

Pipelining in QCA

And how it can be implemented in an FPGA

Quantum Dots

- Quick review: QCA(quantum cellular automaton consists of a sequence of QCA cells, each with 4 quantum dots
- Electrons occupy opposite corners of the quantum dot--2 stable states
- Coulombic repulsion will cause the 2 electrons to tunnel to the right corners



More review...

- The predominant gate used is a 3 input majority gate (although, we'll see Nand later)
- By fixing one of the inputs as a 0, we get an and gate. Fixed as a 1, it's an or gate
- Easy inverters--just offset!
- By switching the orientation of the QCA cells, we can have planar crossing "wires"

We have a complete basis!!

Issues...

- QCA cells work at LOW temperatures
 - We're up to 70 K now, well on our way...
- Propagation might occur in BOTH directions: each QCA cell affects both sides
 - We need to somehow activate & deactivate cells...
- There's a delay in propagation across QCA cells, unlike for CMOS
 - Okay, so a clock is clearly needed...

What do we mean by a clock?

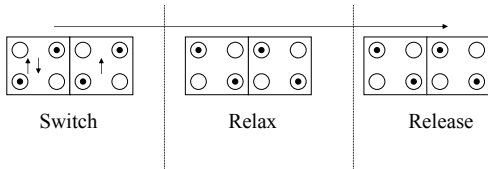
- A clock produces a signal switching between high & low
- Individual components in the circuit are “clocked”--one of the inputs is this signal
- A clock makes sure that everything updates at once.
- A clock doesn't do anything besides synchronization

A QCA clock...

- Divide the entire layout into “clocking zones”.
- There's no “signal”--each zone is in one of 4 phases: Switch, hold, release, relax
- Barriers for electron tunneling: raised, stay raised, lowered, stay lowered.
- Critical: make sure QCA cell doesn't affect previous ones!!

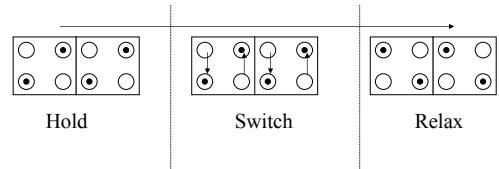
Step through phases...

- Let's assume that QCA cells are in the configuration below, and the phases below
- Remember zones are clocked not individual cells
- Watch the middle zone--in the next “switch”
- Wait, why are the zones in that order?



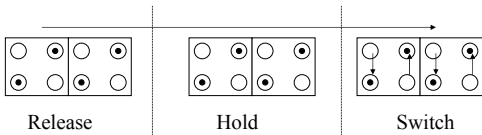
Switching time!

- Now the middle zone is allowed to switch
- This is because of raised barriers-> tunneling
- They're in a polarized state
- This phase is where **computation** occurs
- [note: all the zones are now in their next phase]



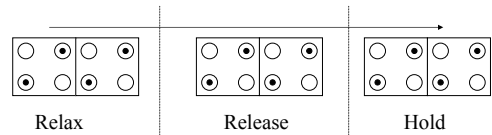
Hold it right there!

- Now the middle zone is in the hold phase
- That is barriers are held high--it can't be affected, but its values can be used as input
- We need to make sure this phase is long enough to cause the cells in the next zone to "switch"



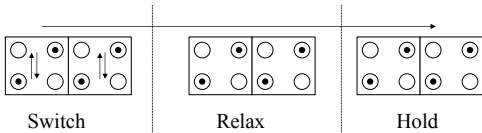
Ahh..release!

- Now the middle zone is in the release phase
- That is barriers for electron tunneling are lowered
- Cells start becoming unpolarized.
- Electrons won't start tunneling because the input cell hasn't changed yet...



Relax, we're done!

- Now the middle zone is in the relax phase
- That is barriers for electron tunneling are held low
- Cells are unpolarized, so they won't react to switching going on in previous cell
- Now this cell can start switching again...



Clock physics

- Metal lines underline the array, larger than the cells--acts as an energy source
- Electric field moving across the QCA cells
- Clock rate for 5 cells: 2 THz /51 cells: 20Ghz
- Switches at approximately 10^{-12} s
- There's power gain, which is needed for restoring logic levels in circuits

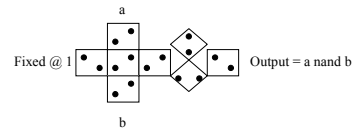
What else can a clock do?

FPGA's

- An FPGA(field programmable gate array) is a collection of functionally complete logic element arranged in a network
- It's programmable, so we want to be able to route signals the way we like
- It's functionally complete, so we have to choose an appropriate logic element.

Logic element

- Let's start with the easier problem--what QCA element can be used as the base logic element?
- A nand gate (and->invert) of course!
 - $\neg a = a \text{ nand } a$
 - $a \wedge b = (a \text{ nand } b) \text{ nand } (a \text{ nand } b)$
 - $a \vee b = (a \text{ nand } a) \text{ nand } (b \text{ nand } b)$



Routing problem

- CMOS uses SRAM to configure “pass transistors”, which allow current to flow
 - memory is tricky in QCA, and we want simplicity
- We could try using multiplexers to select between paths
 - a single multiplexer takes up 6 majority gates (?)
 - again, we can't store a selection bit

Back to the clock...

- Remember the relax phase of the clock cycle?
- Cells are unpolarized, so effectively disabled
- To route, have some zones unlocked in “relax”
- But, there are difficulties here as well:
 - Large clock zones: all cells in it might not switch
 - Small clock zones: takes several cycles to reach

Clocking zones

- Since there are 4 phases, we'll have zones in blocks of 4.
- We want a min of clocking zones, because the more we have the slower it will take
 - because we have to wait for each cell to switch
 - because each zone has to go through 4 phases
- However, adjacent regions should not interfere with each other

1 x1, 2 x2...

- A 1 x 1 zone won't work because there's no routing possible (can't disable any part of it)
- With a 2 x 2, we can have routing, but they might interfere with another routing element
- A 3 x 3 element does the job, because we can route without interference, but can we do better area-wise?

A 3 x 2 zone router

- If we have 3 vertically, and 2 across, then we can have signals propagated vertically.
- Adjacent blocks will not be affected (obviously there won't be horizontal interference)
- We've saved 1/3 of the size (vs. 3 x 3)!
- Also, we might be able to combine multiple clocking elements into a single zone

Putting it together...

- So now, we have our fundamental logic element (nand), and we've also decided the zone size for router (3 x 2)
- Unlike CMOS technology, QCA is "pipelined"
- There are shortcomings--no memory!
 - We can't have a flip-flop, how about 2 inverters (?)
 - The hold phase can be used to preserve values (?)

Constructing architectures

- Now let's move on to bigger architecture
- First of all, QCA is an order of magnitude denser than CMOS, 3 orders at the molecular level--size advantage
- We have to keep clocking zones small as mentioned earlier for speed & efficiency
- A "1 hot" machine has been designed but not implemented YET

A 1 hot machine

- A 1 hot machine has either stop, iFetch or execute "hot", or in the 1 state, with a flip-flop for each state
- Instead of an actual flip-flop, we're simply having certain zones in the "hold" phase
- Difficulty: feedback paths must not be delayed by several clock phases (zones)

Argh! But then our clocking zones get too big!

Simultaneous Switching

- We need some way of switching simultaneously.
- With CMOS, this was easy, everything runs by a standard clock
- In QCA, we might be able to try putting everything that's simultaneous in the same clocking zone->change simultaneously

1 hot examples

- Again, these aren't actually constructed (theoretical so far), but look at fig 8,9, 10
- Fig 8 has the feedback delay problem
- Fig 9 fixes this w/ reduced clocking zones
- Fig 10 shows how data can be latched at once
- discussions?

Conclusion

- So the paper covers more on architecture
 - how much did you understand?
- We've seen how critical the clock is in solving routing, timing, & storage problems
- The pipelined architecture of QCA introduces new problems not found in CMOS devices
- Next step: memory!

Oh, and ACTUAL room temperature QCA devices would be nice too!