** (cycles inside res.)
3. **Instruction Order** (for IF, ID, COM)
4. **Data Dependencies**
5. **Resource Contention** (using the same FU)

Causality Graph = ETDG with the most relevant arcs left.

Latency Impact on Branch Variation



	Res.	Dep.	Lat.
A	FU3		9
B	FU1		6|7
C	FU2	$\{A,B\}$	4
D	FU1	$\{C\}$	4
E	FU2	$\{B\}$	4
F	FU1		1
G	FU2		4

Misprediction region from CFG