MAREK SAWERWAIN
Instytut Sterowania i Systemów Informatycznych,
Uniwersytet Zielonogórski
ul. Podgórna 50, 65-246 Zielona Góra
e-mail: M.Sawerwain@issi.uz.zgora.pl

PWT 2006
Poznańskie Warsztaty Telekomunikacyjne
Poznań 7 - 8 grudnia 2006

SYMULACJA OBLICZEŃ KWANTOWYCH – SYSTEM QCS W ZASTOSOWANIACH DLA EDUKACJI

***Streszczenie*: Informatyka kwantowa to z pewnością przyszłość informatyki. Na obecnym etapie z braku rozwiązań sprzętowych, szczególnie istotne są symulatory obliczeń kwantowych. Przeprowadzanie eksperymentów jest bardzo trudne oraz niewątpliwe drogie. Toteż stosowanie programowych symulatorów jest jedynym dostępnym aktualnie rozwiązaniem istotnym dla szeroko pojętej edukacji oraz dla badań. Ponieważ, symulator programowy, pozwala na wstępne sprawdzenie poprawności teoretycznych założeń dla układów kwantowych.**

1. Wstęp

Informatyka kwantowa to dynamicznie rozwijająca się dyscyplina z którą wiązane są duże nadzieje. Dzisiaj dostępne możliwości techniczne, nie są wystarczające, aby tworzyć skomplikowane układy kwantowe. Obecne osiągnięcia w dziedzinie fizycznej implementacji obliczeń kwantowych, to zaledwie siedem qubitów, za pomocą których w technologii NMR, I.L.Chuang dokonał fizycznej implementacji algorytmu Shora [2,4,5]. Jest to jeden z głównych powodów dla którego symulacje obliczeń kwantowych stają się bardzo istotne. Innym problemem, są koszty tego typu eksperymentów, stosowanie symulatora, pozwala na sprawdzenie pewnych rozwiązań oraz wyeliminowanie podstawowych błędów, jakie napotkać podczas wykonywania tak wyrafinowanego eksperymentu.

Specyficzne cechy jak superpozycja oraz splątanie, występujące w modelu obliczeń kwantowych powodują, że do przeprowadzenia obliczeń wymagana jest w najbardziej ogólnym przypadku wykładnicza ilość dostępnej pamięci. Co niestety, ogranicza wielkość rejestru kwantowego na którym możliwe jest przeprowadzanie eksperymentu, w przypadku typowych komputerów PC do 25-26 qubitów.

2. Model obliczeń kwantowych

Podstawowym pojęciem jest qubit i jest on reprezentowany przez następującą kombinację liniową

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad (1)$$

Współczynniki alpha i beta są nazywane amplitudami prawdopodobieństwa i są to liczby zespolone spełniające następującą zależność: $|\alpha|^2 + |\beta|^2 = 1$. W definicji qubitu zostały użyte dwa wektory tzw. wektory bazowe o następującej postaci:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \; oraz \; |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad (2)$$

Natomiast rejestr kwantowy to układ kilkunastu qubitów:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots \otimes |\psi_n\rangle \qquad (3)$$

Jest to iloczyn tensorowy poszczególnych qubitów, wyniku którego otrzymujemy wektor o wymiarze:

$$D(|\psi\rangle_n) = 2^n \qquad (4)$$

Niemal zawsze symbol iloczynu tensorowego jest pomijany, czyli zapis $|0\rangle \otimes |1\rangle$ jest skracany do $|01\rangle$.

Wymienione, we wstępie, zjawisko splątania nie pozwala na podział rejestru w dowolny sposób. Jedynie dekompozycja Schmida, dopuszcza podział rejestru, jednakże owa dekompozycja nie jest przydatna do przeprowadzania symulacji. Co pociąga za sobą następującą zależność:

$$U|x\rangle = |y\rangle, \qquad (5)$$

opisującą podstawową technikę przeprowadzania symulacji obliczeń kwantowych. Stan kwantowy oznaczony przez wektor x, jest przekształcany za pomocą macierzy unitarnej do nowego stanu. Stan psi ma wymiar wykładniczy oznacz to iż ilość pamięci RAM potrzebnej do poprawnej reprezentacji macierzy U są ogromne.

W ogólnym przypadku złożoność obliczeniowa symulacji obliczeń kwantowych jest określona (pomijając wszelkie stałe) następującą zależnością wykładniczą, gdzie n oznacza ilość qubitów znajdujących się w układzie:

$$T(n) = 2^n \qquad (6)$$

Zakładając, że chcemy symulować układ złożony ze stu qubitow, potrzebna jest macierz i dwa równe sobie wektory stanu. Macierz U posiada wymiary 2^100 na 2^100, każdy element macierzy to dwie liczby zespolone, co oznacza, że potrzebujemy ,,tylko":

2^100 * 2^100 * 8=25711008708143844408
671393477458601640355247900524685364822016

W tym miejscu, warto postawić pytanie czy kiedykolwiek uda się zbudować wydajniejszy system symulacji obliczeń kwantowych. Wydaje się, iż raczej tak się niestanie. W tym miejscu stosowne, wydaje się przytoczenie tezy Churcha-Turinga z lat sześćdziesiątych:

> Any model of computation can be simulated on a probabilistic Turing machine with at most a polynomial increase (i.e. efficiently) in the number of elementary operations required.

Wniosek z tezy jest następujący, jeśli dany problem jest trudny do rozwiązania, to takim pozostanie dla każdego klasycznego komputera którego model obliczeniowy można przedstawić za pomocą maszyny Turinga i wielomianowej ilości operacji elementarnych.

Jednak komputer kwantowy potrafi realizować trudne problemy przykładem jest alg. Shora w czasie krótszym, nie osiągalnym dla komputerów klasycznych. Co więcej, problem Deutscha-Jozsy posiada rozwiązane w czasie wielomianowym na maszynie kwantowej i wynik jest otrzymywany w sposób deterministyczny. Co pozwala, na podanie następującego stwierdzenia: hipoteza Churcha-Turinga nie jest prawdziwa.

W pracy [3] R.P.Feynman argumentuje, iż komputer klasyczny nigdy nie pozwoli na efektywną i pełną symulację zachowania się systemu kwantowego w czasie wielomianowym. Uwaga Feynmana może zostać przekształcona do następującego lematu:

> Zakładamy że istnieje algebra która pozwala na efektywną reprezentacją rejestru kwantowego obejmującego splątanie dla n qubitów. Symbolem $\hat{\otimes}$ oznaczamy specjalny operator, który umożliwia złożenie poszczególnych stanów z zachowaniem splątania pomiędzy qubitami:
>
> $$|\psi\rangle = |\psi_1\rangle \hat{\otimes} |\psi_2\rangle \hat{\otimes} \ldots \hat{\otimes} |\psi_n\rangle$$
>
> To mimo tej efektywnej algebry wektor nadal zwiera wykładniczą ilość informacji o stanie. Proces pomiaru w najgorszym przypadku potrzebuje $O(2^n)$ operacji elementarnych.

Szkic dowodu jest następujący: niech T reprezentuje klasyczną maszynę Turinga. Na taśmie maszyny został zapisany wektor dla n qubitów, a ilość pól użytych niech wynosi w. W przypadku zapisu bez kompresji należy użyć 2^n wartości zespolonych. Podczas procesu pomiaru całości rejestru dla nierównej superpozycji, rejestr może przejść do jednego z 2^n stanów bazowych. Ponieważ, nie wiadomo, gdzie znajduje się potencjalnie największa amplituda prawdopodobieństwa, w najgorszym przypadku należy wykonać 2^n operacji porównywania.

Dlatego, wiele innych podejść do symulacji algorytmów kwantowych [1, 6] poprawia wydajność symulacji, jednak naturalnie nie przełamuje złożoności symulacji obliczeń kwantowych.

3. System QCS

Stawiając sobie następujące pytanie, czy pomimo ograniczeń krótko przedstawionych w poprzednim punkcie zasadne jest projektowanie oprogramowania do przeprowadzania obliczeń kwantowych, odpowiedź jest jak najbardziej twierdząca. Powody dla których warto pokusić się o projektowanie tego typu oprogramowania są następujące:

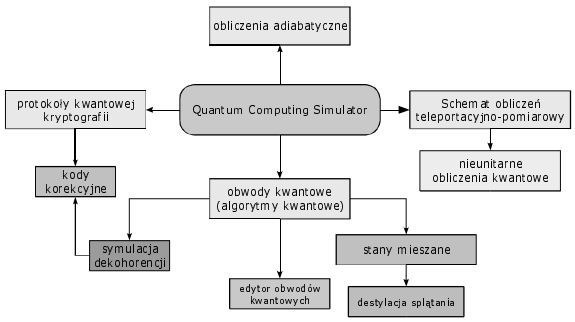
- brak dobrego pakietu do dydaktyki
- brak pakietu który wspomagałby prace naukowe np.: w projektowaniu różnego rodzaju układów kwantowych
- brak narzędzia które obejmowałoby wszystkie aktualnie stosowane techniki symulacji (np.: brak pakietu do symulacji układów opartych na stanach ortogonalnych i CHP)
- żaden z istniejących pakietów oprogramowania, nie oferuje symulacji modelu teleportacyjno-pomiarowego
- z wyjątkiem symulatora FQC [15] żaden z pakietów nie jest bezpośrednio przeznaczony do symulacji na systemach klastrowych i nie wykorzystuje maszyn wieloprocesorowych bądź systemów równoległych

Mając na uwadze dostępność pakietu, i jego rolę edukacyjną, system QCS [7, 8] powstał na bazie darmowego oprogramowania. Zostały zastosowane min. następujące pakiety:

- kompilator GCC v4.x
- kompilator firmy Intel dla języka C oraz Fortran (8.0 oraz 9.0)
- system skryptowy SWIG v1.3.xx
- język Python v2.4
- pakiety do algebry liniowej LAPACK, BLAS, ATLAS
- pakiet do obliczeń równoległych MPICH-G2 (Win32, Linux, Clusterix)

Wszystkie wymienione pakiety są oprogramowaniem darmowym opartym o licencję typu GNU GPL lub podobną. Co z jednej strony, pozwala na radykalne obniżenie kosztów tworzenia oprogramowania, jednakże nie ogranicza w żadnym razie funkcjonalności oprogramowania. Ważną innowacją w porównaniu do innych tego typu pakietów [11, 12] jest oferowanie wielu różnych technik symulacji układów kwantowych.

Zastosowanie przede wszystkim kompilatora GCC pozwala na uzyskanie bardzo istotnej własności a mianowicie przenośności oprogramowania pomiędzy bardzo różne architektury sprzętowo-programowe. Oprogramowanie na obecną chwilę jest dostępne dla najpopularniejszych systemów jak środowisko Windows w wersjach 32 i 64 bitowych oraz analogicznie system Linuks. Oprogramowanie było także testowane na platformie IA-64 Itanium II.



Rys 1. Zawartość systemu QCS

Planowana zawartość systemu symulacji modelu obliczeń kwantowych, obejmuje kilkanaście obszarów. Rysunek 1 prezentuje zawartość systemu QCS. Obecnie system można wykorzystywać poprzez specjalne API przygotowane dla skryptowego języka Python oraz specjalny język programowanie zbliżony do asemblera o nazwie QASM. Język ten powstał w celu łatwiejszego wykorzystania systemu w architekturach równoległych. Stosowanie w takim przypadku języka Python niepotrzebnie obciąża system oraz utrudnia implementację systemu.

4. Przykłady zastosowań QCS

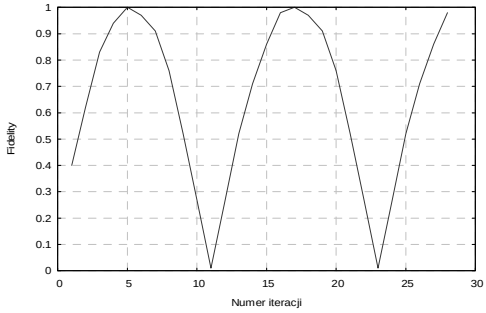
Rysunek 2 przedstawia krótki skrypt korzystający z systemu QCS. Jego zadaniem jest zebranie informacji o podobieństwie pomiędzy stanem aktualnym rejestru q1 a rejestrem q który reprezentuje bazę danych w której szukamy pewnego elementu. Skrypt ten jest realizacją alg. Grovera [10], szukania elementu w bazie danych. Indeks tego elementu jest reprezentowany przez rejestr q1. Wartość Fidelity powinna wskazać o ile stan aktualny stan rejestru q przetwarzanego za pomocą alg. Grovera przypomina szukany stan reprezentowany przez rejestr q1.

System QCS oferuje naturalnie odpowiedni zestaw bramek kwantowych za pomocą których można zrealizować w pełni algorytm Grovera, jednakże dla ułatwienia zostały przygotowane metody które ułatwiają wygenerowanie odpowiednich macierzy. Metoda o nazwie *GroverChangeSignGen* odpowiada za operator zmiany znaku amplitudy, natomiast druga z metod zgodnie z nazwą *GroverTurnAroundMean* tworzy macierz operatora obrotu wokół średniej.

```
import qcs

q1=qcs.QubitReg(6)
q1.Reset()
q1.SetKet("000001")

q=qcs.QubitReg(6)
q.Reset()
q.Had()

m1=q.GroverChangeSignGen(1);
m2=q.GroverTurnAroundMean(2);

print
for i in range(1,20):
  q.MatrixApply(m1)
  q.MatrixApply(m2)
  f=qcs.Fidelity(q1.GenDenMat(),q.GenDenMat())
  print "Fidelity "
  f.Pr()
```

Rys 2. Krótki skrypt wykorzystujący system QCS do badania jak stan aktualny w poszczególnych iteracjach alg. Grovera zbliża się do rozwiązania końcowego.

Wynik działania tego skryptu można przedstawić za pomocą wykresu na Rysunku 3. Ponieważ, algorytm Grovera jest algorytmem iteracyjnym jego cykliczne wywołania powodują iż w okresie określonym przez funkcję $\sqrt{n}$ (gdzie, n to rozmiar bazy danych) algorytm będzie zbliżał się do rozwiązania z największą wartością miary Fidelity.



Rys 3. Zmiana wartości Fidelity w algorytmie Grovera dla układu o sześciu qubitach, gdzie stanem szukanym jest stan bazowy $|000001\rangle$.

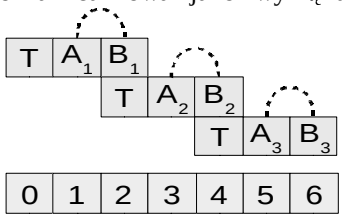
4.1 Teleportacja kwantowa

Innym przykładem jest symulacja protokołu teleportacji kwantowej przedstawionego w pracy [9], pełny skrypt jest przedstawiony na Rysunku 4. W tym skrypcie używany jest trójqubitowy rejestr. Gdzie, qubit o numerze zero reprezentuje teleportowany qubit, natomiast dwa pozostałe qubity stanowią kanał Einsteina-Podolskiego-Rosena, poprzez który zostanie przesłana informacja zawarta w qubitcie „zerowym" rejestru kwantowego.

Teleportacja kwantowa może także zostać wykorzystana jako medium komunikacyjne, które w efekcie może stanowić odpowiednik klasycznej sieci. Choć oryginalny protokół teleportacji kwantowej, wymaga przesłania dodatkowej informacji klasycznej, to sam kanał kwantowy wykorzystujący splątanie jest kanałem bezpiecznym który zapewnia bardzo wysokie bezpieczeństwo. Dodatkowe przesyłane bity klasyczne, wbrew pozorom, nie zdradzają jaka informacja została przesłana za pomocą protokołu teleportacji.

Można, połączyć kaskadowo fragmenty rejestru kwantowego, gdyż splątanie pomiędzy fragmentami rejestru będzie się utrzymywać nawet po fizycznym rozdzieleniu qubitów (choć technicznie jest to dość trudne do zrealizowania).

Poniżej zaprezentowano idę kaskadowej teleportacji, dla rejestru o siedmiu qubitach. W przypadku systemu QCS taka symulacja jest łatwa do implementacji, dzięki szerokim możliwością języka Python, gdyż samą teleportację można zrealizować jako zwykłą funkcję:



```
#! /usr/bin/python

import qcs

_TELEPORT_Q = 0
_ALICE_Q = 1
_BOB_Q = 2

q=qcs.QubitReg(3)
q.Reset()
q.SetKet("000")

q.HadN(_TELEPORT_Q)
q.HadN(_ALICE_Q)
q.CNot(_ALICE_Q, _BOB_Q)

q.CNot(_TELEPORT_Q, _ALICE_Q)
q.HadN(_TELEPORT_Q)

q.Pr() ; print

v=q.MeasureN(_TELEPORT_Q, _ALICE_Q)

if v==0:
  q.Pr()
if v==1:
  q.NotN(_BOB_QUBIT)
  q.Pr()
if v==2:
  q.PhaseFN(_BOB_QUBIT)
  q.Pr()
if v==3:
  q.NotN(_BOB_QUBIT)
  q.PhaseFN(_BOB_QUBIT)
  q.Pr()
```

Rys 4. Skrypt przeznaczony do symulacji teleportacji kwantowej nieznanego stanu qubitu.

4.2 Wydajność systemu

Oprócz możliwości tworzenia skryptów w języku Python a nawet całych programów za pomocą innych języków programowania jak Java czy bezpośrednio w języku C, system QCS oferuje język QASM nazwany roboczo kwantowym asemblerem. Język ten powstał jak już wspomniano aby ułatwić korzystanie z systemu w wersji równoległej. Rysunek 5 przedstawia skrypt w QASM którego zadanie to wprowadzenie rejestru w superpozycję za pomocą bramek Hadamarda. Wielkość rejestru kwantowego określa pierwsza linia skryptu. Zwiększając wielkość rejestru, lecz pozostawiając niezmienione instrukcje można zmierzyć wydajność systemu komputerowego. Poniższa tabela prezentuje czasy uzyskane na maszynie AMD Athlon 3200+, system WIN32 tryb 32-bitowy:

qubits	10	20	21	22	23	24	25
czas	≈0.0s	1.5s	3.1s	6.5s	13.5s	28.2s	58.7s

Jak widać dodanie jednego nowego qubitu, powoduje dwukrotne wydłużenie czasu działania symulacji maszyny kwantowej, co jest zgodne ze wcześniejszymi uwagami o złożoności obliczeniowej.

```
.qubits 10
.start
  reset
  had 0
  had 1
  had 2
  had 3
  had 4
  had 5
  had 6
  had 7
  had 8
  had 9
.end
```

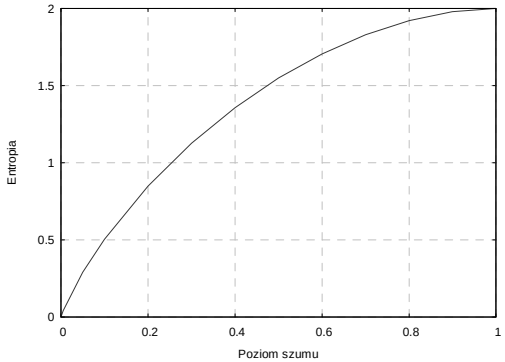
Rys 5. Prosty skrypt w języku QASM który może zostać wykorzystany jako tester wydajności komputera

4.3 Inne zastosowania QCS

System QCS został także wyposażony możliwość badania wpływu zakłóceń na stan kwantowy reprezentowany przez macierz gęstości. W QCS dostępna jest domyślna funkcja AddNoise które zgodnie z następującym wzorem:

$$\rho_f = (1-f)\rho_0 + \frac{fI_d}{d} \qquad (7)$$

gdzie przez f oznaczamy stopień szumu od zera do jedności natomiast I_d / d to stan maksymalnie zmieszany a macierz ρ_0 reprezentuje stan czysty. Funkcja AddNoise, dodaje do określonego stanu szum powodujący splątanie poszczególnych części rejestru, zaburzając w ten sposób jego oryginalny stan:



Rys 6. Zależność pomiędzy poziomem entropii, a stopniem szumu wprowadzonym do stanu, kwantowego reprezentowanego przez macierz gęstości

Wykorzystując wartość entropii można badać jak szum wpływa na czystość rejestru oraz sprawdzać różnicę pomiędzy stanem wzorcowym a mieszanym za pomocą kilku różnych miar oraz metryk dostępnych w systemie QCS:

- Fidelity,
- TraceDistance,

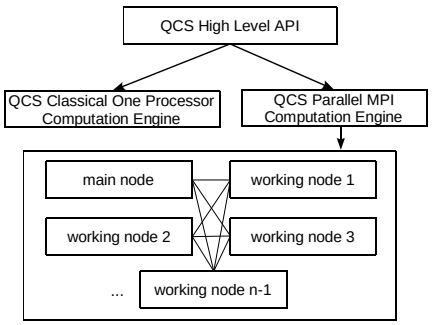
- BuresMetric,
- AngleMetric,
- CMetric.

```
ent=[ ]
for i in [0.0, 0.005, 0.05, 0.1, … , 0.9, 1.0 ]:
  m=q.GenDenMat ( )
  m.AddNoise ( i )
  e=m.Entropy ( )
  ent=ent + [ e ]
  del m
```

Rys 7. Fragment skryptu wyznaczającego wartość entropii w zależności od poziomu szumu w stanie kwantowym

5. Równoległa implementacja QCS

Oprócz możliwości pracy jednostanowiskowej, QCS oferuje też możliwość pracy w systemie równoległym. Sposób realizacji algorytmu jest podobny do typowych aplikacji MPI, występuje jeden węzeł nadzorujący pracę pozostałych stacji roboczych. Zadanie, jakie przed nim stoi, to dystrybucja poleceń do pozostałych węzłów.



Rys 8. Schemat przetwarzania rejestru kwantowego za pomocą systemu równoległego

Jedną z głównych zalet jakie można uzyskać w ten sposób, to wzrost wydajności, bowiem złożoność obliczeniowa dla symulacji jest określona jako:

$$T_{DP}(L) = \left\lceil \frac{2^L}{N_{DP}} \right\rceil \qquad (8)$$

Przy czym, jest to opis przypadku idealnego, gdy nie występuje wymiana informacji pomiędzy węzłami oraz ilość węzłów jest określona przez relację:

$$N_{DP} = \left\lfloor \log_M 2^L \right\rfloor \qquad (9)$$

W wielu przypadkach, pomiędzy węzłami trzeba przesłać dodatkową informację. Najprostszy przykład to zastosowanie jednoqubitowej bramki dla rejestru. Po przekształceniach rejestr przyjmuje następującą postać:

$$\begin{pmatrix} \alpha\alpha_1 + \beta\alpha_3 \\ \alpha\alpha_2 + \beta\alpha_4 \\ \gamma\alpha_1 + \delta\alpha_3 \\ \gamma\alpha_2 + \delta\alpha_4 \end{pmatrix}$$

Ponieważ, jedynym z możliwych sposobów rozproszenia rejestru na kilka maszyn jest równomierny podział, więc do poszczególnych węzłów należy przesłać dodatkową informację, aby lokalne obliczenia były w pełni poprawne. Z drugiej strony dla bramek wieloqubitowych, w wielu przypadkach np.: dla bramki typu CNot, nie zawsze istnieje potrzeba wymiany jakiejkolwiek informacji pomiędzy węzłami, bowiem równomierne rozproszenie wektora stanu powoduje, iż operacje są wykonywane w całości na poszczególnych węzłach. Poniższy przykład to potwierdza, bramka CNot dokonuje zamiany tylko dwóch amplitud:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_4 \\ \alpha_3 \end{pmatrix}$$

Uruchamianie skryptów, w tym przypadku, jak już to wcześniej zasygnalizowano możliwe jest przy wykorzystaniu języka QASM. W tym trybie pracy nie są dostępne wszystkie funkcje metryczne, ponieważ wymagają one tworzenia ogromnych macierzy a operowanie na takich ilościach danych (wielkości liczone w setkach gigabajtów) jest niestety niemożliwe. Jednakże, możliwe jest obliczanie Fidelity oraz dynamiczne wyznaczanie śladów częściowych względem wybranych qubitów. Zastosowane rozwiązanie, pozwala na dynamiczne wyznaczenie macierzy gęstości bez fizycznej generacji tej macierzy.

Zastosowany algorytm również ukazuje wykładnicze zależności, pomiędzy wykonaniem skryptu a ilością węzłów czy uruchomionych procesów na maszynie wieloprocesorowej. Przykładowe czasy wykonania procedury przeprowadzającej odwrotną transformatę Fouriera dla 20 qubitów, w algorytmie Shora, na maszynie Intel Itanium 2 1.4Ghz posiadającej cztery procesory są następujące:

	1-proc	3-proc	5-proc
1q gate	0.3s	0.2s	0.1s
2q gate	0.7s	1.5s/0.4s	1.4s/0.4s

Czasy w nawiasach dla bramek dwu qubitowych, zawierają także czas potrzebny do wymiany informacji pomiędzy węzłami. Natomiast druga wielkość, oznacza czas bez wymiany informacji. Niestety, w przypadku pracy poprzez sieć Ethernet, potrzebna jest dodatkowa kompresja przesyłanych danych, aby nie przesyłać zbyt dużej ilości informacji, co dodatkowo zwiększa czas pracy pojedynczego węzła.

```
def iqft_generator(_r1):
  i=0;
  print "; iqft for ", _r1, " qubits"
  print "; begin code"
  while i < _r1:
    print "had ", i
    for j in range(i,0,-1):
      print "crotalpha ", j, ", ", _r1-i-1, ", ",j+1
     i=i+1
```

Rys 9. Krótki skrypt, do generowania IQFT w alg. Shora

Jednak największą zaletą jest zwiększenie wielkości rejestru kwantowego, choć to zależy od ilości dostępnych

węzłów, możliwe jest także dalsze ograniczenie użycia pamięci, jeśli dane w węzłach będą przechowywane za pomocą nieskomplikowanej techniki kompresji kodowania długości serii (ang. RLE).

W tym przypadku, możliwe stają się nawet symulacje układów 40, 50, a nawet do 60 qubitów. Jednak, ze względu na konieczność indeksowania poszczególnych amplitud, system QCS ogranicza ilość qubitów do 64, ponieważ dokładnie tyle bitów posiada typ *unsigned long long* który jest wykorzystywany do indeksowania amplitud prawdopodobieństwa zapisanych w rejestrze kwantowym.

Wyjątkiem są tryby CHP i PQC [7], gdzie można efektywnie symulować tysiące qubitów, jednak te modele nie są w pełni modelami kwantowymi, szczególnie tryb pracy PQC.

6. Podsumowanie

Opisany system symulacji kwantowego modelu obliczeń, był już stosowany podczas laboratorium przedmiotu „Wstęp do informatyki kwantowej" na czwartym roku studiów magisterskich na kierunku informatyka na Uniwersytecie Zielonogórskim. System był wykorzystywany przez studentów do realizacji zadań w ramach laboratorium. Zadania te dotyczy min. badania algorytmu Grovera, badania jakości kopiowanie stanów kwantowych oraz opracowywania obwodów kwantowych realizujących algorytm Shora.

Należy też podkreślić interesujący aspekt symulacji kwantowych, a mianowicie zastosowania tego systemu do badania wydajności systemu komputerowego. Zarówno pod względem wydajności obliczeniowej procesora oraz wydajności operacji na pamięci.

System QCS podlega także ciągłemu rozwojowi, aktualnie ulepszeniu podlega moduł do równoległych symulacji obliczeń kwantowych.

Ostatnim ważnym aspektem są naturalnie zastosowania naukowe, szczególnie w dziedzinie testowania obwodów kwantowych (naturalnie w ograniczonym zakresie, co do wielkości obwodu) np.: obwodów przybliżonego kopiowania, czy obwodów realizujących przybliżone porównywanie stanów. Odpowiedni skrypt upraszcza testowanie np.: parametrów bramek obrotu.

7. Podziękowania

Serdecznie chciałbym podziękować prof. Romanowi Gielerakowi za cenne uwagi, jakie przyczyniły się do powstania tego artykułu. Jak również za czas, który poświęcił przy testowaniu tworzonego przez nas systemu QCS, co przyczyniło się do ogólnej poprawy jakości systemu.

Podziękowania kieruję też do kolegów z Instytutu Sterowania i Systemów Informatycznych z grupy Q-INFO, którzy współpracowali przy tworzeniu pakietu do obliczeń kwantowych i opracowania testów systemów dla wersji jednostanowiskowej jak i również wersji równoległej.

Spis literatury

[1] S.Aaronson, D.Gottesman, *Improved Simulation of Stabilizer Circuits*, Aug 2004, http://arxiv.org/abs/quant-ph/0406196.

[2] I.L.Chuang, L.M.K.Vanerspyen, X.Zhou, D.W.Leung, S.Lloyd, *Experimental realization of a quantum algorithm*, Nature 393, 143-146 (1998), http://arxiv.org/abs/quant-ph/9801037.

[3] R.P.Feynman, *Simulating physics with computers*, International Journal of Theoretical Physics 21:6/7. 467-488 (1982).

[4] C.Lavor, L.R.U.Manssur, R.Portugal, *Shor's Algorithm for Factoring Large Integers*, Mar 2003, http://arxiv.org/abs/quant-ph/0303175

[5] P.Shor:, *Algorithms for Quantum Computation: Discrete Logarithm and Factoring, Proc. 35th Annual Symposium on Foundations of Computer Science (1994)* 124-134 and SIAM J. Comput. 26 (1997) 1484-1509.

[6] K.De Raedt, K.Michielsen, H.De Raedt, B.Trieu, G.Arnold, M.Richter, Th.Lippert, H.Watanabe, N.Ito, *Massive Parallel Quantum Computer Simulator*, submitted to Computer Physics Communications.

[7] M.Sawerwain, *Symulator obliczeń kwantowych*, Computer Methods and Systems - CMS '05 : V konferencja. Kraków, Polska, 2005 .- Kraków: Oprogramowanie Naukowo-Techniczne, 2005 .- Vol. 2: Regular sessions, s. 185--190 .- ISBN: 83-916420-3-8.

[8] M.Sawerwain, *Równoległa implementacja symulacji obliczeń kwantowych*, Sieci i systemy informatyczne : teoria, projekty, wdrożenia, aplikacje : XIV konferencja. Łódź, Polska, 2006 .- Łódź : "Piątek Trzynastego Wydaw.", 2006, s. 241--244 .- ISBN: 83-7415-132-3.

[9] C.H.Bennett, G.Brassard, C.Crepeau, R.Jozsa, A.Peres, W.K.Wooters, *Teleporting an unknown state via dual classical and Einstein-Podolsky-Rosen channels*, Physical Review Letters 70, 1895-1899 (1993).

[10] L.K.Grover, *A fast quantum-mechanical algorithm for database search*, Proceedings of the 28th Annual ACM Symposium on the Theory of Computing – STOC, 212-219 (1996), http://arxiv.org/abs/quant-ph/9605043.

[11] H.Touchette, P.Dumai, *QuCalc - The quantum computation package for Mathematica*, http://crypto.cs.mcgill.ca/QuCalc/, 2000.

[12] B.Omer, *Structured Quantum Programming, Ph.D. Thesis*, TU Vienna, Vienna, 2003.

[13] P.Gawron, J.A.Miszczak, *Numerical simulations of mixed states quantum computation*, www.arxiv.org/quant-ph/0406211

[14] Ogólno dostępny symulator obliczeń kwantowych FQC, dostępny pod adresem WWW: http://www.qc.fraunhofer.de

Marek Sawerwain
e-mail: M.Sawerwain@issi.uz.zgora.pl