Institut de Recherche en Informatique et Systèmes Aléatoires

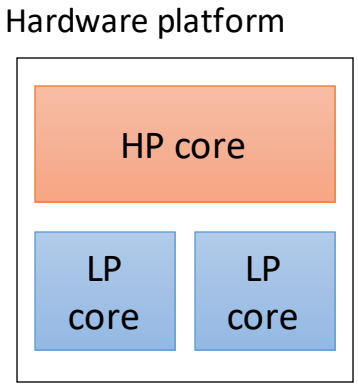
Mitigating Spectre Attacks on a DBT-Based Processor

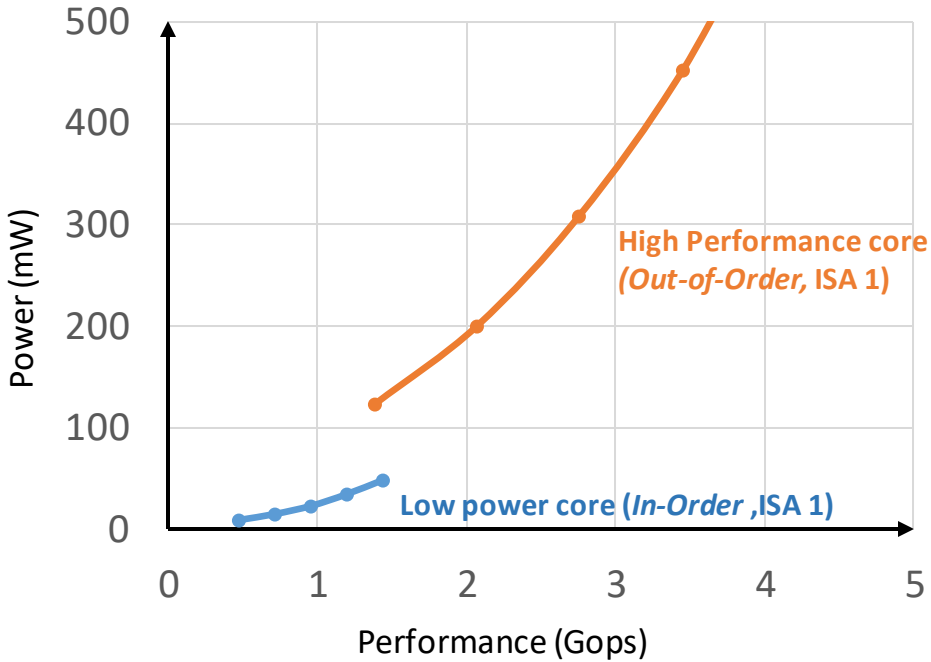
Simon Rokicki

Univ Rennes, INRIA, CNRS, IRISA

Energy efficient computing architectures

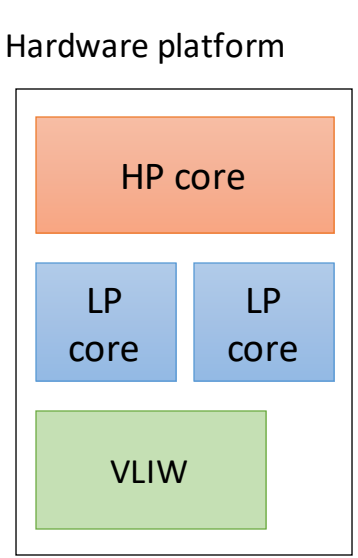
- **DVFS + single ISA heterogeneity (Arm big.LITTLE)**

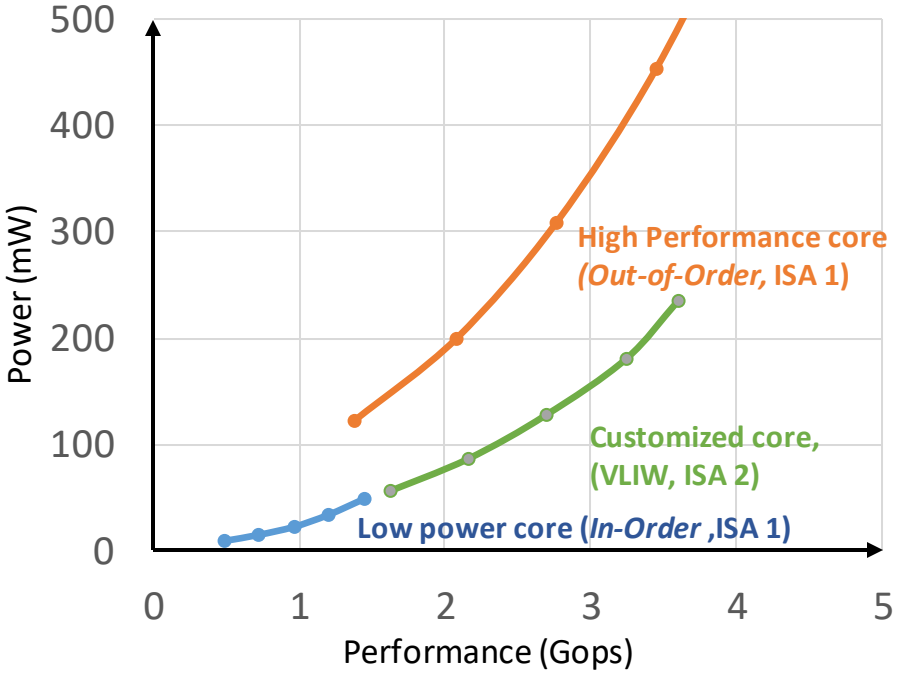




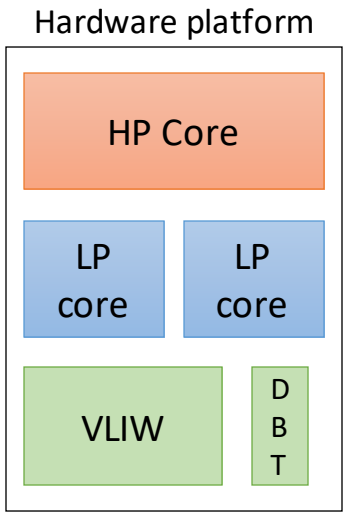
Energy efficient computing architectures

- **DVFS + multi-ISA heterogeneity**





DBT based processors



Dynamic Binary Translation

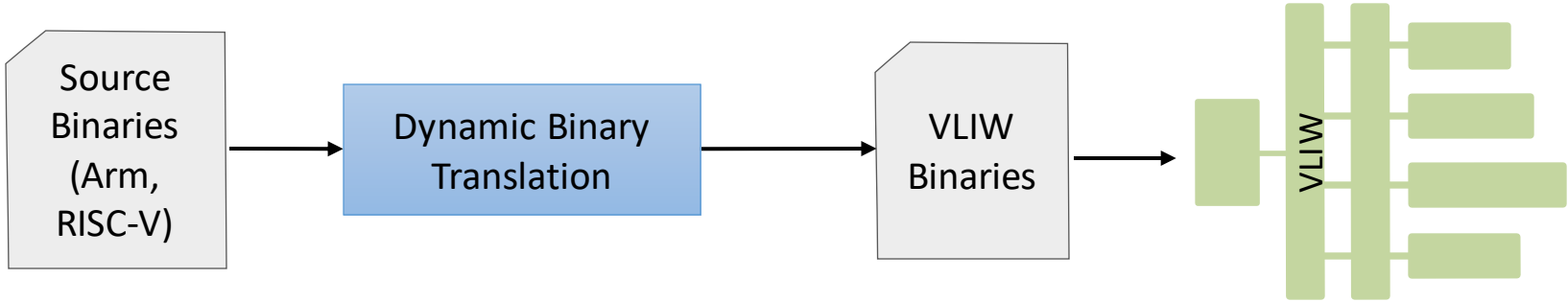
- **Dynamic compilation from binary**
- **Mostly for emulation/sim (QEMU)**

DBT based processors

- **Transmeta Crusoe & Efficeon [1]**
- **Nvidia Denver & Carmel [2]**
- **Hybrid-DBT [3]**

[1] Dehnert et al. *The Transmeta Code Morphing™ Software*. CGO 2003
[2] Boggs et al., *Denver: Nvidia's First 64-bit ARM Processor*. Micro, 2015
[3] Rokicki et al. *Hybrid-DBT: Hardware/Software Dynamic Binary Translation Targeting VLIW*. TCAD, 2018

Speculation on DBT based processors



- **Source binaries are translated into VLIW binaries**
- **Binaries with speculative execution**
- **Is it vulnerable to Spectre attacks ?**
- **Are there countermeasures ?**

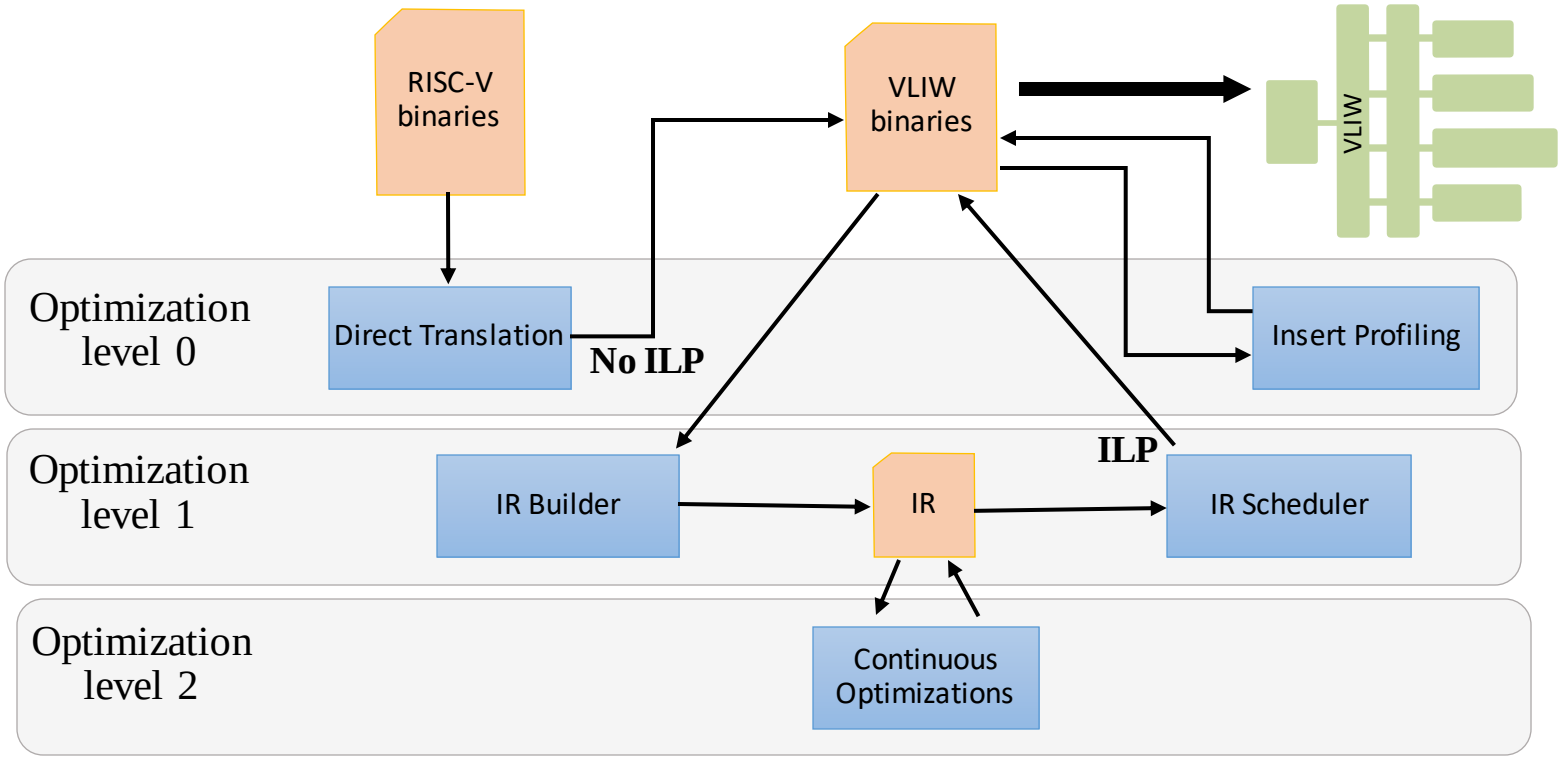
Outline

- **Overview of Hybrid-DBT Optimization Flow**
- **Spectre v1: Bound check bypass**
- **Spectre v4: Speculative Store Bypass**
- **Proposed countermeasure**

Outline

- **Overview of Hybrid-DBT Optimization Flow**
- Spectre v1: Bound check bypass
- Spectre v4: Speculative Store Bypass
- Proposed countermeasure

Hybrid-DBT Optimization Flow



ILP: Instruction Level Parallelism

Outline

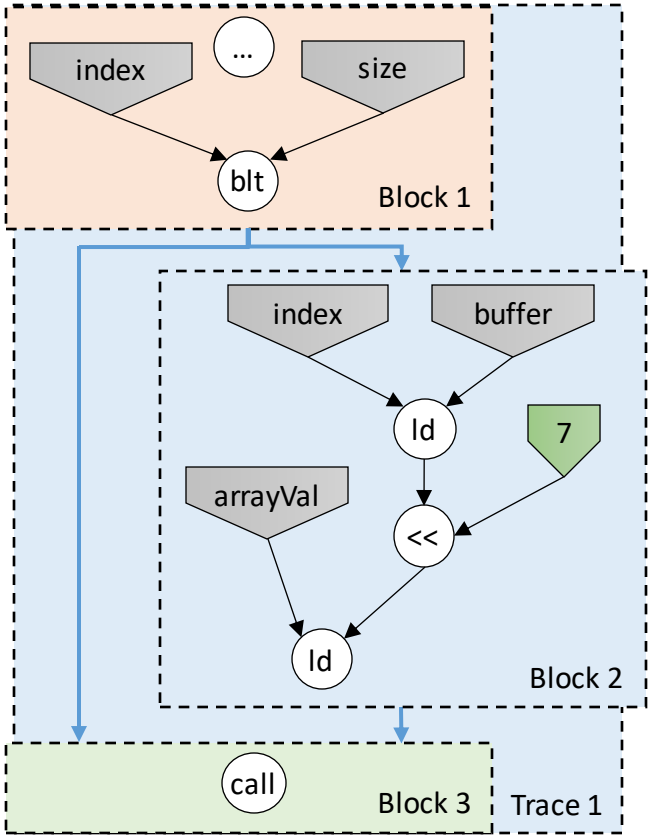
- Overview of Hybrid-DBT Optimization Flow
- **Spectre v1: Bound check bypass**
- Spectre v4: Speculative Store Bypass
- Proposed countermeasure

Spectre v1: Bound check bypass

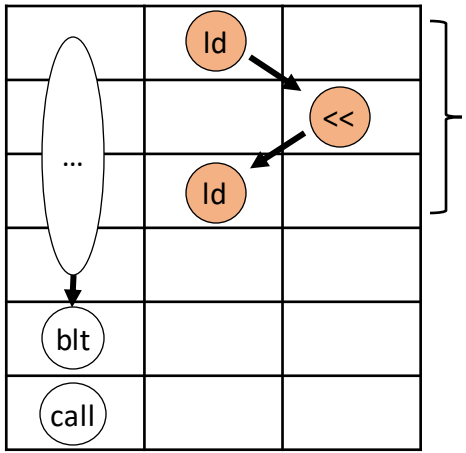
```
char buffer[size];
char arrayVal[256*128];

if (index < size){
    char a = buffer[index];
    char b = arrayVal[a * 128];
}

//Extracting the value from the cache
inspectCache();
```

- **Principle of Spectre v1 attacks:**
  - **The conditional jump is not resolved**
  - **Next instruction are executed speculatively**
  - **Once the jump is resolved, instruction are canceled**
  - **BUT: the cache may've been altered**

Spectre v1 on DBT based processors



- **Builds an Intermediate Representation**
- **Trace construction**
- **Instruction Scheduling**



Write in temporary registers

Outline

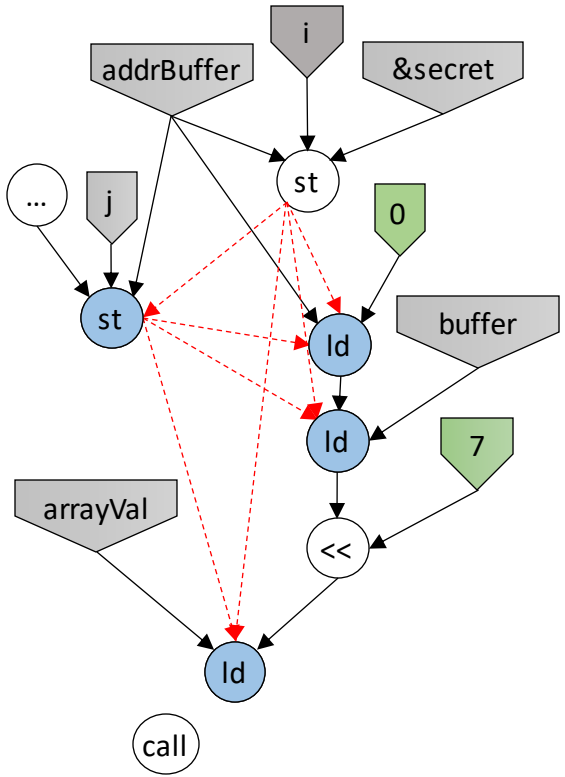
- Overview of Hybrid-DBT Optimization Flow
- Spectre v1: Bound check bypass
- **Spectre v4: Speculative Store Bypass**
- Proposed countermeasure

Spectre v4: Speculative Store Bypass

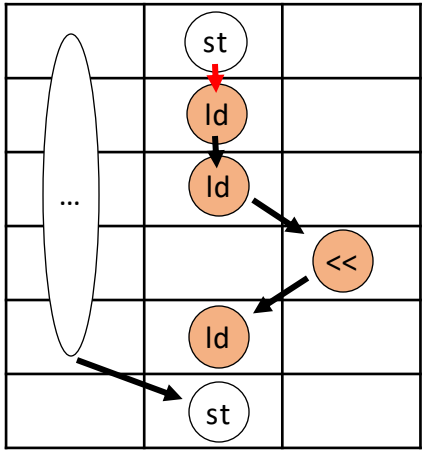
```
int  addrBuf[8];
char buffer[size];
char arrayVal[256*128];

addrBuf[i] = &secret - &buffer;
addrBuf[j] = ... ; //long computation

int a = addrBuf[0];
char b = buffer[a];
char c = buffer[b*128];

//Extracting the value from the cache
inspectCache();
```

- **Principle of Spectre v4 attacks:**
  - **We write the address of the secret in addrBuf[i]**
  - **We overwrite this address with a value resulting from a long computation (i == j == 0)**
  - **Before the end of this long computation:**
    - **We load the address stored in addrBuff[0]**
    - **We access memory at this address**
    - **We access a second array based on the value loaded**
  - **Memory write is commited**

Spectre v4 on DBT based processors



- **Builds an Intermediate Representation**
- **Speculates on RAW dependencies**
- **Instruction scheduling**



Caption

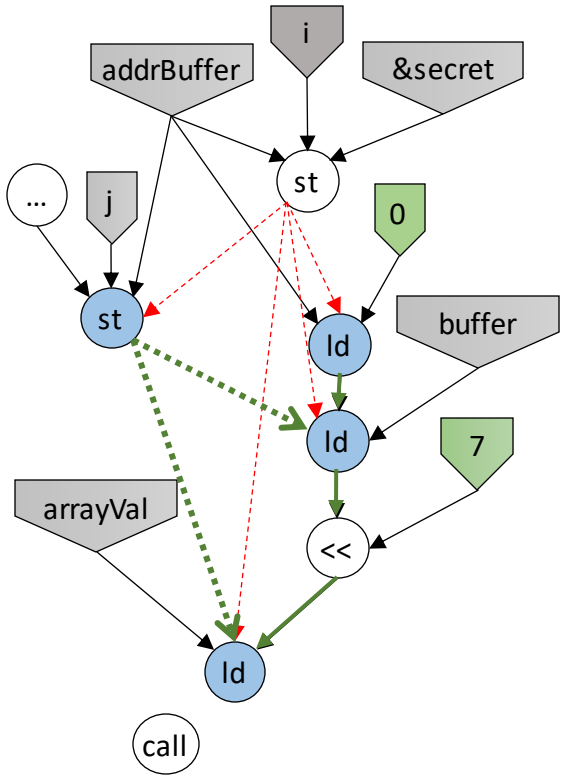
Memory dependency

st Speculative memory access

Outline

- Overview of Hybrid-DBT Optimization Flow
- Spectre v1: Bound check bypass
- Spectre v4: Speculative Store Bypass
- **Proposed countermeasure**

Proposed countermeasure



- **Identification of Spectre patterns**
  - **Tainting of speculated values**
  - **Add dependency when the address of a memory load is speculated**
- **Instruction are scheduled taking into account the new dependencies**

Caption

Tainted value

Additonnal dependency

Results & Discussion

- **The two variants are effectively mitigated**
  - **No observable performance loss on Hybrid-DBT benchmarks**

- **Mitigation specific to DBT based processors**
  - **Software DBT can be patched**
  - **Fine grained control of Instruction scheduling & speculative execution**

Conclusion

- **DBT based processors execute translated binaries on in-order micro-architecture**
- **Speculation is done by the translation engine**
- **DBT based processors are sensible to Spectre vulnerabilities**
- **Translation engine can be patched**