Weekly Report 07/31/2018

1、**Reading List**

- Alwin Zulehner, Alexandru Paler and Robert Wille, “An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures”, in IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD), 2018

2、**Ideas**

*(1) **Bridge Model***

I have finished the basic model of bridge transformation before last week. It has much better effects than the algorithm only using swap, but is worse than the algorithm proposed by the paper above (we can call it Efficient Layer algorithm). So I come up with several ways to improve my bridge algorithm.

➢ **Change the way of initial mapping**

The original initial mapping function is to map the pseudo qubits appearing in the sequence with higher frequencies to the physical qubits with higher out-degree. I find this mapping function cannot take full use of the advantages of bridge transformation. And when the pseudo qubits number is small, it will take extra costs to move the qubits together. This is a big flaw when the qubits number is less than 5.

To take use of the bridge which enables a qubit to interact with more qubits without changing its location, we can map the pseudo qubits with higher frequencies to the center area of the map so they can interact with more qubits. For those less frequent qubits, we map them to the marginal area sequentially according to their frequencies.

In this way, bridge can show its power better. What’s more important, this mapping strategy is suitable for all possible pseudo qubits number because when the qubits number is small, all the qubits naturally get together in the center area.

➢ **Cut layers in the sequence**

Drawing experience from the paper, we can also cut layers for the CNOT gate in the sequence. A layer is a set of CNOT gates which can be implemented in any order. It requires that all the operands of CNOT(x,y) in one layer cannot repeat. Bridge can decrease the latent swap in the next operation, so it is sensitive to the implementation order.

Therefore, we can try each possible order for implementing the CNOT gates in a layer and choose the one with least gate cost. If the layer has n CNOT gates, its all sequence permutation has totally n! situations. Actually the n of a layer is always small in a real benchmark so this strategy will not be computationally expensive.

➢ **Offset the Hadamard gates in the sequence**

Bridge is useful because the Hadamard gates introduced by reversal transformation can offset each other. The equivalent circuit only remains the CNOT gates in the center and several peripheral Hadamard gates. Actually the peripheral Hadamard gates can offset with the Hadamard gates in the sequence as well if there’s no other gate between them. It’s fortunate that all the Hadamard gates in all the benchmarks appear only after CNOT gates. This property is decided by the usage of Hadamard gate. So this kind of offset can reduce the total gate cost efficiently.

*(2) **Error Model***

- My crosstalk-oriented algorithm:

Just take consideration of crosstalk error and the crosstalk error of each physical qubit is in proportion to the sum of coupling strength offered by IBM github. My optimization is to save all the single-qubit gates on a physical qubit and do not execute them until the next CNOT gate on it. Then find the physical qubit with least crosstalk error on its swap path and execute all the single-qubit gates on that qubit.

The original error model has some mistakes because I forget to clear the sequence array after each iteration. After correcting the mistakes, the error model works well.

I set the crosstalk error relatively small, the minimal crosstalk cost is about 1 gate and the maximal crosstalk cost is about 2 gate. For the swap-only algorithm, the average reduction on ibmqx5 is 14%. And for my bridge algorithm, the average reduction is 10%. This algorithm can be more powerful if the equivalent gate cost of crosstalk error is bigger. For example, when I set the difference between the crosstalk error if the most and least reliable physical qubits to be a swap cost (7 gates), the swap-only algorithm can reduce 35% gates.

3、 <u>**Deliverables**</u>

All the ideas mentioned above have been implemented and all the codes are on my github (https://github.com/tilmto/QuantumComputing/tree/master/Source) . I also execute the program offered by the paper mentioned above to make a comparison.

I extract the data from the results and list all the consequences in a form which can be find on my github (https://github.com/tilmto/QuantumComputing/tree/master/Experiments).

The bridge model has much better performance compared with the one without optimization. Among the 156 benchmarks, my model get better performance on 106 compared with the Efficient Layer algorithm. The average ratio between #gate_bridge and #gate_efficient_layer is 93.5%.

4、 <u>**Plan**</u>

All the optimization ways I have come up with up to now have been implemented. If the result is not ideal enough, I can try the way of dynamic initial mapping to find the best initial mapping for individualized benchmarks with few or many qubits, short or long operations. Additionally, an advantage of bridge model is that it can be combined with other algorithms. So I can try to use other algorithms and bridge at the same time.