

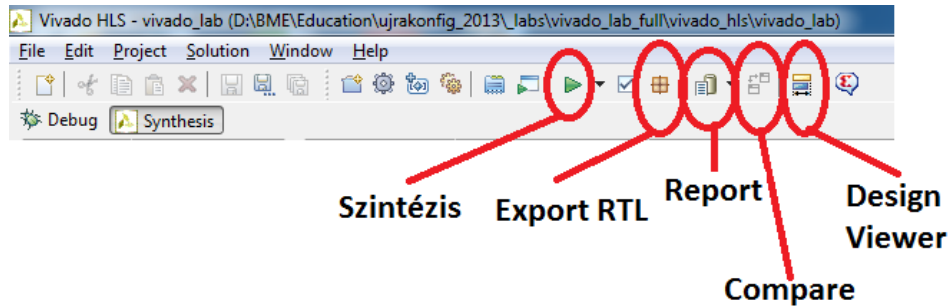
Xilinx Vivado HLS gyakorlat

C implementáció és testbench

1. Töltse le a tárgy honlapjáról a gyakorlathoz tartozó file-t.
 - a. A `fir_sw.cpp` tartalmaz egy referencia implementációt, mely lebegőpontos számábrázolást használ.
 - b. Írja meg a szűrő FPGA implementációra szánt változatát. Ehhez definiálja a szükséges adattípusokat az alábbiaknak megfelelően (`types.h` file-ban).
 - i. A minta bemenetek legyenek 18 bites kettes komplement számok 17 bitnyi tört résszel. Típusnév: `din_t`.
 - ii. Az együtthatók legyenek 18 bites kettes komplement számok 17 bitnyi tört résszel. A lebegőpontos értékekből kerekítéssel képződjenek. Típusnév: `coeff_t`.
 - iii. Az akkumulátor felbontása legyen a szükséges pontosságú, úgy, hogy túlcsonkítási hiba ne fordulhasson elő. Típusnév: `accu_t`.
 - iv. A kimenet legyen 18 bites (17 bit törtreссzel), mely az akkumulátorból csonkolással és szaturációval generálódjon. Típusnév: `dout_t`.
 - c. 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 (egység)impulzus gerjesztést generáljon, majd pedig egy véletlen számsorozatot. A letöltött `main.cpp` ez utóbbit implementálja. A verifikáláshoz a Vivado HLS C debugger-t használja.
2. Indítsa el a Xilinx Vivado HLS szoftvert. Hozzon létre egy projektet a `/vivado_hls` könyvtárban.
 - a. Adja hozzá a projekthez a `main.cpp`, `fir_sw.cpp`, `fir_hw.cpp`, `coeffs.h` és `types.h` fájlokat. A top function legyen `fir_hw`.
 - b. Az órajel periódusideje legyen 100 ns, az FPGA pedig XC6SLX9tqg144-2 (Spartan6).

HW szintézis

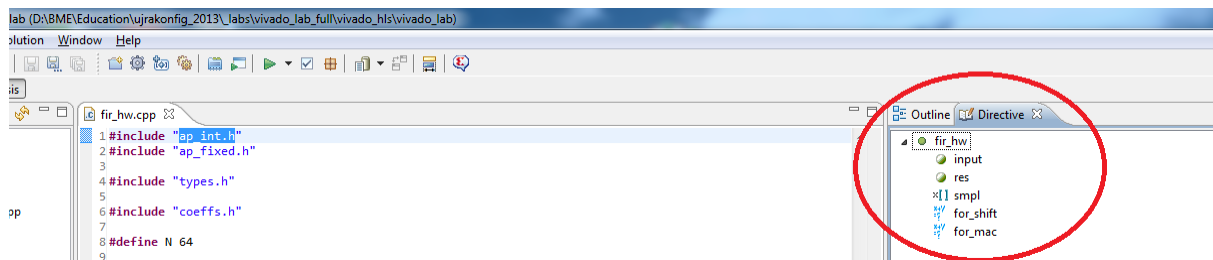
A kezelőfelület fontosabb ikonjai:



A Report-ban találjuk az egyes ciklusok (és a design) összefoglaló paramétereit, valamint a terv erőforrásigényét viszonylag részletes bontásban. A Compare opcióval több implementáció hasonlítható össze (már ha működne a több „Solution” létrehozása). A Szintézis ikon „Synthesize All” opciója után viszont egyetlen implementáció is összehasonlítható, itt némileg más információkat találunk, mint a Report fülön.

A Design Viewer-ben az ütemezés tekinthető meg órajelekre lebontva, így vizsgálható, hogy a Report-ban szereplő ciklus paramétereknek mi az oka, esetlegesen milyen erőforrás okozza a nem megfelelő sebességet. Áttekintése, értelmezése fontos!

1. Futtasson le egy C szintézist. Elemezze az eredményeket a felugró ablak és a Design Viewer segítségével.
 - a. A szintetizált rendszer mennyi BRAM és DSP blokkot használ? Miért?
 - b. Mekkora az egyes ciklusok és a teljes rendszer késleltetése? Miért?
2. Állítsa át az órajel periódusidőt 10 ns-ra (solution1, jobb klikk, Solution Settings/Synthesis), s futtassa újra a szintézist. Mit tapasztal (erőforrásigény, késleltetés)?
3. 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). Mennyiben változik az implementáció eredménye (késleltetés, memória igény, regiszter igény)?

4. Jelölje ki a mintatár tömbjét a Directive ablakban, majd adja hozzá az ARRAY_PARTITION direktívát „complete” opcióval. Mit tapasztal szintézis után?
5. Jelölje ki a for_mac ciklust, s adja meg a PIPELINE direktívát 1-es Iteration Interval használatával. Vizsgálja meg a szintézis eredményét.
6. Jelölje ki a for_mac ciklust, s adja meg az UNROLL direktívát 2-es factor használatával. Vizsgálja meg a szintézis eredményét.
7. Módosítsa az UNROLL factor értékét 4-re, s ismételje meg az implementációt. Mit tapasztal, s mi ennek az oka?

8. Állítsa be az együttható tömbre az ARRAY_REPARTITION direktívát, a factor legyen 2, a típus pedig cyclic. Mit tapasztal a megismételt implementáció után?
9. Állítsa be a for_mac ciklusra a teljes UNROLL-t, és vizsgálja meg az implementáció eredményét.
10. Adja meg a PIPELINE direktívát a fir_hw függvényre II=1 paraméterrel. Mennyiben változik az implementáció utáni eredmény?

Cirkuláris buffer használata

1. Írja át a szűrő kódját úgy, hogy a mintatár BRAM-ban megvalósított cirkuláris buffer legyen. A MAC for ciklusra állítsa be a PIPELINE direktívát II=1 értékkel. Megfelel az implementáció a várakozásoknak?
2. Mennyire ideális a BRAM használat?
3. Állítsa be az ARRAY_MAP direktívát mind a mintatárra, mind pedig az együtthatókészletre. A típus legyen horizontal, az instance pedig ugyanaz (pl. array0) mindkét tömb esetében. Mit tapasztal a szintézis után?
4. Vizsgálja meg az implementáció eredményét a MAC ciklus részleges 2-es UNROLL beállításával. Mi limitálja az elérhető sebességet?
5. Megoldást jelenthet, hogy a memóriák összevonása előtt megduplázzuk a szószélességet, majd az így keletkező tömböket vonjuk össze (így 2 porton 36-36 bit érhető el, azaz 4x18). Ehhez mind az együttható, mind pedig a mintatár tömbre állítsunk be ARRAY_RESHAPE direktívát 2-es factor és cyclic mód használatával, majd az ARRAY_MAP direktívával vonjuk őket össze. Mi lesz a szintézis eredménye?
6. Elrontva a szűrő funkcionalitását a MAC for ciklusában módosítsa a mintatár címzését úgy, hogy az is a ciklusváltozóval történjen. Így mi lesz a szintézis eredménye? Magyarázható az 5. pontban mutatott viselkedés?