A Quantum-Inspired Model for SIMD-Parallel Computation

LCPC 2020
1:40-2:00 October 15, 2020

Henry Dietz, Aury Shafran, & Gregory Murphy
Electrical & Computer Engineering
LCPC 2017:

**How Low Can You Go?**

- Now it’s all about **power / computation**
- Work only on **active bits** (bit-serial)
- Aggressive **gate-level optimization**
- Potential exponential benefit from **Quantum**?
Gate-Level Optimization

```c
int:4 a, b; b=1; c=a+b;
```

becomes 17 or 7 gate operations:
Gate-Level Optimization

```c
int:8 a, b, c;
a = (c * c) ^ 70;
a = ((a >> 1) & 1);
a = b + (c * b) + a;
a = a + ~(b * (c + 1));
```

- About 206,669 gates unoptimized
- Optimized, it’s just `a = 0;`
Quantum Computing

- **Superposition**: 1 qubit, all values
- **Entanglement**: $e$ qubits, $2^e$ values
  - Exponentially less memory
  - Exponentially fewer gate ops
Quantum Computing

- **Superposition**: 1 qubit, all values
- **Entanglement**: $e$ qubits, $2^e$ values
  - Exponentially less memory
  - Exponentially fewer gate ops
- **Limited coherence**, no cloning, only reversible logic gates, ...
Encoding e-way Entanglement

- Array of Values (AoV): array of $2^e k$-bit values
- Array of Bits (AoB): $k 2^e$-bit arrays
- Array indices are entanglement channels
Bit Pattern REs: A Full Adder

\[ a = H(0); \]
\[ b = H(1); \]
\[ \text{cin} = H(2); \]
\[ \text{sum} = \text{xor} \left( \text{xor}(a, b), \text{cin} \right); \]
\[ \text{cout} = \text{or} \left( \text{and}(a, b), \text{and} \left( \text{xor}(a, b), \text{cin} \right) \right); \]

Simplify REs, e.g., by run length encoding (RLE):

\[ (01101001)^+ \rightarrow (0^{1}1^{2}0^{1}1^{1}0^{2}1^{1})^+, \]
\[ (00010111)^+ \rightarrow (0^{3}1^{1}0^{1}1^{3})^+ \]
Parallel Bit Pattern Computing

- Operate directly on compressed REs
  - Up to exponential reduction in storage, gate ops
- Avoids major quantum problems:
  - Forever coherent, error free
  - Cloning: fanout, non-destructive measurement
  - Use any gates, not just reversible logic
  - We know how to build scalable hardware
Where’s the Parallelism?

• 32-way entanglement: AoB is 4294967296 bits
• Each RE symbol is a 4096-bit parallel chunk

<table>
<thead>
<tr>
<th>Time</th>
<th>Work</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>SWAP, ALL, ANY, measurement</td>
</tr>
<tr>
<td>1\ldots2^{20}</td>
<td>1\ldots2^{20}</td>
<td>DUP, POPulation count, ...</td>
</tr>
<tr>
<td>1\ldots2^{20}</td>
<td>1\ldots2^{32}</td>
<td>CSWAP, AND, OR, XOR, ...</td>
</tr>
</tbody>
</table>
The Programming Model

• Two programmer-visible layers of abstraction:
  ◦ pbit – like the earlier example, but pbit_
  ◦ pint – variable precision (un)signed integers

• Just-in-time optimizing compilation lowers pint to pbit and aggressively optimizes pbit gate DAGs, which get lazy evaluated... details in the paper
pint Sqrt(29929): 310 gate ops

```c
int main(int argc, char **argv) {
    pint_init();
    pint a=pint_mk(16,29929); // 16-bit 29929
    pint b=pint_h(8,0xff);    // H(0)..H(7)
    pint c=pint_mul(b,b);     // c=b*b, still 8-way
    pint d=pint_eq(c,a);      // where c==29929
    pint e=pint_mul(d,b);     // make non-sqrts 0
    pint pint_measure(e);     // prints 0,173
}
```
pint factor(221)

int main(int argc, char **argv) {
    pint_init();
    pint a=pint_mk(8,221);   // 8-bit 221
    pint b=pint_h(8,0x00ff); // H(0)..H(7)
    pint c=pint_h(8,0xff00); // H(8)..H(15)
    pint d=pint_mul(b,c);   // d=b*c, now 16-way
    pint e=pint_eq(d,a);    // where d==221
    pint f=pint_mul(e,b);   // make non-factors 0
    pint pint_measure(f);  // prints 0,1,\textbf{13},17,221
}
Implementation Layers

- **Chunk**: 4096-bit AoB, 12-way entanglement
- **FBP** (factored bit parallel): REs of chunks
- **Pbit** (pattern bit): DAGs of gate-level operations
- **Pint** (pattern int): 1-32 pbit DAGs
- **C++** wrapper: not yet complete
Conclusion & Future Work

● Parallel Bit Pattern computing is
  ○ Disturbingly competitive with quantum
  ○ Fully implementable at scale NOW

● Working on…
  ○ Improving prototype software, C++ wrappers
  ○ GPU version, **Tangled** Verilog architecture

● Automatic parallelization for this target?
16 pbits (Q-bits display)
16-way
Entangled
AoB execution