Fast Hardware Architectures for Supersingular Isogeny Diffie-Hellman Key Exchange on FPGA

Paper by: Brian Koziel (Texas Instruments) [corresponding author, kozielbrian@gmail.com], Reza Azarderakhsh (Florida Atlantic University), and Mehran Mozaffari-Kermani (Rochester Institute of Technology)

Indocrypt 2016
Kolkata, India

Outline

Introduction

- Supersingular isogeny Diffie-Hellman (SIDH) as a strong quantum-resistant cryptographic primitive for NIST's PQC standardization
  - Originally presented by Jao and De Feo at PQCrypto 2011
  - Provides small keys, forward secrecy and a Diffie-Hellman key exchange
  - Can be visualized as moving from elliptic curve to elliptic curve
- This work analyzes SIDH implementations targeted for FPGA
  - We show that SIDH can utilize a large amount of parallelism

Contributions

- We present the first constant-time SIDH implementation with projective formulas.
  - Our implementation is approximately 50% faster than the only other SIDH implementation targeting FPGA.
- We target parameters that feature 83-bit and 124-bit quantum security levels.
- We provide a new approach to parallelizing isogeny evaluations that speed-up large-degree isogeny computations by over a factor of 1.5.

Supersingular Isogeny Diffie-Hellman

- Proposed by David Jao and Luca De Feo in 2011[1]
- Isogeny-based cryptography over supersingular elliptic curves is believed to be quantum-resistant
  - Ordinary curves feature a commutative endomorphism ring for which a quantum subexponential attack is known
- An isogeny is defined as a non-constant rational map $\phi : E_1 \rightarrow E_2$ such that the null point is preserved
- Diffie-Hellmanesque $\rightarrow$Alice and Bob can each perform a secret isogeny, exchange them, perform a secret isogeny on the exchanged curves, and agree on a common, secret curve
- Classical and quantum security is approximately $\sqrt[4]{p}$ and $\sqrt[6]{p}$, respectively.
  - Based on the difficulty of computing isogenies between supersingular elliptic curves, i.e. given $E, \phi(E)$, find ϕ

[1] Jao, D., De Feo, L.: *Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies*. PQCrypto 2011: 19-34.

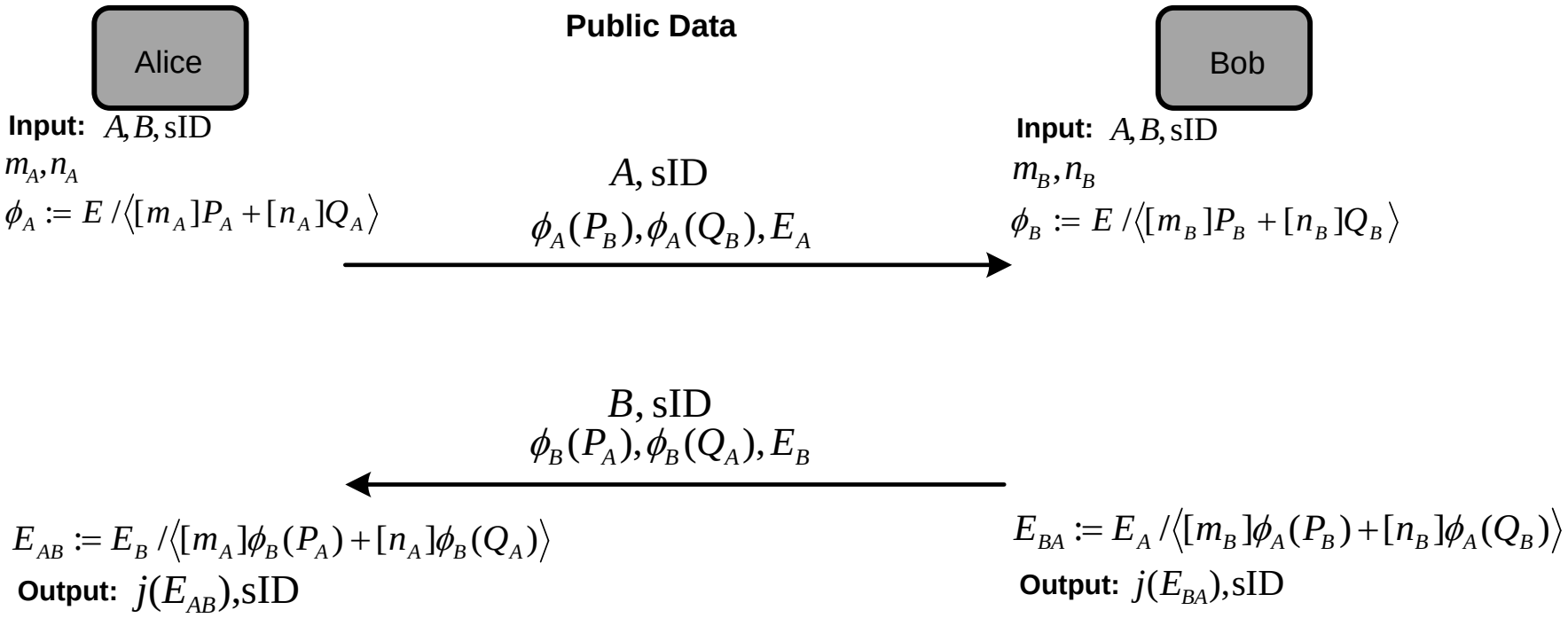
SIDH Overview

- In the SIDH protocol, special "isogeny-efficient" primes are selected such that small degree isogenies can be efficiently computed.
- Public Parameters:
  - Smooth Isogeny Prime - $p = \ell_A^{e_A} \ell_B^{e_B} f \pm 1$, where ℓ_A and ℓ_B are small primes, e_A and e_B are positive integers, and f is a small cofactor to make the number prime
  - Starting Supersingular Elliptic Curve, $E_0/\mathbb{F}_{p^2}$
  - Torsion bases $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$ over $E_0[\ell_A^{e_A}]$ and $E_0[\ell_B^{e_B}]$, respectively

SIDH Overview

- Each round is broken into computing a double point multiplication, $R = mP + nQ$, where m and n are secret scalars, and using R as a secret kernel for an isogeny, $\phi : E \to E/\langle R \rangle$.
  - $\phi_A : E \to E/\langle m_A P_A + n_A Q_A \rangle = E_A$ for Alice and
    $\phi_B : E \to E/\langle m_B P_B + n_B Q_B \rangle = E_B$ for Bob
- After the first round, Alice sends $\{E_A, \phi_A(P_B), \phi_A(Q_B)\}$ and Bob sends $\{E_B, \phi_B(P_A), \phi_B(Q_A)\}$
- After the second round, Alice and Bob have isomorphic curves, so the j-invariant can be used as a shared secret key.
  - $\phi'_A : E_B \to E_B/\langle m_A \phi_B(P_A) + n_A \phi_B(Q_A) \rangle = E_{AB}$ for Alice and
    $\phi'_B : E_A \to E_A/\langle m_B \phi_A(P_B) + n_B \phi_A(Q_B) \rangle = E_{BA}$ for Bob
  - $j(E_{AB}) = j(E_{BA})$

SIDH Protocol



SIDH Computations

- All arithmetic is performed on Kummer line of Montgomery curve for efficiency
  - Represent points as $(x, y) \to (X : Z)$, where $x = X/Z$.
- 3-point Montgomery differential ladder for double point multiplication if $m = 1$[1]
  - Computes $P + [t]Q$ at each iteration
- Fast Montgomery arithmetic to compute isogenies of degree 2, 3, and 4
  - Affine formulas that require an inversion for each isogeny computation[1]
  - Projective formulas that require an inversion at the end of the round[2]

[1] De Feo, L., Jao, D., Plut, J.: *Towards Quantum-resistant Cryptosystems from Supersingular Elliptic Curve Isogenies*. Journal of Mathematical Cryptography, 2014, 8(3):209-247.
[2] Costello, C., Longa, P., Naehrig, M.: *Efficient Algorithms for Supersingular Isogeny Diffie-Hellman*. CRYPTO 2016: 572-601

SIDH Protocol Parameters

- Take advantage of base field and trace-zero[1] to optimize round 1 double point multiplication
  - Find a point $P_A \in E_0(\mathbb{F}_p)[\ell_A^{e_A}]$ as $[f\ell_B^{e_B}](z, \sqrt{z^3+z})$, where z is the smallest positive integer such that $\sqrt{z^3+z} \in \mathbb{F}_p$ and P_A has order $\ell_A^{e_A}$.
  - $Q_A = \tau(P_A)$,
    $\tau: E_0(\mathbb{F}_{p^2}) \to E_0(\mathbb{F}_{p^2}), (x+0i, y+0i) \to (-x+0i, 0+iy)$

Table: SIDH Public Parameters

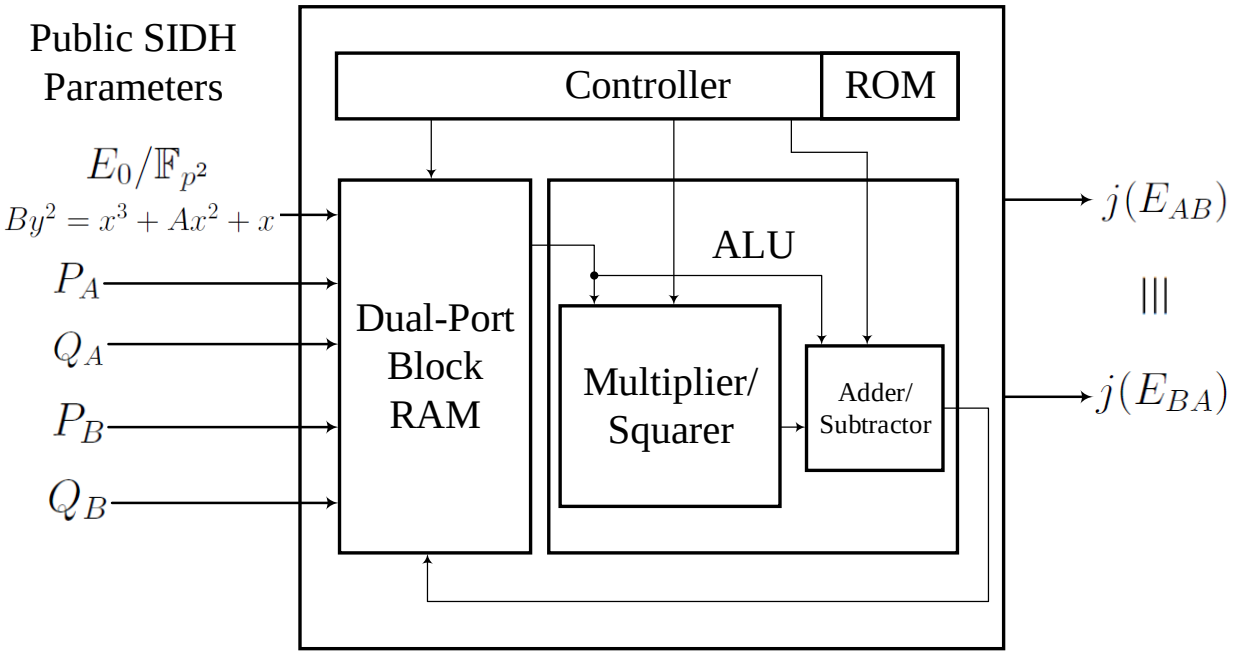
Curve: $E_0/\mathbb{F}_{p^2}: y^2 = x^3 + x$			
Prime	Classical/Quantum Security (bits)	P_A	P_B
$p_{503} = 2^{250}3^{159} - 1$	125/83	$z = 14$	$z = 6$
$p_{751} = 2^{372}3^{239} - 1$	186/124	$z = 11$	$z = 6$

[1] Costello, C., Longa, P., Naehrig, M.: *Efficient Algorithms for Supersingular Isogeny Diffie-Hellman.* CRYPTO 2016: 572-601

SIDH Accelerator Architecture

- FPGA Target: Perform SIDH as fast as possible
- General Architecture
  - Separate multiplier and addition units
  - Dual-port block RAM to store registers (up to 256 registers)
  - Controller and program ROM to carry out operations
  - Special ROM for optimal isogeny strategy splits

SIDH Accelerator Architecture



Addition Unit

- Performs finite-field addition and subtraction
  - i.e. $C = A + B$, where $A, B, C \in \mathbb{F}_p$ and then a conditional subtraction $C = C - p$ if $C > p$.
- Utilized Xilinx IP's for 256 bit addition/subtractions and cascaded for 503 bit and 751 bit operations
- OP1 mux: port A, add_result, or mul_result
- OP2 mux: port B, zero, or p
- p_{503} addition requires 2 cycles, p_{751} addition requires 3 cycles

Multiplier Unit

- Montgomery multiplication for efficient multiplication
- Utilize systolic architecture presented in [1], but originally made by [2]
- Interleaved high-radix Montgomery multiplier over radix 2^{16}
  - Initially resets all processing elements (PE) sequentially using a reset pump
  - Systolic array performs $S_{i+1} = (S_i + q_i\overline{m_j})/2^k + a_i b_j$ for inputs A and B, where $S_0 = 0$ and $q_i = (S_i) \bmod 2^k$
  - Utilize FPGA DSP48 blocks to compute 16x16bit multiplications
  - Requires $3m+7$ cycles to complete and can be interleaved every $2m+3$ cycles
- p_{503} multiplication requires 99 cycles, p_{751} multiplication requires 144 cycles

[1] McIvor, C., McLoone, M., and McCanny, J. V.: *High-Radix Systolic Modular Multiplication on Reconfigurable Hardware*. IEEE International Conference on Field-Programmable Technology, 2005, 13-18.
[2] Koziel, B., Azarderakhsh, R., Kermani, M.M., Jao, D..: *Post-Quantum Cryptography on FPGA Based on Isogenies on Elliptic Curves*. Cryptology ePrint Archive, Report 2016/672

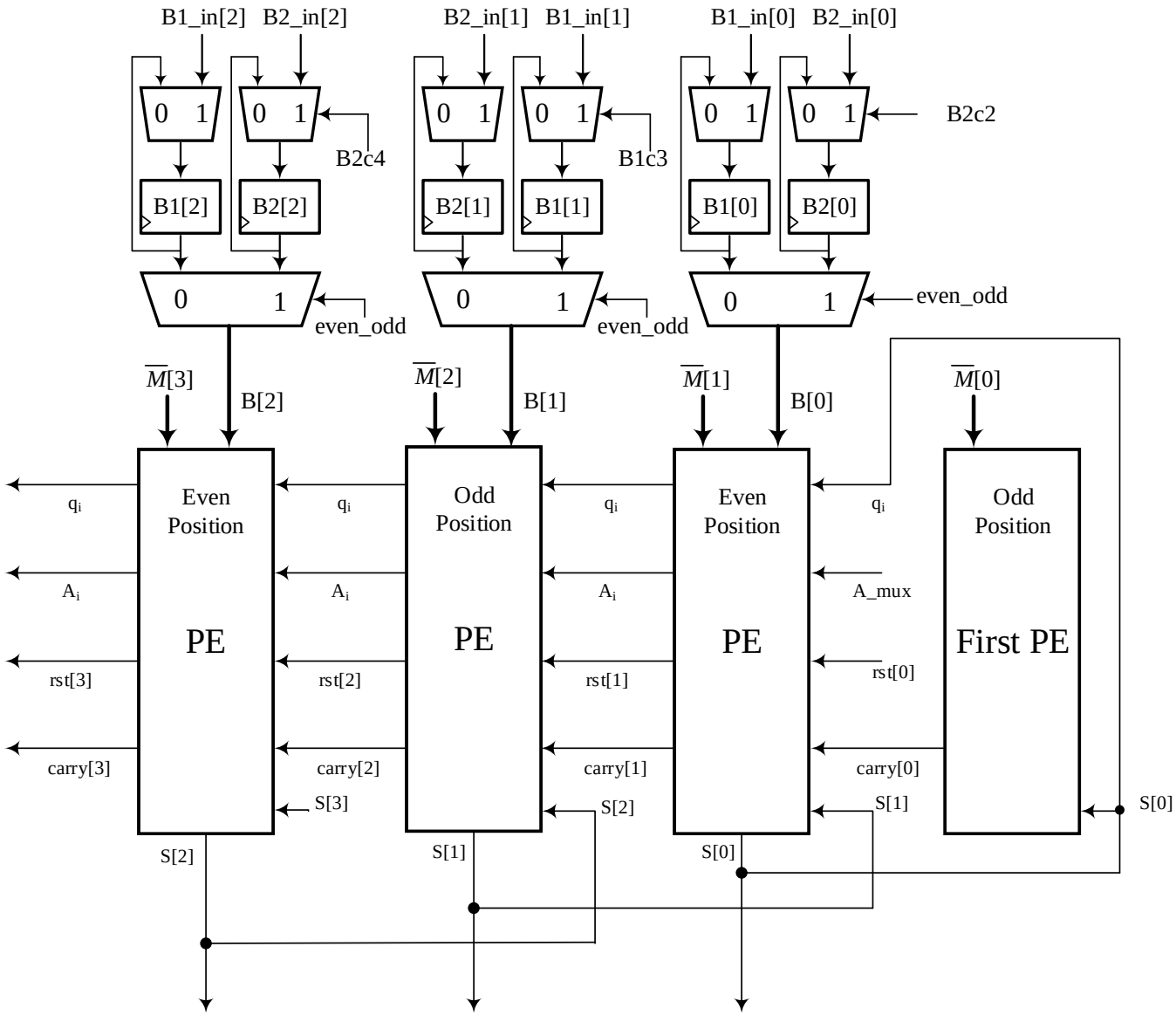
High-Radix Montgomery Multiplication Algorithm[1]

Input: $M = p$, $M' = -M^{-1} \bmod p$,
$A = \sum_{i=0}^{m+2} (2^k)^i a_i, a_i \in \{0, 1 \ldots 2^k - 1\}, a_{m+2} = 0$
$B = \sum_{i=0}^{m+1} (2^k)^i b_i, b_i \in \{0, 1 \ldots 2^k - 1\},$
$\overline{M} = (M' \bmod 2^k) M = \sum_{i=0}^{m+1} (2^k)^i m_i$
$A, B < 2\overline{M}$; $4\overline{M} < 2^{km}$, $R = 2^{\lceil \log_2 p \rceil}$

1. $S_0 = 0$
2. **for** $i = 0$ **to** $m+2$ **do**
   3. $q_i = (S_i) \bmod 2^k$
   4. $S_{i+1} = (S_i + q_i \overline{M})/2^k + a_i B$
5. **end for**
6. **return** $S_{m+3} = A \times B \times R^{-1} \bmod M$

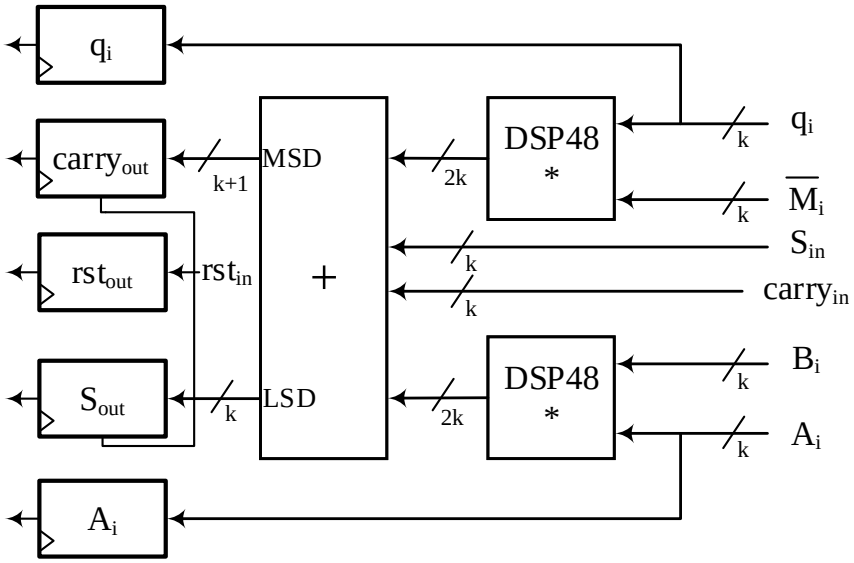
[1] Orup, H.: *Simplifying Quotient Determination in High-Radix Modular Multiplication*. ARITH '95, 1995, 193-9.

Interleaved Systolic Montgomery Multiplier: Systolic Architecture[1]

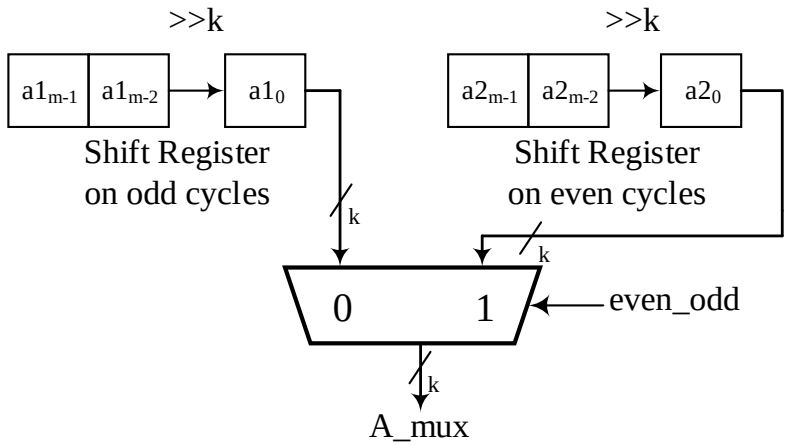


[1] Koziel, B., Azarderakhsh, R., Kermani, M.M., Jao, D..: *Post-Quantum Cryptography on FPGA Based on Isogenies on Elliptic Curves*. Cryptology ePrint Archive, Report 2016/672

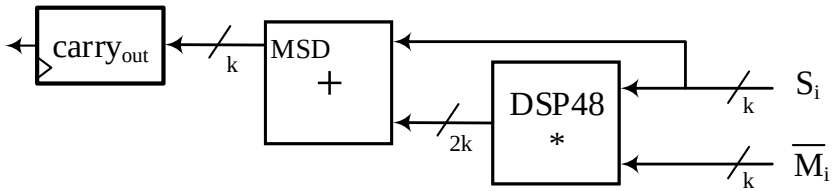
Interleaved Systolic Montgomery Multiplier: Components[1]



Processing Element



A_mux to shift in odd and even A values



First Processing Element

[1] Koziel, B., Azarderakhsh, R., Kermani, M.M., Jao, D..: *Post-Quantum Cryptography on FPGA Based on Isogenies on Elliptic Curves*. Cryptology ePrint Archive, Report 2016/672

Multiplier Unit

- Two multiplications can be performed simultaneously by systolic architecture
  - Creates idea of even_odd cycles to differentiate multiplications
- Since multiplication is the bottleneck of SIDH, replicate base multiplier to perform even more multiplications in parallel
- No dedicated squaring unit, but squaring is rarely used outside of inversion

General Scheduling Methodology

- Greedy algorithm to assemble code describing group level operations (i.e. isogeny evaluation of degree 4)
  - Priority is pushed to the bottleneck multiplications
  - Every instruction specifies what the controller, multiplier, and adder should do
  - Cycle-by-cycle granularity
- A single memory unit can store, write, or read on a single cycle
- Only one adder/subtractor is used for additions and reductions
- Replicated multiplier unit cycles through even_odd multipliers
- Reschedule incorrect even_odd multiplication starting cycles by incurring pipeline stalls to make it fit
  - Rescheduling algorithm from [1]

[1] Koziel, B., Azarderakhsh, R., Kermani, M.M., Jao, D..: *Post-Quantum Cryptography on FPGA Based on Isogenies on Elliptic Curves*. Cryptology ePrint Archive, Report 2016/672

Extension Field Arithmetic

- Extension field arithmetic in $\mathbb{F}_{p^2}$ is a combination of multiple operations in $\mathbb{F}_p$.
- With the choice of $\ell_A = 2$, $x^2 + 1$ is an irreducible polynomial
- Let $A = (A_0, A_1)$, $B = (B_0, B_1) \in \mathbb{F}_{p^2}$.The results of operations in $\mathbb{F}_{p^2}$ are $C = (C_0, C_1)$

$$
\begin{aligned}
A + B &= (A_0 + B_0, A_1 + B_1) \\
A - B &= (A_0 - B_0, A_1 - B_1) \\
A \times B &= (A_0 B_0 - A_1 B_1, (A_0 + A_1), \\
&\quad (B_0 + B_1) - A_0 B_1 - A_1 B_0) \\
A^2 &= ((A_0 + A_1)(A_0 - A_1), 2 A_0 A_1) \\
A^{-1} &= (A_0 (A_0^2 + A_1^2)^{-1}, -A_1 (A_0^2 + A_1^2)^{-1})
\end{aligned}
$$

Extension Field Arithmetic Parallelism

- With extra temporary registers, we can perform several $\mathbb{F}_p$ operations in parallel for a single $\mathbb{F}_{p^2}$ operation
  - i.e. 3 $\mathbb{F}_p$ multiplications can be parallelized for a single $\mathbb{F}_{p^2}$ multiplication
  - Combined with group level operations that do not require data dependencies, a high level of parallelization can be achieved
    - Benefit even at 10 parallel multipliers.

$$
\begin{aligned}
A+B &= (A_0+B_0, A_1+B_1)\\
A-B &= (A_0-B_0, A_1-B_1)\\
A\times B &= (A_0B_0-A_1B_1, (A_0+A_1),\\
&\quad (B_0+B_1)-A_0B_1-A_1B_0)\\
A^2 &= ((A_0+A_1)(A_0-A_1), 2A_0A_1)\\
A^{-1} &= (A_0(A_0^2+A_1^2)^{-1}, -A_1(A_0^2+A_1^2)^{-1})
\end{aligned}
$$

Isogeny Evaluation Parallelism

- Isogeny evaluations convert a point from a curve to an isogenous curve after computing an ℓ-isogeny
- An optimal "strategy" involves storing several points that act as pivots to efficiently compute many ℓ-isogenies
- In other software implementations of SIDH, the isogeny evaluations are performed iteratively
- We unroll the loop to increase the amount of parallelism we can achieve
  - Reduces total isogeny computation time by a factor of 1.5
  - Requires many more BRAMs to hold many more instructions

Virtex-7 Results

- Xilinx Vivado 2015.4 to a Xilinx Virtex-7 xc7vx690tffg1157-3 board
- All results were obtained after place-and-route
- Focus on 3-5 replicated dual multipliers to take advantage of parallelism
- *Constant-time implementation*

Virtex-7 Results

Table: Implementation results of SIDH architectures on a Xilinx Virtex-7 FPGA

# Mults	Area			Time		SIDH/s
	# FFs	# DSPs	# BRAMs	Freq (MHz)	Total time (ms)	
p_{503}						
6	26,659	192	40	181.4	20.9	47.8
8	32,541	256	37.5	186.8	19.4	51.5
10	39,446	320	34.5	175.9	19.8	50.5
p_{751}						
6	36,728	282	47	177.3	46.3	21.6
8	46,857	376	45.5	182.1	42.5	23.5
10	56,979	470	44	172.6	42.9	23.3

Comparison to Previous FPGA Implementation

Table: Hardware comparison of SIDH architectures on a Virtex-7 with 3 replicated multipliers

Work		Area			Time	
	Prime (bits)	# FFs	# DSPs	# BRAMs	Latency ($cc \times 10^6$)	Time (ms)
Koziel et al.	511	30,031	192	27	5.967	33.7
This Work	503	26,659	192	40	3.80	20.9

Comparison of Results

Table: Comparison to the software implementations of SIDH over 512-bit keys

Work	Platform	Time (ms)				
		A R1	B R1	A R2	B R2	Total Time
Jao et al.	2.4 GHz Opt.	365	318	363	314	1360
Jao et al.	2.4 GHz Opt.	28.1	28.0	23.3	22.7	102.1
Azarderakhsh et al.	4.0 GHz i7	-	-	-	-	54.0
Koziel et al.	Virtex-7	9.35	8.41	8.53	7.41	33.70
This Work	Virtex-7	4.83	5.25	4.41	4.93	19.42

Comparison of Results

Table: Comparison to software implementations of SIDH over 768-bit keys

Work	Platform	Time (ms)				
		A R1	B R1	A R2	B R2	Total Time
Jao et al.	2.4 GHz Opt.	65.7	54.3	65.6	53.7	239.3
Azarderakhsh et al.	4.0 GHz i7	-	-	-	-	133.7
Costello et al.	3.4 GHz i7	15.0	17.3	13.8	16.8	62.9
This Work	Virtex-7	10.6	11.6	9.5	10.8	42.5

Conclusions

- Efficient implementation of SIDH for FPGA
- First SIDH hardware implementation utilizing projective isogeny formulas
- First constant-time SIDH hardware implementation
- 48% faster than fastest known hardware and software implementations of SIDH
  - Emphasize parallelism in finite-field operations and isogeny evaluations
- As SIDH continues to be pushed, it appears to be a strong solution for NIST's PQC standardization

Thank You!