Versatile RISC-V ISA GF extension for Cryptography and error-correction codes

Yao-Ming Kuo

UNIVERSIDAD NEBRIJA | ARIES

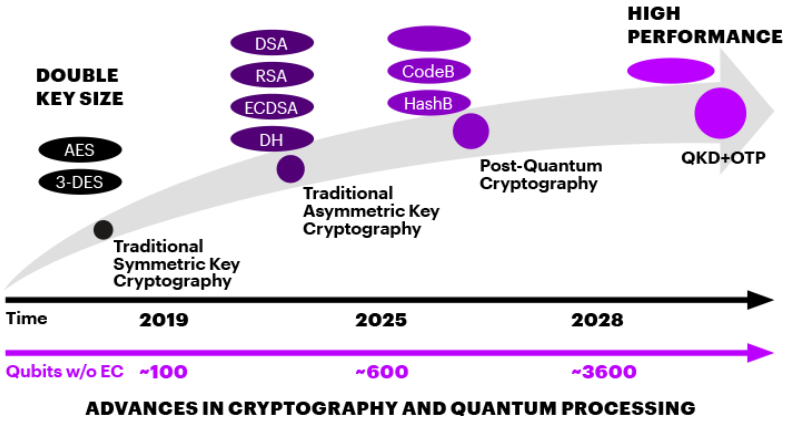
Contents

- Challenges
- Our approach
- Galois field arithmetic
- ISA extension proposal
- Implementation
- Performance and FPGA resources
- Conclusion
- Future work

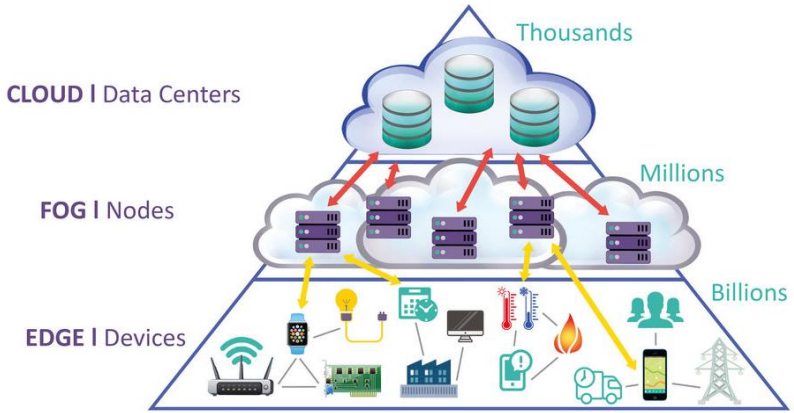
Challenges

- The devices send all the information to a centralized authority, which processes the data.
  - Latency
  - Heavy workload at the cloud side (limitations and availability)
  - Privacy (sec)

- A platform, which enables the computation, communication and storage closer to the network is required.



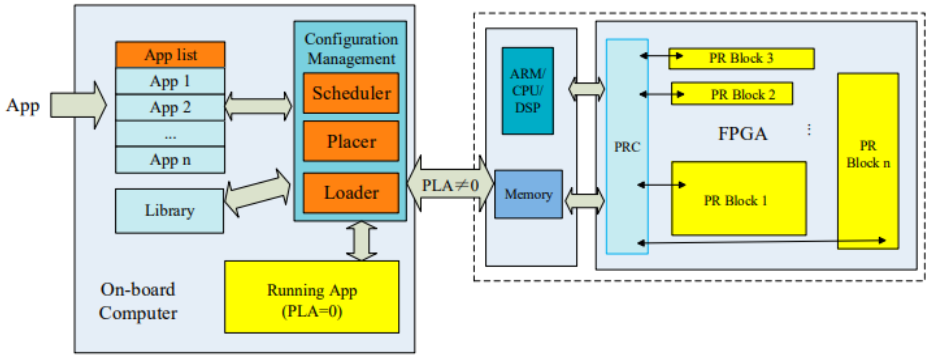
Accenture. (2018). Cryptography in a post-quantum world.



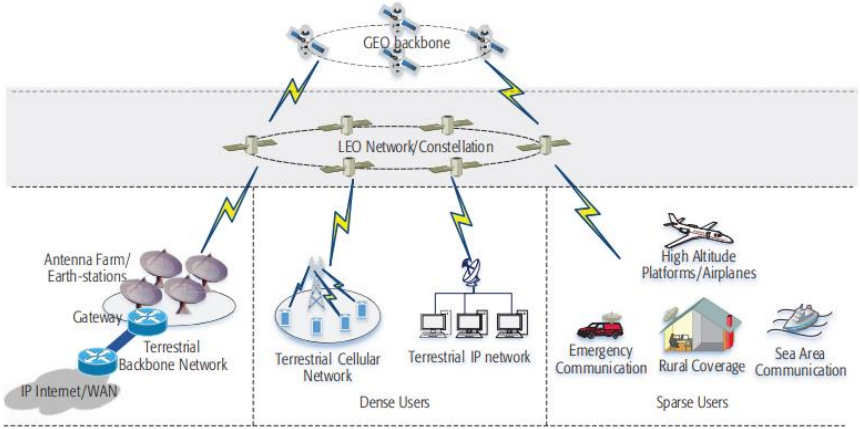
Qian, Jia & Sengupta, Sayantan & Hansen, Lars. (2019). Active Learning Solution on Distributed Edge Computing.

Challenges

- Satellite mobile edge computing: The hardware must be flexible and be able to attend different tasks (different protocols) with low latency and power consumption.



Wang, Y., Yang, J., Guo, X., & Qu, Z. (2019). Satellite edge computing for the internet of things in aerospace. Sensors, 19(20), 4375.



Zhang, Z., Zhang, W., & Tseng, F. H. (2019). Satellite mobile edge computing: Improving QoS of high-speed satellite-terrestrial networks using edge computing techniques. *IEEE network*, *33*(1), 70-76.

Our approach

- Focus on general-purpose lightweight processors, which operates with different algorithms based on GF(2^m).
    - Error-correction codes (i.e, Non-binary LDPC, BCH, RS codes)
    - Pre-quantum cryptography (i.e, AES, Elliptic Curve)
    - Post-quantum cryptography (i.e, McEliece, Rainbow, HQC)

- Proprietary ciphers based on GF(2^m).
- Hardware reutilization (No specific logic for each different algorithm). Flexibility.

Galois field arithmetic

- GF(2^m) Addition

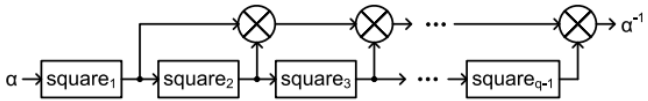
$$a(x) = a_{m-1}x^{m-1} + \ldots + a_1 x + a_0$$
$$b(x) = b_{m-1}x^{m-1} + \ldots + b_1 x + b_0$$
$$a_i \wedge b_i \in GF(2), 0 \leq i \leq m-1$$

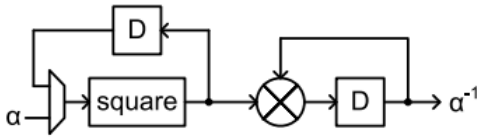
$$s(x) = (a_{m-1} \oplus b_{m-1})x^{m-1} + \ldots + (a_0 \oplus b_0)$$

XOR of its coefficients

- GF(2^m) Multiplication (Traditional two-step multiply)
  - First step: Carry-Less multiplication (CLMULH / CLMUL)
  - Second step: Reduction depending on the irreducible polynomial (FFRED)

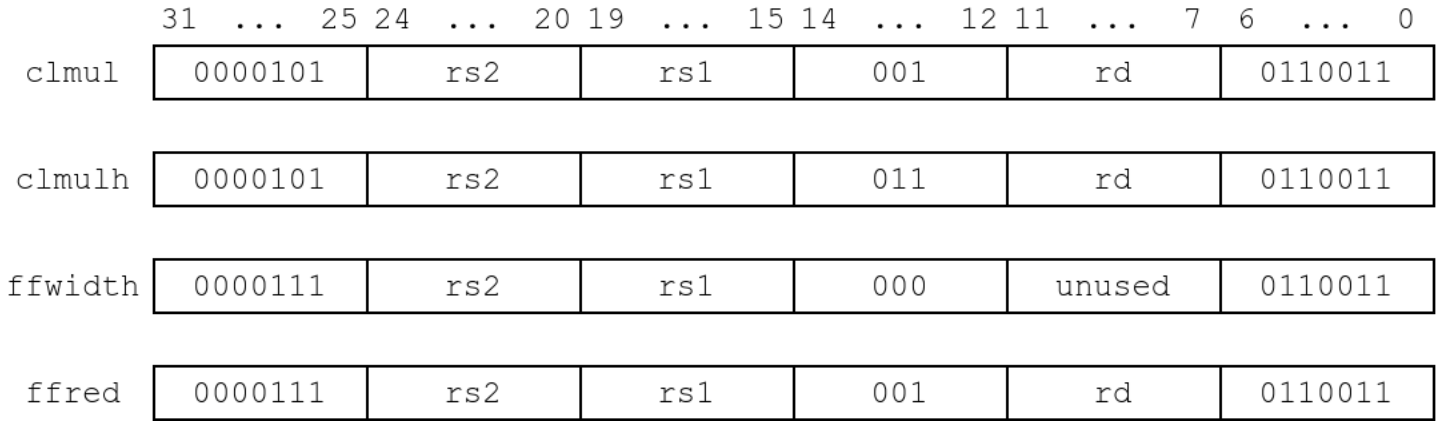
- GF(2^m) Inversion





Xinmiao, Z. (2016). VLSI architectures for modern error-correcting codes. Crc Press.

ISA extension proposal

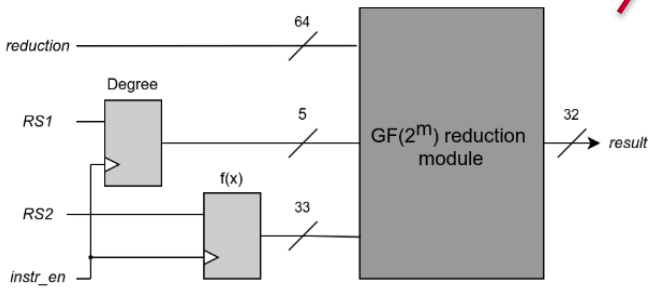


- FFWIDTH: It receives in RS1 the degree of the polynomials, and in RS2, the irreducible polynomial.
- FFRED: It receives the polynomial to be reduced as a parameter. In RS1, it receives the high part, and in RS2, the low part of the polynomial. This instruction returns the reduced polynomial $c\ (x)$.
- CLMULH and CLMUL: The parameters are the same as extension B.

Example: GF multiplication for AES

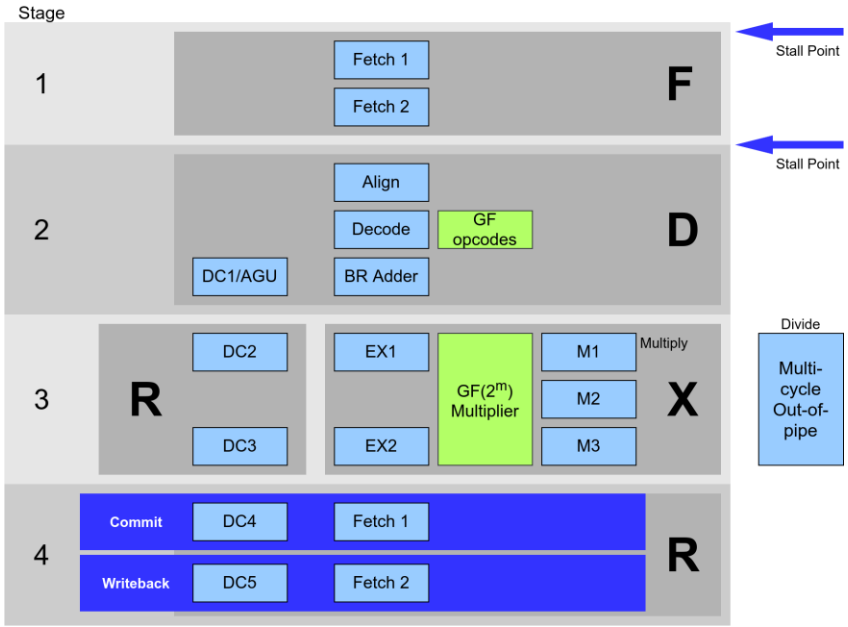
Irreducible polynomial GF(2^8)

$$f(x) = x^8 + x^4 + x^3 + x + 1$$
$$100011011_2 = 11B_{16} = 283_{10}$$

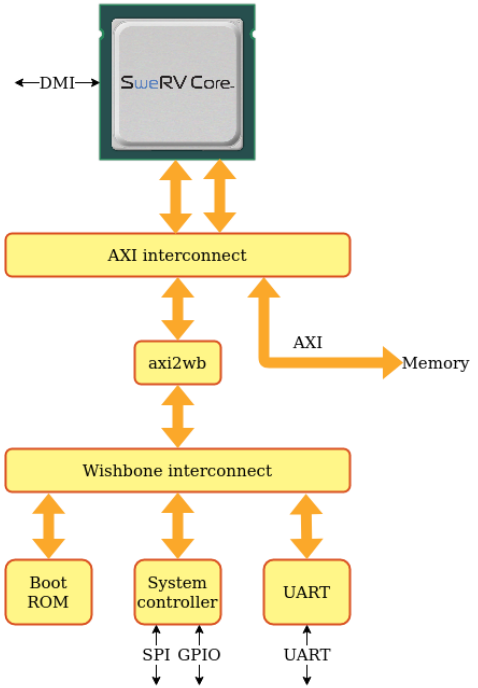



```
li        a5, 8          % Polynomial degree
li        a4,283         % Primitive poly
ffwidth   a5,a5,a4
...
lbu       a4,0(a0)       % Operand A
lbu       a7,0(a1)       % Operand B
clmul     a4,a4,a7       % CL multiplication
li        a5,0
ffred     a7,a5,a4       % Reduction
```

ISA implementation



Western Digital Corporation. 2020. RISC-V SweRV EL2 Programmer's Reference Manual. https://github.com/chipsalliance/Cores-SweRV-EL2/blob/branch-1.3/docs/RISC-V_SweRV_EL2_PRM.pdf



Olof Kindgren, SweRVolf Github repository. https://github.com/chipsalliance/Cores-SweRVolf

Performance

AES128	CBC Enc.	CBC Dec.	CTR Enc.	CTR Dec.	ECB Enc.	ECB Dec.
standard	197,920	198,240	198,208	198,197	50,641	50,726
our proposal	38,328	39,303	39,033	38,995	10,854	11,011
Reduc. %	80.63%	80.17%	80.31%	80.33%	78.57%	78.29%

AES192	CBC Enc.	CBC Dec.	CTR Enc.	CTR Dec.	ECB Enc.	ECB Dec.
standard	242,573	242,695	242,839	242,828	62,572	62,661
our proposal	47,016	48,019	47,637	47,617	13,764	13,939
Reduc. %	80.62%	80.21%	80.38%	80.39%	78.00%	77.75%

AES256	CBC Enc.	CBC Dec.	CTR Enc.	CTR Dec.	ECB Enc.	ECB Dec.
standard	285,245	285,593	285,439	285,425	72,331	72,416
our proposal	53,396	54,548	54,054	54,036	14,462	14,660
Reduc. %	81.28%	80.90%	81.06%	81.07%	80.01%	79.76%

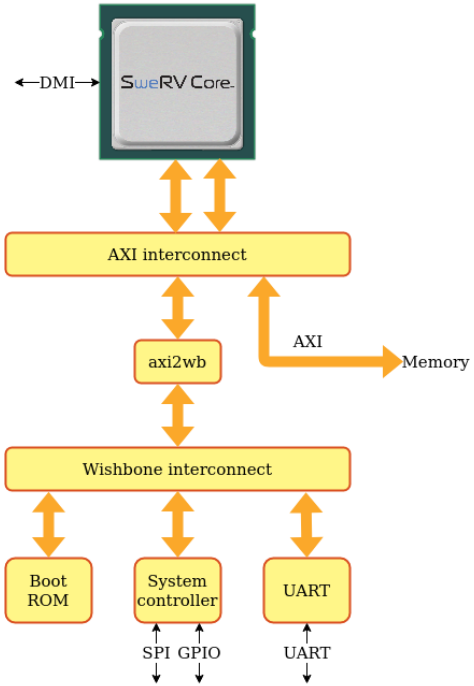
Number of cycles for AES

	RS(255,247)		RS(255,239)	
	Encode	Decode	Encode	Decode
standard	154,003	151,681	300,831	303,289
out proposal	29,006	22,648	58,660	45,237
Reduc. %	81.17%	85.07%	80.50%	85.08%

Number of cycles for RS codes

FPGA resources – Nexys A7

SweRV-EL2	Slice LUTs	Slice Registers	F7 Muxes	F8 Muxes	Slice	LUT as logic
standard	18,605	7651	341	74	5,329	18,605
our proposal	19,974	7688	413	80	5,699	19,974
Inc. %	7.36%	0.48%	21.11%	8.11%	6.94%	7.36%



Olof Kindgren,
SweRVolf Github repository.
https://github.com/chipsalliance/Cores-SweRVolf

Conclusion

- Flexible GF(2^m) arithmetic extension is proposed.
    - Error-correction codes (i.e, Non-binary LDPC, BCH, RS codes)
    - Pre-quantum cryptography (i.e, AES, Elliptic Curve)
    - Post-quantum cryptography (i.e, McEliece, Rainbow, HQC)

- More than 4x acceleration (~80% reduction in clock cycles) for AES and Reed-Solomon.
- Increment of only 7% in logic utilization (slices) for SweRV EL2.
- Max. operation frequency remains the same for SweRV EL2.

Future work

- Square and inverse operation implementation.
- Post-Quantum cryptography performance results.

Thank you!