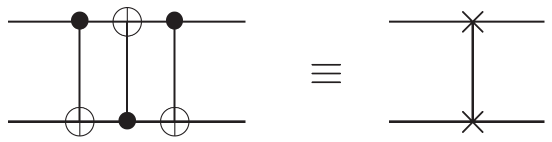

Figure III.11: Diagram for swap [from Nielsen & Chuang (2010)].

C.3 Quantum circuits

A *quantum circuit* is a sequential series of quantum transformations on a quantum register. The inputs are usually computational basis states (all $|0\rangle$ unless stated otherwise). *Quantum circuit diagrams* are drawn with time going from left to right, with the quantum gates crossing one or more "wires" (qubits) as appropriate. The circuit represents a sequence of unitary operations on a quantum register rather than physical wires.

These "circuits" are different in several respects from ordinary sequential logic circuits. First, loops (feedback) are not allowed, but you can apply transforms repeatedly. Second, FAN-IN (equivalent to OR) is not allowed, since it it not reversible or unitary. FAN-OUT is also not allowed, because it would violate the No-cloning Theorem. (N.B.: This does not contradict the universality of the Toffoli or Fredkin gates, which are universal only with respect to logical or classical states.)

Fig. III.9 (right) on page 104 shows the symbol for CNOT and its effect.

The swap operation is defined $|xy\rangle \mapsto |yx\rangle$, or explicitly

$$\text{SWAP} = \sum_{x,y \in \mathbf{2}} |yx\rangle\langle xy|.$$

We can put three CNOTs in series to swap two qubits (Exer. III.32). Swap has a special symbol as shown in Fig. III.11.

In general, any unitary operator U (on any number of qubits) can be conditionally controlled (see Fig. III.12); this is the quantum analogue of an if-then statement. If the control bit is 0, this operation does nothing, otherwise it does U. This is implemented by $|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U$. Effectively, the *operators* are entangled.

Suppose the control bit is in superposition, $|\chi\rangle = a|0\rangle + b|1\rangle$. The effect

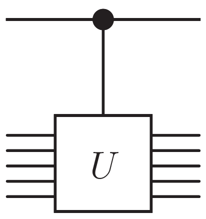

Figure 1.8. Controlled-U gate.

Figure III.12: Diagram for controlled-U [from Nielsen & Chuang (2010)].

of the conditional operation is:

$$\begin{aligned}
&(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U)|\chi, \psi\rangle \\
&= (|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U)(a|0\rangle + b|1\rangle) \otimes |\psi\rangle \\
&= |0\rangle\langle 0|(a|0\rangle + b|1\rangle) \otimes I|\psi\rangle + |1\rangle\langle 1|(a|0\rangle + b|1\rangle) \otimes U|\psi\rangle \\
&= a|0\rangle \otimes |\psi\rangle + b|1\rangle \otimes U|\psi\rangle \\
&= a|0, \psi\rangle + b|1, U\psi\rangle.
\end{aligned}$$

The result is a superposition of entangled outputs. Notice that CNOT is a special case of this construction, a controlled X.

We also have a quantum analogue for an if-then-else construction. If U_0 and U_1 are unitary operators, then we can make the choice between them conditional on a control bit as follows:

$$|0\rangle\langle 0| \otimes U_0 + |1\rangle\langle 1| \otimes U_1.$$

For example,

$$\text{CNOT} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X. \tag{III.17}$$

In quantum circuit diagrams, the symbol for the CCNOT gate is show in Fig. III.10, or with $\bullet$ for top two connections and $\oplus$ for bottom, suggesting $\text{CCNOT}|x, y, z\rangle = |x, y, xy \oplus z\rangle$. Alternately, put "CCNot" in a box. Other operations may be shown by putting a letter or symbol in a box, for example "H" for the Hadamard gate.

The Hadamard gate can be used to generate Bell states (Exer. III.31):

$$\text{CNOT}(H \otimes I)|xy\rangle = |\beta_{xy}\rangle. \tag{III.18}$$

In	Out
$\|00\rangle$	$(\|00\rangle + \|11\rangle)/\sqrt{2} \equiv \|\beta_{00}\rangle$
$\|01\rangle$	$(\|01\rangle + \|10\rangle)/\sqrt{2} \equiv \|\beta_{01}\rangle$
$\|10\rangle$	$(\|00\rangle - \|11\rangle)/\sqrt{2} \equiv \|\beta_{10}\rangle$
$\|11\rangle$	$(\|01\rangle - \|10\rangle)/\sqrt{2} \equiv \|\beta_{11}\rangle$

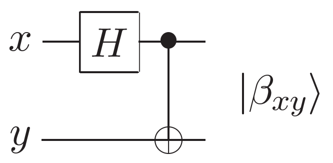

Figure III.13: Quantum circuit for generating Bell states. [from Nielsen & Chuang (2010, fig. 1.12)]

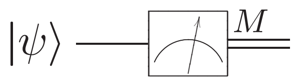

Figure III.14: Symbol for measurement of a quantum state (from Nielsen & Chuang (2010)).

The circuit for generating Bell states (Eq. III.18) is shown in Fig. III.13.

It's also convenient to have a symbol for quantum state measurement, such as Fig. III.14.

C.4 Quantum gate arrays

Fig. III.15 shows a quantum circuit for a 1-bit full adder. As we will see (Sec. C.7), it is possible to construct reversible quantum gates for any classically computable function. In particular the Fredkin and Toffoli gates are universal.

Because quantum computation is a unitary operator, it must be reversible. You know that an irreversible computation $x \mapsto f(x)$ can be embedded in a reversible computation $(x, c) \mapsto (g(x), f(x))$, where c are suitable ancillary constants and $g(x)$ represents the garbage qubits. Note that throwing away the garbage qubits (dumping them in the environment) will collapse the quantum state (equivalent to measurement) by entangling them in the many degrees of freedom of the environment. Typically these garbage qubits will be entangled with other qubits in the computation, collapsing them as well, and interfering with the computation. Therefore the garbage must be

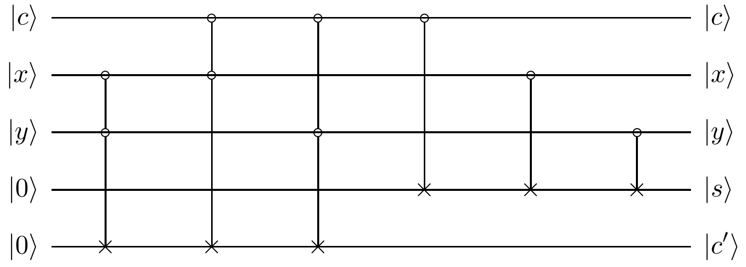

Figure III.15: Quantum circuit for 1-bit full adder [from Rieffel & Polak (2000)]. "x and y are the data bits, s is their sum (modulo 2), c is the incoming carry bit, and c' is the new carry bit."

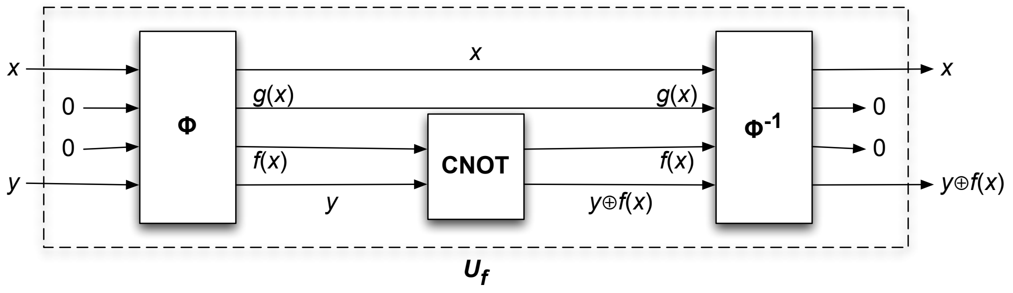

Figure III.16: Quantum gate array for reversible quantum computation.

produced in a *standard state* independent of x. This is accomplished by uncomputing, as we did in classical reversible computing (Ch. II, Sec. C.6, p. 56).

Since NOT is reversible, each 1 bit in c can be replaced by a 0 bit followed by a NOT, so we need only consider $(x, 0) \mapsto (g(x), f(x))$; that is, all the constant bits can be zero.

Therefore, we begin by embedding our irreversible computation of f in a reversible computation Φ, which we get by providing 0 constants and generating garbage $g(x)$; see Fig. III.16. That is, Φ will perform the following computation on four registers (*data, workspace, result, target*):

$$(x, 0, 0, y) \mapsto (x, g(x), f(x), y).$$

The result $f(x)$ is in the result register and the garbage $g(x)$ is in the workspace register. Notice that x and y (data and target) are passed through. Now use CNOTs between corresponding places in the result and target registers to compute $y \oplus f(x)$, where $\oplus$ represents bitwise exclusive-or, in the target register. Thus we have computed:

$$(x, 0, 0, y) \mapsto (x, g(x), f(x), y \oplus f(x)).$$

Now we uncompute with Φ^{-1}, but since the data and target registers are passed through, we get $(x, 0, 0, y \oplus f(x))$ in the registers. We have restored the data, workspace, and result registers to their initial values and have $y \oplus f(x)$ in the target register. Ignoring the result and workspace registers, we write

$$(x, y) \mapsto (x, y \oplus f(x)).$$

This is the standard approach we will use for embedding a classical computation in a quantum computation.

Therefore, for any computable $f : \mathbf{2}^m \to \mathbf{2}^n$, there is a reversible *quantum gate array* $U_f : \mathcal{H}^{m+n} \to \mathcal{H}^{m+n}$ such that for $x \in \mathbf{2}^m$ and $y \in \mathbf{2}^n$,

$$U_f|x, y\rangle = |x, y \oplus f(x)\rangle,$$

See Fig. III.17. In particular, $U_f|x, \mathbf{0}\rangle = |x, f(x)\rangle$. The first m qubits are called the *data register* and the last n are called the *target register.*

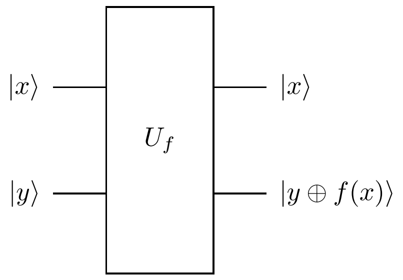

Figure III.17: Computation of function by quantum gate array (Rieffel & Polak, 2000).

C.5 Quantum parallelism

Since U_f is linear, if it is applied to a superposition of bit strings, it will produce a superposition of the results of applying f to them in parallel (i.e., in the same time it takes to compute it on one input):

$$U_f(c_1|\mathbf{x}_1\rangle + c_2|\mathbf{x}_2\rangle + \cdots + c_k|\mathbf{x}_k\rangle) = c_1 U_f|\mathbf{x}_1\rangle + c_2 U_f|\mathbf{x}_2\rangle + \cdots + c_k U_f|\mathbf{x}_k\rangle.$$

For example, if we have a superposition of the inputs $\mathbf{x}_1$ and $\mathbf{x}_2$,

$$U_f\left(\frac{\sqrt{3}}{2}|\mathbf{x}_1\rangle + \frac{1}{2}|\mathbf{x}_2\rangle\right) \otimes |\mathbf{0}\rangle = \frac{\sqrt{3}}{2}|\mathbf{x}_1, f(\mathbf{x}_1)\rangle + \frac{1}{2}|\mathbf{x}_2, f(\mathbf{x}_2)\rangle.$$

The amplitude of a result y will be the sum of the amplitudes of all x such that $y = f(x)$.

If we apply U_f to a superposition of all possible 2^m inputs, it will compute a superposition of all the corresponding outputs *in parallel* (i.e., in the same time as required for one function evaluation)! The Walsh-Hadamard transformation can be used to produce this superposition of all possible inputs:

$$\begin{aligned} W_m|00\ldots 0\rangle &= \frac{1}{\sqrt{2^m}}\left(|00\ldots 0\rangle + |00\ldots 1\rangle + \cdots + |11\ldots 1\rangle\right) \\ &= \frac{1}{\sqrt{2^m}} \sum_{\mathbf{x}\in\mathbf{2}^m} |\mathbf{x}\rangle \\ &= \frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x\rangle. \end{aligned}$$

In the last line we are obviously interpreting the bit strings as natural numbers. Hence,

$$U_f W_m|\mathbf{0}\rangle = U_f \left(\frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x, 0\rangle \right) = \frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} U_f|x, 0\rangle = \frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x, f(x)\rangle.$$

A single circuit does all 2^m computations simultaneously! "Note that since n qubits enable working simultaneously with 2^n states, quantum parallelism circumvents the time/space trade-off of classical parallelism through its ability to provide an exponential amount of computational space in a linear amount of physical space." (Rieffel & Polak, 2000)

This is amazing, but not immediately useful. If we measure the input bits, we will get a random value, and the state will be projected into a superposition of the outputs for the inputs we measured. If we measure an output bit, we will get a value probabilistically, and a superposition of all the inputs that can produce the measured output. Neither of the above is especially useful, so most quantum algorithms transform the state in such a way that the values of interest have a high probability of being measured. The other thing we can do is to extract common properties of all values of $f(x)$. Both of these require different programming techniques than classical computing.