Moumita Dey
GTID: 903099216
Student ID: mdey6

ECE 8853 – Introduction to Quantum Computing
Lab #3 – Qubit Allocation Policies with Variation Awareness
Spring 2019

1. Minimum number of SWAPs required by baseline SABRE for 9-bit Bernstein-Vazirani key ("11111111") = 5

2. PST for all workloads on SABRE on IBM simulator with noise config file:

Workload	**Initial mapping**	**SWAPs**	**PST**
GHZ8	[8, 9, 10, 4, 3, 2, 1, 11, 13, 5, 6, 7, 0, 12]	0	0.36
QAOA1	[1, 11, 8, 10, 2, 3, 6, 0, 12, 5, 4, 13, 7, 9]	0	0.42
bv9	[9, 11, 4, 12, 5, 3, 2, 13, 0, 10, 1, 6, 8, 7]	5	0.23
decod24-v2_43	[3, 11, 4, 5, 10, 12, 0, 2, 9, 6, 13, 12, 7]	10	0.07
fredkin	[13, 2, 12, 0, 10, 6, 3, 8, 1, 4, 5, 11, 7, 9]	3	0.59
ising_model_10	[1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 7, 0]	0	0.586[†]
qft_10	[6, 5, 8, 4, 9, 3, 10, 2, 12, 11, 13, 7, 0, 1]	32	0.022

3. PST for all workloads on variation-aware SABRE on IBM simulator with noise config file:

Workload	**Initial mapping**	**SWAPs**	**PST**
GHZ8	[11, 10, 9, 5, 4, 3, 2, 13, 1, 12, 0, 8, 6, 7]	0	0.38
QAOA1	[11, 2, 10, 3, 6, 8, 13, 4, 7, 5, 9, 0, 1, 12]	0	0.69
bv9	[5, 8, 10, 4, 11, 3, 12, 2, 13, 9, 0, 1, 7, 6]	3	0.25
decod24-v2_43	[11, 4, 10, 9, 3, 7, 8, 0, 1, 5, 6, 2, 13, 12]	11	0.14
fredkin	[12, 1, 13, 10, 4, 5, 0, 9, 3, 11, 8, 7, 2, 6]	3	0.55
ising_model_10	[8, 9, 10, 4, 3, 2, 12, 13, 1, 0, 7, 6, 11, 5]	0	0.52[†]
qft_10	[12, 2, 13, 11, 3, 1, 4, 10, 5, 6, 7, 0, 9, 8]	23	0.02

Comparison of variation-aware SABRE and baseline SABRE:

Workload	**SWAPs Improvement (%)**	**PST Improvement (%)**
GHZ8	0	5.26
QAOA1	0	39.13
bv9	40	8
decod24-v2_43	-10	50
fredkin	0	-7.27
ising_model_10	0	11.26
qft_10	28.13	-10

*compile time measurements have been excluded in this analysis as baseline SABRE hides/ignores all OS calls and is highly optimized. It also considers the time for only one of the iterations, whereas in reality SABRE and variation-aware SABRE need to be run multiple times to converge. Hence there is no basis of comparison for compile time in this case.

[†] For ising_model_10, Hellinger's distance was used to compute PST by taking PST = (1-distance)

Moumita Dey
GTID: 903099216
Student ID: mdey6

Discussions and Insights:

There were various code modifications implemented.

A **mapper.py** file was incorporated on top of main.py that controls the number of times SABRE needs to be called, whether it is the training phase where it tries to converge to the minimum number of SWAPs, or the optimized initial mapping with minimum SWAPs is run once to obtain the QASM schedule, the initial mapping and SWAPs obtained.

For this, after trial and error, SABRE is run 4 times and iterated 50 times with random initial input mapping. This was done so that 50 times a random initial mapping is chosen, that helps in expanding the sample space, thus, making it easier to converge to a minimum SWAP value, instead of running SABRE multiple times with a single random input.

This, according to me is a drawback as the algorithm is not optimized the take care of all, if not most, possibilities of the starting initial mapping. The more the number of iterations, the better the sample space in this case, hence easier to converge to the optimized run. Thus, SABRE is dependent on the user as to how many times it needs to be iterated.

The **utils.py** file was modified to incorporate all types of instructions including unitary qubit gates. In addition, a gate dependency list was made that stores the dependencies of the unitary gates on any previous CNOTs using the same qubits. This is essential as SABRE does take care of CNOT-CNOT dependency, but fails to check for single-qubit gates, which need to be scheduled once the CNOT or SWAP+CNOT instructions have been executed to make sure that the following dependent single qubit-gates correctly follow the new SWAPped architecture.

The **main.py** file was majorly modified.

Scheduling policy –

Baseline SABRE doesn't take into account the instruction PC when it formulates the DAG, hence the concept of instruction ordering is lost. Hence the baseline SABRE needed to be modified so that instructions can be scheduled in order, but also take into account the ordering of scheduling of the CNOTs and CNOTs+SWAPs as per availability. The DAG was thus modified to append the program counter associated with each gate. Once the SWAP is inserted, the CNOT is scheduled. After scheduling the CNOT, the scheduler checks for any single-qubit instructions using the qubits in the CNOT gate after the CNOT gate instruction. Once the last dependent single-qubit instruction is found, the sample space between the CNOT and the last single-qubit instruction is searched to find any instruction that has no dependency with any of the CNOT gates. These independent instructions are then scheduled. To avoid scheduling instructions multiple times, an execute flag was added to each instruction that is marked once an instruction has been already scheduled.

Remapping qubits after SWAPs –

One important step to consider is the remapping of the logical qubits after a SWAP which usually the IBM compiler adds additional instructions along with the SWAP it inserts to maintain code sanity. However, as we are manually inserting the SWAPs, to avoid any spurious instructions being added by the IBM compiler, we manually take care of the remapping logic for the qubits. A remap table was designed that keeps a track of the actual qubit mapped to the original logical qubit. This remap table is updated with the SWAP every time a SWAP is inserted.

Modifying cost function to incorporate variation-aware SABRE –

Moumita Dey
GTID: 903099216
Student ID: mdey6

To make SABRE variation-aware, the error rates provided were parsed and stored in a matrix based on the connections between the 2 qubits. To incorporate the concept of variation-aware policy, the cost function was modified to a "weighted distance" policy. Instead of simply considering the distance between two physical qubits, distance*error between two qubits is used. This gives the notion of weighted distance. This was the variation-aware cost function implemented for choosing and scheduling SWAPs.

Dump schedule –

Finally, the code was modified to dump out the qasm file based on the instructions being scheduled.

Comments –

The simulator code provided is extremely buggy and doesn't work on its own without making 10 fixes on its own. It's poorly commented and coded, extremely difficult to understand what's happening underneath, as half of the things mentioned in the paper isn't coherent with the code presented. The workloads provided also aren't in any supported format with SABRE, whose own file parsing capabilities are questionable. It would have been really useful if at least the baseline code was provided correctly, instead of us breaking our heads and spending half of the time fixing something that we have never seen before.

The point distribution is also extremely skewed. For part 2, majority of the work needs to be done to get the scheduling policy to work and incorporate. This is almost the entire essence of this assignment. Making it variation aware is simply some minor tweaks and modifications that we play around to observe the improvements. The weightage and amount of work needed for each stage should be at least hinted properly before providing the assignment with random scoring policy.