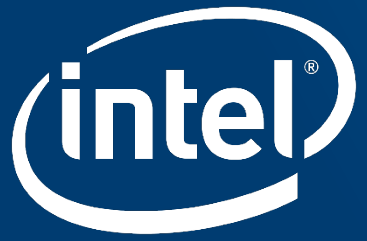


SAFARI

DSPATCH: DUAL SPATIAL PATTERN PREFETCHER

Rahul Bera[1], Anant V. Nori[1], Onur Mutlu[2], Sreenivas Subramoney[1]

[1]Processor Architecture Research Lab, Intel Labs
[2]ETH Zürich

Summary

Summary

Motivation:

- DRAM bandwidth increases with every generation
- Prefetchers need to adapt to effectively utilize this resource

Summary

Motivation:

- DRAM bandwidth increases with every generation
- Prefetchers need to adapt to effectively utilize this resource

Challenge:

- *Significantly boost* ***Coverage*** **AND**
- *Simultaneously optimize* for ***Accuracy***

Summary

Motivation:

- DRAM bandwidth increases with every generation
- Prefetchers need to adapt to effectively utilize this resource

Challenge:

- *Significantly boost* ***Coverage*** **AND**
- *Simultaneously optimize* for ***Accuracy***

Dual Spatial Pattern Prefetcher (DSPatch)

- Simultaneously learn ***two*** spatial bit-pattern representations of program accesses per page
  - Coverage-biased (CovP)
  - Accuracy-biased (AccP)

CovP

AccP

Summary

Motivation:

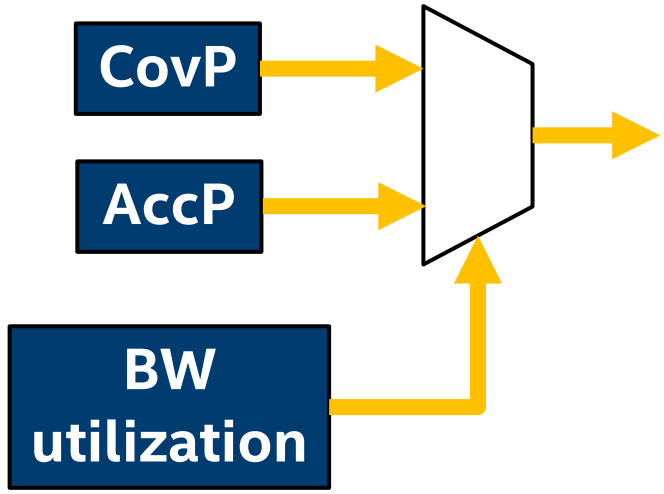
- DRAM bandwidth increases with every generation
- Prefetchers need to adapt to effectively utilize this resource

Challenge:

- *Significantly boost* ***Coverage*** **AND**
- *Simultaneously optimize* for ***Accuracy***

Dual Spatial Pattern Prefetcher (DSPatch)

- Simultaneously learn ***two*** spatial bit-pattern representations of program accesses per page
  - Coverage-biased (CovP)
  - Accuracy-biased (AccP)
- Predict ***one*** of the patterns based on current DRAM bandwidth headroom



Summary

Motivation:

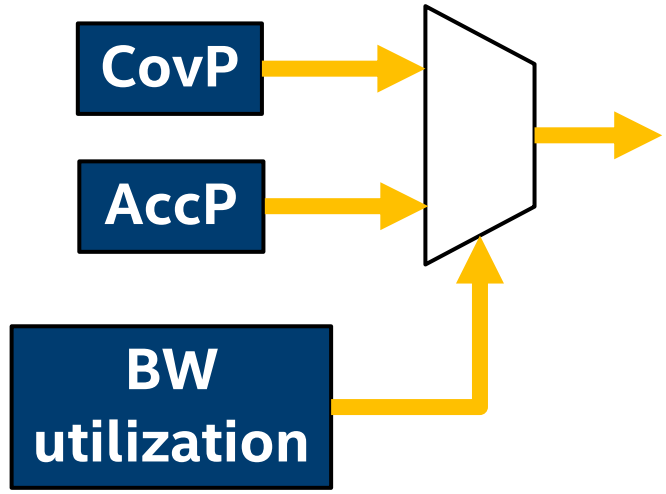
- DRAM bandwidth increases with every generation
- Prefetchers need to adapt to effectively utilize this resource

Challenge:

- *Significantly boost* ***Coverage*** **AND**
- *Simultaneously optimize* for ***Accuracy***

Dual Spatial Pattern Prefetcher (DSPatch)

- Simultaneously learn ***two*** spatial bit-pattern representations of program accesses per page
  - Coverage-biased (CovP)
  - Accuracy-biased (AccP)
- Predict ***one*** of the patterns based on current DRAM bandwidth headroom



- **6% average speedup** over baseline with PC-stride @ L1 and SPP @ L2
- **10% average speedup** when DRAM bandwidth is doubled
- **Only 3.6 KB** of hardware storage

Outline

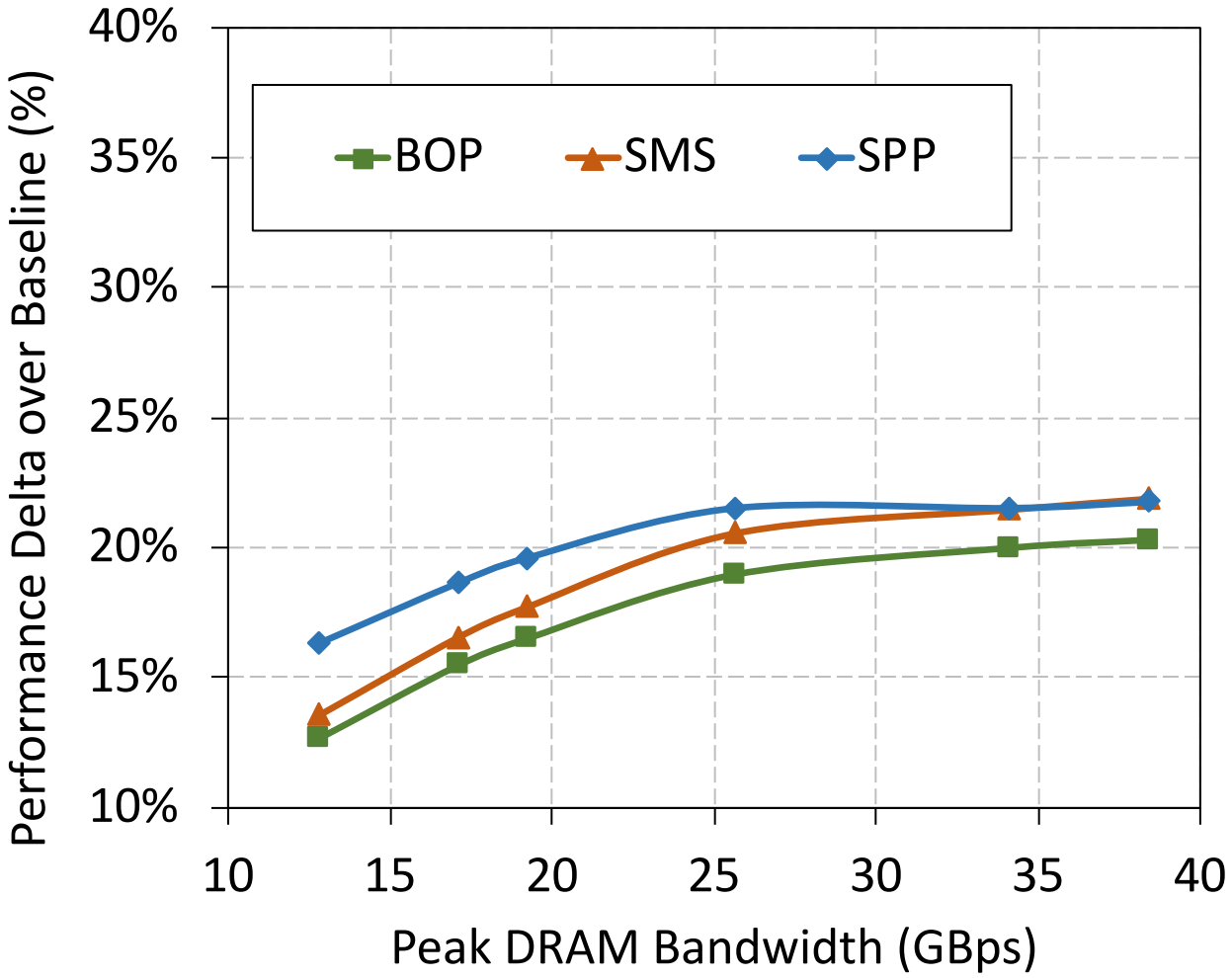
1. Motivation and Observations
2. Dual Spatial Pattern Prefetcher (DSPatch)
3. Evaluation
4. Conclusion

Motivation:

Current prefetchers' performance scales poorly with increasing DRAM bandwidth

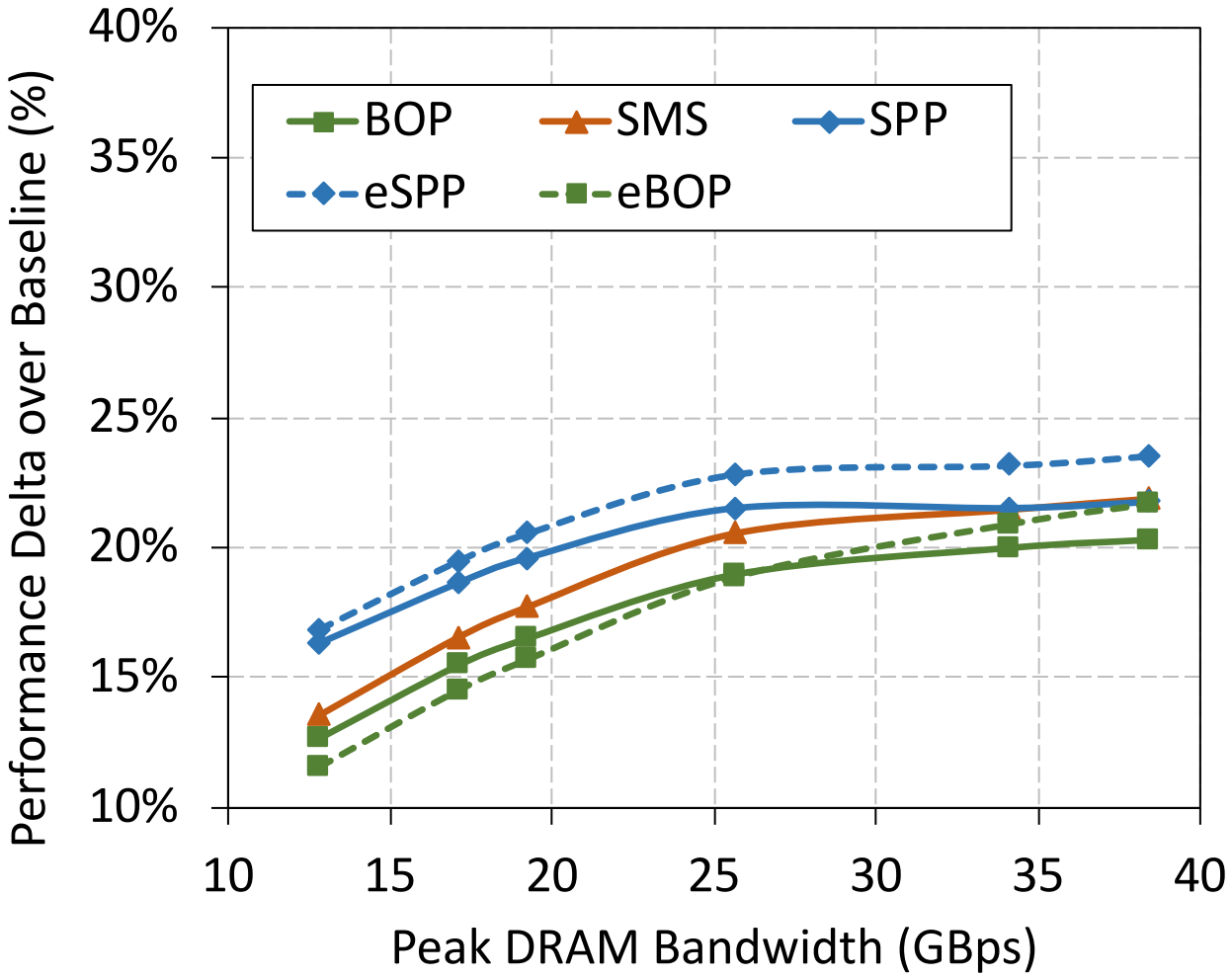
Motivation:

Current prefetchers' performance scales poorly with increasing DRAM bandwidth



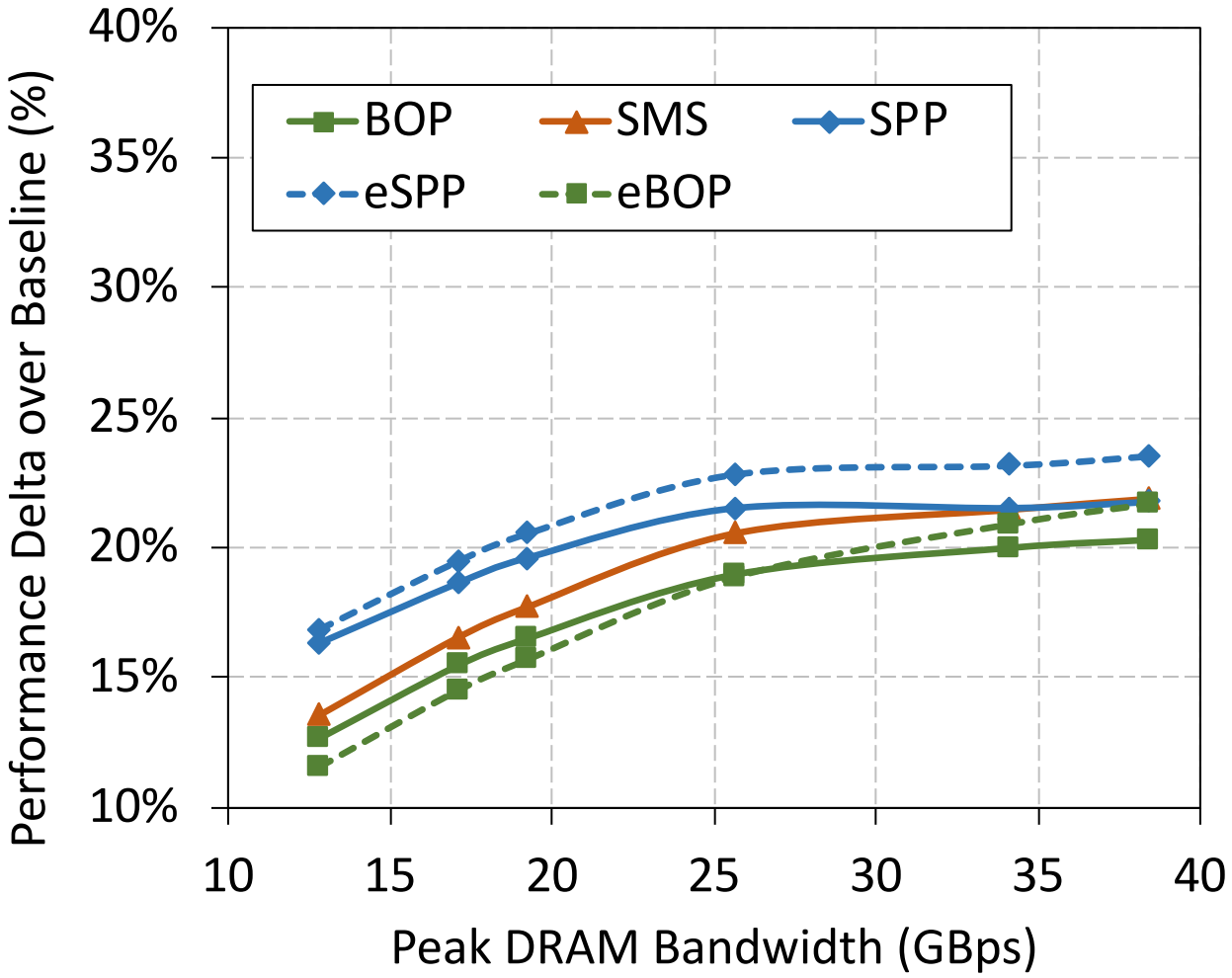
Motivation:

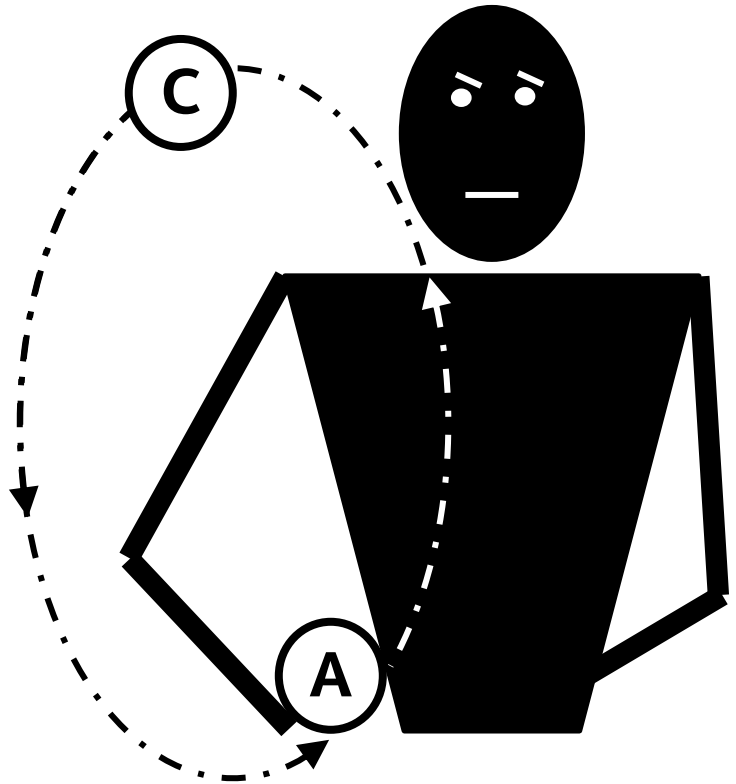
Current prefetchers' performance scales poorly with increasing DRAM bandwidth



Motivation:

Current prefetchers' performance scales poorly with increasing DRAM bandwidth

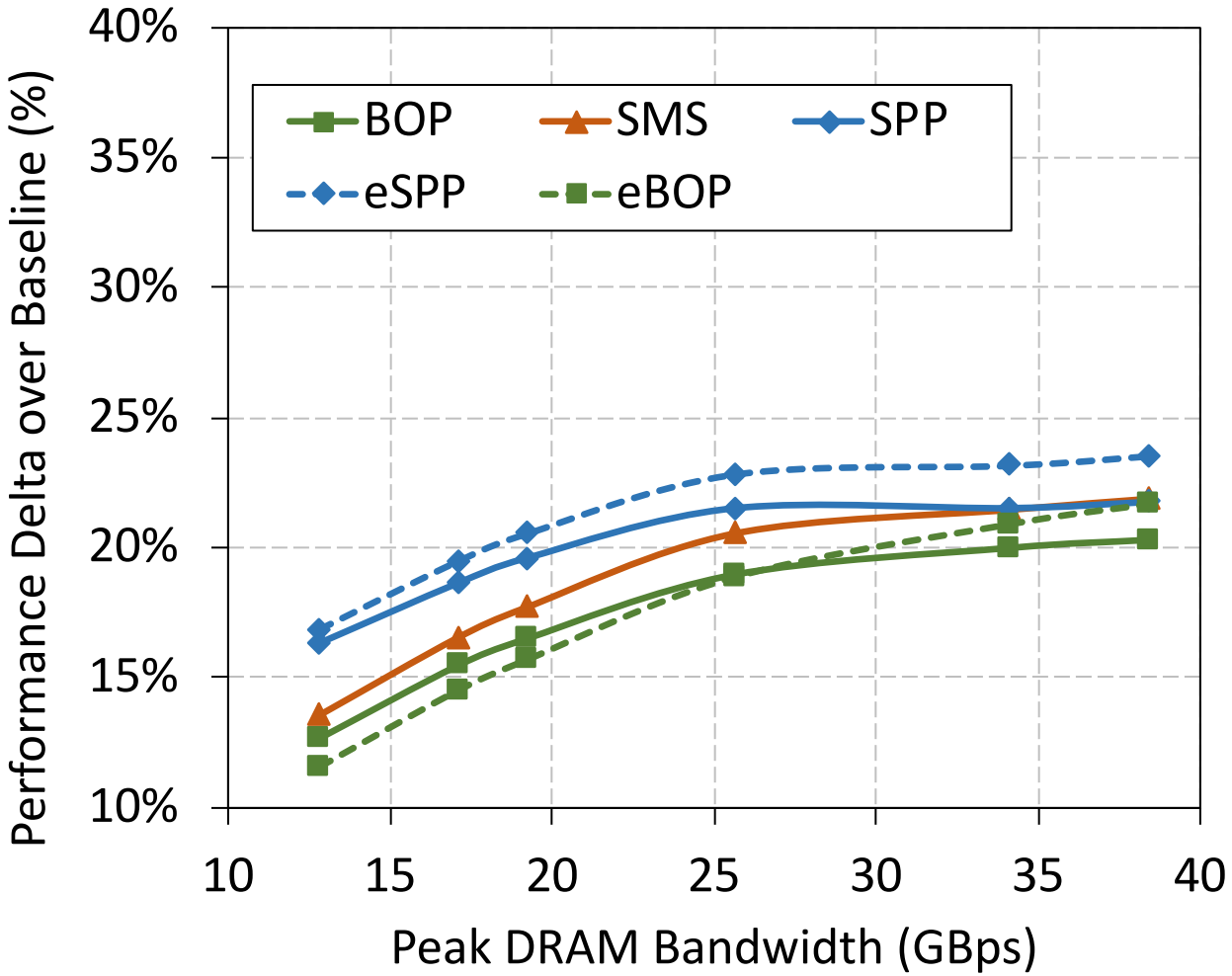


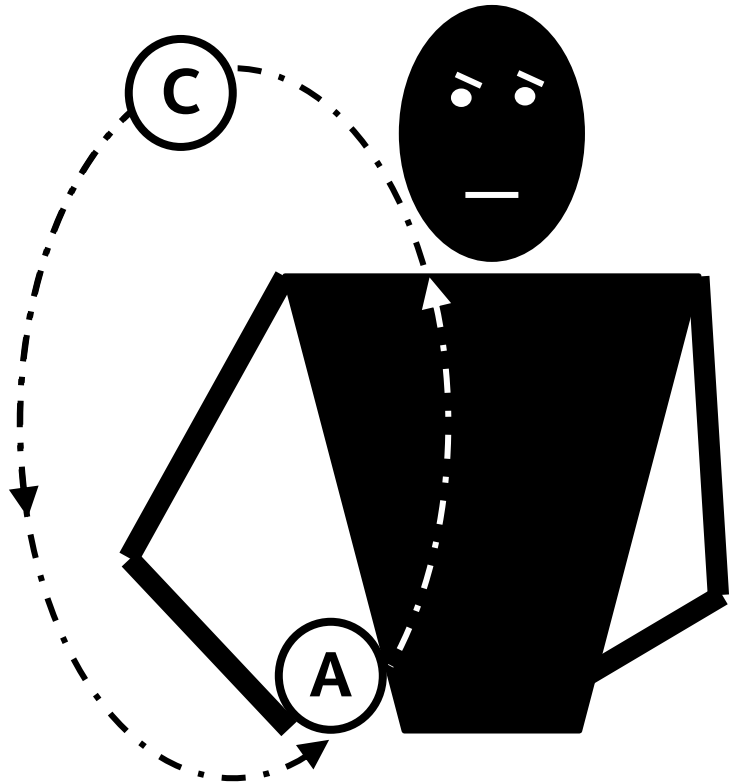


Traditional prefetcher design tradeoff between Coverage and Accuracy

Motivation:

Current prefetchers' performance scales poorly with increasing DRAM bandwidth





Traditional prefetcher design tradeoff between Coverage and Accuracy

Limits ability to significantly boost Coverage when needed

Observation 1

Anchored spatial bit-patterns well suited to capture spatial access patterns

Observation 1

Anchored spatial bit-patterns well suited to capture spatial access patterns

Cacheline offset seq. within a page

A: *[02]-[06]-[05]-[12]-[13]*

B: *[01]-[05]-[04]-[11]-[12]*

C: *[01]-[05]-[11]-[04]-[12]*

D: *[01]-[11]-[05]-[04]-[12]*

E: *[01]-[11]-[04]-[05]-[12]*

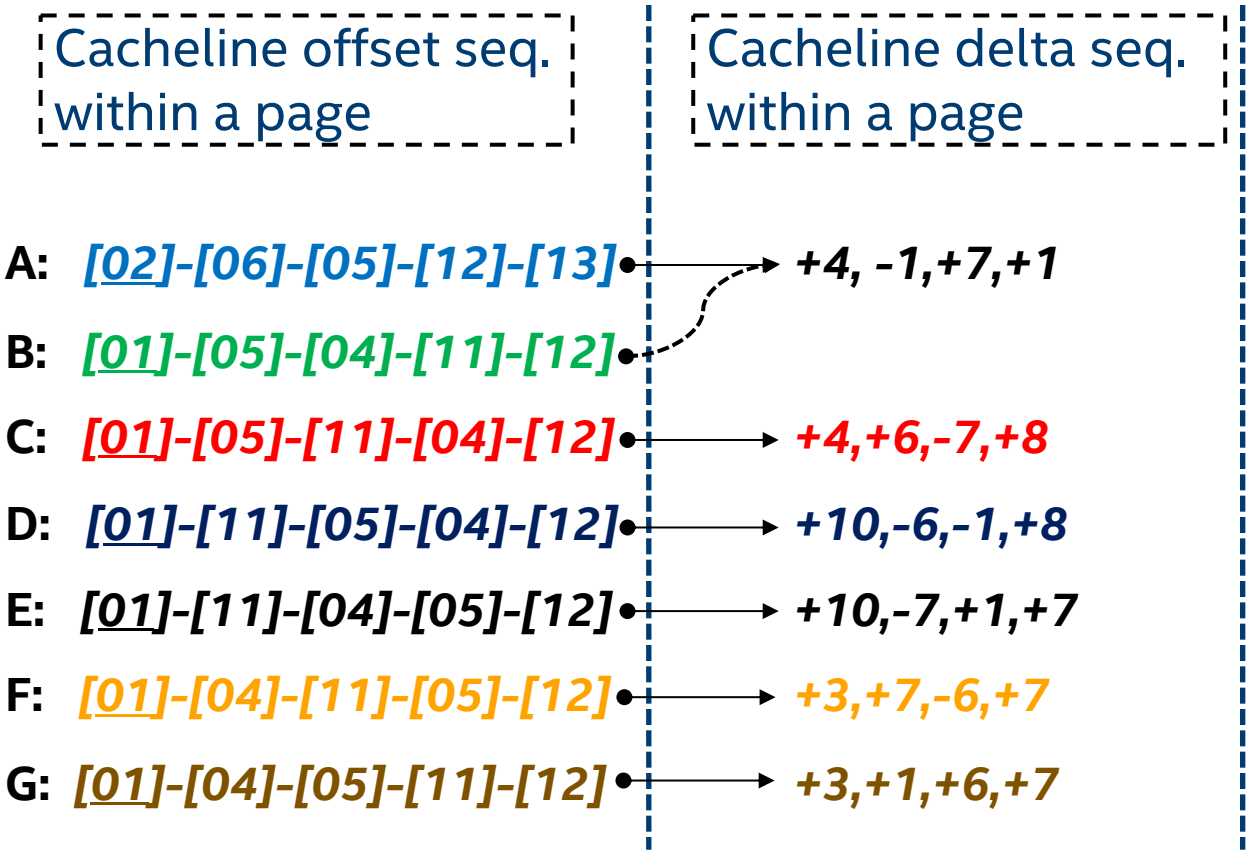
F: *[01]-[04]-[11]-[05]-[12]*

G: *[01]-[04]-[05]-[11]-[12]*

"_" marks the first (triggering) access to the page

Observation 1

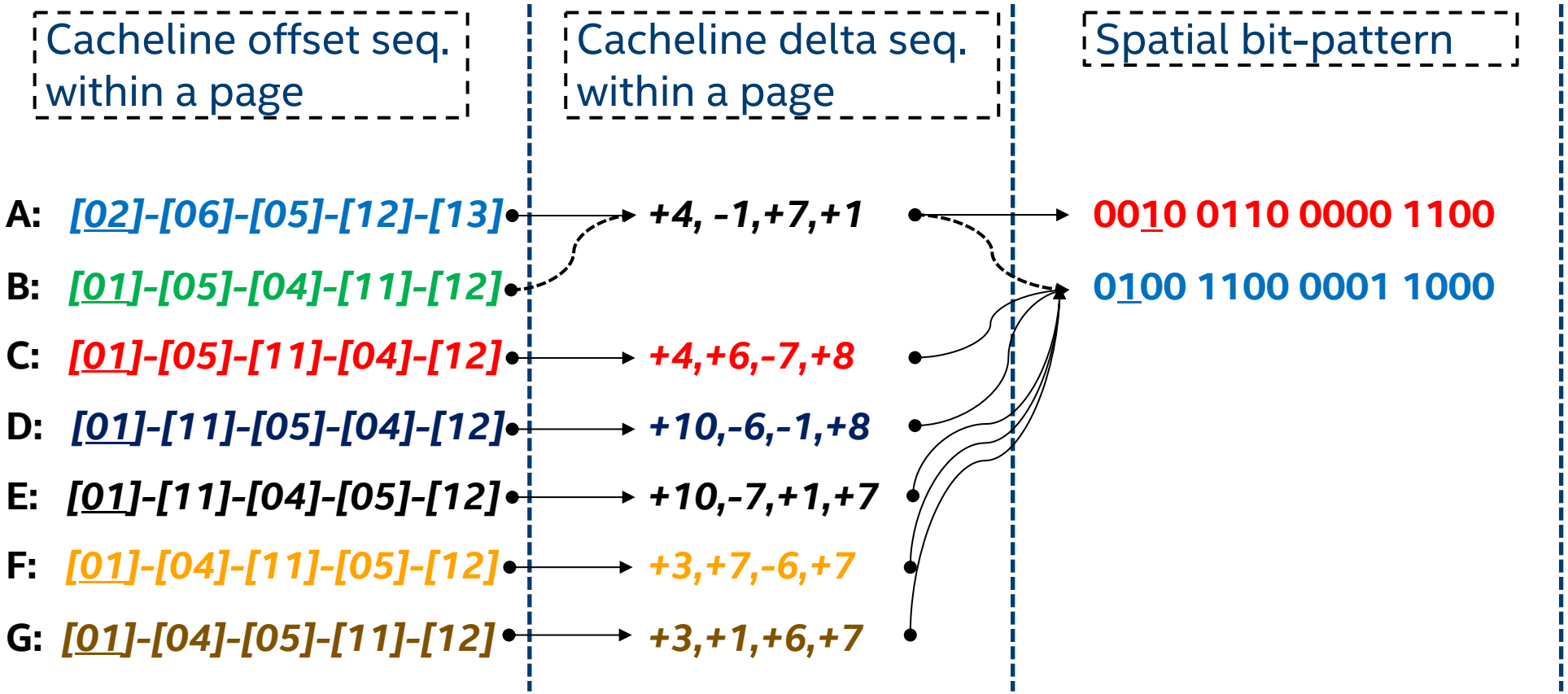
Anchored spatial bit-patterns well suited to capture spatial access patterns



"_" marks the first (triggering) access to the page

Observation 1

Anchored spatial bit-patterns well suited to capture spatial access patterns

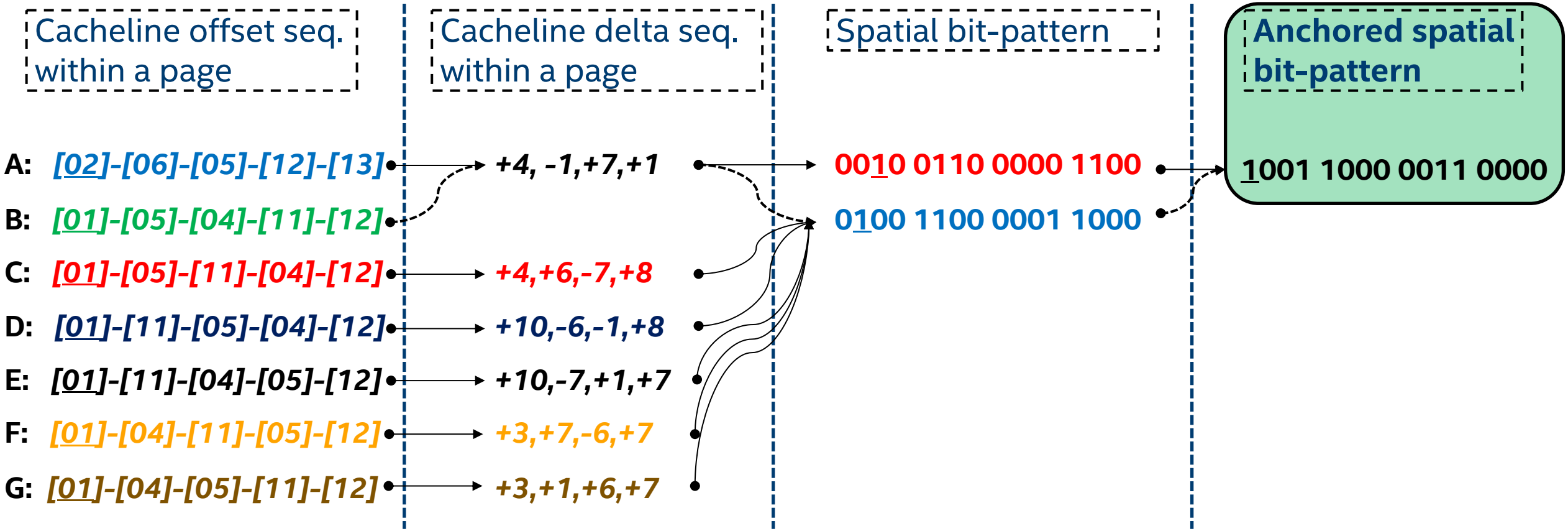


"_" marks the first (triggering) access to the page

Spatial bit-patterns subsume any small temporal variability in accesses

Observation 1

Anchored spatial bit-patterns well suited to capture spatial access patterns



"_" marks the first (triggering) access to the page

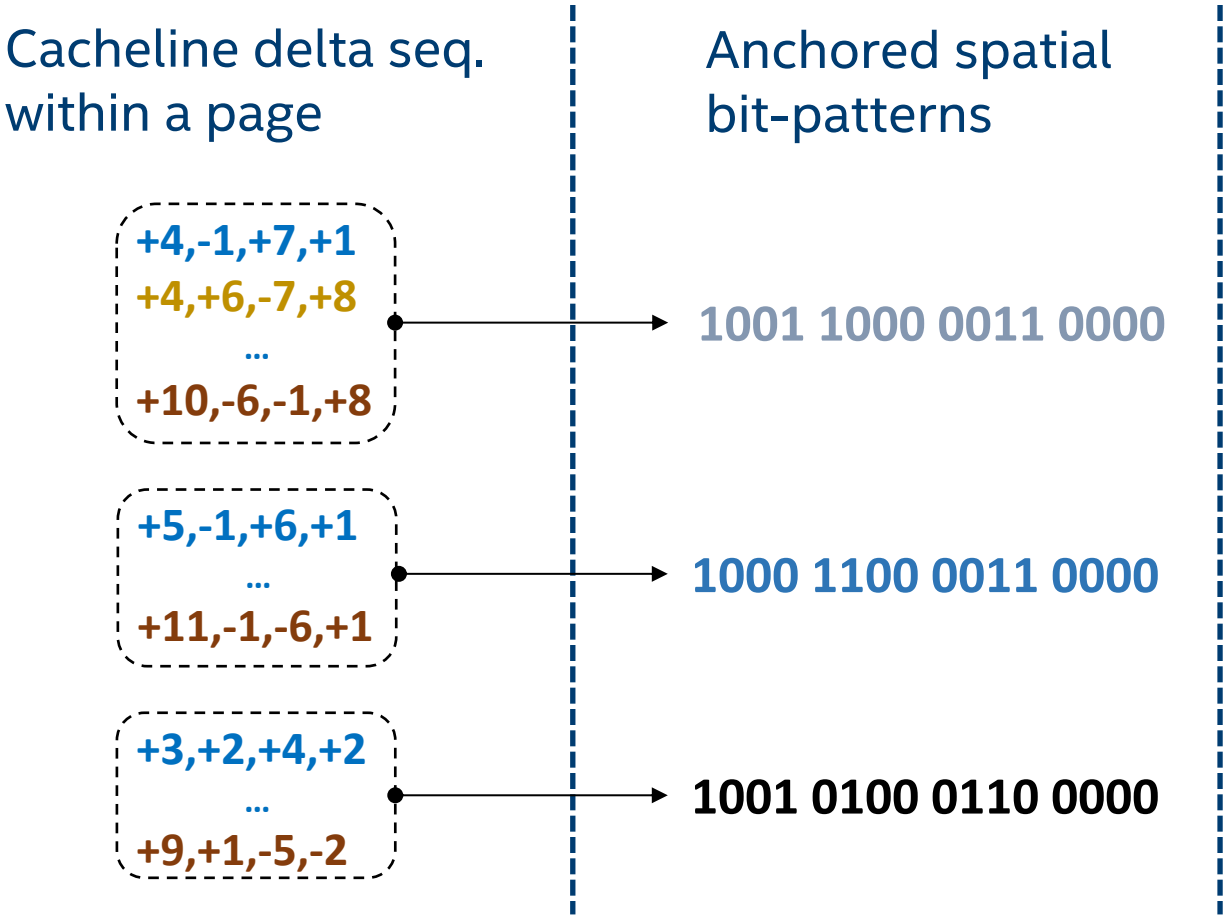
Spatial bit-patterns subsume any small temporal variability in accesses
***Anchored* spatial bit-patterns capture all "global" deltas from the triggering access**

Observation 2

Anchored spatial bit-patterns allow ***modulation*** for Coverage versus Accuracy

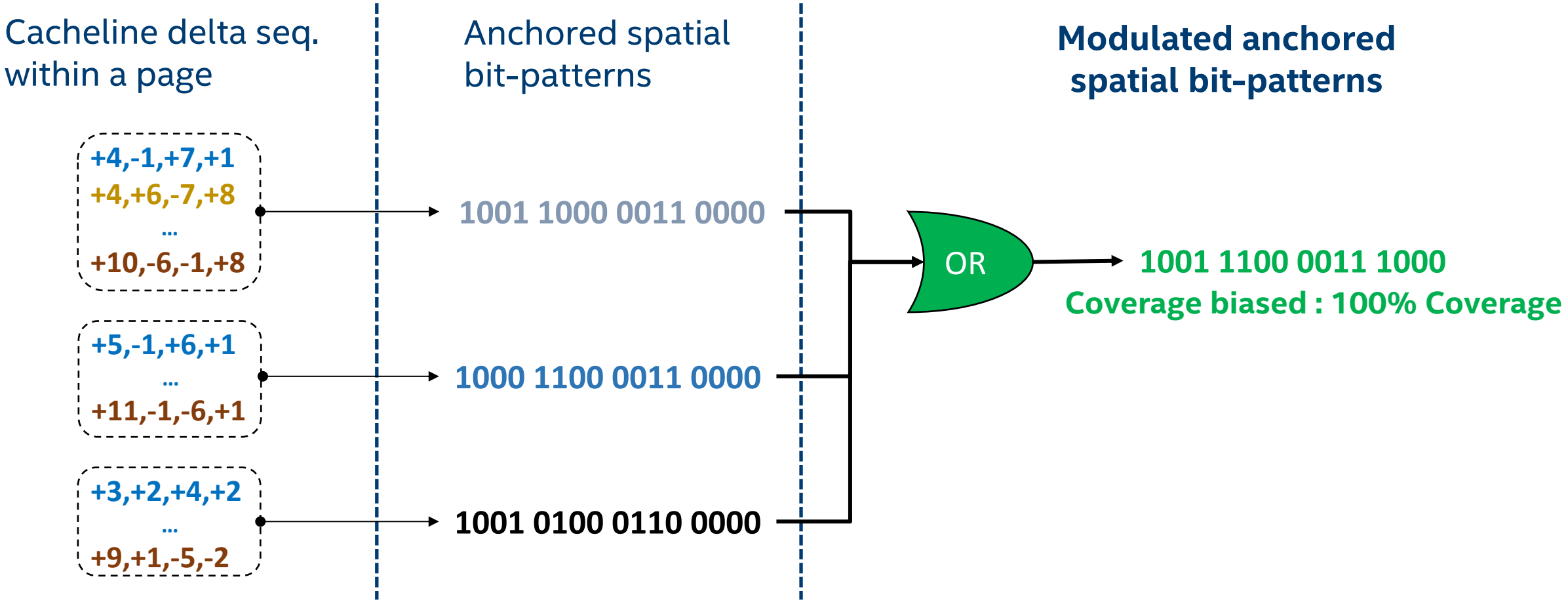
Observation 2

Anchored spatial bit-patterns allow ***modulation*** for Coverage versus Accuracy



Observation 2

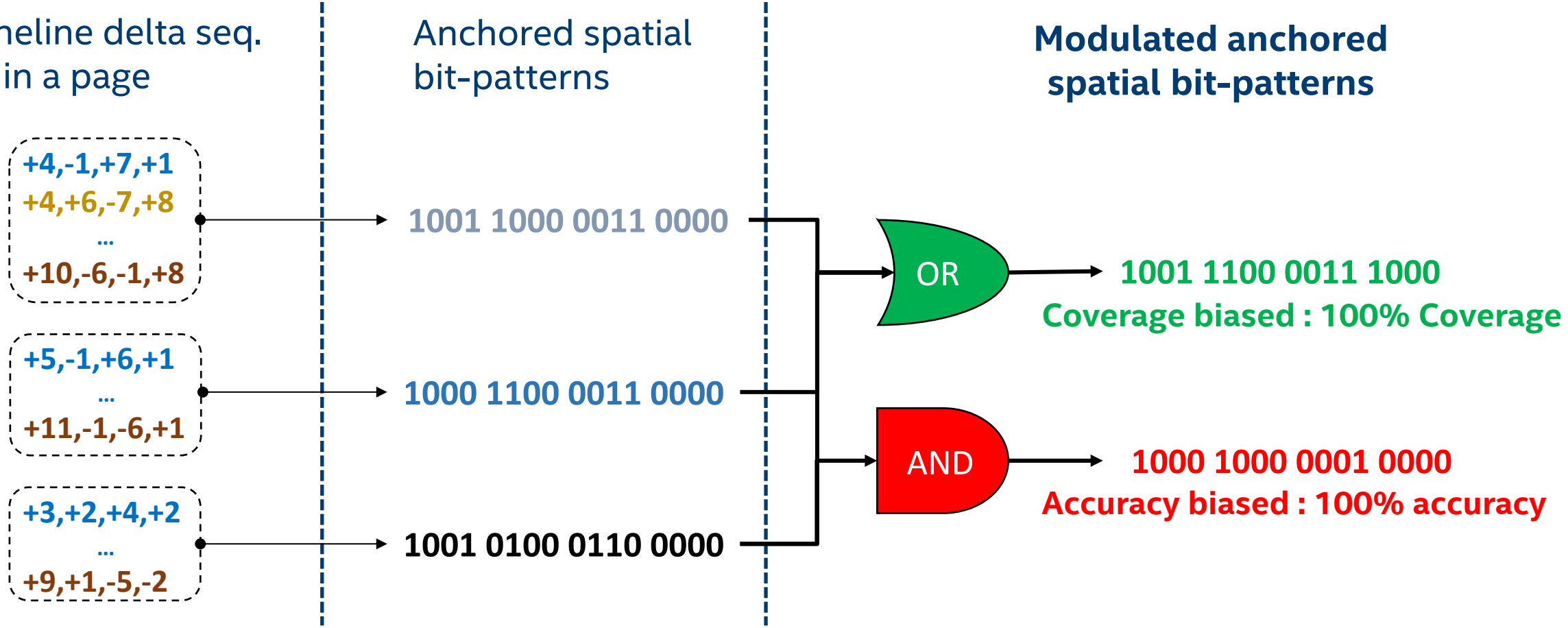
Anchored spatial bit-patterns allow *modulation* for Coverage versus Accuracy



Bit-wise *OR* modulation ⇒ Biased towards *Coverage*

Observation 2

Anchored spatial bit-patterns allow ***modulation*** for Coverage versus Accuracy



Bit-wise *OR* modulation ⇒ Biased towards *Coverage*
Bit-wise *AND* modulation ⇒ Biased towards *Accuracy*

Outline

1. Motivation and Observations
2. Dual Spatial Pattern Prefetcher (DSPatch)
3. Evaluation
4. Conclusion

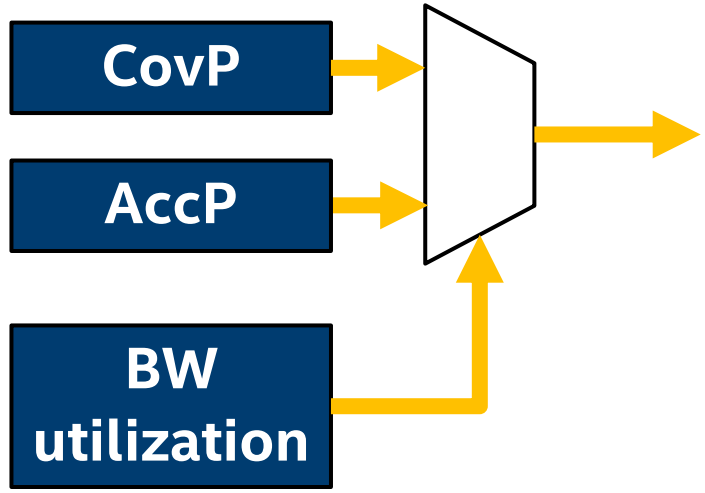
DSPatch: Key Idea

DSPatch: Key Idea

- Simultaneously learn ***two*** modulated spatial bit-pattern representations of program accesses
  - Coverage-Biased (CovP)
  - Accuracy-Biased (AccP)

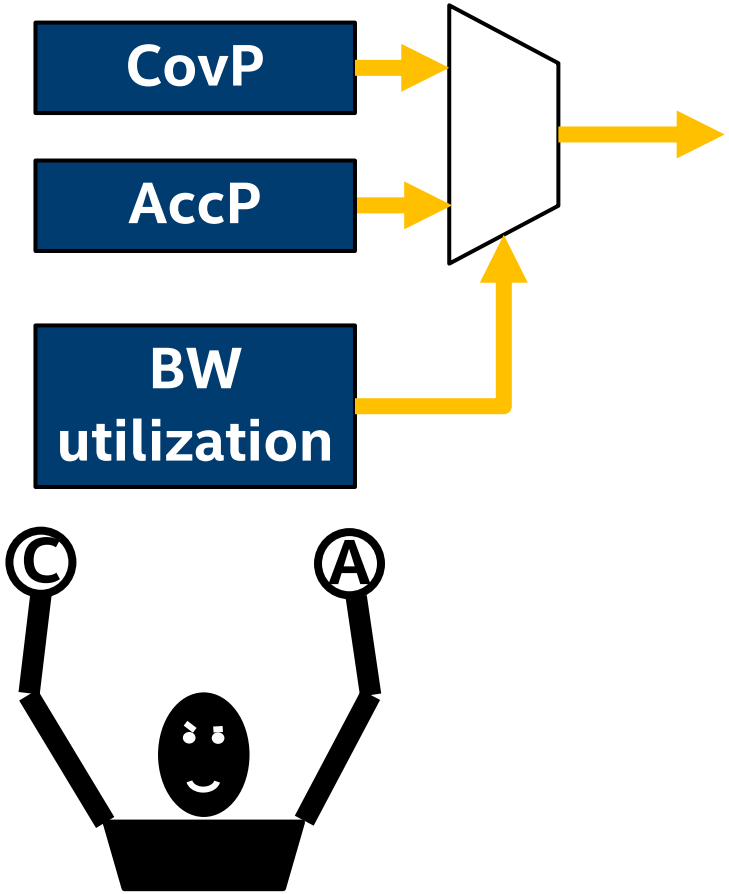
DSPatch: Key Idea

- Simultaneously learn ***two*** modulated spatial bit-pattern representations of program accesses
  - Coverage-Biased (CovP)
  - Accuracy-Biased (AccP)
- Predict ***one*** of the patterns based on current DRAM bandwidth headroom



DSPatch: Key Idea

- Simultaneously learn ***two*** modulated spatial bit-pattern representations of program accesses
  - Coverage-Biased (CovP)
  - Accuracy-Biased (AccP)
- Predict ***one*** of the patterns based on current DRAM bandwidth headroom



Optimize simultaneously for *both*:
- **Significantly higher Coverage**
- **High Accuracy**

DSPatch: Overall Design

DSPatch: Overall Design

Page Buffer (PB)

Page	Trigger PC	Bit-pattern
0x65	0x7ffecca	0010 1100 0001 0010
0x66	0x7ff8980	0001 0010 0011 0001
.	.	.
.	.	.
.	.	.

DSPatch: Overall Design

Page Buffer (PB)

Page	Trigger PC	Bit-pattern
0x65	0x7ffecca	0010 1100 0001 0010
0x66	0x7ff8980	0001 0010 0011 0001
⋮	⋮	⋮

Signature Prediction Table (SPT)

PC Signature	CovP	AccP
0x7ffecca	1011 0011 1110 0011	0001 0000 0011 0010
⋮	⋮	⋮

DSPatch: Overall Design

Page Buffer (PB)

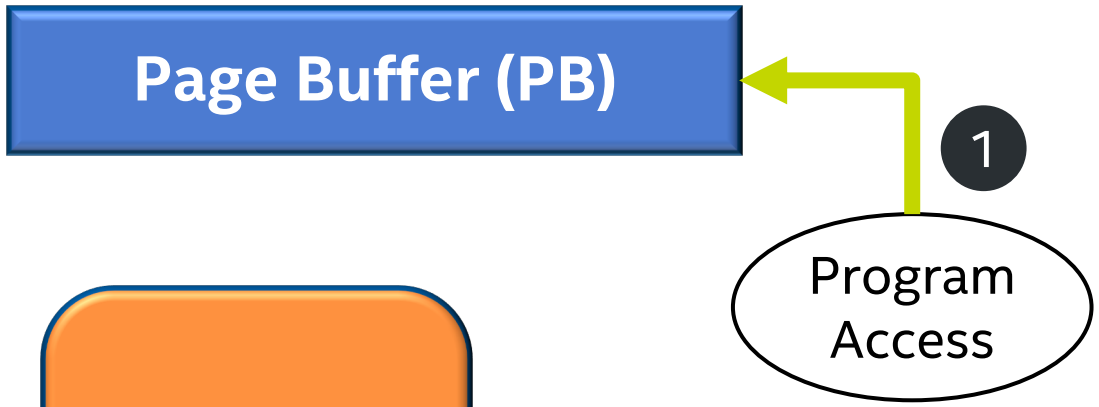
Program Access

Signature Prediction Table (SPT)

Page	Trigger PC	Bit-pattern
0x65	0x7ffecca	0010 1100 0001 0010
0x66	0x7ff8980	0001 0010 0011 0001
⋮	⋮	⋮

PC Signature	CovP	AccP
0x7ffecca	1011 0011 1110 0011	0001 0000 0011 0010
⋮	⋮	⋮

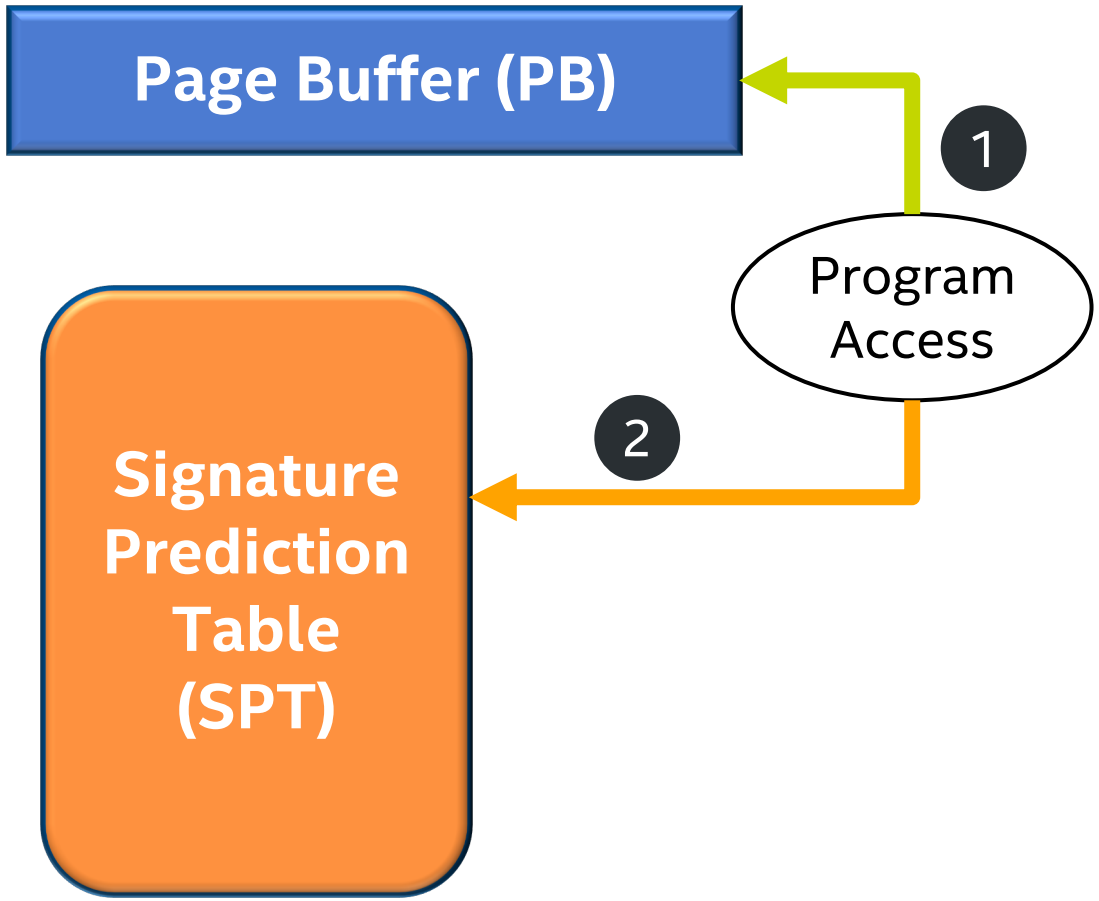
DSPatch: Overall Design



Page	Trigger PC	Bit-pattern
0x65	0x7ffecca	0010 1100 0001 0010
0x66	0x7ff8980	0001 0010 0011 0001
⋮	⋮	⋮

PC Signature	CovP	AccP
0x7ffecca	1011 0011 1110 0011	0001 0000 0011 0010
⋮	⋮	⋮

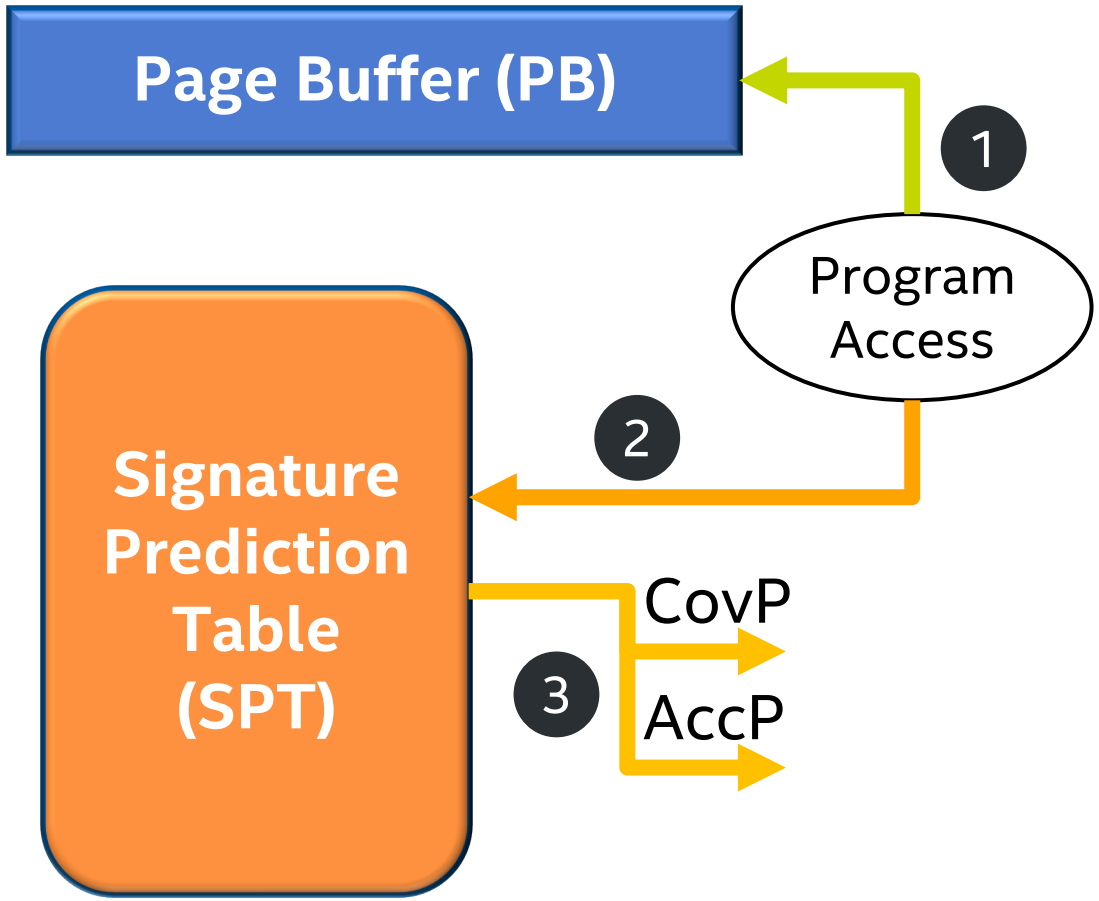
DSPatch: Overall Design



Page	Trigger PC	Bit-pattern
0x65	0x7ffecca	0010 1100 0001 0010
0x66	0x7ff8980	0001 0010 0011 0001
⋮	⋮	⋮

PC Signature	CovP	AccP
0x7ffecca	1011 0011 1110 0011	0001 0000 0011 0010
⋮	⋮	⋮

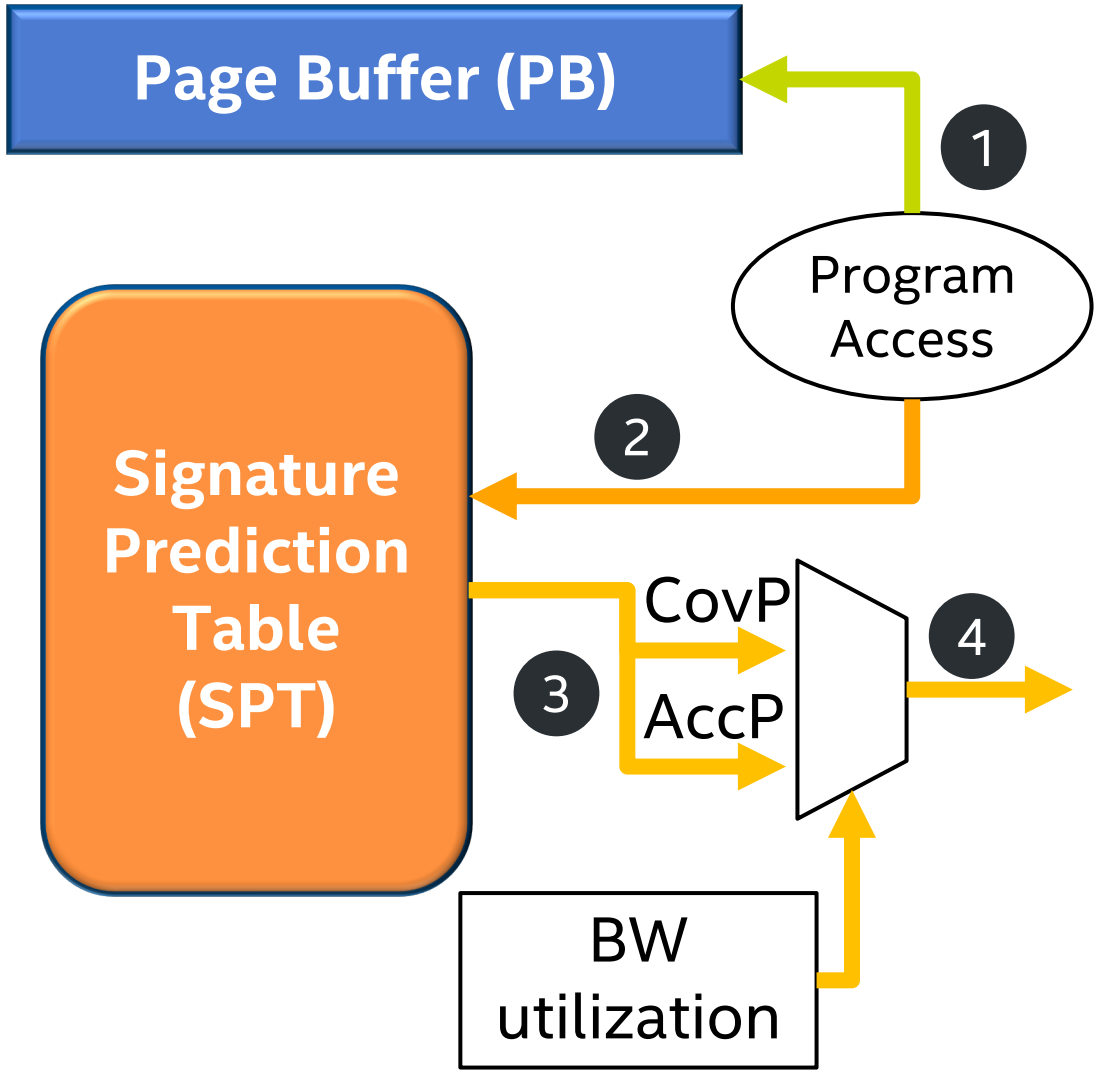
DSPatch: Overall Design



Page	Trigger PC	Bit-pattern
0x65	0x7ffecca	0010 1100 0001 0010
0x66	0x7ff8980	0001 0010 0011 0001
	. . .	. . .

PC Signature	CovP	AccP
0x7ffecca	1011 0011 1110 0011	0001 0000 0011 0010
	. . .	. . .

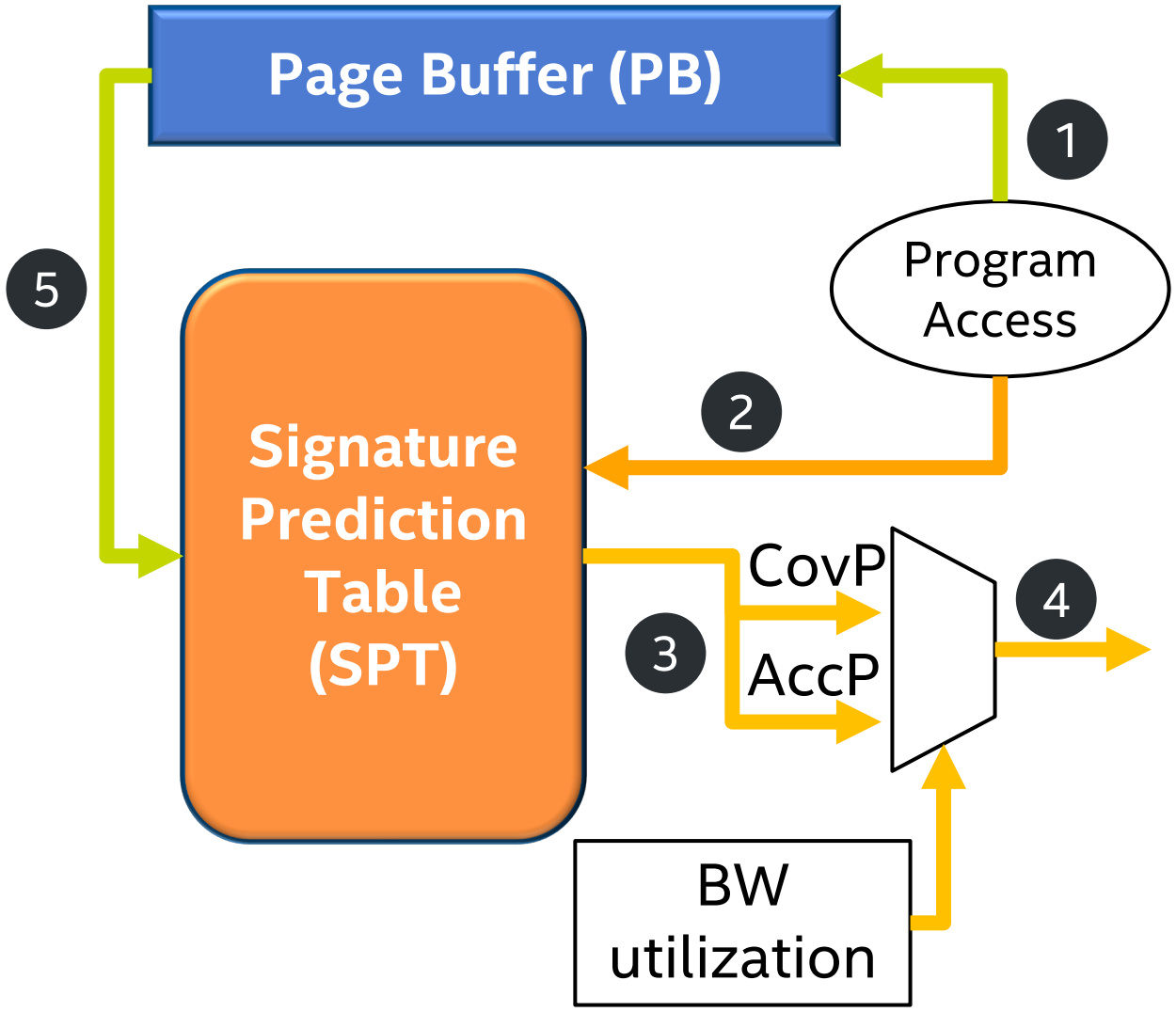
DSPatch: Overall Design



Page	Trigger PC	Bit-pattern
0x65	0x7ffecca	0010 1100 0001 0010
0x66	0x7ff8980	0001 0010 0011 0001
	. . .	. . .

PC Signature	CovP	AccP
0x7ffecca	1011 0011 1110 0011	0001 0000 0011 0010
	. . .	. . .

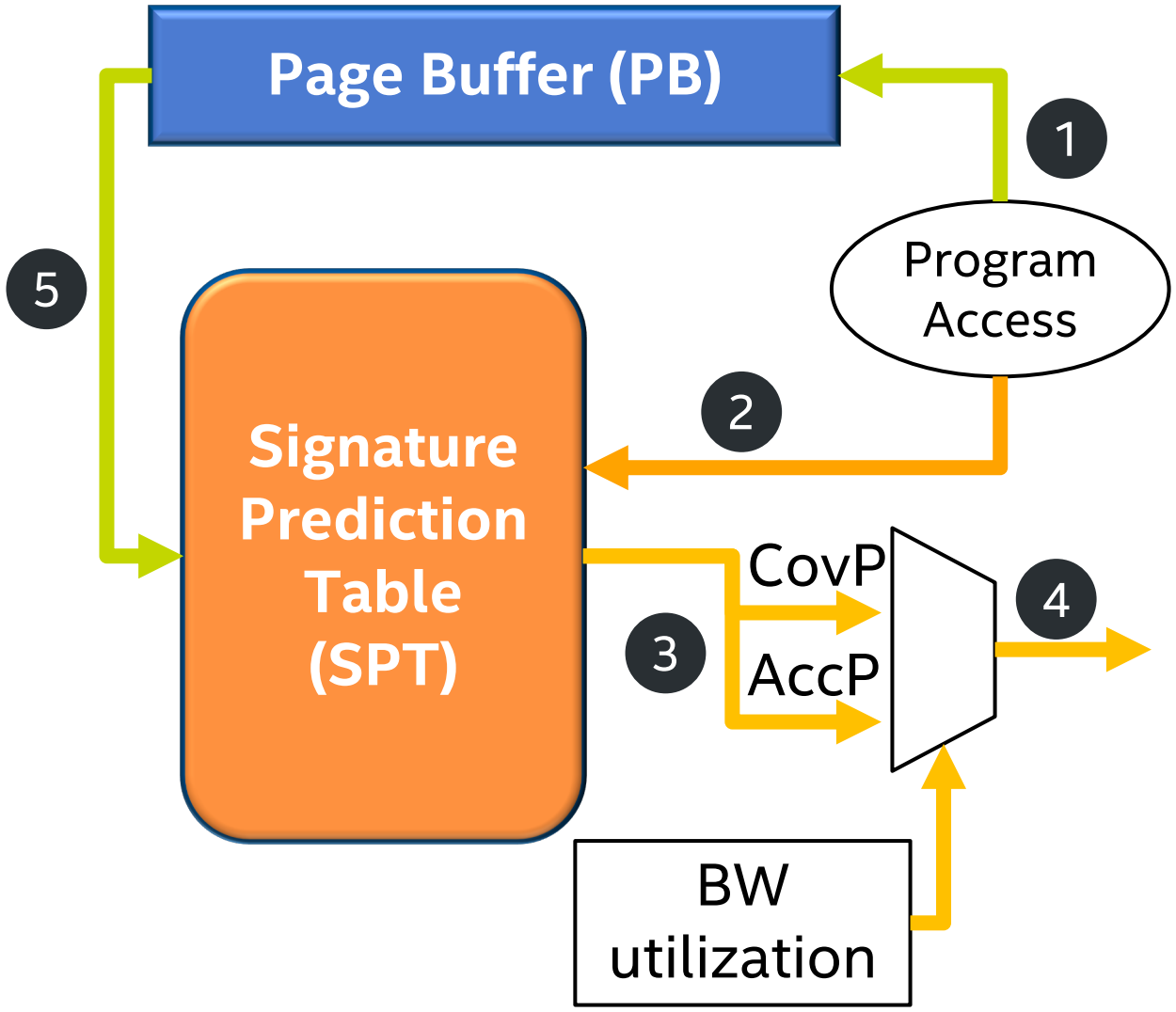
DSPatch: Overall Design



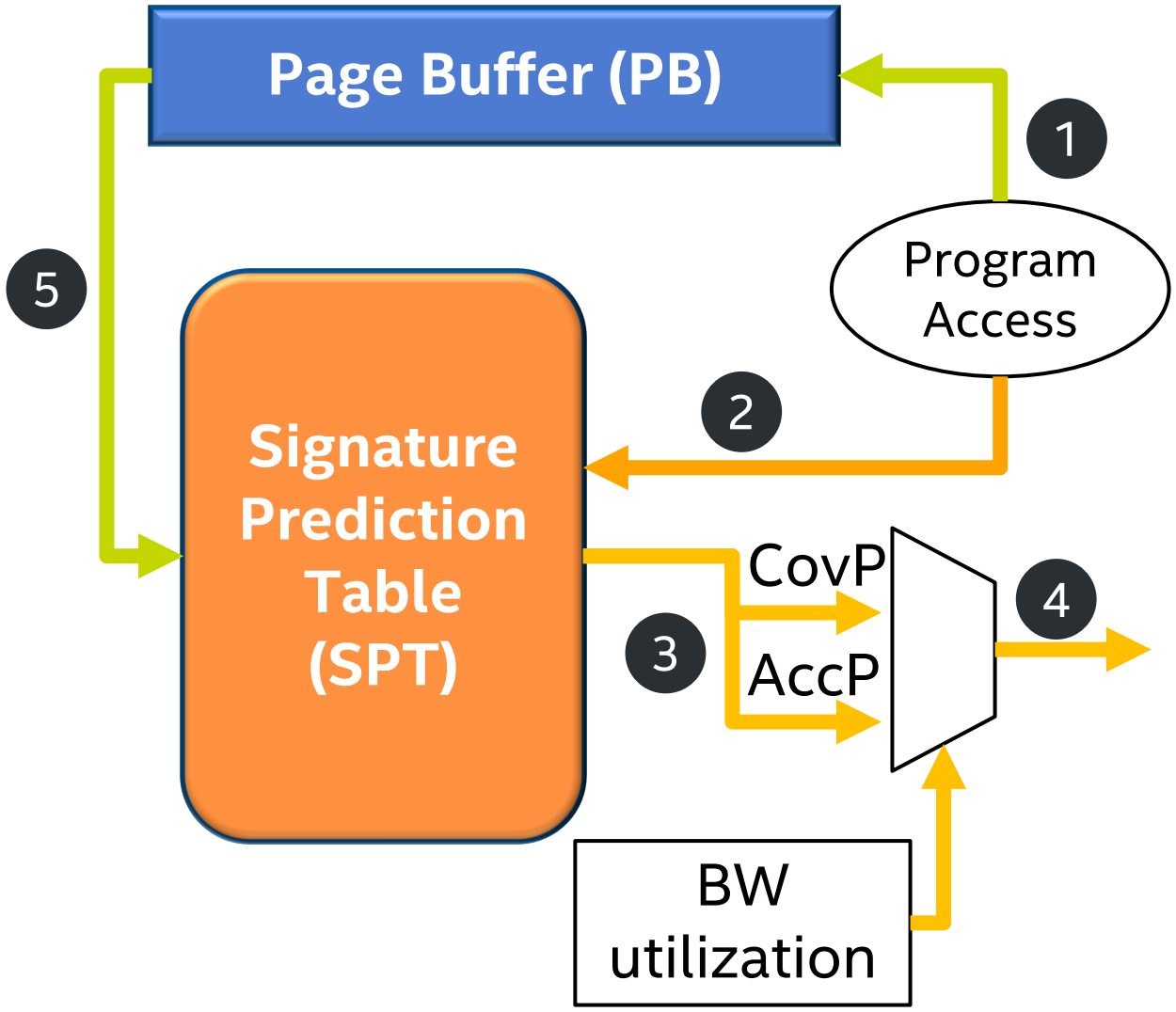
Page	Trigger PC	Bit-pattern
0x65	0x7ffecca	0010 1100 0001 0010
0x66	0x7ff8980	0001 0010 0011 0001
	. . .	. . .

PC Signature	CovP	AccP
0x7ffecca	1011 0011 1110 0011	0001 0000 0011 0010
	. . .	. . .

DSPatch: Quantify Coverage, Accuracy and Bandwidth



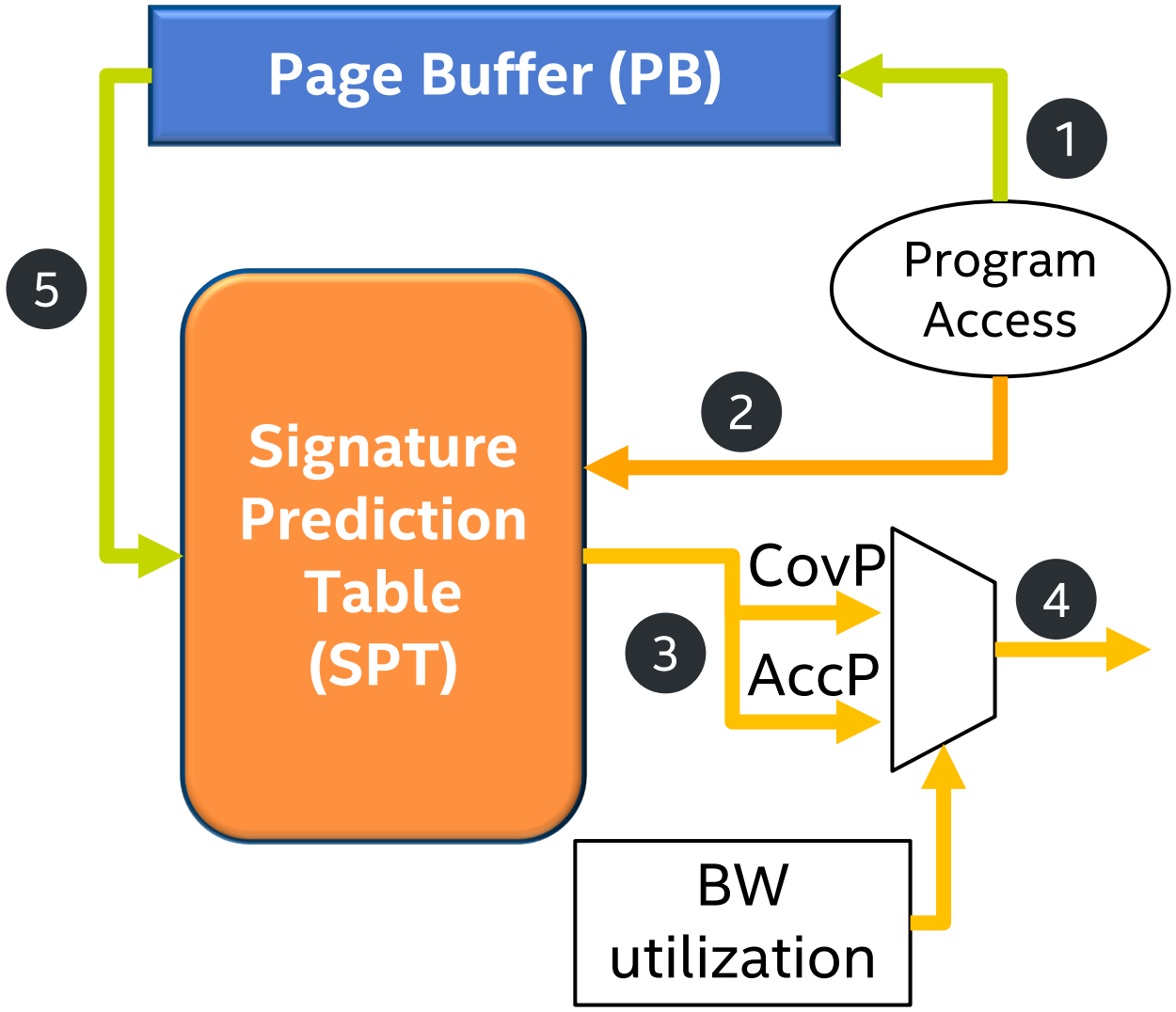
DSPatch: Quantify Coverage, Accuracy and Bandwidth



Quantifying Coverage and Accuracy

Pattern	Value	PopCount
From Program	1011 0100 0011 1100	8
From SPT	1010 0110 0000 0001	5

DSPatch: Quantify Coverage, Accuracy and Bandwidth

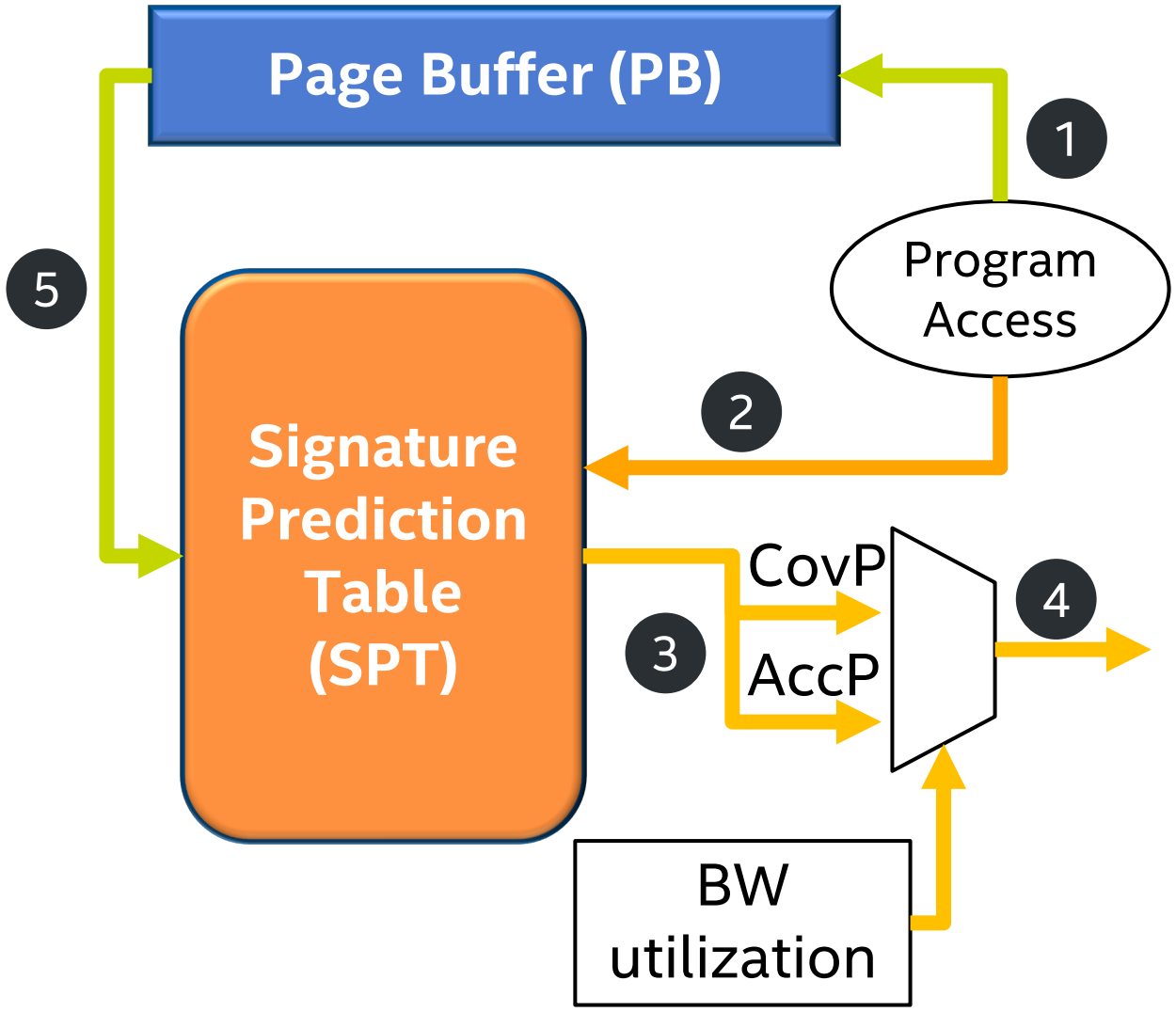


Quantifying Coverage and Accuracy

- Identify good bits in SPT pattern(s)

Pattern	Value	PopCount
From Program	1011 0100 0011 1100	8
From SPT	1010 0110 0000 0001	5
Bitwise-AND	1010 0100 0000 0000	3

DSPatch: Quantify Coverage, Accuracy and Bandwidth



Quantifying Coverage and Accuracy

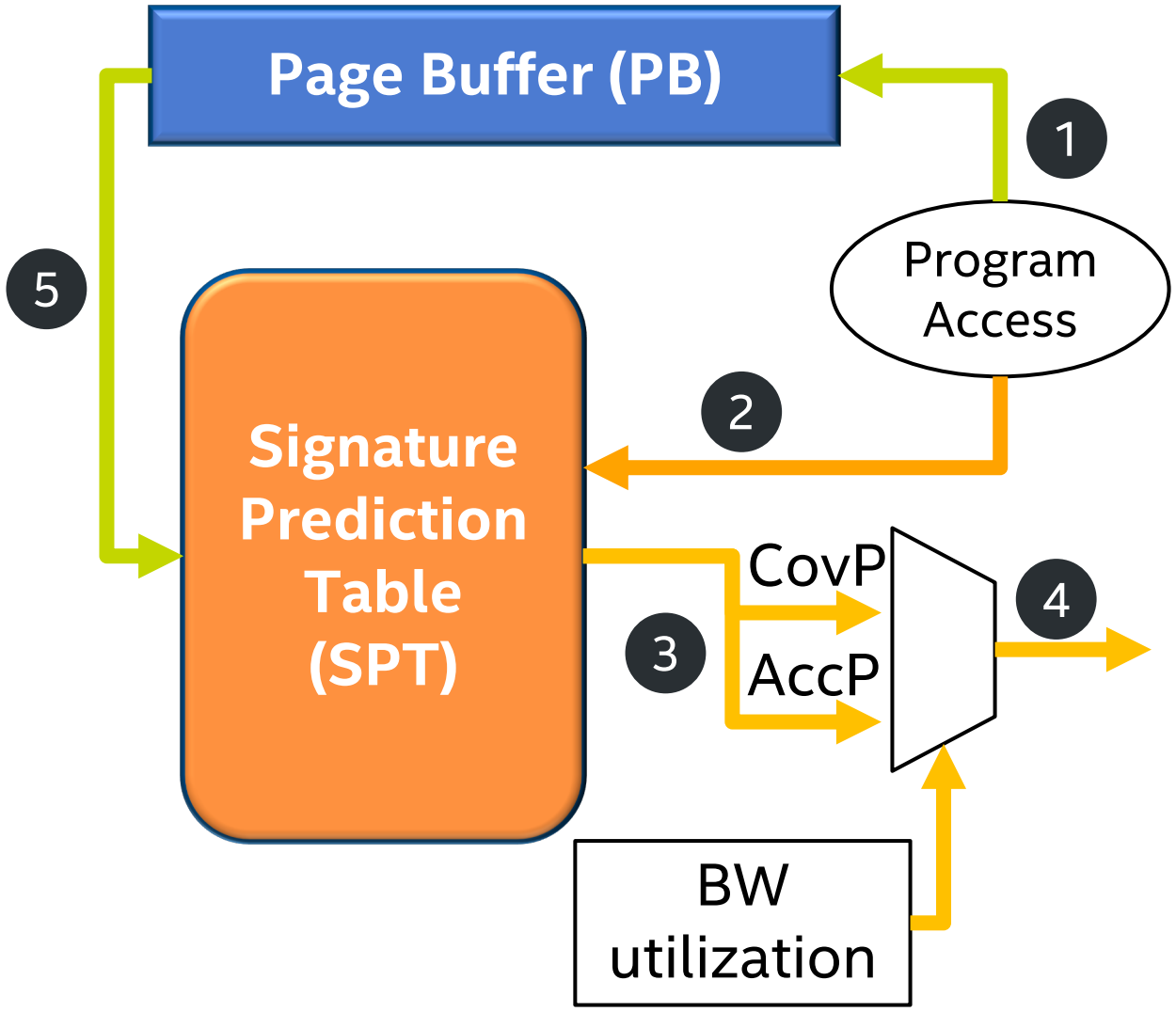
- Identify good bits in SPT pattern(s)
- PopCount compare to program pattern

Pattern	Value	PopCount
From Program	1011 0100 0011 1100	8
From SPT	1010 0110 0000 0001	5
Bitwise-AND	1010 0100 0000 0000	3

Coverage

Accuracy

DSPatch: Quantify Coverage, Accuracy and Bandwidth



Quantifying Coverage and Accuracy

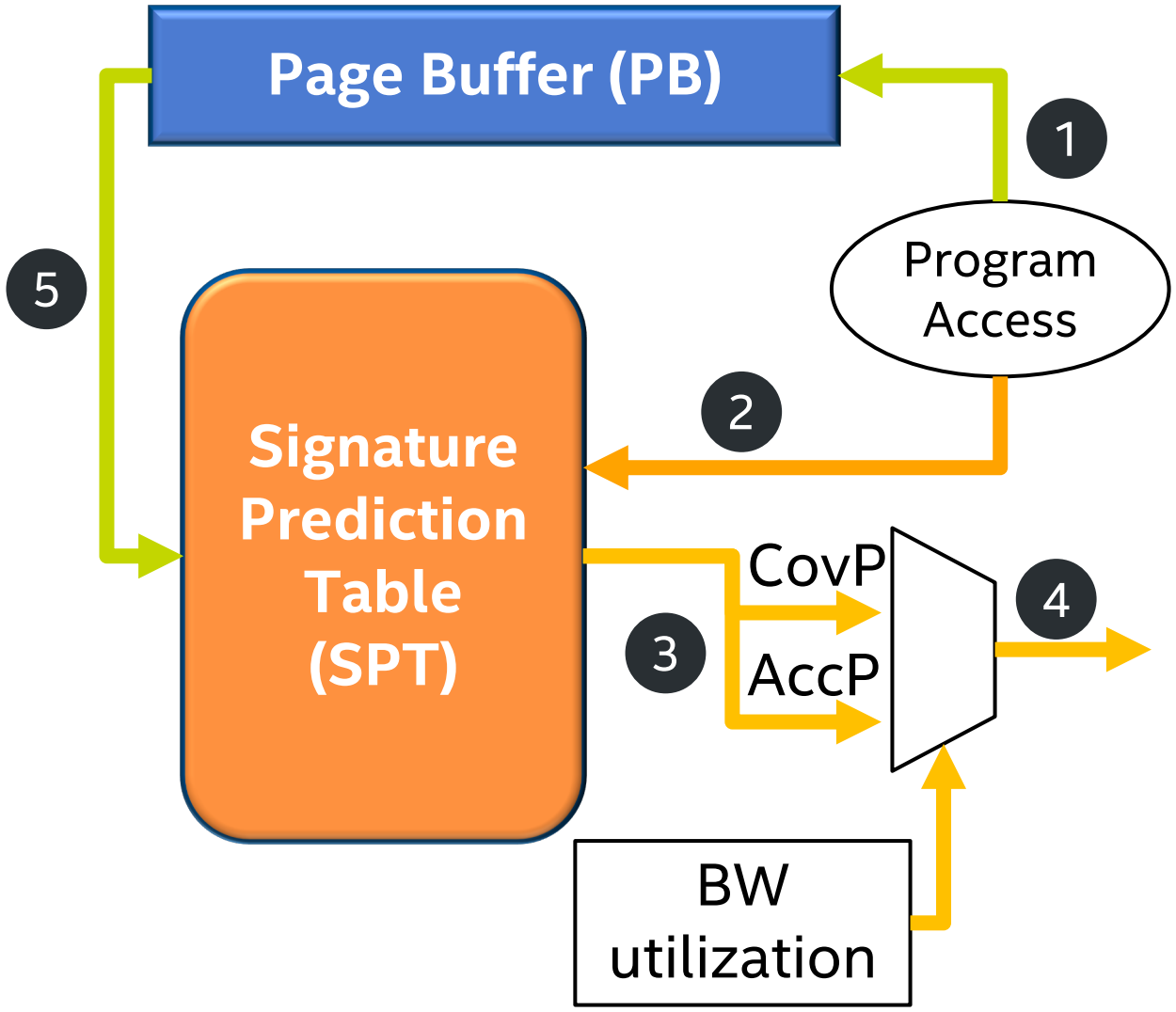
- Identify good bits in SPT pattern(s)
- PopCount compare to program pattern
- *Quantize to quartile buckets (25%,50,75%)*

Pattern	Value	PopCount
From Program	1011 0100 0011 1100	8
From SPT	1010 0110 0000 0001	5
Bitwise-AND	1010 0100 0000 0000	3

Coverage [25%-50%]

Accuracy [50%-75%]

DSPatch: Quantify Coverage, Accuracy and Bandwidth



Quantifying Coverage and Accuracy

- Identify good bits in SPT pattern(s)
- PopCount compare to program pattern
- *Quantize to quartile buckets (25%,50,75%)*

Pattern	Value	PopCount
From Program	1011 0100 0011 1100	8
From SPT	1010 0110 0000 0001	5
Bitwise-AND	1010 0100 0000 0000	3

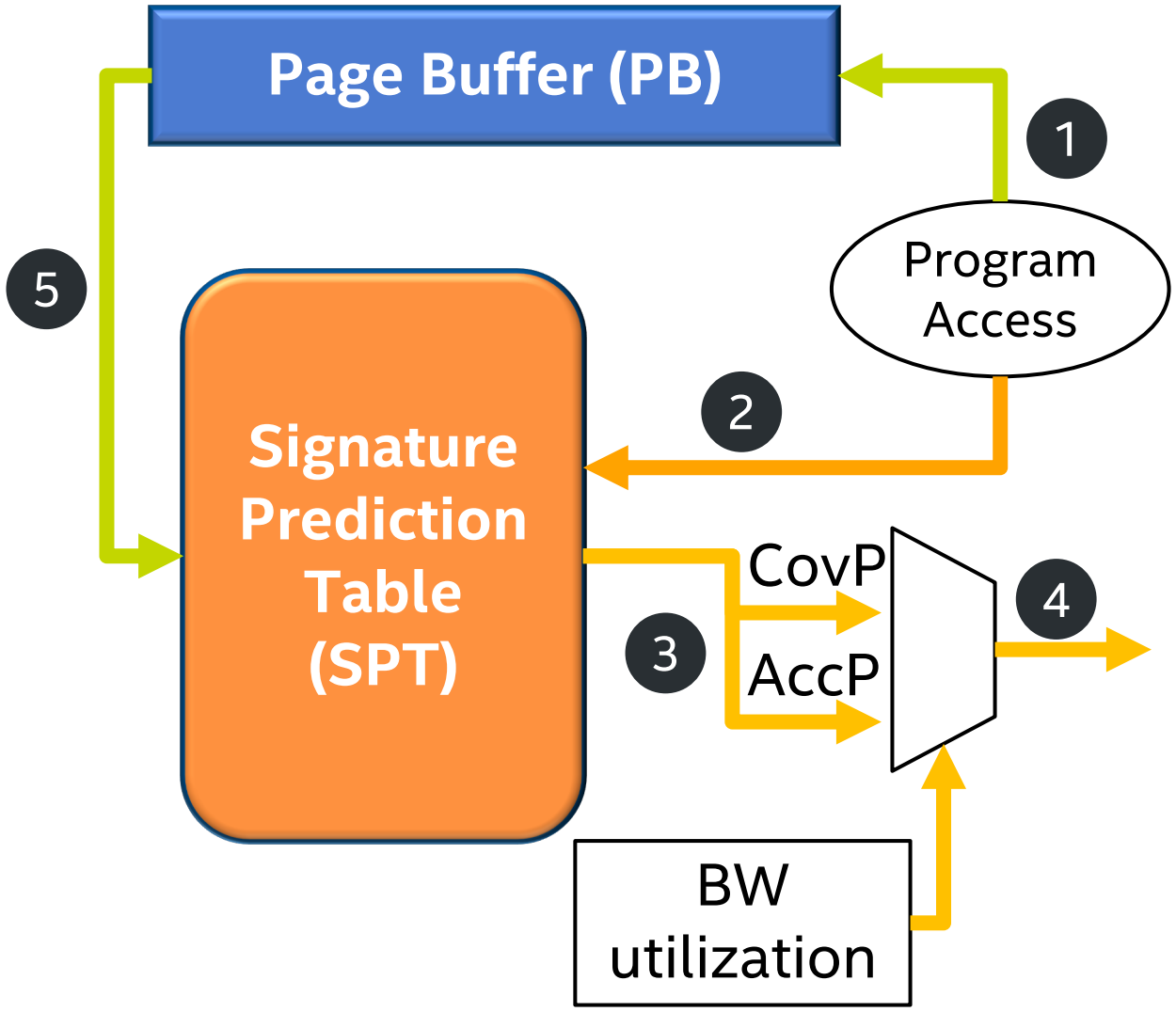
Coverage [25%-50%]

Accuracy [50%-75%]

Quantifying DRAM bandwidth utilization

- Count DRAM accesses in 4 x tRC window

DSPatch: Quantify Coverage, Accuracy and Bandwidth



Quantifying Coverage and Accuracy

- Identify good bits in SPT pattern(s)
- PopCount compare to program pattern
- *Quantize to quartile buckets (25%,50,75%)*

Pattern	Value	PopCount
From Program	1011 0100 0011 1100	8
From SPT	1010 0110 0000 0001	5
Bitwise-AND	1010 0100 0000 0000	3

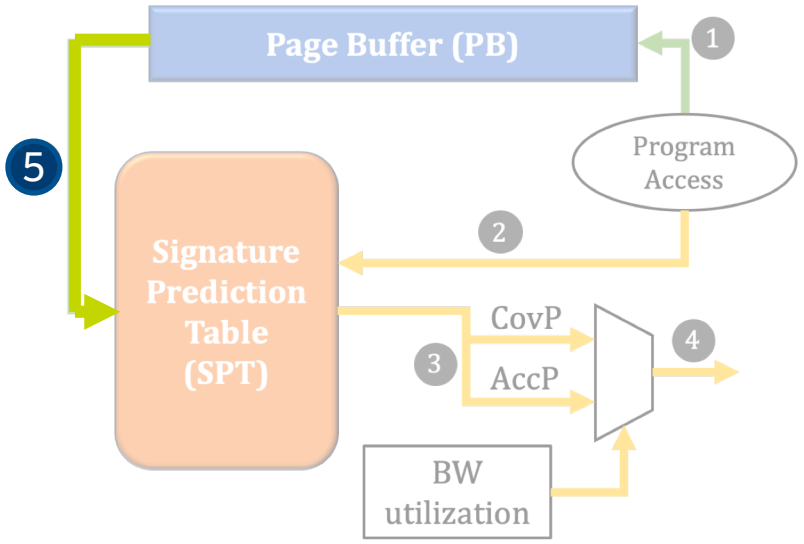
Coverage [25%-50%]

Accuracy [50%-75%]

Quantifying DRAM bandwidth utilization

- Count DRAM accesses in 4 x tRC window
- Compare to peak possible access count
- *Quantize to quartile buckets (25%,50,75%)*

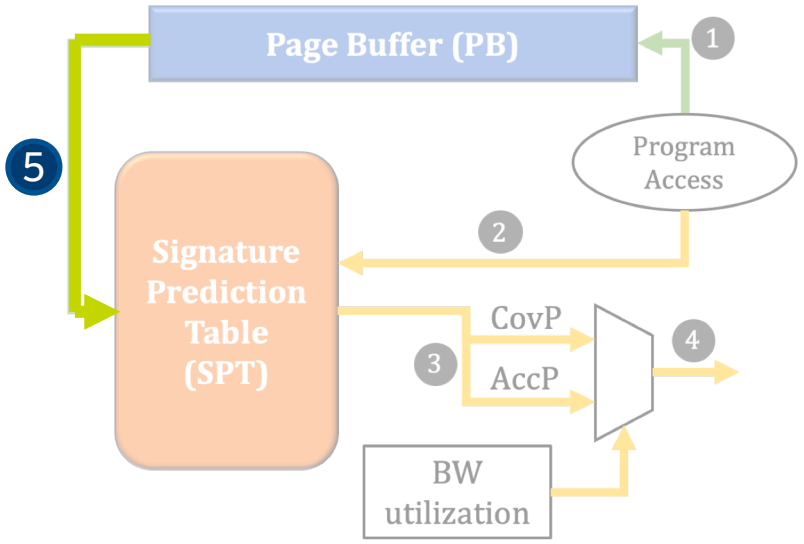
DSPatch: Pattern Update



PC Signature	CovP	AccP
0x7ffecca	1100 0110 0111 1100	1000 0010 0001 1000

Program
1001 0111 0001 0001

DSPatch: Pattern Update



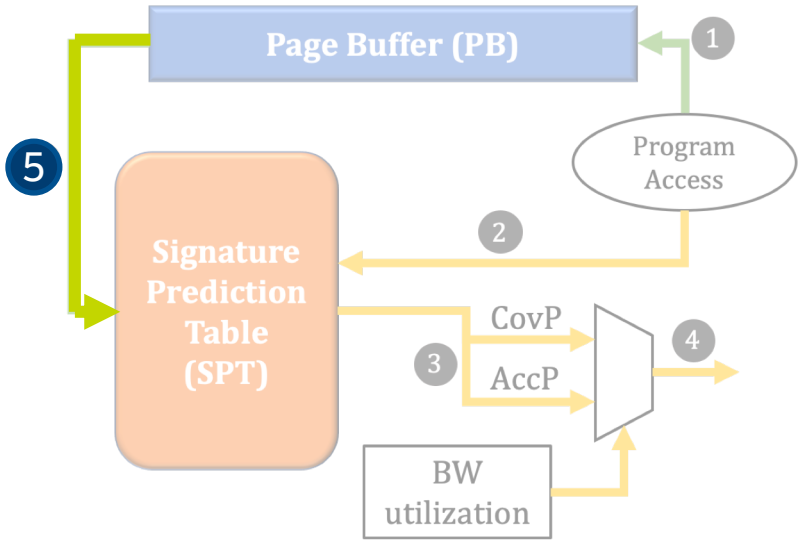
PC Signature	CovP	AccP
0x7ffecca	1100 0110 0111 1100	1000 0010 0001 1000

Program
1001 0111 0001 0001

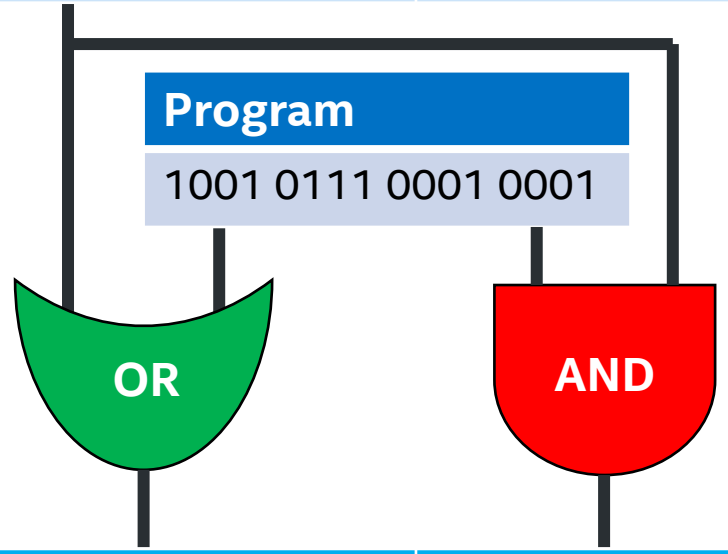
OR

PC Signature	CovP
0x7ffecca	1101 1111 0111 1101

DSPatch: Pattern Update

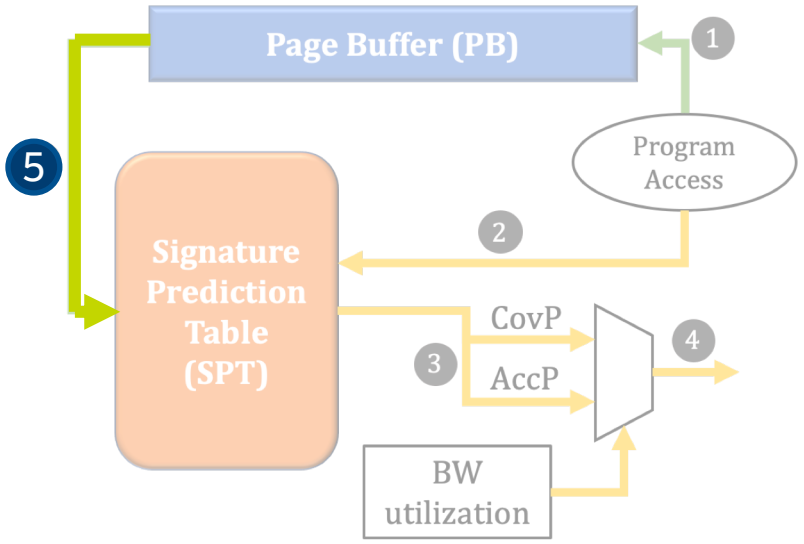


PC Signature	CovP	AccP
0x7ffecca	1100 0110 0111 1100	1000 0010 0001 1000



PC Signature	CovP	AccP
0x7ffecca	1101 1111 0111 1101	1000 0010 0001 0000

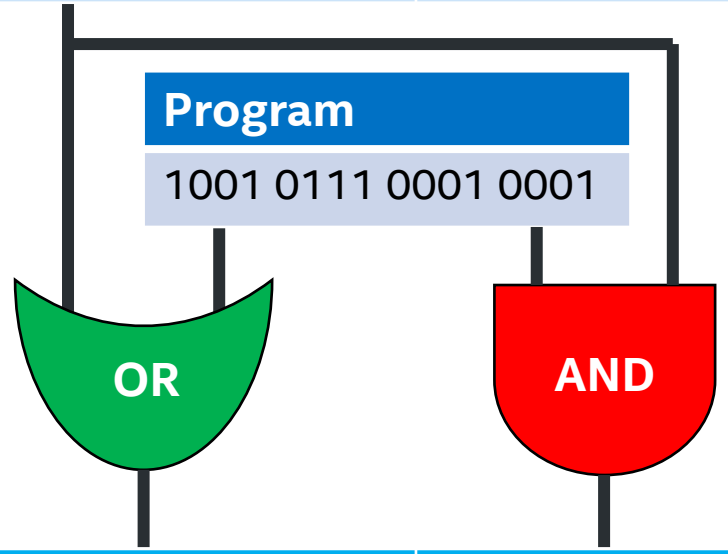
DSPatch: Pattern Update



OR$_{count}$

Limit number of modulation operations

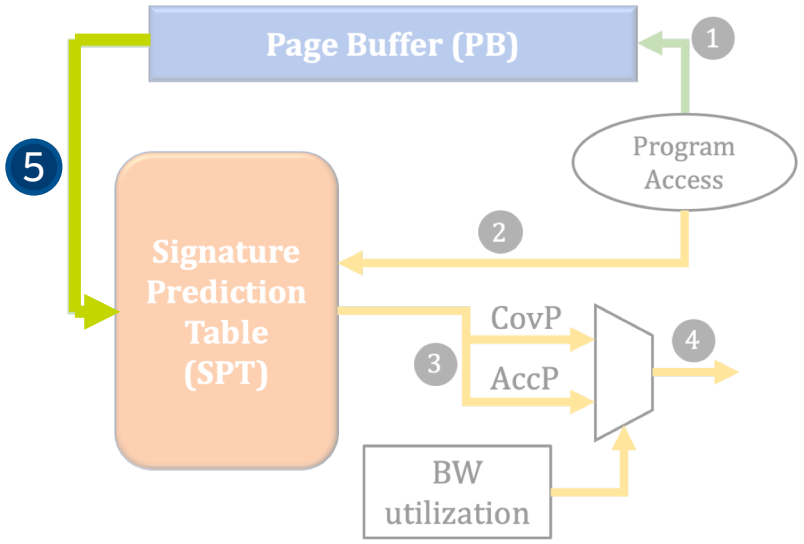
PC Signature	CovP	AccP
0x7ffecca	1100 0110 0111 1100	1000 0010 0001 1000



PC Signature	CovP	AccP
0x7ffecca	1101 1111 0111 1101	1000 0010 0001 0000

2b

DSPatch: Pattern Update



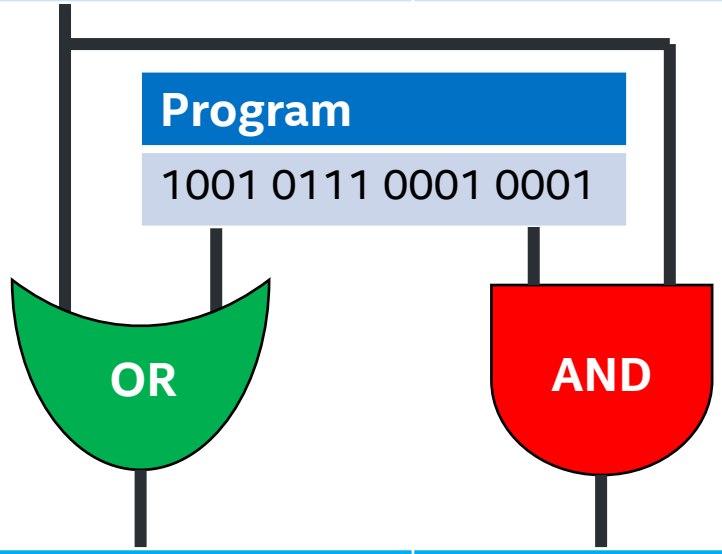
OR_{count}

Limit number of modulation operations

$Measure_{CovP}$

Incr if CovP coverage < 50% || CovP accuracy < 50%

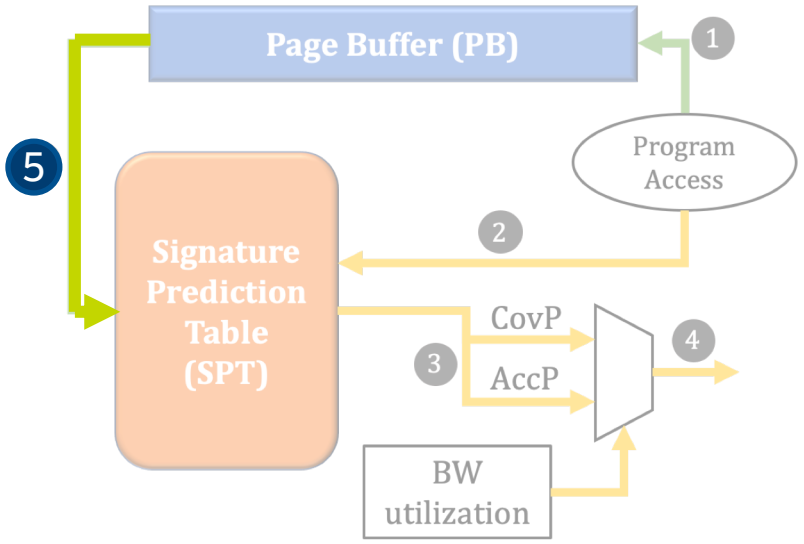
PC Signature	CovP	AccP
0x7ffecca	1100 0110 0111 1100	1000 0010 0001 1000



PC Signature	CovP	AccP
0x7ffecca	1101 1111 0111 1101	1000 0010 0001 0000

2b 2b

DSPatch: Pattern Update



OR_{count}

Limit number of modulation operations

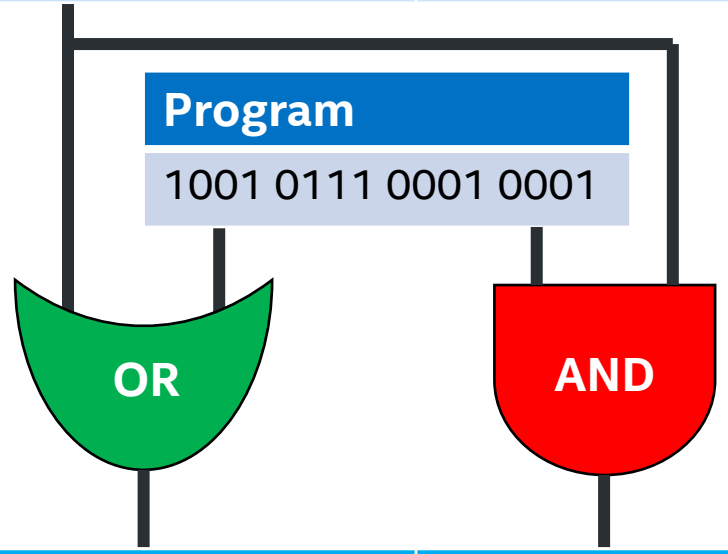
$Measure_{CovP}$

Incr if CovP coverage < 50% || CovP accuracy < 50%

$Measure_{AccP}$

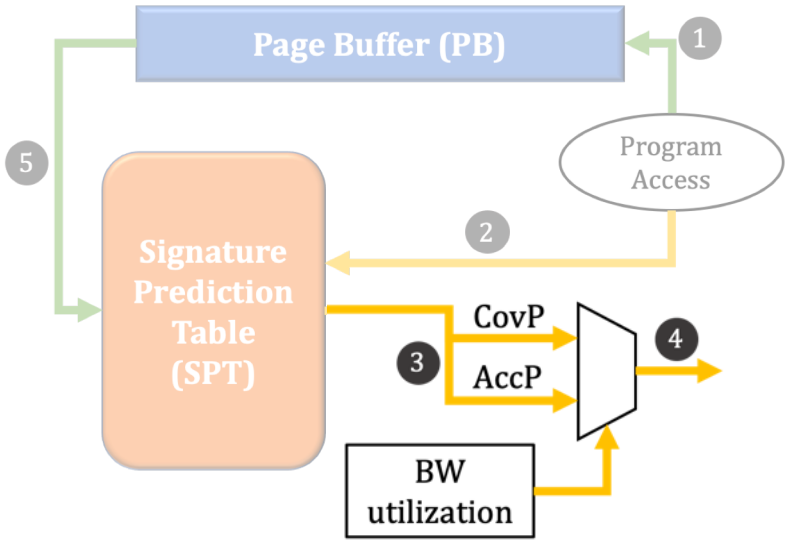
Incr if AccP accuracy < 50%

PC Signature	CovP	AccP
0x7ffecca	1100 0110 0111 1100	1000 0010 0001 1000



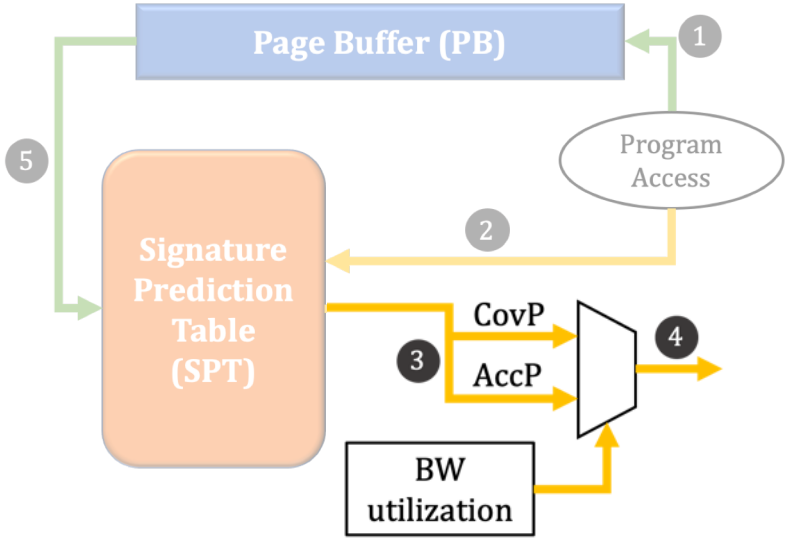
PC Signature	CovP	AccP
0x7ffecca	1101 1111 0111 1101	1000 0010 0001 0000
	2b 2b	2b

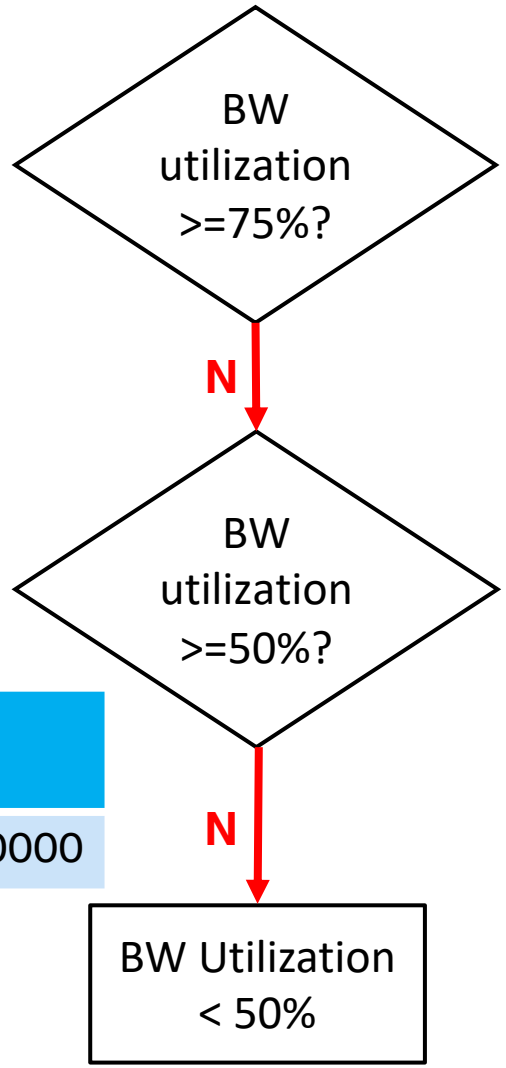
DSPatch: Pattern Predict



PC Signature	CovP	AccP
0x7ffecca	1101111101111101	1000001000010000
	2b 2b	2b

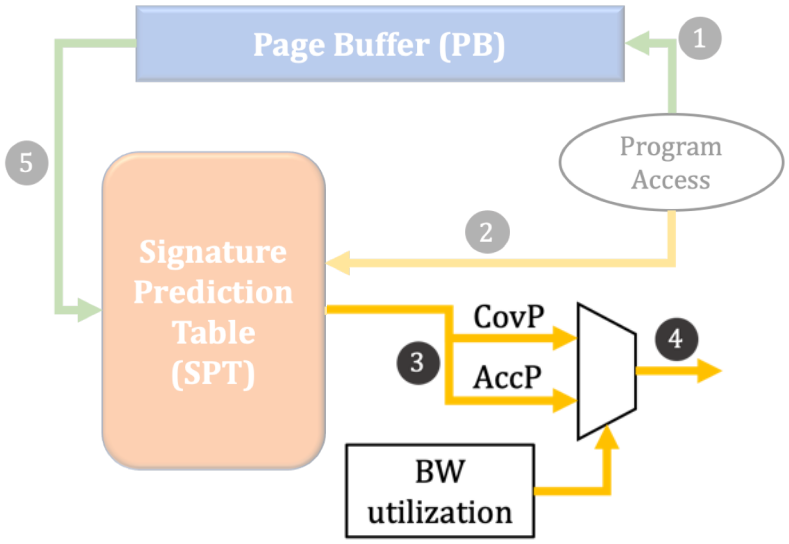
DSPatch: Pattern Predict

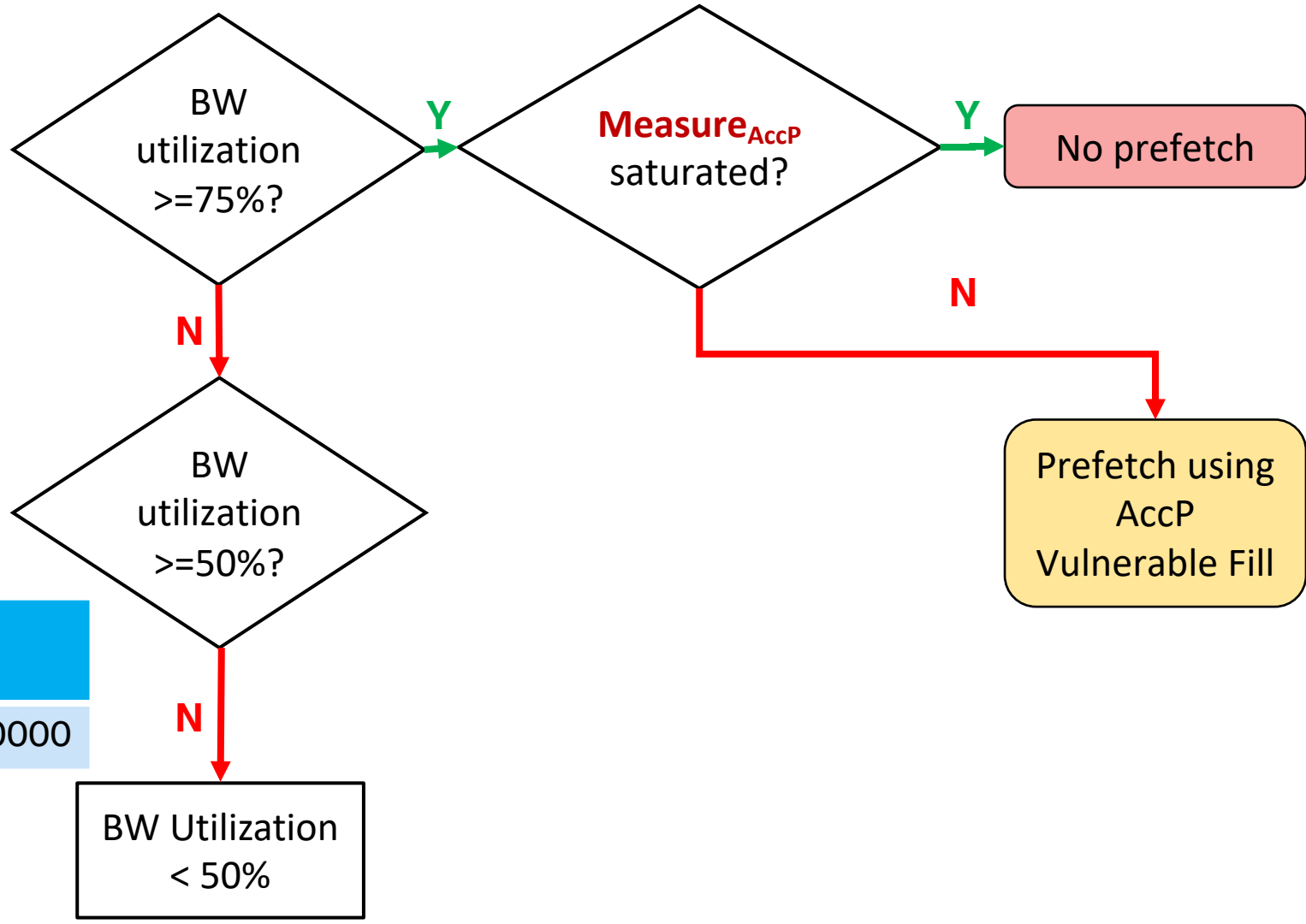




PC Signature	CovP	AccP
0x7ffecca	1101111101111101	1000001000010000
	2b 2b	2b

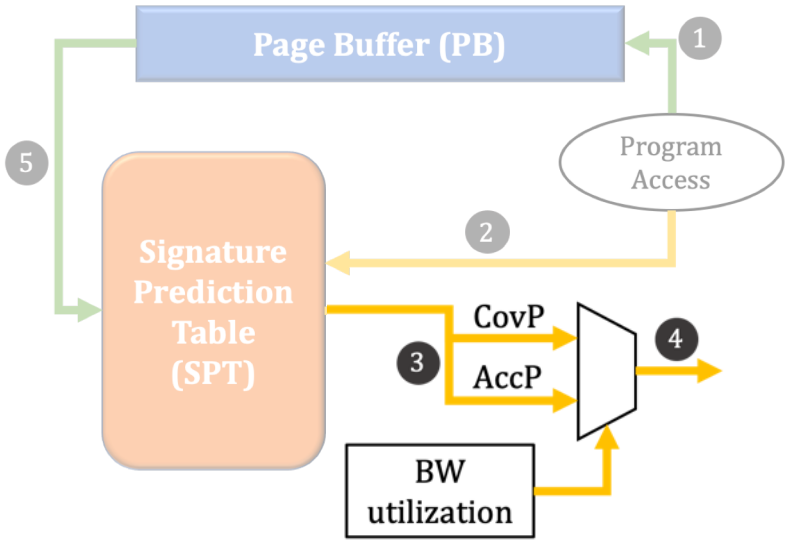
DSPatch: Pattern Predict



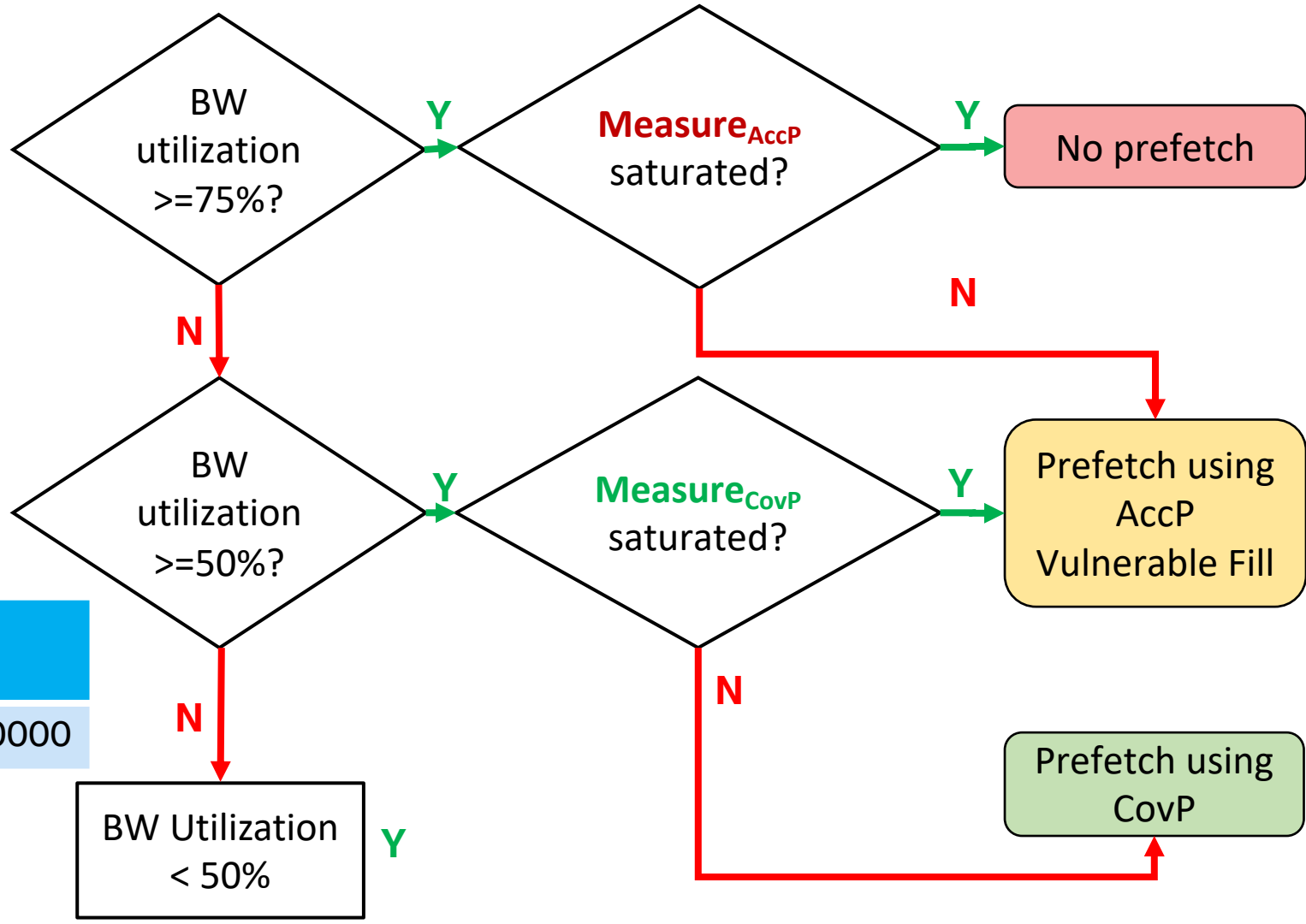


PC Signature	CovP	AccP
0x7ffecca	1101111101111101	1000001000010000
	2b 2b	2b

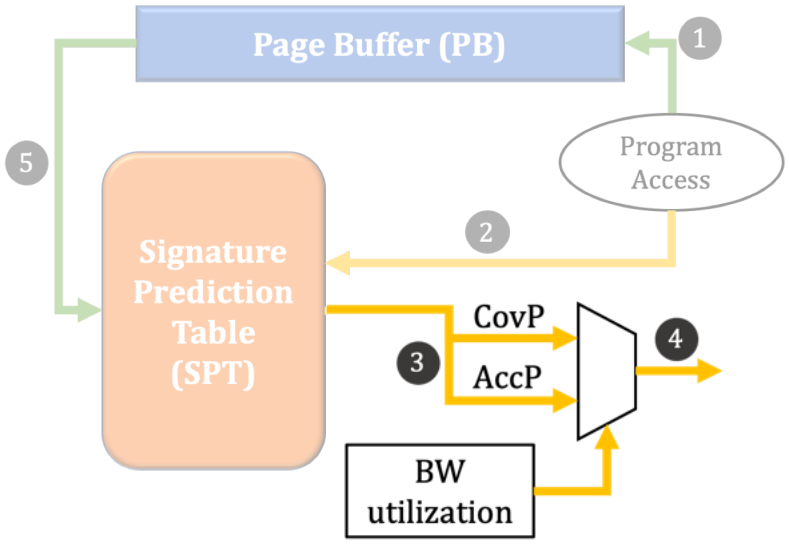
DSPatch: Pattern Predict



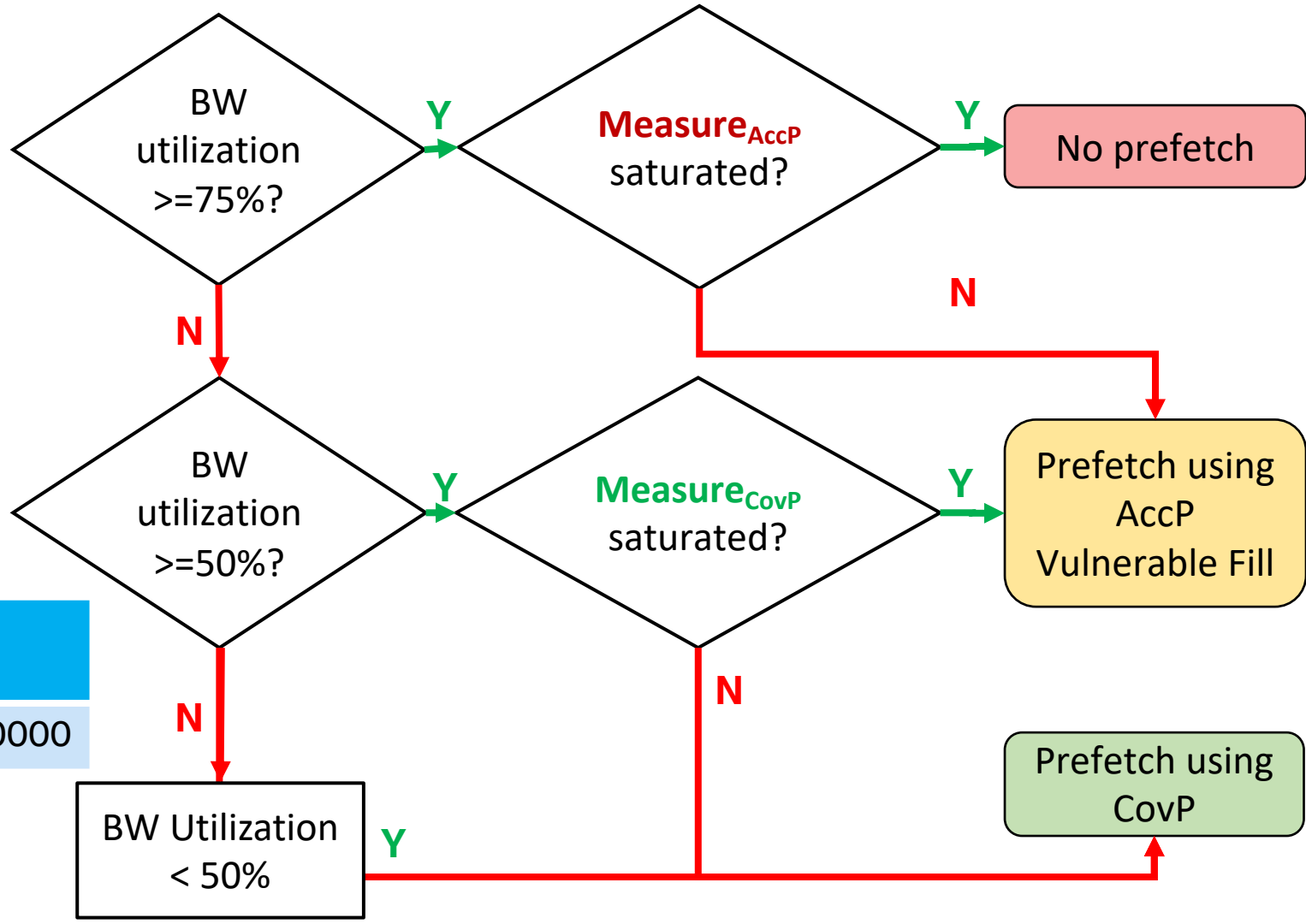
PC Signature	CovP	AccP
0x7ffecca	1101111101111101	1000001000010000
	2b 2b	2b



DSPatch: Pattern Predict



PC Signature	CovP	AccP
0x7ffecca	1101111101111101	1000001000010000
	2b 2b	2b



Outline

1. Motivation and Observations
2. Dual Spatial Pattern Prefetcher (DSPatch)
3. Evaluation
4. Conclusion

Evaluation Methodology

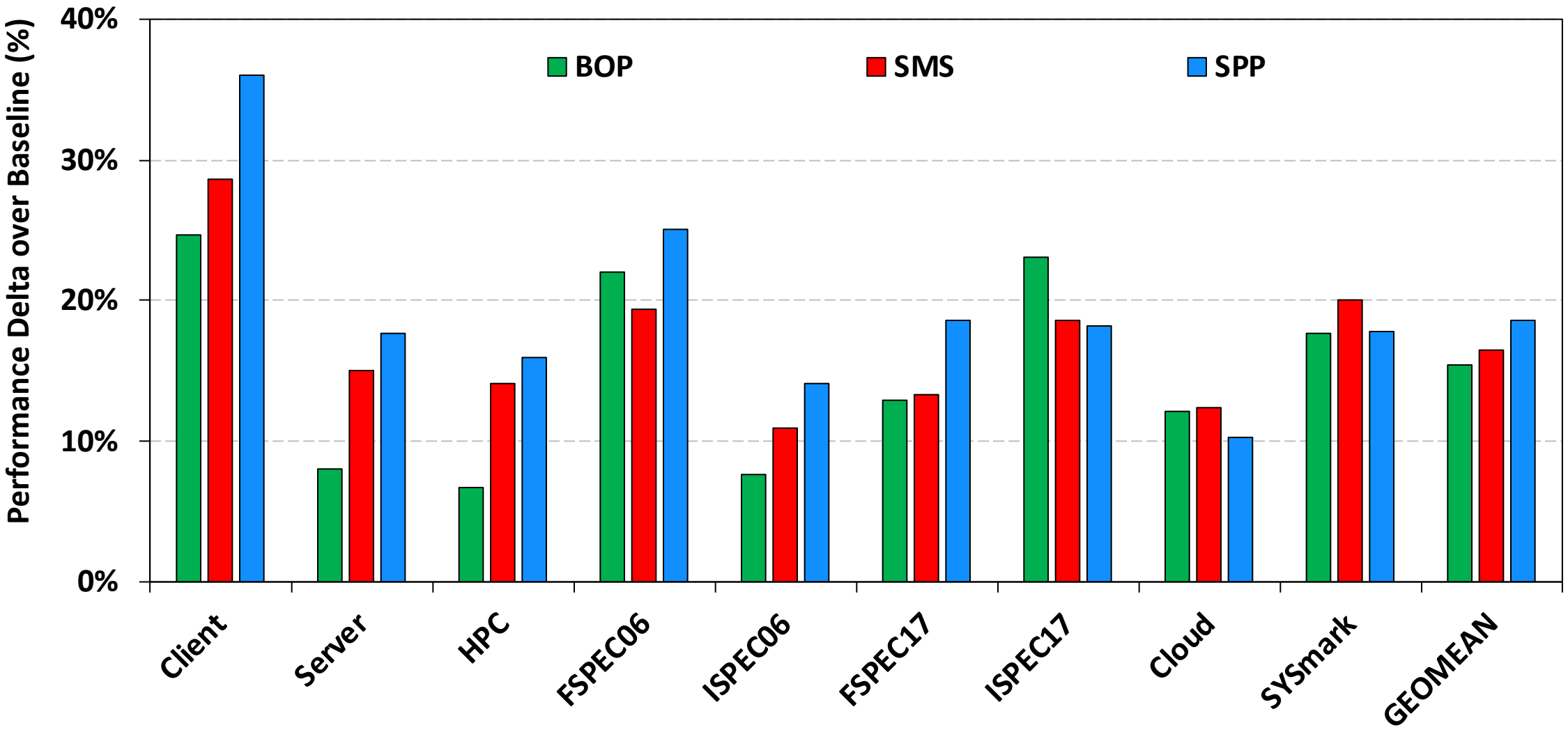
Configuration

- 4 x86 cores @ 4 GHz, 4 wide OoO, 224 deep ROB entries
- 32KB, 8 way Data and Code L1 caches
  - PC-stride prefetcher into Data L1
- 256 KB L2 Cache
  - BOP, SPP, SMS, DSPatch evaluated @ L2
- 2 MB LLC/Core
- Memory Configuration
  - Single Core evaluations: 1Ch-DDR4-2133MHz
  - Multi Core evaluations: 2Ch-DDR4-2133MHz

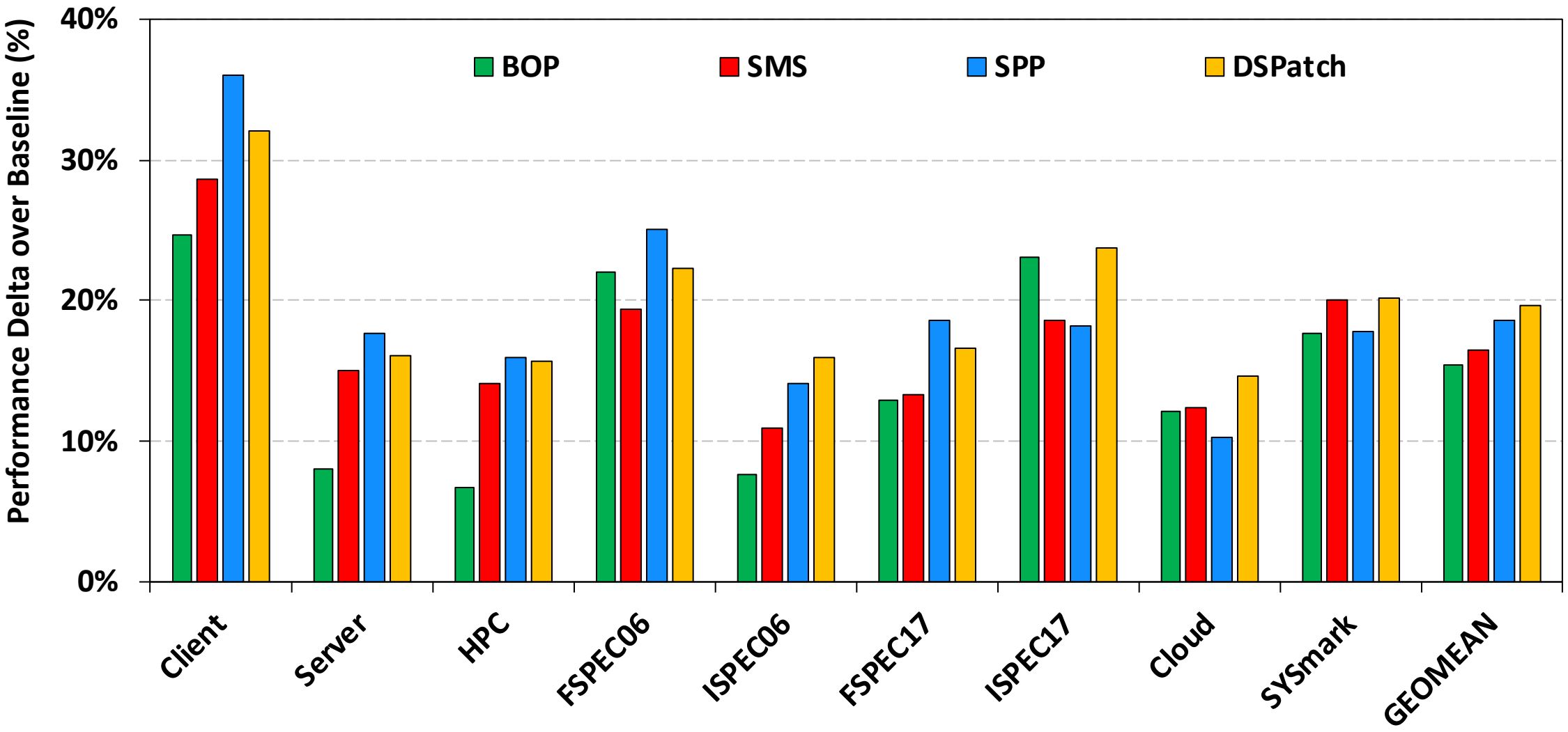
Workloads

- Single-Core: 75 workloads spanning Cloud, Client, Server, HPC, Sysmark, SPEC06 and SPEC17
- Multi-Core: 42 high-MPKI homogeneous mixes, 75 heterogeneous mixes

DSPatch: Single-Core Performance

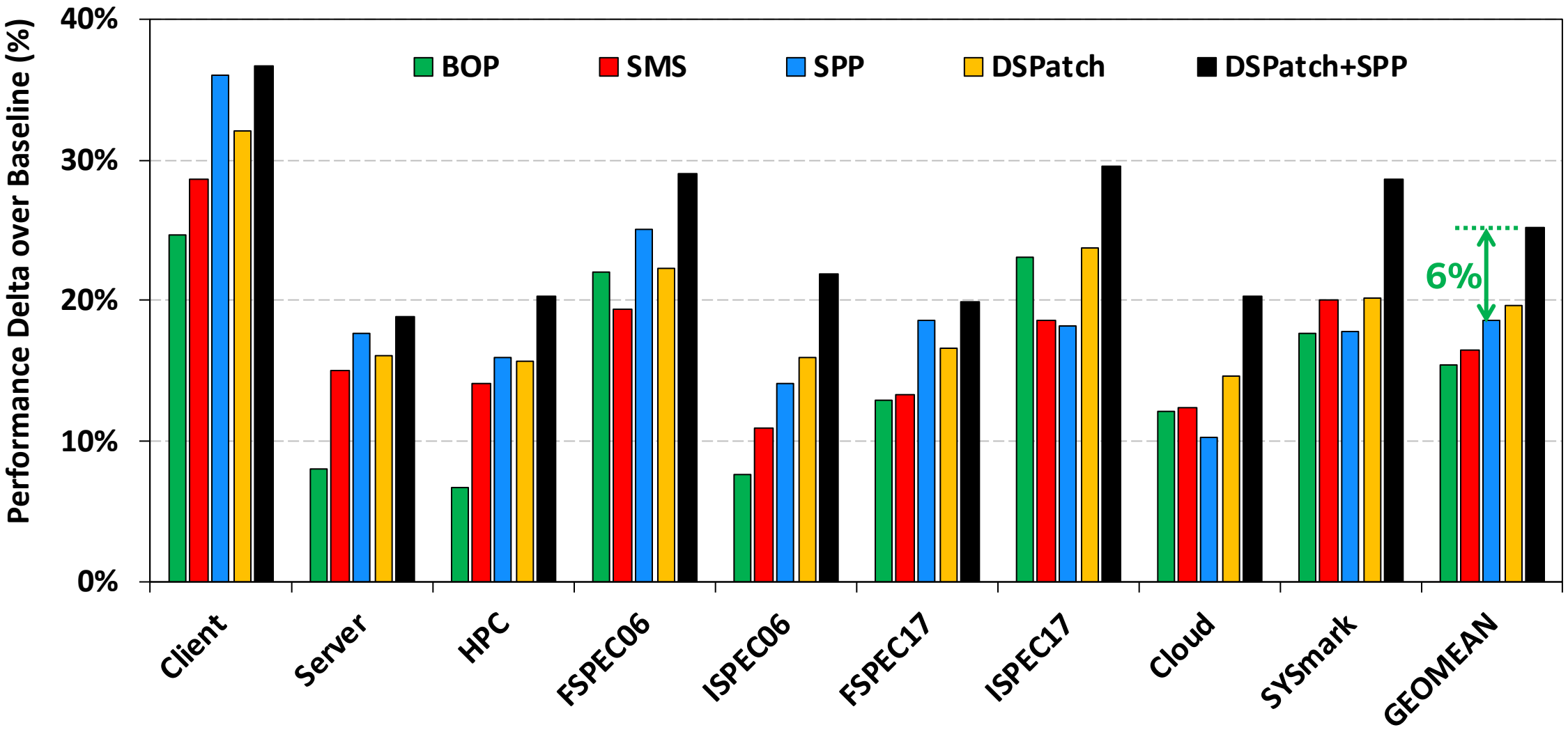


DSPatch: Single-Core Performance



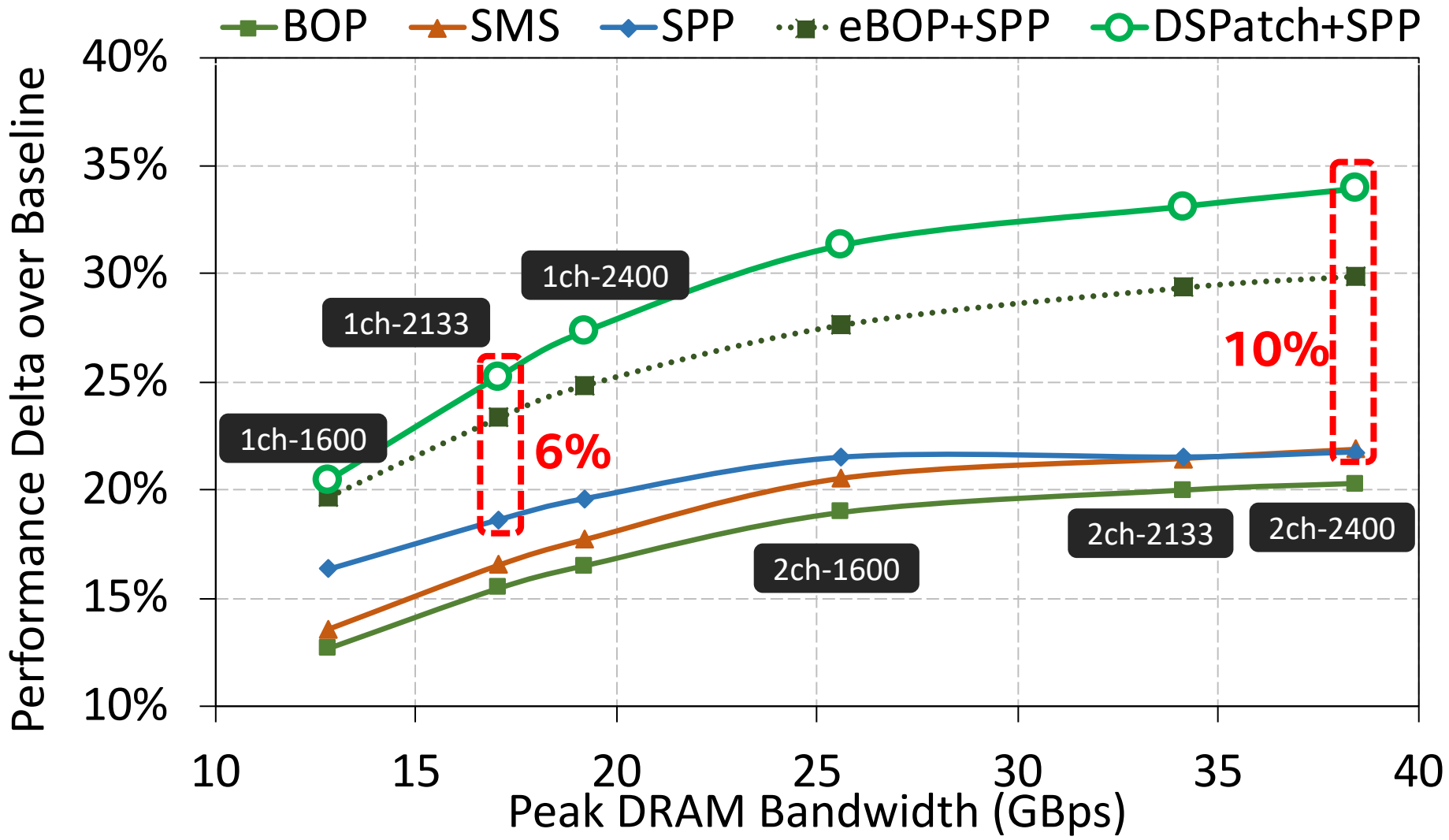
Standalone DSPatch outperforms SPP and SMS by 1% and 3% respectively

DSPatch: Single-Core Performance



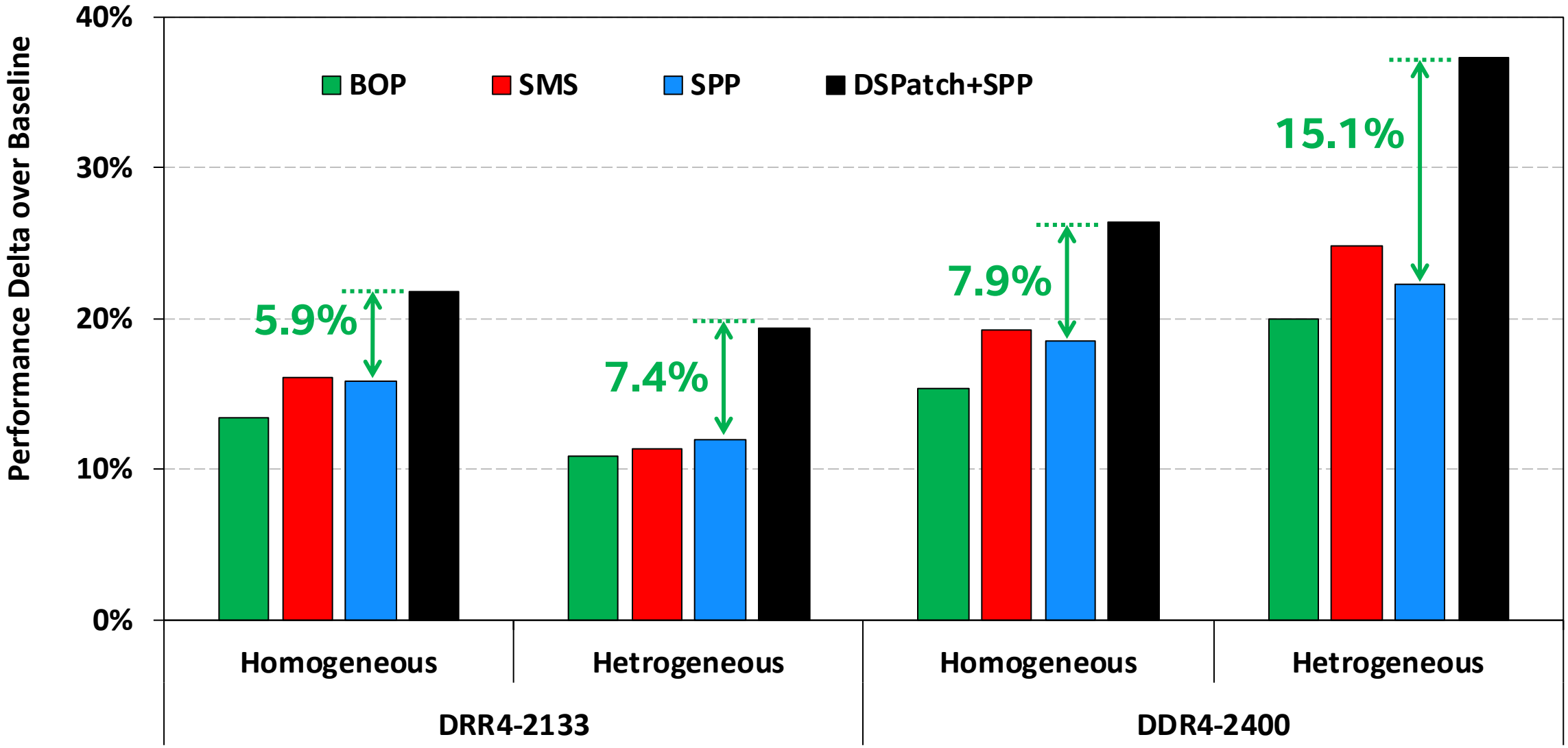
Standalone DSPatch outperforms SPP and SMS by 1% and 3% respectively
DSPatch+SPP outperforms SPP by 6% on average

DSPatch: Performance Scaling With Bandwidth



DSPatch performance scales well with increase in memory bandwidth

DSPatch: Multi-Core Performance



DSPatch performs well in bandwidth-constrained multi-core scenarios

DSPatch: Hardware Storage

Structure	Size Per Entry (in bytes)	No. of Entries	Size (in KB)
PB	19.75 B	64	1.23 KB
SPT	9.5 B	256	2.375 KB
Total			**3.6 KB**

DSPatch is an order of magnitude smaller than SMS and takes only 2/3rd storage of SPP

More In Paper

Design optimizations

- Bit-pattern compression (Reduce storage)
- Multiple triggers per 4KB page (Increase coverage)

Results

- Breakdown of coverage and accuracy of DSPatch
- Performance comparison of ISO-sized adjunct prefetchers
- Contribution of *Coverage vs. Accuracy* biased patterns

Outline

1. Motivation and Observations
2. Dual Spatial Pattern Prefetcher (DSPatch)
3. Evaluation
4. Conclusion

Summary

Summary

Motivation:

- DRAM bandwidth increases with every generation
- Prefetchers need to adapt to effectively utilize this resource

Challenge:

- *Significantly boost* ***Coverage*** **AND**
- *Simultaneously optimize* for ***Accuracy***

Summary

Motivation:

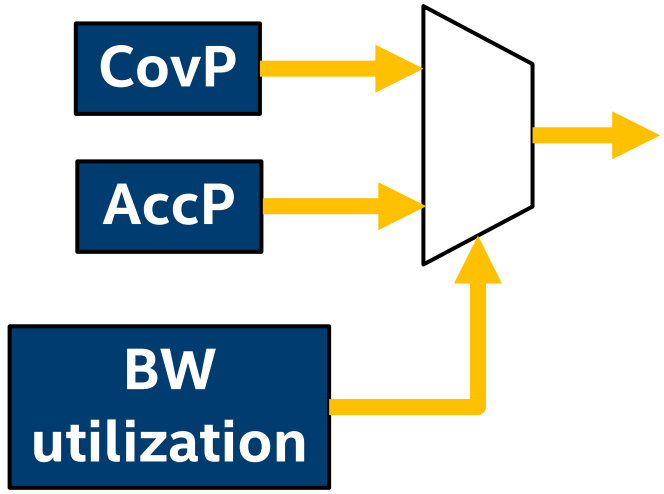
- DRAM bandwidth increases with every generation
- Prefetchers need to adapt to effectively utilize this resource

Challenge:

- *Significantly boost* ***Coverage*** **AND**
- *Simultaneously optimize* for ***Accuracy***

Dual Spatial Pattern Prefetcher (DSPatch)

- Simultaneously learn ***two*** spatial bit-pattern representations of program accesses per page
  - Coverage-biased (CovP)
  - Accuracy-biased (AccP)
- Predict ***one*** of the patterns based on current DRAM bandwidth headroom



Summary

Motivation:

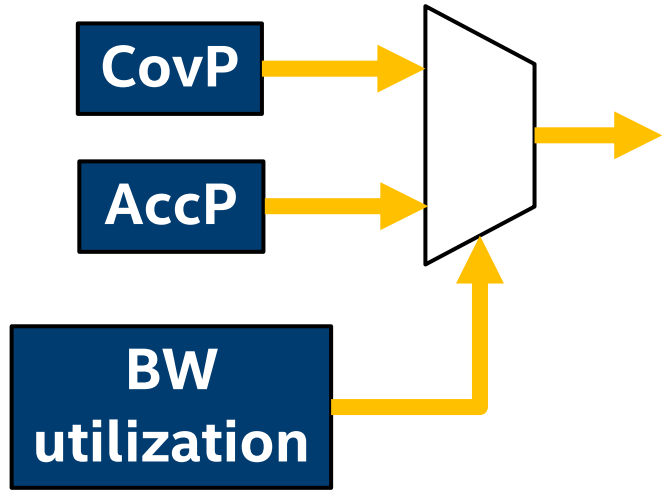
- DRAM bandwidth increases with every generation
- Prefetchers need to adapt to effectively utilize this resource

Challenge:

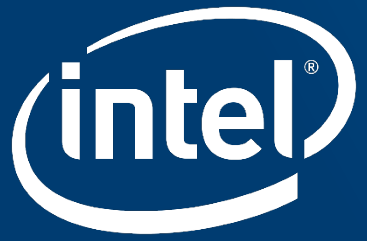
- *Significantly boost* ***Coverage*** **AND**
- *Simultaneously optimize* for ***Accuracy***

Dual Spatial Pattern Prefetcher (DSPatch)

- Simultaneously learn ***two*** spatial bit-pattern representations of program accesses per page
  - Coverage-biased (CovP)
  - Accuracy-biased (AccP)
- Predict ***one*** of the patterns based on current DRAM bandwidth headroom



- **6% average speedup** over baseline with PC-stride @ L1 and SPP @ L2
- **10% average speedup** when DRAM bandwidth is doubled
- **Only 3.6 KB** of hardware storage



SAFARI

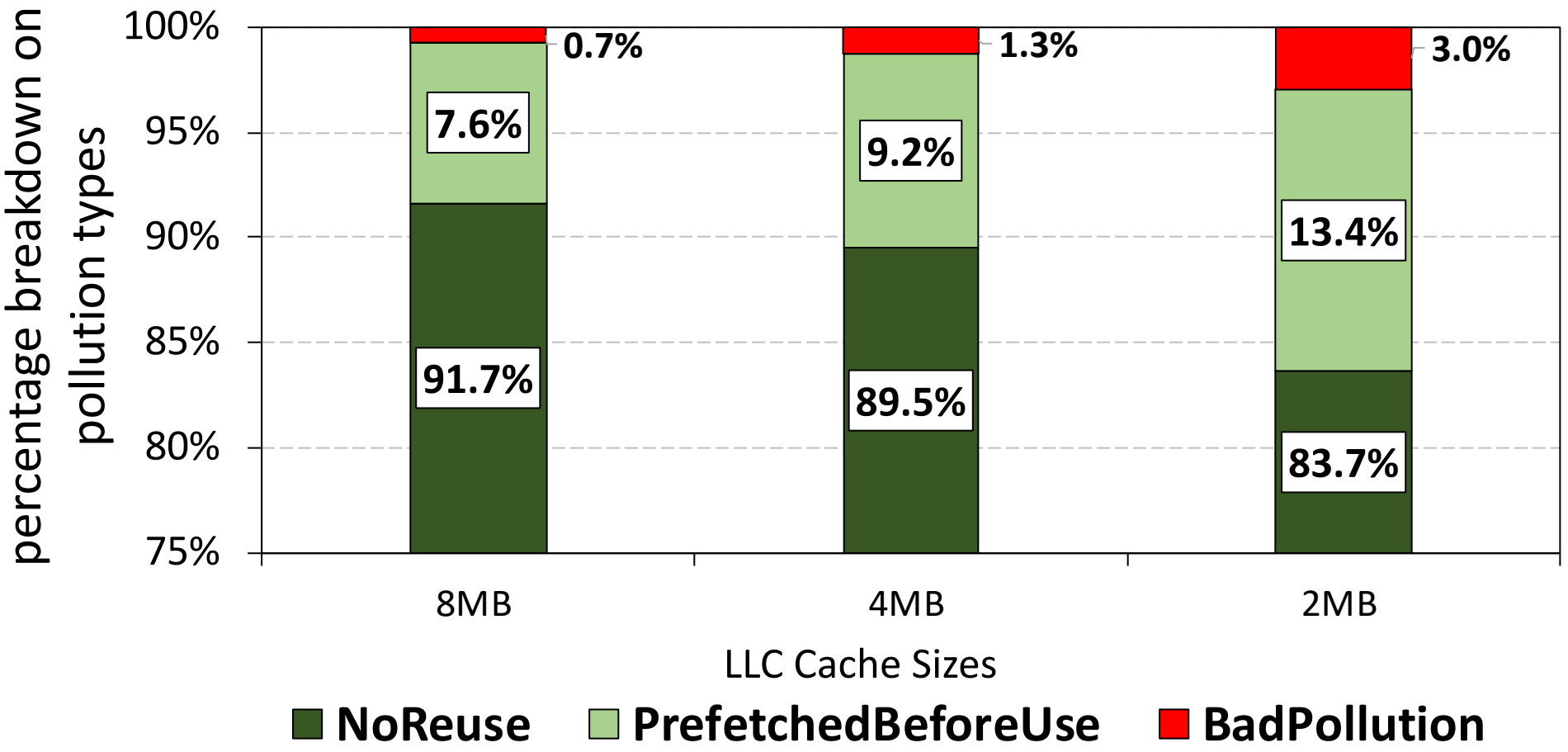
DSPATCH: DUAL SPATIAL PATTERN PREFETCHER

Rahul Bera[1], Anant V. Nori[1], Onur Mutlu[2], Sreenivas Subramoney[1]

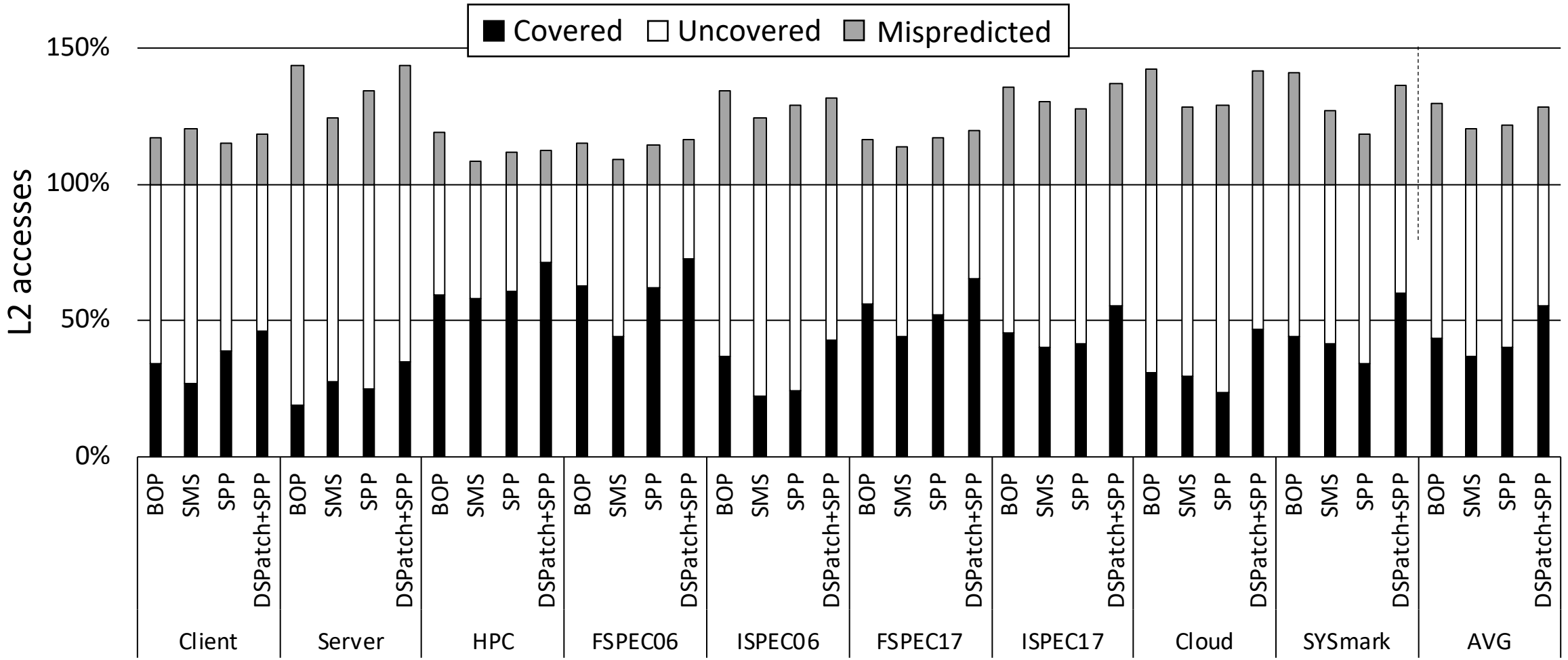
[1]Processor Architecture Research Lab, Intel Labs
[2]ETH Zürich

BACKUP

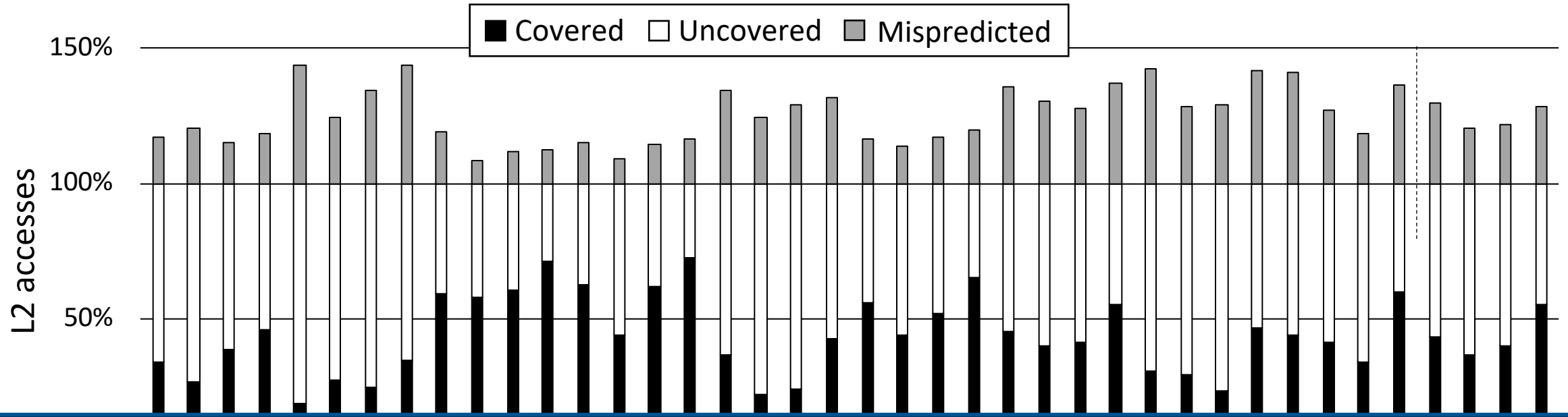
Observation: Pollution



Coverage and Accuracy



Coverage and Accuracy



For DSPatch, every 2% increase in coverage comes at a cost of only 1% increase in misprediction