(Comparing) Hardware Complexity of Cryptographic Algorithms

Liam Marnane

University College Cork

Claude Shannon Institute

Thanks

Work of the following Post-Graduate Students

- Dr Francis Crowe
- Maurice Keller
- Andrew Byrne

Outline

- Hardware & Measuring Hardware Complexity
- Comparing Elliptic Curve Implementations to Other Cryptographic Systems
- Comparing Complexity of Different Elliptic Curve Implementations

Why Hardware?

- Hardware allows exploitation of Parallelism in Algorithm.
- Hardware Implementation for Increase in Processing Speed:
  - Increased Throughput:-

    Number of bits processed per second.
  - Reduce Time Taken:-

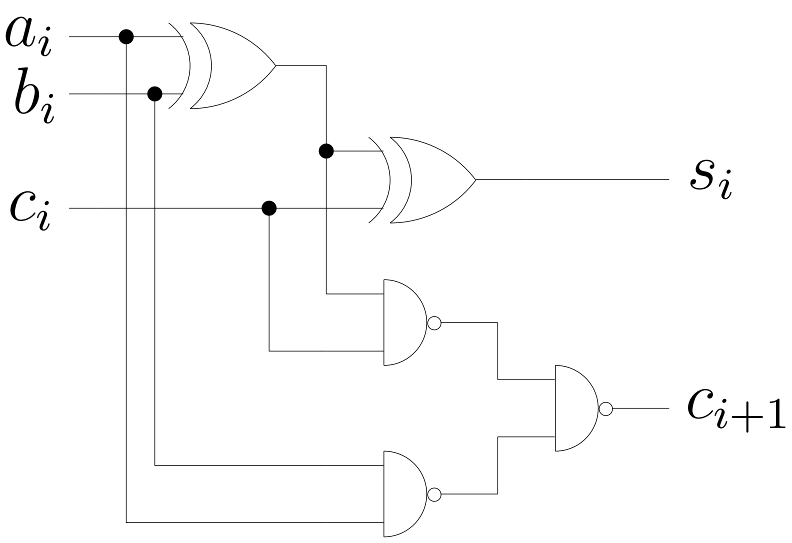
    How long it takes to perform the calculation.
- Other Benefits:
  - Reduce Power
  - Increased Security

Hardware Complexity?

- Cost Benefit Analysis
  - How do we measure the Benefit of the implementation?
  - How do we measure the Cost of the implementation?
- Use Metrics
  - Clock Speed, Throughput, Time taken
  - Area
  - Power, Energy

Cost:- Area

- No such thing as Free Silicon

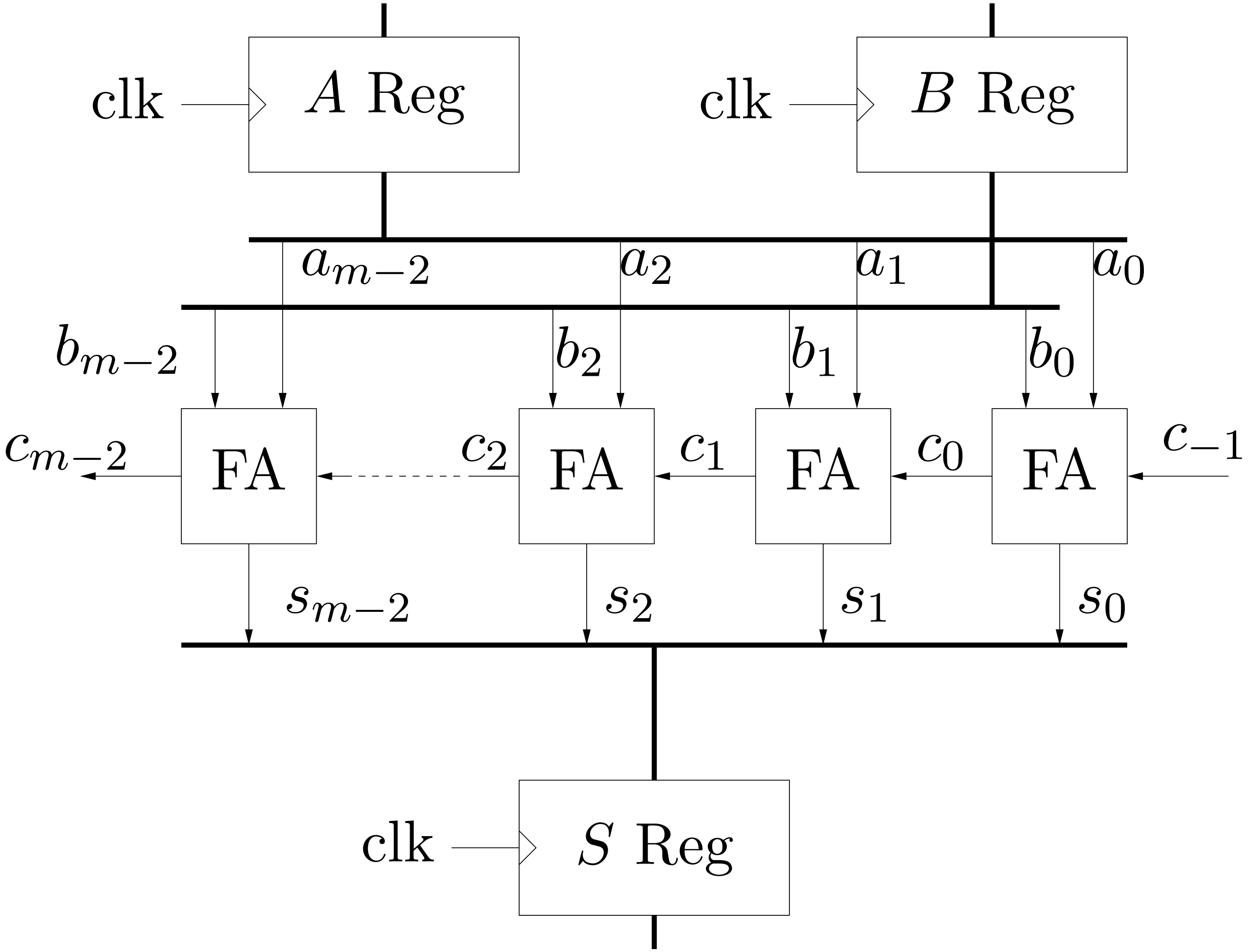


- Number of Transistors
  - Number of Boolean Gates or Combinational Logic
  - Number of Flip Flops or Synchronous Logic (Registers, Memory)
- Wiring or Interconnect
  - Can Dominate Area in Large Designs

Benefit:- Processing Speed

- Clock Speed of Design (MHz GHz)
- Clock Speed determined by the time the hardware takes to carry out an operation.
  - Addition:- Very fast Circuit
  - Multiplication:- Slower Circuit
- Critical Path of Circuit
  - Change Input
  - ⇒ Time through each gate and wire
  - ⇒ Output available

Ripple Carry Adder



Critical Path

Output Delay of A/B Register

+

Ripple of Carry through Combinational Full Adders

+

Setup time of S Register

Throughput vs Time Taken

- Throughput
  - Bits per Second (Hopefully MBits/S or GBits/S).
  - How long it takes to encrypt a Book using AES.
  - How many public key signatures per second can be calculated using RSA on an e-commerce server.
- Time Taken
  - Seconds (Hopefully mS and μS)
  - How long it takes to carry out a single key exchange using ECC on a PDA

Low Power

- Cost or Benefit
- Mobility or Heat Dissipation
- Energy or Power
- Energy :- Current flowing throughout calculation
  - Battery Lifetime
- Power:- Maximum Current flowing at particular time
  - Battery Type
- Trade off in Battery design Power vs Energy

Hardware Complexity of ECC Implementations

- Compare FPGA Implementation of ECC to:
  - Private Key Algorithm AES
  - Hash Algorithm SHA
  - RSA
- Use:- Area, Clock Speed, Throughput and **Throughput per unit area**.

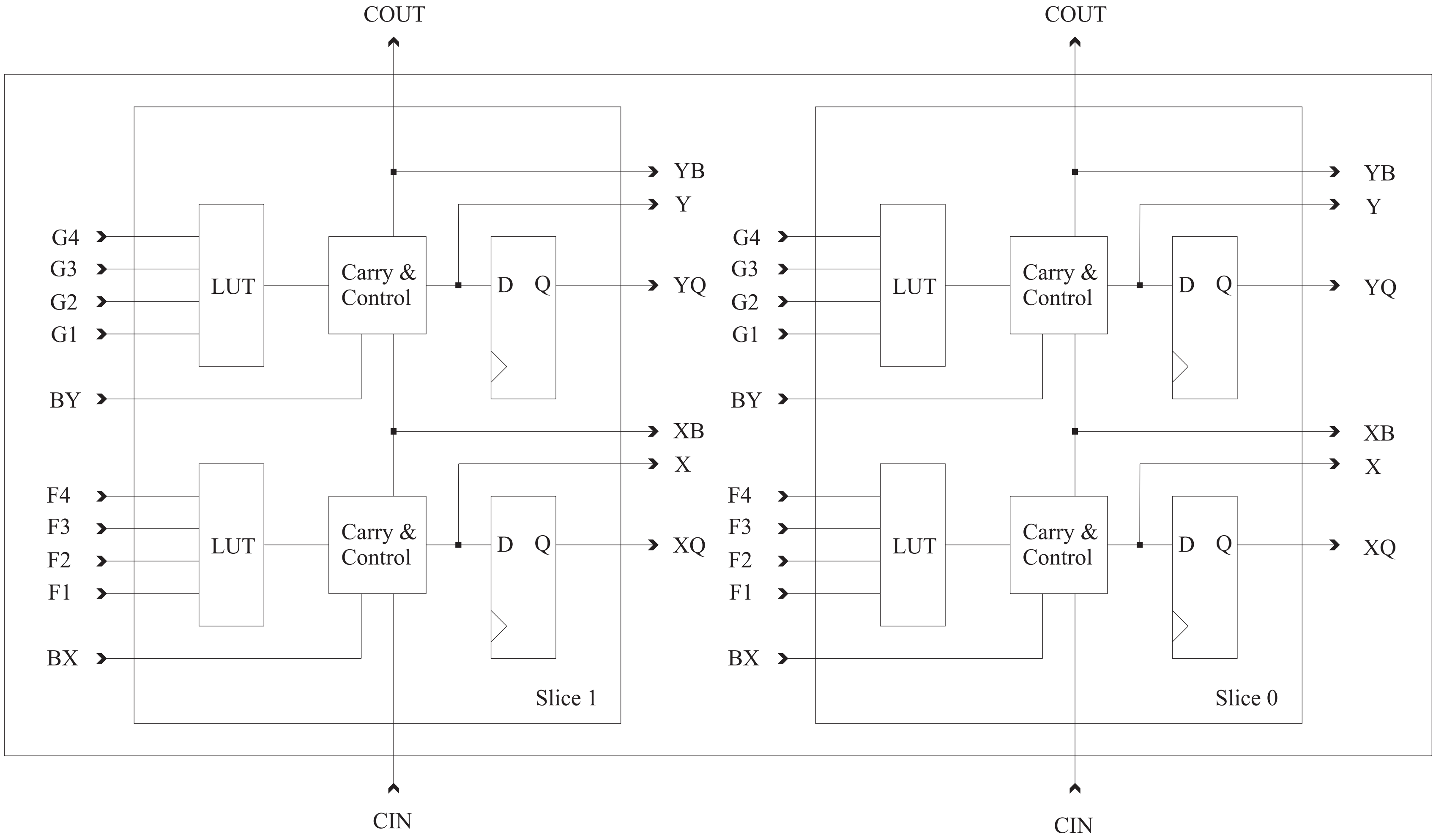
Field Programmable Gate Arrays

- Excellent for Rapid Prototyping of Hardware Implementations of Signal Processing Algorithms.
- Industry:- time to market.
- University Research:- Cost and Reuse.
- Very large FPGAs available.
- Are Suitable for Implementations of Cryptographic Algorithms.
- Complete Security protocol on a single FPGA

Underlying FPGA Architecture

- Typically, Field Programmable Logic Devices consist of:
  - 4-input Lookup Tables:- Boolean Logic
  - Simple D-type Latches:- FSM & Memory
  - Control Logic
- The device is arranged in an array of Configurable Blocks,

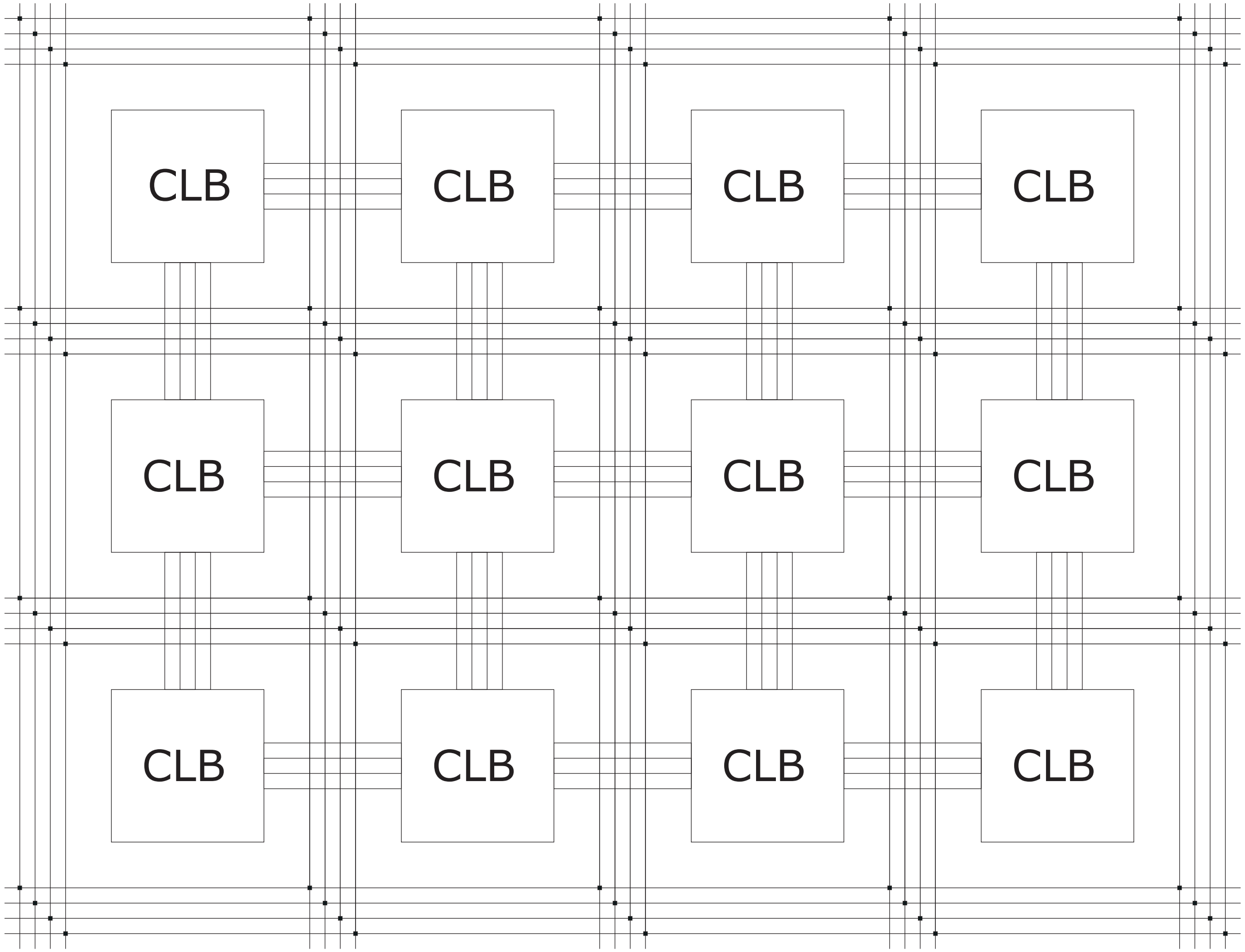
Xilinx Virtex Configurable Logic Block (CLB)



Underlying FPGA Architecture

- Typically, Field Programmable Logic Devices consist of:
  - 4-input Lookup Tables
  - Simple D-type Latches
  - Control Logic
- The device is arranged in an array of Configurable Blocks with communication between them:
  - local interconnect between adjacent CLBs:- **Fast**
  - Global interconnect for Buses and communication between functional blocks on FPGA:- **Slow**

Local and Global Interconnect between CLB's

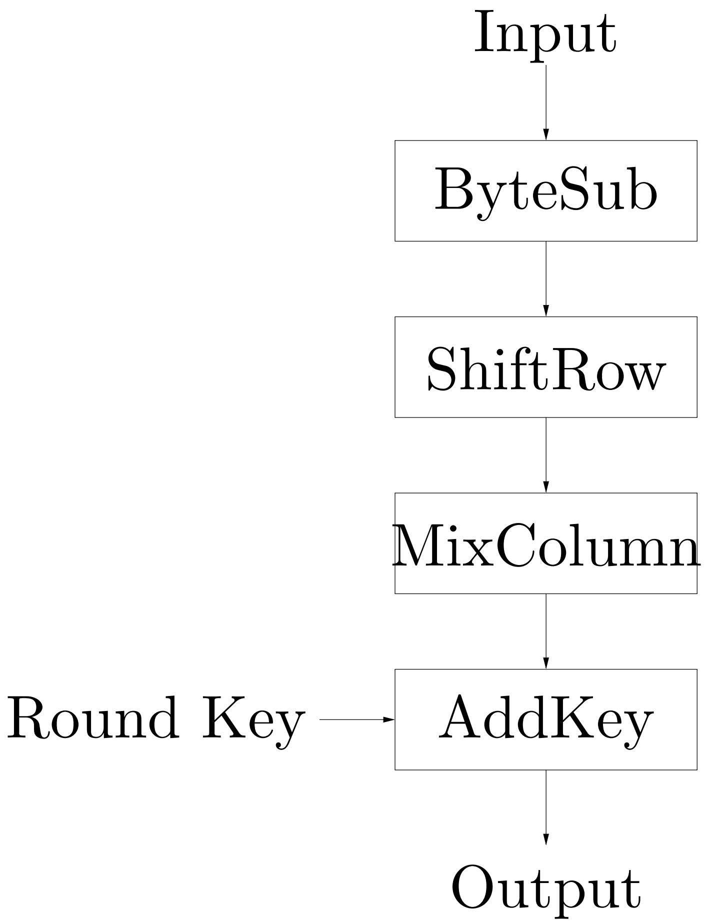


Underlying FPGA Architecture

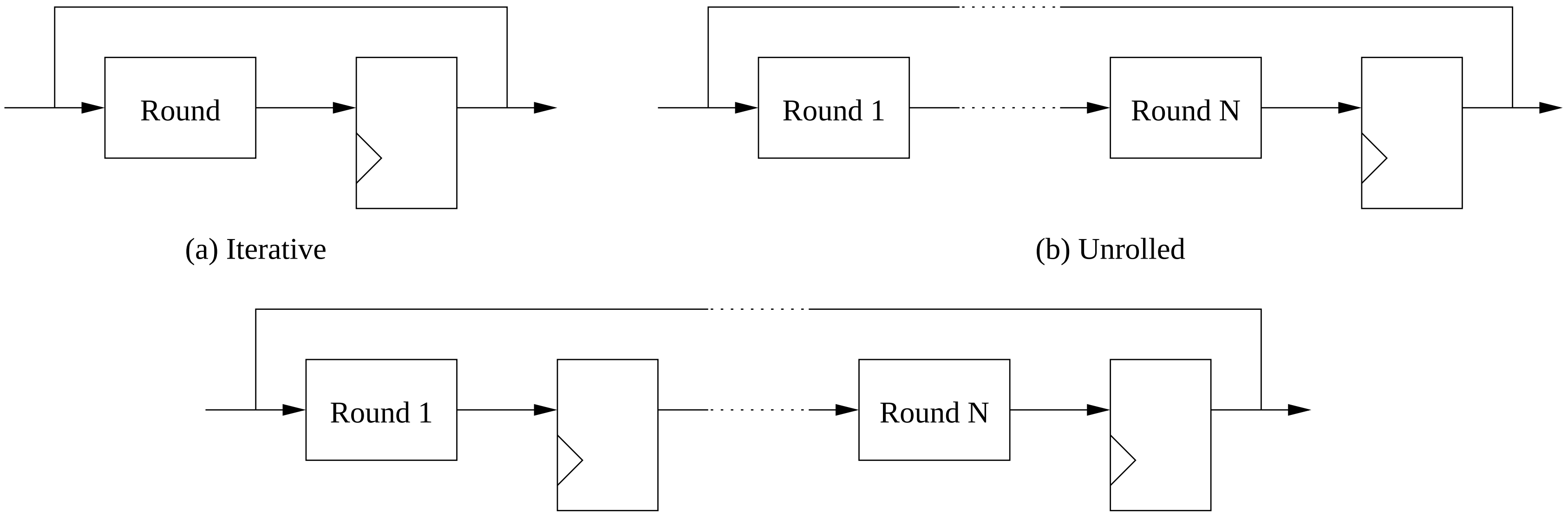
- Typically, Field Programmable Logic Devices consist of:
  - 4-input Lookup Tables
  - Simple D-type Latches
  - Control Logic
- The device is arranged in an array of Configurable Blocks, with local and global interconnect between
- Dedicated High-Speed Carry-Chain Propagation accelerates Arithmetic operations.
- Parallel multipliers, dedicated memory, RISC Processor.

Advanced Encryption Standard

- AES on 128 bits of data.
- Depending on the key size AES repeats 10, 12 or 14 times the basic round function.
- SubBytes Look Up table:-
  - Dominate Area
  - Number used dictates area and throughput.



Architecture Types



(a) Iterative

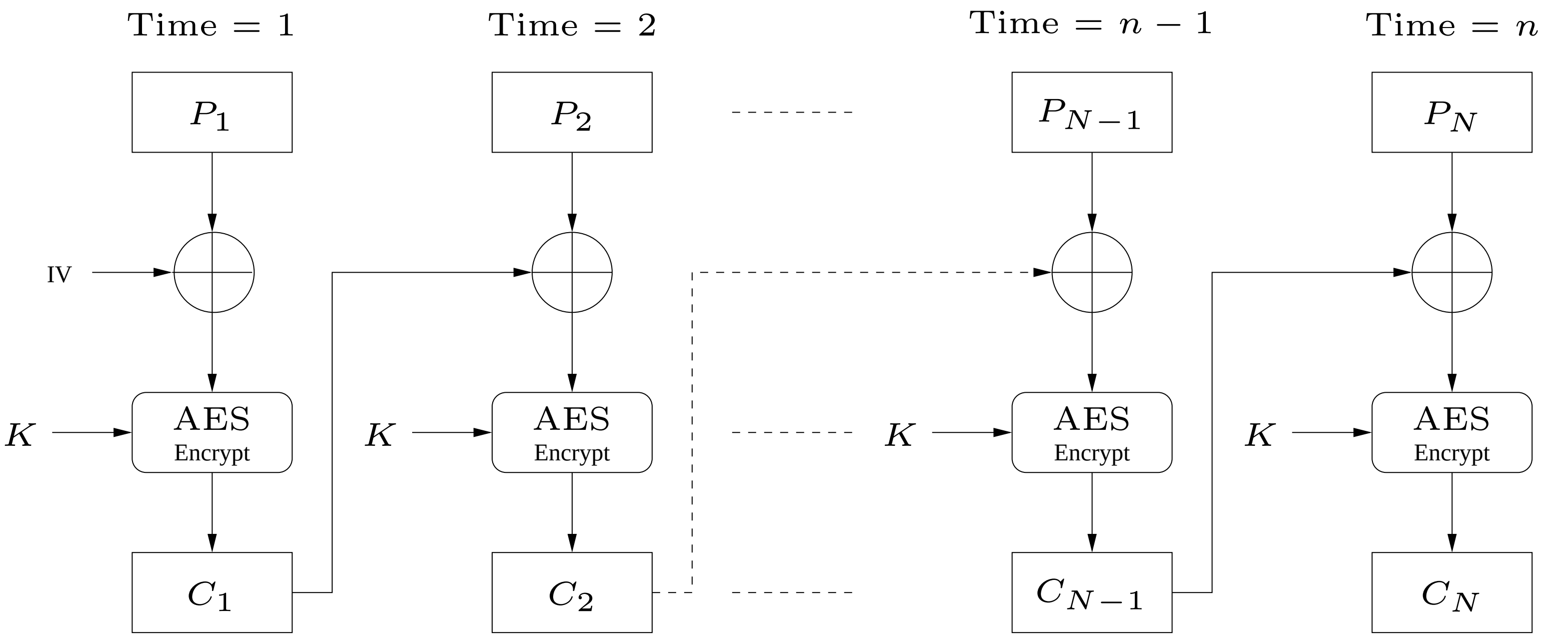
(b) Unrolled

(c) Pipelined

- Exploit Parallelism by Loop Unrolling.
- Increase Throughput through Pipelining:- Reduction in the length of critical path at cost of increased Latency.

Feedback Modes of Operation

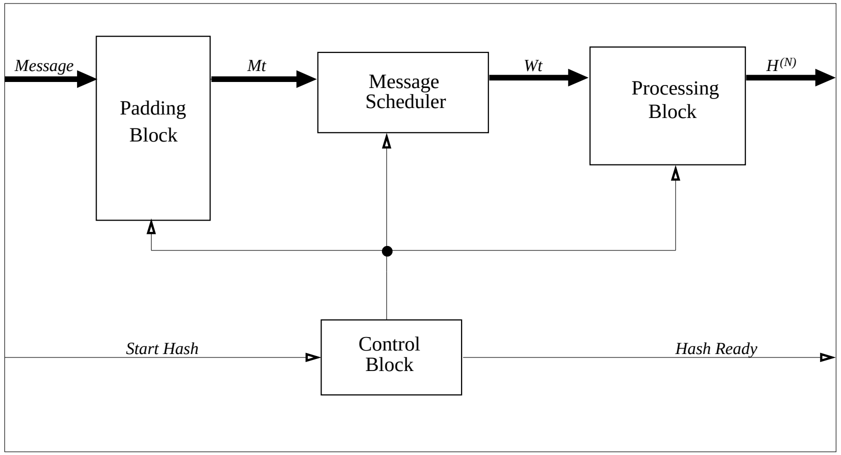
Cipher Block Chaining Encrypt



- Pipeline Cannot be kept filled as C_1 required before proceeding with P_2

Secure Hash Standard

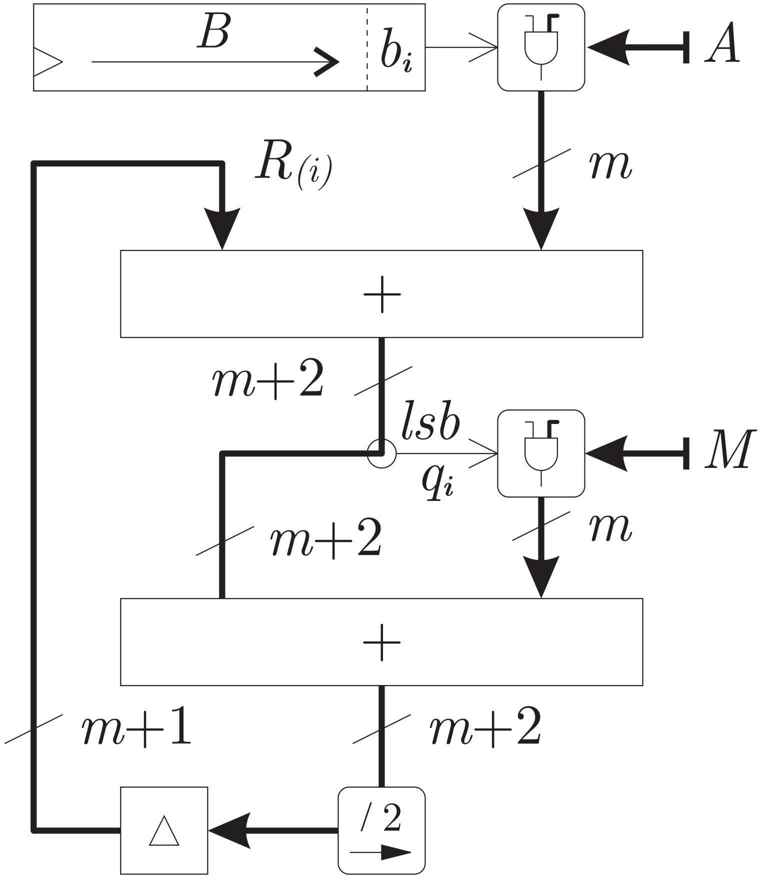
- SHA-512 operates on a message in 1024 bit blocks and produces a 512 bit hash value.
- Processing block operates on 64 bit word through 80 iterations of a compression function.
- Critical path is Five 64 bit additions.



- Architecture choices: Loop unrolling and pipelining.

Modular Multiplication

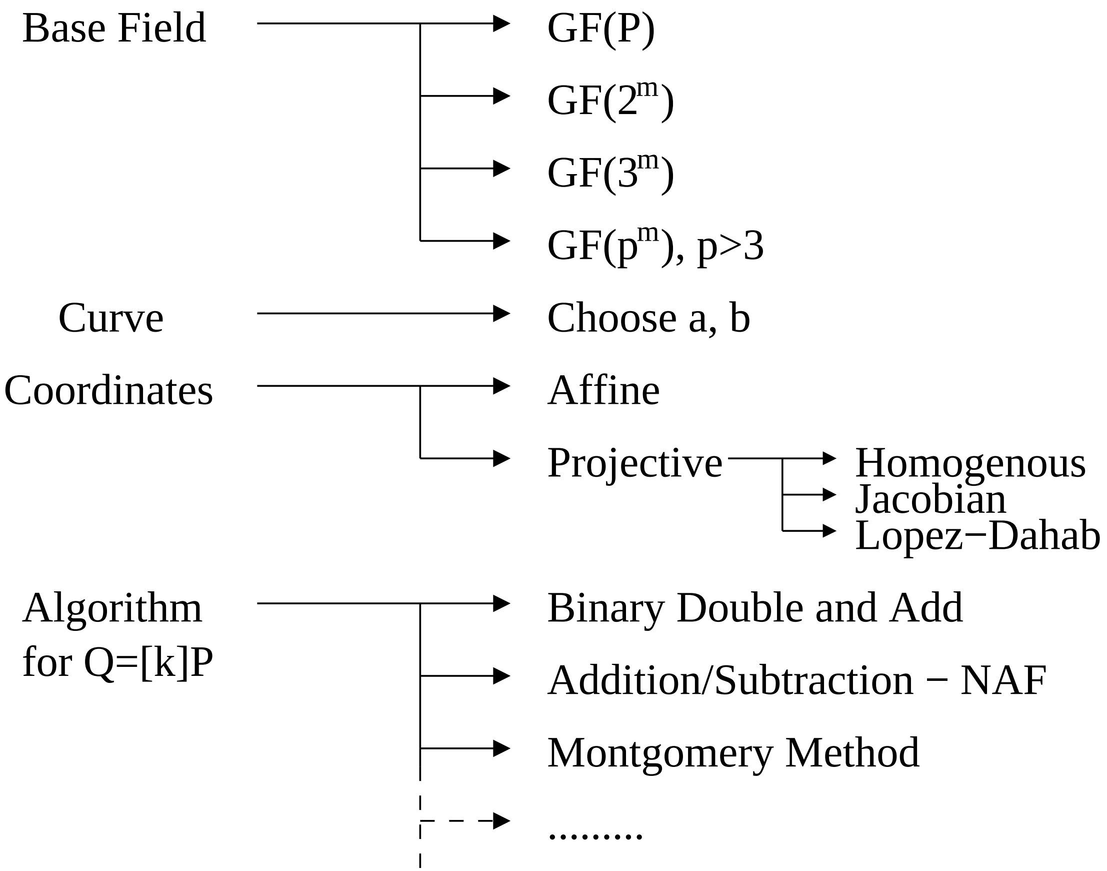
- Montgomery proposed modular multiplication through a series of additions & right-shifts ⇒ Suitable for hardware implementation.
- Bit lengths dictate bit serial or digital serial approach



RSA Architectures

- Number of Multipliers:- Exponentiation Algorithm Used
  - R-L Exponentiation:- 2 modular multipliers in parallel
  - L-R Exponentiation:- Single Modular Multiplier.
- Addition of Large Numbers:- Carry Save versus Carry Propagate, (Area versus time).
- Suitable for Extensive Pipelining to reduce the critical path.

EC Design Choices



Base Field → GF(P)

→ GF(2^m)

→ GF(3^m)

→ GF(p^m), p>3

Curve → Choose a, b

Coordinates → Affine

→ Projective → Homogenous

→ Jacobian

→ Lopez−Dahab

Algorithm for Q=[k]P → Binary Double and Add

→ Addition/Subtraction − NAF

→ Montgomery Method

→ ....

EC Design Implemented

- Prime Field F_p of 256bits (security equivalent to 3072bit RSA)
- Co-ordinates:
  - Affine requiring Modular Multiplication and Inversion
  - Jacobian Projective
- Bit serial Montgomery Multiplier
- Extended Euclidean Algorithm requiring 512 clock cycles

Design Complexity

Algorithm	Area	Clock	Throughput	Thpt/Area
	(CLBs)	(MHz)	(Mbps)	(bps/CLB)
AES	3,259	27.86	324	99417
SHA-512	2,468	40.02	506	205024
RSA - 1024	8,064	51.84	0.051	6.32
ECC-256A	2,718	19.19	0.00873	3.21
ECC-256J	1,353	20.45	0.00439	3.24

- Single Round of AES, with no Pipelining
- Single compression core for SHA
- R-L Algorithm for RSA, 2 Multipliers

- Affine Co-ordinates in F_p for ECC with multiplier, inverter and adder.
- Jacobian in F_p without conversion.

Design Complexity

Algorithm	Area	Clock	Throughput	Thpt/Area
	(CLBs)	(MHz)	(Mbps)	(bps/CLB)
AES	3,259	27.86	324	99417
SHA-512	2,468	40.02	506	205024
RSA - 1024	8,064	51.84	0.051	6.32
ECC-256A	2,718	19.19	0.00873	3.21
ECC-256J	1,353	20.45	0.00439	3.24

- Single Round of AES
- Key Expansion in hardware
- Encryption and Decryption

- No Pipelining
- 16 asynchronous ROMs used (60% of CLBs)

Design Complexity

Algorithm	Area	Clock	Throughput	Thpt/Area
	(CLBs)	(MHz)	(Mbps)	(bps/CLB)
AES	3,259	27.86	324	99417
SHA-512	2,468	40.02	506	205024
RSA - 1024	8,064	51.84	0.051	6.32
ECC-256A	2,718	19.19	0.00873	3.21
ECC-256J	1,353	20.45	0.00439	3.24

- Single Iterative Compression Block Used
- Carry Propagate Adders Used
- 4 Unrolled Architecture:- 3,650 CLBs

- Throughput:- 610 Mbs
- Clock:- 12.51 MHz
- Throughput per CLB of 167000

Design Complexity

Algorithm	Area	Clock	Throughput	Thpt/Area
	(CLBs)	(MHz)	(Mbps)	(bps/CLB)
AES	3,259	27.86	324	99417
SHA-512	2,468	40.02	506	205024
RSA - 1024	8,064	51.84	0.051	6.32
ECC-256A	2,718	19.19	0.00873	3.21
ECC-256J	1,353	20.45	0.00439	3.24

- R-L Algorithm for RSA
- Requires 2 Montgomery Multipliers
- Bit Length of 1026 required

- Carry Propagate Adders Used
- Extensive Pipelining
- Maximum Carry Chain of 130 bit.

Design Complexity

Algorithm	Area (CLBs)	Clock (MHz)	Throughput (Mbps)	Thpt/Area (bps/CLB)
AES	3,259	27.86	324	99417
SHA-512	2,468	40.02	506	205024
RSA - 1024	8,064	51.84	0.051	6.32
ECC-256A	2,718	19.19	0.00873	3.21
ECC-256J	1,353	20.45	0.00439	3.24

- Affine Co-ordinates in F_p
- 256 bit, Bit Serial Montgomery Multiplier
- Extended Euclidean Algorithm

- Point Addition:- Inversion, 3 Multiplications, 6 Additions
- Point Doubling:- Inversion, 4 Multiplications, 4 Additions

Design Complexity

Algorithm	Area	Clock	Throughput	Thpt/Area
	(CLBs)	(MHz)	(Mbps)	(bps/CLB)
AES	3,259	27.86	324	99417
SHA-512	2,468	40.02	506	205024
RSA - 1024	8,064	51.84	0.051	6.32
ECC-256A	2,718	19.19	0.00873	3.21
ECC-256J	1,353	20.45	0.00439	3.24

- Jacobian co-ordinates in F_p without conversion.
- Does not include cost of conversion to Affine.
- Point Addition:- 16 Multiplications, 7 Additions
- Point Doubling:- 10 Multiplications, 4 Additions

Design Complexity

Algorithm	Area (CLBs)	Clock (MHz)	Throughput (Mbps)	Thpt/Area (bps/CLB)
AES	3,259	27.86	324	99417
SHA-512	2,468	40.02	506	205024
RSA - 1024	8,064	51.84	0.051	6.32
ECC-256A	2,718	19.19	0.00873	3.21
ECC-256J	1,353	20.45	0.00439	3.24
Tate-256	8,438	34.74	0.01868	2.21

- Millers Algorithm
- Jacobian Co-ordinates for Point Addition and Doubling
- Security Multiplier $k = 4$

- Karatsuba's Method for Multiplication in F_{p^4}
- Includes final Modular Exponentiation

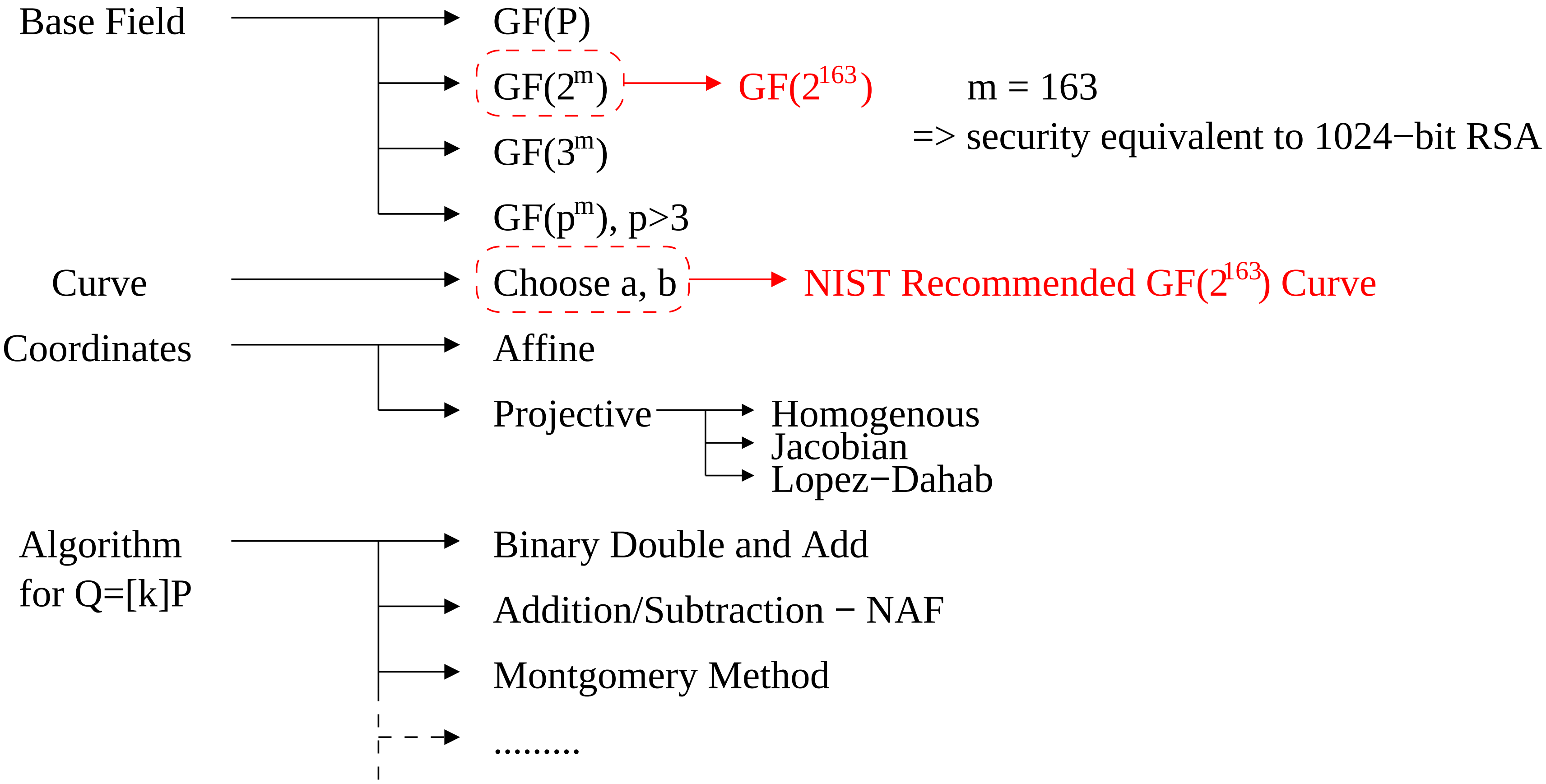
Summary

- Most Area demanding is the RSA algorithm, due to the large 1024 bit key size. (Note Security Level)
- Most Efficient in terms of throughput per CLB is Hash algorithm.
- Mathematical complexity of ECC results in least efficient designs. (Note similar throughput per clb figure for Jacobian and Affine).
- Virtex-E 2000 has 19,200 CLBs and is suitable for implementing all of these algorithms.

Power Consumption & EC Design Choices

- What is the effect of the EC design choices on the Power and Energy consumption of Hardware implementation?
- FPGA platform used
- (FPGAs are not suitable for Low Power implementations)
- Difference between implementations not Absolute value

EC Design Choices



Base Field

GF(P)

GF(2^m) → GF(2^{163})

GF(3^m)

GF(p^m), p>3

m = 163

=> security equivalent to 1024−bit RSA

Curve

Choose a, b → NIST Recommended GF(2^{163}) Curve

Coordinates

Affine

Projective

Homogenous

Jacobian

Lopez−Dahab

Algorithm for Q=[k]P

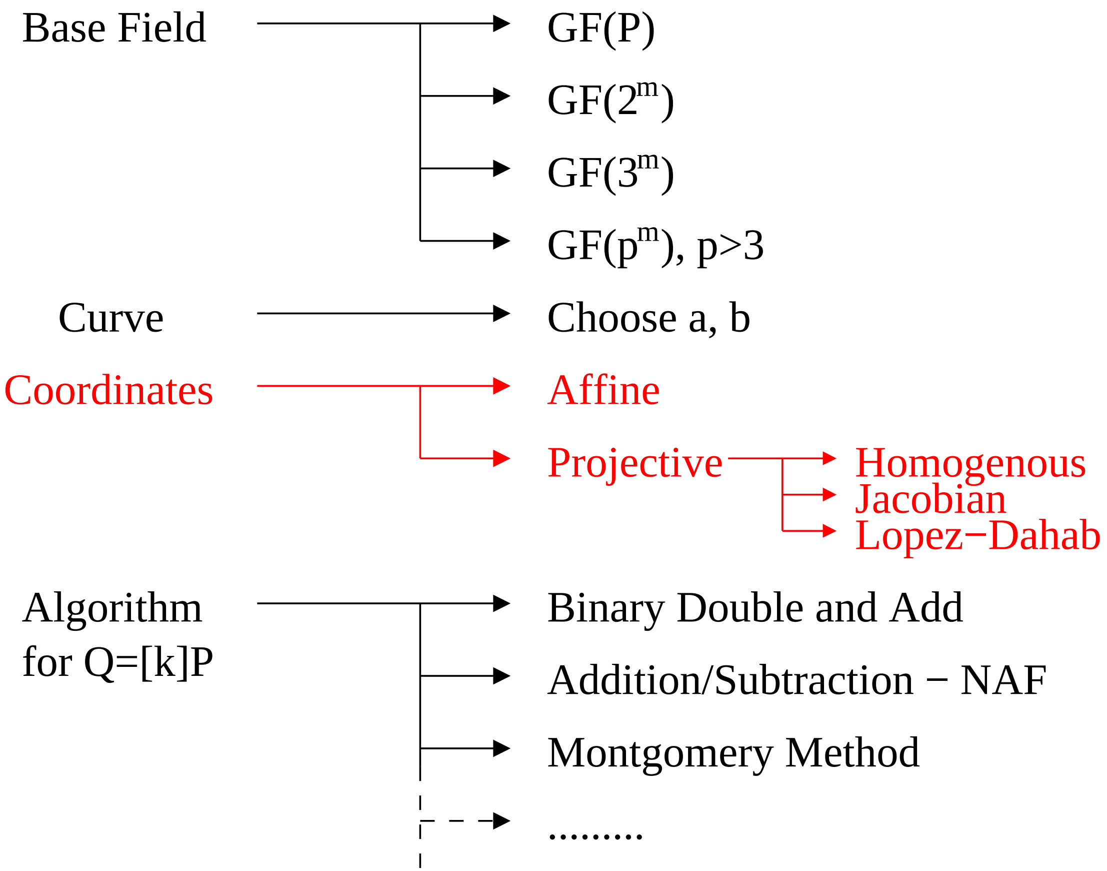
Binary Double and Add

Addition/Subtraction − NAF

Montgomery Method

....

EC Design Choices



Base Field

- GF(P)
- GF(2^m)
- GF(3^m)
- GF(p^m), p>3

Curve

- Choose a, b

Coordinates

- Affine
- Projective
  - Homogenous
  - Jacobian
  - Lopez−Dahab

Algorithm for Q=[k]P

- Binary Double and Add
- Addition/Subtraction − NAF
- Montgomery Method
- ....

EC Coordinate Systems

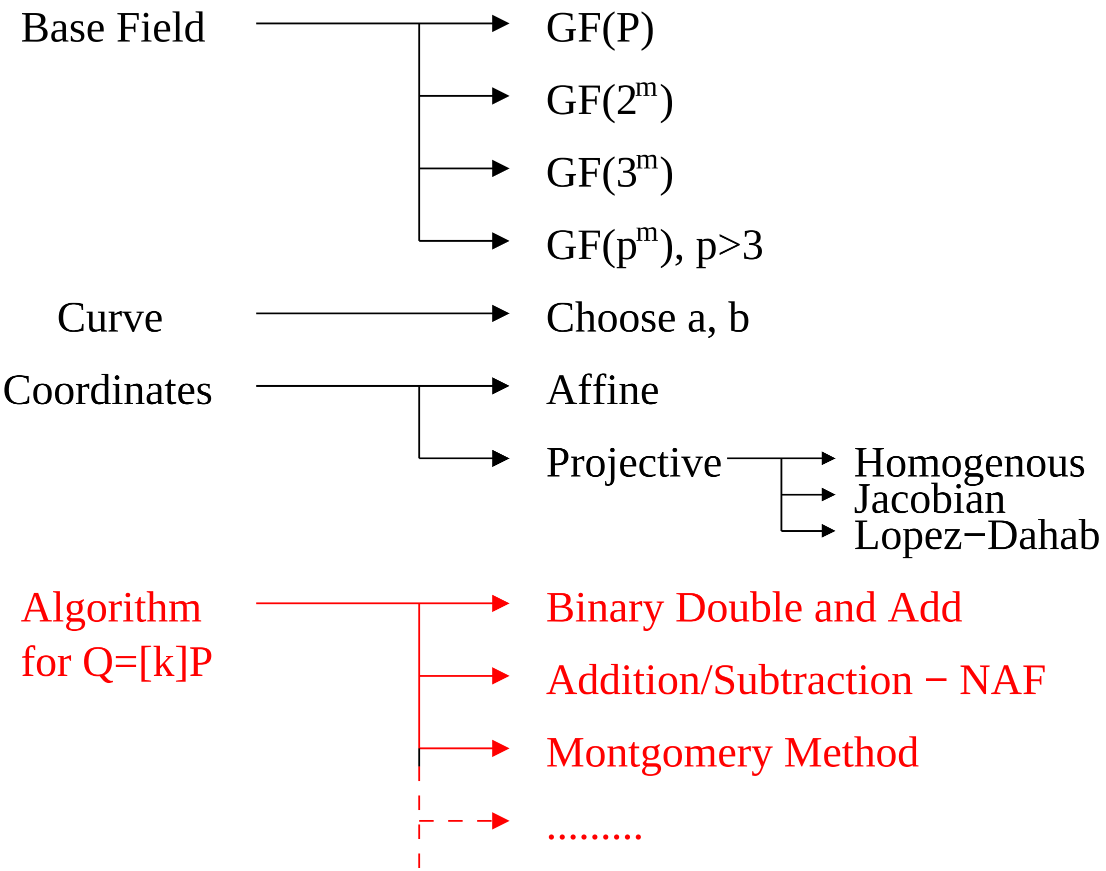
- Affine: $P = (x, y)$
- Projective: $P = (X, Y, Z)$
  - Advantage: Point addition and doubling can be performed without any $GF(2^m)$ division
  - Affine to projective conversion: $(x, y) \rightarrow (x, y, 1)$
  - Generally converted back to affine for transmission
- Two types of projective coordinates used in this work:
  - **Jacobian:** $(X, Y, Z) \rightarrow (\frac{X}{Z^2}, \frac{Y}{Z^3})$
  - **Lopez-Dahab:** $(X, Y, Z) \rightarrow (\frac{X}{Z}, \frac{Y}{Z^2})$

Cost of Point Operations

Affine	Addition:	$1M + 1D + 1S$
	Doubling:	$1M + 1D + 1S$
Jacobian	Addition:	$10M + 4S$
	Doubling:	$5M + 5S$
	Conversion:	$3M + 1D + 1S$
	Conversion*:	$12M + 163S$
Lopez-Dahab	Addition:	$8M + 5S$
	Doubling:	$4M + 5S$
	Conversion:	$2M + 1D + 1S$
	Conversion*:	$11M + 163S$

- $* =$ Conversion with no divider

EC Design Choices



EC Point Scalar Multiplication Algorithm Cost

- Binary Double and Add:
  - $N_{Binary} = (m-1)N_{double} + (\frac{m}{2} - 1)N_{add}$
- Addition/Subtraction – NAF
  - $N_{NAF} = (m-1)N_{double} + (\frac{m}{3} - 1)N_{add}$
- Montgomery Method:
  - $N_{Montgomery} = N_{double} + (m-1)N_{loop} + N_{computey}$

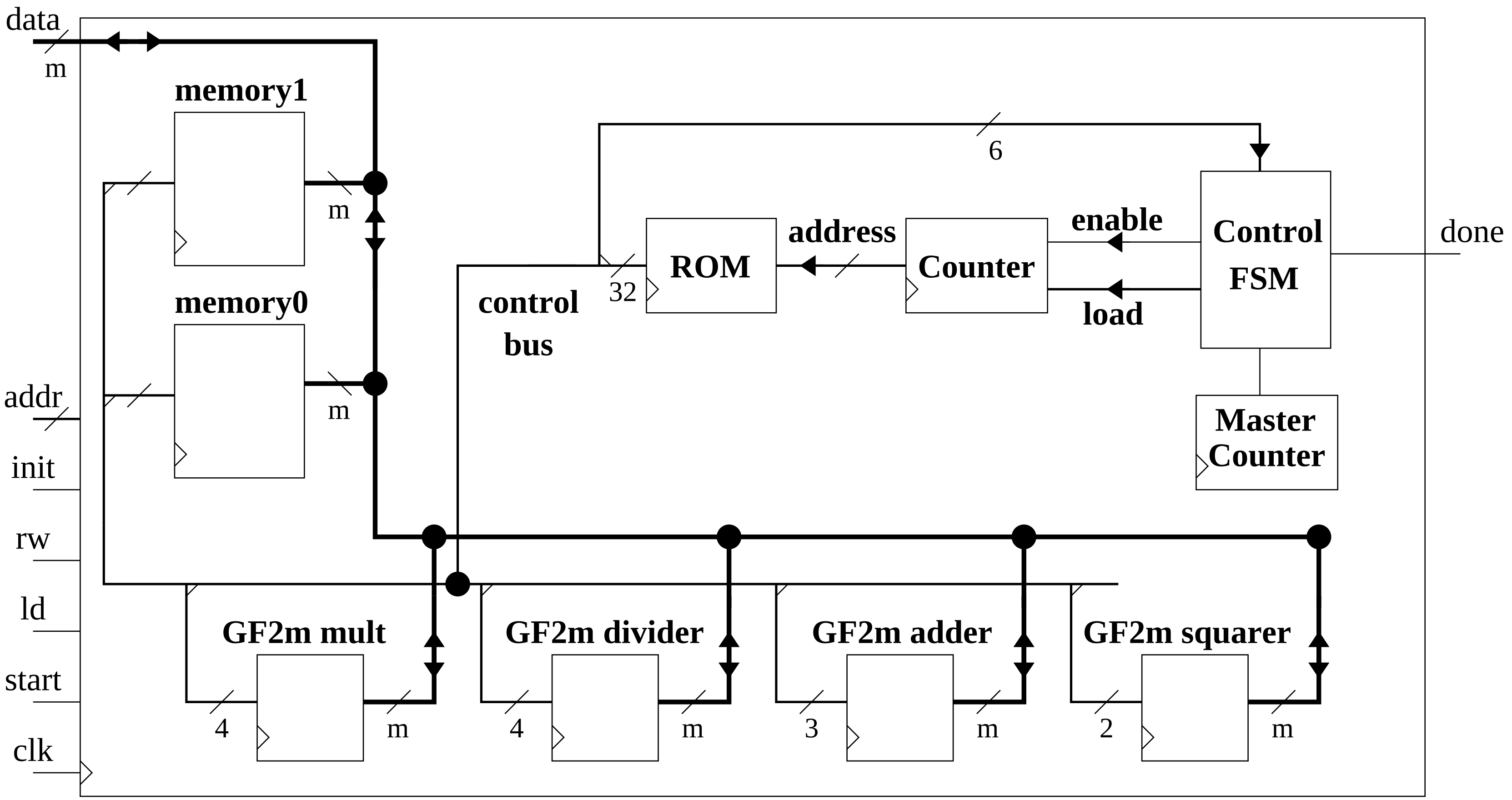
$GF(2^m)$ Hardware Architectures

Operation	Architecture	Clock Cycles
Addition	m XOR Gates	Combinational
Multiplication	Digit-Serial, Digit size d	$n = \lceil \frac{m}{d} \rceil$
Squaring	Bit-Parallel AND-XOR network	Combinational
Division	Extended Euclidean Algorithm	$2m$

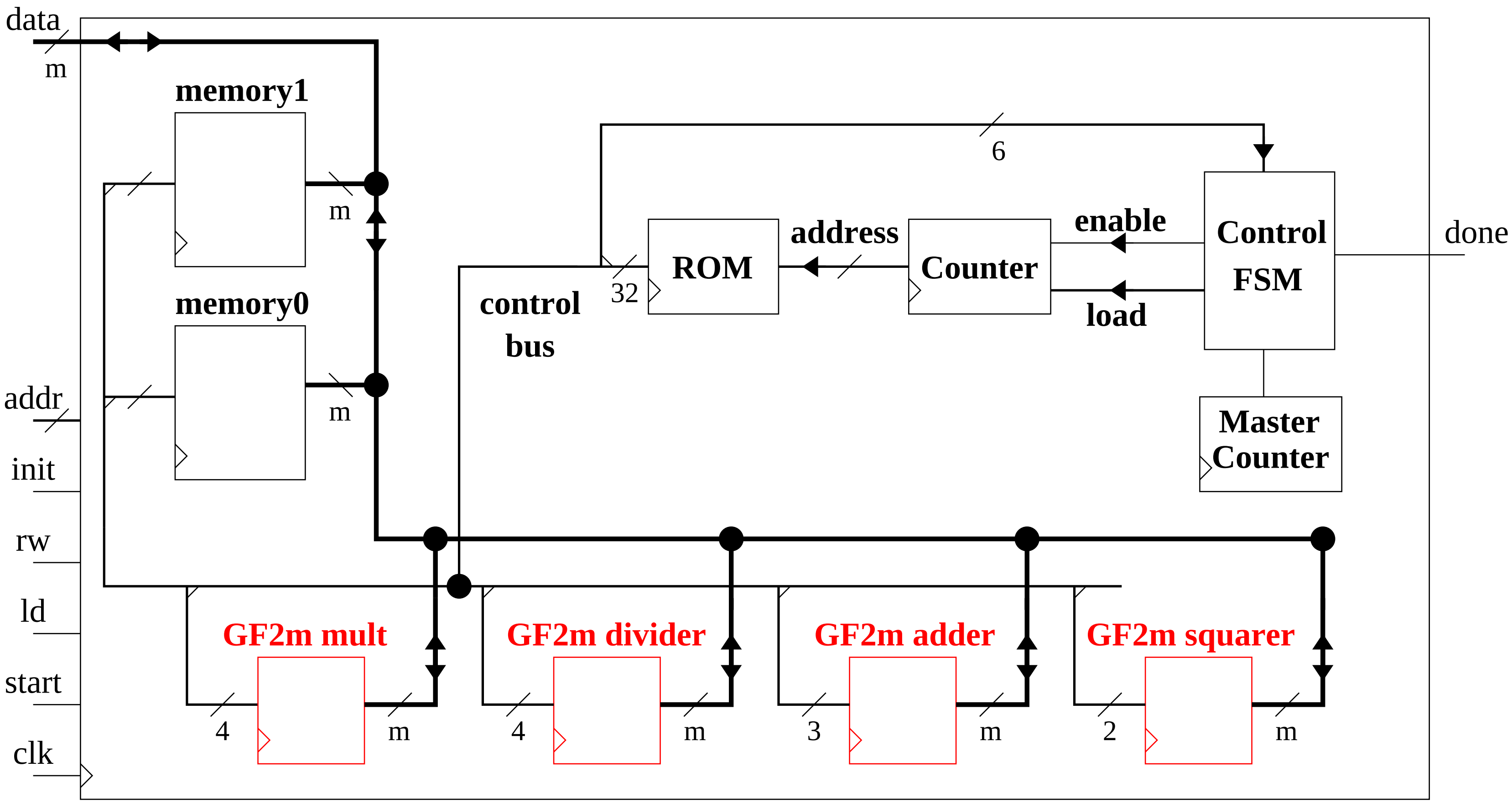
Power Comparision

- This work studies the effect of coordinate and algorithm choice on the power and energy consumption of an elliptic curve processor
- Coordinates investigated:
  - Affine, Jacobian, Lopez-Dahab
- Algorithms investigated:
  - Binary Double and Add
  - Addition/Subtraction – NAF
  - Montgomery Method

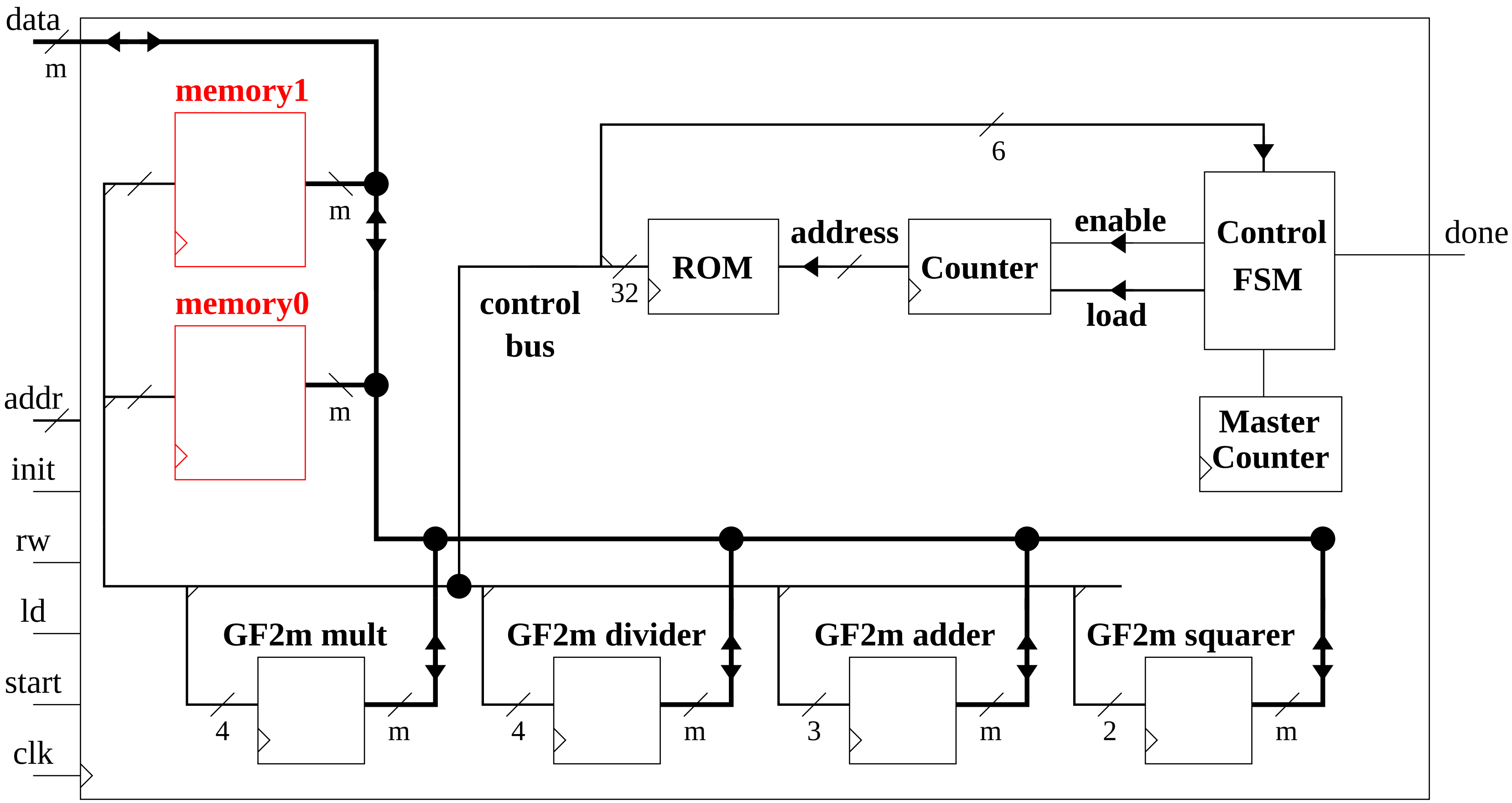
$GF(2^m)$ Elliptic Curve Processor



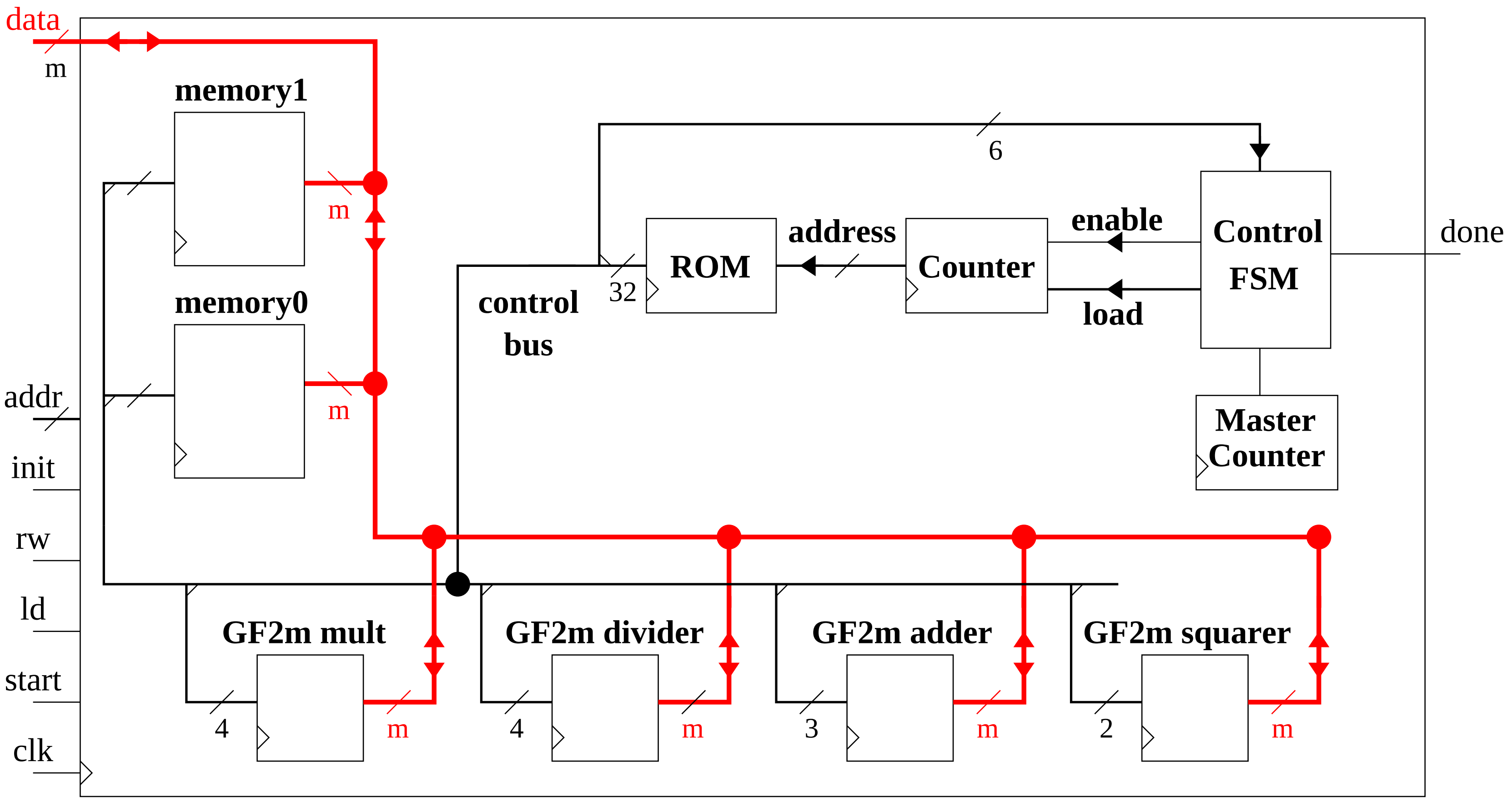
$GF(2^m)$ Elliptic Curve Processor



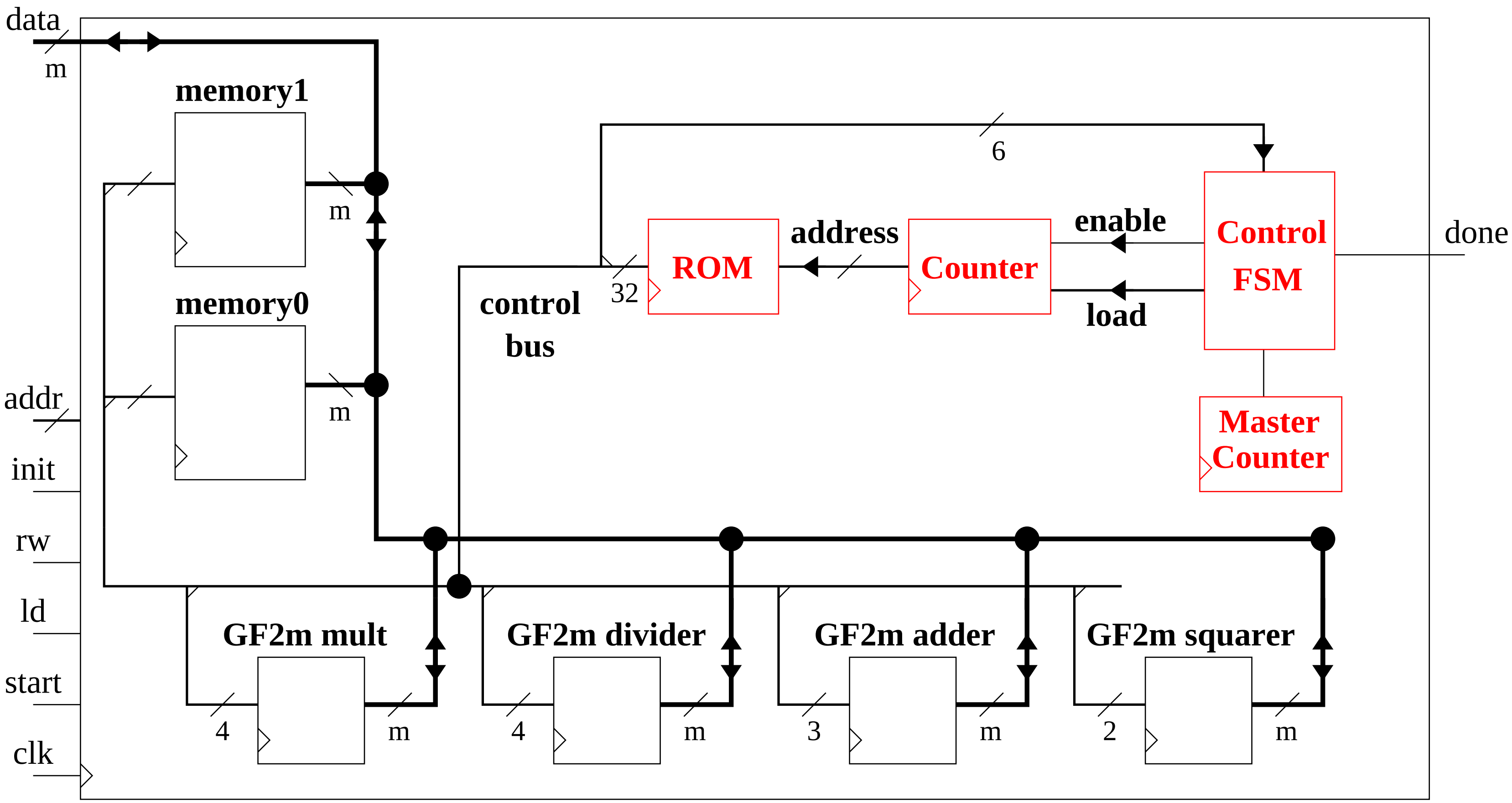
$GF(2^m)$ Elliptic Curve Processor



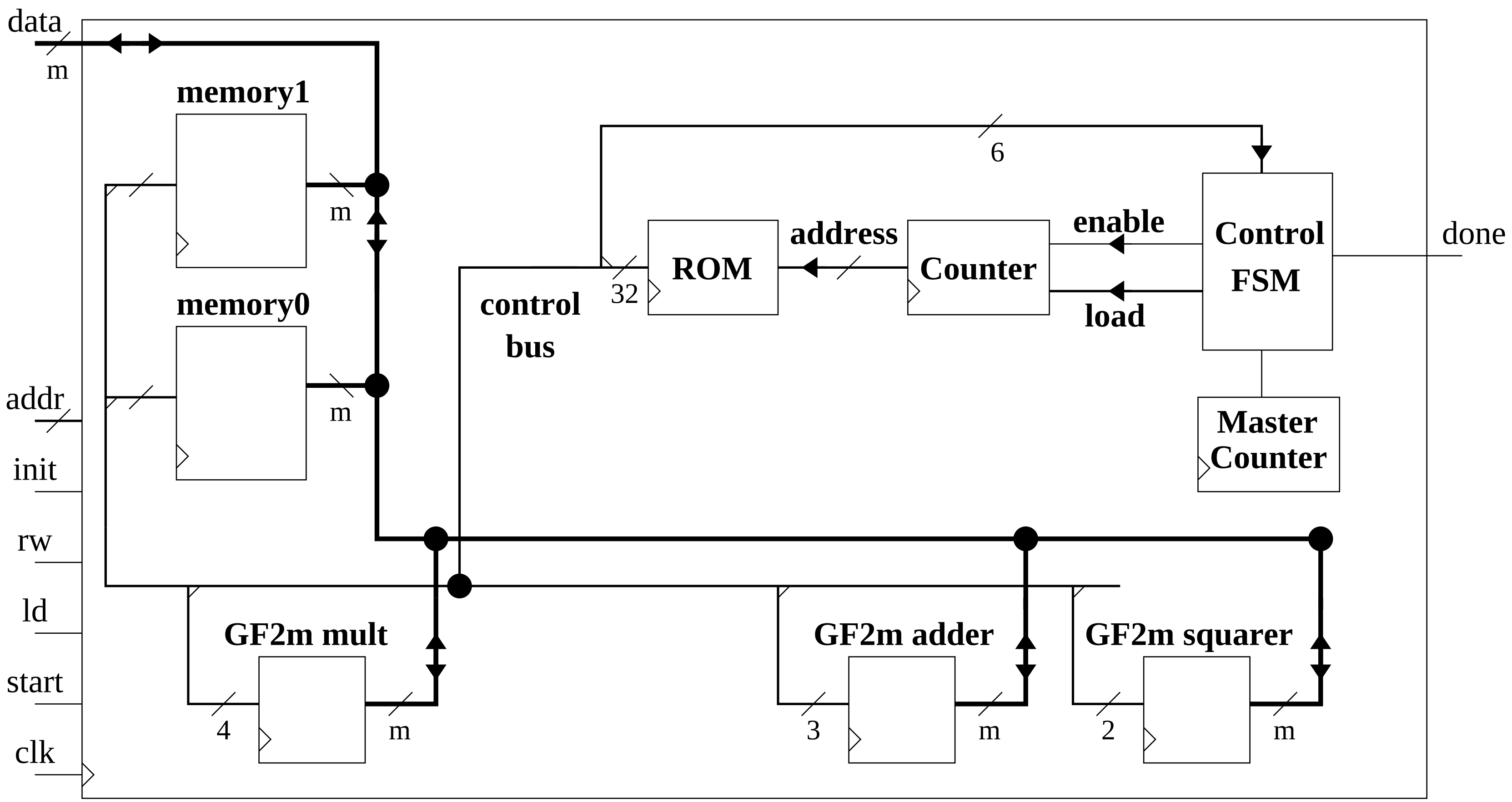
$GF(2^m)$ Elliptic Curve Processor



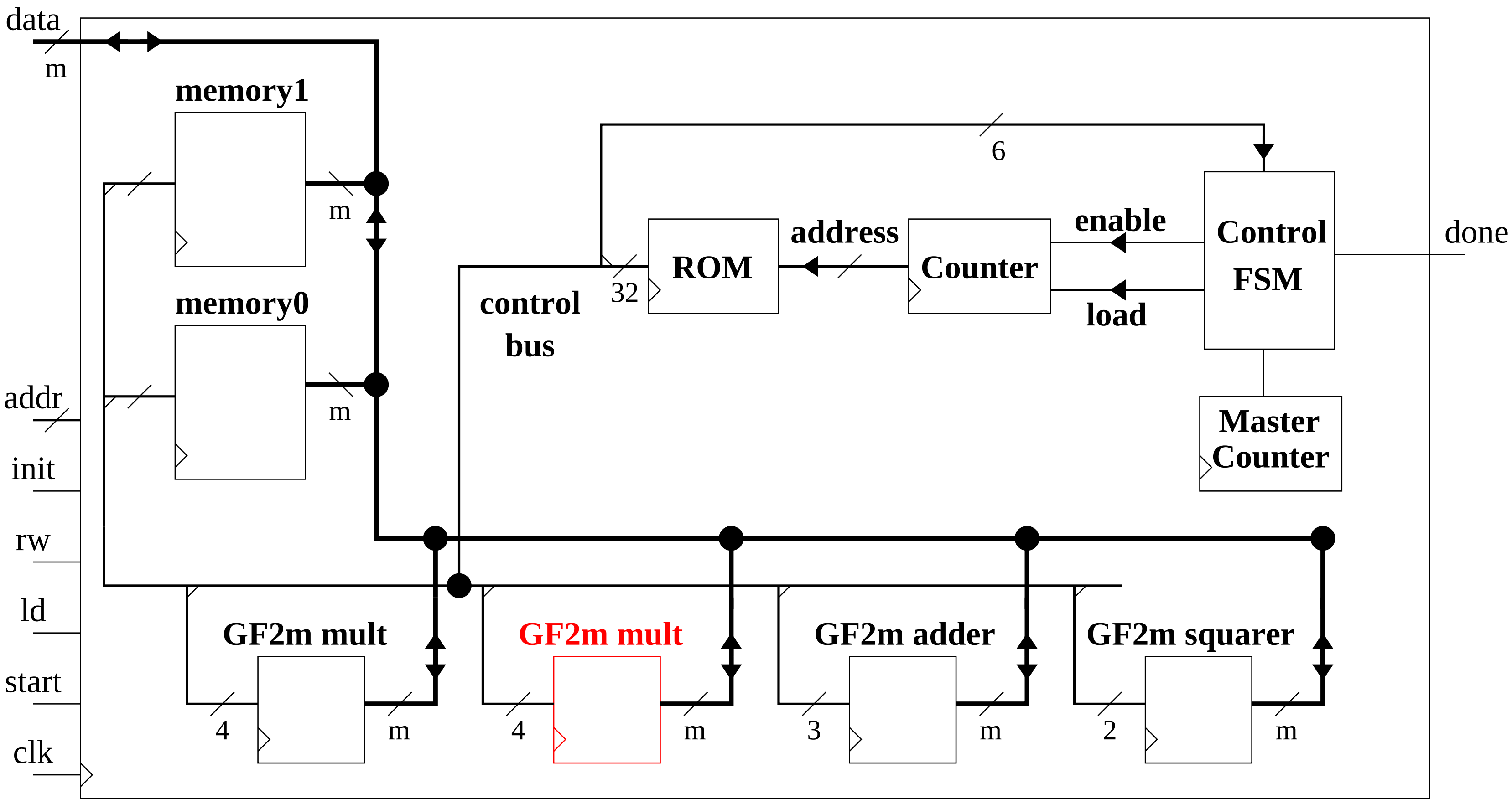
$GF(2^m)$ Elliptic Curve Processor



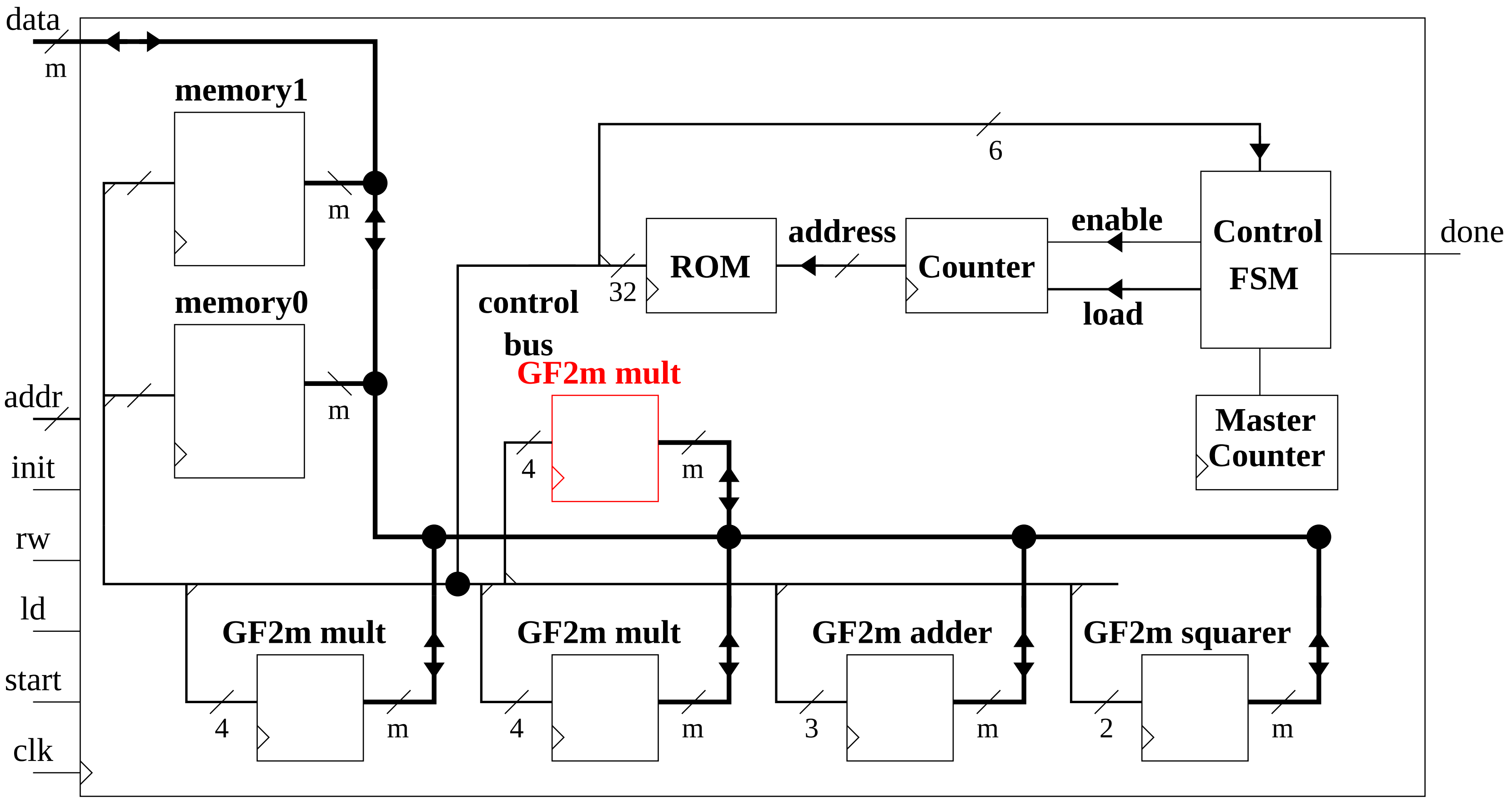
$GF(2^m)$ Elliptic Curve Processor



$GF(2^m)$ Elliptic Curve Processor



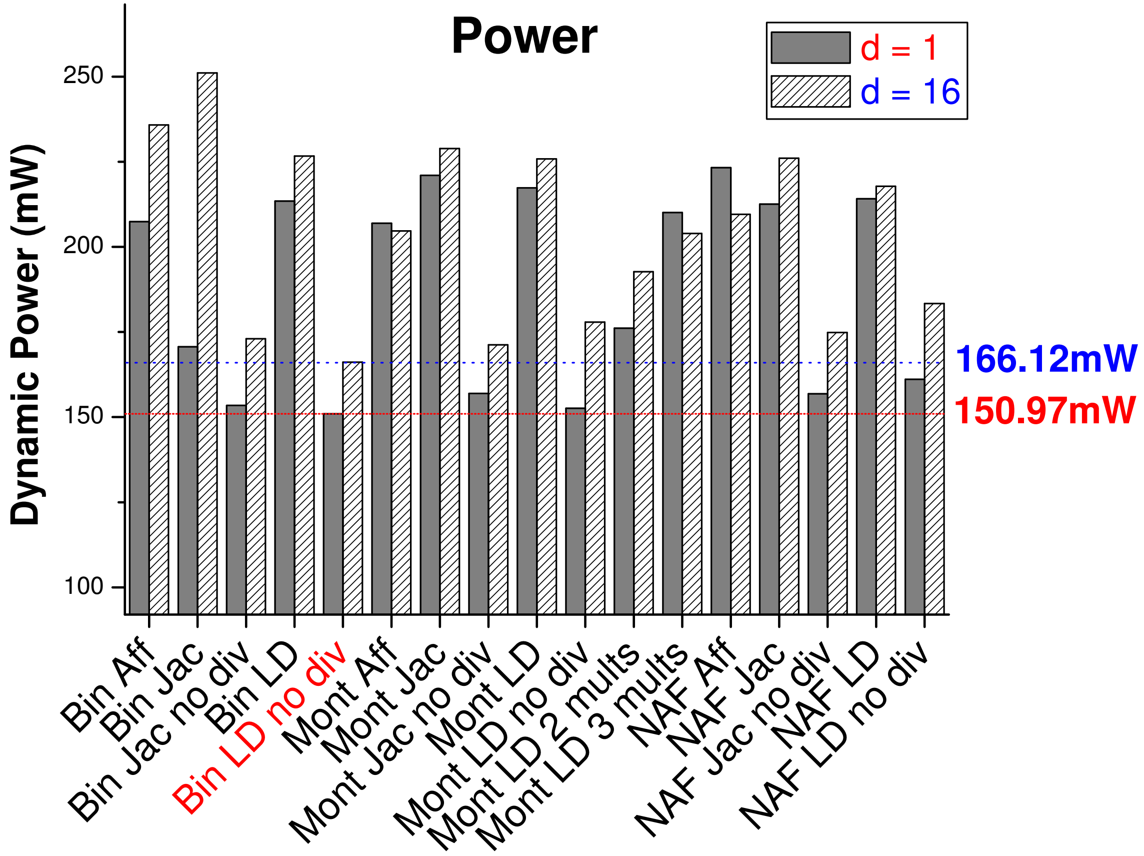
$GF(2^m)$ Elliptic Curve Processor



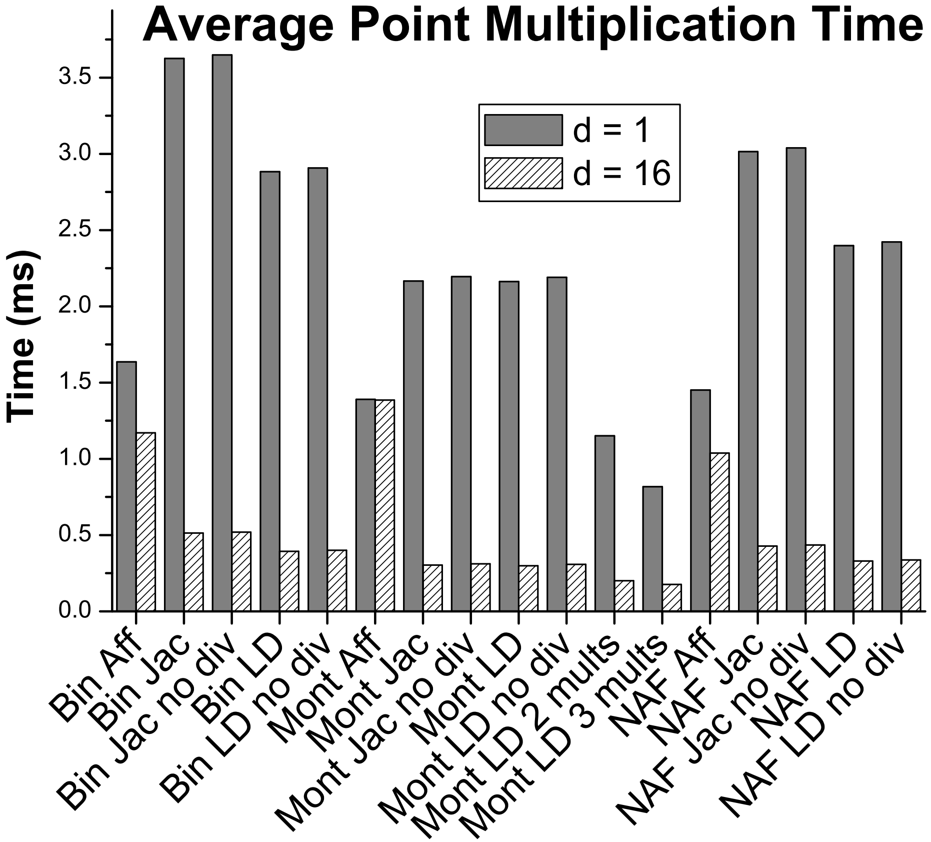
Implementation Results

- Target technology: Xilinx Spartan 3L – xc3s1000l
  - Low Power FPGA
  - Hibernate mode
- Two digit sizes of $GF(2^m)$ multiplier used:
  - d = 1: area $\approx$ 3000 LUTs
  - d = 16: area $\approx$ 5100 LUTs
  - Divider area $\approx$ 1100 LUTs
- Minimum PPR Clock Frequency Reported $= 80MHz$
- Quiescent Power $= 92mW$

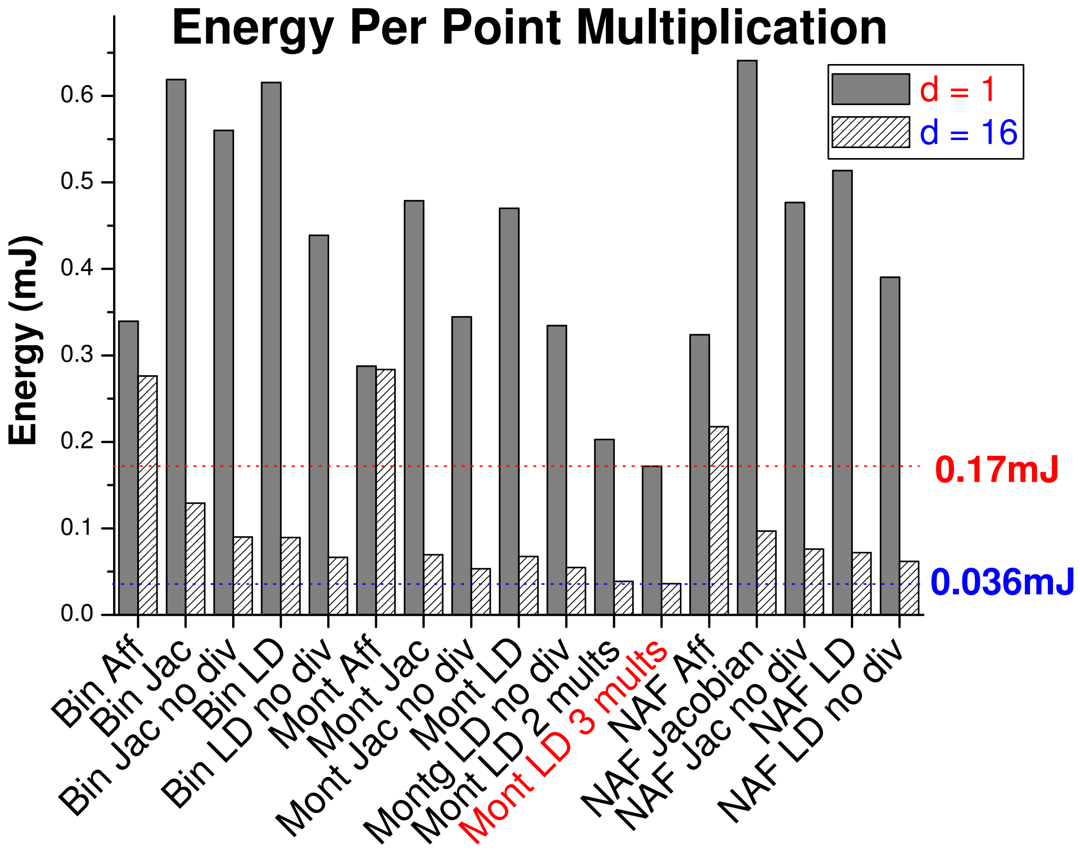
Power Dissipation



Point Multiplication Time



Energy Per Point Multiplication



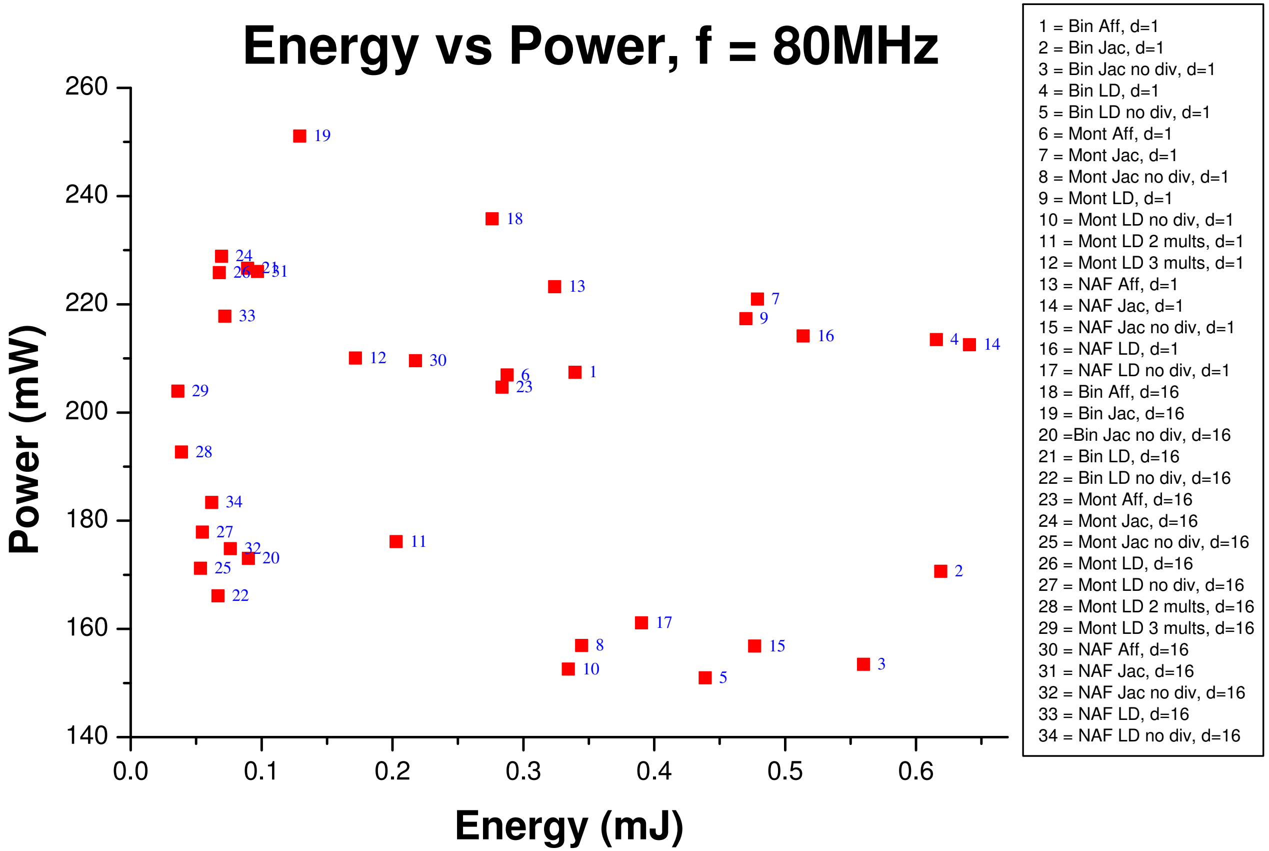
Summary

- **Minimum Power:** $150.97mW$
  - Binary Lopez-Dahab no divider, $d = 1$
  - $f_{CLK} = 80MHz$, Calculation time $= 2.87ms$
  - Energy $= 0.43mJ$
- **Minimum Energy:** $0.036mJ$
  - Montgomery Lopez-Dahab 3 mults, $d = 16$
  - $f_{CLK} = 80MHz$, Calculation time $= 0.18ms$
  - Power $= 203.95mW$

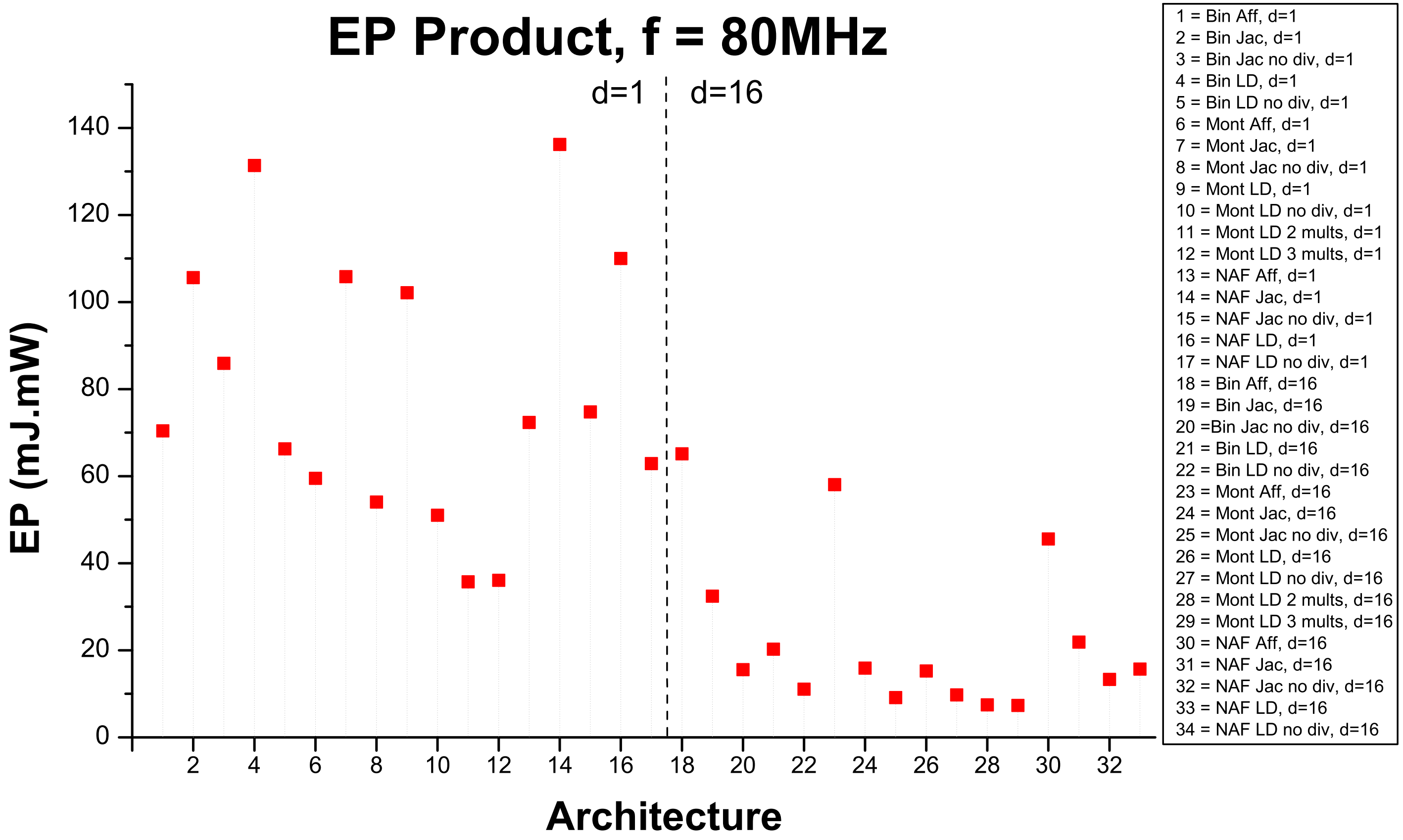
What is the "best" set of choices?

- What is most important, power or energy?
- Need a metric to compare designs...
- Power **and** energy requirements will determine battery size, therefore try to minimise both
- Look at Energy vs. Power

Energy vs. Power



Energy–Power Product



EP Optimised Choices

	Montgomery LD **three mults** $d = 16$	**Montgomery LD** **two mults,** $d = 16$
EP Product:	$7.4 mJ.mW$	$7.5 mJ.mW$
Power:	$203.95 mW$	$192.7 mW$
Energy:	$0.036 mJ$	$0.039 mJ$
Time:	$177 \mu s$	$201 \mu s$
Area:	$9393 LUTs$	$6711 LUTs$
AT Product:	1.66	1.35

EP Optimised Choices

	Montgomery LD **three mults** $d = 16$	**Montgomery LD** **two mults,** $d = 16$
EP Product:	$7.4mJ.mW$	$7.5mJ.mW$
Power:	$203.95mW$	$192.7mW$
Energy:	$0.036mJ$	$0.039mJ$
Time:	$177\mu s$	$201\mu s$
Area:	$9393LUTs$	$6711LUTs$
AT Product:	1.66	1.35

EP Optimised Choices

	Montgomery LD **three mults** $d = 16$	**Montgomery LD** **two mults,** $d = 16$
EP Product:	$7.4mJ.mW$	$7.5mJ.mW$
Power:	$203.95mW$	$192.7mW$
Energy:	$0.036mJ$	$0.039mJ$
Time:	$177\mu s$	$201\mu s$
Area:	$9393LUTs$	$6711LUTs$
AT Product:	1.66	1.35

EP Optimised Choices

	Montgomery LD **three mults** $d = 16$	**Montgomery LD** **two mults,** $d = 16$
EP Product:	$7.4mJ.mW$	$7.5mJ.mW$
Power:	$203.95mW$	$192.7mW$
Energy:	$0.036mJ$	$0.039mJ$
Time:	$177\mu s$	$201\mu s$
Area:	$9393LUTs$	$6711LUTs$
AT Product:	1.66	1.35

Conclusion

- EC Implementation choices **Do** have an effect on the Complexity of Final Design
- Many Metrics Available to Determine Best Design
- Designer/Vendor will always choose Metric that put their design in the best light
- and their competitors in a bad light