Cryogenic Control Analysis and Optimization for Quantum Computation

An infrastructure for analysis of classical control unit architectures for quantum computation

Adam Holmes

September 2016

Framework Overview

To perform quantum computation, classical control units will be located at one (or more) temperature level(s) of dilution refrigerator-based superconducting quantum computers. The question considered in this work is: what are the tradeoffs between targeting lower computation or lower memory usage within classical control units with differing hardware constraints located at different temperature levels? To this end, the parameters under consideration are:

- Dilution Refrigerator Constraints,
- Hardware Characteristics, and
- Benchmark Code Generation and Manipulation.

Dilution Refrigerator Constraints

The architecture of a dilution refrigerator being considered is a multi-stage cooling apparatus, using liquid nitrogen and liquid helium. Of interest right now are two characteristics: cooling capacity of temperature stages and latency of microwave links crossing these thermal boundaries. Using information from the specifications of a typical refrigerator (specifically the Oxford-Instruments TritonXL), these temperature levels and cooling capacities are shown in Table 2.

Temperature Stage (Kelvin)	Cooling Capacity (Watts)
20 mK	25 μW
100 mk	1 mW
4K	1.35-2W

Table 1: Temperature Levels and Cooling Capacities

These figures are approximate, and relate specifically to one realization of a dilution refrigerator, so are definitely subject to change. They differ somewhat from the estimates brought up in discussions, where we saw that the 20 mK level cooling capacity is approximately two orders of magnitude less than the cooling power available at the 4K temperature level. Further clarification will be sought to establish correct figures.

Additionally, there is a range of latencies introduced in passing microwave links through temperature interfaces. These seem to vary between 1-10ns approximately, growing longer as the gap widens between the target temperatures of the stages being crossed.

Hardware Characteristics

There are three primary types of hardware under consideration:

- RSFQ: Rapid Single Flux Quantum
- RQL: Reciprocal Quantum Logic
- CryoCMOS: Cryogenic CMOS

Each of these is characterized by a different energy usage/logic gate relationship. these values are typically characterized similarly to Table 2.

Hardware Type	Energy Required Per Gate (Joules)
RSFQ	10^{-19}
RQL	10^{-19}
CryoCMOS	10^{-15}

Table 2: Hardware Types and Energy Consumed Per Gate

These can be used directly to convert between computation and energy, and will be inserted into the model.

It is also important to consider power consumption of memory systems under these types of conditions. Here, we will analyze power usage of cryogenic CMOS memory devices, and consider implementations of cryogenic persistent memory systems. Persistent memory could potentially be useful in applications that rely upon a precomputed library of rotation decompositions, as memory may only need to be read only for long periods of computation. The qualification is that different applications often require different levels of precision in rotation decompositions, so these databases would potentially need to be changed between apps.

Program Code Generation and Manipulation

The last set of variables being considered are those relating to the generation of instruction code for these quantum benchmarks. These techniques fall into several categories, specifically the relationship between the compilation flattening threshold and code caching behavior, and methods of compressing and decompressing modules.

Roadmap

The analysis will progress in stages. First an abstract model of a classical control unit as a cache is built, along with various methods for communicating program data to and from the cache. In this stage, program modules are extracted from full logical schedules which are compiled with the ScaffCC framework [1], and various cache compression techniques are explored for reducing the minimum required memory size of the control unit. Initial analysis will compare the effect of variable module compression techniques against the required computation introduced by performing the compression and decompression actions, which ultimately affects overall quantum program runtime.

The next stage is the construction of an energy budget cost model, which incorporates cooling capacity constraints of the different temperature interfaces of a dilution refrigerator with the effective energy usage per operation of different hardware materials. The cache model of classical control is expanded by introducing these new costs as constraints on computation, which affects the ability to perform module decompression and maintain the target temperature for a specified region. Techniques for optimizing module compression and decompression will be explored. Additionally, parameters describing latency caused by microwave links traversing thermal boundaries will be introduced as communication costs, which will be incorporated into the cost model.

The following stage will begin to analyze the tradeoffs of different flattening thresholds for code modularization during compilation of quantum programs. Conceptually, flattening quantum benchmarks with a high gate count threshold will allow the compiler to be more efficient and reduce overall runtime, as more context is available during these optimizations. However this increases the size of each module, which increases the complexity and overhead of the caching mechanism. The interaction between these behaviors will be analyzed, and new techniques of both code modularization and module compression and decompression will be explored to optimize quantum program execution.

The ultimate goal of this analysis is to extract and quantify specific tradeoffs with system design choices involving control unit hardware, temperature level placement of control units, and memory sizes of control units. The aim is also to develop novel techniques for optimizing the transmission of large quantum programs through a control unit system, including optimized module compression and decompression methods for both control unit memory size reduction and communication bandwidth requirement reduction.

Stage One: Cache Implementation and Initial Results

Cache Model

The initial phase of analysis is the construction of a model of classical control units as caches for quantum programs. Conceptually, quantum programs are compiled and scheduled with the ScaffCC framework which results in discretization of the programs into modules. Each of the modules compiled such that they self contained, and have no dependencies upon other modules with potential exceptions at module boundaries. The largest uncompressed module size provides a lower bound on the size of the cache memory.

Simulations are performed on a per-program basis. The modules of each program are extracted, and unique binary codes are assigned to individual instructions, quantum register names, as well as module calls themselves. Once each module is converted to binary, each is compressed and decompressed across a series of different compression algorithms, and statistics regarding CPU usage and memory usage are collected with respect to each module. A program call stack is then simulated on control units of varying capacities (parameterized by number of containable modules), and total CPU usage required by decompression of modules is tracked. These statistics are then compared against the overall program runtime to determine significance of the decompression actions. Examples of these simulations are provided in (figure 1) and (figure 2).

Initial Results

Provided here are examples of the first stage statistics. The first set of plots depict the total CPU usage requirements for two different quantum programs, Ising Model and Binary Welded Tree. As can be seen, the CPU usage depends highly on the structure of the program module sizes.

The next analysis compares the CPU usage-padded runtimes against original program runtimes. Provided are two plots, the former (figure 3) of the new program slowdowns due to decompressions, and the latter (figure 4) a scaled view of the same plot. Slowdowns shown here are considerable, higher than 100x for many benchmarks.

This initial analysis is intended to motivate the idea that restricting the memory sizes of control units has a dramatic effect on the runtimes of quantum programs, with dependence upon the internal structure of the program and distribution of module sizes.

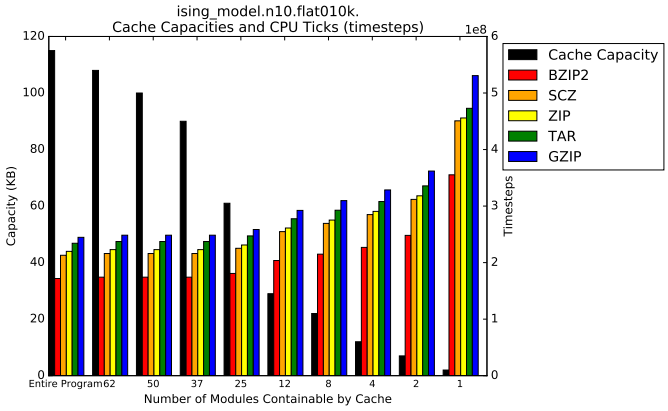

Figure 1: Ising Model

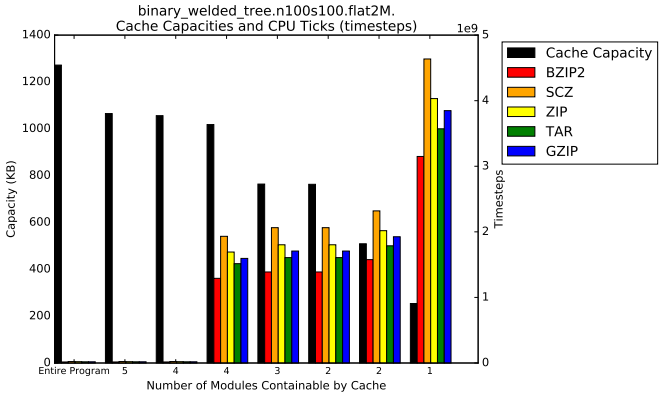

Figure 2: Binary Welded Tree

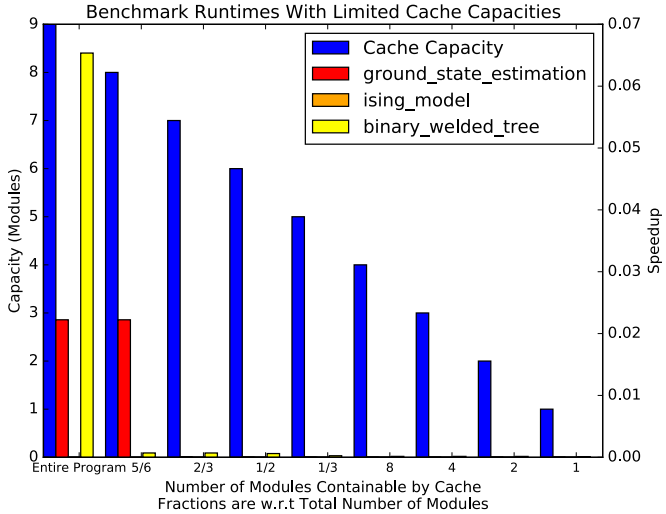

Figure 3: New Program Runtimes with Decompressions

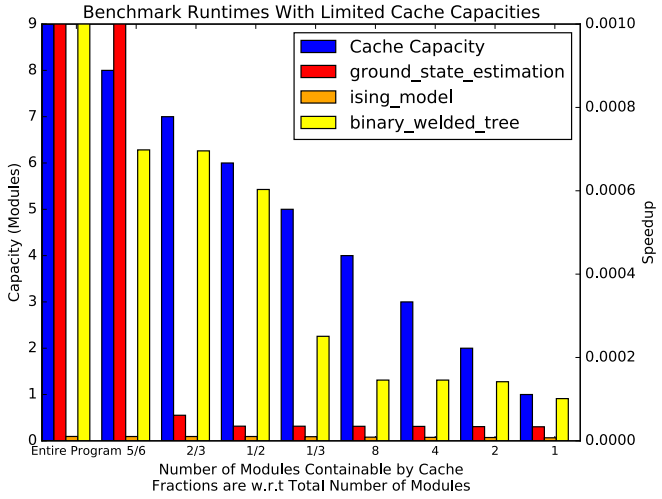

Figure 4: Scaled Depiction of New Program Runtimes

Bibliography

[1] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, and M. Martonosi, "Scaffcc: A framework for compilation and analysis of quantum computing programs," in *Proceedings of the 11th ACM Conference on Computing Frontiers*, ser. CF '14. New York, NY, USA: ACM, 2014, pp. 1:1–1:10. [Online]. Available: http://doi.acm.org/10.1145/2597917.2597939