Iterative Frequency Tuning Targeting Energy Efficiency Ratio for FPGA-based Post-Quantum Cryptographic Cores

Srijeet Guha* and Andrea Guerrieri† *Senior Member, IEEE*
*NVIDIA Graphics Private Limited, India
†School of Engineering, HES-SO Valais-Wallis, Switzerland

***Abstract*—Electronic Design Automation tools such as high-level synthesis (HLS) have raised the level of abstraction, allowing the use of FPGAs in multiple domains, including post-quantum cryptography (PQC). Among power performance area (PPA) tradeoffs, the energy efficiency ratio is of increasing interest, especially with the need to integrate PQC cores into battery-powered portable appliances. Nevertheless, to target energy efficiency at the HLS level, post-place, and route metrics should be taken into account, which is extremely time-consuming. In this paper, we developed an iterative frequency tuning framework capable of extracting the best quality, energy-efficient design in logarithmic time complexity, converging to the best design frequency with a speedup of 2.89× with respect to the classical approach. The framework also allows the selection of the criticality of each metric thus altering the design cost function to suit the requirements of the HLS designers. The framework and the results will be released as open-source, allowing HLS designers to design the highest energy efficiency ratio PQC cores quickly.**

I. Introduction

High-level synthesis (HLS) is a mature Electronic Design Automation (EDA) tool that produces HDL code from C/C++, reducing the time from algorithm to hardware design on an FPGA. Modern HLS tools offer several synthesis directives helping to generate a wide range of micro-architectures, each with different characteristics in terms of latency, resource utilization, and power consumption. Degrees of freedom during logic synthesis are associated with the choice of gate size meanwhile during HLS, the generated designs might be significantly different in terms of the number of components to meet performance requirements.

Design Space Exploration (DSE) is the process of systematically exploring different design configurations to find the optimal hardware implementation for a high-level description of the system. However, this process can be very time-consuming and may not lead to the best configuration if all the implementation metrics have not been considered. To reduce DSE execution time, one might attempt to optimize quality metrics for metrics like initiation interval (II), resource utilization, or operating frequency to achieve the best solution, however, this solution might not be energy-efficient. Therefore, multiple factors should be considered to achieve energy-efficient hardware solutions. While some of these metrics can be extracted after the HLS process (i.e. II) itself, other metrics (i.e. power consumption) will require placement and routing, slowing down the exploration process. To cope with this issue, we developed an iterative frequency-tuning mechanism to accelerate the exploration process and extract the most performance-to-energy-efficient solution quickly.

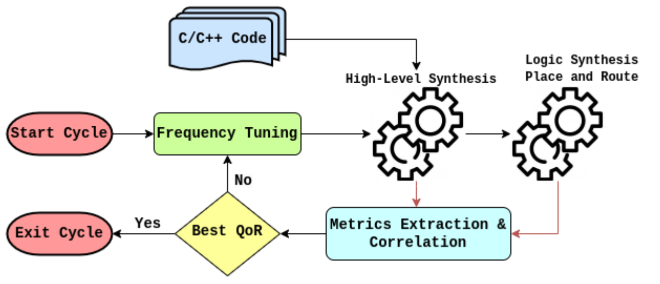

Fig. 1. Iterative Frequency Tuning Methodology. We bridge the results from HLS and after place-and-route, correlate the metrics and tune the synthesis frequency to achieve the best-performing and energy-efficient solution.

II. Contribution and Related Work

An overview of energy consumption in PQC and the urgent need for improving this aspect has been discussed by Roma et al. [1]. Number theoretic transform (NTT) algorithm plays an important part in most lattice-based PQC algorithms. Recent code optimizations of the NTT algorithm, for HLS tools, have majorly focused on generating hardware designs with reduced latency or resource utilization [2] [3]. Many HLS methodologies utilize DSE at the post-synthesis stage to find the Pareto-dominant solution. Some novel methodologies also leverage machine learning techniques [4] and genetics algorithms to prune and reduce the space of solutions [5]. In this paper, we use variations of power performance area product (PPA) as a metric to determine the most energy, latency, and resource-efficient hardware design. We then demonstrate that optimizing just one metric like latency, power consumption or resource utilization may lead to sub-optimal results.

Furthermore, we propose an iterative frequency tuning framework capable of extracting the hardware configuration for the best quality energy-efficient solution in logarithmic time complexity with a speedup of 2.89× with respect to the classical approach. We realize that different design requirements demand different criticality of power, performance, and area in the cost function (ϕ) of the design. Our tuning framework allows altering the cost function by changing the criticality of each metric. In this paper, we have set our framework to have a least count of 50MHz. Our framework allows reducing the frequency granularity (f_g) to as low as 1MHz. The reduction of frequency granularity in our framework gives a better speedup than the one mentioned above. Through our experiments, we realized that the least count of 50MHz is a good enough trade-off between exploration time and achieved design frequency.

To the best of our knowledge, this paper is the first attempt to propose an iterative DSE tool to exploit the energy efficiency of the generated design while designing PQC cores. Instead of performing a full design space exploration, we leverage iterations to execute successive approximations and converge to the best solution faster.

III. Iterative Frequency Tuning

A classical approach often used to determine the optimal hardware configuration and minimize the value of the cost function (ϕ) is a frequency sweep from the minimum feasible frequency to the maximum frequency supported by the FPGA (f_{spec}) with a pre-determined frequency granularity (f_g). This is often time-consuming because at each frequency one has to go through the entire cycle of HLS, placement, and routing to determine the value of the cost function.

To reduce exploration time and converge on the best design configuration faster, we developed our tuning framework based on the following observations:

1) Most of the exploration time is consumed in placement, and routing of the design. HLS consumes comparatively very little time.
2) Some metrics like latency, wall clock time (which are indicative of performance), and LUTs, FFs consumed (which are indicative of area) can be approximated just from the high-level synthesis and we do not need to pass through the entire cycle thus saving exploration time.
3) Dynamic power consumption is directly proportional to the operating frequency and capacitance of the circuit. Capacitance is closely related to the area consumed. Thus, the minima of the power-frequency curve occur in a region near the minima of the area-frequency curve.

Our framework helps achieve design configurations that minimize the value of the cost function after considering performance, power, and area, each with user-defined criticality as detailed in section III-A. The iterative frequency tuning framework uses a variation of binary search, as detailed in section III-B to converge to the best design frequency with an accuracy of $\pm\ f_g$.

A. Choice of the metrics

Multiple composite performance metrics like energy-delay product (EDP) and energy-delay squared product (EDSP) can help determine the most performance, area, and power-efficient design solutions. The main drawback of using EDP and EDSP is the lack of consideration of area as a metric for HLS designs. To overcome this problem, we use a customizable variation of the power, performance, and area product (PPA).

In our iterative frequency tuning framework, we allow HLS designers to adjust the criticality of each parameter in the PPA product as per their design requirements. Thus, the resulting cost function (ϕ) that we can achieve using our frequency tuning framework is as described in equation 1.

$$\phi = Power^p \times Performance^q \times Area^r : p, q, r \in \mathbb{W} \quad (1)$$

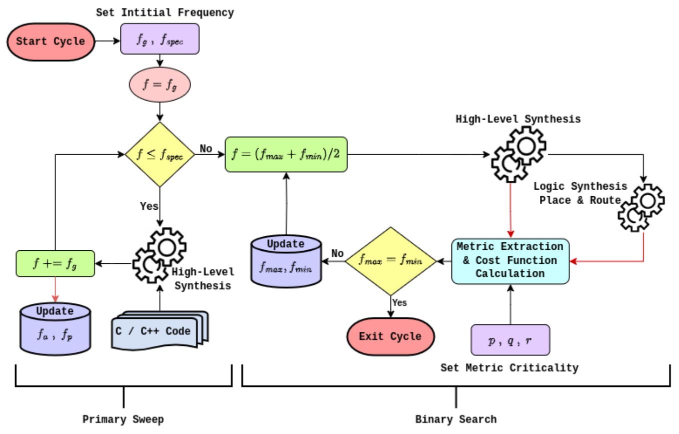

Fig. 2. Iterative Frequency Tuning Framework. The framework has 2 distinct parts: **Primary Sweep** and **Binary Search**. The Primary Sweep is very fast using just HLS. At every iteration, the synthesis frequency is increased by a factor of frequency granularity, to define the range of interest. The Binary Search will instead analyze, with accuracy, the post-place and route data to extract the best PPA.

where p, q, and r denote the criticality of power, performance, and area in the cost function. Through our framework, we try to converge to the best design frequency minimizing the value of ϕ in logarithmic time. Needless to say, with the above flexibility, we can even modify the cost function to mimic other standard composite metrics like EDP or EDSP.

B. Binary Search

The framework starts by reducing the frequency range over which it needs to minimize the cost function. This is done by performing a primary sweep over the entire frequency range from f_g to the maximum frequency supported by the device (f_{spec}) with a granularity of f_g. The primary sweep only involves using an HLS compiler for an approximate measure of performance and area as seen in the primary sweep section of figure 2. This sweep is used to record the frequencies of maximum performance (f_p) and minimum area (f_a) respectively. We calculate the offset frequency f_o as described in equation 2.

$$f_o = (f_p\ +\ f_a)/2 \quad (2)$$

The minimum (f_{min}) and the maximum (f_{max}) frequency limits are thus calculated as mentioned in described in equations 3 and 4.

$$f_{min} = max(min(f_p,\ f_a)\ -\ f_o,\ f_g) \quad (3)$$

$$f_{max} = min(max(f_p,\ f_a)\ +\ f_o,\ f_{spec}) \quad (4)$$

f_{min} and f_{max} are rounded to the nearest integral multiple of f_g at each step during the above-mentioned primary sweep and the following binary search. We apply binary search over the calculated frequency range using the full cycle of HLS, placement, and routing to find the value of ϕ. We compare the values of ϕ at f_{min}, f_{max}, $(f_{min} + f_{max})/2$. Based on the calculated values, we update f_{min} and f_{max} and repeat the above step until we converge at a common frequency as seen in the binary search section of figure 2.

C. Expected Speedup

We assume that a standard HLS compiler requires an average of τ_i secs to compile the code and extract an approximate measure of performance and area, while a complete cycle of HLS, placement, and routing requires an average of τ_f secs for a given frequency. We calculate the number of iterations (η) required to find the optimal hardware design through the classical approach using equation 5.

$$\eta = (f_{spec} - f_g)/f_g + 1 = f_{spec}/f_g \tag{5}$$

However, our framework reduces the frequency range using the primary sweep. Thus, the maximum number of iterations possible after the primary sweep can be calculated using equation 6.

$$\eta_{red} = (f_{max} - f_{min})/f_g + 1 \tag{6}$$

Furthermore, our framework uses binary search to reduce the number of iterations that go through the full cycle to $\log_2 \eta_{red}$. Thus, the time required to find the optimal hardware configuration can be calculated using equation 7.

$$\tau_{tot} = \tau_i \eta_{red} + \tau_f \log_2 \eta_{red} \tag{7}$$

Therefore, the speedup offered by our iterative frequency tuning framework over the classical approach can be best described by equation 8.

$$speedup = \tau_f \eta / \tau_{tot} = \tau_f \eta / (\tau_i \eta_{red} + \tau_f \log_2 \eta_{red}) \tag{8}$$

Based on point 1 of our above observations, we may approximate, $\tau_i \ll \tau_f$, thus simplifying the speedup to equation 9.

$$speedup = \eta / \log_2 \eta_{red} \tag{9}$$

Equations (5), (6) and (9), show how the speedup increases with a decrease in f_g.

IV. Implementation and Experimental Results

Our iterative frequency-tuning framework has been implemented in Java 8.0 capable to run on multiple platforms. It automatically interfaces with EDA tools (in our case Intel HLS and Intel Quartus Prime) to run and scrape the generated report files and collect metric values of designs at different frequencies. The framework accepts frequency granularity (f_g), the maximum frequency supported by the device (f_{spec}), and the criticality of each metric in ϕ. The framework automatically does a primary sweep, calculates f_{max} and f_{min} and performs a binary search over the frequency range.

We applied our methodology to Number Theoretic Transform (NTT) module that executes the polynomial multiplication in multiple standard PQC candidates announced by NIST (i.e. Kyber, Dilithium, and Falcon). When pipelined, the presence of a loop-carried dependency in the inner-most loop makes it hard to achieve an optimal initiation interval (i.e. $II = 1$), as demonstrated in [2]. The source code shown in Listing 1, this irregularity makes our approach even clearer how the target frequency affects the area and the latency of the generated components involved (PPA).

Listing 1. *Number Theoretic Transform NTT*, original code of Kyber-768.

```
k = 1;
for(len = 128; len >= 2; len >>= 1){
  for(start = 0; start < 256; start = j + len){
      zeta = zetas[k++];
      for(j = start; j < start + len; j++){
      #pragma pipeline
         t = fqmul(zeta, r[j + len]);
         r[j + len] = r[j] - t;
         r[j] = r[j] + t;
 } } }
```

We used Intel HLS and Intel Quartus Prime .v24.01 to compile the optimized NTT code proposed by [2] and generate hardware designs at different frequencies on Cyclone-10GX Intel FPGA, part-number 10CX220YF780I5G. However, our approach applies to any HLS tools. On average, at different frequencies, Intel HLS requires 42.83 secs (τ_i) to compile the optimized NTT source code while Intel Quartus Prime requires 1021.56 secs (τ_f) to generate placed and routed designs. This validates points 1 and 2 of our observations in section III.

For validation, we use f_g as 50MHz and the f_m as 1GHz. We also set the criticality of each metric in ϕ as 1 equaling ϕ with the standard PPA metric. To validate our initial proposition, which was to reduce DSE execution time, optimizing for singular metrics, might lead to sub-optimal hardware designs if all the metrics like energy, performance, and area are not taken into consideration, we perform a frequency sweep from f_g to f_{spec} with a frequency granularity of f_g. At each frequency, we go through an entire cycle of HLS, placement, and routing. The data of this frequency sweep can be seen in Table I. At each frequency, we record initiation interval (II), latency, wall clock time of the design (WCT), number of LUTs and FFs used in the design, and the total power (TP) and energy consumed during execution of the hardware design. To calculate ϕ, we used the following assumptions:

$$Power = TP,\ Performance = WCT,\ Area = LUTs$$

We have also set the criticality of each metric in ϕ (ie. p, q, r) as 1. Thus equaling ϕ with the standard PPA metric.

$$\therefore\ \phi = (TP) \times (WCT) \times (LUTs)$$

In table I, we have highlighted the frequency and the individual metrics where they achieve their minimum value. As seen from the table I, latency, LUTs used, and TP consumed achieved their minimum value at 50MHz while wall clock time and total energy consumed achieved their minimum value at 200MHz. However, we achieve the minimum ϕ at 150MHz which is not equal to either of the above frequencies. This proves optimizing just for singular metrics might result in sub-optimal hardware designs. We also observe that the minimum power-frequency curve occurs in a region surrounding the minimum area-frequency curve thus correctly proving observation 3 of section III.

TABLE I

RESULTS: FREQUENCY SWEEPING. TABLE SHOWS: COST FUNCTION (ϕ), WALL CLOCK TIME (WCT), INITIATION INTERVAL (II) AND TOTAL POWER (TP). THE HIGHLIGHTED VALUES REPRESENT THE MINIMUM VALUES FOR EACH PARAMETER, WHICH IS NOT THE SAME AS THE FINAL PPA (150MHZ).

Target Freq. (MHz)	Fmax (MHz)	II	Latency (cc)	WCT (us)	LUTs ($\times 10^3$)	FFs ($\times 10^3$)	TP (mW)	Energy (uJ)	ϕ ($\times 10^8$)
50	153.30	1	**1556**	10.15	**13.33**	13.42	**2389**	24.25	3.23
100	213.72	1	1567	7.33	13.36	13.56	2584	18.95	2.53
150	261.30	1	1574	6.02	13.37	13.77	2670	16.08	**2.15**
200	289.60	1	1583	**5.47**	13.49	14.00	2928	**16.00**	2.16
250	336.36	2	2479	7.37	13.45	14.16	2973	21.91	2.95
300	379.08	2	2500	6.59	13.51	14.20	3046	20.09	2.71
350	388.35	2	2500	6.44	13.52	14.25	3038	19.56	2.64
400	376.36	3	3906	10.38	13.55	14.76	3034	31.49	4.27
450	390.01	3	3914	9.93	13.57	14.81	3071	30.51	4.14
500	375.38	3	3914	10.43	13.57	14.80	3055	31.85	4.32
550	397.62	5	6227	15.66	14.03	20.17	3261	51.07	7.16
600	400.48	11	13734	34.29	16.07	24.18	3715	127.40	20.47
650	402.34	12	15137	37.62	16.07	24.05	3701	139.23	22.37
700	359.97	13	16024	44.51	16.01	23.99	3660	162.92	26.08
750	375.92	14	17449	46.42	16.04	24.03	3823	177.46	28.46
800	382.70	15	18326	47.89	16.11	32.36	3895	186.52	30.06
850	413.24	16	19759	47.81	16.13	32.40	3996	191.05	30.82
900	458.72	17	20631	44.98	16.09	32.37	3890	174.95	28.15
950	411.23	18	22069	53.67	16.10	32.38	3887	208.62	33.59
1000	370.78	19	23447	63.24	16.10	32.42	3892	246.12	39.61

A. *Convergent Validity*

To validate the convergence of our approach, we used our framework to find the optimal hardware implementation, with the minimum ϕ, for the optimized NTT code mentioned above. Based on our assumptions of f_g, f_{spec}, and ϕ, the number of iterations in the classical approach, η is 20. Our framework, after the primary sweep, converged the values of f_a and f_p to 50MHz and 200MHz respectively. f_{max} and f_{min} were thus calculated to be 50MHz and 350MHz, reducing the maximum possible iterations, η_{red} to 7. The framework then applies a binary search over the above frequency range to achieve the minimum ϕ of 2.15×10^8 at 150MHz. Figure 3 shows the 4 frequencies sampled using a full cycle of HLS, placement, and routing during the binary search. The total time required by the classical approach, τ was 5.67 hours while our framework required a total time, τ_{tot} of 1.96 hours resulting in a total speedup of $2.89\times$. As mentioned before, the speedup will further increase if f_g is reduced.

V. CONCLUSION AND FUTURE WORK

In this work, we present a framework to help designers use design automation tools such as HLS to extract the best quality energy-efficient design in logarithmic time complexity.

We demonstrated how using common metrics to evaluate the quality of generated designs might lead to sub-optimal results while considering other factors such as energy efficiency. Future work includes improving our framework to adapt to more diverse design cost functions, applying our method to the other PQC schemes to normalize the approach, and expanding our framework to other EDA vendors' tools. We envision our methodology as promising for designing high-quality and energy-efficient post-quantum cryptography cores using HLS. Our framework will be released as open-source upon publication.

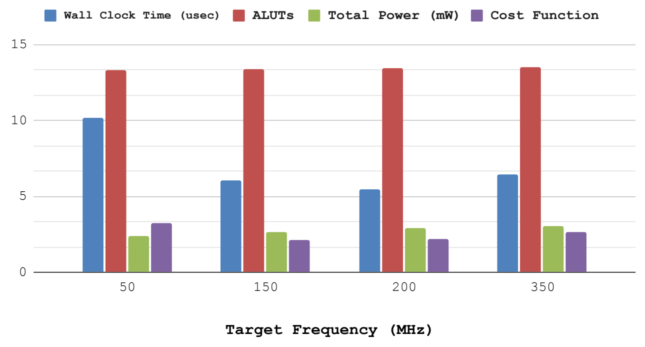

Fig. 3. Frequency Analyzed During the second phase (binary search) of the Iterative Frequency Tuning. The best solution is obtained at 150MHz.

REFERENCES

[1] C. A. Roma, C.-E. A. Tai, and M. A. Hasan, "Energy efficiency analysis of post-quantum cryptographic algorithms," *IEEE Access*, vol. 9, pp. 71 295–71 317, 2021.

[2] A. Guerrieri, G. D. S. Marques, F. Regazzoni, and A. Upegui, "Optimizing post-quantum cryptography codes for high-level synthesis," in *2022 Euromicro Conference on digital systems Design (DSD22)*, Gran Canaria, Spain, 2022, pp. 361–67.

[3] A. Guerrieri, G. Da Silva Marques, F. Regazzoni, and A. Upegui, "H-saber: An FPGA-Optimized version for designing fast and efficient post-quantum cryptography hardware accelerators," in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, 2023, pp. 1–6.

[4] M. I. Rashid and B. C. Schafer, "Fast and inexpensive high-level synthesis design space exploration: Machine learning to the rescue," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 11, pp. 3939–3950, 2023.

[5] Y. Liao, T. Adegbija, and R. Lysecky, "Efficient system-level design space exploration for high-level synthesis using pareto-optimal subspace pruning," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 567–572.