ECC on Your Fingertips: A Single Instruction Approach for Lightweight ECC Design in GF(p)

Debapriya Basu Roy, Poulami Das and Debdeep Mukhopadhyay

June 19, 2015

- With the increase of sensitive information in the Internet, security is becoming an important aspect.

- With the increase of sensitive information in the Internet, security is becoming an important aspect.
- Cryptography provides a method for securing and authenticating the transmission of information over insecure channels.

- With the increase of sensitive information in the Internet, security is becoming an important aspect.
- Cryptography provides a method for securing and authenticating the transmission of information over insecure channels.
- Elliptic Curve Cryptography (ECC): Computationally intensive.

- With the increase of sensitive information in the Internet, security is becoming an important aspect.
- Cryptography provides a method for securing and authenticating the transmission of information over insecure channels.
- Elliptic Curve Cryptography (ECC): Computationally intensive.
- Can be accelerated by designing FPGA based accelerator.

Elliptic Curve Cryptography

- Public Key Cryptography Algorithm

Elliptic Curve Cryptography

- Public Key Cryptography Algorithm
- based on ECDLP

Elliptic Curve Cryptography

- Public Key Cryptography Algorithm
- based on ECDLP
- Scalar multiplication: The most important operation of ECC

Elliptic Curve Cryptography

- Public Key Cryptography Algorithm
- based on ECDLP
- Scalar multiplication: The most important operation of ECC
- Involves point doubling and point addition operations

ECC Scalar Multiplication

Double-and-Add Algorithm

Algorithm 1: Double-and-Add Algorithm

```
Data: Point $P$ and scalar $k = k_{m-1}, k_{m-2}, k_{m-3} \ldots k_2, k_1, k_0$, where $k_{m-1} = 1$
Result: $Q = kP$
$Q = P$
for $i = m - 2$ to 0 do
    $Q = 2Q$ (Point Doubling)
    if $k_i$=1 then
        $Q = Q + P$ (Point Addition)
    end
end
```

- Each point doubling requires 8 field multiplications
- Each point addition requires 11 field multiplications

One Instruction Set Computing

- Single instruction is executed repeatedly

One Instruction Set Computing

- Single instruction is executed repeatedly
- Possible to create a Turing complete machine

One Instruction Set Computing

- Single instruction is executed repeatedly
- Possible to create a Turing complete machine
- Idea of using OISC for cryptographic applications was first proposed in CHES 2010 by David Naccache

One Instruction Set Computing

- Single instruction is executed repeatedly
- Possible to create a Turing complete machine
- Idea of using OISC for cryptographic applications was first proposed in CHES 2010 by David Naccache
- OISC has been also used to support homomorphic encryption architecture

Example of OISC

- ADDLEQ (Add the operands and branch if the answer is less than or equal to zero)

Example of OISC

- ADDLEQ (Add the operands and branch if the answer is less than or equal to zero)
- SUBLEQ (Subtract the operands and branch if the answer is less than or equal to zero)

Example of OISC

- ADDLEQ (Add the operands and branch if the answer is less than or equal to zero)
- SUBLEQ (Subtract the operands and branch if the answer is less than or equal to zero)
- SBN (Subtract the operands and branch if the answer is less than zero)

Example of OISC

- ADDLEQ (Add the operands and branch if the answer is less than or equal to zero)
- SUBLEQ (Subtract the operands and branch if the answer is less than or equal to zero)
- SBN (Subtract the operands and branch if the answer is less than zero)
- RSSB (Reverse subtract and skip if borrow)

Example of OISC

- ADDLEQ (Add the operands and branch if the answer is less than or equal to zero)
- SUBLEQ (Subtract the operands and branch if the answer is less than or equal to zero)
- SBN (Subtract the operands and branch if the answer is less than zero)
- RSSB (Reverse subtract and skip if borrow)
- SBNZ (Subtract the operands and branch if the answer is non-zero)

SBN Execution

Table: SBN and Addition using SBN

Code 1.1 SBN: Subtract and Branch if negative	Code 1.2 Addition using SBN
`SBN A,B,C`	`ADD C,A,B`
`D.Mem[A]=D.Mem[A]-D.Mem[B]`	`1. SBN X,X,2`
`if(D.Mem[A]<0)`	`2. SBN X,A,3`
`jump to C`	`3. SBN X,B,4`
`else`	`4. SBN C,C,5`
`jump to next instruction`	`5. SBN C,X,6`

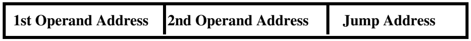

Figure: Instruction format of OISC

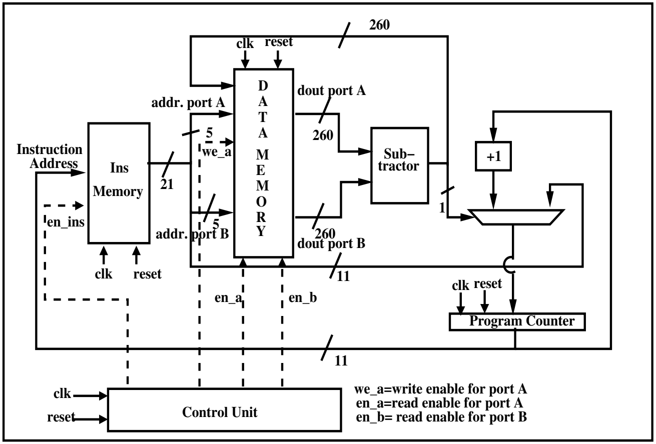

Figure: Architecture of SBN-OISC

Switching-off Memory Write-back

Code 1.3 Field Addition using Traditional SBN	Code 1.4 Field Addition using our Modification
ADD$_p$ C,A,B 1. SBN X,X,2 2. SBN X,A,3 3. SBN X,B,4 4. SBN C,C,5 5. SBN C,X,6 6. SBN R,R,7 7. SBN R,X,8 8. SBN R,P,12 9. SBN X,X,10 10. SBN X,R,11 11. SBN C,X,12 12. SBN ...	ADD$_p$ C,A,B 1. SBN$_w$ X,X,2 2. SBN$_w$ X,A,3 3. SBN$_w$ X,B,4 4. SBN$_w$ C,C,5 5. SBN$_w$ C,X,6 6. SBN$_{nw}$ C,P,8 7. SBN$_w$ C,P,8 8. SBN ...

Operations Practically beyond SBN

- Right Shift operation

Operations Practically beyond SBN

- Right Shift operation
- Shifting Key Register

Operations Practically beyond SBN

- Right Shift operation
- Shifting Key Register
- Multiplication

Different SBN Instructions

Table: Different Variant of SBN Instruction

Instruction	Memory Write-back	Multiplier reset	Key-shift	Right-shift
$SBN_{w\overline{mulks}\overline{rs}}$	✓	x	x	x
$SBN_{nw\overline{mulks}\overline{rs}}$	x	x	x	x
$SBN_{nw\overline{mul}ks\overline{rs}}$	x	x	✓	x
$SBN_{w\overline{mulks}rs}$	✓	x	x	✓
$SBN_{nwmul\overline{ks}\overline{rs}}$	x	✓	x	x

Different SBN Instructions

Table: Different Variant of SBN Instruction

Instruction	Memory Write-back	Multiplier reset	Key-shift	Right-shift
$SBN_{w\overline{mulks}\overline{rs}}$	✓	×	×	×
$SBN_{nw\overline{mulks}\overline{rs}}$	×	×	×	×
$SBN_{nw\overline{mul}ks\overline{rs}}$	×	×	✓	×
$SBN_{w\overline{mulks}rs}$	✓	×	×	✓
$SBN_{nwmul\overline{ks}\overline{rs}}$	×	✓	×	×

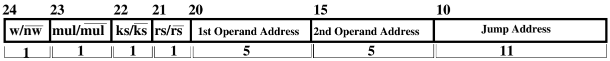

Figure: Modified Instruction format of SBN-OISC

Modular Reduction Algorithm

```
Algorithm 2: Fast Modular Reduction Algorithm for NIST P-256 Curve
Data: 512 bit product $C$ represented as $C = C_{15}||C_{14}||\ldots||C_0$, where each $C_i$ is a 32 bit integer, $i \in \{0, 15\}$
Result: $P = C \bmod \text{P-256}$
$T = (C_7||C_6||C_5||C_4||C_3||C_2||C_1||C_0)$
$S_1 = (C_{15}||C_{14}||C_{13}||C_{12}||C_{11}||0||0||0)$
$S_2 = (0||C_{15}||C_{14}||C_{13}||C_{12}||0||0||0)$
$S_3 = (C_{15}||C_{14}||0||0||0||C_{10}||C_9||C_8)$
$S_4 = (C_8||C_{13}||C_{15}||C_{14}||C_{13}||C_{11}||C_{10}||C_9)$
$D_1 = (C_{10}||C_8||0||0||0||C_{13}||C_{12}||C_{11})$
$D_2 = (C_{11}||C_9||0||0||C_{15}||C_{14}||C_{13}||C_{12})$
$D_3 = (C_{12}||0||C_{10}||C_9||C_8||C_{15}||C_{14}||C_{13})$
$D_4 = (C_{13}||0||C_{11}||C_{10}||C_9||0||C_{15}||C_{14})$
$P = T + 2S_1 + 2S_2 + S_3 + S_4 - D_1 - D_2 - D_3 - D_4 \bmod \text{P-256}$
```

Architecture of the Multiplier

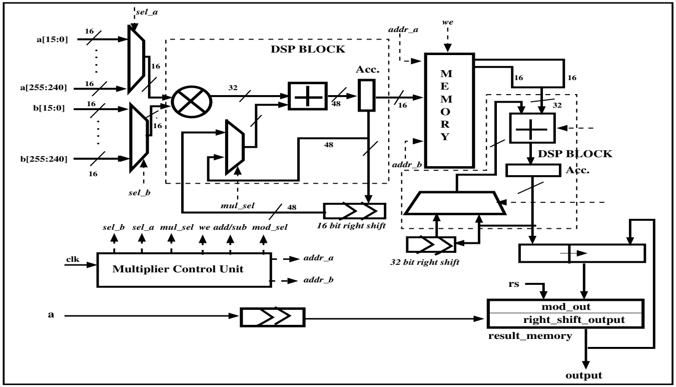

Figure: Architecture of Lightweight Field Multiplier

Full Processor Architecture

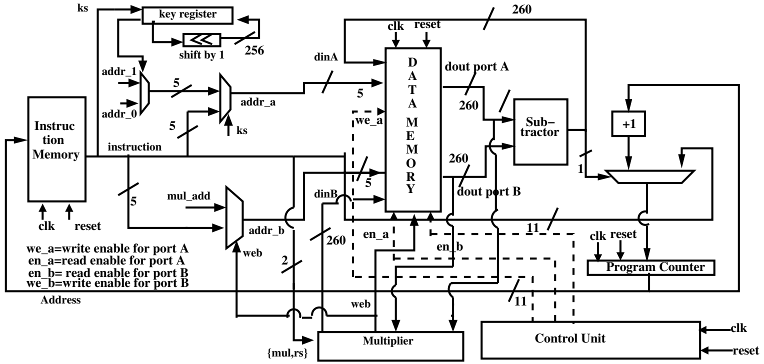

Figure: ECC SBN-OISC Processor Architecture

Table: Area and Timing Performance of the proposed ECC SBN-OISC processor

Platform	Freq. (MHz)	Slices	LUTs	Flip-Flops	DSP for ALU	DSP for Multiplier	Block-RAM	Time (ms)
Virtex-5	171.5	81	212	35	6	2	22	11.1
Spartan-6	156.25	72	193	35	6	2	24	12.2

Table: Comparison of ECC SBN-OISC Processor with Existing Designs

Reference	Slices	MULTs	BRAMs	Freq (MHz)	Latency (ms)	FPGA
Micro-ECC P-256 16 bit [1]	773	1	3	210	10.02	Virtex-II Pro
Micro-ECC P-256 32 bit [1]	1158	4	3	210	4.52	Virtex-II Pro
[2] 16 bit any prime curve	1832	2	9	108.20	29.83	Virtex-II Pro
[2] 32 bit any prime curve	2085	7	9	68.17	15.76	Virtex-II Pro
[3] P-256	1715	32 (DSP)	11	490	.62	Virtex-4
[4] P-256	221	1	3	Not shown	Not shown	Spartan-6
Present Work, P-256	81	8(DSP)	22	171.5	11.1	Virtex-5
Present Work, P-256	72	8(DSP)	24	156.25	12.2	Spartan-6

Scope of Improvement

- Reducing the latency of the design

Scope of Improvement

- Reducing the latency of the design
- Reducing the block ram usage

Conclusion

- Merged two design strategies to create an extremely light-weight ECC crypto-processor for scalar multiplication in NIST P-256 curve.

Conclusion

- Merged two design strategies to create an extremely light-weight ECC crypto-processor for scalar multiplication in NIST P-256 curve.
- First strategy: using single instruction processor

Conclusion

- Merged two design strategies to create an extremely light-weight ECC crypto-processor for scalar multiplication in NIST P-256 curve.
- First strategy: using single instruction processor
- Second strategy: optimum usage of FPGA hard-IPs

Conclusion

- Merged two design strategies to create an extremely light-weight ECC crypto-processor for scalar multiplication in NIST P-256 curve.
- First strategy: using single instruction processor
- Second strategy: optimum usage of FPGA hard-IPs
- First implementation to reduce the slice count to less than 100.

Michal Varchola, Tim Güneysu, and Oliver Mischke.
MicroECC: A Lightweight Reconfigurable Elliptic Curve Crypto-processor.
In *2011 International Conference on Reconfigurable Computing and FPGAs, ReConFig 2011, Cancun, Mexico, November 30 - December 2, 2011*, pages 204–210, 2011.

Jo Vliegen, Nele Mentens, Jan Genoe, An Braeken, Serge Kubera, Abdellah Touhafi, and Ingrid Verbauwhede.
A Compact FPGA-based Architecture for Elliptic Curve Cryptography over Prime Fields, booktitle = 21st IEEE International Conference on Application-specific Systems Architectures and Processors, ASAP 2010, Rennes, France, 7-9 July 2010, pages = 313–316, year = 2010, crossref = DBLP:conf/asap/2010, url = http://dx.doi.org/10.1109/ASAP.2010.5540977, doi = 10.1109/ASAP.2010.5540977, timestamp = Thu, 30 Sep 2010 10:42:37 +0200, biburl = http://dblp.uni-trier.de/rec/bib/conf/asap/VliegenMGBKTV10, bibsource = dblp computer science bibliography, http://dblp.org.

Tim Gneysu and Christof Paar.
Ultra High Performance ECC over NIST Primes on Commercial FPGAs.
In *In Proceedings of CHES*, pages 62–78, 2008.

Benedikt Driessen, Tim Güneysu, Elif Bilge Kavun, Oliver Mischke, Christof Paar, and Thomas Pöppelmann.
IPSecco: A lightweight and reconfigurable IPSec core.
In *2012 International Conference on Reconfigurable Computing and FPGAs, ReConFig 2012, Cancun, Mexico, December 5-7, 2012*, pages 1–7, 2012.