Quantum Programming HW2

B490 : Spring 2020

Joshua Larkin

1 Response

1.4 In order to implement fanout, we must have a bit we want to copy, say x, and two other bits. The second of the other bits will be a utility bit and the third will be the copy of x. We will set the other two bits to be 1 and 0, respectively. Then we do a `ccx` on the three bits (x, 1, 0). This will result in the third bit being 1 if $x = 1$ and 0 if $x = 0$, and we have not altered x. Since we took the second bit to be 1, and it would technically start at 0, we will apply the `x` gate to it afterwards, to set it back to its starting value.

2.3 Fix a positive number n. We show how to construct `toffoli`(n).

Note that this means we are given n bits, so consider them as an array called B.

We will need $n - 3$ ancilla bits, so consider them as an array called Z.

The high-level idea is to use `ccx` on two bits of B and one bit of Z until there are only two bits of B that have not been used. Then we will use `ccx` on three bits of Z at a time, until there is only one bit of Z that has not been used. It is important that everytime we write one of these uses of `ccx`, we make a copy of the instruction we are writing, and we add remaining instructions in the space between the fresh instruction and its copy. This maintains all ancilla bits being reset. The last step is to use `ccx` on the $n-1$ bit of B, the $n-3$ bit of Z, and the n bit of B. We provide a pseudocode algorithm on the next page

```
def toffoli(B):
''' consider instructions a double linked list with a method add(x)
    where add(x) inserts x into the current pointer of the list,
    inserts x again, then moves the pointer  back one element
    This maintains the symmetry needed to reset ancilla bits

    The put(x) method does standard insertion (no copies)
'''
# we write instructions here & return this variable
#  as the constructed gate
   instructions = []
# for n bits, the n-th bit is the (n-1) index of B
    n = len(B)
# ancilla bit array
    Z = [0 for x in range(n - 2)]
# used to keep track of how many ancilla bits we have used
    aniclla_used = 0
    for i = 0; i < (n-2); i += 2:
       # we are ignoring the necessary string formatting
       # that would occur here
          instructions.add("ccx B[i], B[i+1], Z[ancilla_used]")
          ancilla_used++

    for i = 0; i < (n-4); i++:
        instructions.add("ccx Z[i], Z[i+1], Z[ancilla_used]")
        aniclla_used++

    instructions.put("ccx B[n-2], Z[n-4], B[n-1]")
    return instructions
```

3.6 Fix a number n and two n-bit strings s_1 and s_2. We show how to construct a reversible circuit that swap s_1 and s_2 while leaving all other inputs fixed. Since there are 2^n possible inputs, we can enumerate them from 0 to $2^n - 1$. Let $s_1 = j$ and $s_2 = k$, where $0 \leq j < k \leq 2^n - 1$. We may assume without loss of generality that $j < k$ In order to swap inputs and leavce others fixed, we will use transpositions of length two and conjugacy to swap elements. This means that we want a function that behaves like, in cycle notation, the transformation $(j\ k)$. We will use $(k-1\ k)$ and $(j\ k-1)$ in order to perform our swap. We shall construct our circuit as such:

$$(j\ k-1)(k-1\ k)(j\ k-1)$$

And it follows, by multiplication, that

$$(j\ k-1)(k-1\ k)(j\ k-1) = (j\ k)$$