Quantum Gates - Hardware Implementation

ECE 572, Fall 2001

Goran Negovetic

Quantum Gates

- Quantum Circuits (QC) are composed of quantum gates
- Quantum computation results in exponential amount of calculation in a polynomial amount of space and time
- This leads to Quantum Parallelism
- How to implement QCs (finite precision, observation etc).

Qubit

- Qubit is the elementary unit of storage
- Qubit has three states: zero, one, and superposition of zero and one
- Superposition is expressed as amplitude of complex probability
- 2^N superposition states for N Qubits

Quantum Gates

- Quantum gates can be represented as matrix transformations.
- Multiplication of 2^N vector by 2^N x 2^N matrix.
- One bit unitary operations:

$$\mathbf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \mathbf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad \mathbf{W} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Quantum Gates (cont)

- Many quantum operations can be implemented as register content transfer
- Controlled NOT (XOR) gate:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ a_3 \\ a_2 \end{bmatrix}$$

Swap Amplitudes a_2 and a_3

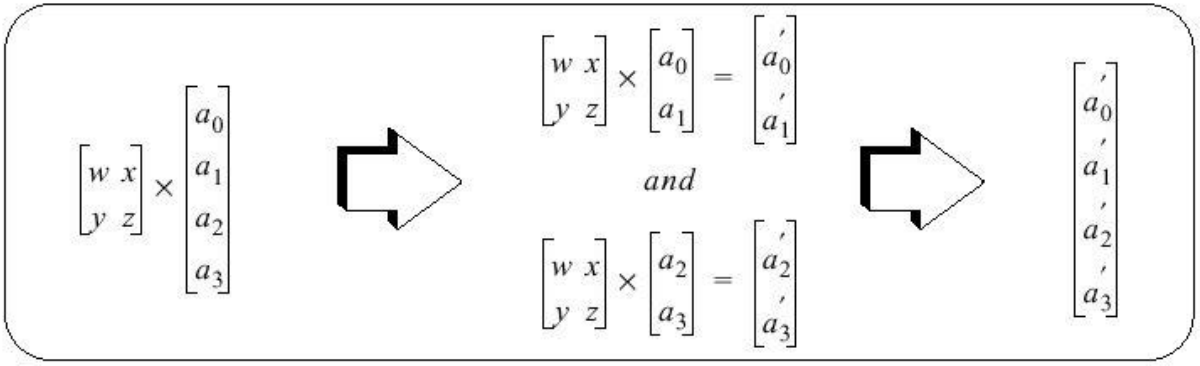
Matrix Operations

- Quantum operations consist of:

1. Complex number multiplication
2. Kronecker product
3. Register content transfer
4. Matrix splitting

Matrix Operations (cont)

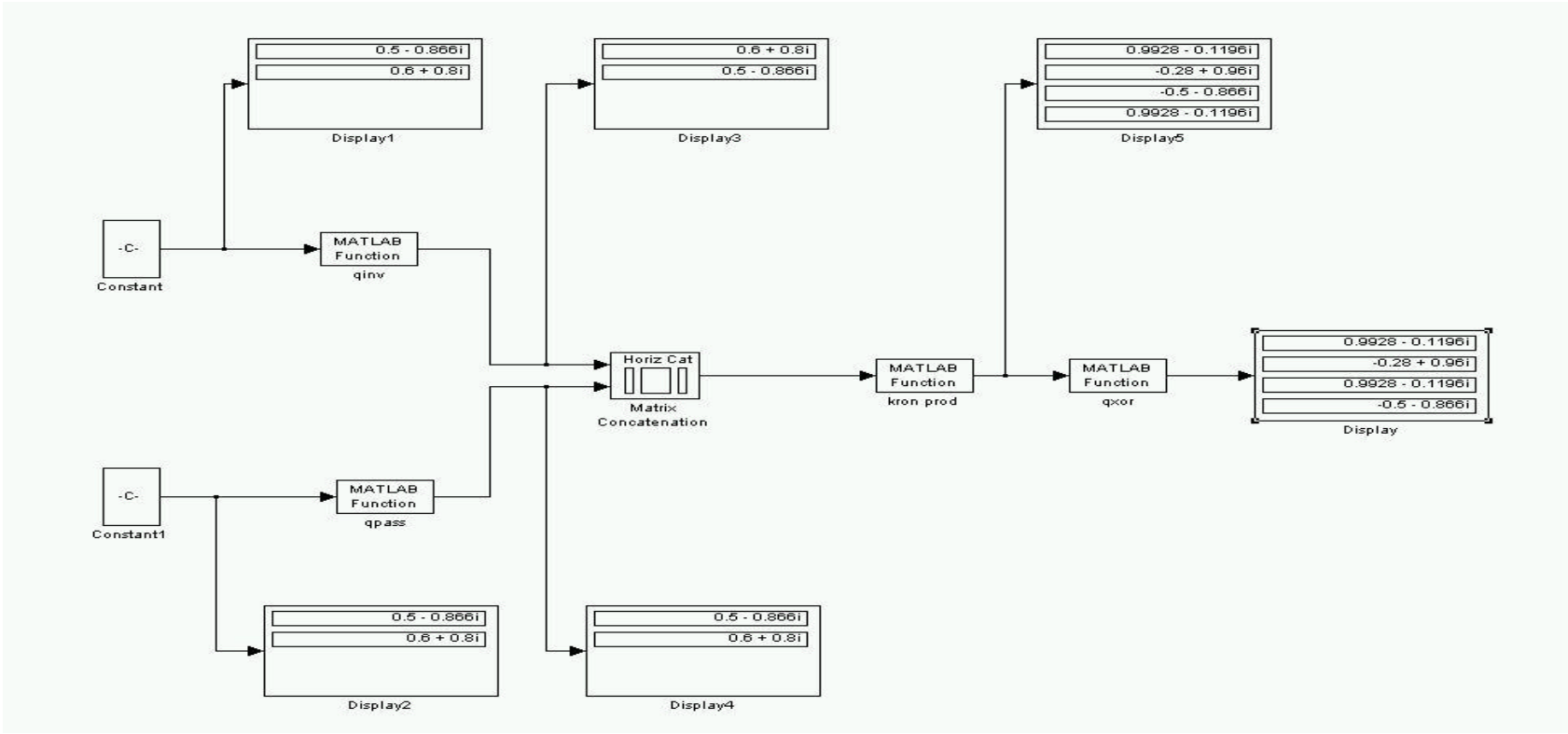
- Each elementary gate is a 2x2 or 4x4 matrix.
- Operation on a large vector space is broken into smaller chunks:



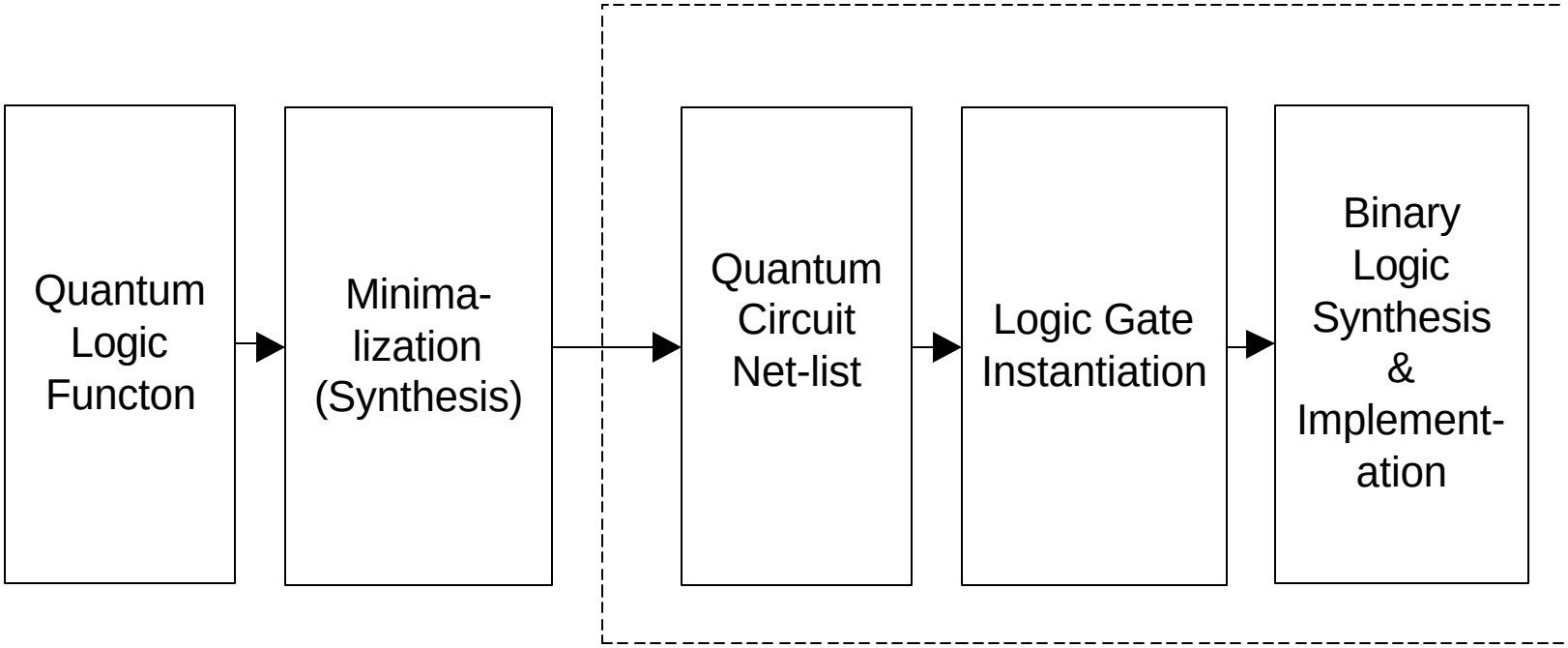
Example 1

```
Had = sqrt(2)*[1 1; 1 -1];
inv = [0 1; 1 0]; %Pauli_X
Pauli_Y = [0 -1i; 1i 0];
Pauli_Z = [1 0; 0 -1];

phase = [1 0; 0 1i];
pi_8 = [1 0; 0 exp((pi/4)*i)];

% 2 x 2
xor = [1 0 0 0; 0 1 0 0; 0 0 0 1; 0 0 1 0];
swap = [1 0 0 0; 0 0 1 0; 0 1 0 0; 0 0 0 1];

conZ = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 -1];
conPhase = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 i];
```

Example 2


0.5 - 0.866i
0.6 + 0.8i
Display1
0.6 + 0.8i
0.5 - 0.866i
Display3
0.9928 - 0.1196i
-0.28 + 0.96i
-0.5 - 0.866i
0.9928 - 0.1196i
Display5
-C-
Constant
MATLAB Function
qinv
Horiz Cat
Matrix Concatenation
MATLAB Function
kron prod
MATLAB Function
qxor
0.9928 - 0.1196i
-0.28 + 0.96i
0.9928 - 0.1196i
-0.5 - 0.866i
Display
-C-
Constant1
MATLAB Function
qpass
0.5 - 0.866i
0.6 + 0.8i
Display2
0.5 - 0.866i
0.6 + 0.8i
Display4

Proposed design flow



Design Flow (cont)

- My goal:

1. Implement quantum gate primitives in Verilog
2. Implement other necessary matrix operations (Kronicker product, complex multiplication etc.)
3. Make a utility program to translate the Quantum circuit net-list to the library gate instantiation.

Design considerations

- Data representation

- Unsigned multiplication and sign logic
- 1, 0.99998, … , 0.0000305, 0
- Virtex II architecture: up to 8M gates, 1.5 Mb SRAM, 18 x 18-bit multipliers, CLA adders

Conclusion

- Limitation: Simulation of quantum circuits is memory/resource exhaustive
- Similarity to unitary transforms used for Signal Processing

Acknowledgment

1. A Parallel Quantum Computer Simulator, Kevin M. Obanland and Alvin M. Despain
2. Quantum Computation: Theory and Implementation, Edward S. Boyden
3. Efficient Quantum Transforms, Peter Hoyer
4. Elementary Gate for Quantum Computation, Adriano Barenco et al.