dr hab. inż. Ernest Jamro, prof. AGH
Instytut Elektroniki
Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie
al. Mickiewicza 30 30-059 Kraków

Kraków 8-05-2021

Recenzja rozprawy doktorskiej mgr. inż. Michał Andrzejczak
dla Rady Dyscypliny Naukowej Informatyka Techniczna i Telekomunikacja
Wojskowej Akademii Technicznej im. Jarosława Dąbrowskiego
(podstawą opracowania recenzji jest uchwała Rady Dyscypliny z dnia 9-03-2021)

Tytuł rozprawy: Acceleration of Lattice Based Algorithms

Autor Rozprawy: mgr. inż. Michał Andrzejczak

Promotor: dr hab. inż. Andrzej Paszkiewicz

Dziedzina: nauki techniczne
Dyscyplina: informatyka (aktualnie informatyka techniczna i telekomunikacja)

1. Zagadnienie naukowe i aktualność tematu

Tematyka rozprawy doktorskiej skupia się na sprzętowej akceleracji w układach programowalnych FPGA przykładowych algorytmów kryptoanalizy (część I rozprawy) oraz kryptografii (część II rozprawy) bazujących na teorii kratach.
Doktorant dokonał poprawnego wyboru tematyki pracy doktorskiej. Dotyczy to zarówno wyboru algorytmów kryptograficznych jak i sprzętowej implementacji z użyciem układów programowalnych FPGA. Wiele współcześnie wykorzystywanych algorytmów kryptograficznych nie jest odpornych na ataki za pomocą dostatecznie wydajnego komputera kwantowego. Wielu kryptografów tworzy nowe algorytmy, które będą wykorzystywane w czasach, kiedy komputery kwantowe staną się zagrożeniem dla bezpieczeństwa danych. Prace nad algorytmami postkwantowymi nabierają coraz większego tempa a ich znaczenie dla bezpieczeństwa jest trudne do przecenienia. Warto podkreślić, że jeśli na chwilę obecną nie potrafimy złamać zaszyfrowanych informacji, to potrafimy je rejestrować i w przyszłości powtórnie do nich wrócić.
Podobnie niezwykle aktualną tematyką jest sprzętowa akceleracja algorytmów w szczególności z użyciem układów FPGA. Powoli zauważalne staje się, że wykładniczy wzrost liczby tranzystorów w układach scalonych określony prawem Moore'a przestaje obowiązywać. Ponadto wzrost wydajności procesorów CPU poprzez użycie większej liczby dostępnych tranzystorów nie jest liniowy (reguła Pollacka). Dlatego dalszym etapem zwiększenia dostępnej mocy obliczeniowej staje się budowanie układów obliczeniowych zorientowanych na konkretną aplikację, najczęściej z użyciem układów programowalnych FPGA lub dedykowanych układów scalonych ASIC.
Poprawność przyjętej tematyki widoczna jest również we wnioskach płynących z pracy. Doktorantowi udało się osiągnąć znaczącą akcelerację zaimplementowanych algorytmów w stosunku do referencyjnych rozwiązań opartych na procesorach ogólnego przeznaczenia (CPU).

2. Zawartość i ocena merytoryczna

Praca składa się z 11 rozdziałów rozbitych na dwie części. Pierwszej części poświęcone są rozdziały od 2 do 6 i opisuje ona nowatorską architekturę sprzętowej implementacji algorytmów przesiewania krat. Część druga jest poświęcona sprzętowej akceleracji algorytmu Round 5, jednemu ze zgłoszeń na nowy postkwantowy algorytm szyfrowania oraz mechanizm enkapsulacji klucza.

Rozdział 1 stanowi wstęp do teorii krat. Rozdział jest raczej krótki, ma tylko cztery strony, i jak na rozdział wprowadzający do dość nowego i specjalistycznego zagadnienia relatywnie krótki. Niemniej na samym początku rozdziału podana jest literatura, z której korzystał doktorant.

Część I, pracy opisuje teorię oraz algorytmy oraz zaproponowaną implementację przesiewania krat. **Rozdział 2.** Jest jednostronicowym streszczeniem tego co zostało zamieszczone w części I pracy.

Rozdział 3 pt. "Introduction to lattice sieving" dokonuje przeglądu algorytmów służących do przesiewania krat. Zadaniem tych algorytmów jest odpowiedni dobór dwóch (trzech) wektorów, które następnie są redukowane względem siebie zgodnie z algorytmem Gaussa (Gauss-reduced). Dwa wektory są zredukowane algorytmem Gaussa poprzez takie odejmowanie (dodawanie) wektorów, że otrzymujemy wektor krótszy.
Rozdział ten poświęcony jest głównie przeglądowi literatury, która opisuje sposoby odpowiednio dobierania kandydatów do otrzymania krótszego wektora. Jednym z najpopularniejszych algorytmów jest algorytm przesiewania o nazwie GaussSieve. Podrozdział 3.2 pokazuje równoległą implementację algorytmu GuassSieve zaprezentowaną w literaturze.

W rozdziale 3.3 zaprezentowano 4 pytania (tezy) pracy:
Pytanie 1: Czy redukcja kraty może być akcelerowana poprzez sprzętową implementację.
Pytanie 2: Czy akceleracja może być użyta w praktyce.
Pytanie 3: Jaki jest możliwy do osiągnięcia współczynnik akceleracji w przypadku użycia nowoczesnych układów FPGA.
Pytanie 4: Jak sprzętowa akceleracja wpływa na rozważany problem [znajdowania najkrótszego wektora].
Według opinii recenzenta pytania są trochę naiwne, jedno wynika z drugiego, pytania powinny być raczej połączone. Po drugie pytania są tak ogólnikowe, że niewiele przekazują treści same w sobie. Dużo więcej mówiący jest opis pod każdym pytaniem. Według recenzenta pytania powinny być dłuższe. Opis pytania 1 skupia się raczej na zasobach sprzętowych (obliczeniowych) znajdujących się w układach FPGA. Pytanie 2 skupia się na transmisji danych do / z układu FPGA. Pytanie 3 skupia się, według recenzenta (bo jest na tyle ogólne), na zajętych zasobach sprzętowych układu FPGA i w związku z tym na możliwym stopniu zrównoleglenia algorytmu. Pytanie 4 rozważa zagadnienie: jak akceleracja wpływa na rozmiar kraty, która może być w sensownym czasie zredukowana. Odpowiedź na to pytanie jest względnie oczywista, ponieważ akceleracja jest liniowa a złożoność rozwiązywanego problemu wykładnicza.

Rozdział 4 opisuje sprzętową architekturę modułu redukcji wektorów. Jedną z najważniejszych elementów tego rozdziału, ale również całej pracy doktorskiej jest zaproponowany algorytm do akceleracji sprzętowej i jego sprzętowa realizacja. Jest to algorytm 3, *Reduce*(v, u). Propozycja ta i jej implementacja jest jednym z głównych oryginalnych osiągnięć Doktoranta. Ze swojego doświadczenia wiem, że znalezienie odpowiedniego algorytmu do sprzętowej akceleracji nie jest rzeczą trywialną i wymaga wielomiesięcznych studiów literatury, analizy problemu, prowadzonych badań. Z drugiej strony w literaturze znajdują się wskazówki dotyczące tego

zagadnienia np. [51]. Głównym składnikiem algorytmu 3 jest iloczyn skalarny dwóch wektorów. Zaproponowana sprzętowa implementacja tego iloczynu jest wykonywany masowo równolegle, czyli każdemu mnożeniu przyporządkowany jest sprzętowy układ mnożący. W układach FPGA liczba dostępnych mnożarek jest liczona w tysiącach, dlatego układy FPGA umożliwiają przyśpieszenie obliczeń w stosunku do procesorów CPU a nawet GPGPU (general-purpose computing on graphics processing units). Warto podkreślić, że układy FPGA są bardzo wydajne przy wykonywaniu operacji arytmetycznych całkowitoliczbowych, a z takimi tutaj mamy do czynienia. Co więcej jednostka wektorowa AVX procesorów CPU x86 jest bardziej nastawiona na operacje zmiennoprzecinkowe, co powoduje, że pomimo tego, że operacje zmiennoprzecinkowe są bardziej skomplikowane (zajmują więcej powierzchni krzemu) od operacji całkowitoliczbowych, moc obliczeniowa procesorów CPU jest często mniejsza dla operacji całkowitoliczbowych.
Z punktu widzenia projektanta układów FPGA najbardziej skomplikowane jest nie samo mnożenie ale sumowanie wyników mnożeń, podobnie jest zresztą przy programowaniu procesorów CPU. W tym celu potrzebna jest odpowiednia konstrukcja drzewa dodającego. Pewnym mankamentem pracy jest użycie binarnego drzewa sumatorów i brak dyskusji nad możliwością użycia układów dodających z trzema wejściami. W układach FPGA firmy Xilinx możliwa jest realizacja drzewa dodającego, w którym następuje sumowanie trzech wejść, ang. ternary adder. Układ taki zajmuje tyle samo zasobów co sumator dwuwejściowy lecz jest nieznacznie wolniejszy.
Implementacja drzewa dodającego w układach FPGA wydaje się trywialna, ale wydajne sprzętowe (czy też programowe) zaprojektowanie takiego drzewa dla różnej liczby wejść n, wymaga wielu dni projektowania i testowania. Należy dodać, że w projekcie należy dodatkowo uwzględnić architekturę potokową, w takim wypadku konieczne jest dopasowanie czasów propagacji układów mnożących i liczby poziomów drzewa dodającego przypadającej na pojedynczy takt zegara. Wymagało to wielu różnych implementacji i porównywania czasów opóźnień. *Dlatego oceniając niniejszą pracę doktorską należy zwrócić uwagę, że projekt został zaimplementowany w układach FPGA, w konsekwencji nawet trywialne rozwiązanie jeśli chodzi o algorytm obliczeń staje się skomplikowane jeśli rozważa się jego sprzętową implementację.* W pracy bardzo mało uwagi (jeśli w ogóle) poświęcono części projektowej, nie opisano trudności i czasochłonność wynikające ze sprzętowej implementacji. Jest to typowe dla publikacji naukowych, niemniej moim zdaniem Doktorant powinien w rozprawie zawrzeć chociaż jeden akapit przybliżający skalę problemu z jaką się zetknął. Niniejsza uwaga dotyczy właściwie całej pracy.
Ciekawym i oryginalnym zagadnieniem poruszonym w tym rozdziale jest wykonywana operacja dzielenia. Standardowo operacja dzielenia jest wolna i wymaga wielu taktów zegara. Doktorant zaproponował własną architekturę układu dzielącego opartą na analizie wyników dzielenia, który zawiera się w zakresie od 0 do 4.

Rozdział 5. Analiza teoretycznej wydajności. W pierwszej części rozdziału Doktorant skupia się na istniejącym rozwiązaniu programowym w szczególności na profilowaniu kodu, dzięki któremu możliwe stało się znalezieniu fragmentów kodu, które zajmują najwięcej czasu procesora. Zadane to nie jest łatwe ponieważ np. rozwiązanie programowe *gök* projektowane jest w różnych językach: C/C++, python, Cython. Jest to coraz częstsza technika programowania, gdzie kod początkowy projektowany jest w języku wyższego poziomu, np. python, a następnie na podstawie profilowania kodu, część kodu, która zabiera najwięcej czasu procesora przenoszona jest do języka niższego poziomu o wyższej wydajności obliczeniowej, zwykle języka C/C++. Warto podkreślić, że Doktorant w swojej pracy idzie o krok dalej, czyli fragment kodu, który zajmuje najwięcej czasu procesora w języku C/C++ przenosi do platformy sprzętowej – układów programowalnych FPGA. Wiąże się to z dodatkowym nakładem pracy

ponieważ programowanie układów FPGA odbywa się na niższym poziomie – poziomie języków opisu sprzętu (recenzent domyśla się, że z użyciem języka VHDL lub Verilog). W konsekwencji powstaje piramida składająca się z trzech różnych poziomów / języków programowania.

W tym miejscu warto podkreślić, że Doktorant porównuje się do istniejących rozwiązań programowych. Nie wspomniana jak skutecznie dokonano optymalizacji / akceleracji kodu po stronie programowej. Brak odpowiedniej akceleracji jest możliwy i zawsze warto dokonać analizy dostępnej szczytowej mocy obliczeniowej. Przykładowo dla operacji zmiennoprzecinkowych podwójnej precyzji i jednostki wektorowej AVX (emulacja operacji całkowitoliczbowych do szerokości mantysy 54-bitów) wydajność szczytowa wynosi 32 FLOPS/cykl zegara (https://en.wikichip.org/wiki/flops). W konsekwencji mnożenie skalarne wektorów dla n = 128 zajmuje (2*n) / 32= 8 taktów zegara. Algorytm 3, w pierwszym przybliżeniu, wymaga dwóch mnożeń skalarnych wektorów, co skutkuje czasem rzędu 16 taktów zegara. Przy częstotliwości zegara 4.1 GHz i powyższych założeniach jesteśmy w stanie wykonać $256 \cdot 10^6$ wywołań funkcji *Reduce* na sekundę. Fig. 5.2 pokazuje, że osiągnięto około $12 \cdot 10^6$ wywołań na sekundę.
Referencyjny kod był pisany w języku python i C++. Nawet użycie języka C/C++ nie gwarantuje uzyskaniem dobrego wyniku, na przykład kod mnożenia macierzy wykonywany przy użyciu dedykowanej biblioteki BLAS może być nawet o rząd wielkości szybszy niż ten sam kod napisany w C/C++ bez poprawnej optymalizacji. Należy podkreślić, że ze względu na obszerność rozprawy Doktorskiej, porównanie się z istniejącym rozwiązaniem jest uzasadnione, niemniej sformułowanie pytania 3, mówiące o granicach akceleracji sprzętowej jest niezmiernie trudne do udowodnienia, jeśli nie określi się dokładnie warunków porównania.
W rozdziale 5 określono trzy różne scenariusze sprzętowej akceleracji obliczeń. Scenariusz I zakłada wykonywanie akceleracji tylko dla pojedynczej funkcji *Reduce* (algorytm 3). Głównym czynnikiem wpływającym na szybkość akceleracji dla tej architektury jest transfer danych pomiędzy procesorem CPU a modułem sprzętowym. W konsekwencji taki moduł nie ma praktycznego znaczenia i został tutaj pokazany głównie w celu łatwiejszego zrozumienia następnych scenariuszy. Scenariusz II zakłada, że $2 \cdot j$ wektorów wejściowych jest wykorzystywane wielokrotnie dla j^2 wywołań funkcji *Reduce*. W związku z tym liczba potrzebnych transmisji jest dużo mniejsza i kluczowa staje się szybkość prowadzonych obliczeń, już dla $j \geq 6$. Jednak ze względu na dalsze ulepszenia ten przypadek użycia nie jest zastosowany w praktyce. W całej pracy, nie wspomniano o tym, że układy FPGA najczęściej mają dostęp do zewnętrznych pamięci DDR SDRAM, których pojemności są rzędu 16 GB. Szkoda że tego nie uwzględniono na Fig. 5.3. Dodatkowo niektóre układy FPGA mają dostęp do szybkiej pamięci HBM, pamięci SDRAM osadzonej w tej samej obudowie co układ FPGA. Szybkość transmisji danych do / z takich pamięci jest znacząca. W takim wypadku transmisji pomiędzy układami FPGA a CPU mógłby? podlegać tylko i wyłącznie: indeks wektorów oraz współczynników q oraz ewentualnie długości skróconego wektora.
W rozdziale 5.5.1 wprowadzono istotną modyfikację zaproponowanego algorytmu. Do tej pory potokowa architektura układu nie była w pełni wykorzystywana, następna dana wejściowa była wprowadzana po zakończeniu obliczeń dla poprzedniej danej. Dlatego krytycznym elementem algorytmu była latencja układu potokowego i przetwarzanie pojedynczego wywołania funkcji *Reduce* trwało około 7 taktów zegara. W tym podrozdziale zaproponowano przetwarzanie danych co takt zegara, co wiąże się z lekką modyfikacją algorytmu obliczeń. Według recenzenta, Doktorant w tym rozdziale myląco używa pojęcia przetwarzania równoległego mając na myśli przetwarzanie potokowe (a właściwie pełne jego użycie). Użycie rozwiązania potokowego jest oczywiste. Według recenzenta wprowadzenie Scenariusza I wraz ze skomplikowaną analizą przypadku tylko zaciemnia, a nie wyjaśnia zagadnienie, tym bardziej, że Scenariuszowi I

poświęca się tak dużo miejsca, a prosta i oczywista modyfikacja właściwie przechodzi niezauważalnie.
Skupienie się na Scenariuszu I powoduje również brak dyskusji o częstotliwości zegara. Dla architektury potokowej, dla której tylko pojedyncza dana jest przetwarzana w potoku (Scenariusz I), zwiększenie częstotliwości zegara kosztem liczby etapów potokowych (latencji) nie przynosi dobrych rezultatów (iloczyn okresu zegara i latencji z reguły ulega zwiększeniu). Natomiast w przypadku pełnego wykorzystania architektury potokowej, okres zegara jest kluczowy i bardzo często można uzyskać dużo lepszą szybkość obliczeń wprowadzając dodatkowe etapy potokowe.
Scenariusz III zakłada, że transmisja danych oraz obliczenia są prowadzone równocześnie na różnych etapach architektury potokowej. Niemniej największą modyfikacją tego scenariusza jest wielokrotne wykorzystanie danych, tak aby transmisja danych pomiędzy procesorem CPU a FPGA miała znikome znaczenie przy dokonywanych obliczeniach. Redukcji podlegają dwa zbiory wektorów: zbiór S (k elementów z tego zbioru) oraz lista L.
Scenariusz III został podzielony na trzy etapy. W pierwszym etapie redukowane są elementy zbioru S i L każdy z każdym (co wymaga $k \cdot |L|$ redukcji). W następnym etapie redukowane są elementy zbioru S pomiędzy sobą, co wymaga $(k\text{-}1) + (k\text{-}2) \ldots + 1 = (k \cdot (k\text{-}1))/2$ redukcji. W trzecim etapie redukowane są elementy listy L przez k elementów zbioru S (co wymaga $k \cdot |L|$ redukcji). W rozważaniach teoretycznych w tym podrozdziale w równaniach (5.21), (5.24) i (5.25) użyto tego samego symbolu $f_{el}(n, w)$ mimo tego, że jego znaczenie się zmienia, jest to trochę mylące. Szkoda, że w drugim etapie obliczeń i (5.22) ciągle występuje latencja modułów obliczeniowych, architektura potokowa nie jest w pełni wykorzystana. Moim zdaniem, rozwiązaniem tego problemu mogłoby być zwiększenie parametru k, tak aby $f_{cl}(n) < 2k$.
Następną modyfikacją scenariusza III jest wprowadzenie zrównoleglenia zaproponowanych wcześniej modułów obliczeniowych. Stało się to możliwe ponieważ transmisja danych wymaga mniej czasu niż obliczenia. Doktorant pokazał we wzorze (5.29) stosunek czasu (liczby cykli zegara) dla transmisji $C_T(n)$ oraz czasu obliczeń $C_C(n)$. Na uwagę zasługuje zmodyfikowany Algorytm 4 względem [51]. Szkoda, że powyższa modyfikacja nie została dogłębnie omówiona w treści rozprawy.

Sugestia 1 (optymalizacji kodu na CPU): Dla funkcji *Reduce* najwięcej obliczeń wymaga otrzymanie wyniku iloczynu skalarnego dwóch wektorów. Doktorant w takcie zaproponowanej optymalizacji / zrównoleglenia algorytmu *Reduce* pogrupował mnożenia wektorów w taki sposób, że przypominają one mnożenie dwóch macierzy: etap pierwszy i trzeci scenariusza III. Jak częsta jest redukcja dwóch wektorów? Czy możliwe jest zastosowanie mnożenia macierz-macierz zamiast wielu mnożeń wektor-wektor i wielokrotnego wywoływania funkcji *Reduce*? Zastosowanie mnożenia macierzy zmniejsza liczbę potrzebnych transmisji danych i być może spowoduje lepsze wykorzystanie jednostki wektorowej AVX. Dodatkowo jeśli równocześnie mamy dostęp do wielu parametrów q (algorytm 3), możliwy jest taki wybór redukcji aby otrzymane wektory były najkrótsze (wybieramy q najbliższe liczbie całkowitej). Warto podkreślić, że istnieją wydajne metody mnożenia macierzy w procesorach CPU jak i procesorach GPGPU, np. biblioteka BLAS, cuBLAS. Możliwe jest też wykonywanie obliczeń na liczbach całkowitoliczbowych, zmiennoprzecinkowych podwójnej precyzji i według mnie pojedynczej lub połówkowej precyzji z ewentualnym sprawdzeniem czy błędy zaokrągleń nie zaburzają wyniku – następuje w rzeczywistości skrócenie długości wektora. Warto podkreślić, że w dzisiejszych systemach komputerowych coraz bardziej krytyczna staje się transmisja danych kosztem nawet większej liczby przeprowadzonych obliczeń. Zastosowanie mnożenia macierzy w układach FPGA umożliwiłoby również zmniejszenie użytych układów mnożących (algorytm Coppersmith–Winograd).

Rozdział 6 podaje wyniki implementacji algorytmów zaproponowanych w Rozdziale 5. Uzyskanie jednoznacznego rezultatu jest jednak bardzo trudne ponieważ, mamy do czynienia z wieloma ograniczeniami, przede wszystkim dostępne zasoby mnożące (DSP), logiczne ALM, pamięciowe, liczba etapów potokowych, szybkość transmisji danych pomiędzy układem FPGA a procesorem CPU, częstotliwość zegara, wielkość zbioru L, wybór układu FPGA i jego koszt. Bardzo ciekawe porównanie podane w pracy to analiza kosztów obliczeń w chmurze. Ponieważ zarówno procesory CPU jak i układy FPGA można używać odpłatnie w chmurze np. Amazona, możliwe jest porównanie kosztów tych dwóch platform obliczeniowych, jest to bardzo ciekawe rozwiązanie ponieważ porównanie procesora CPU z FPGA nie jest łatwe: liczy się koszt zakupu, koszt eksploatacji, również koszt zużytej energii, o którym w tej pracy niestety się nie wspomina. Podrozdział 6.6 rozważa różne aspekty projektowe w celu dokonania akceleracji obliczeń. Rozważane jest zmniejszenie szerokości układów mnożących, tak aby zwiększyć częstotliwość zegara. Odbywałoby się to jednak kosztem zmniejszenia przepustowości układu. Według mnie Doktorant w trakcie implementacji projektu dokonał mylnego szacowania maksymalnej częstotliwości pracy układów mnożących. Sprzętowe moduły mnożące DSP są bardzo szybkie i mogą zawierać rejestry potokowe (bezpośrednio przed i po operacji mnożenia) i mogą pracować z częstotliwościami nawet powyżej 500 MHz (a nie jak w rozprawie 200 MHz).
Jednym z głównych tematów dyskusji jest zmniejszenie latencji (liczby etapów potokowości) jednak odbywać się to będzie kosztem zmniejszenia częstotliwości zegara. Ja bym tutaj odwrócił pytanie: skupiłbym się na zwiększeniu liczby etapów potokowości w celu zwiększenia częstotliwości zegara. Jak pokazano w (6.11) wpływ latencji na szybkość obliczeń jest minimalny.

Część II. Część druga pracy skupia się na zupełnie odmiennym zagadnieniu badawczym a mianowicie na: szyfrowaniu z użyciem krat. Wybór padł na algorytm Round 5. Pewną wadą tego rozdziału, jest to, że algorytm ten nie został wybrany jako obowiązujący standard szyfrowania. Fakt ten zdecydowanie wpływa na praktyczne zastosowanie projektu. Część ta rozpoczyna się rozdziałem 7, który wprowadza nas w temat oraz pokazuje wcześniejsze prace. Rozdział 8 przybliża działanie algorytmu Round 5.

Rozdział 9 opisuje sprzętową implementację algorytmu Round 5. Rozpoczyna się on od profilowania kodu. Wyniki profilowania przedstawione są w Tab. 9.1 i są bardzo obiecujące: często występują funkcje, które wymagają 99% czasu pracy procesora, dlatego sprzętowa akceleracja może dać znaczące skrócenie czasu obliczeń. Implementacja algorytmu Round 5 odbywać się będzie w systemie wbudowanym z użyciem procesora Zynq, dla którego w jednym układzie scalonym występuje wydajny sprzętowy procesor ARM oraz układ programowalny FPGA. Komunikacja pomiędzy procesorem ARM i FPGA (częścią programowalnej logiki) jest więc znacznie ułatwiona, jest wspomagana przez środowisko programistyczne Vivado. Komunikacja pomiędzy częścią programową i logiczną odbywa się przez standard magistrali AXI z użyciem modułu DMA (modułu IP-core najprawdopodobniej dostarczony przez firmę Xilinx). Sam moduł sprzętowy zaprojektowanego akceleratora wykorzystuje uproszczoną magistralę strumieniową AXI Stream. Zaproponowane podejście jest jak najbardziej poprawne i jest to główny trend w systemach wbudowanych a nawet obliczeń chmurowych. Jedyna wątpliwość jaką ma recenzent to dlaczego tego typu podejścia nie zastosowano w pierwszej części pracy. Dzięki temu komunikacja pomiędzy układem FPGA a procesorem CPU miałaby mniejsze znaczenie. Niestety w układach Zynq zasoby programowalne FPGA są z reguły mniejsze, podobnie procesor ARM ma mniejszą moc obliczeniowe jak standardowy procesor x86. Częściowo tłumaczy to wybór części I.

Rozdział 10 opisuje implementację algorytmu Round 5, ale priorytetem tej implementacji jest zmniejszenie zajmowanych zasobów układu FPGA. Dzięki temu możliwa jest implementacja w tańszym / mniejszym układzie FPGA. W konsekwencji czas wykonywania obliczeń się wydłuża. Takie podejście, gdzie możemy zmniejszyć zajmowane zasoby sprzętowe kosztem dłuższego czasu obliczeń jest jak najbardziej poprawne i często stosowane w sprzętowej akceleracji obliczeń. Warto podkreślić, że zarówno w rozdziale 9 jak i w rozdziale 10 podano dużą ilość wyników eksperymentu badawczego co czyni te rozdziały niezwykle cenne. Wyniki są również obiecujące ponieważ uzyskano znaczącą akcelerację.

3. Pytania i uwagi krytyczne

W rozprawie doktorskiej często zamieszcza się wprowadzenie do tematu badawczego. W tej rozprawie bazuje się bardziej na podaniu referencji naukowych, np. wstęp teoretyczny ma tylko cztery strony. Jest to poprawne naukowo, ale stosowane z reguły w krótkich artykułach. W konsekwencji rozprawa nie jest łatwą lekturą dla czytelnika. Powyższa uwaga dotyczy również opisu oryginalnych osiągnięć Doktoranta. Na przykład, jedną z największych osiągnięć Doktoranta jest zaproponowany Algorytm 4 i jego sprzętowa implementacja. Algorytm ten jest modyfikacją podobnego algorytmu zaproponowanego w [51]. Jednak aby zrozumieć Algorytm 4 potrzebne jest przeczytanie referencji [51], co więcej w rozprawie doktorskiej nie odniesiono się bezpośrednio do wprowadzonych modyfikacji, o skali osiągnięcia i nowatorstwie rozprawy czytelnik musi zdecydować analizując różnice (każdy algorytm zajmuje całą stronę i oba algorytmy różnią się notacją). Podsumowując, odtworzenie oryginalnych rozwiązań zawartych w rozprawie może być pracą naukową samą w sobie. Pewnym uzasadnieniem jest objętość rozprawy, jej nowatorstwo, skala skomplikowania a przede wszystkim wdrożeniowy charakter rozprawy.

Doktorant swoją pracę naukową oparł głównie na literaturze naukowej związanej z kryptografią. Natomiast praca doktorska w rzeczywistości oparta jest na „dwóch nogach". Tą drugą „nogą" jest sprzętowa implementacja algorytmów, w szczególności, w układach programowalnych FPGA. Doktorant temu zagadnieniu naukowemu poświęca relatywnie niewiele miejsca, traktując sprzętową implementację bardziej jako narzędzie inżynierskie niż zagadnienie naukowe. Dużym usprawiedliwieniem niniejszej sytuacji jest nowatorski charakter rozprawy doktorskiej, która implementuje nowe algorytmy w układach FPGA. Niemniej recenzentowi brakuje odwołania do podobnych realizacji, np. mnożenia macierzy, sieci neuronowych. Brakuje również podstawowych informacji, np. w jakim języku (np. Verilog / VHDL) projektowane były opisane moduły sprzętowe. Brakuje odniesienia się do zużycia energii dla rozwiązań opartych na procesorach CPU, GP-GPU czy też FPGA.

Niemniej powyższe wady są mało znaczące w stosunku do nakładu pracy Doktoranta, przy sprzętowej implementacji zaproponowanych algorytmów. Zdecydowanym osiągnięciem Doktoranta jest zaproponowany i zaimplementowany równoległy algorytm przesiewania krat tak aby była możliwa wydajna jego implementacja w układach FPGA oraz aby transmisja danych pomiędzy procesorem CPU a układem programowalnym FPGA nie stanowiła wąskiego gardła.

W drugiej część pracy właściwie wzorowo dokonano implementacji i akceleracji algorytmu Round 5. Zaproponowano wersję zoptymalizowaną na szybkość przetwarzania ale wymagającą znacznych zasobów sprzętowych oraz wersję wykorzystującą mniejsze zasoby sprzętowe kosztem dłuższego czasu wykonywania. Na uwagę zasługują podane wyniki implementacji sprzętowej, które pokazują zdecydowane przyśpieszenie dla wersji sprzętowej.

Recenzent ma następujące pytania do Doktoranta, które nie tyle są krytycznymi uwagami dotyczącymi treści pracy ale bardziej sugestią dalszej pracy naukowej:

Pytanie 1.
Proszę porównać wydajność szczytową (teoretyczną) rozwiązań: CPU, GPGPU i FPGA (uwzględniając tylko moduły DSP) dla wykonywania operacji mnożenia skalarnego wektorów. Jaki jest rzeczywisty stopień wykorzystania wydajności szczytowej i co może być tego powodem? Doktorant w pracy porównuje się do implementacji programowej algorytmu Reduce. Na ile zdaniem Doktoranta rozwiązanie programowe (CPU lub GPGPU) może podlegać dalszej optymalizacji?

Pytanie 2.
W sprzętowej implementacji algorytmu przesiewania krat nie wspomniano o możliwości użycia pamięci zewnętrznej SDRAM lub HBM. Podobnie nie wspomniano o możliwości prowadzenia obliczeń w układach heterogenicznych: procesor dedykowany ARM układ FPGA, np. w układach Zynq, które zostały użyte w drugiej części pracy. Proszę o uzasadnienie podjętej decyzji.

W niniejszej recenzji zamieszczono uwagi krytyczne, niemniej nie powinny one w żaden sposób przysłonić pozytywnej opinii i bardzo dobrej jakości recenzowanej rozprawy. Praca jest nowatorska, rozważa do tej pory nieimplementowane sprzętowo zagadnienia kryptografii i kryptoanalizy opartej na teorii krat. Wkraczając na zupełnie nowy temat bardzo łatwo jest popełnić błąd lub przeoczenie, nie zauważyć możliwej optymalizacji. W recenzowanej pracy nie zauważyłem znaczących błędów merytorycznych, główne moje uwagi krytyczne są skierowane raczej w stronę akcentów pracy i miejscami naiwnego opisu w jaki sposób dochodzono do ostatecznego rozwiązania. Podstawowa moja uwaga merytoryczna dotyczy referencyjnego rozwiązania programowego (na procesorach CPU lub GPGPU), z którym porównuje się Doktorant i czy jest ono optymalne. Niemniej rozważanie to jest poza zakresem niniejszej pracy. Praca jest dość obszerna i ma znaczący wkład wdrożeniowy, wymagała również znaczącego zaangażowania projektowego Doktoranta. Praca została napisana w języku angielskim, nie znalazłem w pracy znaczących błędów językowych. Oprócz przedstawionej rozprawy doktorant jest autorem 3 publikacji naukowych.

4. Wniosek końcowy
Uważam, że przedstawione w rozprawie doktorskiej oryginalne pomysły oraz wyniki eksperymentów wnoszą istotny wkład w rozwój dyscypliny informatyka (aktualnie informatyka techniczna i telekomunikacja). Podsumowując stwierdzam, że rozprawa doktorska Pana mgr. inż. Michała Andrzejczaka spełnia wymagania stawiane w "Ustawie o stopniach naukowych i tytule naukowym oraz o stopniach i tytule w zakresie sztuki", zatem wnoszę o dopuszczenie rozprawy doktorskiej do publicznej obrony i dalszych etapów przewodu doktorskiego w dyscyplinie informatyka (aktualnie informatyka techniczna i telekomunikacja).

Jamro Ernest
.....
Ernest Jamro