STUDIA INFORMATICA 2013
Volume 34 Number 3 (113)

Joanna WIŚNIEWSKA
Wojskowa Akademia Techniczna, Instytut Systemów Informatycznych

IMPLEMENTACJA FUNKCJI LOGICZNYCH ZA POMOCĄ SIECI BRAMEK KWANTOWYCH

Streszczenie. W poniższym artykule podniesiony został problem modelowania funkcji logicznych za pomocą kwantowych obwodów unitarnych bez korzystania z tzw. kwantowych bitów pomocniczych. Podano algorytm opisujący tworzenie postaci macierzowej operatora kwantowego, za pomocą którego można wyznaczyć wartość funkcji logicznych *n* zmiennych, a także wskazano, w jaki sposób skonstruować sieć znanych bramek kwantowych realizującą działanie wspomnianego operatora.

Słowa kluczowe: funkcje logiczne, bramki kwantowe, macierze unitarne

THE BOOLEAN FUNCTIONS IMPLEMENTATION WITH USE OF QUANTUM GATES CIRCUIT

Summary. The following publication concerns on Boolean functions modelling with use of quantum gates, but without so-called ancilla qubits. First, the algorithm of matrix form calculating for quantum operator is presented – mentioned operator may be used to determine the value of Boolean function. Next, it is shown how to construct quantum gates circuit solving given problem.

Keywords: logic functions, quantum gates, unitary matrices

1. Wstęp

Dowolną funkcję logiczną może realizować układ klasycznych bramek *NAND*. Ze względu na to, że tzw. bramka Toffoliego może wykonywać operację *NAND*, w latach 90. XX wieku pojawiły się publikacje, m.in. [2], proponujące budowę układów kwantowych realizujących funkcje logiczne z użyciem tejże bramki. Rozwiązanie to ma wadę, mianowicie każda bramka Toffoliego ma tzw. qubit (kwantowy bit) sterujący, który decyduje o tym, czy bramka wykonuje operację *AND* czy *NAND* na pozostałych qubitach przetwarzanych przez bramkę.

Biorąc po uwagę problem dekoherencji należałoby zmniejszać liczbę kwantowych qubitów potrzebnych do rozwiązania zadania, a każda bramka *NAND* w układzie generuje potrzebę korzystania z dodatkowego qubitu sterującego. Poniżej przedstawiony został sposób konstruowania sieci znanych bramek kwantowych, realizującej funkcję logiczną, bez korzystania z qubitów sterujących. Eliminacja qubitów sterujących została otrzymana poprzez sprowadzenie problemu modelowania funkcji logicznej do problemu przyporządkowania danej funkcji pewnej macierzy unitarnej, a następnie do problemu syntezy kwantowego obwodu unitarnego, realizującego działanie operatora wyrażonego za pomocą wspomnianej macierzy. Prezentowane rozwiązanie różni się od aktualnie stosowanych, np. [1, 8, 10], brakiem elementów probabilistycznych, a celem rozwijanej metody jest opracowanie deterministycznego algorytmu wskazującego, w jaki sposób można konstruować układy realizujące funkcje logiczne o jak najmniejszej liczbie bramek.

2. Rozpatrywane funkcje logiczne

Funkcją logiczną *f*, *n* zmiennych, będzie nazwane odwzorowanie:

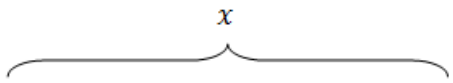
$$f : B^n \to B, \tag{1}$$

gdzie $B = \{0,1\}$. Dodatkowo, zakłada się, że funkcja *f* nie jest funkcją stałą i jej wartość zależy od wartości wszystkich *n* zmiennych rozpatrywanego zadania.

Dla każdej funkcji *f* o *n* zmiennych, które będą oznaczane jako x_i ($i = \overline{1,n}$), można podać tabelę prawdy. Niech tabelą prawdy dla funkcji *f* będzie tabela o (*n* + 2) kolumnach oraz (2^n + 1) wierszach – łącznie z tzw. nagłówkiem tabeli. Kolumny, które w nagłówku mają oznaczenia x_i ($i = \overline{1,n}$) zawierają wszystkie możliwe kombinacje wartości zero-jedynkowych, które mogą przyjmować zmienne x_i funkcji *f*, w taki sposób, iż wartości te czytane wierszami są kolejnymi liczbami binarnymi, dla których najbardziej istotny bit znajduje się zawsze w kolumnie x_1. Wartość liczby binarnej zapisanej w dowolnym wierszu jest zawsze o jeden mniejsza niż wartość *lp* w tymże wierszu – liczbę porządkową $lp = \overline{1,2^n}$ $lp = \overline{1, 2^n}$zawiera pierwsza kolumna tabeli prawdy, o oznaczeniu *lp* w nagłówku. Oznacza to, że *lp* jest numerem konkretnego przyporządkowania (pary) wartości wektorów:

$$\left(x^{lp}, F\left(x^{lp}\right)\right), \tag{2}$$

gdzie: zapis x^{lp} oznacza wektor *x*, *n* składowych, znajdujący się w wierszu tabeli prawdy o numerze *lp*, a $F\left(x^{lp}\right)$ to wartość decyzji dla danego wektora x^{lp} $\left(F\left(x^{lp}\right) \in \{0,1\}\right)$. Ostatnia kolumna tabeli prawdy, z nagłówkiem *f*(*x*), zawiera zbiór możliwych wartości decyzji *F*(*x*), odpowiadający permutacjom wszystkich wartości zmiennych x_i.



Lp.	x_1	x_2	...	x_{n-1}	x_n	$f(x)$
1	0	0	...	0	0	$F(x^1)$
2	0	0	...	0	1	$F(x^2)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
2^n	1	1	...	1	1	$F(x^{2^n})$

Rys. 1. Wzorzec tabeli prawdy
Fig. 1. The pattern of truth table

Warto zauważyć, że dla zadania *n* zmiennych w kolumnie *f*(*x*) można podać 2^{2^n} różnych ciągów zero-jedynkowych. Liczba funkcji *f*, dokładnie *n* zmiennych, zostanie podana, jeżeli od liczby ciągów 2^{2^n} zostaną odjęte ciągi reprezentujące funkcje stałe ($f(x)=0$ i $f(x)=1$) oraz ciągi odpowiadające funkcjom, które można uprościć do mniejszej liczby zmiennych niż *n*. Niech wartość mówiąca o liczbie funkcji *f* dokładnie *n* zmiennych zostanie oznaczona jako lf_n. W przypadku funkcji jednej zmiennej, po wykluczeniu funkcji stałych, lf_1 wynosi:

$$lf_1 = 2^{2^1} - 2 = 2, \tag{3}$$

co jest zgodne z rzeczywistością, bo dla jednej zmiennej możemy podać tylko dwie różne funkcje logiczne: $f(x)=x_1$ i $f(x)=\bar{x}_1$.

Obliczając lf_n dla $n > 1$ należy uwzględniać liczbę funkcji o mniejszej liczbie zmiennych, których liczność można wyrazić jako kombinację *k* zmiennych ($k=\overline{1,n-1}$) spośród *n*-elementowego zbioru zmiennych:

$$lf_n = 2^{2^n} - 2 - \left(\sum_{k=1}^{n-1} \frac{n!}{k!(n-k)!} \cdot lf_k \right) \quad dla \quad n>1 \tag{4}$$

Ze względu na to, że wzór (4) ma charakter rekurencyjny, aby wyliczyć dowolne lf_n potrzebne jest wcześniejsze wyliczenie wszystkich lf_k, przy założeniu, że lf_1 jest znane (3).

3. Operator kwantowy wyznaczający wartość funkcji logicznej

Przyjęte zostało założenie mówiące, że zero-jedynkowe wartości przyjmowane przez zmienne x_i rozpatrywanej funkcji *f* zostaną przypisane kolejnym *n* qubitom rejestru kwantowego $|h\rangle$:

$$|h\rangle = |x_1, x_2, ..., x_n\rangle \tag{5}$$

Korzystając z tabeli prawdy funkcji *f* można podać macierz *U*, będącą reprezentacją operatora kwantowego, który realizuje przekształcenie

$$|h\rangle \xrightarrow{U} |h'\rangle \tag{6}$$

takie, że ze stanu końcowego $|h'\rangle$ rejestru kwantowego można odczytać wartość funkcji *f*(*x*), korzystając z algorytmu opisanego w tym podrozdziale. Stan $|h'\rangle$ można opisać analogicznie do (5):

$$|h'\rangle = |x_1', x_2', ..., x_n'\rangle \tag{7}$$

Wspomniany algorytm można podzielić na dwa etapy – w pierwszym z nich, na podstawie tabeli prawdy, tworzona jest tablica pomocnicza i zapamiętywany jest klucz, służący odczytaniu wyników zadania ze stanu $|h'\rangle$, następnie w drugim etapie wyliczana jest postać macierzy *U*.

Tablica pomocnicza ma 2^n wierszy oraz dwie kolumny zwierające odpowiednio: stany początkowe $|h\rangle$ i stany końcowe $|h'\rangle$ rejestru obliczeniowego dla zadania *n* zmiennych. Wartości stanów początkowych rejestru są znane (są to kolejne całkowite liczby binarne od 0 do $(2^n - 1)$, czyli liczby zapisane w tablicy prawdy, w kolumnach należących do wektora *x*). Zmiennej pomocniczej *lp* należy przypisać wartość 1, a zmienna *stan_końcowy* jest inicjowana wartością 0, zapisaną w postaci *n*-cyfrowej liczby binarnej. Zmienna *z* będzie przechowywała wartość potrzebną do odczytania wyników zadania. W celu wyznaczenia wartości znajdujących się w drugiej kolumnie tablicy pomocniczej wykonywane są następujące kroki:

1. Jeżeli w tabeli prawdy w wierszu o numerze *lp* kolumna z nagłówkiem *f*(*x*) zawiera wartość 0, to przejście do kroku 2; w przeciwnym wypadku zwiększana jest wartość zmiennej *lp* o jeden. Jeżeli ($lp > 2^n lp > 2^n$), to przejście do kroku 3; w przeciwnym wypadku powtarzany jest krok 1.
2. Stanowi początkowemu, który jest reprezentowany przez liczbę binarną, co do wartości o jeden mniejszą niż aktualna wartość zmiennej lp^{lp}, przypisywany jest stan końcowy o aktualnej wartości zmiennej *stan_końcowy*, a następnie inkrementowane są wartości zmiennych: *lp* i *stan_końcowy*. Jeżeli ($lp > 2^n lp > 2^n$), to przejście do kroku 3; w przeciwnym wypadku powtarzany jest krok 1.
3. Zmiennej *z* przypisywana jest wartość (*stan_końcowy* + 1). Zmiennej *lp* ^{lp}przypisywana jest wartość 1 i następuje przejście do kroku 4.
4. Jeżeli w tabeli prawdy w wierszu o numerze *lp* kolumna z nagłówkiem *f*(*x*) zawiera wartość 1, to przejście do kroku 5; w przeciwnym wypadku zwiększana jest wartość zmien-

nej *lp* o jeden. Jeżeli (*lp* > $2^n$$lp > 2^n$), to koniec algorytmu; w przeciwnym wypadku powtarzany jest krok 4.

5. Stanowi początkowemu, który jest reprezentowany przez liczbę binarną, co do wartości o jeden mniejszą niż aktualna wartość zmiennej $lp$$lp$, przypisywany jest stan końcowy o aktualnej wartości zmiennej *stan_końcowy*, a następnie inkrementowane są wartości zmiennych: *lp* i *stan_końcowy*. Jeżeli (*lp* > $2^n$$lp > 2^n$), to koniec algorytmu; w przeciwnym wypadku powtarzany jest krok 4. ■

Korzystając z tablicy pomocniczej, zawierającej 2^n par $|h\rangle - |h'\rangle$, wyznacza się postać macierzy U o rozmiarze $2^n \times 2^n$. W tym celu stany $|h\rangle$ i $|h'\rangle$ mogą zostać przedstawione jako kolumnowe wektory amplitud, których wartości są znane – odpowiednio H i H'.

$$H = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix}, \quad H' = \begin{bmatrix} \alpha_0' \\ \alpha_1' \\ \vdots \\ \alpha_{2^n-1}' \end{bmatrix} \tag{8}$$

Przekształcenie z (6) może zostać zapisane:

$$U \cdot H = H', \tag{9}$$

czyli wyliczenie 2^{2n} elementów U wymaga rozwiązania układu 2^{2n} równań, utworzonych na podstawie zależności (10).

$$\begin{bmatrix} u_{0,0} & u_{0,1} & \dots & u_{0,2^n-1} \\ u_{1,0} & u_{1,1} & \dots & u_{1,2^n-1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{2^n-1,0} & u_{2^n-1,1} & \dots & u_{2^n-1,2^n-1} \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \begin{bmatrix} \alpha_0' \\ \alpha_1' \\ \vdots \\ \alpha_{2^n-1}' \end{bmatrix} \tag{10}$$

Wartość *z*, obliczana dla danej funkcji *f* w trzecim kroku algorytmu tworzenia tablicy pomocniczej, pozwala na odczytanie wartości funkcji *f* ze stanu $|h'\rangle$ w następujący sposób:

$$f(x) = \begin{cases} 0 & dla \quad z > 2^{n-1} \cdot x_1' + 2^{n-2} \cdot x_2' + \ldots + 2^1 \cdot x_{n-1}' + 2^0 \cdot x_n' \\ 1 & dla \quad z \le 2^{n-1} \cdot x_1' + 2^{n-2} \cdot x_2' + \ldots + 2^1 \cdot x_{n-1}' + 2^0 \cdot x_n' \end{cases} \tag{11}$$

4. Charakterystyka macierzy *U*

Zanim zostaną opisane sposoby realizacji działania operatora kwantowego, przedstawianego za pomocą macierzy *U*, warto zwrócić uwagę na pewne charakterystyczne cechy tej macierzy.

Macierz *U*, o wymiarach $2^n \times 2^n$ dla funkcji *n* zmiennych, jest binarną macierzą unitarną, ponieważ jej elementy są wyznaczane na podstawie 2^n różnych postaci wektora kolumnowego *H* oraz 2^n 2^nróżnych postaci wektora kolumnowego *H'X*, przy czym *H* i *H'* opisują wyłącznie stany kwantowe z tzw. bazy standardowej (elementy każdego wektora, czyli amplitudy stanu kwantowego to (2^n – 1) zer i jedna jedynka). Macierz $U_{2^n \times 2^n}$ zawsze będzie pewną permutacją kolumn/wierszy macierzy identyczności $I_{2^n \times 2^n}$ $I_{2^n \times 2^n}$, a co za tym idzie kolumny i wiersze *U* będą wzajemnie ortogonalne oraz suma podniesionych do kwadratu wartości elementów każdej kolumny i każdego wiersza będzie równa jedności, a z tego wynika, że *U* Ujest unitarna – dowód w [5].

Skoro $U_{2^n \times 2^n}$ jest pewną permutacją kolumn macierzy $I_{2^n \times 2^n}$, to można ją zapisać w formie $2^n 2^n$-elementowego ciągu (*c*) liczb całkowitych dodatnich, w którym każda *j*-ta liczba ($j = \overline{1,2^n}$ $i = \overline{1, 2^n}$) odpowiada *j*-tej kolumnie *U*, a wartość *j*-tej liczby to numer wiersza, w którym
w macierzy *U* Uznajduje się wartość 1, np.:

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{12}$$

wtedy 4-elementowy ciąg (*c*) ma postać: 1, 3, 2, 4.

Sposób tworzenia *U*, opisany w poprzednim podrozdziale, powoduje, że *U* może być macierzą jednostkową *I*, jeżeli w kolumnie *f*(*x*) tablicy prawdy znajduje się posortowany niemalejąco ciąg zero-jedynkowy (czytając tablicę od wiersza *lp* = 1 do *lp* = 2^n). W pozostałych przypadkach ciąg (*c*) dla macierzy *U* można podzielić na dwa rosnące (różnica ciągu *r* = 1) ciągi liczb całkowitych dodatnich, których elementy mogą się przeplatać – jeden z tych ciągów jest skorelowany z wystąpieniami wartości „0" w kolumnie *f*(*x*) tabeli prawdy, a drugi z wystąpieniami w tej kolumnie wartości „1".

Można zauważyć, że liczba macierzy $U_{2^n \times 2^n}$, potrzebnych do realizacji obliczeń dla wszystkich funkcji *f*, dokładnie *n* zmiennych, jest trochę mniejsza niż liczba funkcji *f* podana w (4), z tego powodu, że w grupie tychże funkcji mogą występować różne od siebie funkcje

dokładnie *n* zmiennych, dla których ciąg (*c*) będzie ciągiem rosnącym, a wtedy $U_{2^n\times 2^n}=I_{2^n\times 2^n}$. Wspomniana różnica między liczbą macierzy *U* a liczbą funkcji *f* dla konkretnej liczby zmiennych jest tak nieznaczna, iż można przyjąć, że obydwie te wartości są tego samego rzędu: 2^{2^n}, co wynika z (4). Istotne jest to, że przedstawiony sposób modelowania funkcji logicznej wymaga skonstruowania macierzy $U_{2^n\times 2^n}$, które są tylko pewnym podzbiorem wszystkich permutacji kolumn macierzy jednostkowej $I_{2^n\times 2^n}$, których liczba wynosi $(2^n)!$, a można pokazać, że $2^{2^n} << (2^n)!$ dla $n \to \infty$. W dodatku, jeżeli $U \neq I$, wiadomo, że *U* ma tę cechę, iż zrzutowana na ciąg (*c*) składa się z dwóch rosnących ciągów, których wyrazy mogą się przeplatać, co wpływa na konstrukcję algorytmów budowania sieci bramek kwantowych, realizujących operację *U* oraz pomaga zwiększyć dokładność oszacowania złożoności tychże układów.

5. Implementacja *U* za pomocą sieci bramek kwantowych

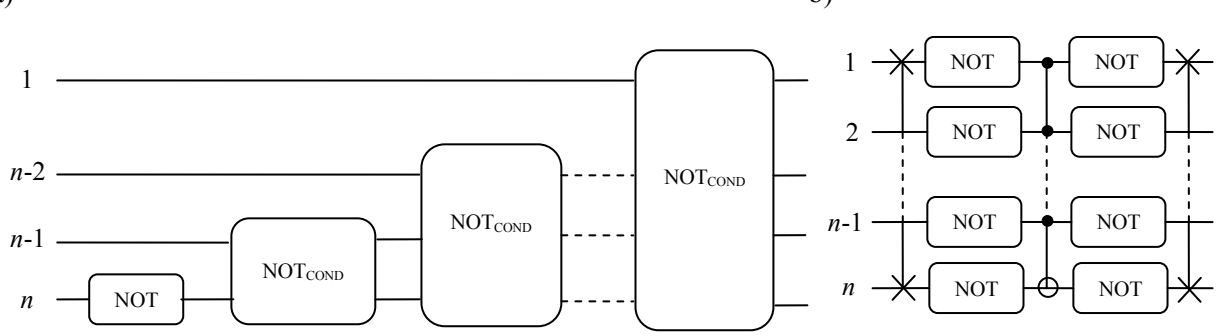
Macierz *U*, będąca reprezentacją pewnego operatora kwantowego, jest binarną macierzą unitarną, więc może zostać zaimplementowana za pomocą układu kwantowego, zbudowanego z bramek Toffoliego oraz operacji *Roll*, której reprezentacja macierzowa, będąca pewną permutacją macierzy *I*, została podana w (13). W pracy [11] pokazano dowód, że wspomniane wyżej operacje stanowią zbiór operacji bazowych dla binarnych macierzy unitarnych.

$$Roll = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix} \tag{13}$$

Na rys. 2a przedstawiono układ realizujący *n*-qubitową operację *Roll*, przy czym operacja NOT_{COND} (rys. 2b) może zostać przedstawiona jako taka permutacja kolumn macierzy *I*, w której pierwsza kolumna *I* została zamieniona miejscami z kolumną o numerze $2^{n-1}+1$. Poniżej znajduje się przykład 2-qubitowej operacji NOT_{COND} (14).

$$NOT_{COND} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

Dla przyspieszenia wykonywanych operacji zostanie użyta także operacja $Roll^{-1}$, która może zostać zaimplementowana jako układ odwrotny do układu *Roll*.



Rys. 2. (a) układ *n*-qubitowy dla operacji *Roll*, (b) układ *n*-qubitowy dla operacji NOT_{COND}
Fig. 2. (a) *n*-qubit circuit for *Roll* operation, (b) *n*-qubit circuit for NOT_{COND} operation

Jeżeli dla pewnej funkcji logicznej *n* zmiennych macierz $U_{2^n\times 2^n}=I_{2^n\times 2^n}$, to układ bramek może składać się z jednej bramki *I* o *n* wejściach/wyjściach. W pozostałych przypadkach układ będzie konstruowany z operacji *Roll*, $Roll^{-1}$ i bramki Toffoliego, korzystając z następujących obserwacji: *Roll* przesuwa kolumny konstruowanej macierzy w lewo, a $Roll^{-1}$ w prawo; bramka Toffoliego zamienia miejscami dwie ostatnie kolumny; jako punkt startowy przyjmowana jest macierz *I*, więc skoro *U* może zostać przedstawione jako ciąg (*c*) składający się z dwóch ciągów rosnących, to wystarczy odpowiednio uporządkować tylko jeden z nich; znana jest wartość *z* (klucz do odczytania wyników zadania), mówiąca o długości ciągu rosnącego korespondującego z wartościami funkcji równymi „0” (odczytywane z tablicy prawdy).

Niech celem będzie skonstruowanie macierzy $U_{2^n\times 2^n}\neq I_{2^n\times 2^n}$ za pomocą iloczynów postaci macierzowych operacji *Roll*, $Roll^{-1}$ i bramki Toffoliego o wymiarach $2^n \times 2^n$. Macierz pomocnicza $X_{2^n\times 2^n}$ zostaje zainicjowana jako macierz jednostkowa $I_{2^n\times 2^n}$. Zmiennej pomocniczej *pozycja* przypisywana jest wartość 2^n. W celu skonstruowania układu realizującego *U* należy wykonać następujące kroki:

1. Należy sprawdzić czy $z \le 2^{n-1}$ – jeżeli tak, to przejście do kroku 2; w przeciwnym wypadku przejście do kroku 5.
2. Zmiennej pomocniczej *k* przypisywana jest wartość *z*.
3. Jeżeli $k = 0$, to przejście do kroku 8; w przeciwnym wypadku: pod zmienną j_1 przypisywany jest numer kolumny, w której w *U* występuje „1” w *k*-tym wierszu. Pod zmienną j_2 przypisywany jest numer kolumny, w której w *X* występuje „1” w *k*-tym wierszu. Jeżeli $j_1 = j_2$, to *k* zostaje zmniejszone o 1 i następuje powtórzenie kroku 3; w przeciwnym wypadku przejście do kroku 4.

4. Na macierzy X należy wykonać operację $Roll^{-1}$ ($pozycja - 1 - j_2$) razy. Zastosować operację Toffoli jeden raz. Następnie $Roll$ i Toffoli naprzemiennie ($j_1 - j_2 - 1$) razy. Zmiennej $pozycja$ przypisać wartość j_1. Zmniejszyć k o jeden i przejść do kroku 3.
5. Zmiennej pomocniczej k przypisywana jest wartość ($z + 1$).
6. Jeżeli $k = 2^n + 1$, to przejście do kroku 8; w przeciwnym wypadku: pod zmienną j_1 przypisywany jest numer kolumny, w której w U występuje „1” w k-tym wierszu. Pod zmienną j_2 przypisywany jest numer kolumny, w której w X występuje „1” w k-tym wierszu. Jeżeli $j_1 = j_2$, to k zostaje zwiększone o 1 i następuje powtórzenie kroku 6; w przeciwnym wypadku przejście do kroku 7.
7. Jeżeli $j_2 \leq pozycja$, to na macierzy X należy wykonać operację $Roll^{-1}$ ($pozycja - j_2$) razy; w przeciwnym wypadku należy wykonać na macierzy X operację $Roll$ ($pozycja - j_2$) razy. Zastosować operację Toffoli jeden raz. Następnie $Roll^{-1}$ i Toffoli naprzemiennie ($j_2 - j_1 - 1$) razy. Zmiennej $pozycja$ przypisać wartość ($j_1 + 1$). Zwiększyć k o jeden i przejść do kroku 6.
8. Na macierzy X wykonać operację $Roll$ ($2^n - pozycja$) razy.
9. Sieć bramek realizującą operację U należy zbudować wstawiając podukłady bramek kwantowych odpowiadające użytym operacjom $Roll$, $Roll^{-1}$ i Toffoli zaczynając od wyjścia, a kończąc na wejściu całego układu.

6. Podsumowanie

Zaletą rozwiązania przedstawionego w tym rozdziale jest możliwość modelowania obwodu bramek kwantowych, realizującego operację logiczną, bez użycia qubitów sterujących (budując układ z bramek Toffoliego, pracujących jako uniwersalne bramki *NAND* należy uwzględniać zwiększenie rejestru kwantowego o liczbę qubitów równą liczbie użytych bramek). W podrozdziale 5 została przedstawiona propozycja implementacji operacji U, realizującej działanie funkcji logicznej, za pomocą obwodu znanych bramek kwantowych – ze względu na konstrukcję wykorzystanego w tym celu algorytmu można stwierdzić, że najbardziej złożone układy powstawałyby dla przypadków, w których modelowane U miałoby postać ciągu (c):

$$2^{n-1}+1, 2^{n-1}+2, \ldots, 2^n, 1, 2, \ldots, 2^{n-1} \tag{15}$$

lub

$$2^{n-1}+1, 1, 2^{n-1}+2, 2, \ldots, 2^n, 2^{n-1} \tag{16}$$

Wtedy liczba warstw bramek – uwzględniając to, że każda operacja $Roll$ bądź $Roll^{-1}$ to ($5n - 5$) warstw dla $n > 1$, a bramka Toffoliego ze względu na możliwość efektywnej implementacji [6] traktowana jest jako jedna warstwa – byłaby rzędu 2^n. Przypadki te jednak nie należą do zbioru funkcji f dokładnie n zmiennych, bo upraszczają się do funkcji jednej

zmiennej: (15) to $f(x) = \bar{x}_1$, a (16) można przedstawić jako $f(x) = \bar{x}_n$. Funkcje te można realizować za pomocą 1-qubitowej bramki *NOT*. Potrzebna jest dalsza ewaluacja użytych metod w kierunku badania złożoności powstających układów oraz zoptymalizowania zaproponowanego w podrozdziale 5 algorytmu, co również przełoży się na zmniejszenie liczby warstw bramek w sieci.

BIBLIOGRAFIA

1. Ambainis A., de Wolf R.: How Low Can Approximate Degree and Quantum Query Complexity be for Total Boolean Functions? arXiv:01206.0717 v2 [quant-ph], 2013.
2. Barenco A., Ekert A., Vedral V.: Quantum Networks for Elementary Arithmetic Operations. arXiv:quant-ph/9511018, 1995.
3. Bugajski S., Klamka J., Węgrzyn S.: Foundation of quantum computing. Part1. Archiwum Informatyki Teoretycznej i Stosowanej, vol. 1, no. 2, 2001, p. 97-114.
4. Bugajski S., Klamka J., Węgrzyn S.: Foundation of quantum computing. Part2. Archiwum Informatyki Teoretycznej i Stosowanej, vol. 14, no. 2, 2002, p. 93-106.
5. Chudy M.: Wprowadzenie do Informatyki Kwantowej. Akademicka Oficyna Wydawnicza EXIT, Warszawa 2011.
6. Gilchrist A., Ralph T.C., Resch K.J.: Efficient Toffoli Gates Using Qudits. arXiv: 0806.0654v1 [quant-ph], 2008.
7. Hardy Y., Steeb W.H.: A Sequence of Quantum Gates. arXiv:1202.2259v1 [quant-ph], 2012.
8. Kameyama M., Lukac M., Perkowski M.: Evolutionary Quantum Logic Synthesis of Boolean Reversible Logic Circuits Embedded in Ternary Quantum Space Using Heuristics. arXiv:1107.3383v1 [quant-ph], 2011.
9. Kaye P., Laflamme R., Mosca M.: An Introduction to Quantum Computing. Oxford University Press, Oxford 2007.
10. Montanaro A., Osborne T.J.: Quantum Boolean Functions. arXiv:0810.2435v5 [quant-ph], 2010.
11. Wiśniewska J.: Quantum Computer Network Model for a Decision Making Algorithm. Communications in Computer and Information Science, Springer-Verlag, vol. 291, p. 73-81, Berlin 2012.

Wpłynęło do Redakcji 13 marca 2013 r.

Abstract

The chapter concerns on Boolean functions modelling with use of quantum gates, but without so-called ancilla qubits. The second subsection of this publication contains the description of modelled Boolean functions. To reduce both: the number of needed functions and the size of circuits constructed in the fifth subsection, the exact number of function of n variables was designated – equations (3) and (4). In the third subsection the algorithm for calculating matrix form of quantum operator is presented. This operator solves formulated problem – i.e. it transforms initial state (5) of n-qubit register to the state (7) from which the value of modelled Boolean function can be read. Next subsection contains the characteristic of quantum operator's matrix form U. The matrices $U_{2^n \times 2^n}$ are some permutations of identity matrix's $I_{2^n \times 2^n}$ columns, so they are binary and unitary matrices. Additionally, if $U \neq I$ then U is projectable to positive integer sequence (c), which consists of two arithmetic progressions (with common difference of 1) – the progressions' elements may occur alternately in (c). This special feature of matrices U allows to determine the number of needed operators and also to simplify the algorithm of constructing quantum gates circuit performing U.

The fifth subsection refers to the problem of quantum circuits' constructing with use of known quantum gates. To perform U three operations are used: Toffoli gate, *Roll* (13) and $Roll^{-1}$. The Toffoli gate is well-known universal operation [6]. The way of implementing *Roll* is shown at the Fig. 2 ($Roll^{-1}$ is the reverse operation to *Roll*). Using given in this subsection algorithm it is possible to construct a circuit calculating the value of Boolean function.

Adres

Joanna WIŚNIEWSKA: Wojskowa Akademia Techniczna, Wydział Cybernetyki,
Instytut Systemów Informatycznych, ul. Kaliskiego 2, 00-908 Warszawa, Polska,
jwisniewska@wat.edu.pl.