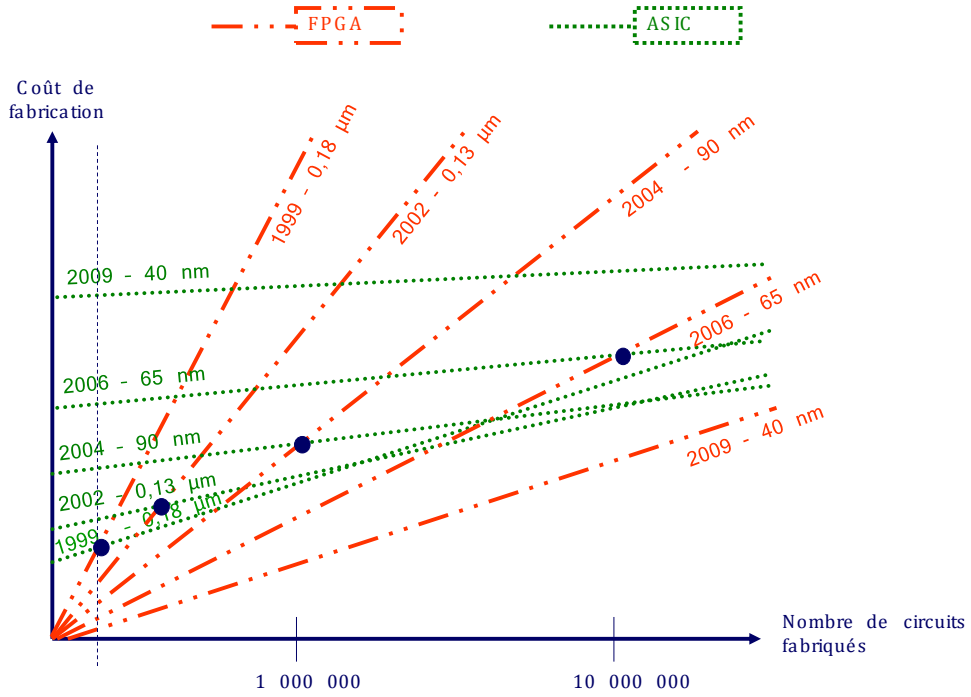
Remote bitstream update protocol preventing replay attacks: A practical implementation

Florian Devic[1,2] **Lionel Torres**[1] **Benoît Badrignans**[2]

[1]LIRMM UMR - CNRS 5506, University of Montpellier 2, Montpellier, France
[2]SAS NETHEOS, Montpellier, FRANCE

Contexte

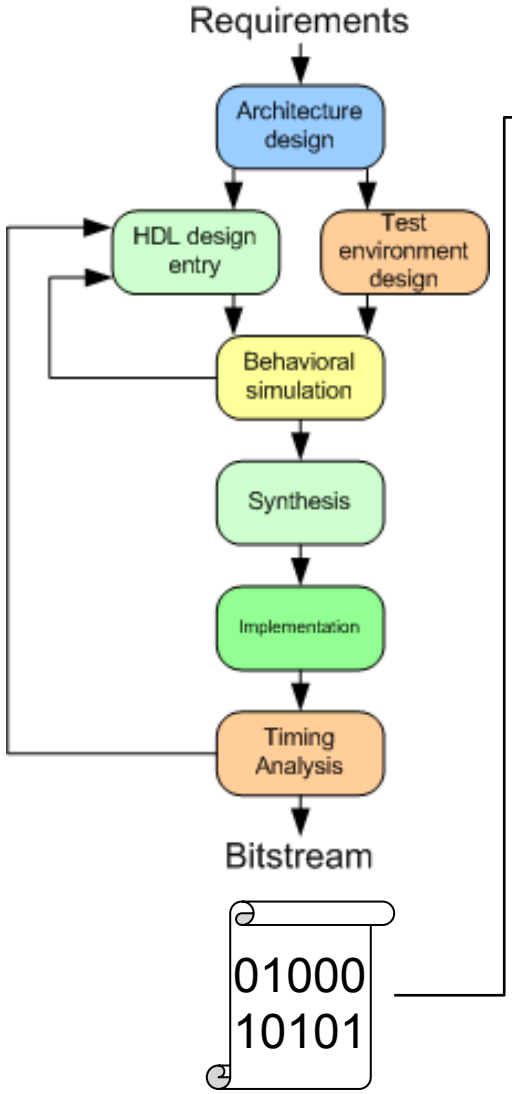
- Les FPGA sont aujourd'hui de réelles alternatives aux ASIC (capacité, prix et performances en adéquation avec les besoins)
- **Sécurité flexible**[1] - besoins du matériel reconfigurable pour
  - Suivre l'évolution de normes de sécurité
  - Suivre l'évolution d'algorithmes cryptographiques
  - Contrer les nouvelles attaques
  - S'adapter aux différentes protocoles cryptographiques
  - Permettre l'interopérabilité
  - Permettre les mis à jour du matériel (erreurs de conception)

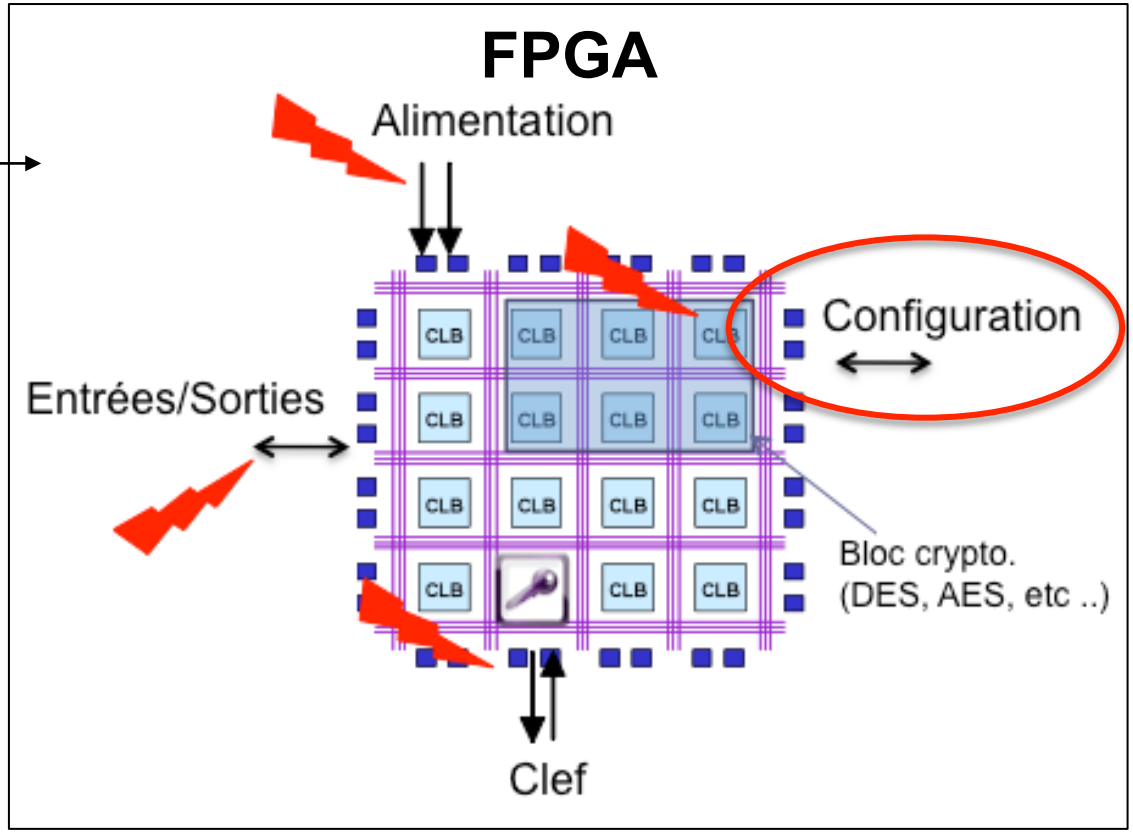


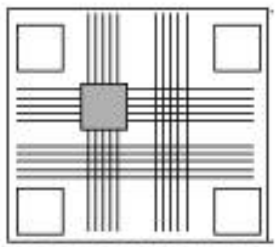
S. Tredennick, The Rise of Reconfigurable Systems, ERSA

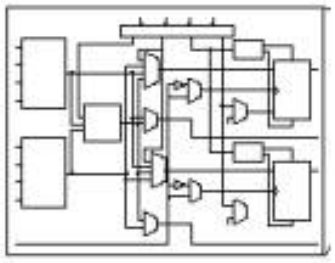
[1] P. Davies, Flexible Security, White paper

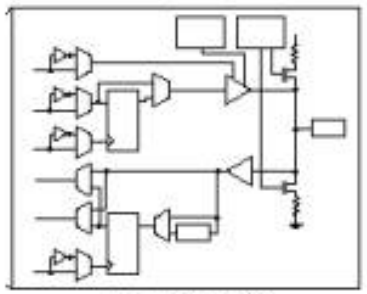
Contexte



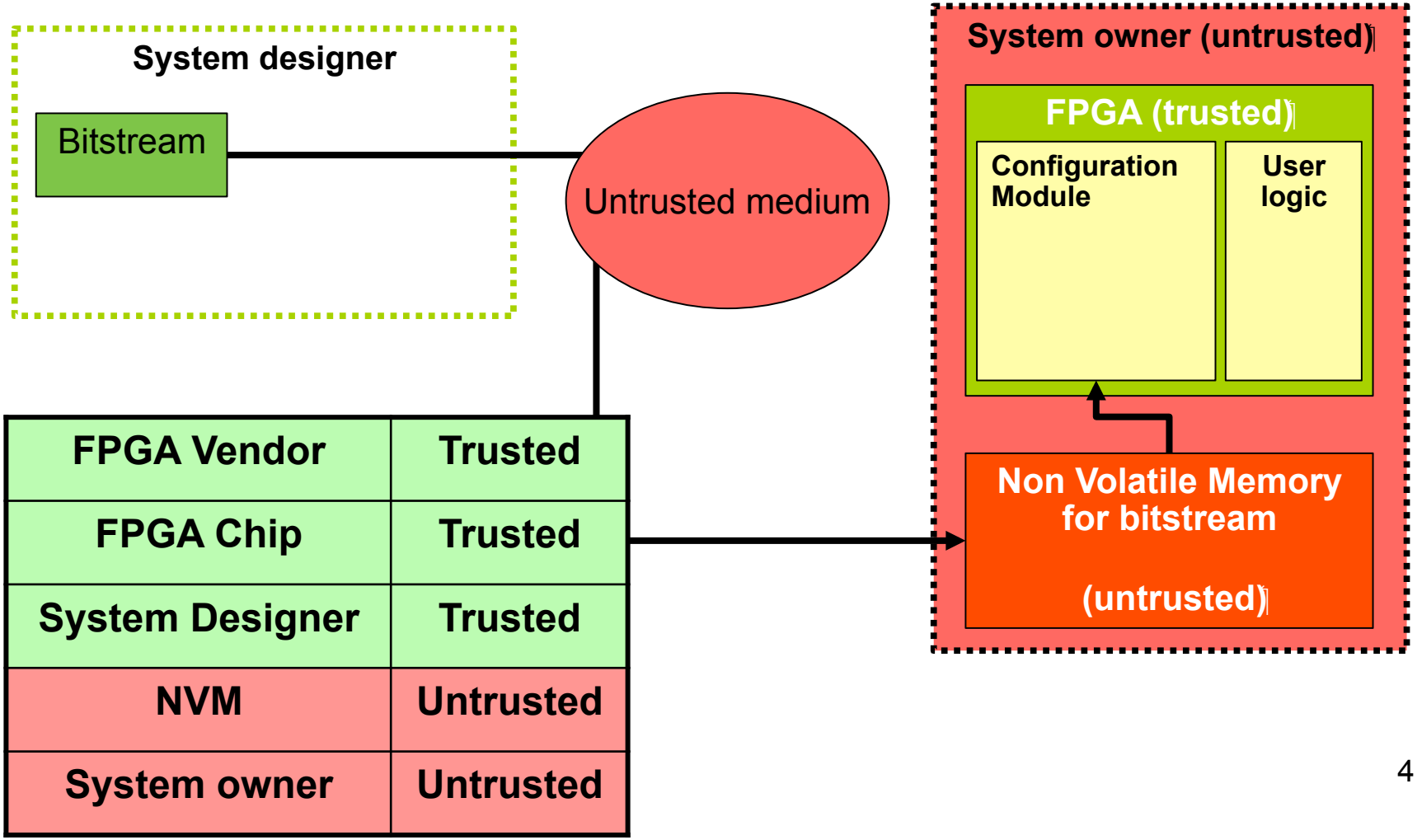



PROGRAMMABLE INTERCONNECT


LOGIC BLOCKS


I/O BLOCKS

Contexte


System designer
Bitstream
Untrusted medium
System owner (untrusted)
FPGA (trusted)
Configuration Module
User logic
Non Volatile Memory for bitstream
(untrusted)
FPGA Vendor
Trusted
FPGA Chip
Trusted
System Designer
Trusted
NVM
Untrusted
System owner
Untrusted

Problematic

Bitstream:

- Confidentiality
- Integrity
- Authenticity
- Downgrade

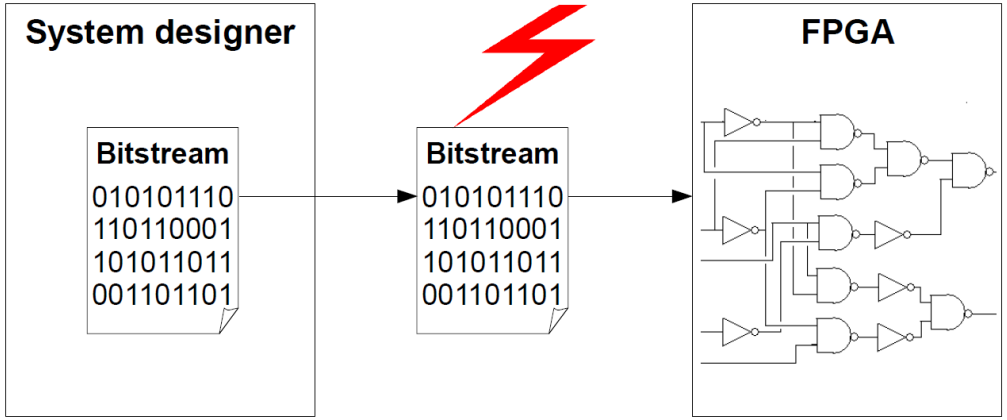
Proposed solutions:

- Most of the FPGA vendors focus only about confidentiality
- None of then prevents downgrades.

Summary

1. Context
2. State of the art
3. Secure update principle
4. Implementation
5. Case Study
6. Conclusion

2. State of the art



Focus on:

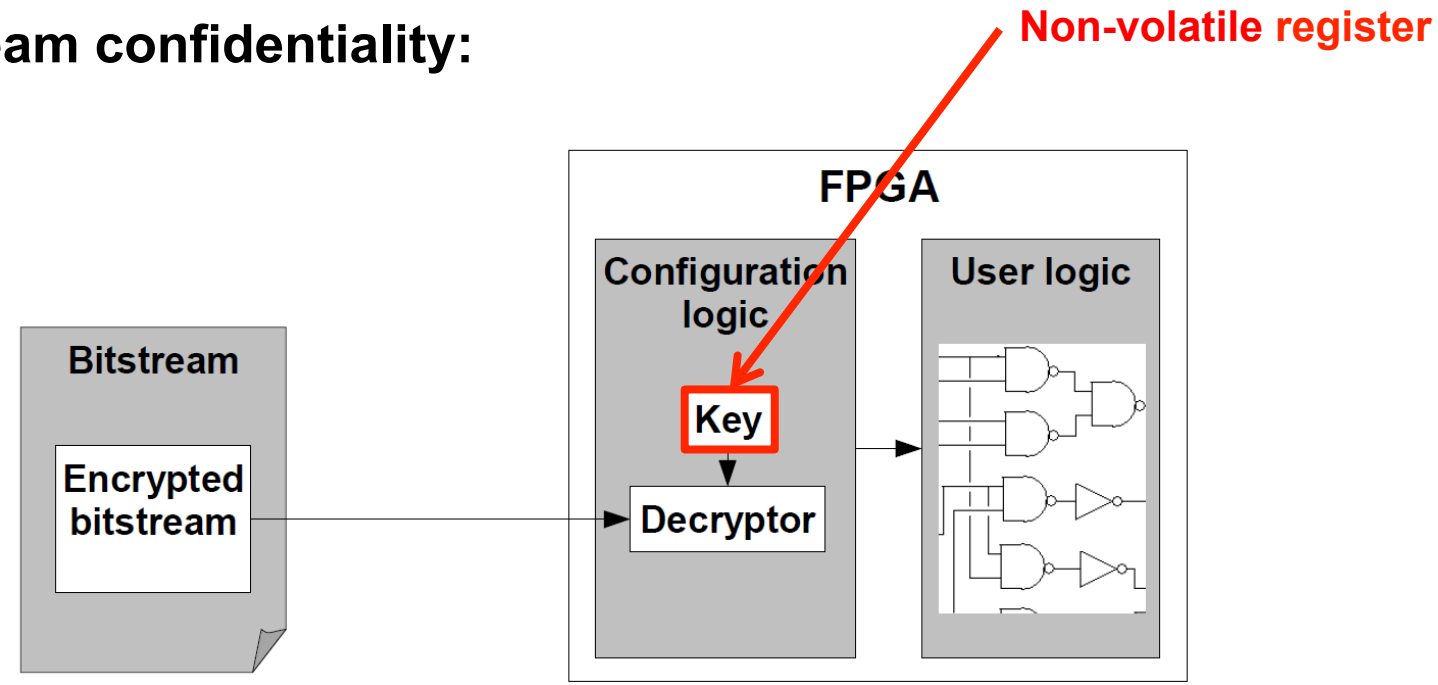
- Cloning & Reverse engineering *(confidentiality)*
- Spoofing & Fault injection *(integrity)*
- Replay attacks *(temporal integrity)*

Not considered:

- Invasive attacks
- Side channel attacks
- Fault attacks

2. State of the art

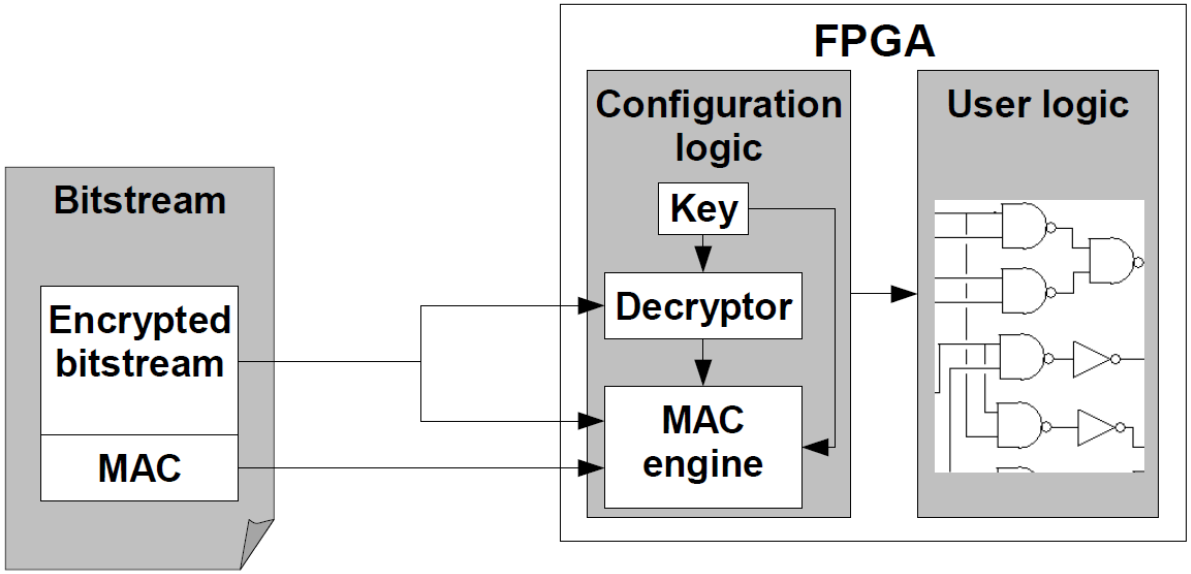
Bitstream confidentiality:



- **Prevent cloning**
- **Prevent reverse engineering**

2. State of the art

Bitstream confidentiality and integrity:

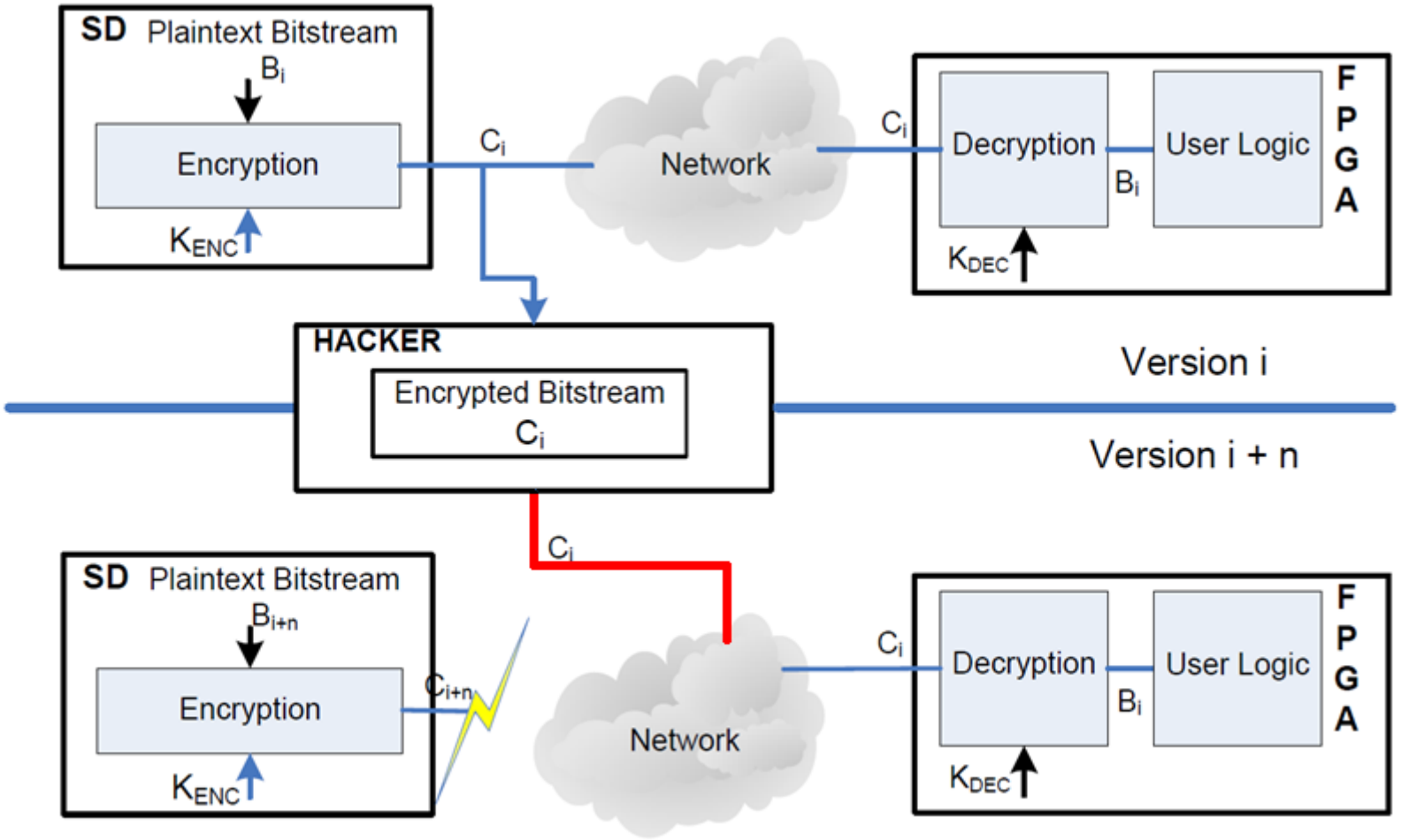


- Prevent cloning
- Prevent reverse engineering
- **Prevent modifications**

2. State of the art

- S. Drimer & al, 2009, http://www.cl.cam.ac.uk/~mgk25/arc2009-remoteupdates.pdf
- B. Badrignans, 2008

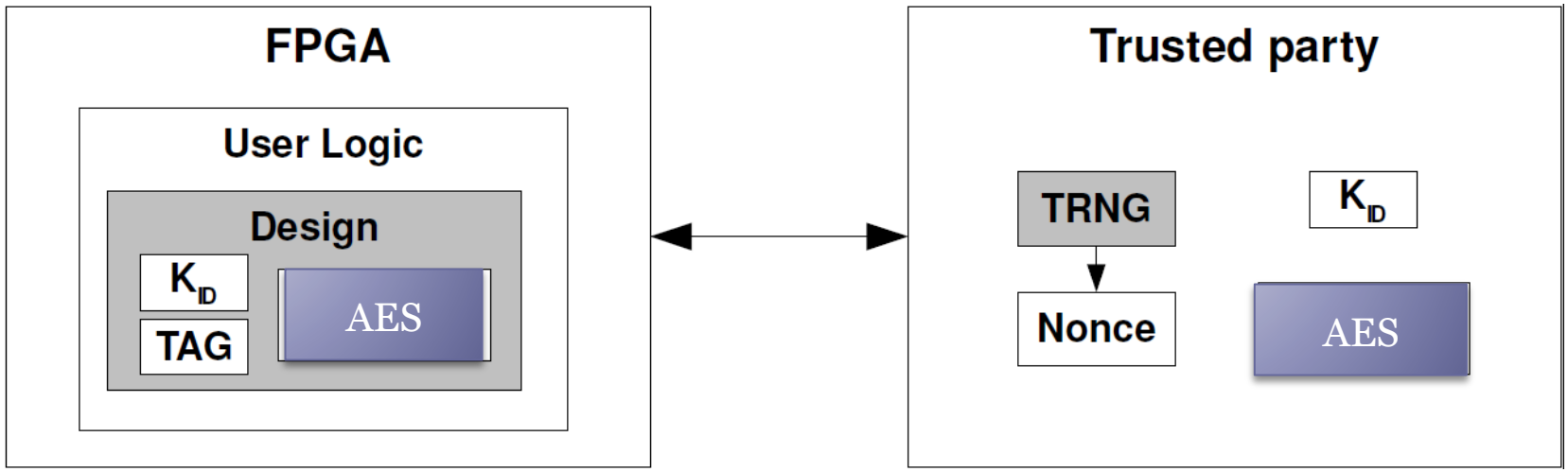
Replay attack:



3. Secure update principle

Solution 1: An external party attests the current bitstream version (polling)

- K_{ID} is a unique key
- TAG is the current bitstream version.

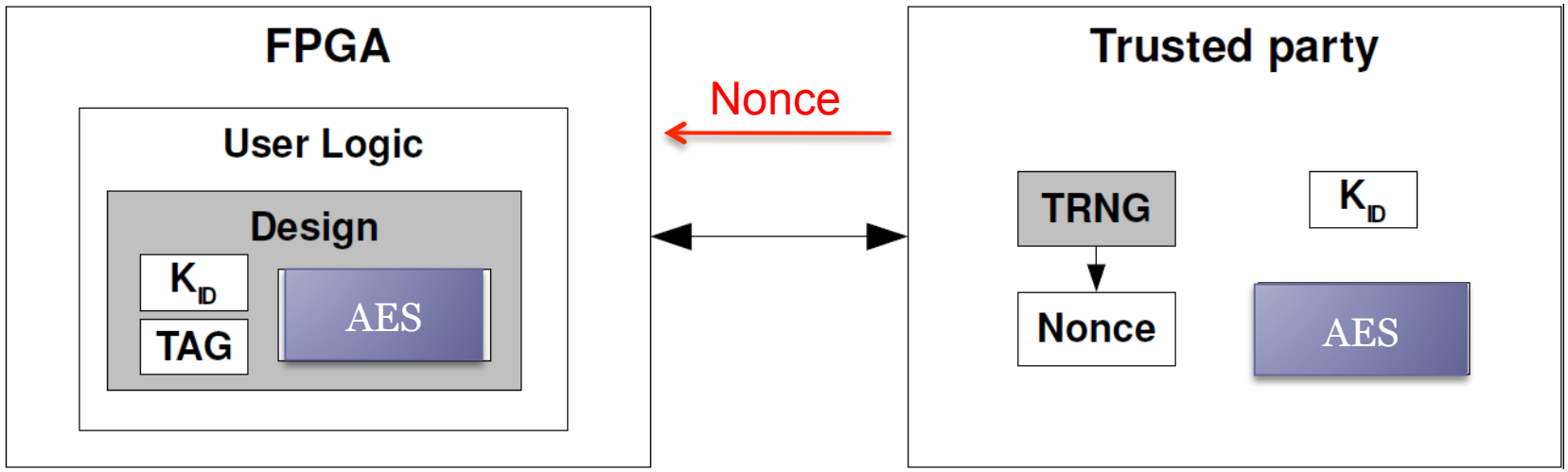


This solution could be applied on any FPGA

3. Secure update principle

Solution 1: An external party attests the current bitstream version

- K_{ID} is a unique key
- TAG is the current bitstream version.

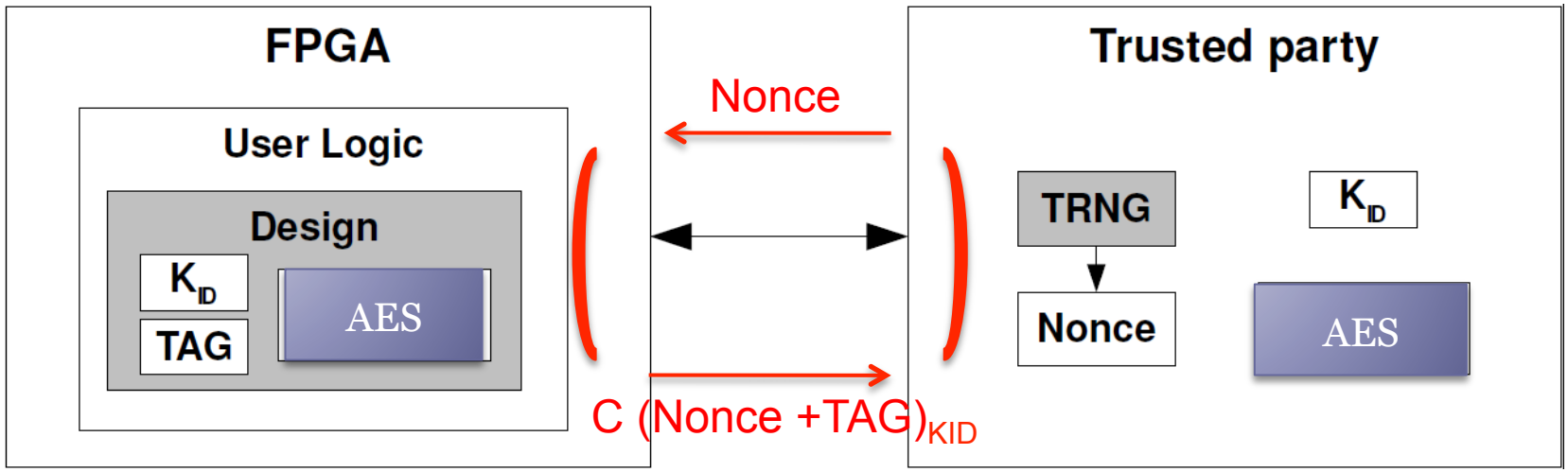


This solution could be applied on any FPGA

3. Secure update principle

Solution 1: An external party attests the current bitstream version

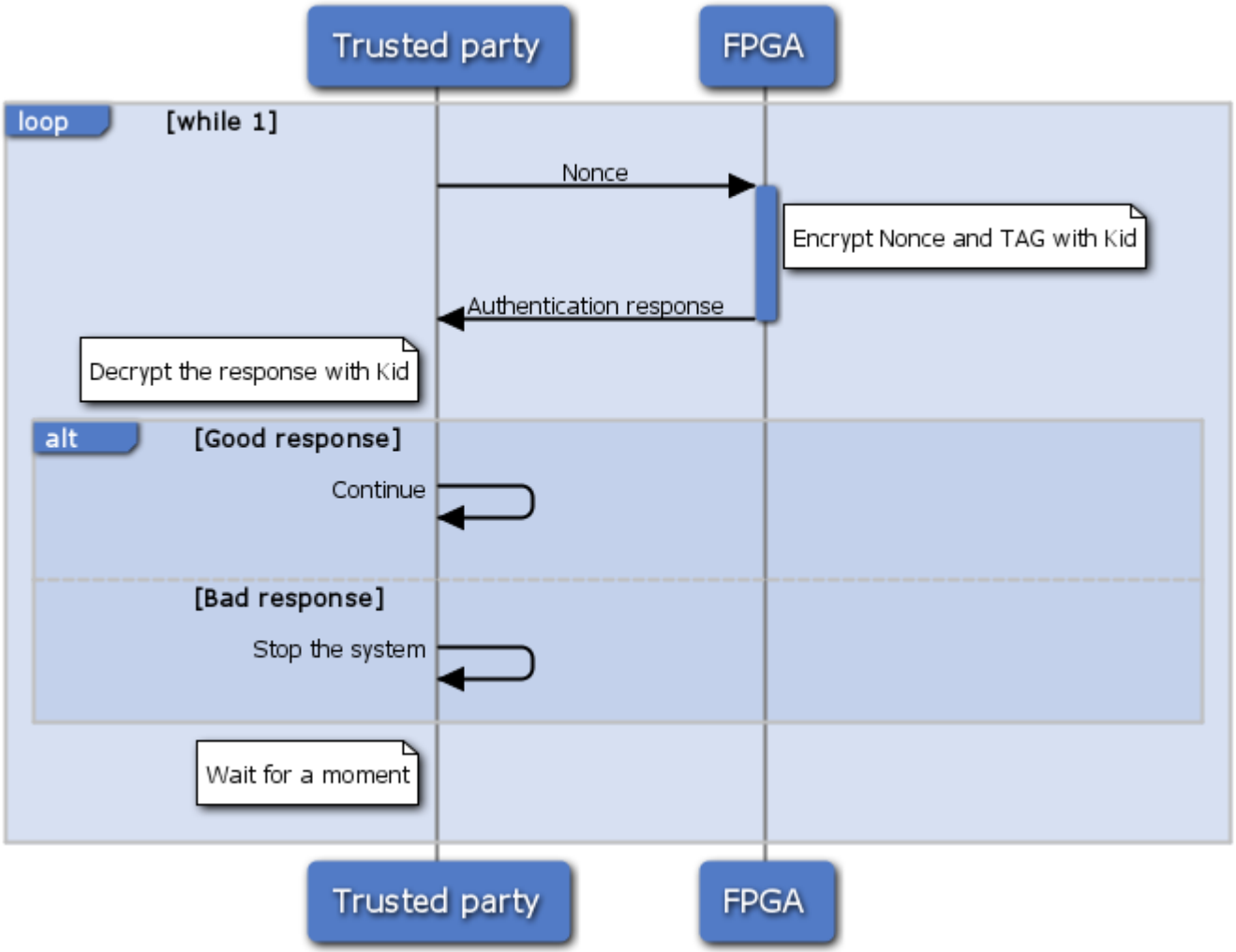
- K_{ID} is a unique key
- TAG is the current bitstream version.



This solution could be applied on any FPGA

3. Secure update principle

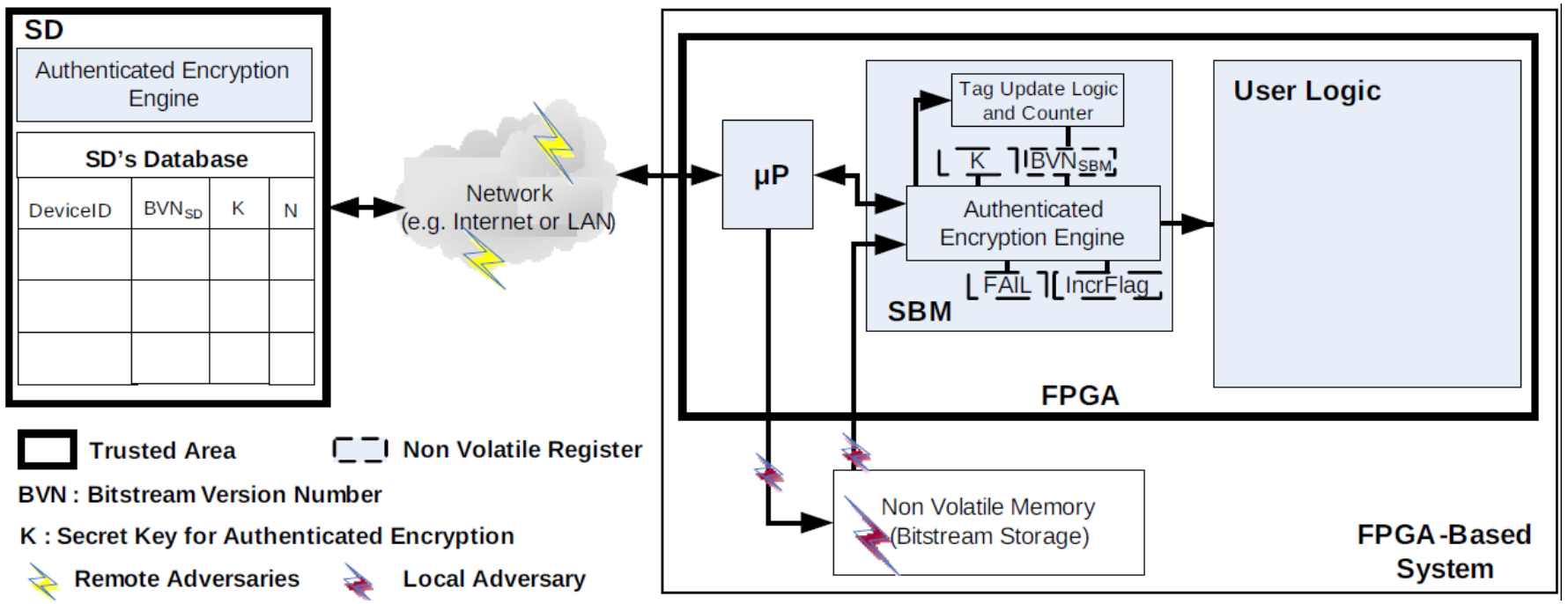
Protocol overview (solution 1):



3. Secure update principle

Solution 2: Lock the FPGA to a dedicated version directly thanks to the config. logic

- Modify the configuration logic (drawback)
- MAC of the bitstream and TAG (for the update)



B.Badrignans, R.Elbaz, L.Torres, "Secure FPGAconfiguration architecture preventing system downgrade", FPL 2008 Conference, September 2008.

3. Secure update principle

Solution 2: Lock the FPGA to a dedicated version directly thanks to the config. logic

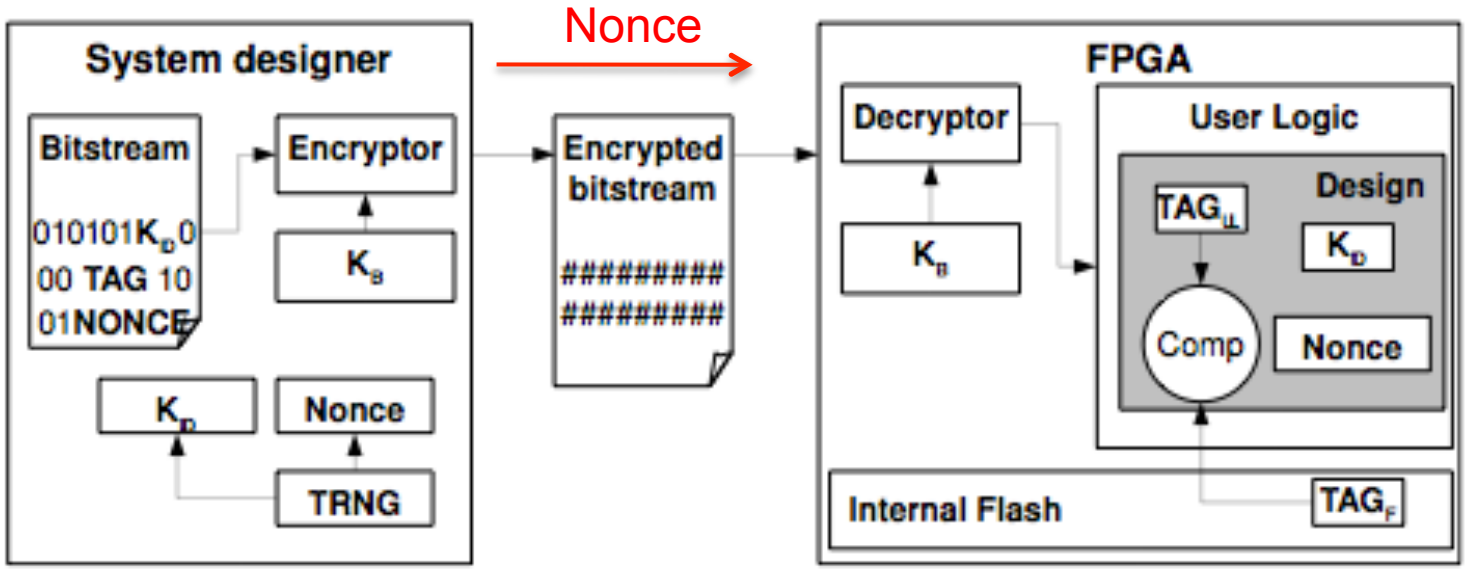
- Modify the configuration logic (drawback)
- MAC of the bitstream and TAG (for the update)

	Area	**Crypto engine Throughput**	**Max. configuration speed**
No security	0	-	3.2Gb/s [1]
Confidentiality (AES-CBC)	~15 kGates [2]	1000 Mb/s [2]	580 Mb/s [1]
Confidentiality and integrity (AES-CCM)	~ 23 kGates [2]	430 Mb/s [2]	430 Mb/s [2]
SUM (With AES-CCM)	~ 24 kGates	430 Mb/s	430 Mb/s

3. Secure update principle

Solution 3: Lock the FPGA to a dedicated version

- K_{ID} is a unique key
- TAG_F and TAG_{UL} are the current bitstream version.
- Nonce included in the bitstream: different for each device

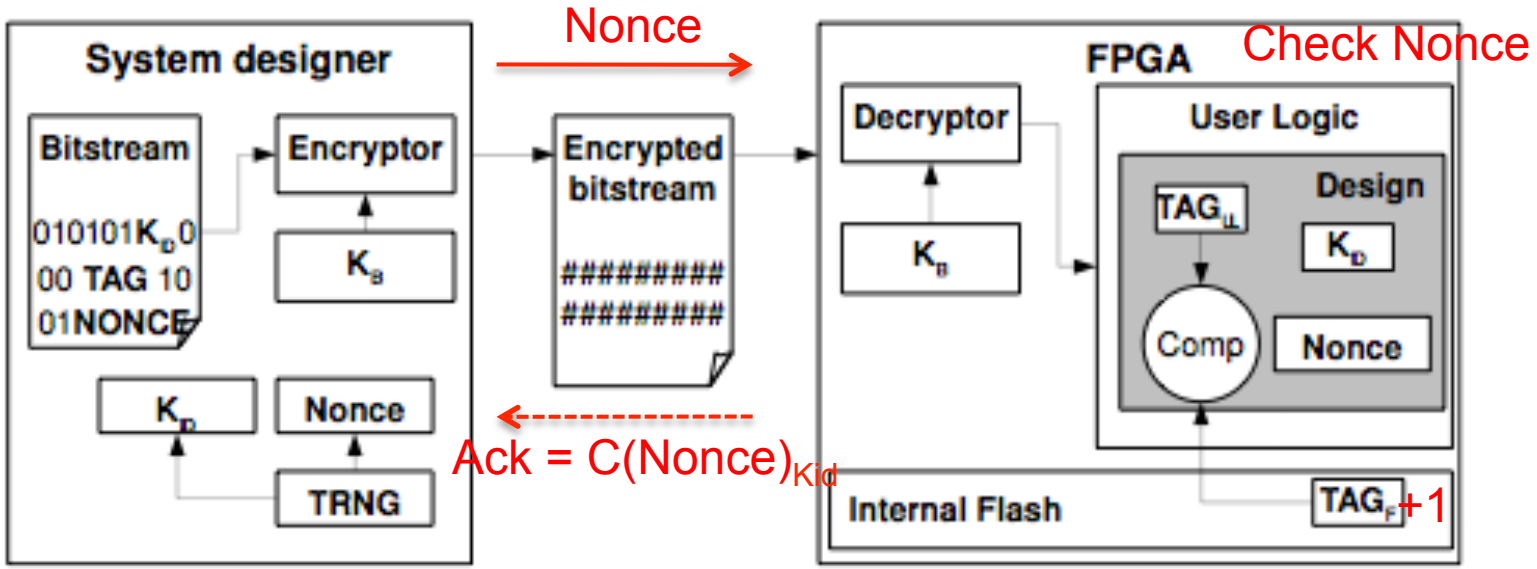


- TAG_{UL} : bitstream version
- Nonce unique for each device
- Initial : TAG_F=TAG_{UL}

3. Secure update principle

Solution 3: Lock the FPGA to a dedicated version

- K_{ID} is a unique key
- TAG_F and TAG_{UL} are the current bitstream version.
- Nonce included in the bitstream: different for each device

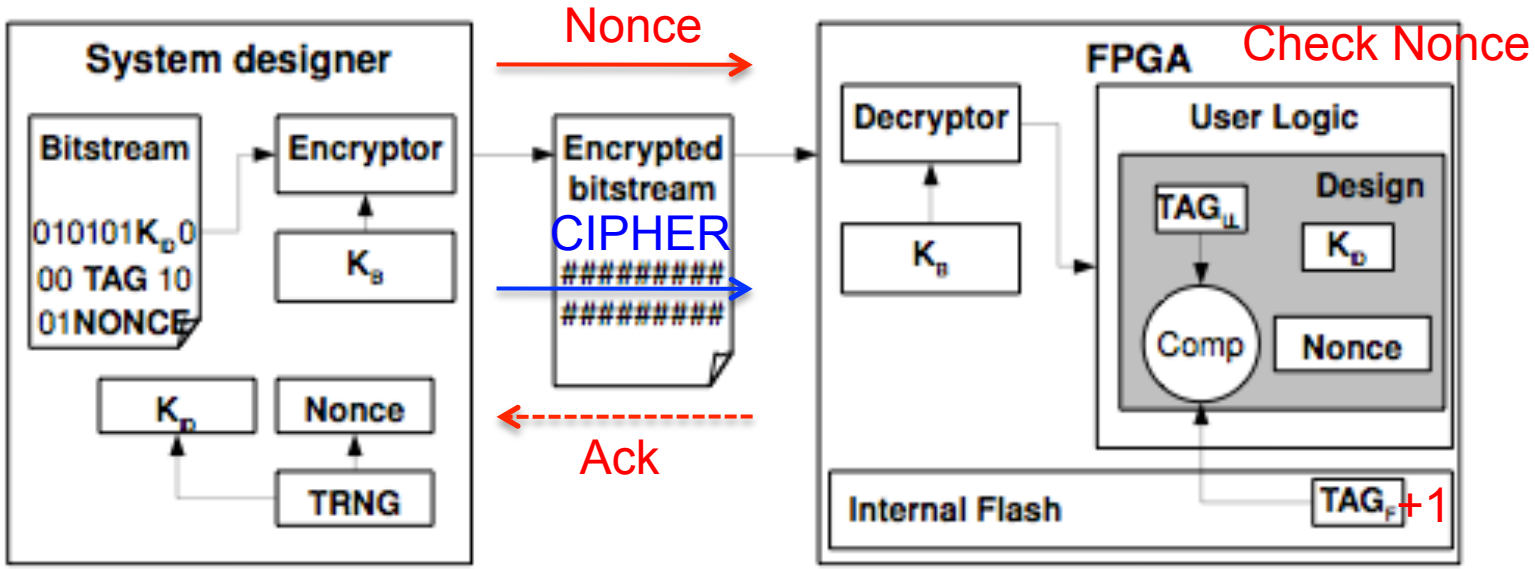


- TAGUL : bitstream version
- Kid use for DoS
- Nonce unique for each device
- Initial : TAG_F=TAG_{UL}

3. Secure update principle

Solution 3: Lock the FPGA to a dedicated version

- K_{ID} is a unique key
- TAG_F and TAG_{UL} are the current bitstream version.
- Nonce included in the bitstream: different for each device

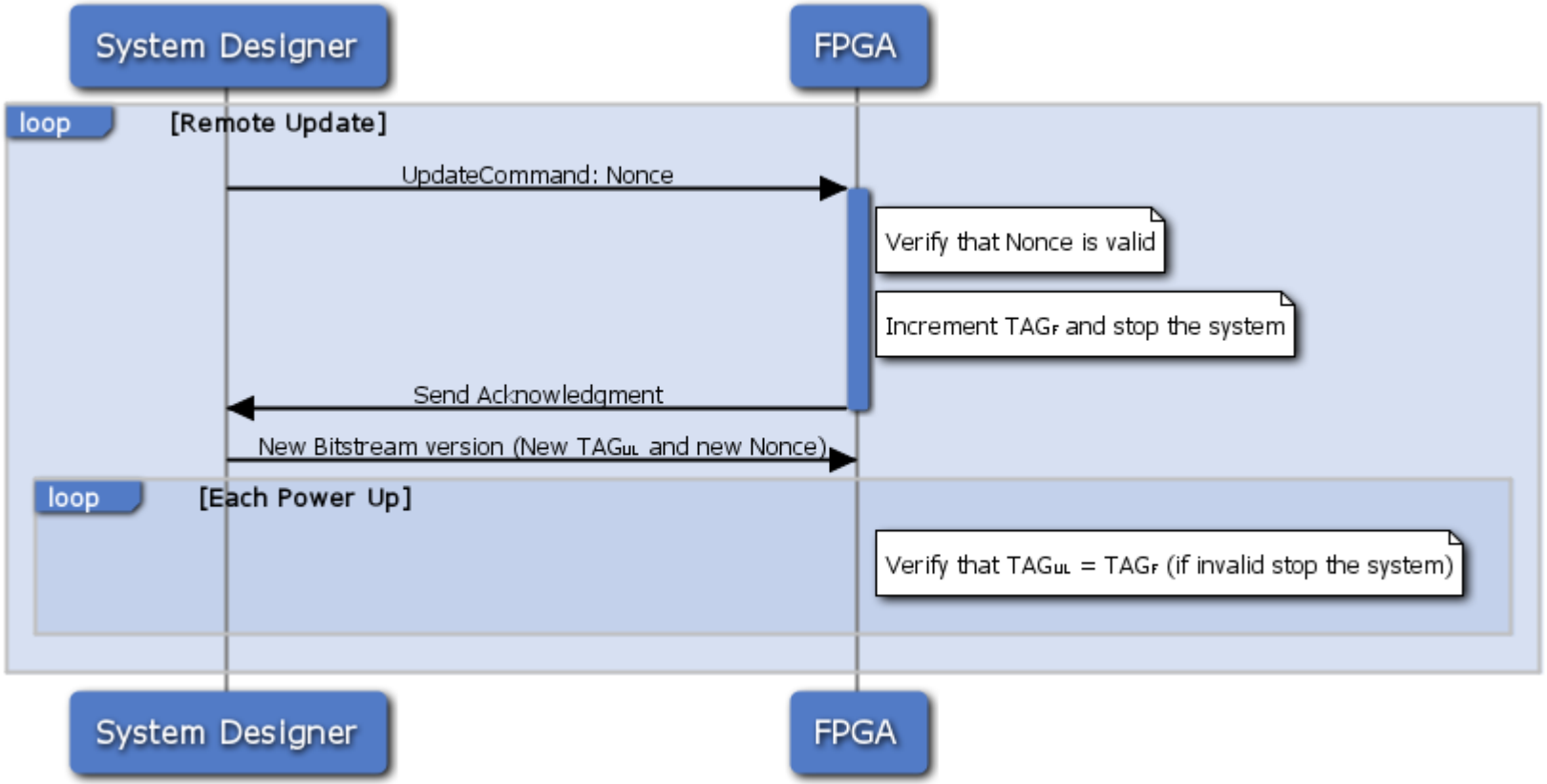


- TAG_{UL} : bitstream version
- Nonce unique for each device
- Initial : TAG_F=TAG_{UL}

CIPHER = C (Nonce+TAG+Bitsream)$_{Kb}$
CHECK TAG_{UL}, TAG_F, Nonce

3. Secure update principle

Protocol overview (solution 3):



3. Secure update principle

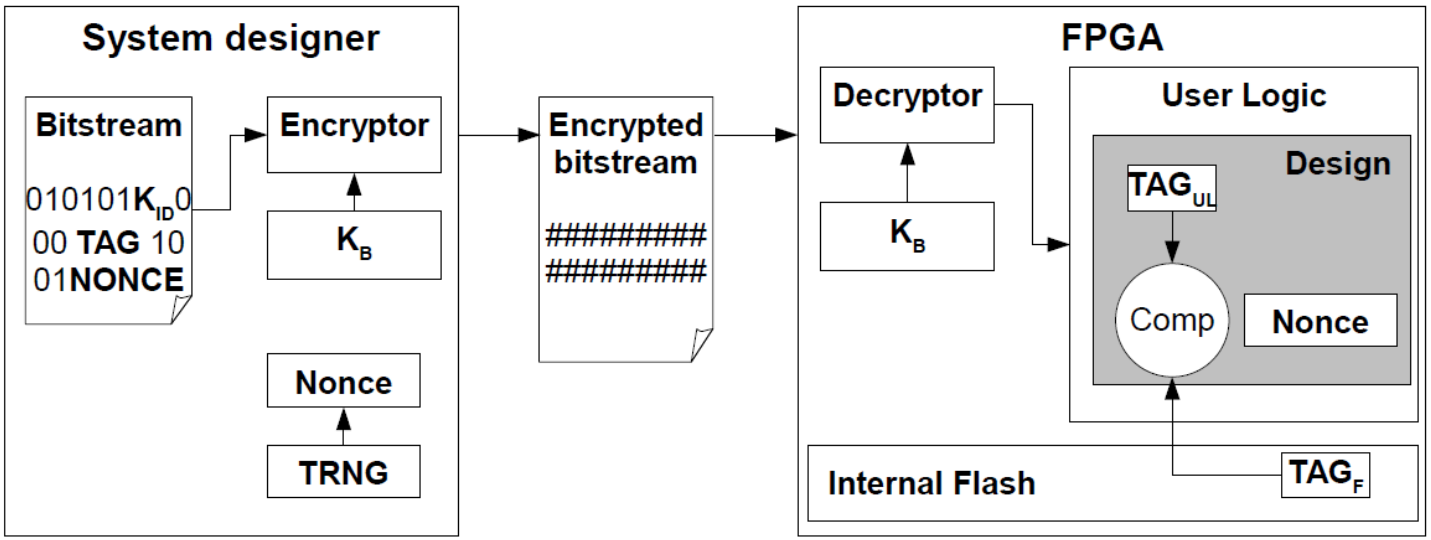
Solution	Modification of the static logic	Development time	Area overhead	Additional cost
1	None	High	High	Regular polling
2	Yes	Low	None	None
3	Low for Flash based FPGA	Medium	Low / Medium	Specific bitstream

Red: makes industrial implementation difficult

3. Secure update principle

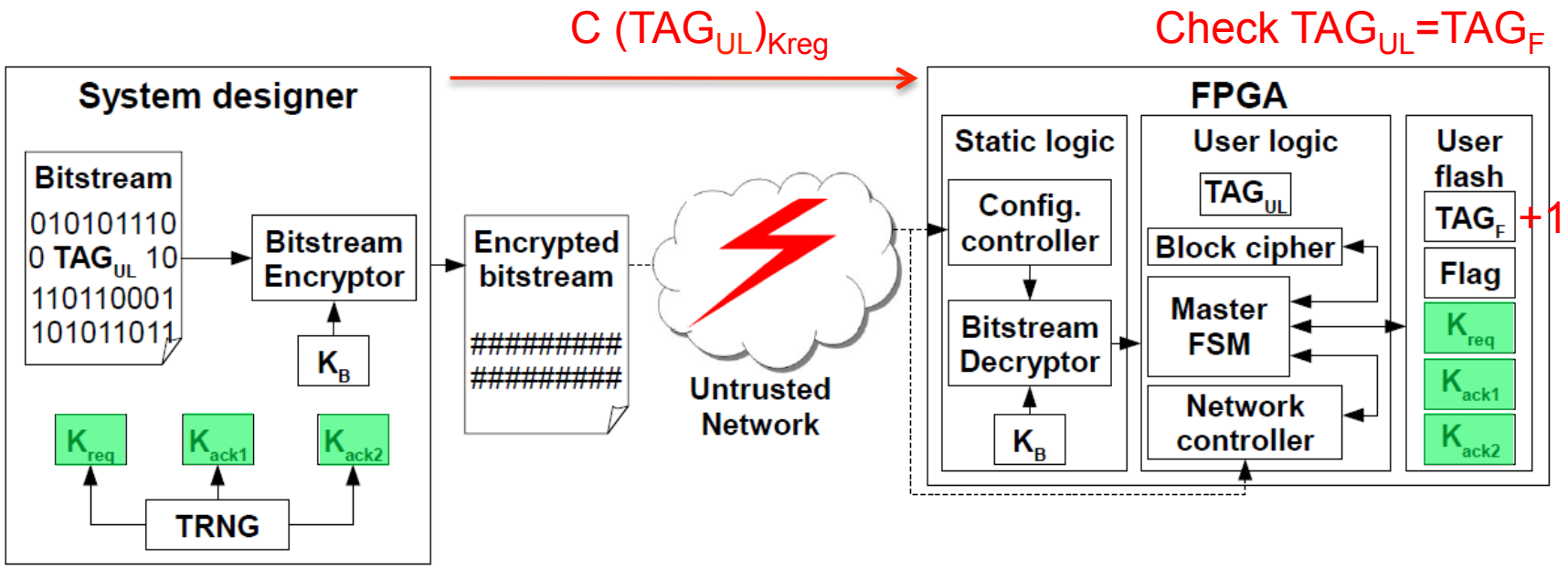
Goal: Solution 3 approach ++, avoid the Nonce and to lock a dedicated version

- Provide confidentiality, integrity, bitstream freshness
- Low cost FPGA
- Easy to use (non-specific bitstream)
- Implementation



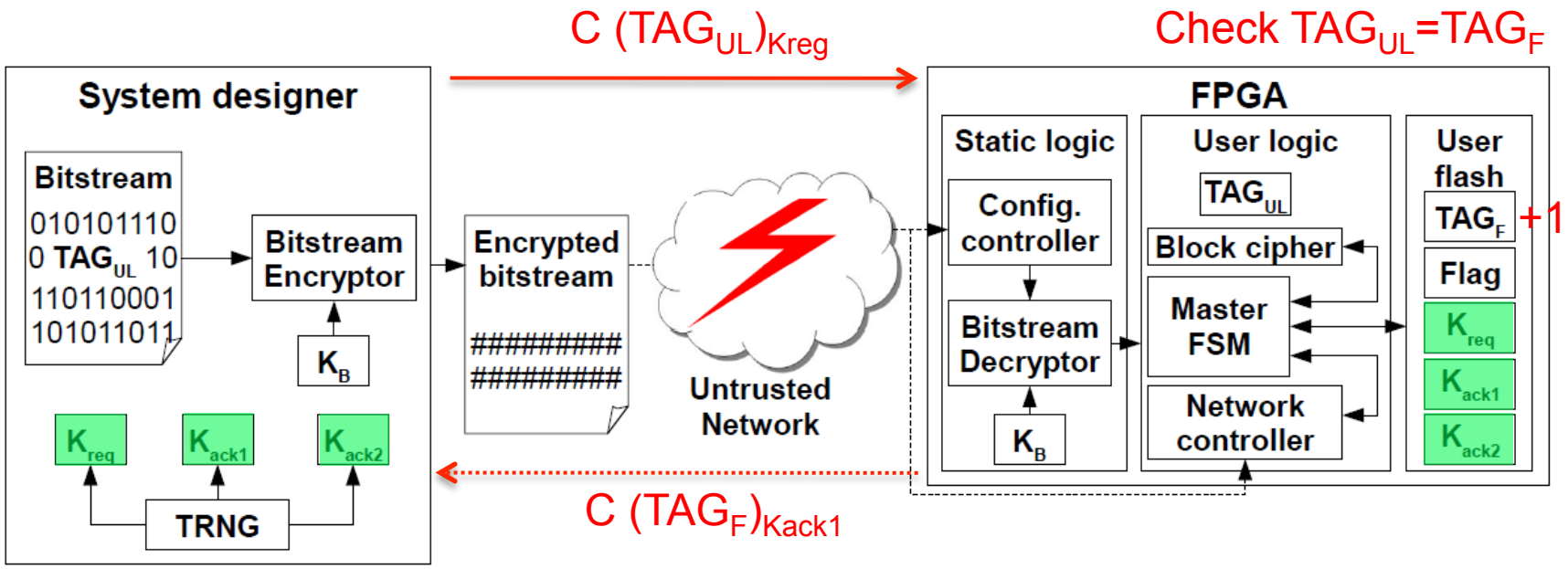
Considering non-volatile FPGA with on chip user non-volatile memory

3. Secure update principle



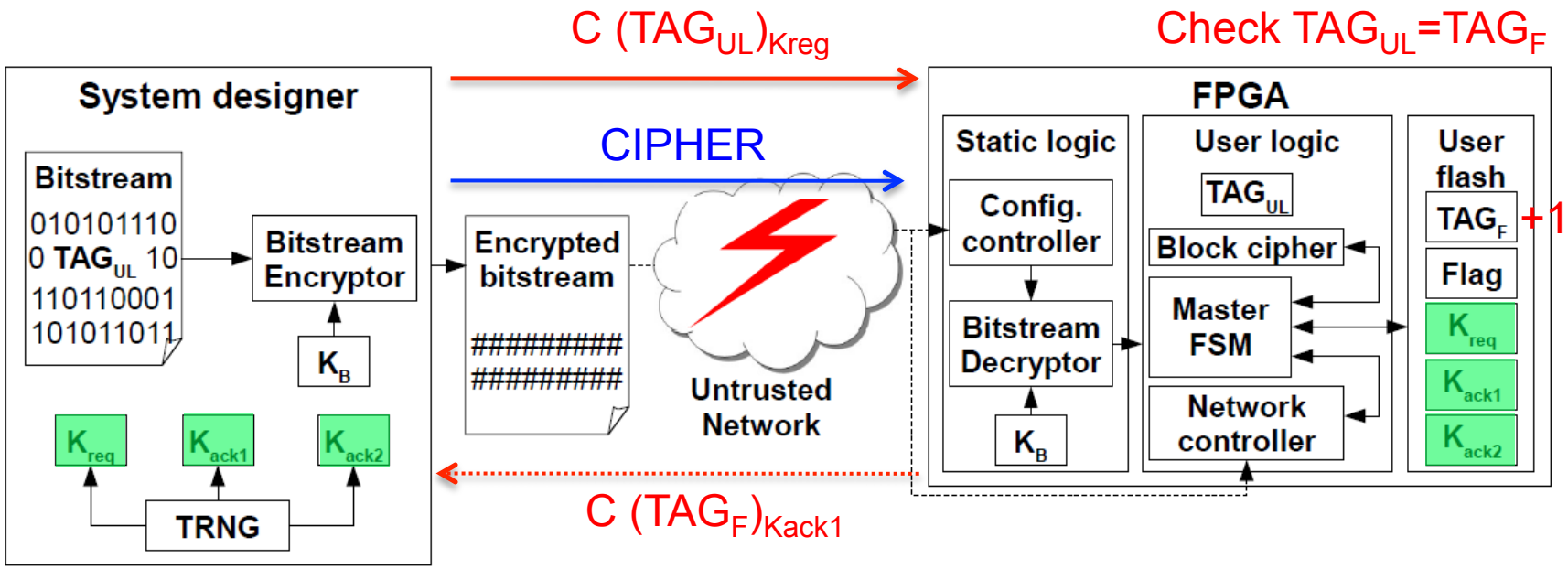
- Kreq: for the Update command
- Kack1: for the Update command acknowledgement
- Kack2: for the new bitstream version reception and startup acknowledgement

3. Secure update principle



- Kreq: for the Update command
- Kack1: for the Update command acknowledgement
- Kack2: for the new bitstream version reception and startup acknowledgement

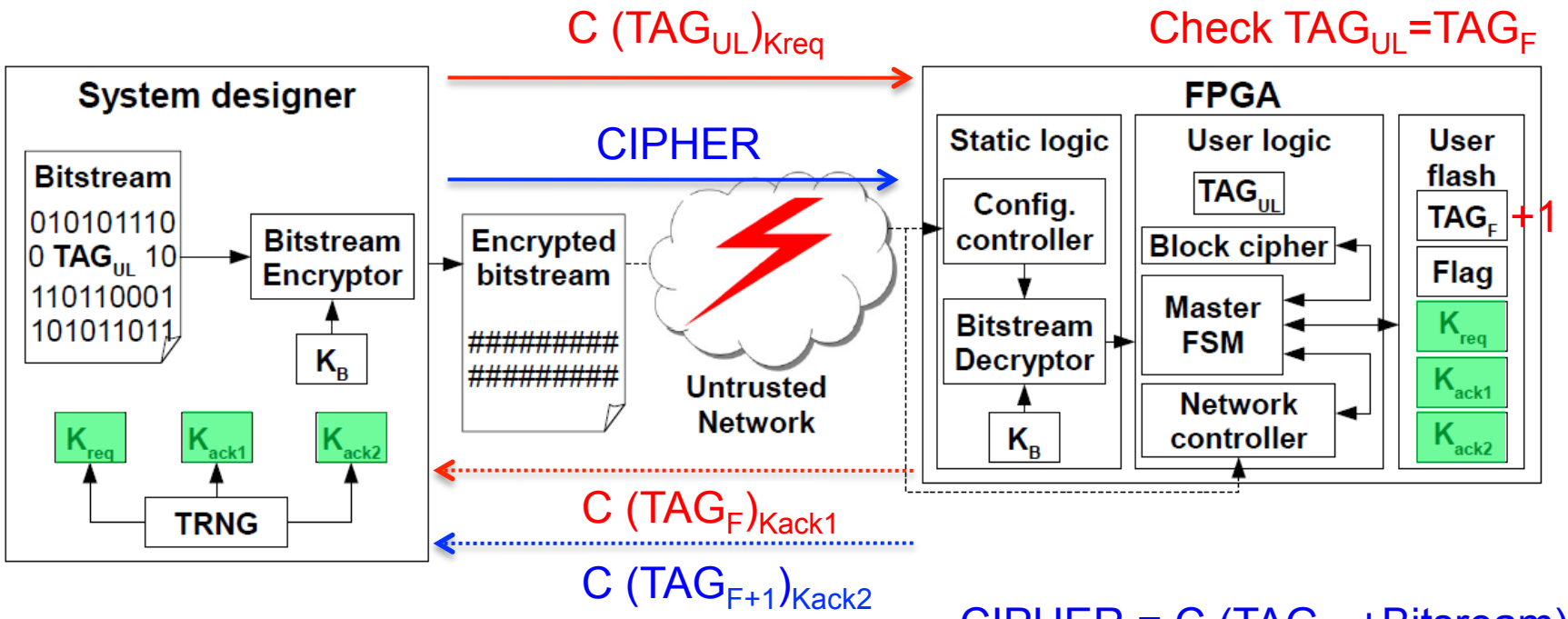
3. Secure update principle



CIPHER = C (TAG$_{UL}$+Bitsream)$_{Kb}$
CHECK TAG$_{UL}$, TAG$_F$, Nonce

- Kreq: for the Update command
- Kack1: for the Update command acknowledgement
- Kack2: for the new bitstream version reception and startup acknowledgement

3. Secure update principle

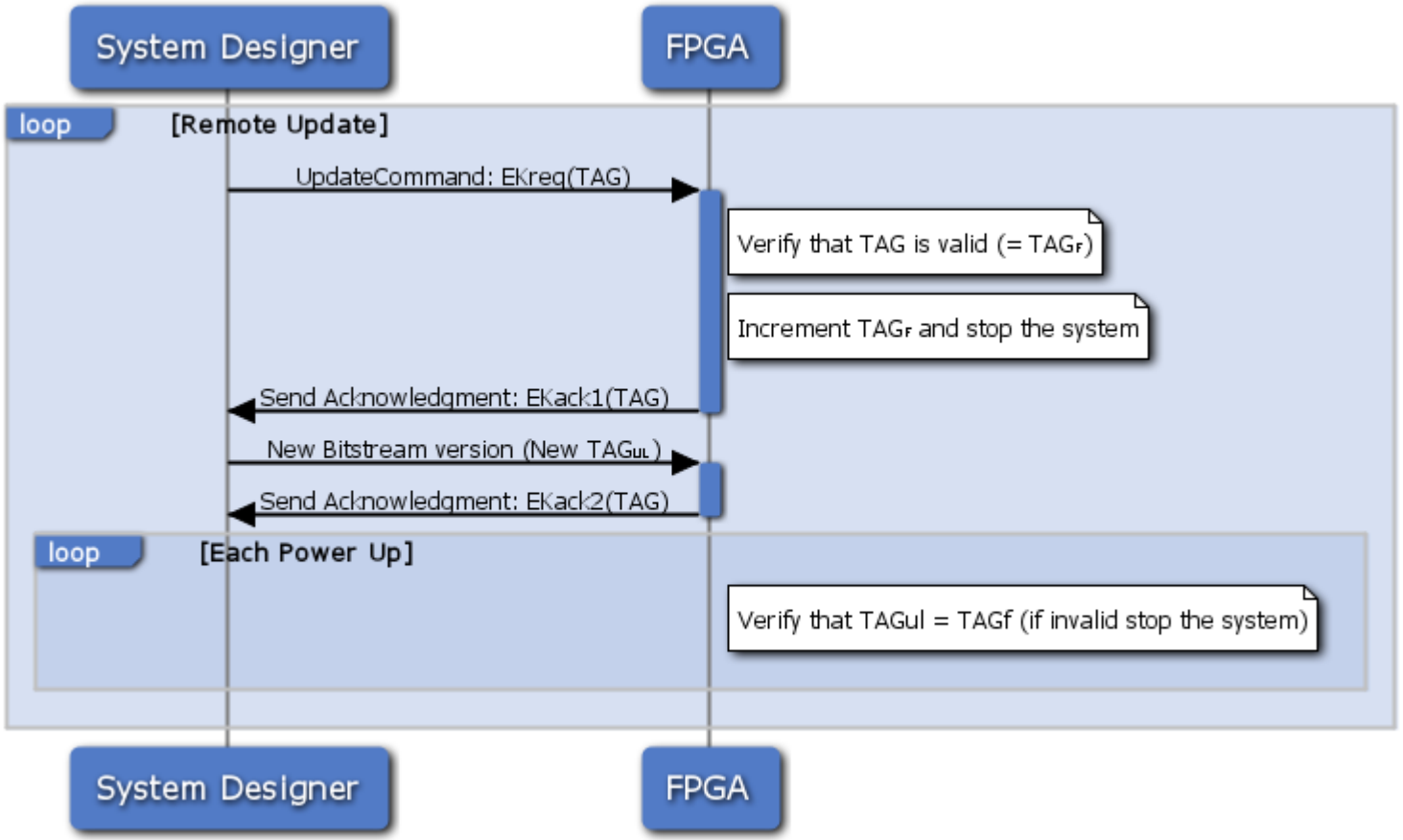


CIPHER = C (TAG$_{UL}$+Bitsream)$_{Kb}$
CHECK TAG$_{UL}$, TAG$_F$, Nonce

- Kreq: for the Update command
- Kack1: for the Update command acknowledgement
- Kack2: for the new bitstream version reception and startup acknowledgement

3. Secure update principle

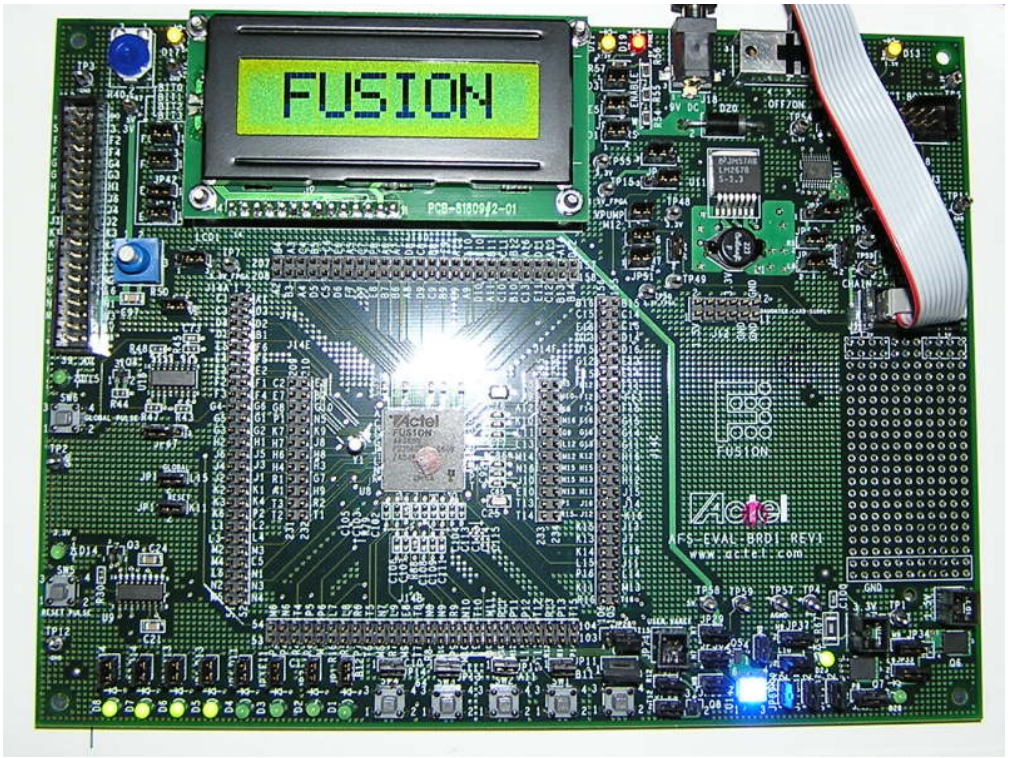
Protocol overview:



4. Implementation

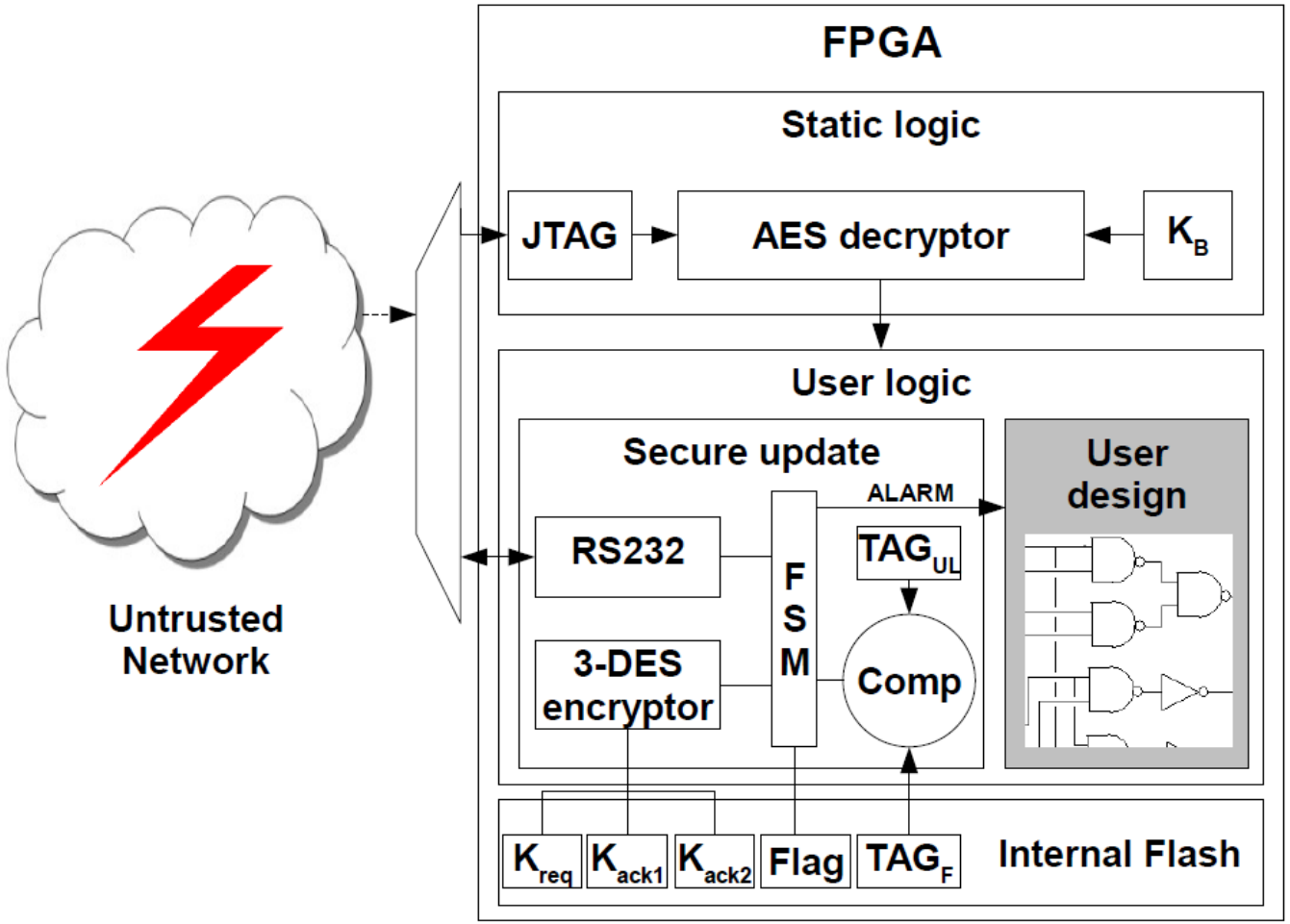
Based on an Actel Fusion starter kit (Fusion AFS600):

- Flash FPGA with flash memory
- ISP: AES-based MAC (Confidentiality and integrity)
- Low cost FPGA



4. Implementation

Based on an Actel Fusion starter kit (Fusion AFS600):



4. Implementation

Step 1: Power-up

```
1   Read (TAG_F)
2   if (TAG_F ≠ TAG_UL) then
3     goto 22
4   end if;
```

Step 2: First power-up

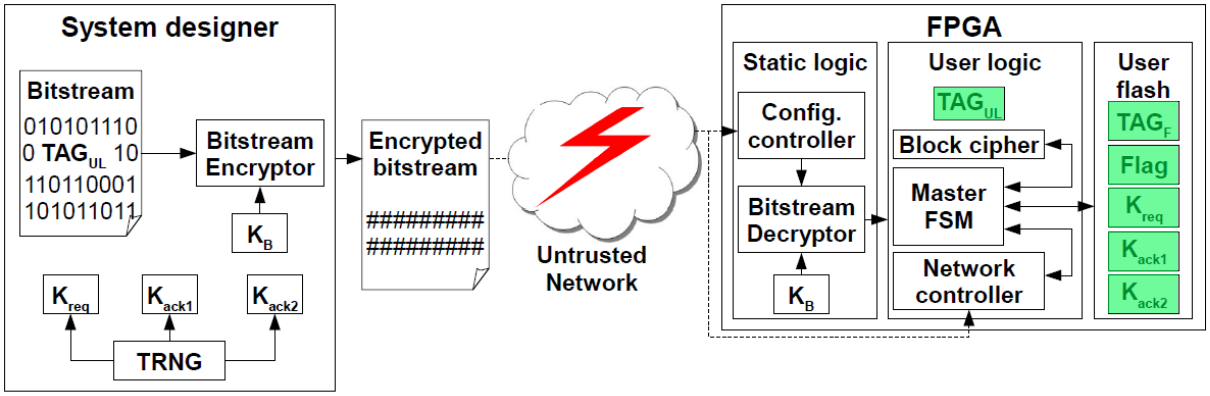
```
5   Read (flag)
6   if (flag = true) then
7     Read (K_ack2)
8     CTAGKack2 := E_Kack2 (TAG_UL)
9     Send(CTAGKack2)
10  end if;
```

Step 3: Authentication

```
11  Read (K_req)
12  CTAGKreq := E_Kreq (TAG_UL)
13  Read (K_ack1)
14  CTAGKack1 := E_Kack1 (TAG_UL)
15  Wait for CMD
16  If (CMD = CTAGKreq) then
```

Step 4: TAG_F incrementation

```
17    Write (TAG_F+1)
18    Send(CTAGKack1)
19  Else
20    goto 15
21  end if;
22  SYSTEM SHUTDOWN
```



4. Implementation

Timing overhead:

Step	# Cycles	Duration(ns) F=60MHz
1. Power-up	54	900
Read TAG_F + Read flag	54	900
2. First power-up	187	3,117
Read K_{ack2}	47	783
Write flag + $E_{K_{ack2}}$(Tag)	140	2,333
Send CTAGKack2	N/A	N/A
3. Authentication	175	2,917
Read K_{req}	79	1,317
Read K_{ack1} + $E_{K_{req}}$(Tag)	48	800
$E_{K_{ack1}}$(Tag)	48	800
4. TAG_F incrementation	108	1,800
Write TAG_F + Write flag	108	1,800
Send CTAGKack1	N/A	N/A
Total	**524**	**8,733**

Hidden

Hidden

Not significant

4. Implementation

Area overhead:

Entity	# Tiles	Fraction of Actel AFS600	Fraction of Actel AFS1500
3-DES	1305	9%	3%
RS232	418	3%	1%
Flash Controller (including Actel Core CFI)	1005	7%	3%
Master FSM	777	6%	2%
Total	**3505**	**25%**	**9%**

Only master FSM is dedicated for bitstream protection: Not reusable

4. Implementation

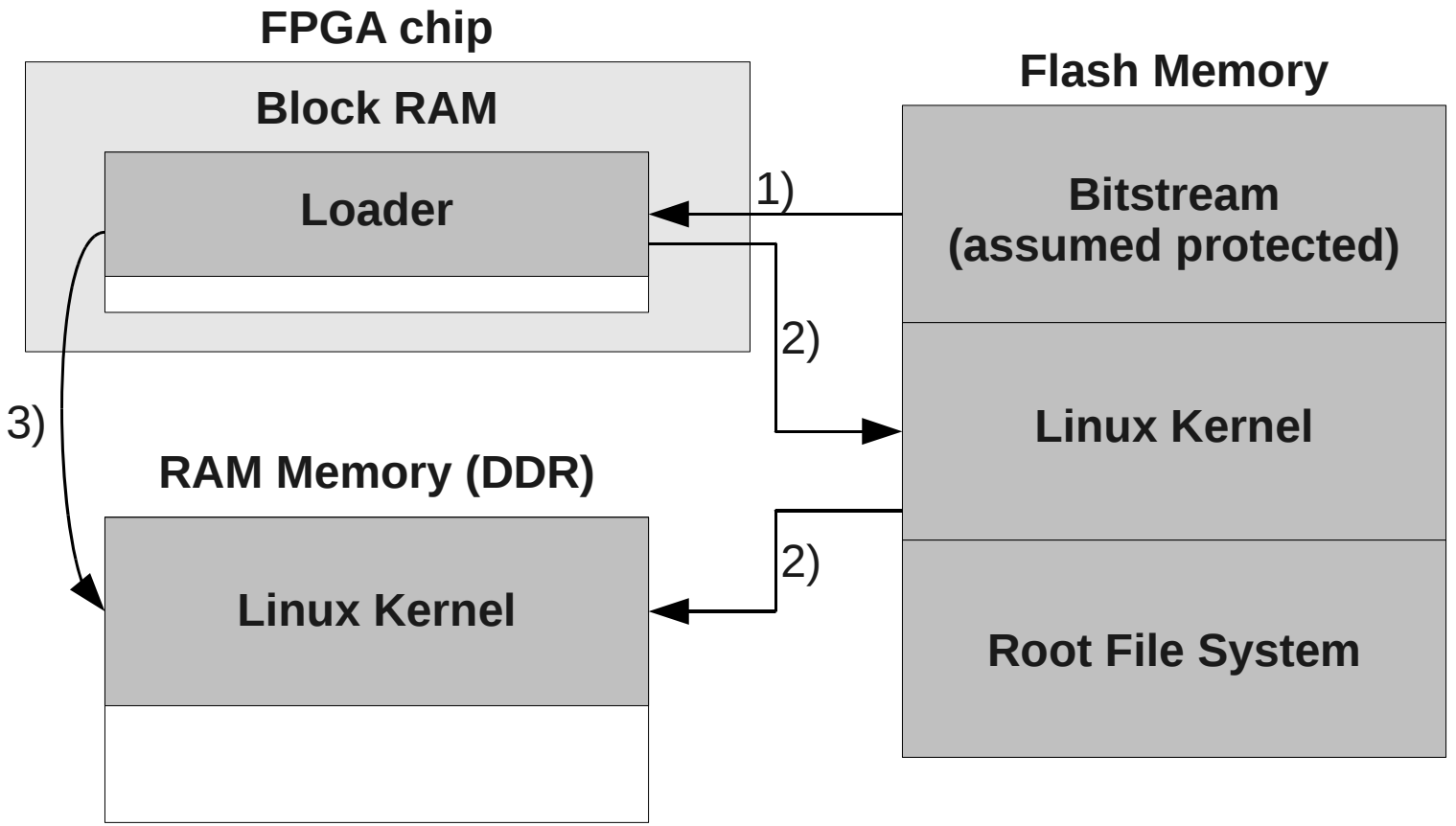
- **Easy to use (non-specific bitstream)**
- **Quasi zero timing overhead**
- **Low area overhead when reusability is possible**
- **Implemented**

Solution	Modification of the static logic	Development time	Area overhead	Additional cost
1	None	High	High	Regular polling
2	Yes	Low	None	None
3	Low for Flash based FPGA	Medium	Low / Medium	Specific bitstream
New	Low for Flash based FPGA	Medium	Low / Medium	None

Requires a non-volatile register

5. Case Study

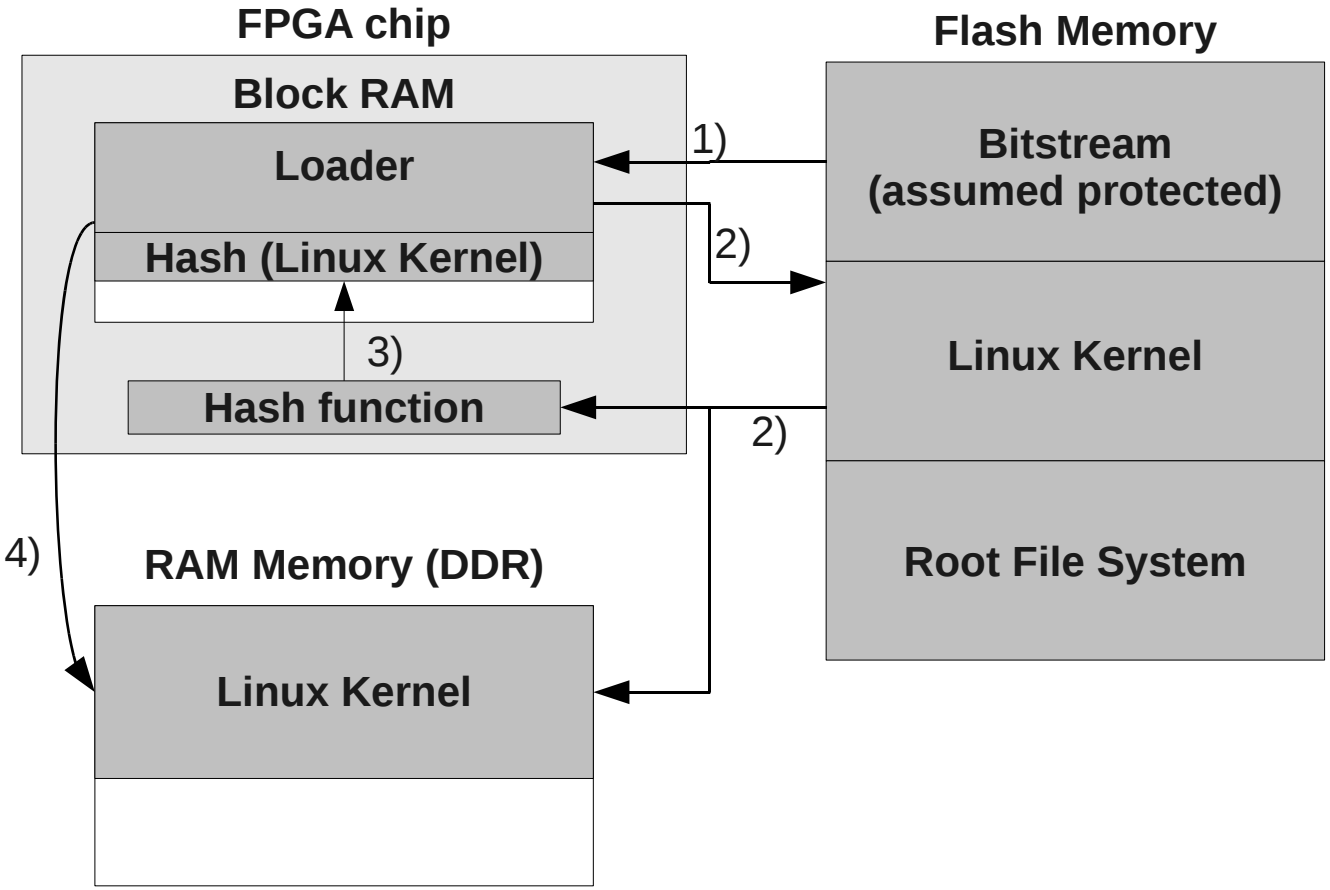
- **Applied to OS kernel update**



Boot steps :

1) The loader is stored in block RAM at power-up from bitstream
2) Loader copies Kernel from Flash to RAM
3) The loader branches to the Kernel and Linux boots

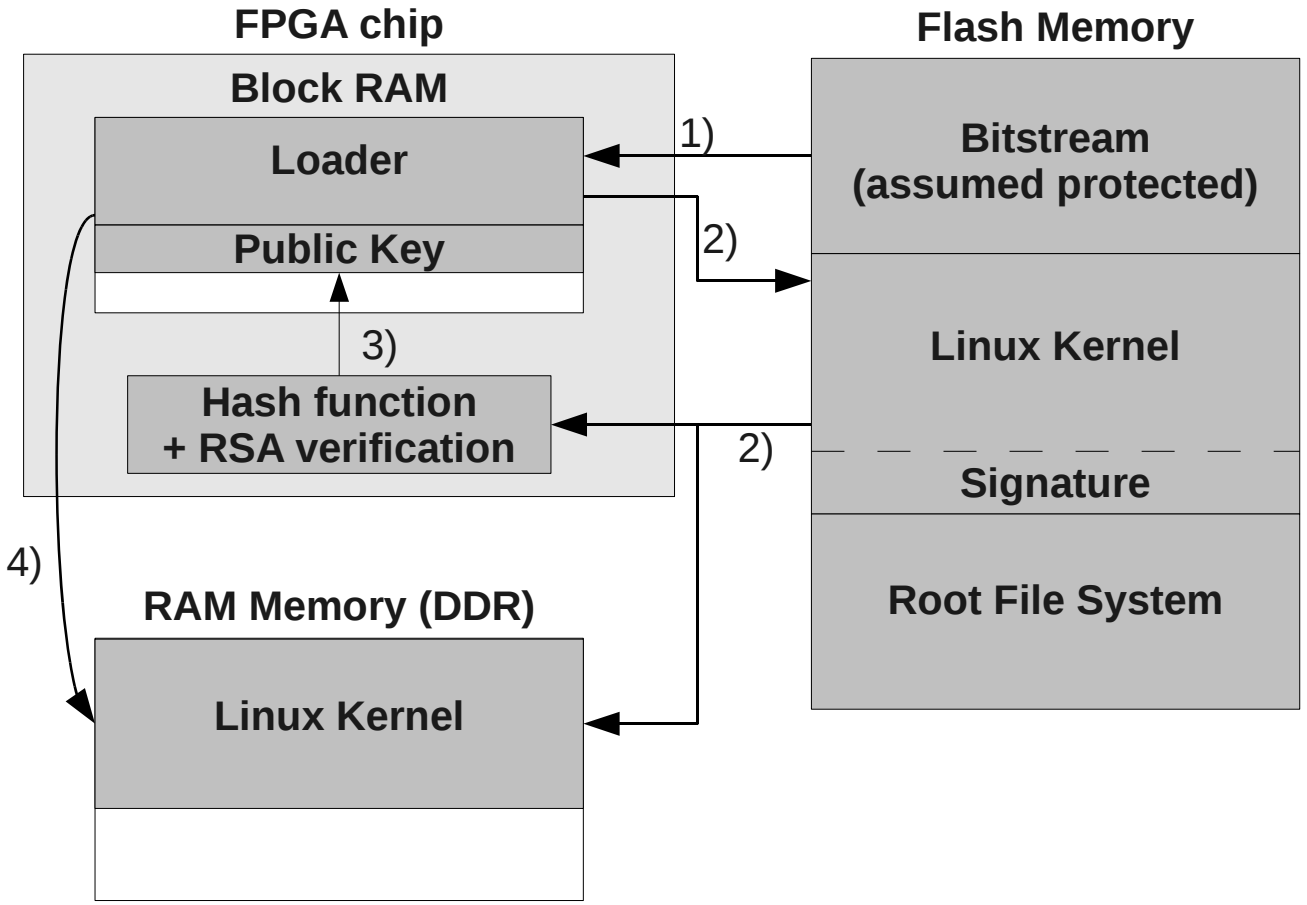
5. Case Study



Boot steps :

1) The loader is stored in block RAM at power-up from bitstream
2) The loader copies Kernel from Flash to RAM and compute its hash
3) The loader verify the Kernel integrity thanks to the hash
4) The loader branches to the Kernel and Linux boots

5. Case Study



Boot steps :

1) The loader is stored in block RAM at power-up from bitstream
2) The loader copies Kernel from Flash to RAM and compute its hash
3) The loader verify the Kernel integrity by verifying the signature
4) The loader branches to the Kernel and Linux boots

Results :SHA-256 – Linux Kernel (2.8 Mo)
Virtex 6 : Processor Frequency, 100Mhz

Soft Implementation = 295 860 775 Cycles, 2.96s
Hard without DMA = 38 376 545 Cycles, 0.39s x7.7
Hard with DMA = 4 221 304 Cycles, 0.04s x9.1 x70

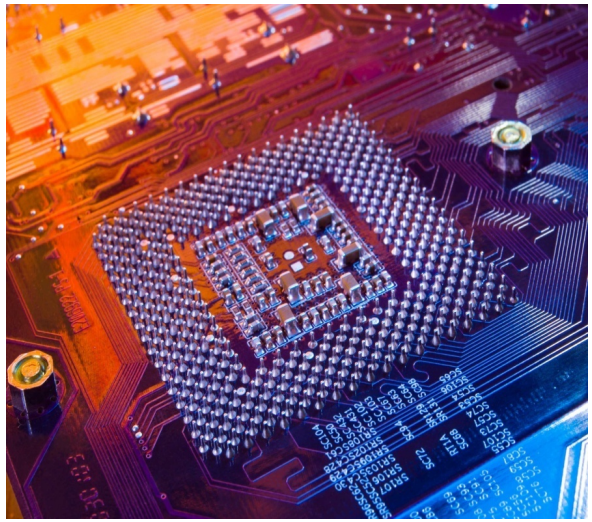
Signature verification (RSA 1024
RSA = 92 867 Cycnles, 0.001s

Total Overhead < 50ms

	Details				Total			
	Components	**Slice FF**	**Slice LUT**	**BRAM**	**Slice FF**	**Slice LUT**	**BRAM**	***Fraction of V6 VLX240T***
Base system (or with soft SHA-256)	Microblaze	3 196	3 874	19				
	Cache	6	14	16				
	DDR3	5 091	4 245	11				
	Flash	479	389					
	PLB	178	657		***8 950***	***9 179***	***46***	***6% + 11% de BRAM***
+ Hard SHA-256	SHA (+wrapper)	1 509	1 897	1				
	Intc	190	180		***10 649***	***11 256***	***47***	***7% + 11% de BRAM***
+ DMA	DMA	561	799		***11 210***	***12 055***	***47***	***8% + 11% de BRAM***
+ RSA	RSA (+wrapper)	684	989	4	***11 894***	***13 044***	***51***	***9% + 12% de BRAM***

5. Conclusions

- Set of solution for Bitstream protection
  - Confidentiality
  - Integrity (spoofing, replay)
  - Denial of Service
  - Downgrade
- Compatible with actual FPGA technology
- Use of NVM register is advised
- Example applied on Boot Loader
- Extend this work to memory and bus transactions



Thank you for your attention!

Questions?