# Accelerated quantum circuit simulation on reconfigurable devices - detail on the proposed algorithm

Thomas Parks

## A slightly clearer example

### The path integral interpretation of quantum mechanics

Feynman introduced a interpretation and corresponding mathematical formalism for evaluating the time evolution of a quantum state as a integral over all possible histories that of the system.

In classical mechanics, lagangian is used to find closed form solutions for the path of particles - but not really, for a double. pend. the nested differentials give intractable solutions.

FM. postulates:

1. The probability for an event is given by the squared modulus of a complex number called the "probability amplitude".

- The events we want to sample are full final computational basis states. A computational basis state is equivalent to a bit-string. This is the same as finding the probability of observing a given bit-string as output.

2. The probability amplitude is given by adding together the contributions of all paths in configuration space.

- We start with a state, and evolve it in a maximum likelihood manner to generate a sequence of intermediate states of the circuit. In the process of finding the intermediate states, we use this property to fully evaluate the quantum contributions to each gates operation. As each gate operates on a limited qubit set, we explore a 1-ball in basis space to find relevent possible histories.

3. The contribution of a path is proportional to eiS/ħ, where S is the action given by the time integral of the Lagrangian along the path.

- Obtaining the Lagrangian is explained in [1]. In a full writeup I will need to explain the introduction, descriteisation, and quantum circuit interpretation that this explains really well.

## Hardware implementation, pipeline filling, bounding instant state, and non-interacting subsystems.

The path integral method offers a way to calculate the probability of a final state by working backwards (and recursively).

In order to determine the probability of a given state, we need (basically the state of the qubits that interact with the last gate with rest traced out) the classical states of the input qubits. A 1-ball around the state of interest.

We propose to compute final state probabilties as follows. We instantiate a network of matrix operators over 3-qubit input states, representing individual gate types present in the circuit.

A set of top-level driver blocks that follow the initial state through the system. Each takes "responsibility" for solving the forward and reverse problems for some subset of the circuit.

The driver block containing the input state calculates the (quantum) next state (by passing the e-ball of all input states to the block), and takes n samples biased by the classical observation probability of each resulting state in comp. basis.

In order to calculate the output state, the probabilities of all input states in a 1-ball need to be evaluated.

The block performs a reduction, reducing list of n resulting states to calculate into a m<=n list of states and repeat counts. Loop until final output state is being produced.

We need to transition between blocks. This basically means adding a request to either calculate forward from a state or backward to a queue for each block. Hope for limited communication between blocks so can allocate on blades.

### Implementation/Optimisation details

The blocks are 3qubit ops, so 27x27 complex matricies. Build with kiwi. Pipeline: either have host block track when output expected or (as var latency perhaps) associate with a timestamp. Timestamps would be only unique within control blocks, and limited in length to the "stack size" of the control blocks.

We also hope to opportunistically exploit redundant calculation. If we stored the result of every partial calculation, we would not need to recompute any histories. Unfortunately this will require memory approximately the same as storing the entire state vector. Rather than caching results, the control blocks can fold requests to compute the same history.

When computing a history, you can either go depth or breath first as computing this history forms a computational tree. This algorithm is computing multiple tress in parallel, and so we want a exploration strategy that leads to maximum synchronous overlap of executing computation.

Probably neither depth or breadth first really affects the probability of intersection, but delaying computing a result for some time will give a larger window for finding overlap. This is probably memory and time equivalent to caching results for limited time, a optimisation identified by [2].

I would like to do some stimulations to see if tree exploration strategy has any benefit. Also, the tree can probably be transformed into a convex exploration space that could benefit from loop transformation approaches.

## Comparison with the 49-qubit simulation

The most recent fully general quantum circuit simulator[3] is able to compute the full output state for 49 qubits in a random circuit. The tensor reduction method used requires a large amount of memory, and whilst it is possible to compute the output probabilities for for a arbitrary subset of output states to scale further it was not shown that the output state selection could be biased to states with high probability.

Our proposed method using a pilot wave to guide evaluation is biased towards finding the probabilities if likely output states. This is useful for quantum algorithms that attempt to use entanglement to amplify the desired result.

[1] B. Rudiak-Gould, “The sum-over-histories formulation of quantum computing,” *arXiv preprint quant-ph/0607151*, 2006.

[2] M. P. Frank, U. H. Meyer-Baese, I. Chiorescu, L. Oniciuc, and R. A. Van Engelen, “Space-efficient simulation of quantum computers,” in *Proceedings of the 47th annual southeast regional conference*, 2009, p. 83.

[3] E. Pednault *et al.*, “Breaking the 49-qubit barrier in the simulation of quantum circuits,” *arXiv preprint arXiv:1710.05867*, 2017.