Tampere University

Väinö-Waltteri Granat

Kypsyysnäyte - Ohjelmistopinon kehittäminen syväoppvien neuroverkkojen kiihdyttämiseksi mukautetulla RISC-V järjestelmäpiirillä

Informaatioteknologian ja viestinnän tiedekunta (ITC)
Kypsyysnäyte ylemmässä korkeakoulututkinnossa
Marraskuu 2024

Sisällys

1 Kypsyysnäyte

Järjestelmäpiirit ovat heterogeenisen laskennan muoto, jossa yhdelle piirille sijoitetaan useita itsenäisiä komponentteja, jotka yhdessä muodostavat integroidun funktionaalisen järjestelmän. Järjestelmäpiiri voi esimerkiksi sisältää prosessoreita, kiihdyttimiä, muistia sekä tulo- ja lähtörajapintoja.

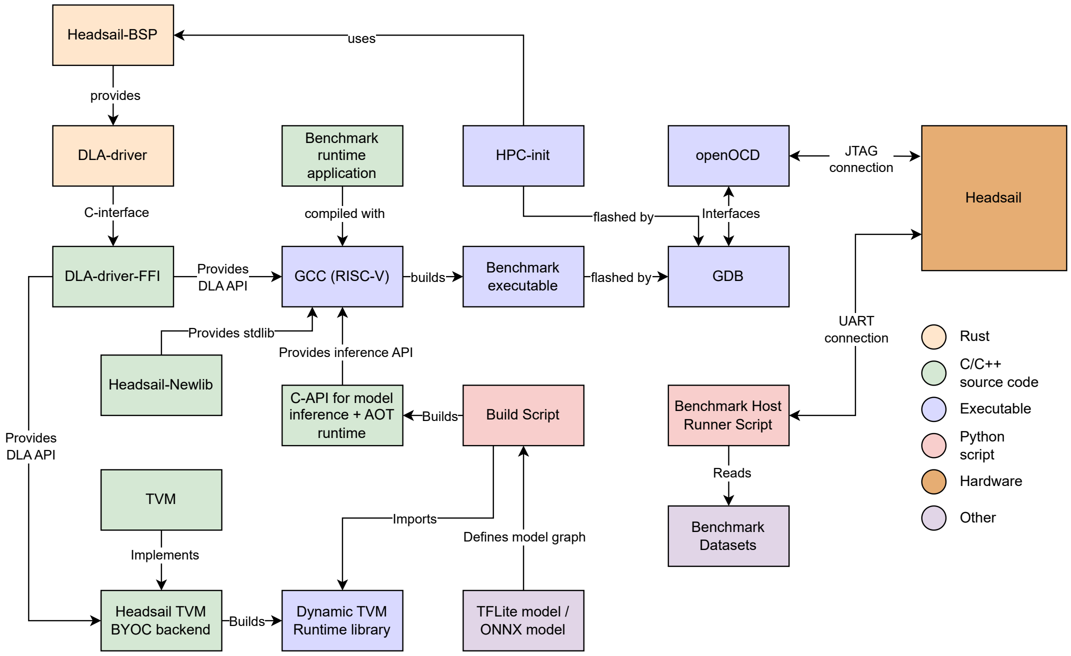
Tässä työssä pyrittiin suunnittelemaan ja toteuttamaan ohjelmistopino konvoluutionaalisten neuroverkkojen päätelaskennan kiihdyttämiseksi Tampereen yliopiston SoC Hub -tutkimusryhmän [5] Headsail-järjestelmäpiirin syväoppimiskiihdyttimellä. Tavoitteena oli kehittää ohjelmistopino, joka pystyy muuntamaan koulutetun neuroverkon Headsaililla suoritettavaan muotoon ja allokoimaan verkon konvoluutiooperaatiot suoritettaviksi syväoppimiskiihdyttimellä.

Työn toinen tavoite oli mallintaa Headsail-järjestelmäpiiri, erityisesti sen syväoppimiskiihdytin, virtuaalisesti Renode-järjestelmäemulaattorilla [1]. Tämä mahdollisti ohjelmistokehityksen aloittamisen ennen Headsail-piirien valmistumista, nopeuttaen piirien käyttöönottoa. Koska syväoppimiskiihdytin on SoC Hubin kehittämä, sen virtuaalista vastinetta ei löydy valmiina Renoden komponenttikirjastosta, joten se kehitettiin tämän työn aikana hyödyntäen Renoden Python-oheislaiterajapintaa.

Lopullinen ohjelmistopino koostui seuraavista komponenteista: Rust-pohjaisesta alustatukipaketista, joka tarjoaa rajapinnan Headsailin ja sen oheislaitteiden ohjelmointiin; tukipakettiin sisällytettävästä ajurista syväoppimiskiihdyttimen ohjaamiseen; TVM-koneoppimiskääntäjästä [3], johon integroitiin Headsail-koodikäännöstuki; sekä Headsail-yhteensopivasta C-standaardikirjaston toteutuksesta. Kuvassa 1.1 on esitetty syväoppimiskiihdyttimen käyttöön kehitetty ohjelmistopino kokonaisuudessaan.

Kiihdyttimen ja sen ohjelmistopinon käyttötapaukseksi valittiin MLPerf Tiny-suorituskykytesti [2], joka on päätelaskennan latenssin testaamiseen sulautetuilla alustoilla tarkoitettu avoin testikehys. MLPerf Tiny sisältää neljä erillistä testitapausta, jotka mallintavat kiihdyttimen tosielämän käyttötapauksia tunnetuilla neuroverkkoarkkitehtuureilla. Näistä tapauksista kolme perustuu konvoluutionaalisiin verkkoihin, siispä ne sopivat hyvin Headsailin syväoppimiskiihdyttimen testaamiseen.

Ensimmäinen käyttötapaus on avainsanojen tunnistaminen, jossa ääninäytteen logaritmisesta mel-spektrogrammista pyritään tunnistamaan yksi kahdestatoista mahdollisesta avainsanasta. Toinen käyttötapaus on visuaalisen herätteen havaitseminen, jossa binääriluokittelevaa konvoluutionaalista verkkoa hyödyntämällä pyritään tunnistamaan, esiintyykö syötteenä annetussa kuvassa ihminen vai ei. Viimeinen käyttötapaus on kuvan luokittelu, jossa ResNet [4] -neuroverkkoa hyödyntämällä



Kuva 1.1 *Headsailin syvöoppimiskiihdyttimen käyttöön kehitetty ohjelmistoarkkitehtuuri.*

pyritään luokittelemaan CIFAR-10-tietokannan kuvia oikeisiin luokkiin.

Neuroverkon kääntäminen Headsailille perustuu TVM:n kykyyn tuottaa ilman käyttöjärjestelmää suoritettavaa C-lähdekoodia, johon voidaan sisällyttää korkeantason syväoppimiskirjastoilla koulutettuja neuroverkkoja. Laajentamalla TVM:ää Headsail-yhteensopivalla koodigeneraatiomoduulilla voitiin lisätä toiminnallisuus, joka mahdollistaa C-koodin tuottamisen siten, että konvoluutiokerrokset allokoidaan suoritettavaksi syväoppimiskiihdyttimellä.

Jotta koodigeneraatio voidaan suorittaa, korkeantason neuroverkon esitys on ensin muunnettava TVM Relay -esitysmuotoon. Relayssä neuroverkko esitetään graafina, jossa jokainen laskuoperaatio vastaa yhtä solmua ja graafin kaaret kuvaavat datan liikkumista operaatioiden välillä. Tälle graafiesitykselle voidaan suorittaa muunnoksia, joilla neuroverkko sovitetaan kohdealustalle. Tämän jälkeen koodigeneraatiomoduuli etsii graafista määritellyt solmukuviot, jotka on ennalta määritelty kiihdyttimellä suoritettaviksi, ja tuottaa näille C-koodia, joka käyttää alustatukipaketin konvoluutiokomentoja päätelaskennan aikana.

Valmiin ohjelmistopinon avulla suoritettiin edellä kuvatut MLPerf Tiny -suorituskykytestin kolme konvoluutiopohjaista käyttötapausta Headsailin virtuaalimallissa. Teknisten ongelmien takia, testikehystä ei pystytty suorittamaan oikealla järjestelmäpiirillä, eikä latenssimittausta saatu siten suoritettua. Virtuaalimallin testaaminen kuitenkin todistaa ohjelmistopinon toimivuuden ja syväoppimiskiihdyttimen yhteensopivuusasteen testikehyksen referenssimallien kanssa. Testitulosten perusteella Headsailin syväoppimiskiihdytin pystyi vaihtelevasti kiihdyttämään yleis-

tapauksellisia neuroverkkoja säilyttäen referenssimallin tarkkuuden. Vaihtelevuuden pääsyyksi havaittiin kiihdyttimen rajoitettu ulostuloleveys, joka osoittautui liian kapeaksi monille neuroverkoille. Jotta syväoppimiskiihdyttimellä voitaisiin suorittaa neuroverkkoja paremmalla tarkkuudella, tulisi verkot kouluttaa rajoitettu ulostuloleveys erityisesti huomioonottaen.

Lähteet

[1] Antmicro. *Renode.* `https://renode.io`. Accessed: 2024-11-26. 2024.

[2] Colby Banbury et al. *MLPerf Tiny Benchmark.* arXiv:2106.07597 [cs]. Elokuu 2021. DOI: `10.48550/arXiv.2106.07597`. URL: `http://arxiv.org/abs/2106.07597` (viitattu 08. 08. 2024).

[3] Tianqi Chen et al. ”TVM: an automated end-to-end optimizing compiler for deep learning”. Teoksessa: *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation.* OSDI’18. Carlsbad, CA, USA: USENIX Association, 2018, s. 579–594. ISBN: 9781931971478.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren ja Jian Sun. *Deep Residual Learning for Image Recognition.* en. Joulukuu 2015. URL: `https://arxiv.org/abs/1512.03385v1` (viitattu 03. 10. 2024).

[5] Henri Lunnikivi, Roni Hämäläinen ja Timo D. Hämäläinen. ”Keelhaul: Processor-Driven Chip Connectivity and Memory Map Metadata Validator for Large Systems-on-Chip”. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2024), s. 1–12. DOI: `10.1109/TVLSI.2024.3454431`.