Attaque et sécurisation d'un schéma d'attestation à distance vérifié formellement

Jonathan CERTES †, Benoît MORGAN ‡

IRIT, Université Paul Sabatier, Université de Toulouse †
Intel Corporation, IRIT ‡

2 juin 2022

Attaque et sécurisation d'un schéma d'attestation à distance vérifié formellement

Jonathan CERTES †, Benoît MORGAN ‡

IRIT, Université Paul Sabatier, Université de Toulouse †
Intel Corporation, IRIT ‡

2 juin 2022

Contexte

- Détecter la compromission d'un algorithme **à distance**

Contexte

- Détecter la compromission d'un algorithme **à distance**
- Modèle de menace fort

Contexte

- Détecter la compromission d'un algorithme **à distance**
- Modèle de menace fort
- Adversaire sans accès physique

Contexte

- Détecter la compromission d'un algorithme **à distance**
- Modèle de menace fort
- Adversaire sans accès physique
- Vérification formelle possible

Contexte

- Détecter la compromission d'un algorithme **à distance**
- Modèle de menace fort
- Adversaire sans accès physique
- Vérification formelle possible
  - ~~ARM Trustzone~~, ~~Intel TXT (ACM), SGX, TDX~~

Contexte

- Détecter la compromission d'un algorithme **à distance**
- Modèle de menace fort
- Adversaire sans accès physique
- Vérification formelle possible
  - ~~ARM Trustzone~~, ~~Intel TXT (ACM), SGX, TDX~~

Cas d'étude

- Microprocesseur moderne

Contexte

- Détecter la compromission d'un algorithme **à distance**
- Modèle de menace fort
- Adversaire sans accès physique
- Vérification formelle possible
  - ~~ARM Trustzone~~, ~~Intel TXT (ACM), SGX, TDX~~

Cas d'étude

- Microprocesseur moderne
- Attestation à distance

Contexte

- Détecter la compromission d'un algorithme **à distance**
- Modèle de menace fort
- Adversaire sans accès physique
- Vérification formelle possible
  - ~~ARM Trustzone~~, ~~Intel TXT (ACM), SGX, TDX~~

Cas d'étude

- Microprocesseur moderne
- Attestation à distance
  - Protocole

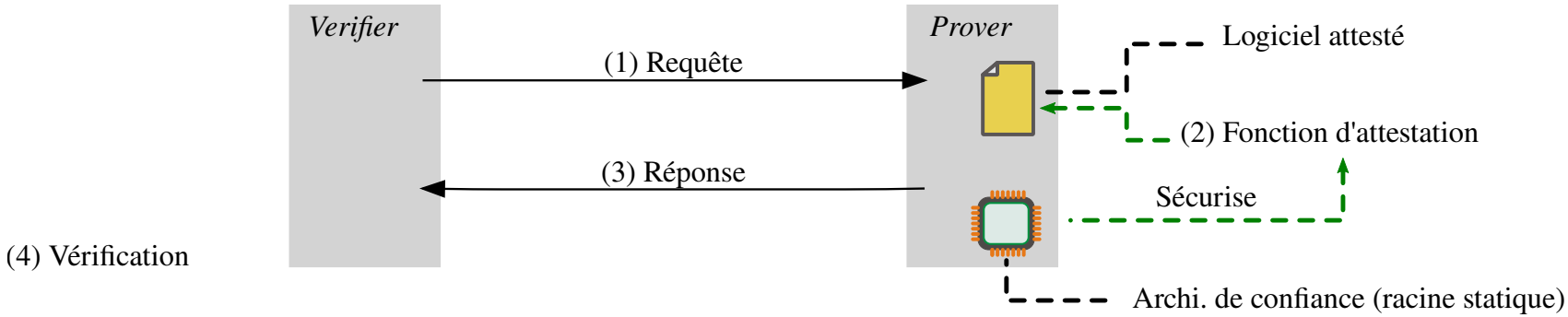
Contexte

- Détecter la compromission d'un algorithme **à distance**
- Modèle de menace fort
- Adversaire sans accès physique
- Vérification formelle possible
  - ~~ARM Trustzone~~, ~~Intel TXT (ACM), SGX, TDX~~

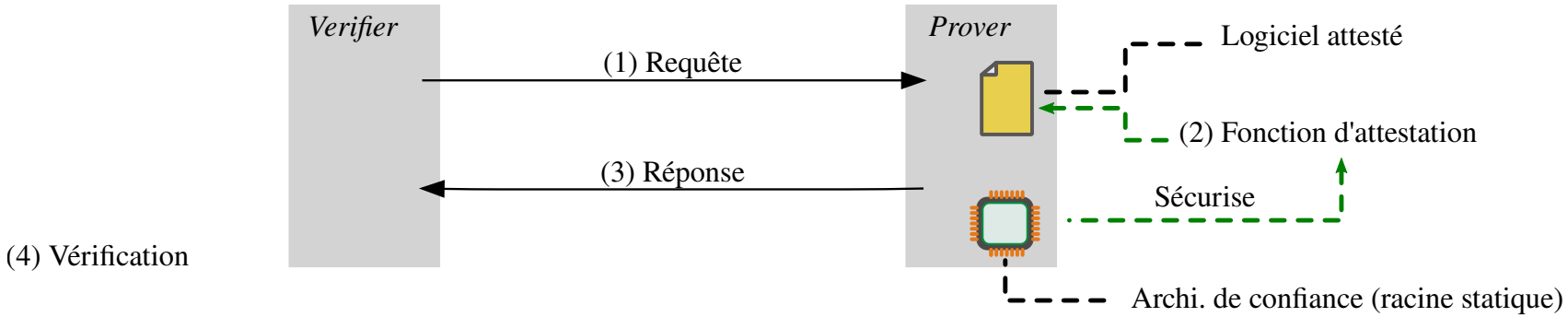
Cas d'étude

- Microprocesseur moderne
- Attestation à distance
  - Protocole
  - Architecture de confiance (racine de confiance statique)

Attestation à distance

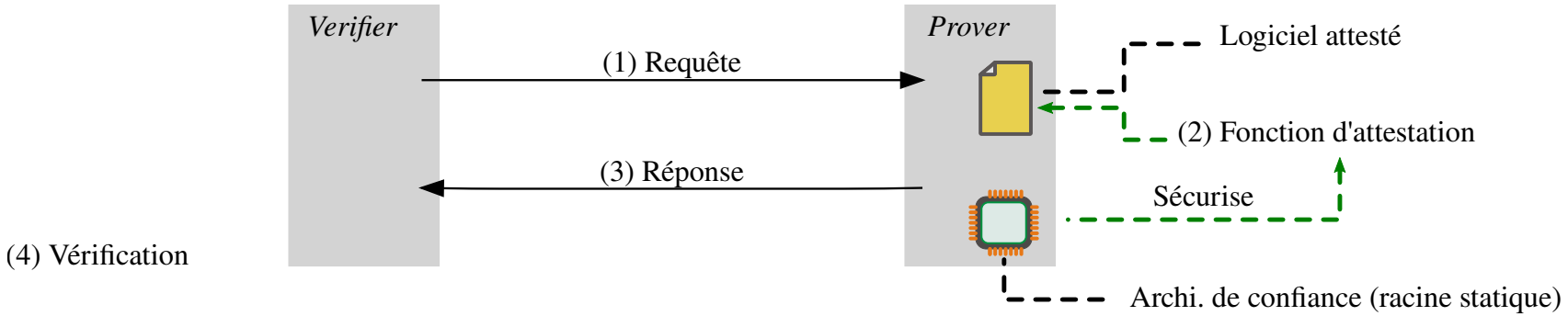


Attestation à distance



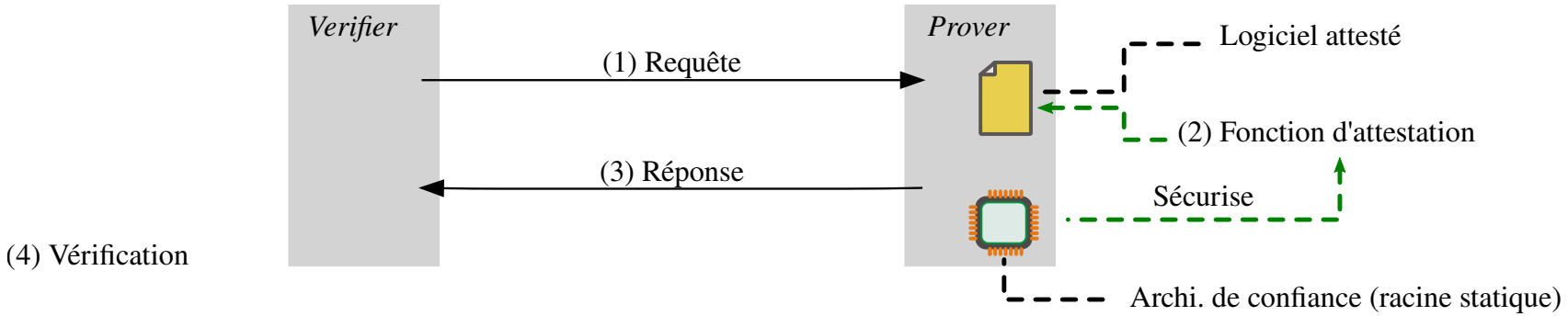
1. $\mathcal{V}$rf envoie une requête ainsi qu'un challenge à $\mathcal{P}$rv.

Attestation à distance



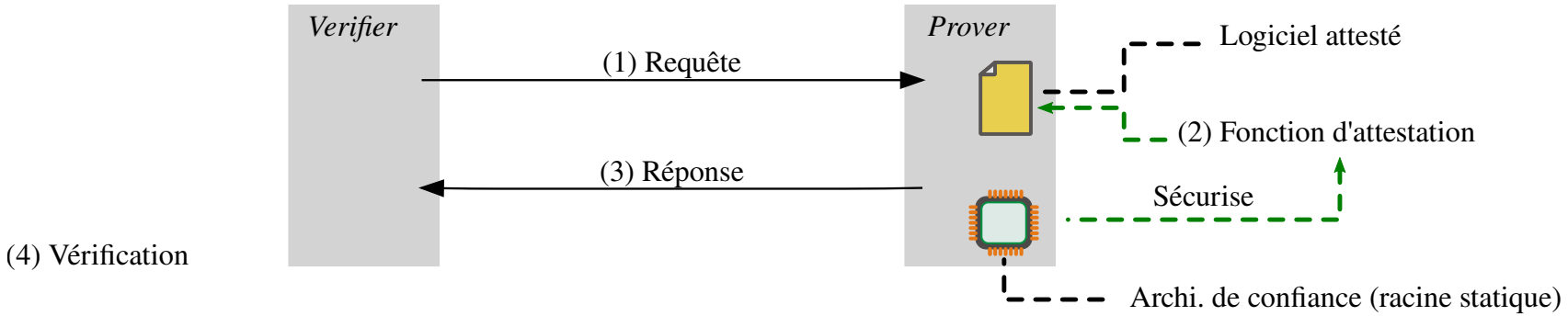
1. $\mathcal{V}$rf envoie une requête ainsi qu'un challenge à $\mathcal{P}$rv.
2. $\mathcal{P}$rv calcule un test d'intégrité authentifié Σ sur son environnement et le challenge.
   fonction d'attestation $\in$ architecture de confiance.

Attestation à distance



1. $\mathcal{V}$rf envoie une requête ainsi qu'un challenge à $\mathcal{P}$rv.
2. $\mathcal{P}$rv calcule un test d'intégrité authentifié Σ sur son environnement et le challenge.
   fonction d'attestation $\in$ architecture de confiance.
3. $\mathcal{P}$rv renvoie Σ à $\mathcal{V}$rf

Attestation à distance



1. $\mathcal{V}$rf envoie une requête ainsi qu'un challenge à $\mathcal{P}$rv.
2. $\mathcal{P}$rv calcule un test d'intégrité authentifié Σ sur son environnement et le challenge.
   fonction d'attestation $\in$ architecture de confiance.
3. $\mathcal{P}$rv renvoie Σ à $\mathcal{V}$rf
4. $\mathcal{V}$rf vérifie Σ et décide s'il correspond à un état de $\mathcal{P}$rv valide

Modèle d'attaquant

Objectif :

masquer la corruption de l'environnement du $\mathcal{P}rv$

Modèle d'attaquant

Objectif :

masquer la corruption de l'environnement du $\mathcal{P}rv$

- en altérant l'exécution de la fonction d'attestation

Modèle d'attaquant

Objectif :

masquer la corruption de l'environnement du $\mathcal{P}rv$

- en altérant l'exécution de la fonction d'attestation
- en obtenant le secret

Modèle d'attaquant

Objectif :

masquer la corruption de l'environnement du $\mathcal{P}rv$

- en altérant l'exécution de la fonction d'attestation
- en obtenant le secret

Modèle d'attaquant

Objectif :

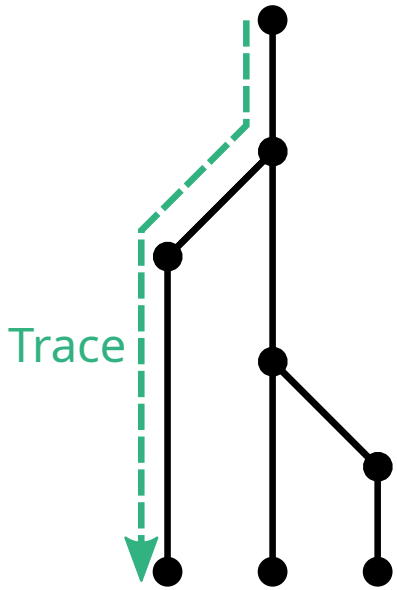
masquer la corruption de l'environnement du $\mathcal{P}rv$

- en altérant l'exécution de la fonction d'attestation
- en obtenant le secret

Rappels techniques

Rappels techniques

- *CoreSight* : périphérique de *debug*



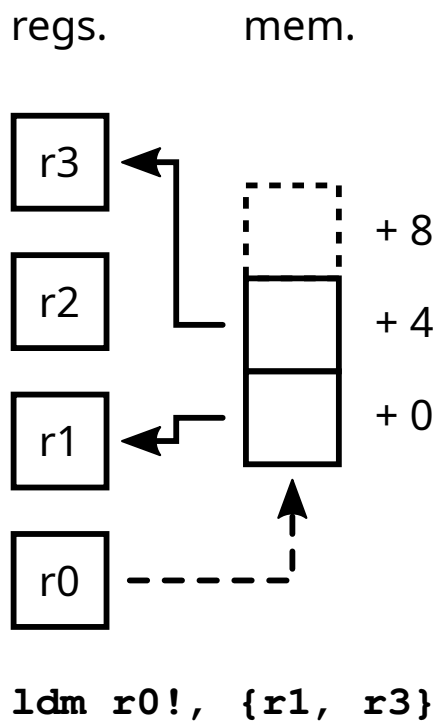
Graphe de flôt de contrôle

Rappels techniques

- *CoreSight* : périphérique de *debug*
- AXI / AXI-Lite : bus de communication

Rappels techniques

- *CoreSight* : périphérique de *debug*
- AXI / AXI-Lite : bus de communication
- Chargement multiple : `ldm r0!, {r1, r3}`

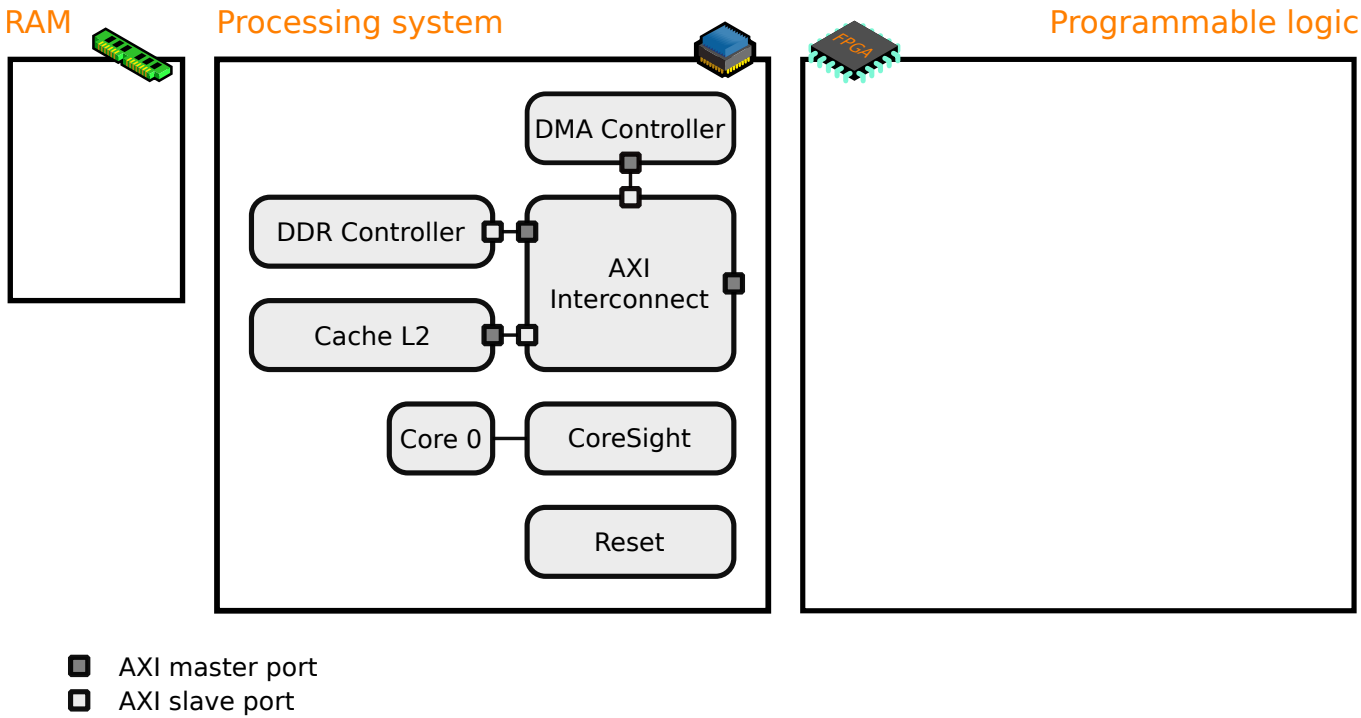


Rappels techniques

- *CoreSight* : périphérique de *debug*
- AXI / AXI-Lite : bus de communication
- Chargement multiple : `ldm r0!, {r1, r3}`
- Branchement indirect

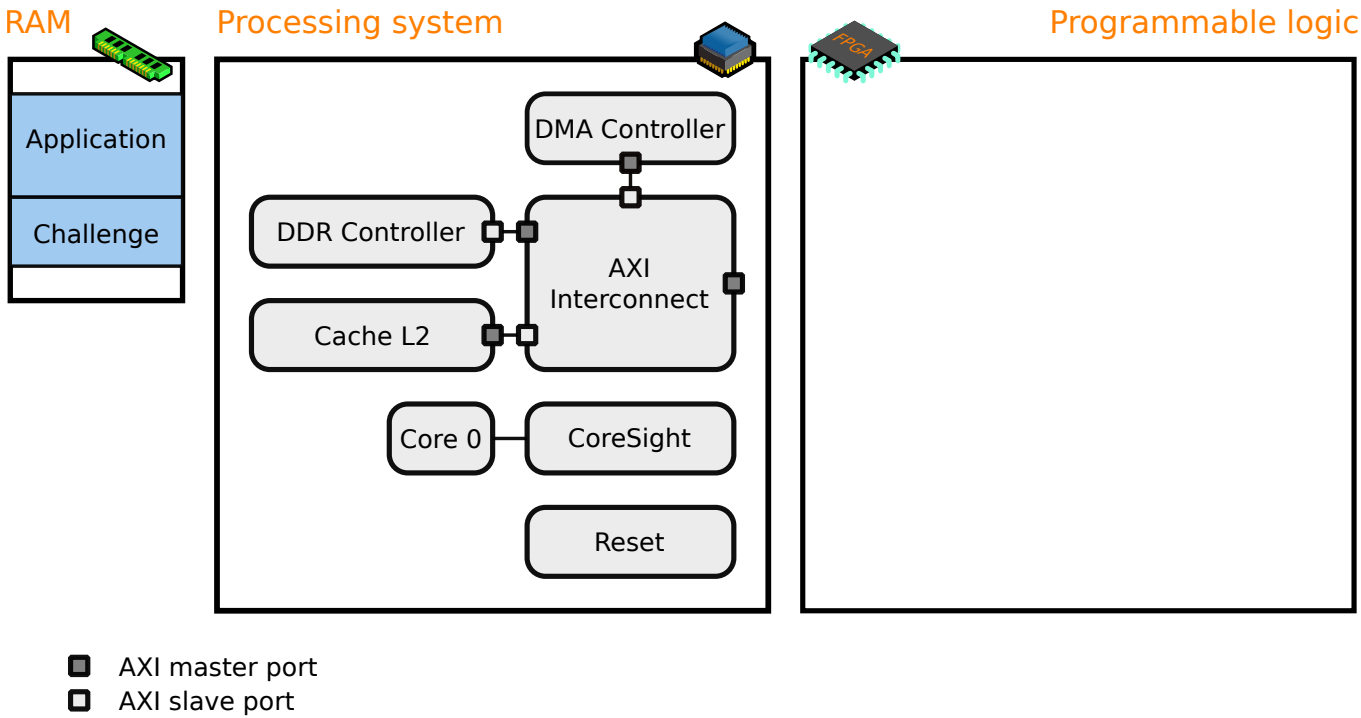
Architecture de confiance : Xilinx Zynq-7000 : ARM v7 + FPGA Artix 7

Architecture de confiance : Xilinx Zynq-7000 : ARM v7 + FPGA Artix 7



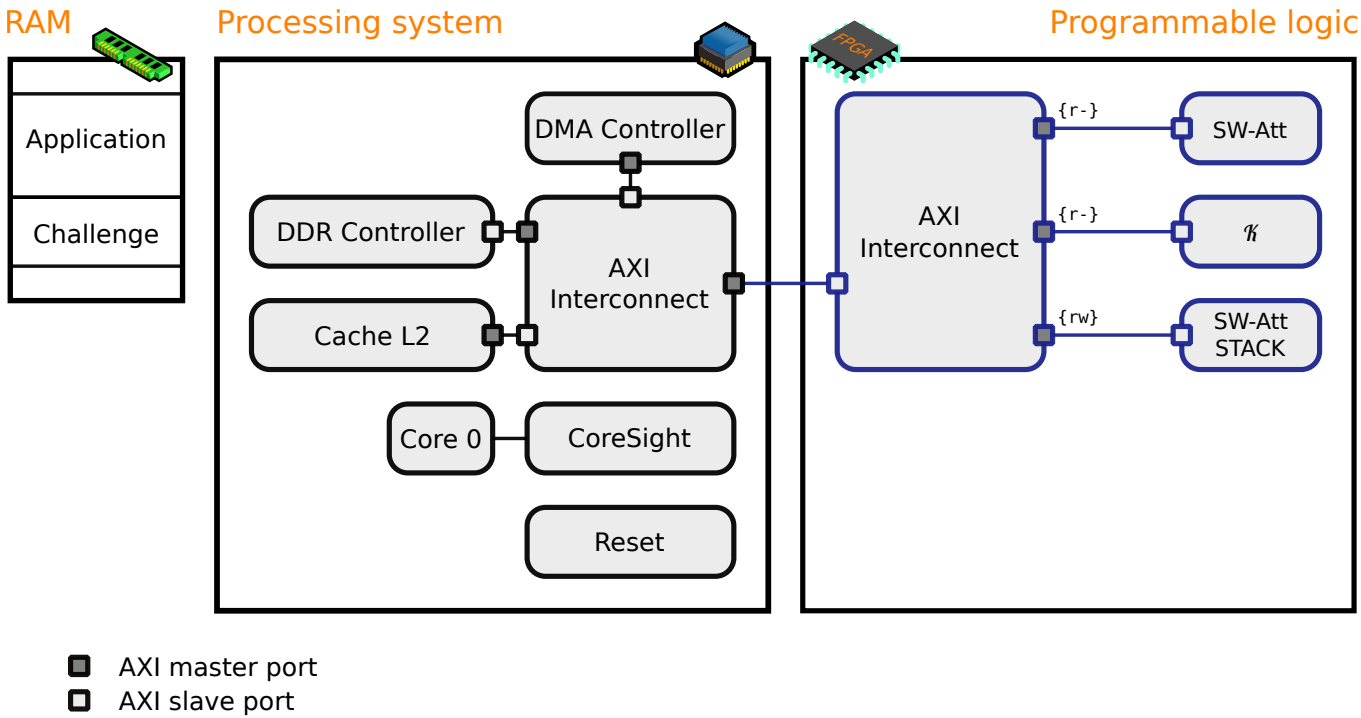
Implémentation de l'architecture de confiance

Architecture de confiance : Xilinx Zynq-7000 : ARM v7 + FPGA Artix 7



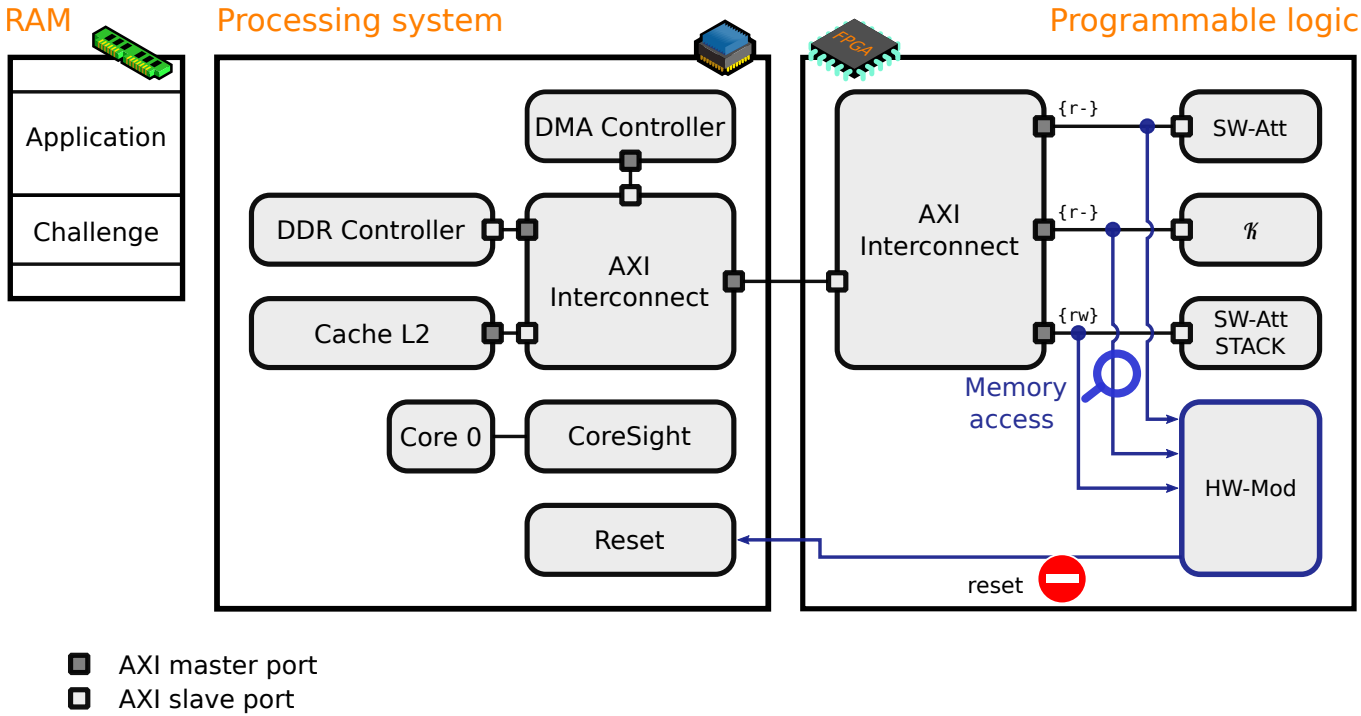
Le logiciel attesté et le challenge sont en mémoire que l'attaquant peut corrompre

Architecture de confiance : Xilinx Zynq-7000 : ARM v7 + FPGA Artix 7



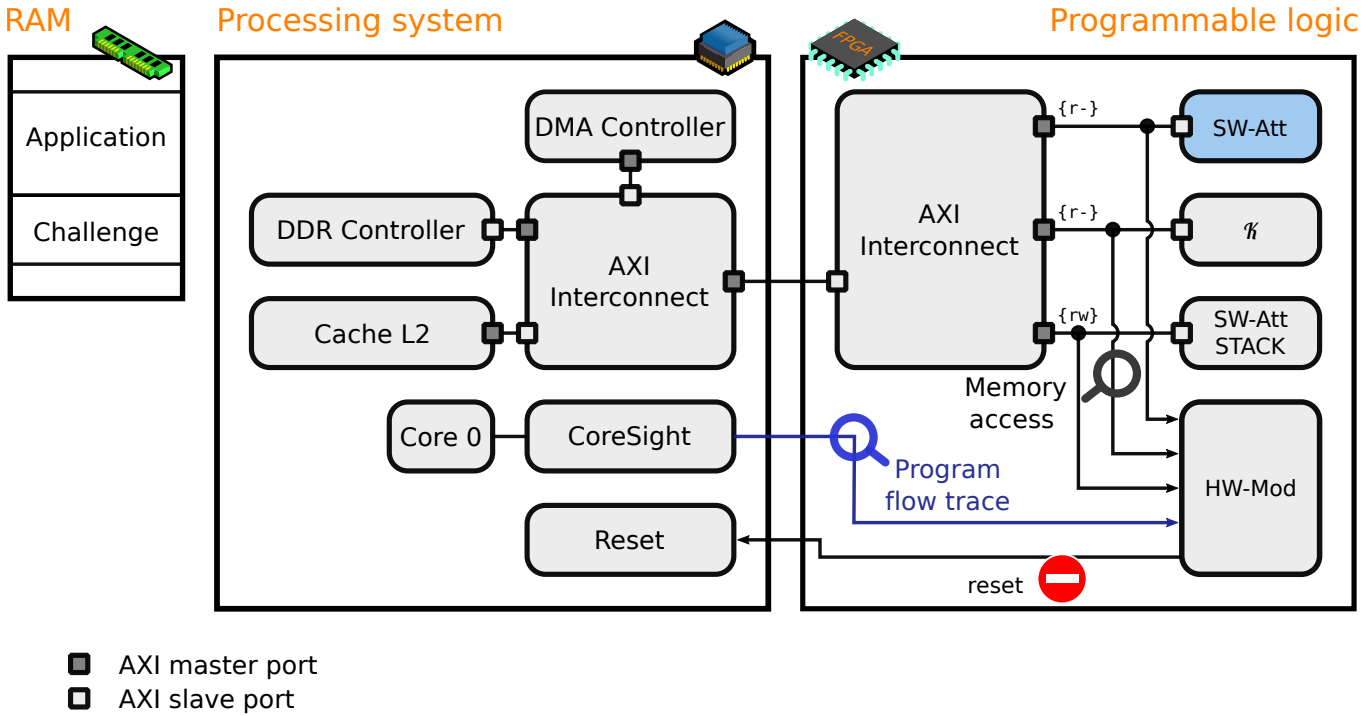
création de zones mémoires protégées

Architecture de confiance : Xilinx Zynq-7000 : ARM v7 + FPGA Artix 7



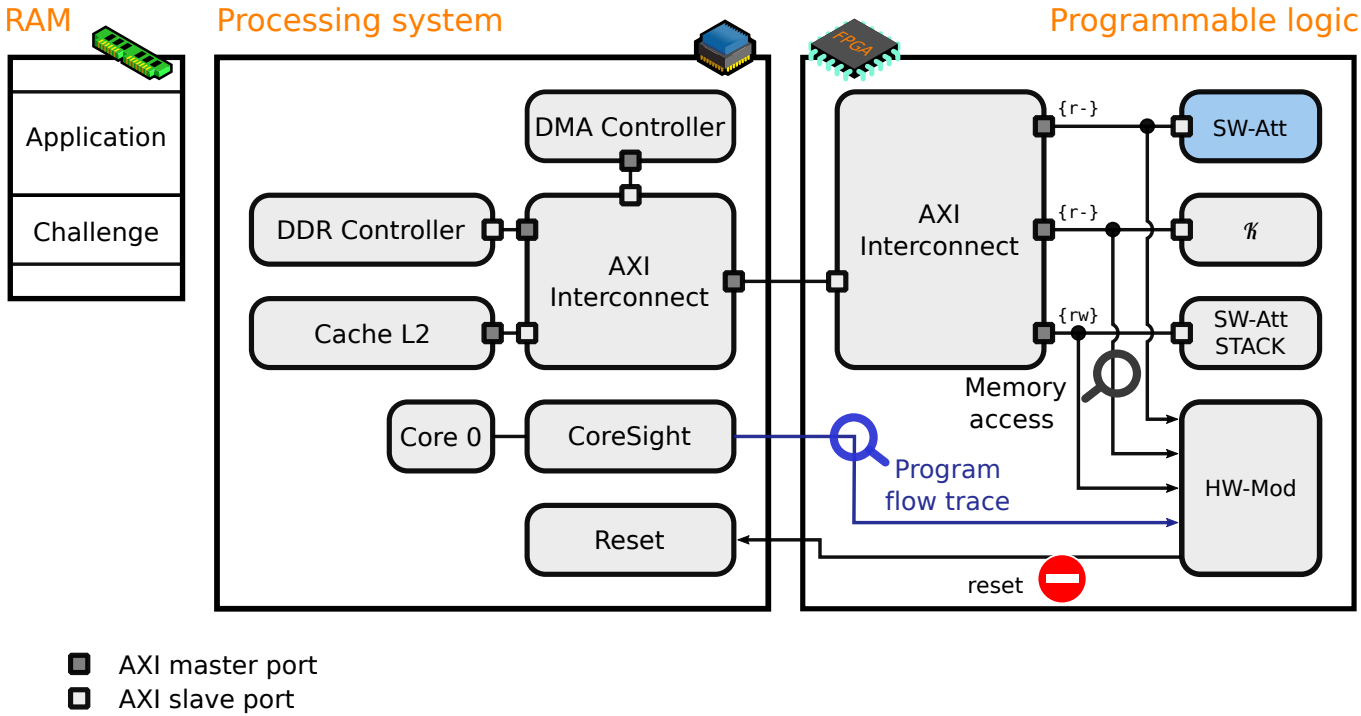
moniteur AXI : contrôle d'accès sur le bus

Architecture de confiance : Xilinx Zynq-7000 : ARM v7 + FPGA Artix 7



CoreSight + AXI : reconstruire le flot d'exécution

Architecture de confiance : Xilinx Zynq-7000 : ARM v7 + FPGA Artix 7



Pas de confiance au CPU corrompu

$\rightarrow$ Reconstruction d'un mécanisme de gestion de privilège externe

Vérification formelle du moniteur

Réutilisation de moniteur vérifié formellement

Vérification formelle du moniteur

Réutilisation de moniteur vérifié formellement

- modèle de menace haut niveau

Vérification formelle du moniteur

Réutilisation de moniteur vérifié formellement

- modèle de menace haut niveau
- modèle du microprocesseur plus ou moins abstrait

Vérification formelle du moniteur

Réutilisation de moniteur vérifié formellement

- modèle de menace haut niveau
- modèle du microprocesseur plus ou moins abstrait
- propriétés vérifiés sur le moniteur lui-même

Vérification formelle du moniteur

Réutilisation de moniteur vérifié formellement

- modèle de menace haut niveau
- modèle du microprocesseur plus ou moins abstrait
- propriétés vérifiés sur le moniteur lui-même

Remote attestation of bare-metal microprocessor software : a formally verified security monitor
J. Certes and B. Morgan (2021),
`https://hal.archives-ouvertes.fr/hal-03576711`

Vérification formelle du moniteur

Réutilisation de moniteur vérifié formellement

- modèle de menace haut niveau
- modèle du microprocesseur plus ou moins abstrait
- propriétés vérifiés sur le moniteur lui-même

Remote attestation of bare-metal microprocessor software : a formally verified security monitor
J. Certes and B. Morgan (2021),
`https://hal.archives-ouvertes.fr/hal-03576711`

$\rightarrow$ extension du modèle de menaces

Vérification formelle du moniteur

Réutilisation de moniteur vérifié formellement

- modèle de menace haut niveau
- modèle du microprocesseur plus ou moins abstrait
- propriétés vérifiés sur le moniteur lui-même

Remote attestation of bare-metal microprocessor software : a formally verified security monitor
J. Certes and B. Morgan (2021),
`https://hal.archives-ouvertes.fr/hal-03576711`

→ extension du modèle de menaces

1. empoisonnement des mémoires cache

Vérification formelle du moniteur

Réutilisation de moniteur vérifié formellement

- modèle de menace haut niveau
- modèle du microprocesseur plus ou moins abstrait
- propriétés vérifiés sur le moniteur lui-même

Remote attestation of bare-metal microprocessor software : a formally verified security monitor
J. Certes and B. Morgan (2021),
`https://hal.archives-ouvertes.fr/hal-03576711`

→ extension du modèle de menaces

1. empoisonnement des mémoires cache
2. reconfiguration de *CoreSight*

Vérification formelle du moniteur

Réutilisation de moniteur vérifié formellement

- modèle de menace haut niveau
- modèle du microprocesseur plus ou moins abstrait
- propriétés vérifiés sur le moniteur lui-même

Remote attestation of bare-metal microprocessor software : a formally verified security monitor
J. Certes and B. Morgan (2021),
`https://hal.archives-ouvertes.fr/hal-03576711`

→ extension du modèle de menaces

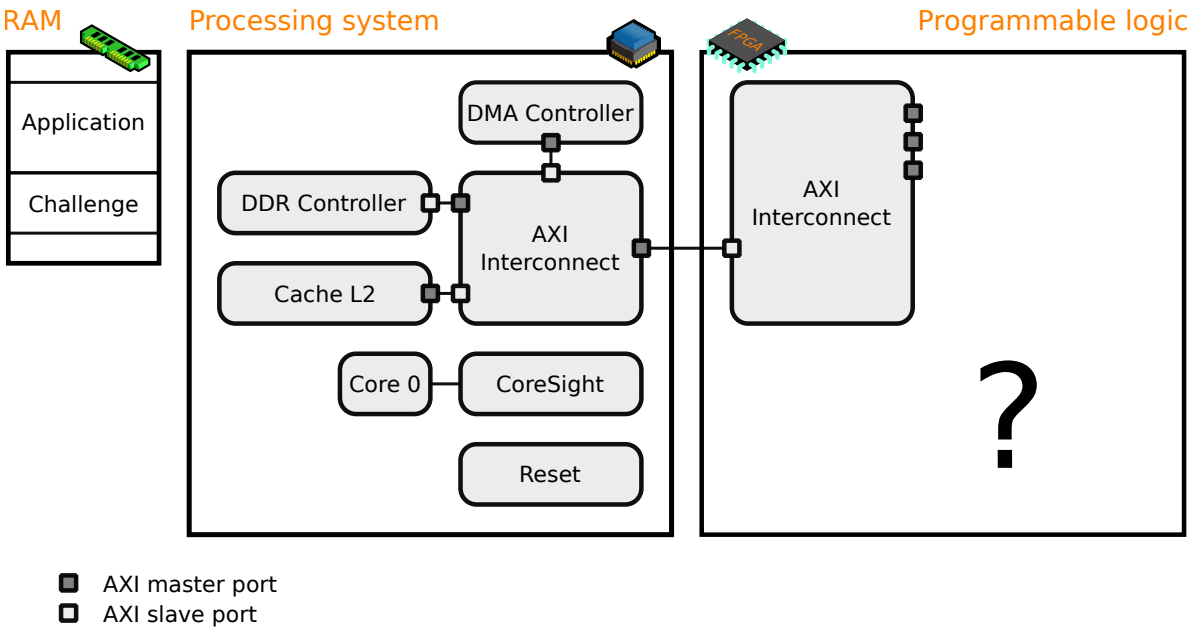
1. empoisonnement des mémoires cache
2. reconfiguration de *CoreSight*
3. reconfiguration de la MMU

Contribution

Objectif

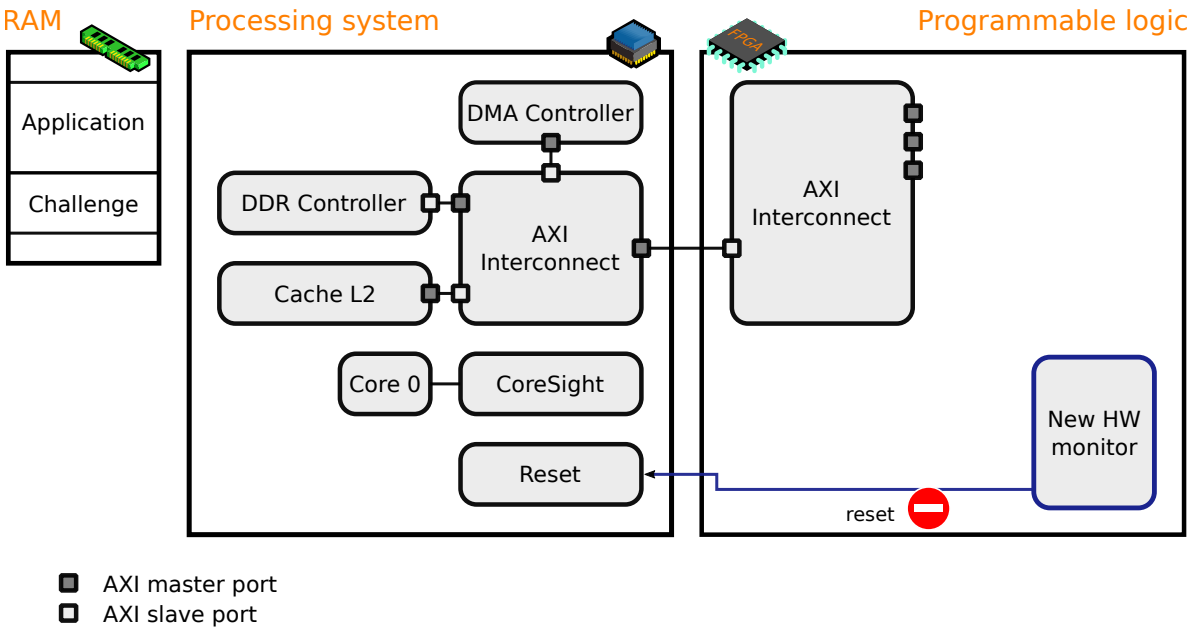
Proposer une extension qui permet de décharger le moniteur des hypothèses simplifiantes

Solution proposée



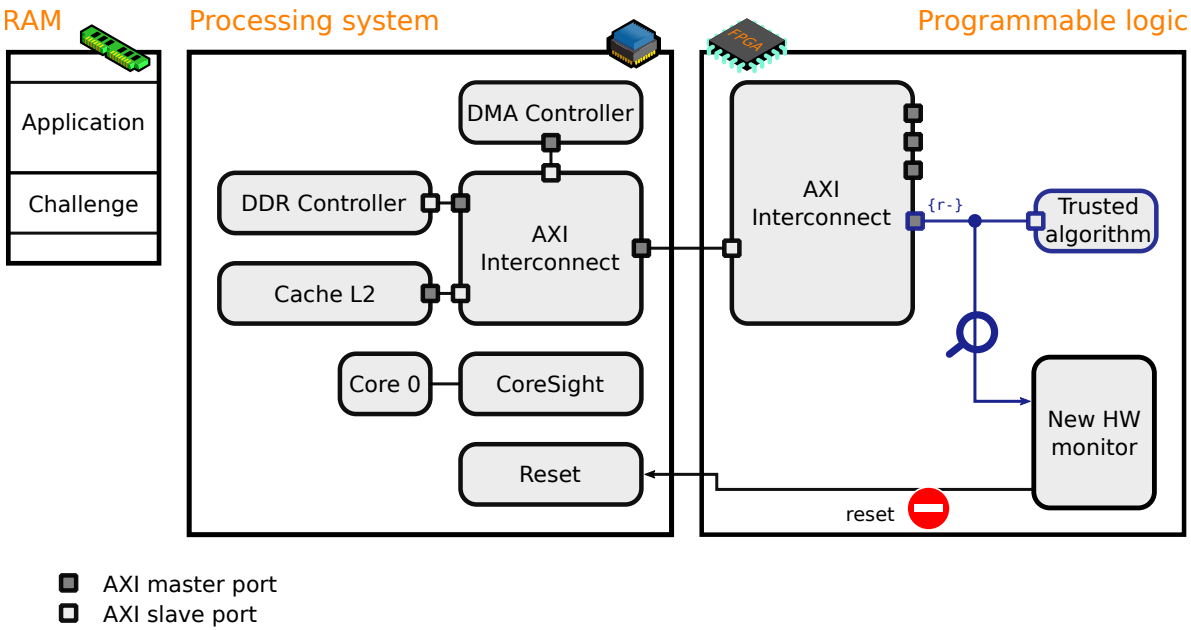
Solution proposée

- nouveau moniteur matériel : stoppe le microprocesseur



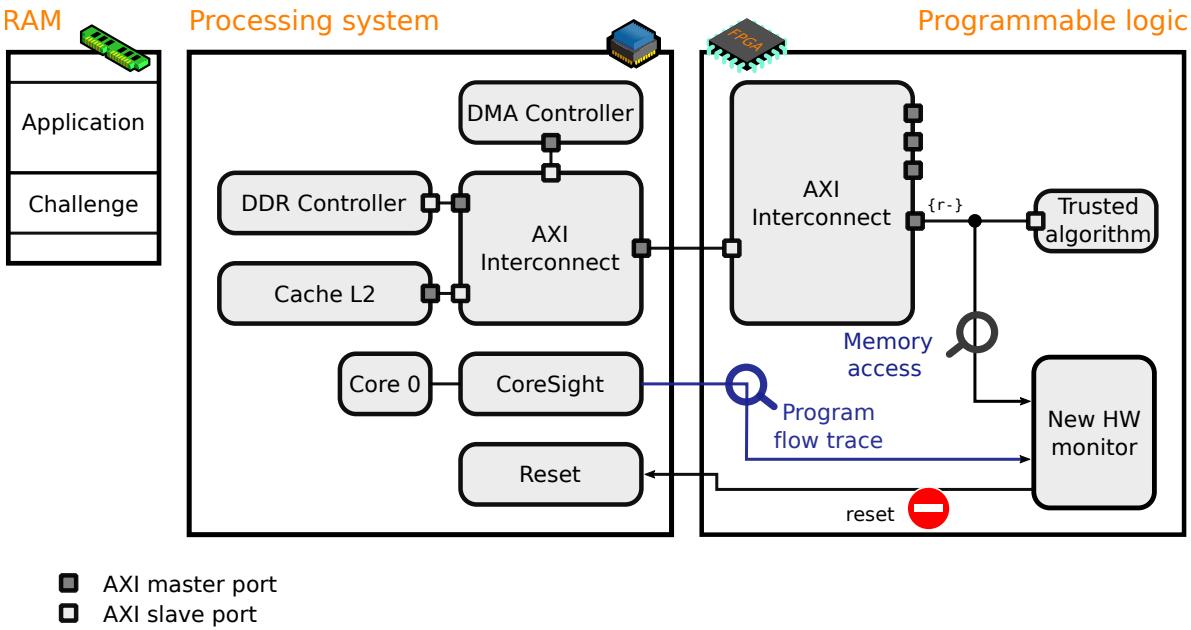
Solution proposée

- nouveau moniteur matériel : stoppe le microprocesseur
- ROM : algorithme de confiance



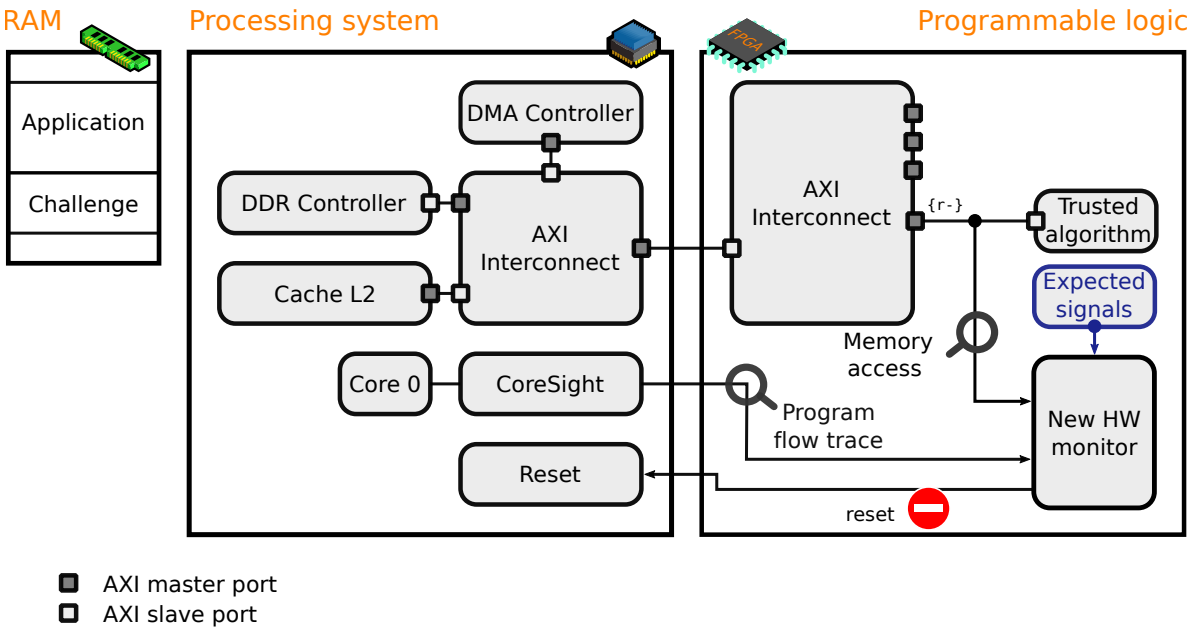
Solution proposée

- nouveau moniteur matériel : stoppe le microprocesseur
- ROM : algorithme de confiance
- observe les traces *Coresight*



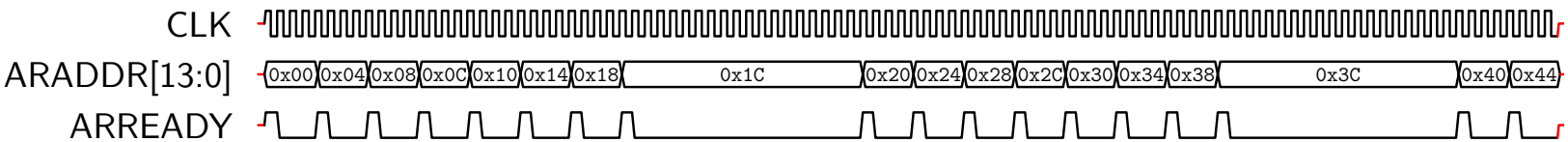
Solution proposée

- nouveau moniteur matériel : stoppe le microprocesseur
- ROM : algorithme de confiance
- observe les traces *Coresight*
- ROM : signaux attendus

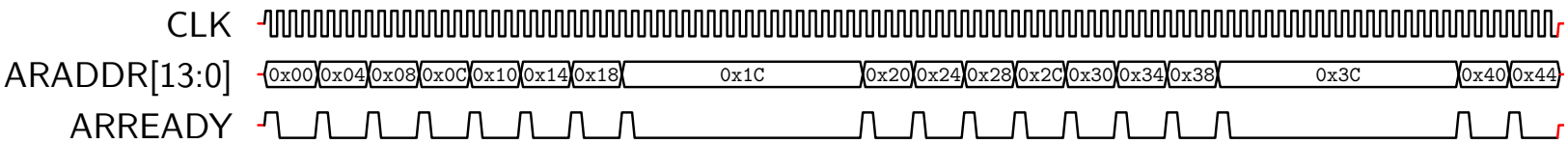


Caractérisation

Caractérisation

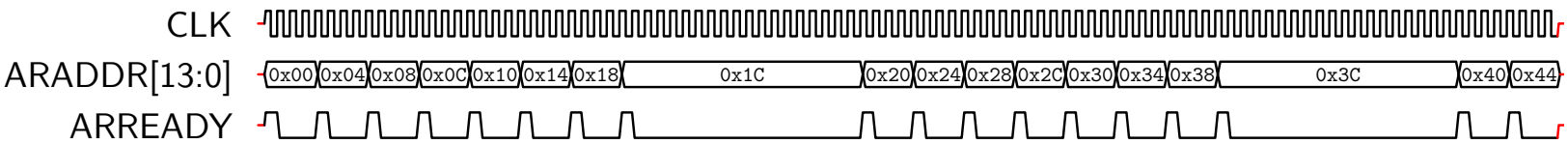


Caractérisation



- signaux AXI identiques entre *read* et *fetch*

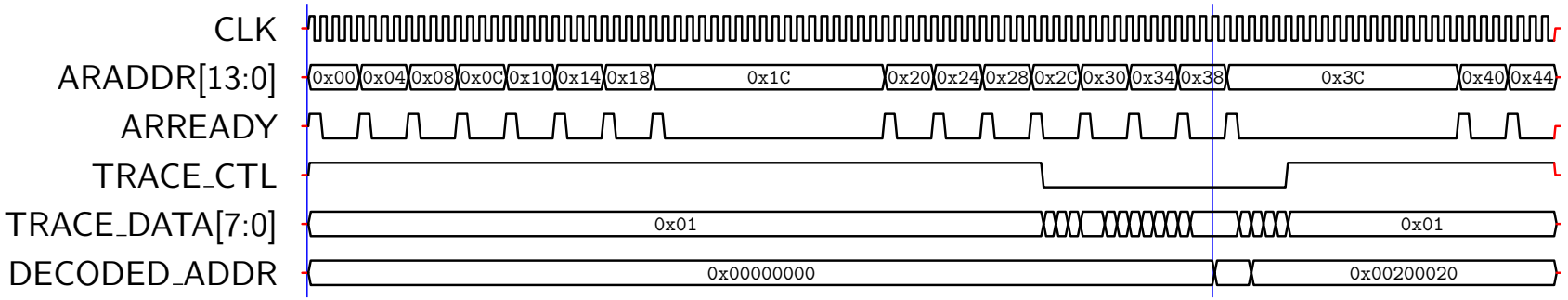
Caractérisation



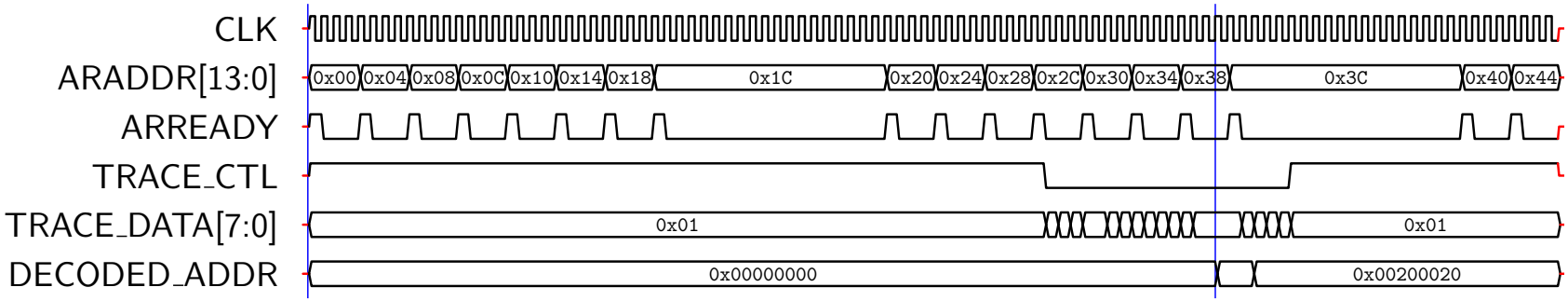
- signaux AXI identiques entre *read* et *fetch*

```
nop
nop
nop
nop
nop
nop
add r3, pc, #0   // pc = pc + 4;    r3 = pc + 4 + 0;
mov pc, r3       // indirect branch
// <- destination
```

Caractérisation

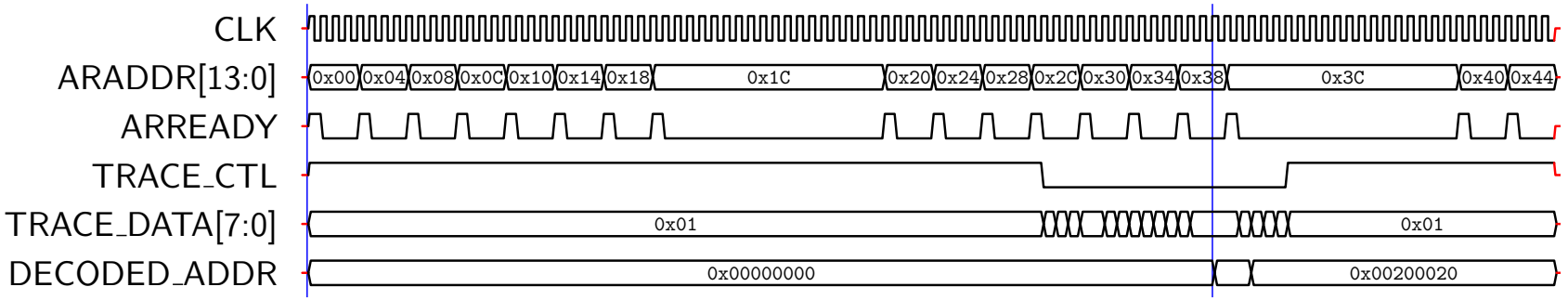


Caractérisation



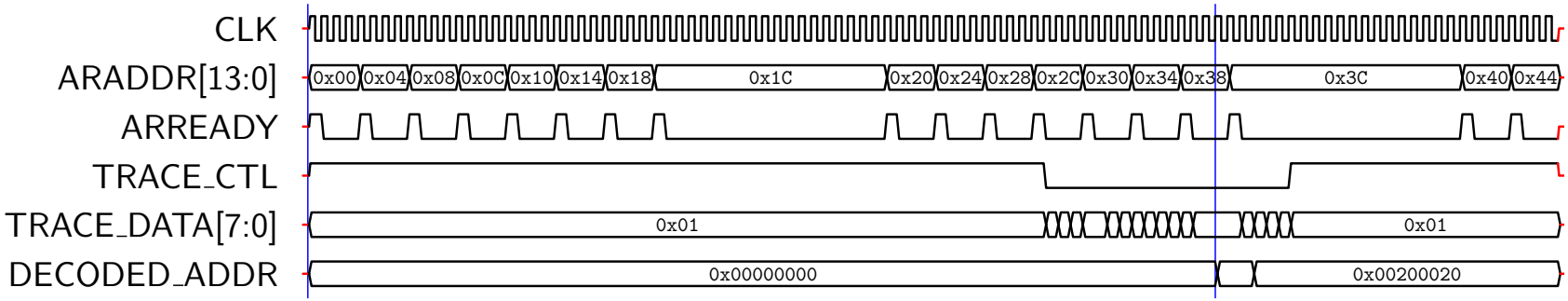
- faible latence $\rightarrow$ *fetch*

Caractérisation



- faible latence $\rightarrow$ *fetch*
- retard $\rightarrow$ *read* depuis un autre programme

Caractérisation

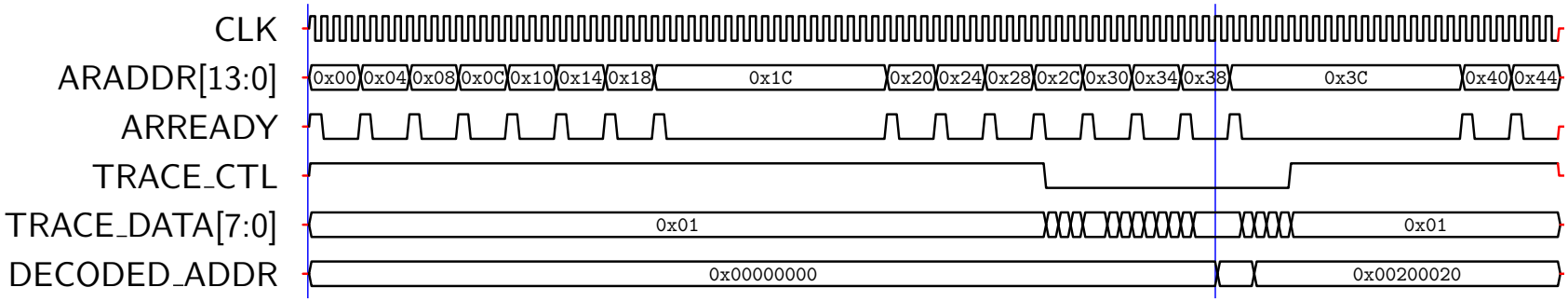


- faible latence $\rightarrow$ *fetch*
- retard $\rightarrow$ *read* depuis un autre programme

Vérification formelle

- exécution $\rightarrow$ signaux identiques (modélisation possible)

Caractérisation

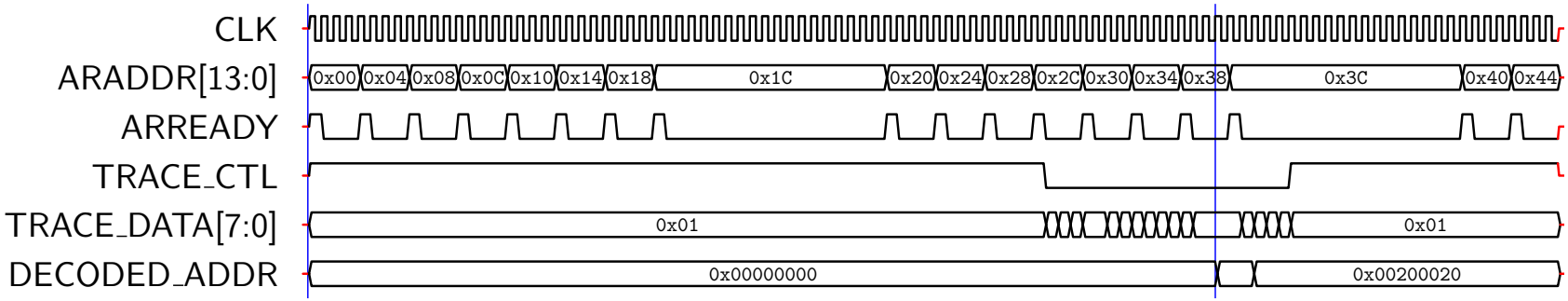


- faible latence $\rightarrow$ *fetch*
- retard $\rightarrow$ *read* depuis un autre programme

Vérification formelle

- exécution $\rightarrow$ signaux identiques (modélisation possible)
- signaux identiques $\rightarrow$ exécution ?

Caractérisation

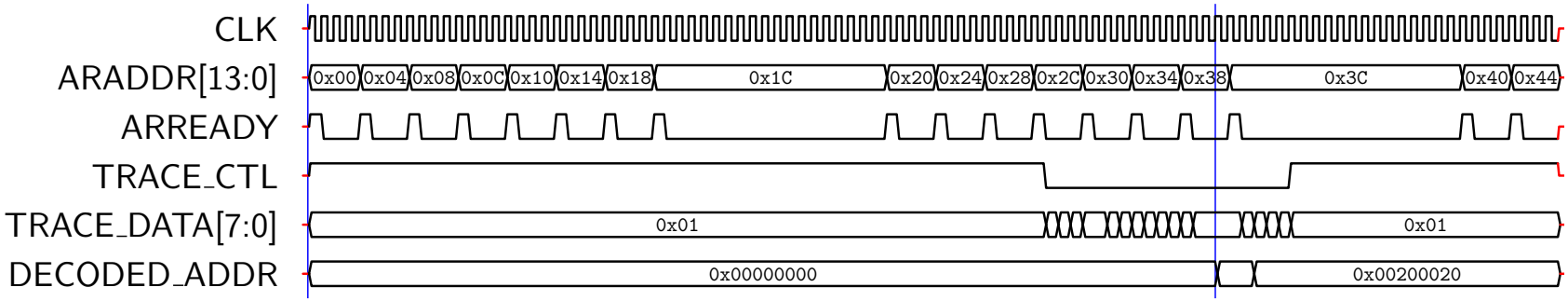


- faible latence $\rightarrow$ *fetch*
- retard $\rightarrow$ *read* depuis un autre programme

Vérification formelle

- exécution $\rightarrow$ signaux identiques (modélisation possible)
- signaux identiques $\rightarrow$ exécution ? (hypothèse)

Caractérisation



- faible latence $\rightarrow$ *fetch*
- retard $\rightarrow$ *read* depuis un autre programme

Vérification formelle

- exécution $\rightarrow$ signaux identiques (modélisation possible)
- signaux identiques $\rightarrow$ exécution ? (hypothèse)

Hypothèse

Nous pouvons écrire un logiciel tel qu'il est impossible pour un adversaire de reproduire les signaux d'accès lus par le moniteur.

Audit (exemple)

Audit (exemple)

1. mémoires cache

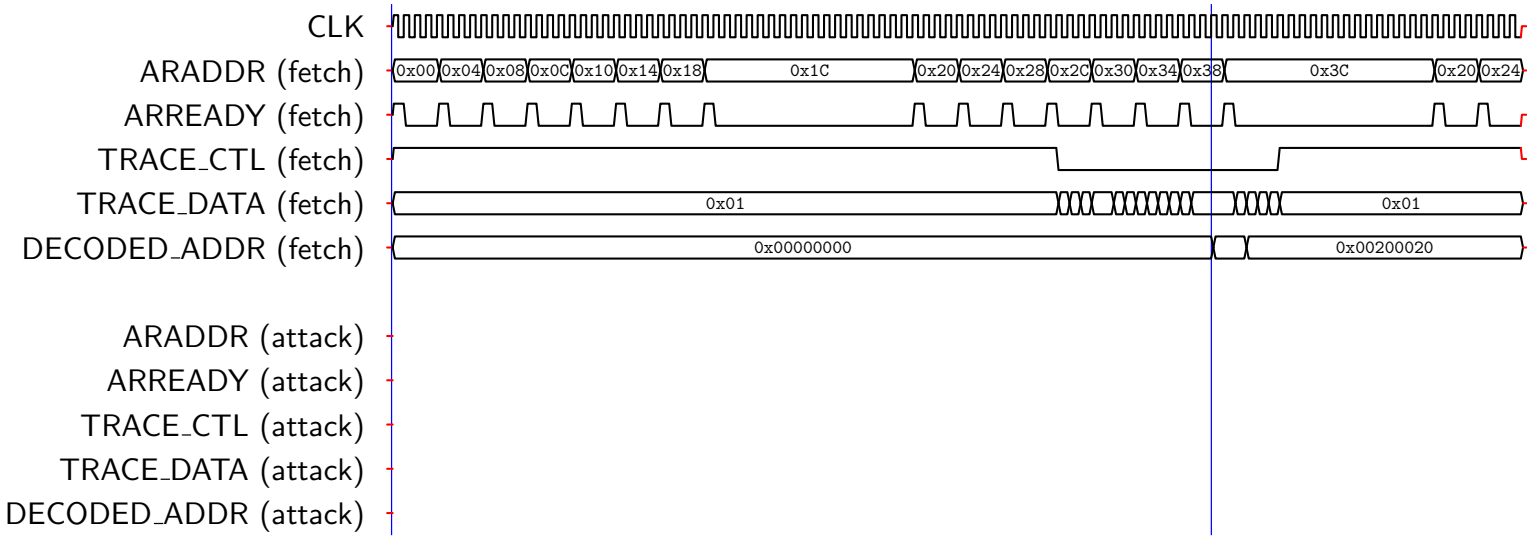
Audit (exemple)

1. mémoires cache
2. configuration de *CoreSight*

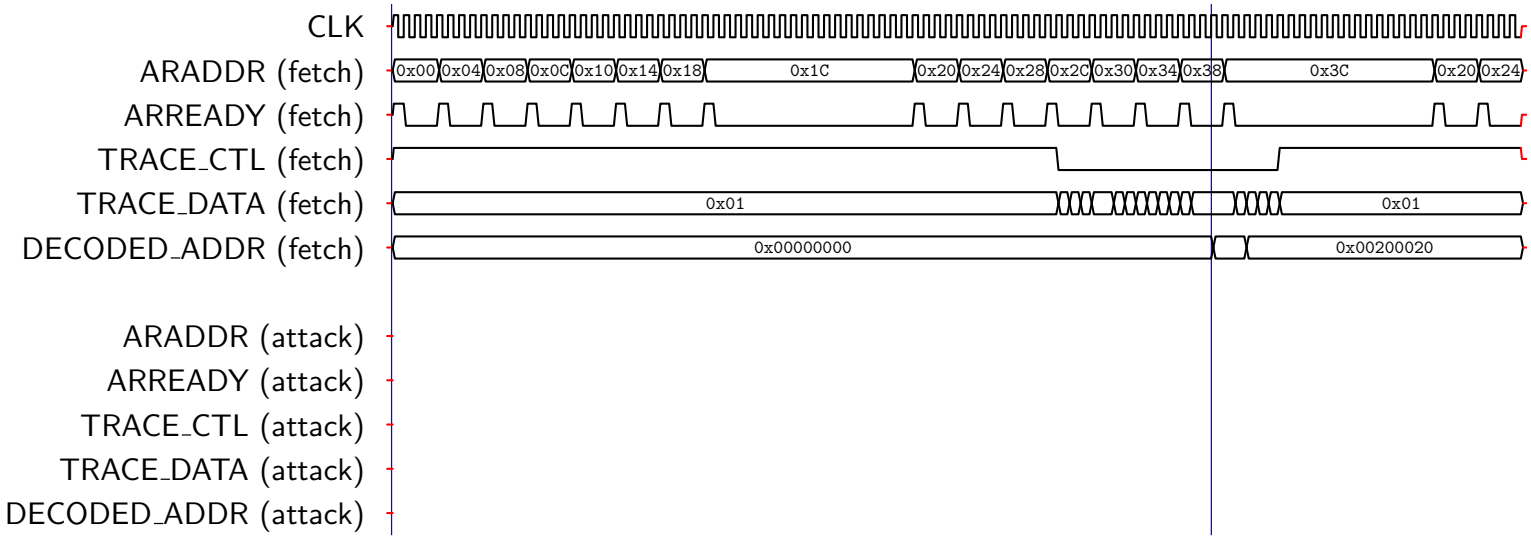
Audit (exemple)

1. mémoires cache
2. configuration de *CoreSight*
3. configuration de la MMU

Audit (exemple)

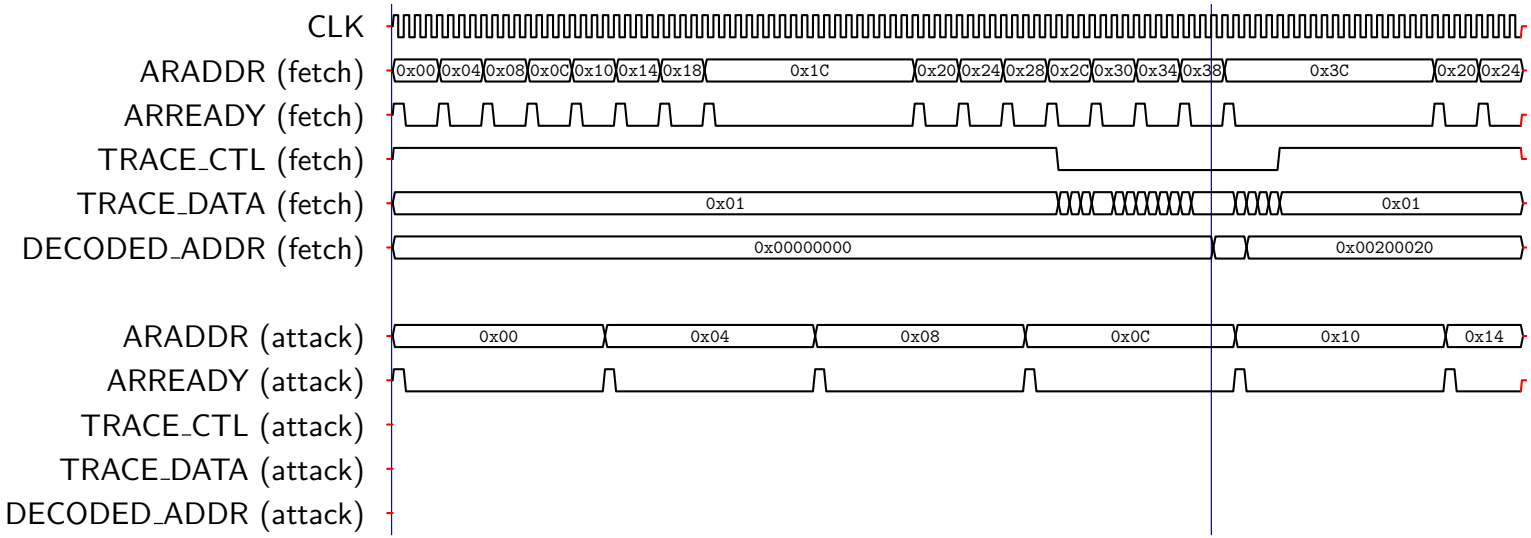


Audit (exemple)



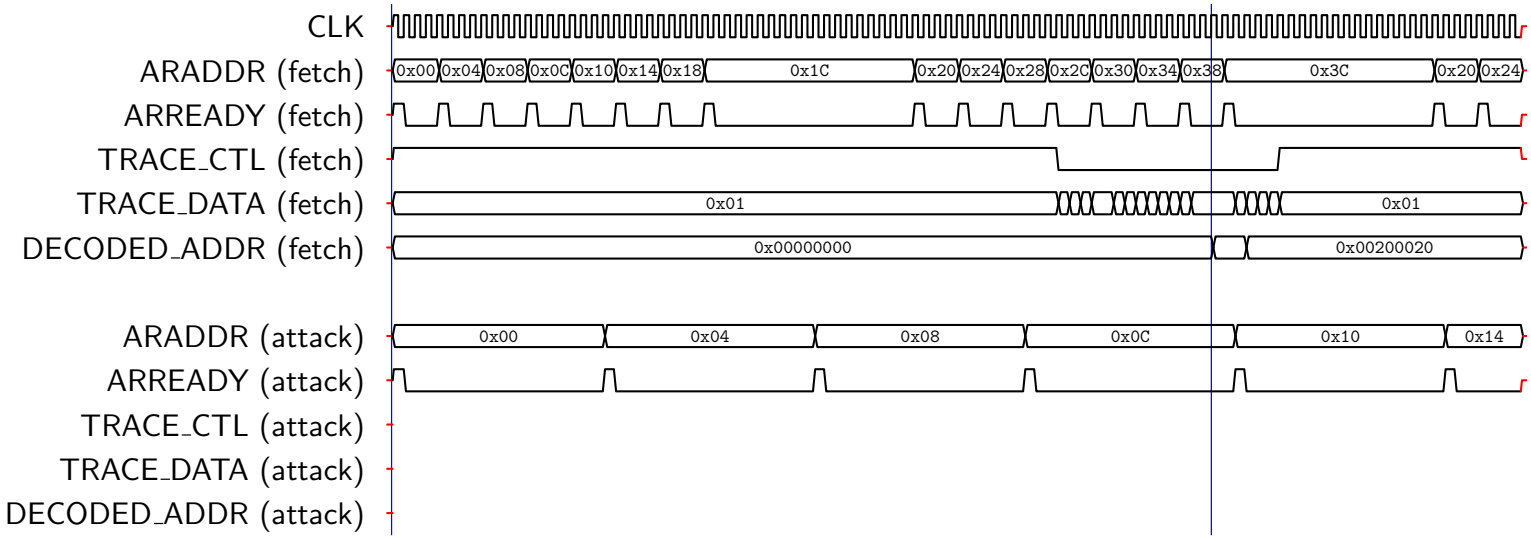

```
mov r0, #0x00200000 // virtual address
ldm r0!, {r1}       // r0 = r0 + 4 (!)
ldm r0!, {r1}
ldm r0!, {r1}
ldm r0!, {r1}
ldm r0!, {r1}
ldm r0!, {r1}
ldm r0!, {r1}
ldm r0!, {r1}
```

Audit (exemple)



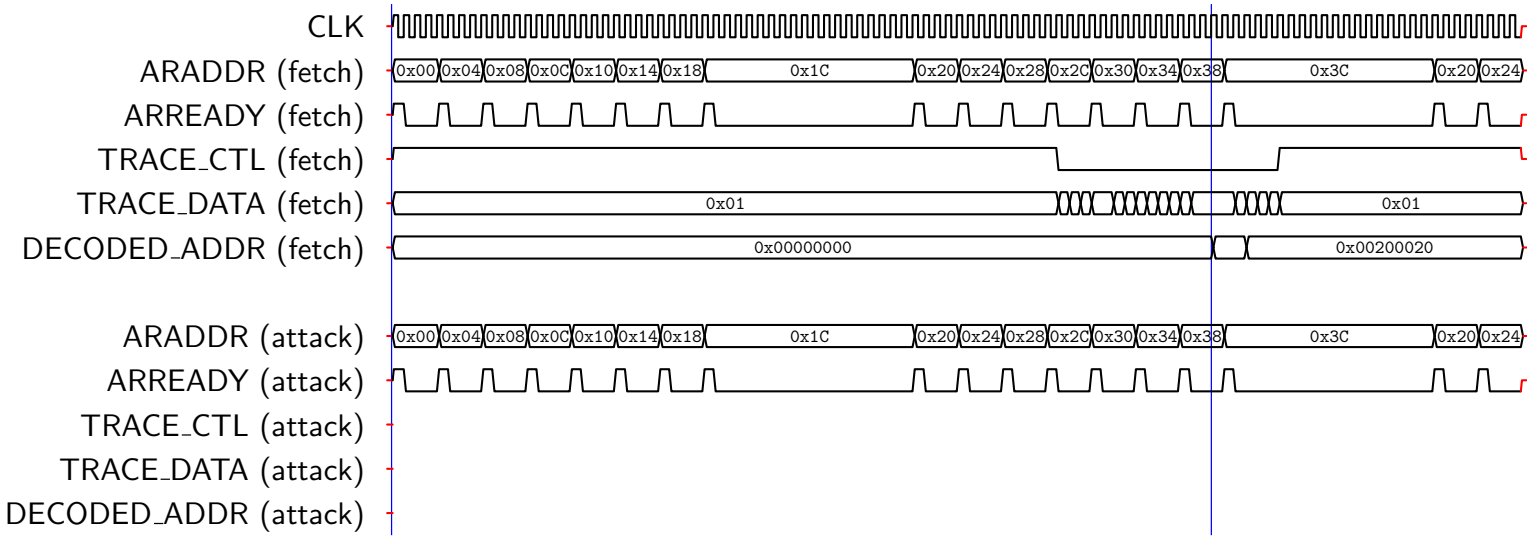

```
mov r0, #0x00200000 // virtual address
ldm r0!, {r1}       // r0 = r0 + 4 (!)
ldm r0!, {r1}
ldm r0!, {r1}
ldm r0!, {r1}
ldm r0!, {r1}
ldm r0!, {r1}
ldm r0!, {r1}
ldm r0!, {r1}
```

Audit (exemple)



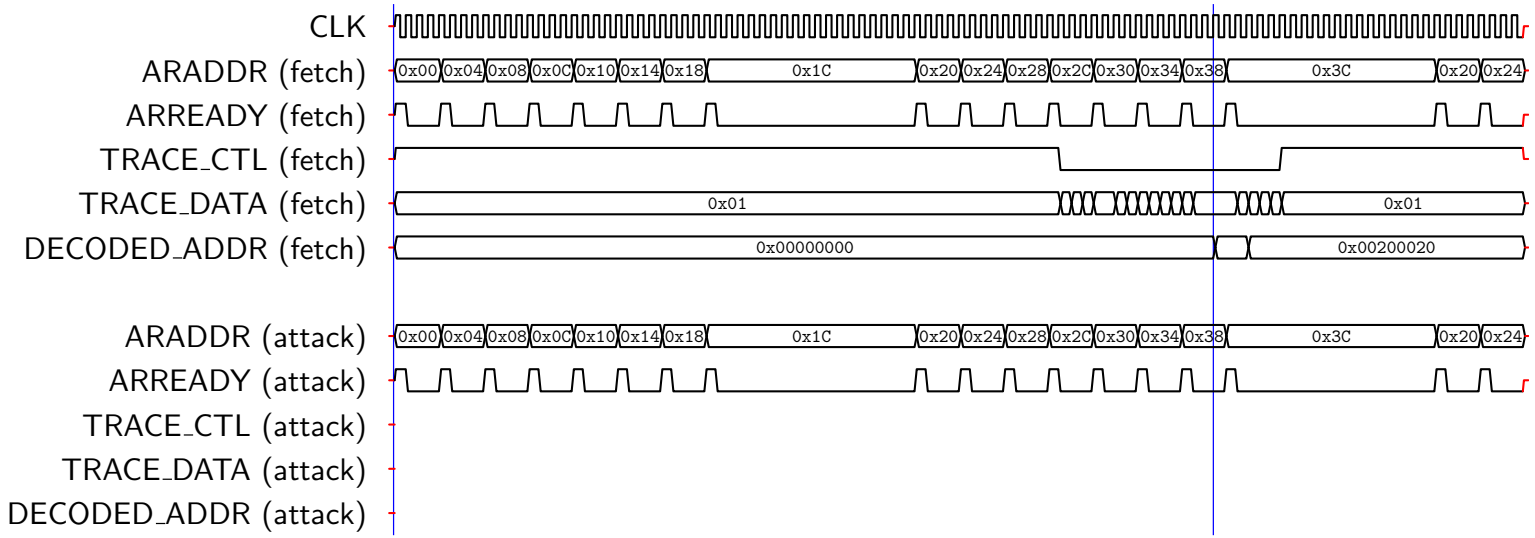

```
mov r0, #0x00200000                    // virtual address
ldm r0!, {r1,r2,r3,r4,r5,r6,r7,r8}     // r0 = r0 + 8 × 4 (!)
ldm r0 , {r1,r2,r3,r4,r5,r6,r7,r8}     // r0 = r0 + 0
// ...
```

Audit (exemple)



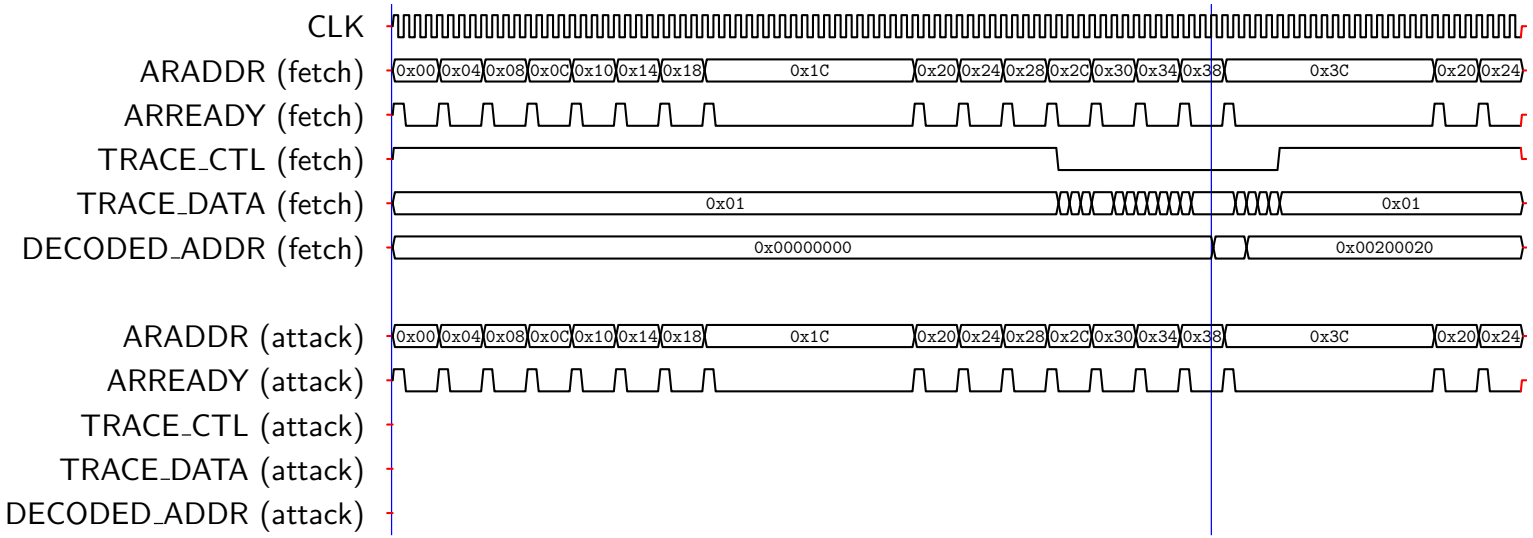

```
mov r0, #0x00200000                    // virtual address
ldm r0!, {r1,r2,r3,r4,r5,r6,r7,r8}     // r0 = r0 + 8 × 4 (!)
ldm r0 , {r1,r2,r3,r4,r5,r6,r7,r8}     // r0 = r0 + 0
// ...
```

Audit (exemple)



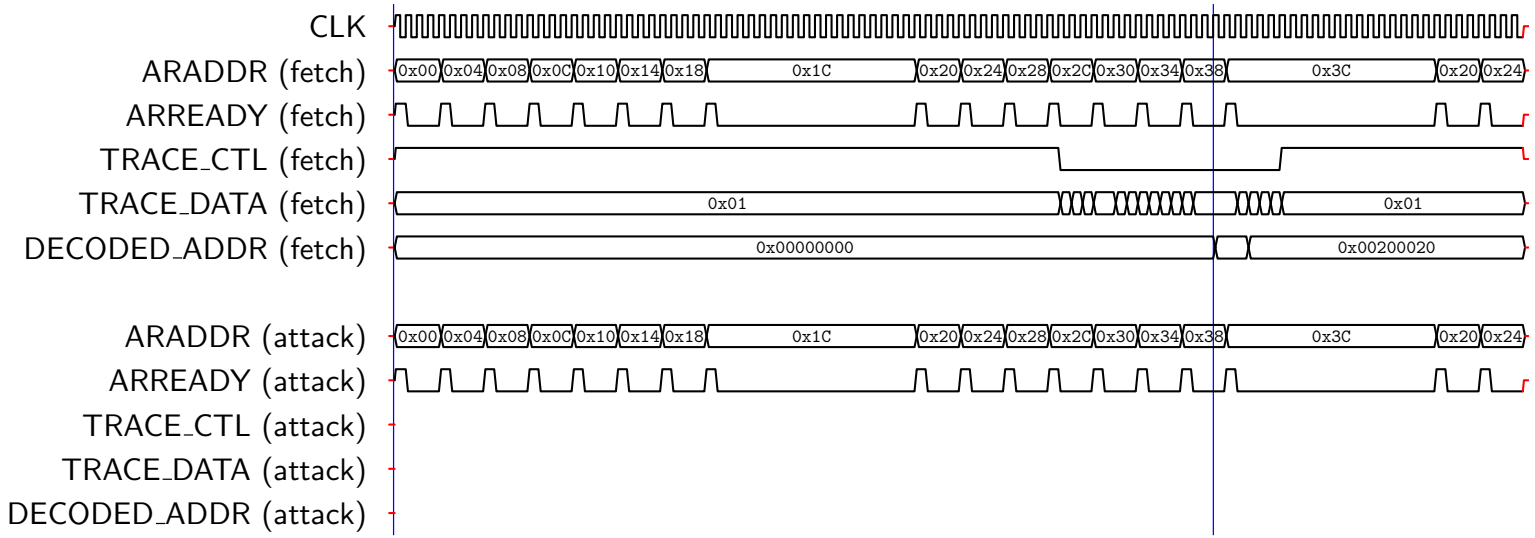

```
→ mov r0, #0x00200000                    // virtual address
  ldm r0!, {r1,r2,r3,r4,r5,r6,r7,r8}  // r0 = r0 + 8 × 4 (!)
  ldm r0 , {r1,r2,r3,r4,r5,r6,r7,r8}  // r0 = r0 + 0
  // ...
```

Audit (exemple)



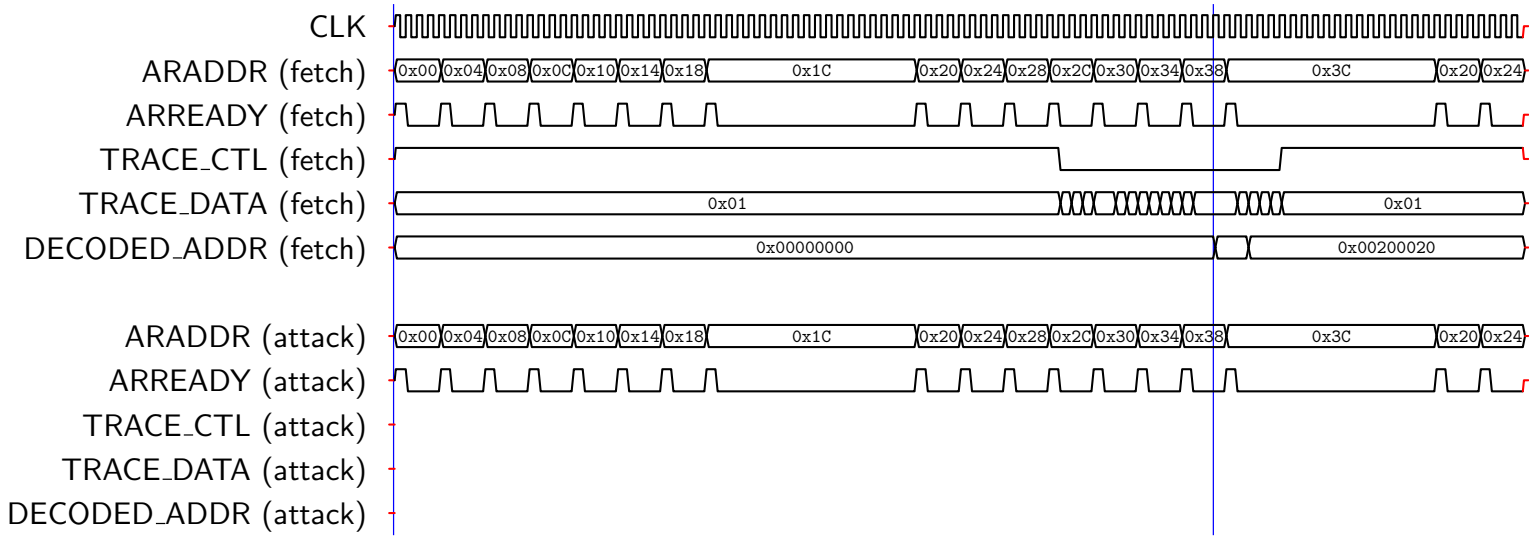

```
mov r0, #0x40000000 // new virtual address (AXI slave)
ldm r0!, {r1,r2,r3,r4,r5,r6,r7,r8} // r0 = r0 + 8 × 4 (!)
ldm r0 , {r1,r2,r3,r4,r5,r6,r7,r8} // r0 = r0 + 0
// ...
```

Audit (exemple)



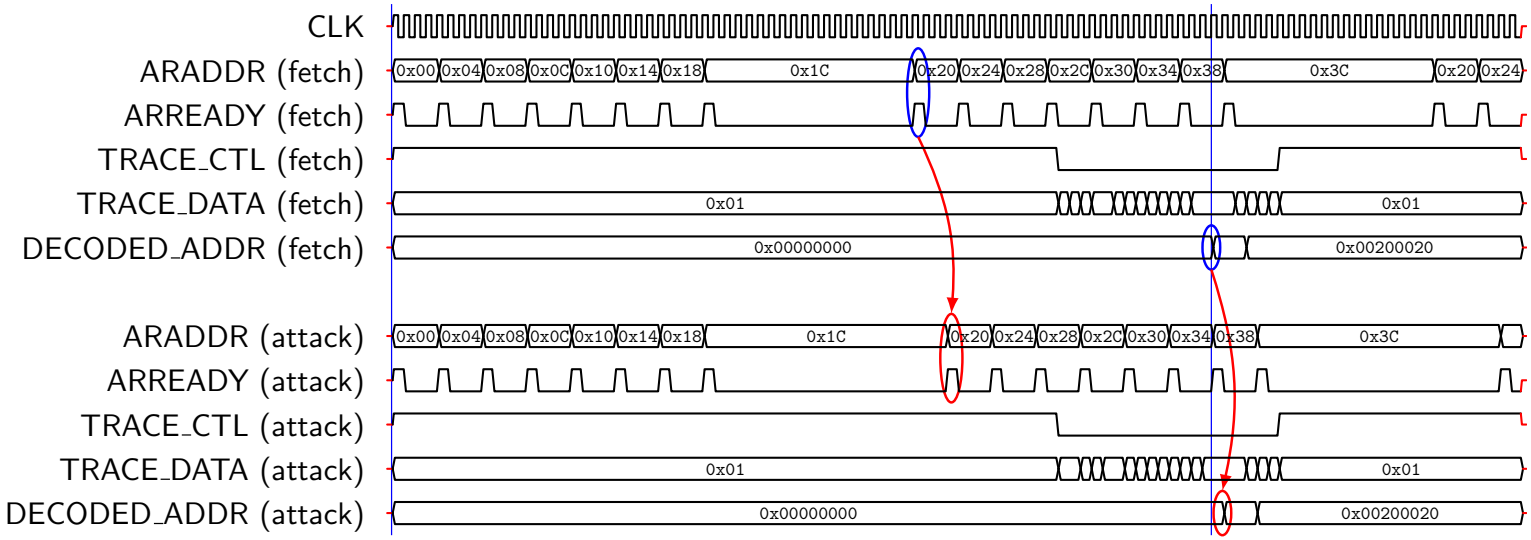

```
mov r0, #0x40000000 // new virtual address (AXI slave)
ldm r0!, {r1,r2,r3,r4,r5,r6,r7,r8} // r0 = r0 + 8 × 4 (!)
→
ldm r0 , {r1,r2,r3,r4,r5,r6,r7,r8} // r0 = r0 + 0
// ...
```

Audit (exemple)



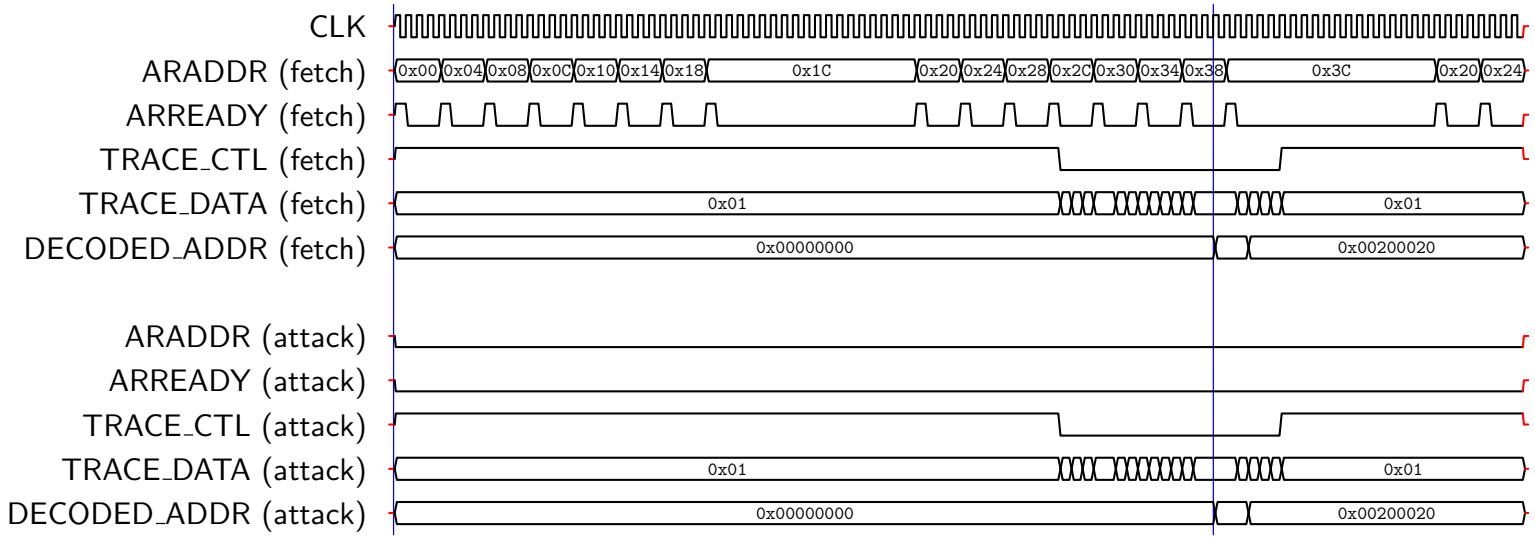

```
mov r0, #0x40000000 // new virtual address (AXI slave)
ldm r0!, {r1, r2, r3, r4, r5, r6, r7, r8}

add r9, pc, #16      // pc = pc + 4; r9 = pc + 4 + 16;
mov pc, r9           // indirect branch
nop
nop
nop
nop
// <- destination (virtual address = 0x00200020)

ldm r0 , {r1,r2,r3,r4,r5,r6,r7,r8} // r0 = r0 + 0
// ...
```

Audit (exemple)



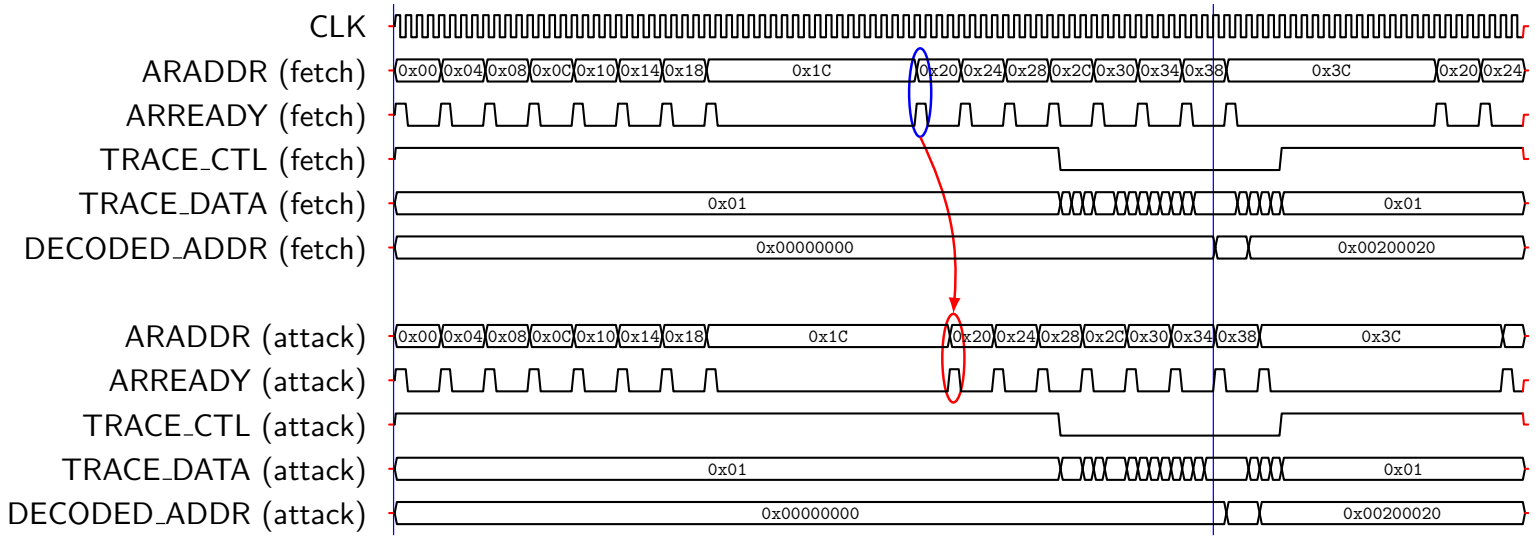

```
mov r0, #0x40000000 // new virtual address (AXI slave)
ldm r0!, {r1, r2, r3, r4, r5, r6, r7, r8}

add r9, pc, #16       // pc = pc + 4; r9 = pc + 4 + 16;
mov pc, r9            // indirect branch
nop
nop
nop
nop
// <- destination (virtual address = 0x00200020)

ldm r0 , {r1,r2,r3,r4,r5,r6,r7,r8} // r0 = r0 + 0
// ...
```

Audit (exemple)



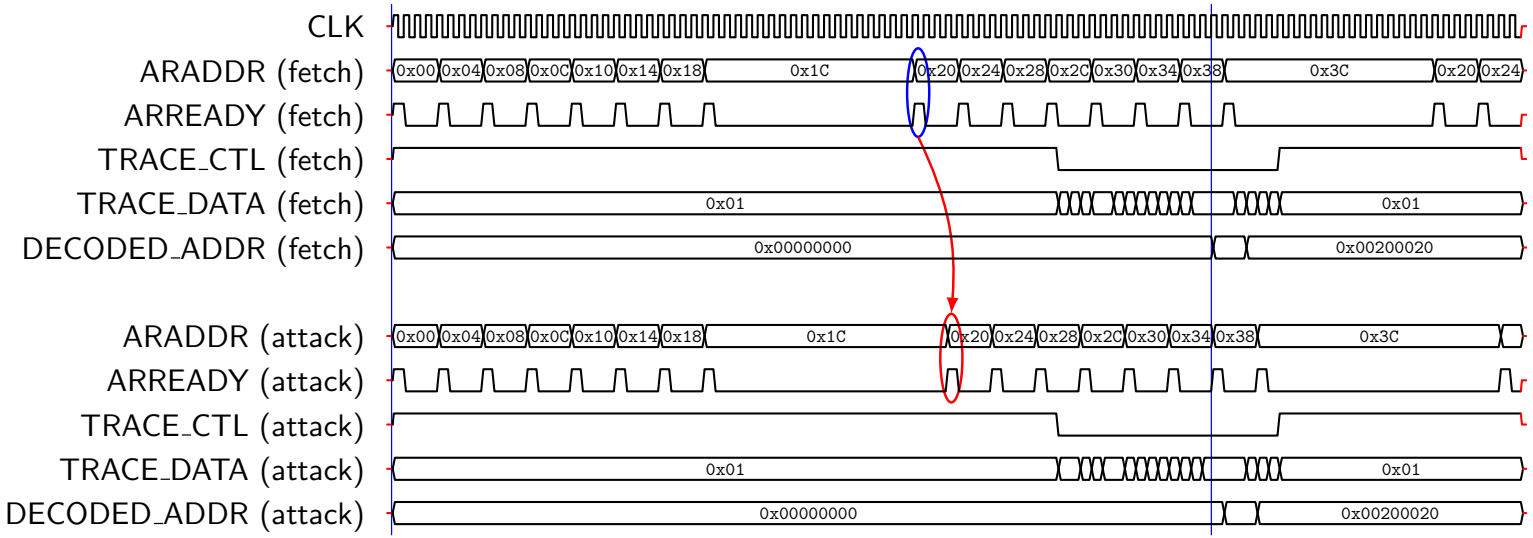

```
  mov r0, #0x40000000 // new virtual address (AXI slave)
  ldm r0!, {r1, r2, r3, r4, r5, r6, r7, r8}
→ // delay here
  add r9, pc, #16       // pc = pc + 4; r9 = pc + 4 + 16;
  mov pc, r9            // indirect branch
  nop
  nop
  nop
  nop
  // <- destination (virtual address = 0x00200020)

  ldm r0 , {r1,r2,r3,r4,r5,r6,r7,r8} // r0 = r0 + 0
  // ...
```

Audit (exemple)



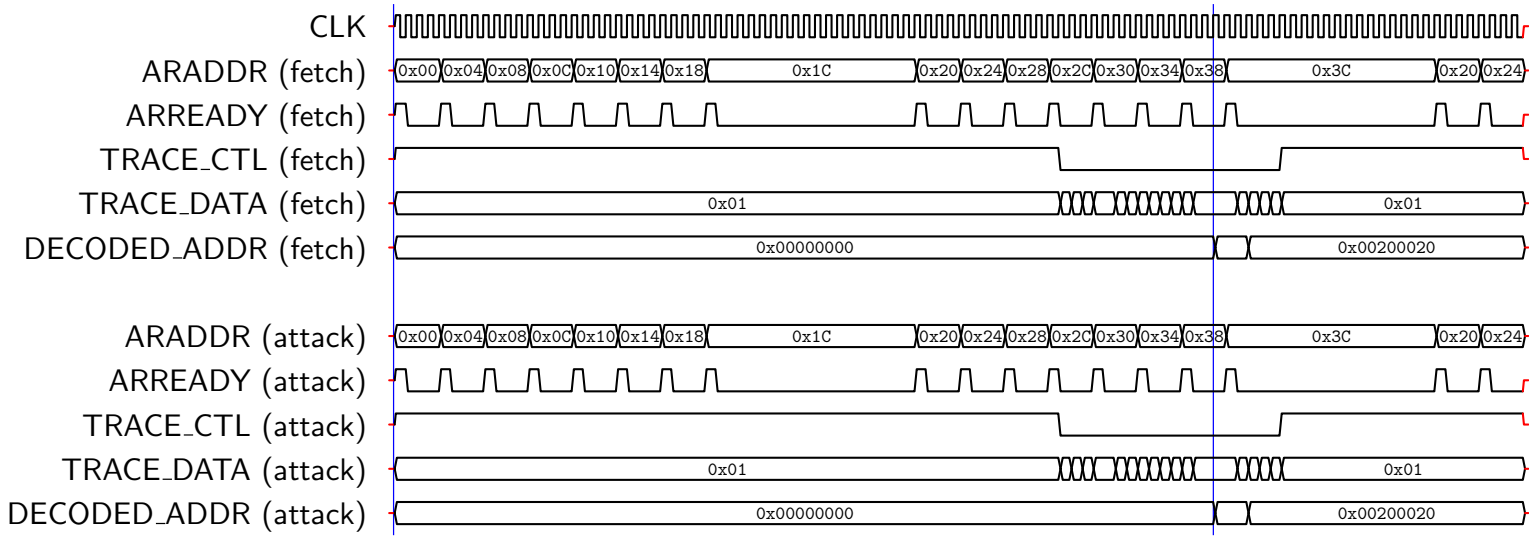

```
mov r0, #0x40000000 // new virtual address (AXI slave)
ldm r0!, {r1, r2, r3, r4, r5, r6, r7, r8}

→ add r9, pc, #16        // pc = pc + 4; r9 = pc + 4 + 16;
mov pc, r9               // indirect branch
nop
nop
nop
nop
// <- destination (virtual address = 0x00200020)

ldm r0 , {r1, r2, r3, r4, r5, r6, r7, r8}
// ...
```

Audit (exemple)




```
// indirect branch destinations in r9, r10, r11 et r12:
movw r9, #0x0020
movt r9, #0x0020      // r9  = 0x00200020
add r10,  r9, #0x20   // r10 = 0x00200040
add r11, r10, #0x20   // r11 = 0x00200060
add r12, r11, #0x20   // r12 = 0x00200080
```

Audit (exemple)




```
  mov r0, #0x40000000 // new virtual address (AXI slave)
  ldm r0!, {r1, r2, r3, r4, r5, r6, r7, r8}

→ mov pc, r9              // indirect branch
→ nop
  nop
  nop
  nop
  nop
  // <- destination (virtual address = 0x00200020)

  ldm r0 , {r1, r2, r3, r4, r5, r6, r7, r8}
  // ...
```

Impact (exemple)

Impact (exemple)

Architecture

- 13 registres généraux
- 2 registres spéciaux (SP, LR) + 1 (PC)

Impact (exemple)

Architecture

- 13 registres généraux
- 2 registres spéciaux (SP, LR) + 1 (PC)

Pour jouer l'attaque

- 9 registres : reproduire les signaux sur le bus AXI
- 6 registres : destinations des branchements indirects

Conclusion

Conclusion

Attestation à distance

- détection d'intrusion / racine de confiance dynamique

Conclusion

Attestation à distance

- détection d'intrusion / racine de confiance dynamique
- racine de confiance statique pour microprocesseur

Conclusion

Attestation à distance

- détection d'intrusion / racine de confiance dynamique
- racine de confiance statique pour microprocesseur

Vérification formelle

- extension matérielle : propriétés de sécurité garanties

Conclusion

Attestation à distance

- détection d'intrusion / racine de confiance dynamique
- racine de confiance statique pour microprocesseur

Vérification formelle

- extension matérielle : propriétés de sécurité garanties
- comportement du microprocesseur : hypothèse

Conclusion

Attestation à distance

- détection d'intrusion / racine de confiance dynamique
- racine de confiance statique pour microprocesseur

Vérification formelle

- extension matérielle : propriétés de sécurité garanties
- comportement du microprocesseur : hypothèse
- audit de sécurité $\rightarrow$ hypothèse de confiance

Conclusion

Attestation à distance

- détection d'intrusion / racine de confiance dynamique
- racine de confiance statique pour microprocesseur

Vérification formelle

- extension matérielle : propriétés de sécurité garanties
- comportement du microprocesseur : hypothèse
- audit de sécurité $\rightarrow$ hypothèse de confiance

Sources pour reproduire l'expérience / améliorer

`https://gitlab.irit.fr/these-jonathan-certes-public/ressources/sstic_2022`

Remote attestation of bare-metal microprocessor software : a formally verified security monitor

J. Certes and B. Morgan (2021),

`https://hal.archives-ouvertes.fr/hal-03576711`

Sources pour reproduire l'expérience / améliorer

`https://gitlab.irit.fr/these-jonathan-certes-public/ressources/sstic_2022`