2^n Uses for a Live/Dead Cat

Physicists describe the value of a quantum bit, or ***qubit***, as a continuous wave function that defines probability amplitudes for 0 and 1 in terms of coordinates (θ, ϕ) on a Bloch sphere. Computationally, that means a qubit can represent the value 0, 1, or a ***superposition*** of both. By having wave functions interact, multiple qubits can be ***entangled*** so their values are linked, or their values can be made to ***interfere***. Thus, a quantum computer has the exciting potential to act upon up to 2^n values by performing just a single operation.

College of Engineering
Electrical and Computer Engineering

SEE WHAT'S *Wildly Possible*

Parallel bit pattern computing isn't quantum computing nor does it use qubits, but it does provide very similar computational properties. It's a new model of computation dramatically reducing power by minimizing active gates.

Let's start with how conventional logic can efficiently model superposition and entanglement. Each *n*-way entangled pattern bit, or ***pbit***, is an ordered set of 2^n single-bit values; entangled values are grouped together by their position. For example, two pbits with the value {0, 1, 2, 3} and two pbits with the value {3, 1, 3, 0} represent the entangled set of value pairs {(0,3), (1,1), (2,3), (3,0)}. Adding these values produces the 2-way entanged 3-pbit value {3, 2, 5, 3}:

0	1	2	3
0	1	0	1
0	0	1	1

+

3	1	3	0
1	1	1	0
1	0	1	0

=

3	2	5	3
1	0	1	1
1	1	0	1
0	0	1	0

That is the compute mechanism behind KREQC, Kentucky's Rotationally Emulated Quantum Computer, shown at SC18 and live online via **`http://aggregate.org/KREQC`**:

Each gate-level operation in that 6 "Q"-bit machine is efficiently processed using SWAR parallelism on a conventional processor. In fact, that approach also is used for the 16 "Q"-bit KREQC in our SC19 exhibit. To be precise, an old laptop executes fully-entangled 16-pbit computations in real time and drives servos so the angle of the physical "Q"-bits displays the probability of 1 vs. 0. However, the parallel bit pattern model of computation can do much more.

Instead of representing a pbit value as 2^n bit positions, we can represent it by a ***regular expression*** that can generate the ordered set of bit values. For example, {0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1} could be $0^6 1^4 0^4 1^2$, but we actually use symbols much larger than individual bits. Not only does this reduce memory footprint, but by *operating directly on regular expressions*, each gate-level operation can effectively compute up to exponentially many results.

There are many subtle differences between pbits and qubits. Pbits don't have wave functions per se, but bit patterns are analagous to waves and a variety of interference-like pbit properties can be efficiently computed. Unlike qubit gates, pbit gates do not need to be adiabatic, so classical gates including AND, OR, and XOR are available. The results of pbit computations are always coherent, free of noise, and can be measured without collapsing. Probabilities of discrete values theoretically could be arbitrary for qubits, but are always in parts per 2^n for an *n*-way entangled pbit.

Pbit computation also uses runtime compiler analysis and transformations to dramatically reduce active gate counts. Thus, there is almost no penalty in programming at the pattern int, or ***pint***, level. For example, the following uses 310 gate-level pbit operations to compute sqrt(29929):

```
main() {
  pbit_init();
  pint a = pint_mk(16, 29929); //16-pbit value 29929
  pint b = pint_h(8, 0xff);    //all 8-bit values
  pint c = pint_mul(b, b);     //square them
  pint d = pint_eq(c, a);      //where square is 29929
  pint e = pint_mul(d, b);     //make non-sqrt's all 0
  pint_measure(e);             //prints 0, 173
}
```

There's a lot more to parallel bit pattern computing. We're several years into this research, but still in very early stages of working-out all the details, prototyping, and evaluating. The best current reference is Henry Dietz, *Parallel Bit Pattern Computing*, 10th International Green and Sustainable Computing Conference (IGSC), Oct. 21, 2019; a preprint version is linked at **`Aggregate.Org`** and **`QERKY.Org`**.

This SC19 white paper should be cited as:

```
@techreport{sc19pbp,
author={Dietz, Henry},
title={{$2^{n}$ Uses for a Live/Dead Cat}},
institution={{University of Kentucky}},
address={{http://aggregate.org/WHITE/sc19pbp.pdf}},
month={Nov}, year={2019}}
```

SC19 Denver, CO | hpc is now.

Aggregate.Org UNBRIDLED COMPUTING