FPGA Energy Consumption of Post-Quantum Cryptography

Luke Beckwith, Jens-Peter Kaps, Kris Gaj
George Mason University, USA
{lbeckwit, jkaps, kgaj}@gmu.edu

***Abstract*—Current public-key standards are threatened by quantum computing. To combat this threat, NIST is currently evaluating candidates for quantum-secure digital signatures and key exchanges. Several algorithms were recently selected for standardization, and others are being evaluated further for potential standardization in the future. As a part of the evaluation process, candidate algorithms are being investigated for their security and efficiency. This evaluation covers the performance and resource utilization of implementations running in software and hardware. One area that has not received significant effort is evaluating candidates for energy consumption when implemented in hardware. This work helps address this gap by evaluating a number of key encapsulation and digital signature implementations on Artix-7 FPGAs.**

I. Introduction

Current cryptographic standards are built on the difficulty of the integer factorization and discrete logarithm problems. However, these problems can be solved in polynomial time using Shor's algorithm [1]. Due to the upcoming threat of quantum computing to current public-key standards [2], there is an ongoing competition run by NIST to establish new post-quantum public-key standards that are built on hard problems that are not vulnerable to quantum attacks [3]. These algorithms currently fall into the following five families: code-based, isogeny-based, lattice-based, multivariate, and symmetric-based. Of these families, lattice-based algorithms have the best performance along with competitive key sizes. However, code-based and symmetric-based algorithms are the most mature and have the longest history of analysis. Algorithm security is described in relation to the difficulty of breaking symmetric algorithms, such as block ciphers and hash functions. NIST describes security levels 1, 3, and 5 as equivalent to the difficulty of a key search in AES-128, AES-192, and AES-256, respectively, and levels 2 and 3 as equivalent to the difficulty of a collision search in SHA-256 and SHA-384.

The NIST competition recently concluded its third round, selecting CRYSTALS-Kyber as the first Key Encapsulation Mechanism (KEM) winner and selecting CRYSTALS-Dilithium, Falcon, and SPHINCS$^+$ to be standardized as digital signatures [4]. Kyber, Dilithium, and Falcon are all lattice-based algorithms. Kyber was selected for its security, high performance, and moderate key and ciphertext sizes. Dilithium was selected for similar reasons. However, Falcon was also selected since it has smaller signatures than Dilithium which is beneficial for many applications. SPHINCS$^+$ has small public keys but large signatures and slow key generation and signing performance, but has high confidence in its security and provides diversity to the families of algorithms being standardized. NIST is continuing into another round of evaluation to investigate the potential standardization of more non-lattice algorithms to provide more diversity in the standards for security and flexibility. Of particular interest are efficient code-based and symmetric-based algorithms, as multivariate and isogeny-based algorithms have been recently shown vulnerable to powerful classical attacks [5], [6].

TABLE I: Summary of some of the most recent RTL implementations for round 3 candidates

Algorithm	Type	High-Speed		Low-Area	
		Design	Source Available	Design	Source Available
Code-based					
Classic McEliece	KEM	[7]		[7]	
BIKE	KEM	[8]	✓	[8]	✓
HQC	KEM	[9]^H	✓		
Isogeny-based					
SIKE	KEM	[10]	✓		
Lattice-based					
Kyber	KEM	[11]		[12]	✓
NTRU	KEM	[11]			
Saber	KEM	[11]	✓	[13]	✓
NTRU Prime	KEM	[14]	✓	[14]	✓
FrodoKEM	KEM				
Dilithium	DS	[15]	✓	[16]	✓
Falcon	DS	[15]	✓		
Multivariate					
Rainbow	DS	[17]			
GeMSS	DS				
Symmetric-based					
SPHINCS$^+$	DS	[18]			
Picnic	DS	[19]	✓		

^{H} High level synthesis implementation

The candidates of the competition are primarily evaluated based on their security, cost in terms of key size, ciphertext size, signature size, and performance in hardware and software. The primary targets for performance evaluation are Cortex-M4 for microcontrollers and Artix-7 for FPGA. For hardware evaluation, high-performance designs are primarily focused on achieving the lowest latency within some resource limit, such as the size of a particular FPGA. Lightweight implementations focus on low area and low power, the former giving the benefit of lower manufacturing cost or being able to

run on a smaller FPGA, and the latter being more suitable for devices that have limitations on the power available to them.

Thus far, most of the focus has been on key encapsulation finalist algorithms, though some alternates and digital signature algorithms have been implemented in hardware. The most recent work on the hardware implementations of post-quantum algorithms is summarized in Table I. Most of the hardware designs reported thus far have focused on the performance and area of the design and not on the power consumption. Several works have investigated software power consumption of post-quantum algorithms, e.g., [20] which compared energy costs on Intel CPUs, but there has not been a similar effort for hardware implementations.

Contribution: To the best of our knowledge, this work provides the first look into the power consumption of post-quantum cryptography in hardware. In particular, we gather measurements for 8 implementations covering 5 algorithms, including at least partial results for half of the future standards. Additionally, Power estimates were generated using Xilinx Vivado for a subset of the algorithms to provide a comparison in the accuracy of the estimate and actual power measurements. We also compare our results for FPGAs with results for the same algorithms running on high-end CPUs to investigate the benefit of offloading these algorithms to FPGAs.

II. Background

In this section, we provide a brief background on the algorithms included in this work and their implementations. The overview is to provide context for the energy results as the operations of the different families have a significant impact on the approach used for their implementation in hardware. For further details, see the algorithm specifications.

A. Key Encapsulation Mechanisms

Key Encapsulation Mechanisms (KEM) are used to establish a shared secret key between two parties over insecure networks. This is accomplished using three operations: key generation, encapsulation, and decapsulation. During a connection, key generation and decapsulation are performed by the server, and encapsulation is performed by the client. Key generation creates the key pair. The public key is sent to the client who uses it to perform encapsulation, generating the ciphertext and a shared secret. The ciphertext is then sent back to the server, which uses the secret key to perform decapsulation, calculating the same shared secret. For KEMs, the transmission cost is the size of the public key plus the size of the ciphertext since both must be transmitted during the key exchange.

1) Saber: Saber is a lattice-based algorithm based on the Module-Learning with Rounding (M-LWR) problem. Unlike Kyber's, Saber's parameters were chosen for optimal use of standard polynomial multiplication. The modulus is a power of two so that reduction is free in hardware, and only a small coefficient by large coefficient multiplications are used, which can be performed efficiently with shifts and additions. Both a high-speed design with several optimization levels and a low-area design are included in this work. The high-speed design supports all operations at a single security level as well as three levels of optimizations based on the unrolling factor for the polynomial multiplier [11]. These optimization levels are specified by the suffix `x1`, `x2`, `x4` to denote the unrolling factor used for the polynomial multiplier. The low-area design supports encapsulation and decapsulation at security level 3 [13]. Saber was not selected for standardization and was also not selected to move onto the fourth round due to its similarity to Kyber.

2) NTRU Prime: NTRU Prime is another lattice-based algorithm, however it is built upon a different type of lattice than Kyber and Saber. NTRU Prime has received both a high-performance and lightweight implementation for the Streamlined NTRU Prime variant [14]. These designs only fully support the security level 3 parameter set, but all operations are supported. NTRU Prime was not selected for standardization by NIST. However, it has received interest from some open-source projects. In particular, OpenSSH integrated Streamlined NTRU Prime and currently uses it as the default with the classical Elliptic Curve Cryptography `x25519` algorithm for hybrid key exchanges [22].

3) BIKE: BIKE is a code-based KEM, built on Quasi-Cyclic Moderate Density Parity Check (QC-MDPC) codes, which was advanced to the fourth round for evaluation as a potential future standard. The size of its private keys is in the same order as for lattice-based candidates, but BIKE has much larger public keys, ciphertexts as well as slower key generation and decapsulation. While the performance is lower, BIKE is built using well-studied hard problems, giving high confidence in its security. BIKE has received both a high-performance and low-area design [8]. The design has separate instances for each operation and security level.

B. Digital Signatures

While KEMs provide confidentiality in key exchanges, they do not confirm the identity of the sender or the integrity of the received message. Digital signatures allow message receivers to confirm the identity of the sender and confirm that the message has not been corrupted or modified. Three operations are used to accomplish this: a) key generation, which generates a key pair; b) signing, which uses the private key and generates a signature for a particular message; and c) verifying, which determines if the signature-message pair is valid for a given public key. For digital signatures, the transmission cost is the size of the public key plus the size of the signature since both must be sent.

1) Dilithium: Dilithium is a Module-Learning with Errors (M-LWE) signature scheme built using the Fiat-Shamir with aborts framework [23]. Like Kyber, its parameters were selected to enable efficient use of the Number Theoretic Transform (NTT) for polynomial multiplication. The NTT is a version of the Fast Fourier Transform (FFT) performed over integers that can be used to reduce the number of operations needed in polynomial multiplication. Dilithium was selected for standardization due to its security, reasonable key and

TABLE II: Area and performance overview of measured implementation - A7: Artix-7 ZUS+: Zynq UltraScale+

Algorithm	Design	Transmission Cost (KB)	Area				Kilocycles			Family
			LUT	FF	DSP	BRAM	KeyGen	Encaps /Sign	Decaps /Verify	
KEM (Security Level 3)										
LW Saber	[13]	2.1	6,713	7,363	32	0	-	46.7	52.8	A7
HS Saber x1	[11]	2.1	21,352	14,232	0	1.5	2.7	3.7	4.7	ZUS+
HS Saber x2	[11]	2.1	32,099	21,037	0	1.5	1.8	2.2	2.8	ZUS+
HS Saber x4	[11]	2.1	48,895	27,715	0	1.5	1.3	1.5	1.9	ZUS+
HS SNTRU Prime[1]	[14]	2.2	40,879 /36,789	22,382 /22,700	6/9	4.5/3.5	64	5	11	A7
LW SNTRU Prime[1]	[14]	2.2	6,379 /6,279	3,069 /3,086	6/7	4.5/3	629.3	29.3	85.6	A7
LW Kyber[1]	[12]	2.3	7,412 /6,875	4,644 /3,981	2/2	3/3	6.3	7.9	10	A7
HS BIKE[1]	[8]	49.6	9,868 /29,767	3,372 /5,072	0/13	7/30	-	44.6	611.2	A7
LW BIKE[1]	[8]	49.6	6,331 /8,678	3,339 /3,990	0/7	5/16	-	604.8	5,970.4	A7
Digital Signatures (Security Level 5)										
LW Dilithium	[16]	7.2	44,653	13,814	45	31	51	145.9	52.7	A7
HS Dilithium	[15]	7.2	53,187	28,318	16	29	14	55.1	14.6	A7
HS Falcon	[15]	3.1	13,956	6,737	4	2	-	-	4.7	A7
HS SPHINCS$^+$-256s-simple	[18]	29.8	51,130	74,576	1	22.5	-	-	-	A7
HS SPHINCS$^+$-256s-robust	[18]	29.8	50,070	75,738	1	30	-	-	-	A7
HS SPHINCS$^+$-256f-simple	[18]	49.2	51,009	74,539	1	22.5	-	-	-	A7
HS SPHINCS$^+$-256f-robust	[18]	49.2	50,341	75,664	1	30	-	-	-	A7

[1] Area reported separately for encapsulation/decapsulation

TABLE III: Measurement results for signatures on CW305 target board at 75 MHz

Author	Alg.	Level	Design	Op.	Freq. (MHz)	Throughput (Op/s)	Latency (Cycles/Op)	Power (mW)	Energy (μJ)
[15]	Falcon	1	hs	Verify	75	31,236	2,401	114.37	3.66
[15]	Falcon	5	hs	Verify	75	15,994	4,689	117.49	7.35
[15]	Dilithium	2	hs	Sign	75	2,051	36,559	214.48	104.55
[15]	Dilithium	3	hs	Sign	75	1,005	74,561	211.54	210.30
[15]	Dilithium	5	hs	Sign	75	793	94,537	216.21	272.54
[15]	Dilithium	2	hs	Verify	75	11,392	6,583	211.90	18.60
[15]	Dilithium	3	hs	Verify	75	7,712	9,725	217.34	28.18
[15]	Dilithium	5	hs	Verify	75	5,125	14,634	222.53	43.42

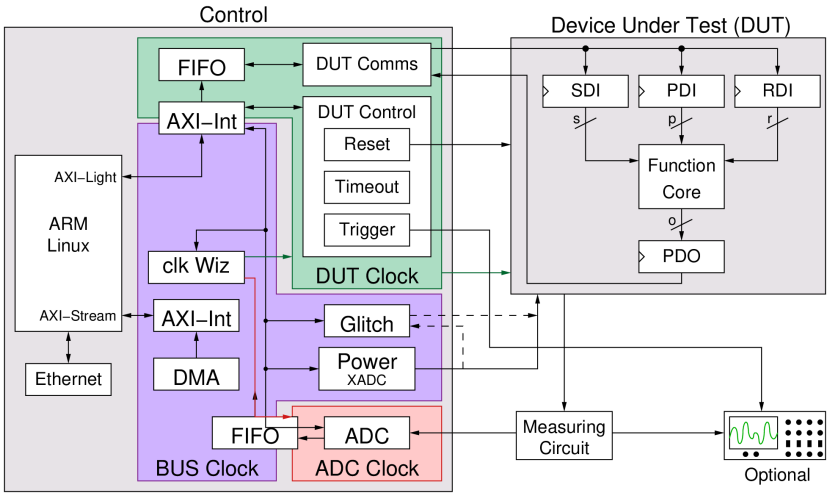

Fig. 1: Block diagram of FOBOS setup [21]

signature sizes, and fast performance for all operations. High-performance implementation of Dilithium is included in this work. This implementation supports all operations and security levels in a single core selectable at runtime [24].

2) Falcon: Unlike Dilithium, Falcon is an NTRU-based digital signature algorithm. It was selected for standardization

TABLE IV: Measurement results for KEMs on CW305 target board at 75 MHz

Author	Alg.	Level	Design	Op.	Freq. (MHz)	Throughput (Op/s)	Latency (Cycles/Op)	Power (mW)	Energy (μJ)
[13]	Saber	3	lw	Encap	75	1,758	42,654	83.57	47.53
[13]	Saber	3	lw	Decap	75	1,556	48,184	84.22	54.10
[11]	Saber	1	hsx1	Decap	75	25,134	2,984	172.19	6.85
[11]	Saber	1	hsx1	Encap	75	32,594	2,301	181.00	5.55
[11]	Saber	1	hsx2	Decap	75	38,013	1,973	229.81	6.05
[11]	Saber	1	hsx2	Encap	75	48,512	1,546	240.73	4.96
[11]	Saber	3	hsx1	Decap	75	15,117	4,961	157.30	10.40
[11]	Saber	3	hsx1	Encap	75	18,958	3,956	167.97	8.86
[11]	Saber	3	hsx2	Decap	75	24,398	3,074	241.87	9.91
[11]	Saber	3	hsx2	Encap	75	30,574	2,453	258.64	8.46
[11]	Saber	3	hsx4	Decap	75	35,014	2,142	317.71	9.07
[11]	Saber	3	hsx4	Encap	75	43,782	1,713	333.77	7.62
[11]	Saber	5	hsx1	Decap	75	10,028	7,479	154.66	15.42
[11]	Saber	5	hsx1	Encap	75	12,222	6,136	163.70	13.39
[11]	Saber	5	hsx2	Decap	75	16,793	4,466	220.65	13.14
[11]	Saber	5	hsx2	Encap	75	20,632	3,635	234.86	11.38
[11]	Saber	5	hsx4	Decap	75	25,184	2,978	298.79	11.87
[11]	Saber	5	hsx4	Encap	75	31,210	2,403	317.18	10.16
[14]	SNTRU Prime	3	hs	Encap	75	14,880	5,040	180.83	12.15
[14]	SNTRU Prime	3	hs	Decap	75	6,804	11,022	172.92	25.41
[14]	SNTRU Prime	3	lw	Encap	75	2,561	29,279	73.57	28.72
[14]	SNTRU Prime	3	lw	Decap	75	891	84,082	69.17	77.55
[8]	BIKE	3	lw	Encap	75	124	604,833	46.91	378.32
[8]	BIKE	3	hs	Encap	75	1,675	44,757	59.16	35.31
[8]	BIKE	3	hs	Decap	75	122	611,198	69.52	566.55
[8]	BIKE	3	lw	Decap	75	12	5,970,388	55.26	4,399.06

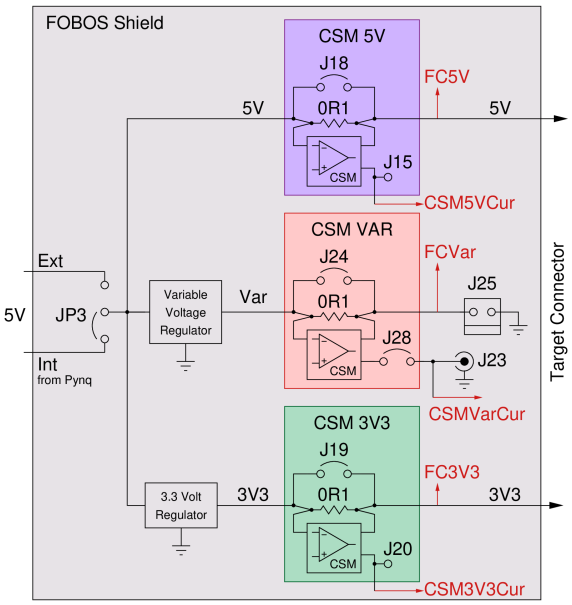

Fig. 2: Measurement circuit on FOBOS shield [21]

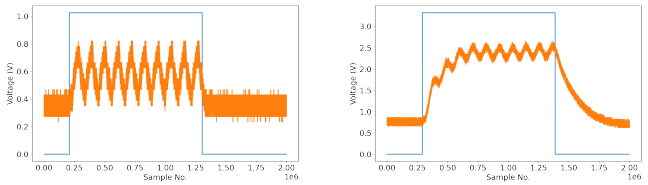

Fig. 3: 10 runs of HSx4 Saber encapsulation at 5MHz (left) and 50MHz (right) on CW305

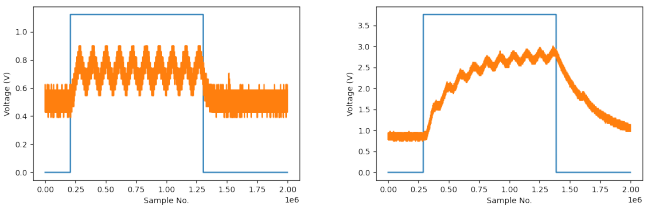

Fig. 4: 10 runs of HSx4 Saber encapsulation at 5MHz (left) and 50MHz (right) on Nexys 4

because its small transmission cost and fast verification make it better suited to some applications than Dilithium. However, key generation and signing are substantially slower for Falcon than Dilithium. Falcon has received only a hardware implementation of the verification operation thus far, likely due to the complexity of the key and signature generation operations. Verification is performed on polynomials with integers, allowing the NTT to be applied. Key generation and signing, however, use polynomials with floating-point coefficients requiring the more costly FFT. The existing design implements both security levels in separate instances [15].

3) SPHINCS$^+$: SPHINCS$^+$ was also selected for standardization as a digital signature scheme to provide diversity to the types of algorithms standardized. It is built upon symmetric primitives and provides parameter sets for various hash algorithms. The different parameter sets also make trade-offs between operation performance and signature size. The existing SPHINCS$^+$ implementation did provide public source code, but the corresponding paper provides energy results that

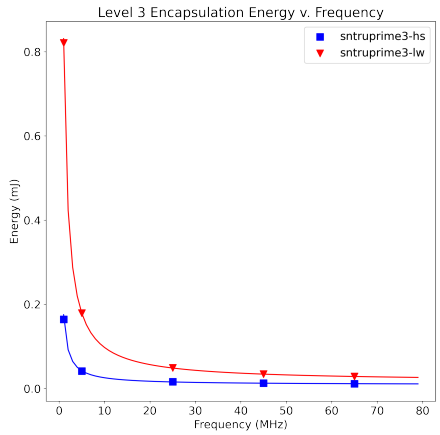

Fig. 5: KEM energy consumption vs. frequency for the Streamlined NTRU Prime encapsulation

can be used for comparison.

III. Methodology

In this section, we discuss the approach used for gathering accurate measurements and estimates for the selected algorithms.

A. Measurement Setup

Measurements were gathered using the Flexible Opensource workBench fOr Side-channel analysis (FOBOS), which is a platform for side-channel analysis of physical implementations which can also be used to measure average power [21]. The FOBOS platform consists of a set of hardware modules for a control board and target board, a Python library of drivers for these modules, and optionally a custom PYNQ shield for easy power measurement. As shown in Fig. 1, the control board contains modules to write test vectors to the target board, read actual outputs from the target board, and measure the power being used by the target board. If the custom PYNQ shield is used, the power to the target board can be supplied using one of the three power supplies. Each of these power sources has an INA225 current sense monitor (CSM) for measuring the current, as well as the ability to measure the voltage. The circuit for these power sources and their corresponding measurement circuits are shown in Fig. 2. The measurement circuits can be sampled either using an XADC on the control board or using an external oscilloscope. When measuring the PYNQ's FPGA, samples are taken approximately every 7.14μs, and the average and maximum are calculated in the FPGA while the DUT is running. The average and maximum values can be read from the memory-mapped registers after the measurement is complete.

The DUT module provides a wrapper that can receive test vectors and instructions from the control board and provides a simple interface to the target module. The design to be tested is instantiated in the `Function Core` in the DUT. The DUT design provides three FIFO inputs that are used to load test vectors into the target design. For this application, designs that use two interfaces, one for key inputs and one for ciphertext inputs, use the `SDI` input for loading the keys and `PDI` for ciphertexts and messages. Designs that use only a single interface use only the `PDI` interface. The `RDI` is used to provide randomness for side-channel protected implementations and thus is not used in this work. The DUT sends a trigger signal to the control board while the `Function Core` is in operations which allows it to know when to begin and finish the trace used to calculate the average power. In our setup, the control board is used only to power the target FPGA. Specifically, the V_{ccint} and V_{ccbram} power sources are given 1V from the variable power supply. All other power is supplied through the USB connection.

1) Calibration: The ADC measurements were calibrated against known currents and voltages on a DC load. Measurements were taken for current consumption from 0-350 milliamps, and the measured current was compared against the actual current to calculate the error at different currents. This was used to calculate an error-correction polynomial which is applied to the measurement. This process was performed for both the PYNQ ADC and the oscilloscope. The accuracy of the error correction and of the average power calculation was verified by programming the DC load to step through a series of different power draw and comparing the measured average against the known average of the DC load.

2) Trace Acquisition: Measurements were taken on a Nexys 4 board which use a `xc7a100tcsg324-1` chip as well as a Chipwhisperer CW305 which uses a `xc7a100tftg256-2L` chip. Measurements were taken using both an ADC on the PYNQ FPGA and an external oscilloscope. All figures are generated using results for the CW305 target board with measurements from an external oscilloscope.

When measuring the power consumption of the devices, it was seen that the initial power was substantially lowered by the discharge of the capacitors in the FPGA's power circuitry. Figs. 3 and 4 exemplify this. These figures show the voltage measurements from the current sense monitor for 10 sequential runs of Saber encapsulation. At high frequencies, the initial power consumption is greatly impacted by the discharge of the capacitors in the target board's power circuitry. The smoothing of the change in current from idle to running lasts for several runs of the operation. During these initial runs, the energy stored in the capacitor is being discharged and so the measured current is not reflective of the current consumed by the design. To account for this, several operations are performed in a single trace, and the average is calculated once the current consumption has stabilized. These figure also show how this varies between boards, with the CW305 showing less impact from the capacitors. In general, having more capacitors leading to more smoothing would increase the accuracy of measurements as short spikes in consumption are less likely to be missed. However, if the sample rate is much higher than the target clock frequency, the effect is minor.

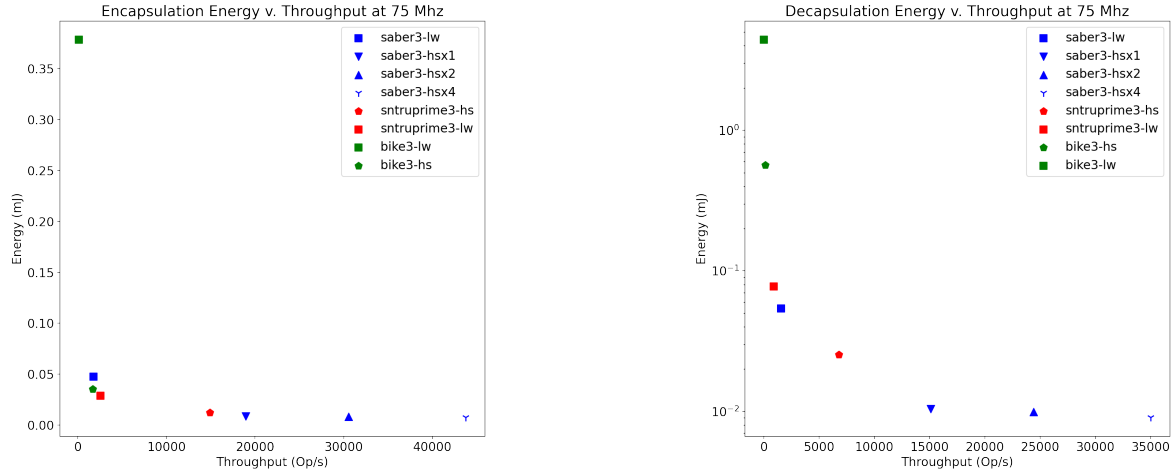

Fig. 6: KEM level 3 energy vs. throughput comparison for encapsulation (left) and decapsulation (right) on CW305

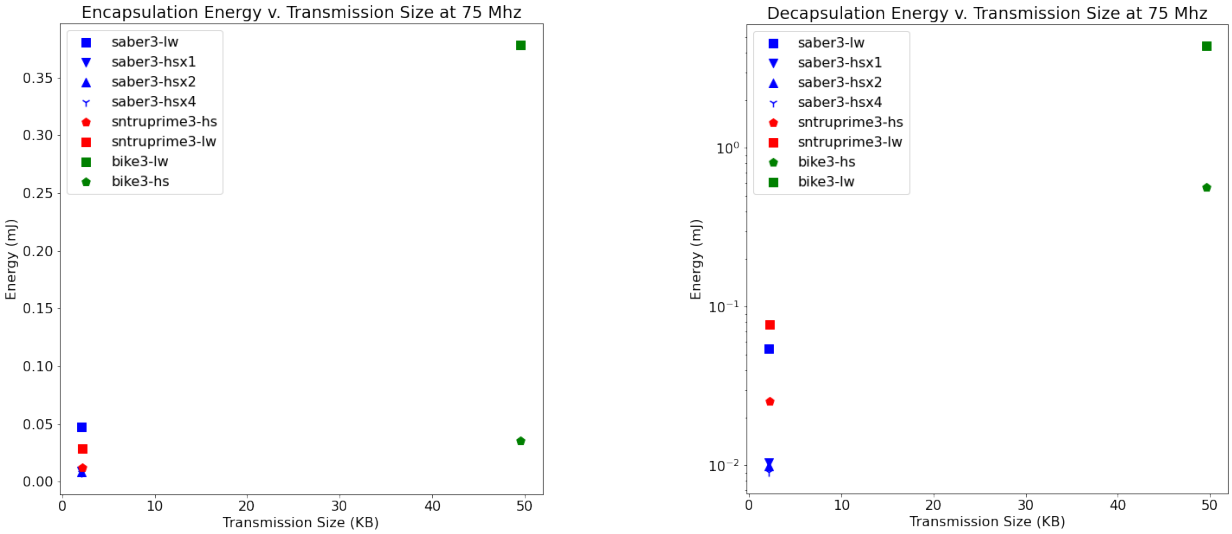

Fig. 7: KEM level 3 energy vs. transmission size (*PK* + *CT* bytes) comparison for encapsulation (left) and decapsulation (right) on CW305

B. Determining Minimum Energy Consumption

The power consumption of any design is the sum of the static power, which is independent of frequency, and the dynamic power, which is directly proportional to frequency. The static power is caused by the leakage current of the transistors, while the dynamic power is caused by the energy required to change the state of a transistor. Thus the power of consumption can be defined as:

$$\begin{aligned} p_{total} &= p_{dyn} + p_{stat} \\ &= \delta_{dyn} * frequency + p_{stat}. \end{aligned}$$

Further, the energy consumption of an operation is simply the average power consumption multiplied by the latency:

$$e = p_{total} * latency$$

$$\begin{aligned} &= (\delta_{dyn} * frequency + p_{stat}) * (\tfrac{cycles}{frequency}) \\ &= \delta_{dyn} * cycles + \tfrac{p_{stat} * cycles}{frequency}. \end{aligned}$$

Thus as the frequency increases, the static power contribution to energy consumption is reduced while the dynamic contribution remains constant. This is intuitive since the static power results from leakage current, and lower latency reduces the impact of this leakage, whereas dynamic power comes from the change of state in transistors, and the total number of changes is unaffected by the frequency. Therefore the minimum energy is $\delta_{dynamic} * cycles$. As exemplified in Fig. 5, the static contribution is minimal when operating at frequencies above low megahertz. Thus we can reasonably compare the energy consumption of all implementations at a single frequency so long as it is is in the high tens of

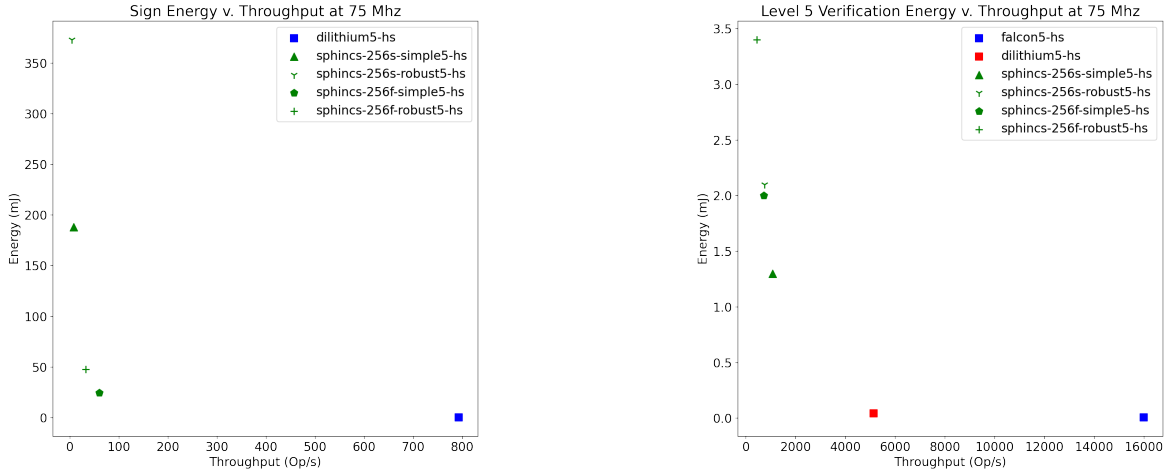

Fig. 8: Level 5 digital signature energy vs. throughput comparison for sign (left) and verify (right)

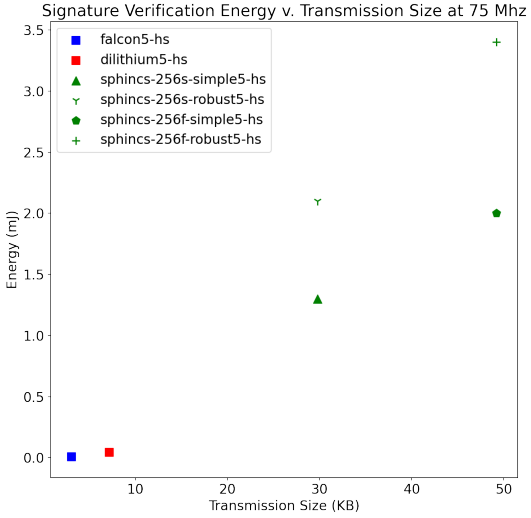

Fig. 9: Digital signature verification energy vs. transmission size (*PK* + *SIG* bytes)

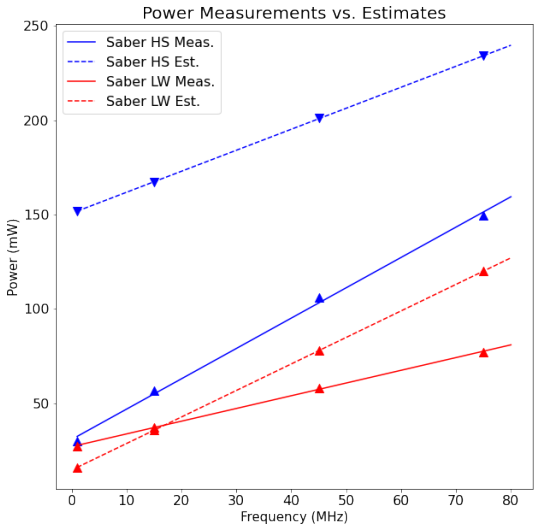

Fig. 10: Comparison of Vivado estimates and measurements for Saber encapsulation

megahertz. We chose 75 MHz as all designs were able to run at this frequency and it is high enough to make the static contribution minimal.

C. Power Estimates

Vivado estimates were generated for a subset of the implementations in order to compare the accuracy of these estimates verse lab measurements. The estimates were generated using Vivado 2021.2 following the approach given in Xilinx's example document for power analysis [25]. A signal activity file was generated using the post-implementation simulation with the same test vectors that were used in the lab measurements. Only the estimates for the V_{ccint} and V_{ccbram} power consumption are used for the comparison since only these two power supplies are measured on the target board. The other power sources provided by the estimates are not relevant to the energy consumed by the algorithm running within the FPGA.

IV. Results

A. Comparison Between Algorithms

Table II provides an overview of the performance and area results of the investigated implementations. For the sake of brevity, only the security levels with the most-complete results are included in this table and in the figures. For candidates with separate cores for different operations, the results are reported for the encapsulation/decapsulation area. Full results

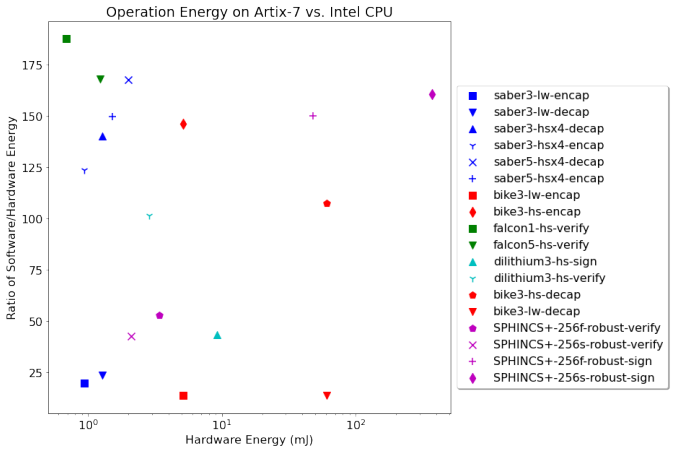

Fig. 11: Comparison of FPGA energy consumption of Artix-7 running at 75 MHz and Intel Core i7 processor energy consumption

for all measured implementations at 75 MHz are provided in Tables III and IV.

1) KEM Results: Security level 3 is commonly recommended as it gives a beneficial trade-off in performance and security. It is also the security level with the most-complete results and thus will be the focus of the KEM results.

As shown in Table IV, the lightweight implementations of Saber, Streamlined NTRU (SNTRU) Prime, and BIKE have significantly lower power requirements than the corresponding high-performance implementations but ultimately require more energy per operation. The performance increase of the high-performance designs outweighs the increase in average power consumption. In terms of throughput and energy usage, Saber has significantly better performance than SNTRU Prime for both encapsulation and decapsulation, as shown in Fig. 6. Saber also has slightly lower transmission costs than SNTRU Prime. Comparing the lattice-based candidates to the code-based candidates shows the significant advantage lattice-based algorithms have in performance, energy, and transmission size. The energy for BIKE encapsulation is similar to that of Saber and NTRU Prime, but it is close to an order of magnitude worse for decapsulation.

2) Signature Results: For digital signatures, the only security level all algorithms support is security level 5. Thus, we focus on results for this security level. Further, while the SPHINCS$^+$ implementation [18] does not provide a public source and thus was not measured on the same setup, it does provide energy results for the Artix-7. The design runs with two clock domains: the main controller and datapath run at 250MHz, while the SHA3 core runs at 500MHz. As discussed previously, the energy consumption at high frequencies is dominated by the constant dynamic contribution, so we can make reasonable comparisons between our results and the results reported for energy consumption. This is not true for power consumption, so we only include SPHINCS$^+$ in the energy consumption comparison. In order to allow for performance comparison, the throughput for SPHINCS$^+$ is scaled down proportionally to the same frequency as the lattice base designs. The throughput was scaled by the factor $75/500$ to estimate the performance of the design when the SHA3 core is running at 75 MHz.

As shown in Table II, SPHINCS$^+$ and Dilithium use roughly the same number of LUTs and BRAM, but Dilithium utilizes more DSPs, and SPHINCS$^+$ uses more registers. The Falcon implementation is smaller than both, but it only implements the verify operations, whereas the high-speed Dilithium implements all operations and security levels, and the SPHINCS$^+$ design implements sign and verify. For signature verification, Falcon provides the lowest energy consumption, highest throughput, and lowest transmission size as shown in Figs. 8 and 9. Dilithium performance is lower than Falcon but still substantially better than SPHINCS$^+$. For a signature generation, this trend continues with Dilithium having much lower energy consumption and much high performance than SPHINCS$^+$.

B. Comparison with Estimates

The comparison of a selected implementation's estimates generated with Vivado and measurements are shown in Fig. 10. In order to see if the complexity of the design impacts the accuracy of the estimate, comparisons are included for both the high-speed and lightweight Saber implementations. The relation between power and frequency for each of the entries in the graph are:

$$p_{hs\_meas.} = 1.6 \times freq + 30.1$$
$$p_{hs\_est.} = 1.11 \times freq + 150.7$$
$$p_{lw\_meas.} = 0.67 \times freq + 27.2$$
$$p_{lw\_est.} = 1.4 \times freq + 15.8$$

The Vivado estimates gave a lower contribution of dynamic power for both the lightweight and high-speed design. For the lightweight designs, the static power is also lower than measured. For the high-performance designs, static power is higher than measured. These results suggest that this basic approach for power estimation may provide a rough estimate but is not a replacement for the physical measurement of implementations.

These results match the results of several previous works comparing estimates and measurements for older versions of Xilinx's and Intel's power estimation tools. One previous work had performed similar experiments with common cryptographic implementations such as AES and DES and found the error of the estimate varied between 17% and 200% [26]. Another work compared estimates and measurements for Spartan-6 FPGAs in which they compared results for several different types of algorithms including AES, a Fast Fourier Transform, and multiple different implementations of a 32×32 bit multiplier. The estimation's error varied between -5% and 377% depending on the design and implementation language [27]. A similar work which compared estimates and measurements for various implementations of a 32×32 bit multiplier on an Cyclone-III FPGA found that the estimates

were much more accurate, varying between -11.7% and 13% error [28].

C. Comparison with Software

Fig. 11 shows the energy of the hardware implementations from this work compared with the results on an Intel i7-6700 CPU from [20]. Results in [20] are based on round 2 submissions that have minor tweaks to parameters. The results show that FPGA implementations have significant improvement in reducing energy consumption when compared to the software implementation, particularly for the high-performance implementations. Dilithium shows $43\times$ and $101\times$ improvement for signing and verification, respectively. Falcon verification at the highest security level uses $167\times$ less energy. The high-speed Saber implementation for security level 3 uses $123\times$ less energy for encapsulation and $140\times$ less for decapsulation. SPHINCS$^+$ requires between 38-$122\times$ less energy in hardware than software.

V. Conclusions

In this work, the power consumption of various post-quantum algorithms has been experimentally measured. For KEMs, the M-LWR scheme Saber has been shown to be more efficient in terms of performance and energy usage than the NTRU-based Streamlined NTRU Prime algorithm. However, both lattice-based algorithms require substantially less energy than the code-based algorithm BIKE. For digital signatures, it has been shown that the future lattice-based standards Dilithium and Falcon are substantially more efficient than the symmetric-based standard SPHINCS$^+$. The comparison was also made with the energy usage of these algorithms in software, showing that offloading these operations to FPGAs can reduce energy consumption up to two orders of magnitude.

References

[1] L. Chen, S. Jordan, Y.-K. Liu, *et al.*, "Report on Post-Quantum Cryptography," National Institute of Standards and Technology, Tech. Rep. NIST IR 8105, Apr. 2016.

[2] P. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," in *35th Annual Symposium on Foundations of Computer Science*, Santa Fe, NM, USA: IEEE Comput. Soc. Press, 1994.

[3] L. Chen, S. Jordan, Y.-K. Liu, *et al.*, "Report on Post-Quantum Cryptography," Apr. 2016. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf.

[4] D. Moody, "Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process," no. NIST IR 8413, 2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413.pdf.

[5] W. Beullens, "Improved Cryptanalysis of UOV and Rainbow," in *Advances in Cryptology – EUROCRYPT 2021*, A. Canteaut and F.-X. Standaert, Eds., Cham: Springer International Publishing, 2021.

[6] W. Castryck and T. Decru, "An Efficient Key Recovery Attack on SIDH (preliminary version)," *Cryptology ePrint Archive, Paper 2022/975*, Jul. 2022.

[7] P.-J. Chen, T. Chou, S. Deshpande, *et al.*, "Complete and Improved FPGA Implementation of Classic McEliece," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 3, Jun. 2022. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9695.

[8] J. Richter-Brockmann, J. Mono, and T. Güneysu, "Folding BIKE: Scalable Hardware Implementation for Reconfigurable Devices," *IEEE Transactions on Computers*, vol. 71, no. 5, 2022.

[9] C. Aguilar Melchor, N. Aragon, S. Bettaieb, *et al.*, "Hamming Quasi-Cyclic (HQC) Third round version,"

[10] R. El Khatib, R. Azarderakhsh, and M. Mozaffari-Kermani, "High-Performance FPGA Accelerator for SIKE," *IEEE Transactions on Computers*, vol. 71, no. 6, 2022.

[11] V. B. Dang, K. Mohajerani, and K. Gaj, "High-Speed Hardware Architectures and FPGA Benchmarking of CRYSTALS-Kyber, NTRU, and Saber," *Cryptology ePrint Archive, Paper 2021/1508*, Nov. 2021.

[12] Y. Xing and S. Li, "A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-Kyber on FPGA," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 2, Feb. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8797.

[13] A. Abdulgadir, K. Mohajerani, V. B. Dang, J.-P. Kaps, and K. Gaj, "A Lightweight Implementation of Saber Resistant Against Side-Channel Attacks," in *Progress in Cryptology – INDOCRYPT 2021*, A. Adhikari, R. Küsters, and B. Preneel, Eds., Cham: Springer International Publishing, 2021.

[14] B.-Y. Peng, A. Marotzke, M.-H. Tsai, B.-Y. Yang, and H.-L. Chen, "Streamlined NTRU Prime on FPGA," *Cryptology ePrint Archive 2021/1444*, Oct. 2021.

[15] L. Beckwith, D. T. Nguyen, and K. Gaj, "High-Performance Hardware Implementation of Lattice-Based Digital Signatures," *Cryptology ePrint Archive 2022/217*, Feb. 2022.

[16] G. Land, P. Sasdrich, and T. Güneysu, "A Hard Crystal - Implementing Dilithium on Reconfigurable Hardware," in *Smart Card Research and Advanced Applications*, V. Grosso and T. Pöppelmann, Eds., Cham: Springer International Publishing, 2022.

[17] A. Ferozpuri and K. Gaj, "High-speed FPGA Implementation of the NIST Round 1 Rainbow Signature Scheme," in *2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Cancun, Mexico: IEEE, Dec. 2018, pp. 1–8.

[18] D. Amiet, L. Leuenberger, A. Curiger, and P. Zbinden, "FPGA-based SPHINCS+ Implementations: Mind the Glitch," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, 2020.

[19] D. Kales, S. Ramacher, C. Rechberger, R. Walch, and M. Werner, "Efficient FPGA Implementations of LowMC and Picnic," in *The Cryptographers' Track at the RSA Conference 2020, CT-RSA 2020*, San Francisco: Springer, Feb. 2020.

[20] C. A. Roma, C.-E. A. Tai, and M. A. Hasan, "Energy Efficiency Analysis of Post-Quantum Cryptographic Algorithms," *IEEE Access*, vol. 9, 2021.

[21] A. Abdulgadir, W. Diehl, and J.-P. Kaps, "Flexible, Opensource workBench fOr Side-channel analysis (FOBOS), user guide," Dec. 2019.

[22] "OpenSSH 9.0 Release Notes," *OpenSSH*, [Online]. Available: https://www.openssh.com/txt/release-9.0.

[23] S. Bai, L. Ducas, E. Kiltz, *et al.*, "CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation (version 3.1)," Feb. 2021.

[24] L. Beckwith, D. T. Nguyen, and K. Gaj, "High-Performance Hardware Implementation of CRYSTALS-Dilithium," in *2021 International Conference on Field-Programmable Technology (ICFPT)*, 2021.

[25] Xilinx, "Vivado Design Suite Tutorial: Power Analysis and Optimization," [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2021_2/ug997-vivado-power-analysis-optimization-tutorial.pdf.

[26] D. Meidanis, K. Georgopoulos, and I. Papaefstathiou, "FPGA Power Consumption Measurements and Estimations Under Different Implementation Parameters," in *2011 International Conference on Field-Programmable Technology*, Dec. 2011.

[27] J. P. Oliver, J. P. Acle, and E. Boemo, "Power Estimations vs. Power Measurements in Spartan-6 Devices," in *2014 IX Southern Conference on Programmable Logic (SPL)*, 2014.

[28] J. P. Oliver and E. Boemo, "Power Estimations vs. Power Measurements in Cyclone III Devices," in *2011 VII Southern Conference on Programmable Logic (SPL)*, Cordoba, Argentina: IEEE, Apr. 2011.