High Throughput GPU Implementation of HQC Key Encapsulation Mechanism

Wai-Kong Lee, Ramachandra Achar, Mingxiang Chen, Wijden Elmetamri

Universiti Tunku Abdul Rahman, Carleton University, Institut Supérieur d'Informatique et de Mathématiques de l'Université de Monastir

Introduction

The HQC (Hamming Quasi-Cyclic) [2] Key Encapsulation Mechanism (KEM) is a code-based cryptographic scheme designed to provide secure communication in the era of quantum computing. It was selected as one of the NIST post-quantum standards in 2025. Although HQC was extensively studied since its inception, most of the prior works focuses on its security aspects, the implementation aspects are less researched compared to its lattice counterparts like Kyber [3]. This work focuses on the implementation of the HQC KEM using Graphics Processing Units (GPUs) to achieve high throughput key encapsulation and decapsulation performance.

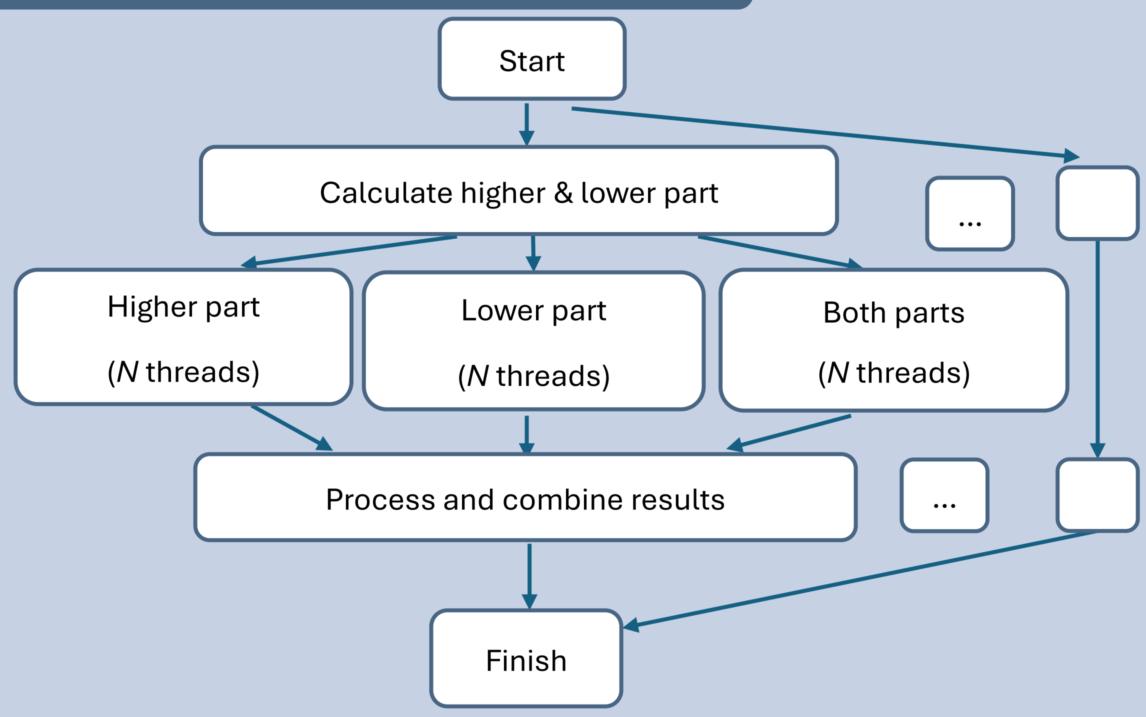
Polynomial multiplication in *GF(2)* is the most time-consuming operation in HQC, which is also used frequently in encapsulation and decapsulation. Based on our profiling experiments on the reference implementation, it takes up ≈ **90**% of the encapsulation and decapsulation time. The HQC authors proposed to use the recursive Karatsuba algorithm, which is not suitable for parallel implementation on a GPU. This is because recursive function calls are commonly implemented using "dynamic parallelism" feature in a GPU, which is costly for many levels of recursion.

Proposed Solutions

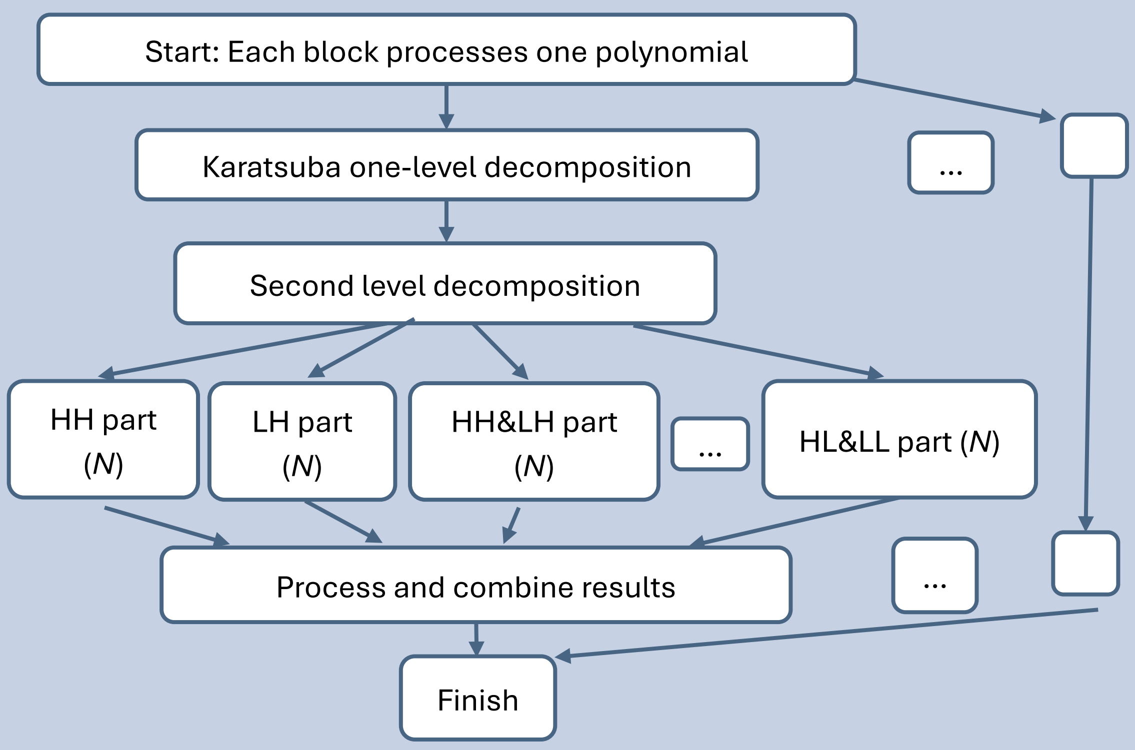
In this work, we proposed a parallel and iterative version of Karatsuba algorithm to remove the recursive function call. We also heavily optimized the kernel by utilizing various cache memories available in a GPU and evaluated its performance with different levels of recursion. We found that although Karatsuba algorithm can reduce the overall computational complexity, the more recursion level reduces its parallelism, leading to lesser gain in performance.

Diagrams

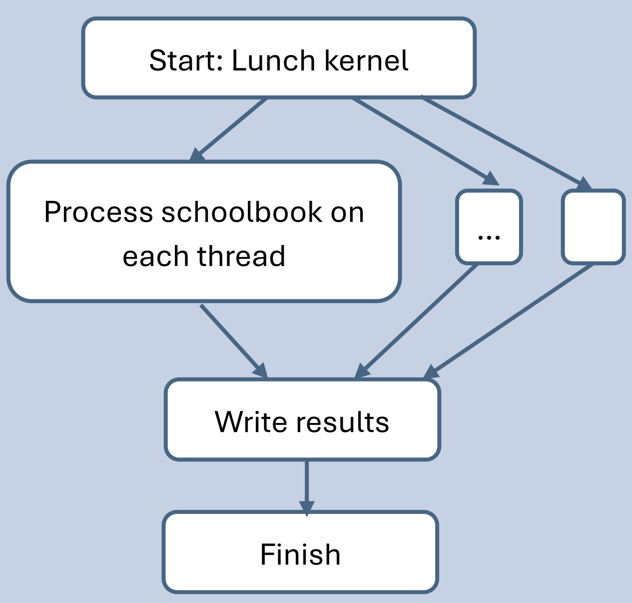
Karatsuba One-level Decomposition on GPU



Karatsuba Two-level Decomposition on GPU



Schoolbook on GPU



Preliminary Experimental Results

Our implementation (HQC-128) utilizes multiple blocks concurrently; each block utilizes multiple threads to compute one polynomial multiplication.

Referring to Table 1, on an RTX4060 Ti GPU, Karatsuba with one level recursion achieved **2.02×** speed-up compared with schoolbook method, at batch size 65536. The best throughput achieved is 42695 multiplications per second with two levels of Karatsuba recursion. However, on an A100 GPU, Karatsuba two levels is actually slower than one level. This shows that more recursion in Karatsuba algorithm does not always yield better results on a GPU.

Table: Performance of the proposed GPU-based polynomial multiplication in *GF(2)*

Variants	**Throughput (Mult/s)**		**Batch Size**	**Speed-up (RTX4060 Ti)**
	A100	RTX4060 Ti		
Schoolbook	13542	15220	256	**1×**
	22979	16557	4096	**1×**
	29034	17785	65536	**1×**
Karatsuba -1-level	30815	29679	256	**1.95×**
	45291	33191	4096	**2.00×**
	57445	35946	65536	**2.02×**
Karatsuba -2-level	30815	35213	256	**2.18×**
	58503	41850	4096	**2.53×**
	45962	**42695**	65536	**2.40×**

On-going Explorations

- **64-bit × 64-bit carry-less multiplication**: Unlike conventional polynomial multiplication, HQC uses carry-less multiplication. The HQC authors utilized a state-of-the-art version [1] which is faster than a bit-by-bit computation, but is this optimal for GPU architectures?
- **Tensor cores:** Can we formulate the carry-less multiplication as a matrix-matrix multiplication, so as to utilize tensor cores for a higher performance?
- **Other performance bottlenecks:** Including the random sample generation (SHAKE). The complete GPU-accelerated HQC KEM will be open-source it to the public in near future.

Contacts

- **Wai-Kong Lee**
  Universiti Tunku Abdul Rahman
  wklee@utar.edu.my
- **Ramachandra Achar**
  Carleton University
  achar@doe.carleton.ca
- **Mingxiang Chen**
  Carleton University
  EddieChen@cmail.carleton.ca
- **Wijden Elmetamri**
  Institut Supérieur d'Informatique et de Mathématiques de l'Université de Monastir
  wijdene.mootamri66@gmail.com