Xilinx Vivado HLS gyakorlat (2021.)

C implementáció és testbench

1. Töltse le a tárgy honlapjáról a gyakorlathoz tartozó file-t.
2. Hozzon létre egy Vivado HLS projektet az alábbi beállításokkal.
    a. Adja hozzá a projekthez a fir_hw.cpp és types.h fájlokat. A top function legyen fir_hw.
    b. Testbench file-ként adja hozzá a projekthez a main.cpp, fir_sw.cpp file-okat.
    c. Az órajel periódusideje legyen 50 ns; uncertanity: 0.1; az FPGA pedig xc7k70tfbg484-1 (Kintex 7).
3. Definiálja a HW implementációra szánt szükséges adattípusokat az alábbiaknak megfelelően (types.h file-ban).
    a. A minta bemenetek legyenek 24 bites kettes komplemens számok 23 bitnyi tört résszel, wrap és truncate opciókkal. Típusnév: din_t.
    b. Az együtthatók legyenek 18 bites kettes komplemens számok 17 bitnyi tört résszel. A lebegőpontos értékekből kerekítéssel és wrap-olással képződjenek. Típusnév: coeff_t.
    c. Az akkumulátor felbontása legyen a szükséges pontosságú, úgy, hogy túlcsordulási hiba ne fordulhasson elő, plusz wrap és truncate. Típusnév: accu_t.
    d. A kimenet legyen 24 bites (23 bit törtrésszel), mely az akkumulátorból csonkolással és szaturációval generálódjon. Típusnév: dout_t.
4. A fir_sw.cpp file-ba implementáljon egy FIR szűrőt float adattípus használatával. A mintatárat „tömb shifteléssel” valósítsa meg. Az együtthatókészletet a file tartalmazza (coeff_sw tömb).
5. Ellenőrizze a működést Dirac gerjesztéssel!
6. Ellenőrizze a működést egységugrás gerjesztéssel!
7. Írja meg a szűrő FPGA implementációra szánt változatát (azaz kb. copy-paste, a HW implementációra szánt adattípusokkal).
8. A main() függvényben verifikálja, hogy a fixpontos adattípusokat használó megoldás megfelel az elvárásoknak. Azaz hasonlítsa össze a két implementáció hibáját ~100 mintára, s ezt írja ki a konzolra. A verifikációhoz először Dirac gerjesztést generáljon, majd pedig egy véletlen számsorozatot. A letöltött main.cpp ez utóbbit implementálja. A verifikáláshoz a Vivado HLS C debugger-t használja. Mekkora float verzióhoz képest a várt eltérés maximuma?

HW szintézis

1. Futtasson le egy C szintézist. Elemezze az eredményeket:
    a. A Synthesis Summary ablakban a „Performance & Resource Estimates” részben jobb kilikk a fir_hw modulra, majd „Synthesis DEtails”, illetve „Schedule Viewer”.
    b. A szintetizált rendszer mennyi BRAM és DSP blokkot használ? Miért?
    c. Mekkora az egyes ciklusok és a teljes rendszer késleltetése? Miért?
2. Állítsa át az órajel periódusidőt 3.3 ns-ra. Futtassa újra a szintézist.
    a. Mit tapasztal (erőforrásigény, késleltetés)?
3. A fir_hw.cpp fájlt választva a jobb oldali „Directive” fülön mindkét cilkusra adja meg a PIPELINE kikapcsolását (Off).
    a. Hogy változik a rendszer erőforrás igénye és ütemezése?
4. Törölje a PIPELINE direktívát mindkét ciklusról (ez visszakapcsolja a beállítást).
5. Módosítsa az együttható adattípust 32 bitesre (előjeles, 31 bit törtrész), és adaptálja ehhez a többi változó szélességét is.
    a. Mit tapasztal szintézis után?
6. Állítsa vissza a 18 bites együtthatókat.
7. A jobb oldali ablakban válassza ki a Directive fület. Válassza ki a for_shift ciklust, majd adja hozzá a forrás file-hoz az UNROLL direktívát (teljes unroll, így a factor-t nem kell megadni) és törölje a PIPELINE direktívát.
    a. Mennyiben változik az implementáció eredménye (késleltetés, memória igény, regiszter igény)?
    b. Mit tapasztal a Schedule Viewer-ben?
8. Jelölje ki a mintatár tömbjét a Directive ablakban, majd adja hozzá az ARRAY_PARTITION direktívát „complete” opcióval.
    a. Mit tapasztal szintézis után?
9. Állítsa át az órajel periódusidőt 2.0 ns-ra, majd fordítsa le a tervet.
    a. Mit tapasztal?
10. Állítsa vissza az órajel periódusidejét 3.3 ns-ra. Jelölje ki a for_mac ciklust, s adja meg az UNROLL direktívát 2-es factor használatával. Vizsgálja meg a szintézis eredményét.
11. Módosítsa az UNROLL factor értékét 4-re, s ismételje meg az implementációt. Mit tapasztal?
12. Állítsa be a for_mac ciklusra a teljes UNROLL-t, és vizsgálja meg az implementáció eredményét.
13. Törölje az összes eddigi direktívát, majd állítsa be a PIPELINE direktívát a fir_hw függvényre II=1 paraméterrel. Mennyiben változik az implementáció utáni eredmény?
14. Állítsa vissza az ARRAY_PARTITION-t az smpl tömbre. A (11) és (12) pontokhoz képest milyen eredményre jutunk?
15. Írja át a kódot úgy, hogy a MAC művelet (és csak az) külön függvényben van, amelynek paraméterei (együttható, minta, akkumulátor) pointer-ek. Hívja meg a for_mac ciklusban ezt a függvényt. Mit tapasztal implementáció után?

HW implementáció

1. Állítsa vissza a kódot:
    a. 18 bites együtthatók
    b. UNROLL shift
    c. Szekvenciális, pipeline MAC végrehajtás
2. Szintetizálja a C kódot.
3. Hozzon kétre egy Vivado projetet, amelyhez adja hozzá „solution1\syn\verilog\" mappában található fájlokat (azaz a C szintézis eredményét).
4. Szintetizálja a tervet.
    a. Vizsgálja meg a szintézis erőforrás igényét. Megfelel ez az elképzeléseinknek?
    b. Nyissa meg a szintetizált terv kapcsolási rajzát, és keresse meg a mintatár megvalósítását. Mit tapasztal?
    c. Keresse meg a HDL kódban a mintatár → MAC adatút kódját, és nézze meg a HLS által generált megoldást.
5. Módosítsa a C kódot a Vitis HLS shift regiszter osztályának használatával.
    a. Ehhez a #include "ap_shift_reg.h" header fájl szükséges.
    b. Mintatár deklaráció: static ap_shift_reg<din_t, N> smpl;
    c. Metódusok:
        i. X-edik elem olvasása: smpl.read(x)
        ii. X-edik elem olvasása, és bemenet beléptetés: smpl.shift(smpl_din, X);
6. Futtassa le a C szintézist, és vizsgálja meg az erőforrás- és időzítési paramétereket.

Cirkuláris buffer használata

1. Maradva a 18 bites adattípusoknál, írja át a szűrő kódját úgy, hogy a mintatár BRAM-ban megvalósított cirkuláris buffer legyen. Az eddig használt direktívákat távolítsa el! Milyen változótípust célszerű használni a tömbök címzéséhez?
2. A C testbench segítségével ellenőrizze a kód működését!
3. A MAC for ciklusra állítsa be a PIPELINE direktívát II=1 értékkel (Vitis HLS esetében ez az alap beállítás). Megfelel az implementáció a várakozásoknak?
4. Mennyire ideális a BRAM használat?
5. Milyen direktívát lehetne használni a tömbök összevonására? (Megj.: Sajnos a Vitis HLS már nem támogatja ezt a direktívát.)
6. A MAC ciklusra állítson be 2-es UNROLL-t. Megfelel az implementáció a várakozásoknak?
7. Állítson be 4-es UNROLL-t a MAC ciklusra. Mi tapasztal az implementáció után? Miért?
8. Milyen pragma jöhet szóba a 7. probléma áthidalására? Megoldja ez a problémát? Miért?