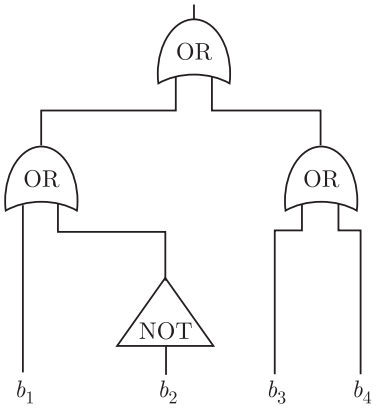
# O modelach obliczeń komputerowych

*Tomasz KAZANA*

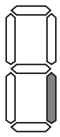
Zastanówmy się nad następującym pytaniem: czym jest komputer? Sądzę, że odpowiedź na tak zadane pytanie może zależeć w znacznej mierze od tego, kogo o to pytamy. Taka sytuacja nie jest, oczywiście, czymś wyjątkowym. Jeśli zamiast informatyką zajmiemy się cukiernictwem i zapytamy: czym jest tort, to też różne osoby będą różnie odpowiadać. Cukiernik opisujący tort widzi go jako kolejne poziome warstwy, które musi odpowiednio przygotować i w dobrej kolejności ułożyć. Łakomczuch – raczej widzi jego przekrój pionowy, stanowiący brzeg konkretnego kawałka. Jeśli zostaniemy przy gastronomii, ale przeskoczymy tym razem do piecyków kuchennych, znowu zaobserwujemy postulowaną dwoistość. Inżynier projektujący piecyk myśli o jego wydajności, o precyzji nastawienia temperatury, o tym, jak sprawić, żeby piecyk odpowiednio szybko się nagrzał itp., itd. Z kolei kucharz zakłada, że piecyk spełnia założenia podane w jego instrukcji i skupia się przede wszystkim na myśleniu, jak za jego pomocą wyczarować coś pysznego.

Spojrzenie na komputery jest w jakimś stopniu podobne do tego opisywanego wyżej. To znaczy mamy konstruktorów (fizyków, inżynierów), którzy chcą zbudować odpowiednio szybką i sprawną maszynę. Z drugiej strony, mamy użytkowników (informatyków, matematyków), którzy chcieliby z tego urządzenia po prostu korzystać. Te dwa spojrzenia mogą być bardzo różne, dlatego potrzebujemy *czegoś*, co jest odpowiednikiem instrukcji obsługi piecyka. To znaczy czegoś na tyle konkretnego, żeby konstruktorzy wiedzieli, co mają stworzyć, a z drugiej strony – na tyle abstrakcyjnego, żeby użytkownicy mogli z komputera korzystać, wcale nie znając szczegółów jego budowy. Tym *czymś* jest właśnie tytułowy formalny model obliczeń.

Zanim przejdziemy do opisów konkretnych modeli obliczeń – mała dygresja. Otóż przykład przejścia od świata fizyków i inżynierów do świata informatyków to szczególny przykład ogólniejszego zjawiska – tak zwanego *abstrahowania*. Pojęcie to ma, pozwolę sobie tutaj na arbitralne stwierdzenie, fundamentalne znaczenie w całej informatyce. Więcej – często występuje szeregowo, jako zbiór tak zwanych *kolejnych warstw abstrakcji*. Jest to z pewnością temat na cały oddzielny szczegółowy artykuł. Tym razem podam tylko ogólnikowy przykład: sieć Internet składa się z warstw abstrakcji: fizycznej, łącz danych, sieciowej, transportowej, sesji, prezentacji i aplikacji. Zawsze polega to na tym, że projektując pewną warstwę, *zapominamy* o szczegółach warstwy niższej, pozostawiając w naszej głowie tylko ogólną instrukcję jej obsługi. I dalej: efektem naszej pracy ma być nie tylko jakiś bardziej złożony produkt, ale również uproszczona instrukcja obsługi do niego. Czyli jak w życiu: równie ważne (a może i ważniejsze) od tego, z kim warto się znać i z kim się spotykać, jest to, kogo nie warto znać i gdzie nie bywać.




Obwód obliczający, czy liczba $(b_4b_3b_2b_1)_2$ zapala zaznaczony segment na wyświetlaczu kalkulatora




Układ programowalny Altera Stratix IV GX FPGA realizujący model obliczeń oparty o obwody logiczne

Po tym (przyznaję, przydługim) wstępie czas już przejść do konkretów. Zacznijmy więc od modelu obliczeń klasycznego komputera.

Dla programisty komputer to coś, co:
- ma *pamięć*, do której potrafimy *wpisać* jakieś dane;
- potrafi uruchomić zaprojektowany (w specjalnym *języku*) przez użytkownika *program* i jego wynik zapisać do pamięci;
- pozwala na odczytanie zawartości pamięci.

Oczywiście, modele takie jak wyżej mogą się istotnie różnić, zależnie od tego, jaki charakter ma pamięć (zwykle ciąg bitów ustalonej długości) oraz – przede wszystkim – jaki język opisu programu dopuszczamy. Przykładowe modele w tym duchu to: model maszyny Turinga, model maszyny RAM (*random-access machine*), interpreter języka Java czy model oparty o obwody logiczne złożone z bramek. Skupmy się na chwilę na tym ostatnim.

Model obwodów logicznych moglibyśmy opisać, na przykład, tak (modele tego typu są bardzo często stosowane do opisu układów scalonych):
- pamięć stanowią dwa wektory: $v_{\text{in}} \in \{0,1\}^n$ oraz $v_{\text{out}} \in \{0,1\}^m$. Użytkownik potrafi dowolnie ustalić wartość wektora $v_{\text{in}}$ (czyli, żargonowo, „mamy $n$ bitów wejściowych”).
- Użytkownik może opisać dowolną sieć co najwyżej $T$ bramek OR, AND oraz NOT łączących wektor $v_{\text{in}}$ z wektorem $v_{\text{out}}$. Wówczas komputer jest w stanie przypisać do wektora $v_{\text{out}}$ wynik obliczeń opisanej sieci na wektorze $v_{\text{in}}$.
- Użytkownik po skończonym obliczeniu potrafi poznać wartość wektora $v_{\text{out}}$.

Przykład *programu* w tym modelu pokazuje rysunek obok.

Poziom abstrakcji w tym przykładzie jest chyba dość jasny. Inżynier projektujący tak zdefiniowany komputer (Czytelnik Lubiący Konkrety może zapoznać się z technologią FPGA) ma na celu zastanowienie się, w jaki sposób stworzyć urządzenie, które:
- zawiera i potrafi (na żądanie użytkownika) ustawić dowolnie $T$ bramek logicznych;
- potrafi przyjąć podane przez użytkownika dane wejściowe (opis wektora $v_{\text{in}}$);
- potrafi dokonać obliczenia i *zwrócić* użytkownikowi wynik obliczeń, czyli $v_{\text{out}}$.

Powyższe zadanie zapewne jest bardzo trudne, wymaga znajomości fachowej wiedzy z elektroniki, wiedzy o działaniu tranzystorów itp., itd.

Z drugiej strony: informatyk, który z takiego urządzenia chce korzystać, może zupełnie nie znać fizyki i już myśleć tylko o algorytmice, czyli – w tym przypadku

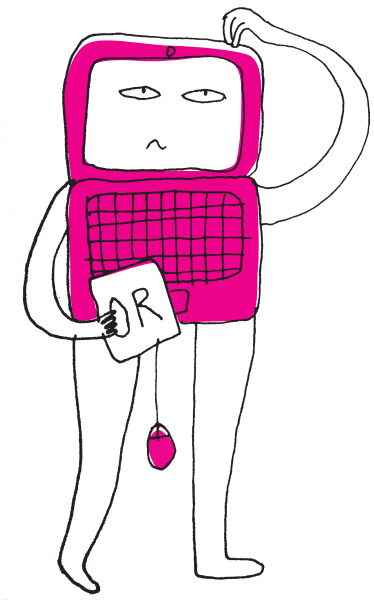
– nauce o takim przestawianiu klocków (bramek), żeby obliczało się dokładnie to, co chcemy i to możliwie szybko (a więc przy użyciu możliwie małej liczby bramek).

Przejdźmy teraz do tego, co stanowi esencję całego tego numeru *Delty*, czyli do komputerów kwantowych. Za chwilę przedstawimy formalny model obliczeń dla komputera kwantowego (czyli jego „instrukcję obsługi"). To, jakie problemy natury fizycznej napotykają projektanci takich potencjalnych komputerów, opisuje Rafał Demkowicz-Dobrzański (str. 6–9). Z drugiej strony – jakie cuda potrafiłby zdziałać informatyk mający dostęp do takiego (hipotetycznego) urządzenia opisuje Wojciech Czerwiński na stronach 10–12, przybliżając szczegóły algorytmu Shora na rozkład dużych liczb na czynniki pierwsze. Prezentowany model korzysta z języka abstrakcyjnej algebry liniowej. Podstawy tej dziedziny prezentuje Maciej Zdanowicz na stronie 5 (oraz Marek Kordos na stronie 4), przy okazji dowodząc, że programiści za kilkadziesiąt lat będą musieli znać chyba trochę więcej matematyki niż ci obecni. Na ile blisko (a na ile daleko) od stworzenia Prawdziwego Komputera Kwantowego jesteśmy w tej chwili pisze Piotr Zalewski na stronie 24. Dodatkowo w artykułach na stronach 12–15 i 16–17, kwartet Gardas, Jałowiecki, Dajka, Mierzejewski oraz (solo) Łukasz Rajkowski próbują przybliżyć Czytelnikowi, co już dziś jest komercyjnie dostępne. Po pierwsze omawiamy komputer D-Wave, który jednak realizuje (na 2048 kubitach) inny niż tu opisany model obliczeń kwantowych. Po drugie: omawiamy praktyczny protokół BB84, zakładający istnienie oraz możliwość szybkiego i taniego tworzenia „komputerów jednokubitowych" na żądanie. Gorąco zachęcam do lektury tych i innych artykułów z tego numeru i bezzwłocznie przystępuję do prezentacji modelu obliczeń kwantowych.

Istnieje więcej niż jeden model obliczeń kwantowych. Ten opisywany tutaj uznaje się za standardowy. Inne modele to np.: kwantowe automaty komórkowe, jednokierunkowe komputery kwantowe, topologiczne komputery kwantowe czy adiabatyczne komputery kwantowe. W tym numerze *Delty* odnosimy się niemal wyłącznie do modelu standardowego, z wyjątkiem artykułu ze stron 12–15, który skupia się na modelu adiabatycznym.

- Do opisu stanu pamięci komputera kwantowego potrzebne są liczby zespolone, których zbiór oznaczamy symbolem $\mathbb{C}$ (więcej o nich pisze Marek Kordos na stronie 4). Zazwyczaj operować będziemy ciągami $m$ liczb zespolonych (tak zwanymi zespolonymi wektorami wymiaru $m$), których zbiór oznaczamy $\mathbb{C}^m$. (Przykładowo: $\phi = (2+i, 6, 10-3i, i) \in \mathbb{C}^4$.) Dla wektorów określamy ich długość jako pierwiastek z sumy kwadratów modułów jego kolejnych współrzędnych. W naszym przykładzie długość $\phi$ wynosi więc:
$$\sqrt{|2+i|^2 + |6|^2 + |10-3i|^2 + |i|^2} = \sqrt{5+36+109+1} = \sqrt{151}.$$
Opisem stanu pamięci komputera kwantowego jest jeden wektor o długości 1 z przestrzeni $\mathbb{C}^{2^n}$, np. $\psi = (0, 0, 0, \frac{1}{2}, 0, \frac{i\sqrt{2}}{2}, 0, \frac{-i}{2}) \in \mathbb{C}^{2^3}$. W świecie kwantowym często zapisujemy to samo w nieco innym (równoważnym) języku, mianowicie:
$$\psi = \frac{1}{2}|011\rangle + \frac{i\sqrt{2}}{2}|101\rangle + \frac{-i}{2}|111\rangle,$$
gdzie (jak łatwo się domyślić) $|\mathrm{bin}(k)\rangle$ oznacza wektor złożony z $(2^n - 1)$ zer i jedynki na $(k+1)$-szej współrzędnej, np. $|011\rangle = (0,0,0,1,0,0,0,0)$. Wektory $|\mathrm{bin}(k)\rangle$ nazywamy wektorami bazy standardowej. Stany pamięci, które nie są takimi wektorami, a więc są sumą co najmniej dwóch różnych wektorów bazowych, fizycy lubią nazywać *superpozycją*.
Jeśli stan pamięci jest opisany wektorem z $\mathbb{C}^{2^n}$, to mówimy, że nasz komputer operuje $n$ kubitami. Warto zwrócić uwagę, że stan pamięci klasycznego komputera operującego $n$ bitami opisujemy po prostu jednym ciągiem zer i jedynek długości $n$ (np. $b_1 b_2 \dots b_n$), co możemy interpretować jako ustalenie *jednego* wektora bazy standardowej $|b_1 b_2 \dots b_n\rangle \in \mathbb{C}^{2^n}$. Przewaga komputera kwantowego polega zaś na tym, że w pamięci możemy trzymać bardziej wyrafinowane obiekty, a więc kombinacje liniowe takich wektorów (superpozycje), jak chociażby opisany wyżej wektor $\psi$, będący przykładem stanu pamięci komputera trójkubitowego.
- Stan początkowy komputera użytkownik może ustalić zupełnie dowolnie, przy czym może używać iloczynu tensorowego do opisu (ta uwaga jest o tyle istotna, że cała przestrzeń ma wymiar wykładniczy, więc sam opis może czasem być ogromny; iloczyn tensorowy jest tu więc potencjalnym ułatwieniem). Iloczyn tensorowy $\otimes$ wektorów to bardzo prosta operacja, o której więcej piszemy na stronie 5. Na razie wystarczy nam tylko własność $|b_1 \dots b_n\rangle \otimes |b'_1 \dots b'_m\rangle = |b_1 \dots b_n b'_1 \dots b'_m\rangle$, by zrozumieć, że ten sam wektor można opisać długo bądź zwięźle:
$$\alpha = \frac{1}{4}(|0000\rangle + |0001\rangle + |0010\rangle + |0011\rangle - |0100\rangle - |0101\rangle - |0110\rangle - |0111\rangle$$
$$+|1000\rangle + |1001\rangle + |1010\rangle + |1011\rangle - |1100\rangle - |1101\rangle - |1110\rangle - |1111\rangle)$$
lub
$$\alpha = \frac{1}{4}(|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle).$$

bin($k$) oznacza binarny zapis liczby $k$.



$$\mathrm{H} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

$$\mathrm{T} = \begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}$$

$$\mathrm{CNOT} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$$

Tak naprawdę opisem obliczenia, na pewnym poziomie abstrakcji, może być dowolna macierz unitarna $M$ z przestrzeni $\mathbb{C}^{2^n\times 2^n}$. W wyniku obliczenia stan pamięci $\phi$ zmienia się na $M\phi$. Użytkownik może podać dowolną macierz, co więcej, ma prawo używać iloczynu tensorowego do opisu (bezpośredni opis miałby rozmiar wykładniczy). Okazuje się, że (twierdzenie o uniwersalności) każdą macierz unitarną da się (z dowolnym przybliżeniem) uzyskać (korzystając z mnożenia i iloczynu tensorowego), mając do dyspozycji wyłącznie macierze H, T, CNOT i identyczność. Oczywiście, aby takie obliczenie było efektywne, ilość użytych macierzy podstawowych nie może być ogromna.

- Pojedyncze obliczenie na komputerze kwantowym odpowiada przemnożeniu stanu pamięci przez podaną przez użytkownika (nie byle jaką) *macierz*. W tym miejscu Czytelnik Algebraicznie Kulejący bardzo proszony jest o nieprzerażanie się tym pojęciem. Okazuje się, że nawet nie musimy dokładnie wiedzieć, jak się mnoży dowolną macierz przez wektor, bo wystarczą nam tylko trzy przykłady (dla macierzy H, T i CNOT z marginesu oraz macierz identyczności I):

$$\mathrm{H}(a|0\rangle + b|1\rangle) = \frac{1}{\sqrt{2}}\left((a+b)|0\rangle + (a-b)|1\rangle\right),$$
$$\mathrm{T}(a|0\rangle + b|1\rangle) = a|0\rangle + be^{i\pi/4}|1\rangle,$$
$$\mathrm{CNOT}(a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle) = a|00\rangle + b|01\rangle + d|10\rangle + c|11\rangle.$$

  Jak widzimy macierze H i T dotyczą tylko wektorów jednokubitowych, a macierz CNOT – dwukubitowych. Aby opisać operację w wyższych wymiarach znów wolno nam się posłużyć iloczynem tensorowym, tym razem zastosowanym do macierzy. Ponownie jest to dość naturalna operacja (opisana szerzej później), a nam na razie wystarczy tylko własność $(M_1 \otimes M_2)(\phi_1 \otimes \phi_2) = M_1(\phi_1) \otimes M_2(\phi_2)$, która jest prawdziwa, gdy *wymiary się zgadzają*. Intuicyjnie oznacza ona, że rozpatrywane macierze można przykładać *lokalnie* do dowolnie wybranych współrzędnych wielowymiarowego wektora stanu pamięci. Podajmy przykład, który powinien rozjaśnić tę operację:

$$\begin{aligned}
&(\mathrm{H}\otimes\mathrm{H}\otimes\mathrm{CNOT}\otimes\mathrm{I})\left(\frac{1}{\sqrt{2}}(|01101\rangle + |10011\rangle)\right) =\\
&\quad= \frac{1}{\sqrt{2}}\big((\mathrm{H}\otimes\mathrm{H}\otimes\mathrm{CNOT}\otimes\mathrm{I})(|0\rangle\otimes|1\rangle\otimes|10\rangle\otimes|1\rangle) +\\
&\qquad + (\mathrm{H}\otimes\mathrm{H}\otimes\mathrm{CNOT}\otimes\mathrm{I})(|1\rangle\otimes|0\rangle\otimes|01\rangle\otimes|1\rangle)\big) =\\
&\quad= \frac{1}{\sqrt{2}}\big(\mathrm{H}(|0\rangle)\otimes\mathrm{H}(|1\rangle)\otimes\mathrm{CNOT}(|10\rangle\otimes\mathrm{I}(|1\rangle)) +\\
&\qquad + \mathrm{H}(|1\rangle)\otimes\mathrm{H}(|0\rangle)\otimes\mathrm{CNOT}(|01\rangle\otimes\mathrm{I}(|1\rangle))\big) =\\
&\quad= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\otimes|11\rangle\otimes|1\rangle +\right.\\
&\qquad \left. + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\otimes|01\rangle\otimes|1\rangle\right) =\\
&\quad= \frac{1}{2\sqrt{2}}\big(|00111\rangle - |01111\rangle + |10111\rangle - |11111\rangle +\\
&\qquad + |00011\rangle + |01011\rangle - |10011\rangle - |11011\rangle\big).
\end{aligned}$$

  Użytkownik może wybrać do opisu obliczenia dowolnie wybrany iloczyn tensorowy opisanych wyżej macierzy.
- Odczyt z pamięci jest w tym modelu bardzo nietrywialny. Przede wszystkim pomiar (zwykle) nie jest deterministyczny i może zwrócić różne wyniki dla tego samego stanu pamięci. Spróbujemy zaprezentować tutaj pewien uproszczony (ale wystarczający, by śledzić chociażby artykuł ze stron 10–12 o faktoryzacji Shora) opis odczytu z komputera kwantowego. Użytkownik, chcąc dokonać pomiaru pamięci w pewnym $n$-kubitowym komputerze, musi podać pewien podzbiór indeksów $(i_1, \ldots, i_k) \subset (1, \ldots, n)$. Jeśli teraz stan komputera to po prostu $|b_1 \ldots b_n\rangle$ (czyli pewien wektor bazowy), to uzyskamy wynik $(b_{i_1}, \ldots, b_{i_k})$. Jeśli natomiast stan komputera jest superpozycją $\sum_{v\in\{0,1\}^n} \alpha_v|v\rangle$, gdzie $\alpha_v \in \mathbb{C}$ są współczynnikami przy wektorach bazowych $|v\rangle$, to pomiar jest istotnie niedeterministyczny. Uzyskamy wynik $u = (b_{i_1}, \ldots, b_{i_k})$ z prawdopodobieństwem $P_u = \sum_{v\in S_u} |\alpha_v|^2$, gdzie $S_u$ to zbiór tych wektorów $v \in \{0,1\}$, które na współrzędnych $i_1, \ldots, i_k$ mają dokładnie wartości $b_{i_1}, \ldots, b_{i_k}$.
  W komputerze kwantowym stan pamięci po pomiarze *zmienia się* (w świecie kwantowym pomiar musi zmienić stan pamięci!) na superpozycję tych składowych starego stanu, które są *zgodne* z pomiarem. Oczywiście niektóre stare składowe nie są zgodne z pomiarem, w związku z tym te, które są zgodne, muszą mieć zmienione współczynniki, aby długość całego wektora pamięci pozostała równa 1. Konkretnie rzecz biorąc, nowy stan pamięci po pomiarze to:

$$\frac{1}{\sqrt{P_u}}\sum_{v\in S_u}\alpha_v|v\rangle.$$

  Zauważmy, że po pomiarze współrzędne $i_1, \ldots, i_k$ nie będą już nigdy istotne, bo są i tak zawsze takie same dla każdej składowej.
- Użytkownik może wykonać dowolną sekwencję wielu obliczeń i odczytów (może je przeplatać).

W ogólności, przy odczycie z pamięci $\phi$, użytkownik może wybrać dowolną macierz hermitowską, która – jak wiadomo z algebry liniowej – rozkłada się na sumę $\sum_i h_i P_i$ przeskalowanych rzutów na swoje podprzestrzenie własne (macierze $P_i$). Wynikiem pomiaru jest pewien indeks $i$ (nic więcej nie poznajemy!), który otrzymujemy z prawdopodobieństwem $\phi^T P_i \phi$ oraz stan pamięci po pomiarze zmienia się na

$$\frac{P_i\phi}{\sqrt{\phi^T P_i \phi}}$$