O'REILLY®

Komputer kwantowy

Programowanie, algorytmy, kod

Eric R. Johnston
Nicholas Harrigan
Mercedes Gimeno-Segovia

Helion

Tytuł oryginału: Programming Quantum Computers: Essential Algorithms and Code Samples

Tłumaczenie: Katarzyna Wojtkowiak

ISBN: 978-83-283-6778-4

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/komkwa
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/komkwa.zip

Printed in Poland.

Spis treści

Część II. Prymitywy QPU

Część IV. Perspektywy

ROZDZIAŁ 1.

Wprowadzenie

Niezależnie czy zajmujesz się inżynierią oprogramowania, grafiką komputerową, analizą danych, czy też jesteś po prostu entuzjastą komputerów, ta książka została napisana, żeby pokazać Ci możliwości informatyki kwantowej. Kiedy już przekonasz się, że dziedzina ta może być dla Ciebie istotna, dowiesz się, jak wykorzystać procesory kwantowe w praktyce.

Aby rzecz ułatwić, kolejne rozdziały *nie* zawierają dokładnych wyjaśnień reguł fizyki kwantowej (praw stojących za informatyką kwantową) czy nawet teorii informacji kwantowej (opisującej, jak te prawa determinują zdolność przetwarzania informacji). Zamiast tego zaprezentowaliśmy w nich robocze przykłady, dające wgląd w możliwości nowej i fascynującej technologii. Co najważniejsze, pokazany w książce kod ćwiczeniowy może być modyfikowany i ulepszany. To z kolei pozwoli Ci na najbardziej efektywną naukę: poprzez praktykę. Podczas stosowania kluczowych pojęć zostaną one przy okazji omówione — i tylko o tyle, o ile rozwijają wyczucie potrzebne do pisania kwantowych programów.

Mamy nieśmiałą nadzieję, że zainteresowani czytelnicy będą w stanie wykorzystać to zrozumienie i rozszerzyć zastosowanie aplikacji kwantowych na dziedziny, o których fizycy kwantowi mogli nawet nie słyszeć. Wprawdzie nadzieja na wywołanie rewolucji kwantowej nie jest *taka* skromna, ale bycie pionierem jest zdecydowanie ekscytujące.

Wymagania wstępne

Fizyka, która stoi za informatyką kwantową, jest naszpikowana poważną matematyką. Tak samo fizyka, która stanowi podstawę działania tranzystorów, a przecież nauka C++ nie wymaga poznania ani jednego równania fizycznego. W niniejszej książce stosujemy to samo podejście, stawiające *w centrum programistę*, a matematykę pozostawiamy w cieniu. Po tym wstępie zamieszczamy krótką listę zagadnień, które mogą się okazać pomocne w przyswojeniu pojęć, jakie wprowadzamy:

- Znajomość struktur kontrolnych programowania (`if`, `while`, etc.). Używamy JavaScriptu, by zapewnić prosty dostęp do przykładów, które można uruchomić online. Jeśli JavaScript jest dla Ciebie nowością, ale masz jakieś doświadczenie w programowaniu, wiedza, jakiej potrzebujesz, prawdopodobnie jest do opanowania w ciągu godziny. Jeśli szukasz bardziej dokładnego wprowadzenia do JavaScriptu, możesz zobaczyć *Learning JavaScript* Ethana Browna (wydawnictwo O'Reilly).

- Trochę matematyki adekwatnej dla programisty:
  - zrozumienie funkcji matematycznych,
  - znajomość funkcji trygonometrycznych,
  - łatwość posługiwania się liczbami binarnymi, konwertowanie ich na liczby dziesiętne i odwrotnie,
  - zrozumienie podstawowych zagadnień z zakresu liczb zespolonych.
- Bardzo podstawowe zrozumienie, jak ocenić złożoność obliczeniową algorytmu (np. notacja *duże-O*).

Jedyną częścią książki, która wychodzi poza te wymagania, jest rozdział 13., w którym badamy zastosowanie obliczeń kwantowych w uczeniu maszynowym. Z powodu ograniczonego miejsca nasz przegląd sprowadza się jedynie do bardzo pobieżnego wprowadzenia w każdą z aplikacji uczenia maszynowego przed zaprezentowaniem, jaką komputer kwantowy ma przewagę. Chociaż w zamierzeniu treść ma być zrozumiała dla każdego czytelnika, ci, którzy chcieliby poeksperymentować z tymi aplikacjami, skorzystają, jeśli dowiedzą się więcej o uczeniu maszynowym.

To książka o programowaniu (nie budowie, nie badaniach) kwantowych komputerów, dlatego poradzimy sobie bez zaawansowanej matematyki i teorii kwantowej. Jednakże jeśli chciałbyś poznać bardziej akademicką literaturę na ten temat, rozdział 14. podaje wartościowe pozycje i łączy pojęcia przez nas wprowadzone z zapisem matematycznym używanym w środowisku naukowców, zajmujących się dziedziną informatyki kwantowej.

Czym jest QPU?

Pomimo swej wszechobecności termin **komputer kwantowy** może być trochę mylący. Przywołuje obrazy całkiem nowego i obcego rodzaju maszyny — takiej, która zamienia całe istniejące oprogramowanie na futurystyczną alternatywę.

W momencie pisania tej książki takie przeświadczenie jest powszechnym, ale bardzo dużym błędem. Nadzieja łączona z komputerami kwantowymi nie wiąże się z *wycofywaniem konwencjonalnych komputerów*, ale raczej z umiejętnością radykalnego rozszerzenia katalogu problemów możliwych do rozwiązania przez informatykę. Istnieją ważne problemy obliczeniowe, z którymi komputery kwantowe potrafią sobie poradzić, a których rozwiązanie byłoby niemożliwe na jakimkolwiek standardowym urządzeniu komputerowym, jakie kiedykolwiek moglibyśmy lub mieliśmy nadzieję zbudować[1].

[1] Jednym z naszych ulubionych sposobów na uwypuklenie tego jest przykład oparty na prowizorycznych wyliczeniach. Przypuśćmy, że normalne tranzystory mogłyby mieć wielkość atomu i chcielibyśmy zbudować zwykły komputer wielkości magazynu, żeby dogonił komputer kwantowy w rozkładaniu liczb pierwszych. Musielibyśmy upchać tranzystory tak ściśle, że stworzylibyśmy grawitacyjną osobliwość. Osobliwości grawitacyjne bardzo utrudniają obliczenia (i istnienie).

Ale co najważniejsze, procesor kwantowy ułatwia rozwiązanie tylko niektórych problemów (wiele z nich omówimy później) i jakkolwiek przypuszcza się, że zostanie odkrytych jeszcze więcej, to jednak mało prawdopodobne, że kiedykolwiek sensowne okaże się uruchamianie *wszystkich* obliczeń na komputerach kwantowych. Jeśli chodzi o większość zadań zajmujących cykl zegara w Twoim laptopie, komputer kwantowy nie radzi sobie wcale lepiej.

Innymi słowy — z punktu widzenia programisty — komputer kwantowy tak naprawdę jest koprocesorem. W przeszłości komputery wykorzystywały wiele koprocesorów, każdy z nich miał swoją specjalność, np. arytmetykę zmiennoprzecinkową, przetwarzanie sygnałów czy przetwarzanie grafiki w czasie rzeczywistym. Mając to na uwadze, będziemy używać terminu **QPU** (ang. *Quantum Processing Unit*) w odniesieniu do urządzenia, na którym uruchamiany jest nasz kod. Myślimy, że takie podejście podkreśla kontekst, w którym informatyka kwantowa powinna być rozumiana.

Tak jak w przypadku innych współprocesorów, np. GPU (ang. *Graphics Processing Unit*), programowanie dla komputerów kwantowych wymaga od programisty napisania kodu, który będzie uruchamiany głównie na CPU (ang. *Central Processing Unit*) normalnego komputera. CPU wydaje polecenia współprocesorowi QPU tylko dla zainicjowania zadań dostosowanych do jego możliwości.

Podejście praktyczne

Praktyczne przykłady tworzą kręgosłup tej książki. Jednakże w czasie jej pisania rozwinięte QPU o powszechnym zastosowaniu jeszcze nie istnieją — jak zatem uruchomisz nasze kody? Na szczęście (i to jest ekscytujące) nawet w czasie pisania kilka prototypów QPU jest dostępnych — dostęp do nich można uzyskać z chmury. Co więcej, dla mniejszych problemów możliwa jest symulacja zachowania QPU na konwencjonalnym sprzęcie komputerowym. Aczkolwiek symulacja większych programów QPU okazuje się niemożliwa, to w przypadku krótszego kodu wiedza o tym, jak kontrolować QPU, sporo upraszcza. Kody z książki są kompatybilne z oboma scenariuszami i pozostaną przydatne do pracy i nauki nawet wtedy, gdy pojawią się bardziej rozbudowane QPU.

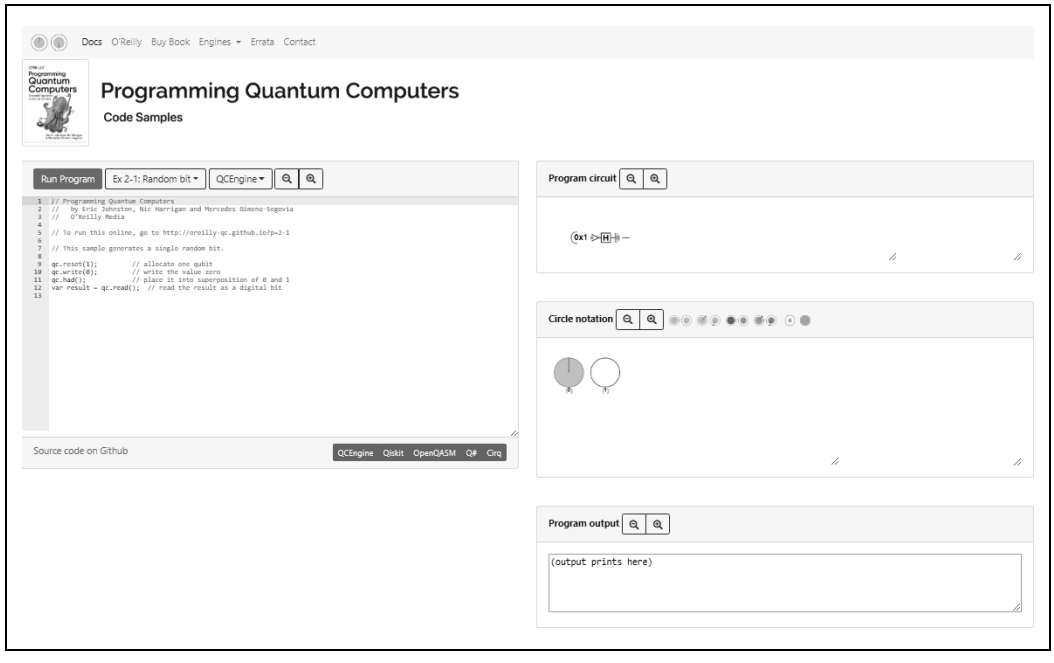
Istnieje wiele symulatorów QPU, bibliotek i dostępnych systemów. Listę linków do kilkunastu systemów z dobrym wsparciem możesz znaleźć pod adresem: *http://oreilly-qc.github.io*. Na tej stronie, kiedy to możliwe, udostępniamy kody z książki w różnych językach. Jednakże, aby zapobiec przeładowaniu książki kodem, przedstawiamy próbki jedynie w JavaScripcie — dla QCEngine. QCEngine to darmowy symulator obliczeń kwantowych online, pozwalający użytkownikom na uruchamianie kodu w przeglądarce — bez konieczności instalowania czegokolwiek. Ten symulator został opracowany przez autorów początkowo dla ich własnego użytku, a teraz stanowi dodatek do niniejszej książki. QCEngine jest dla nas szczególnie użyteczny, ponieważ raz — że może być uruchomiony bez instalowania jakiegokolwiek oprogramowania, a dwa — że zawiera *notację kołową*, którą w książce wykorzystujemy jako narzędzie wizualizacyjne.

Elementarz QCEngine

Skoro będziemy nagminnie korzystać z QCEngine, warto przeznaczyć trochę czasu na przyjrzenie się sposobowi pracy z symulatorem. Znajdziesz go na *http://oreilly-qc.github.io*.

Uruchamianie kodu

Interfejs webowy QCEngine, pokazany na rysunku 1.1, pozwala w prosty sposób tworzyć różne wizualizacje, które pomogą nam w analizie działania kodu. Możesz je wygenerować, wprowadzając po prostu kod do edytora QCEngine.



Rysunek 1.1. Interfejs użytkownika aplikacji QCEngine

Żeby uruchomić jeden z książkowych kodów, wybierz go z rozwijanej listy na górze edytora i kliknij przycisk *Run Program*. Pojawi się kilka nowych elementów interfejsu, żeby zwizualizować rezultat uruchomienia kodu (zob. rysunek 1.2).

Wizualizator obwodów kwantowych

Ten element przedstawia Twój kod w postaci obwodu. W rozdziałach 2. i 3. wprowadzamy symbole, których używa się w tych obwodach. Można również użyć tego pola do interaktywnego śledzenia programu krok po kroku (zob. rysunek 1.2).

Wizualizator notacji kołowej

Wyświetla tzw. zapis kołowy rejestru QPU (czy symulatora). W rozdziale 2. wyjaśnimy, jak tego używać.

Konsola wyjścia QCEngine

Tutaj pojawi się każdy tekst, który może zostać wyświetlony z użyciem polecenia `qc.print()` zawartego w kodzie (np. w celu wykrycia błędów). Wszystko wyświetlone za pomocą standardowej funkcji JavaScript `console.log()` nadal będzie przekazywane do konsoli JavaScript w Twojej przeglądarce.

Rysunek 1.2. Elementy interfejsu użytkownika w QCEngine do prezentowania rezultatów QPU

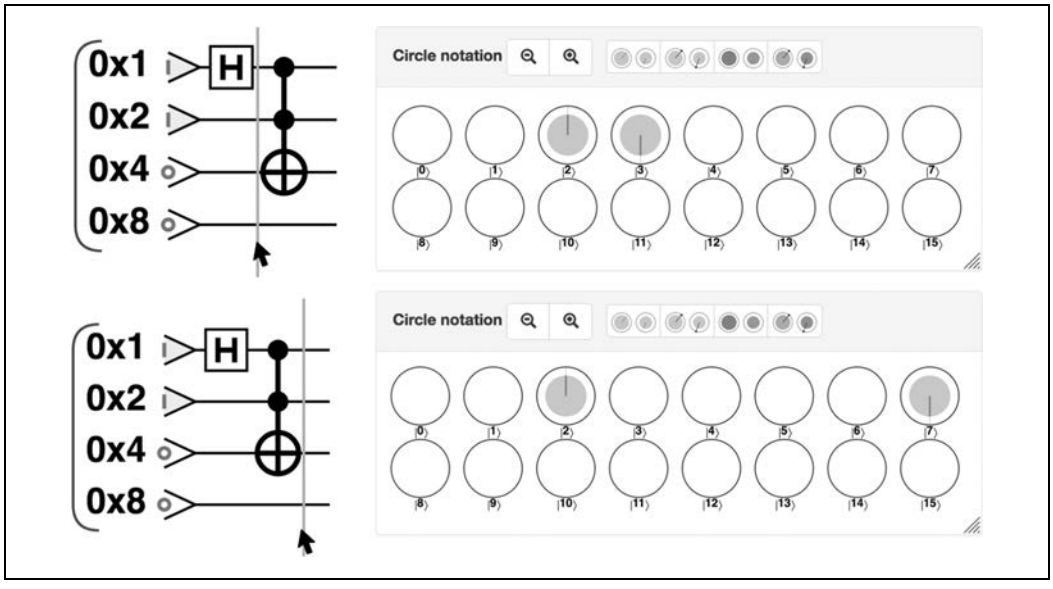
Poprawianie błędów w kodzie

Poprawianie błędów w programach QPU może być trudne. Dość często najprostszym sposobem na zorientowanie się, jak działa program, jest powolne przechodzenie go krok po kroku i badanie wizualizacji w każdym z kroków. Po najechaniu myszą na wizualizator powinieneś zobaczyć pionową, pomarańczową linię na ustalonej pozycji oraz szarą, pionową linię w obwodzie, w którym akurat znajduje się wskaźnik myszy. Pomarańczowa linia wskazuje, którą pozycję w obwodzie (więc także w programie) wizualizator obwodu aktualnie reprezentuje. Domyślnie jest to koniec programu, ale po kliknięciu na inne części obwodu wizualizator może pokazać konfigurację QPU dla tych punktów w programie. Na przykład rysunek 1.3 ilustruje zmiany w notacji kołowej wizualizatora po przełączeniu się pomiędzy dwoma różnymi krokami w domyślnym programie QCEngine.

Skoro masz dostęp do symulatora QPU, to pewnie chciałbyś już go wypróbować. Nie wstrzymuj się! W rozdziale 2. będziemy analizować kod coraz bardziej złożonych programów QPU.

Natywne instrukcje QPU

QCEngine jest jednym z kilku narzędzi pozwalających na uruchamianie i badanie kodu QPU — ale jak właściwie wygląda kod QPU? Konwencjonalne języki wysokiego poziomu są powszechnie używane do kontrolowania instrukcji QPU niskiego poziomu (jak już zdążyliśmy zauważyć przy omawianiu CQEngine na bazie JavaScriptu). W tej książce będziemy regularnie przechodzić między tymi poziomami. Opis programowania QPU za pomocą specyficznych operacji kwantowych na poziomie maszyny pozwoli nam pojąć fundamentalną, nowatorską logikę QPU. A jednocześnie zrozumiesz, jak manipulować tymi operacjami z poziomu języków wyższego stopnia, takich jak



Rysunek 1.3. Przechodzenie programu QCEngine za pomocą wizualizatorów obwodu i notacji kołowej

JavaScript, Python lub C++, które są bardziej pragmatycznym sposobem pisania kodu. Opracowywanie nowych języków programowania, dostosowanych do prawideł *informatyki kwantowej* jest aktywnie rozwijanym obszarem. Nie będziemy się tym zajmować w tej książce, ale ciekawski Czytelnik znajdzie odniesienia w rozdziale 14.

Żeby zaostrzyć Twój apetyt, w tabeli 1.1 podajemy niektóre z podstawowych instrukcji QPU. Każda z nich zostanie omówiona w następnych rozdziałach.

Tabela 1.1. Podstawowy zestaw instrukcji QPU

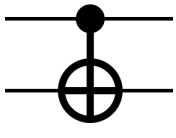
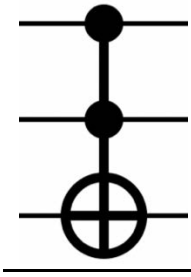
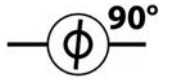
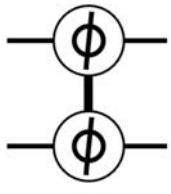
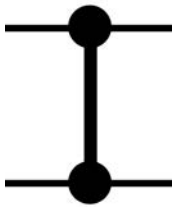
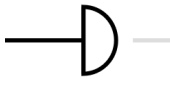
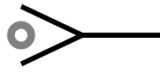
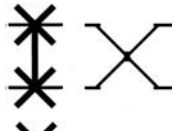
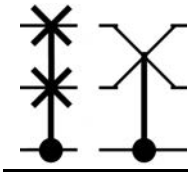
Symbol	Nazwa	Użycie	Opis
	NOT (także X)	`qc.not(t)`	rozumiane logicznie NOT
	CNOT	`qc.cnot(t, c)`	kontrolowane NOT: `if(c) then NOT(t)`
	CCNOT (Toffoli)	`qc.cnot(t, c1|c2)`	`if (c1 AND c2) then NOT(t)`

Tabela 1.1. Podstawowy zestaw instrukcji QPU — ciąg dalszy

Symbol	Nazwa	Użycie	Opis
	HAD (Hadamard)	`qc.had(t)`	bramka Hadamard
	PHASE	`qc.phase(kąt, c)`	Relatywna rotacja fazy
	Z	`qc.phase(180, c)`	Relatywna rotacja fazy o 180º
	S	`qc.phase(90, c)`	Relatywna rotacja fazy o 90º
	T	`qc.phase(45, c)`	Relatywna rotacja fazy o 45º
	CPHASE	`qc.phase(kąt, c1|c2)`	Warunkowa rotacja fazy (ang. *Conditional phase rotation*)
	CZ	`qc.phase(180, c1|c2)`	Warunkowa rotacja fazy o 180º
	READ	`val = qc.read(t)`	Odczytuje kubity, zwraca dane cyfrowe
	WRITE	`qc.write(t, val)`	Zapisuje konwencjonalne dane do kubitów
	ROOTNOT	`qc.rootnot(t)`	operacja pierwiastek z NOT
	SWAP (EXCHANGE)	`qc.exchange(t1|t2)`	Wymiana dwóch kubitów
	CSWAP	`qc.exchange(t1|t2, c)`	Warunkowa wymiana: `if(c) then SWAP(t1, t2)`

W przypadku każdej z tych operacji konkretne instrukcje i timing będą zależały od marki QPU i jego architektury. Jednakże jest to podstawowy zestaw operacji, który powinien być dostępny na każdej maszynie. Te operacje stanowią podstawę naszego programowania QPU, tak jak instrukcje `MOV` i `ADD` dla programistów CPU.

Ograniczenia symulatorów

Jakkolwiek symulatory oferują fantastyczną możliwość prototypowania małych programów QPU, to w porównaniu do prawdziwych QPU mają beznadziejnie małą moc. Jedynym sposobem zmierzenia mocy QPU jest określenie liczby *kubitów*, na których dany procesor może operować[2] (kwantowy równoważnik bitów, o którym wkrótce powiemy sobie znacznie więcej).

W momencie publikowania książki światowy rekord największej symulacji QPU wynosi 51 kubitów. W praktyce symulatory i sprzęt dostępne Czytelnikowi zazwyczaj będą w stanie podołać ok. 26 kubitom, zanim się zatrzymają.

Przykłady kodu napisaliśmy, mając to na uwadze. Jest to dobry punkt startowy, ale każdy dodany kubit podwoi pamięć potrzebną do przeprowadzenia symulacji, a prędkość zmniejszy o połowę.

Ograniczenia sprzętu

Największy sprzęt QPU dostępny w czasie pisania tej książki ma ok. 70 *fizycznych* kubitów, podczas gdy największy QPU dostępny publicznie, poprzez *open-source'owy* zestaw developerski Qiskit (*https://qiskit.org/*), zawiera 16 kubitów[3]. Pisząc „fizycznych", w odróżnieniu od logicznych, mamy na myśli to, że te 70 kubitów nie ma żadnej korekty błędów, co czyni je zaszumionymi i niestabilnymi. Kubity są dużo bardziej kruche niż ich normalne odpowiedniki, najmniejsza interakcja z otoczeniem może zepsuć obliczenia.

Praca z kubitami *logicznymi* pozwala programiście pozostać obojętnym na sprzęt QPU i zaimplementować jakikolwiek algorytm z podręcznika bez konieczności zamartwiania się konkretnymi ograniczeniami sprzętowymi. W tej książce skupiamy się wyłącznie na programowaniu z wykorzystaniem kubitów logicznych i chociaż przykłady są wystarczająco niewielkie, żeby je uruchomić na mniejszych QPU (takich jak te, które dostępne są w momencie publikacji), to nawet taka praca w oderwaniu od sprzętu fizycznego pozwoli na rozwinięcie umiejętności i intuicji, które pozostaną nieocenione, kiedy sprzęt rozwinie się w przyszłości.

QPU a GPU: charakterystyka ogólna

Idea programowania całkiem nowego rodzaju procesora może być przerażająca, nawet jeśli istnieje już społeczność na Stack Exchange (*https://quantumcomputing.stackexchange.com/*). Oto lista istotnych faktów na temat programowania QPU:

[2] Pomimo swojej popularności medialnej jako benchmark w informatyce kwantowej zliczanie kubitów, z którymi sprzęt może sobie poradzić, jest tak naprawdę uproszczeniem i niezbędne jest bardziej subtelne podejście do oceny prawdziwej mocy QPU.

[3] Te liczby mogą stać się przestarzałe w czasie, gdy książka będzie w sprzedaży!

- Bardzo rzadko program będzie uruchamiany *w całości* na QPU. Zazwyczaj program działający na CPU uruchomi instrukcje QPU, a potem uzyska wyniki końcowe.
- Niektóre zadania doskonale się nadają dla QPU, inne — nie.
- QPU działa na innym zegarze niż CPU i zazwyczaj ma swoje własne interfejsy sprzętowe przeznaczone dla urządzeń zewnętrznych (takie jak wyjścia optyczne).
- Typowy QPU ma swój własny, specjalny RAM, z którego CPU nie potrafi efektywnie korzystać.
- Prosty QPU będzie w formie chipa, do którego dostęp będzie miał laptop. Być może nawet takie QPU znajdzie się w obrębie innego chipa. Bardziej zaawansowany QPU jest dużym i drogim dodatkiem — i zawsze wymaga specjalnego chłodzenia.
- Wczesne QPU, nawet te proste, są wielkości lodówki i wymagają specjalnych gniazdek wysokiego napięcia.
- Po wykonaniu obliczeń projekcja wyniku jest przekazywana do CPU. Wynik większości pracy wewnętrznej QPU zostaje utracony.
- Proces naprawiania błędów w przypadku QPU może być skomplikowany; wymaga specjalnych narzędzi i technik. Przechodzenie przez program może być trudne i często najlepszym rozwiązaniem okazuje się wprowadzanie zmian do programu i obserwowanie efektu wynikowego.
- Optymalizacja, która przyspieszy jeden QPU, może spowolnić inny.

Brzmi ambitnie. Rzecz w tym, że w każdym z tych stwierdzeń możesz zastąpić *QPU* przez *GPU* i nadal będą prawdziwe.

Co prawda QPU są prawie pozaziemską technologią o niezwykłej mocy, jednak kiedy przychodzi do problemów, które możemy napotkać podczas nauki programowania, nie różnią się one od tych, z którymi wcześniej mierzyły się już pokolenia inżynierów. Oczywiście prawdą jest, że są pewne niuanse w programowaniu QPU, które są całkowicie nowatorskie (inaczej ta książka nie byłaby konieczna!), jednakże niesamowita liczba podobieństw powinna Cię uspokoić. Damy radę!

QPU: zrób pierwszy krok ku technologii przyszłości!

Komputery kwantowe nie są już tylko urządzeniami teoretycznymi. Nie są też futurystycznym monstrum, obcą maszyną, która zamieni całe istniejące oprogramowanie w jakąś jego niezrozumiałą alternatywę. Komputery kwantowe (QPU) staną się raczej radykalnym rozszerzeniem możliwości współczesnej informatyki, które pozwoli nam rozwiązać problemy dotychczas nierozwiązywalne. Istnieją ważne zadania, z którymi QPU potrafią sobie świetnie poradzić, a których rozwiązanie byłoby niemożliwe na jakimkolwiek standardowym urządzeniu komputerowym. Z drugiej strony z wieloma standardowymi obliczeniami QPU nie radzi sobie dużo lepiej niż najzwyklejszy laptop. Z punktu widzenia programisty zatem stanowi rodzaj koprocesora o ciekawych możliwościach.

Oto przewodnik po wspaniałym, nieodkrytym jeszcze do końca świecie informatyki kwantowej. Aby ją zrozumieć, niepotrzebny jest doktorat z fizyki kwantowej i wyższej matematyki. Dzięki tej książce opanujesz zestaw pojęć niezbędnych do zrozumienia działania QPU, dowiesz się, jakie problemy mogą rozwiązać aplikacje QPU, i nauczysz się korzystać z narzędzi do pisania programów dla QPU. Zaprezentowane tu koncepcje są bogato ilustrowane przykładami, które można łatwo uruchomić na darmowym symulatorze QCEngine. Istnieje też możliwość korzystania z fizycznych QPU (kilka prototypów QPU udostępniono w chmurze). Interesującą, choć nieco trudniejszą częścią przewodnika jest rozdział poświęcony zastosowaniu obliczeń kwantowych w uczeniu maszynowym.

W książce:

- koncepcje programowania procesorów kwantowych
- kubity, superpozycja i teleportacje kwantowe
- prymitywy QPU
- wzmacnianie amplitudy, kwantowa transformacja Fouriera i szacowanie fazy
- przykłady aplikacji QPU

Eric R. Johnston napisał symulator QCEngine. Był badaczem inżynierii kwantowej na Uniwersytecie Bristolskim i programował efekty filmowe dla Lucasfilm. Jest inżynierem kwantowym, akrobatą i gimnastykiem.

Dr Nicholas Harrigan jest fizykiem, programistą i popularyzatorem nauki. Pracuje na Uniwersytecie Bristolskim i jako architekt kwantowy w start-upie. Lubi się wspinać.

Dr Mercedes Gimeno-Segovia jest fizykiem kwantowym. Rozwija kolejne generacje technologii kwantowych. Pracuje nad projektem komputera kwantowego do ogólnego użytku. Nieźle gra na skrzypcach.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-6778-4
9 788328 367784
INFORMATYKA W NAJLEPSZYM WYDANIU
Cena: 67,00 zł