1. Vytvorenie nového projektu v programe Vivado HLS použitím Zynq xc7z020clg484-1 (ZedBoard) alebo xc7z010clg400-1 (Zybo)

1.1. Spusťte Vivado HLS cez **Start > Všetky programy > Xilinx Design Tools > Vivado HLS 2015.4.**

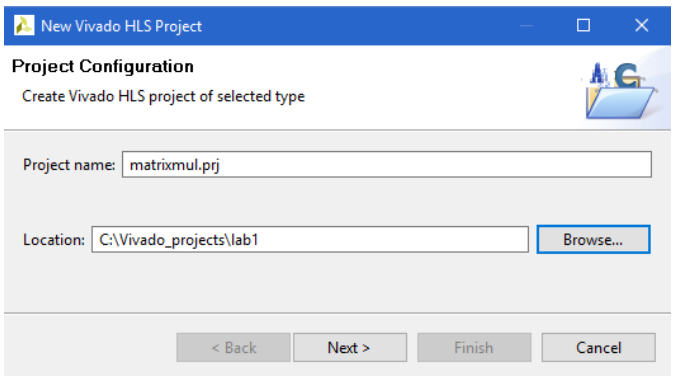
Zobrazí sa úvodná strana programu.

Obrázok 1 Úvodný obraz po spustení Vivado HLS

1.2. Na úvodnej strane kliknite na **Create New Project** alebo choďte cez **File > New Project...** Otvorí sa dialógové okno New Vivado HLS Project.

1.3. V riadku **Project name** nazvite projekt *matrixmul.prj*.

1.4. Kliknite na **Browse...** pre voľbu umiestnenia projektu a uložte ho do **C:\Vivado_projects\lab1**. Potom kliknite na **OK.**



Obrázok 2 Nový projekt vo Vivado HLS

1.5. Kliknite na **Next.**

1.6. V okne *Add/Remove Files* v riadku **Top Function** napíšte matrixmul ako názov (ten sa musí zhodovať so zdrojovými súbormi projektu).

1.7. Kliknite na **Add Files...** vyberte *matrixmul.cpp* súbor z **C:\Vivado_projects\lab1** a potom kliknite na Open.

1.8. Kliknite na **Next**.

1.9. V okne *Add/Remove Files* pre test kliknite na **Add Files...**, vyberte matrixmul_test.cpp súbor z C:\Vivado_projects\lab1 a následne kliknite na **Open**.

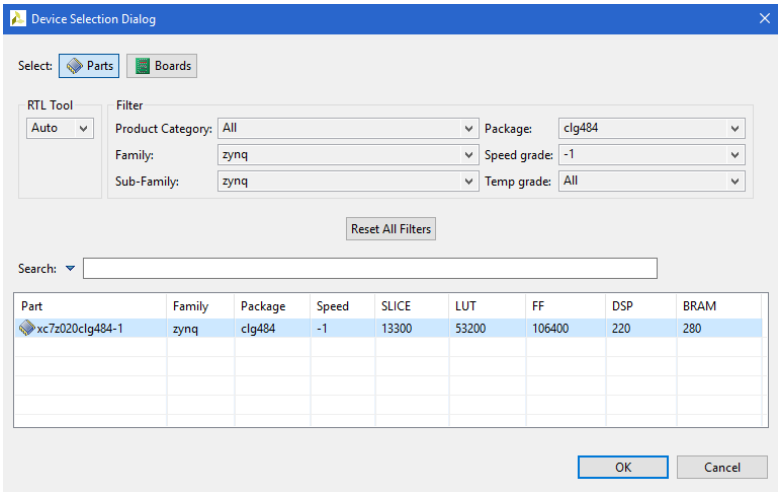
1.10. Kliknite na súbor matrixmul_test.cpp v zozname súborov a následne kliknite na **Edit CFLAGS...** a ako CFLAGS položku uveďte **-DHW_COSIM**, kliknite na **OK.**

1.11. Kliknite na **Next**.

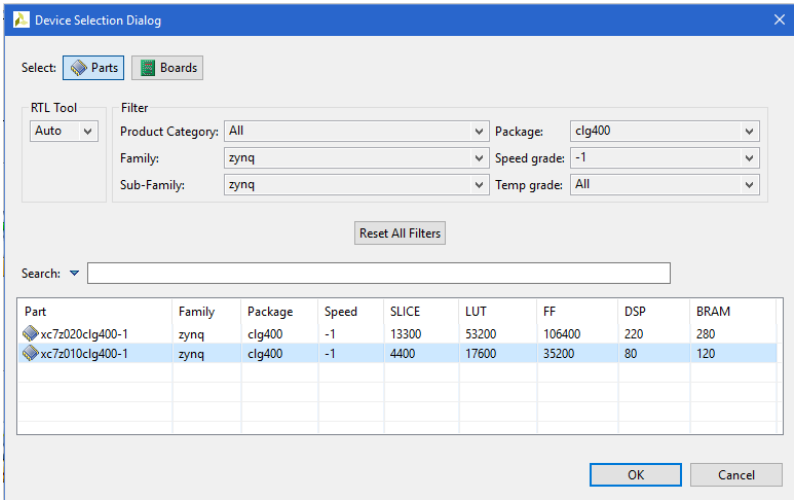
1.12. V okne *Solution Configuration* nechajte Solution Name **solution1** a zmeňte taktovací cyklus na **10** (pre ZedBoard) alebo **8** (pre Zybo). Riadok Uncertainty nechajte prázdny (ZedBoard si stanoví predvolenú hodnotu na 1.25 a Zybo na 1).
Kliknite na **...** v časti *Part Selection*.

1.13. Zobrazí sa nové okno *Device Selection Dialog*, v ktorom vyberte súčasť **xc7z020clg484-1** (ZedBoard) alebo **xc7z010clg400-1** (Zybo).
Vo filtri vyberte:

Family: **zynq**
Sub_Family: **zynq**
Package: **clg484** (ZedBoard) alebo **clg400** (Zybo)
Speed grade: **-1**

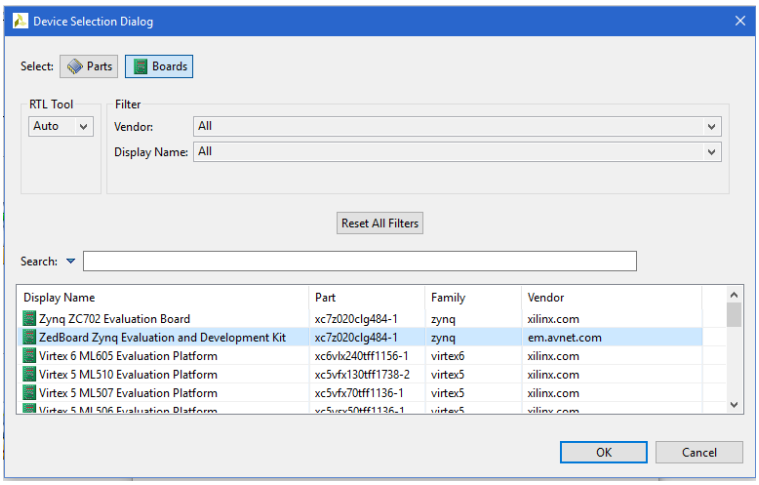


Obrázok 3 Výber súčastí pre ZedBoard



Obrázok 4 Výber súčastí pre Zybo

Pre ZedBoard je možné v časti **Boards** vybrať príslušnú dosku, pokiaľ nejaké na výber máme.

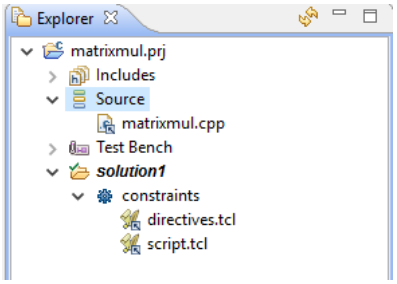


Obrázok 5 Výber Boards pre ZedBoard

1.14. Kliknite na **OK**

1.15. Kliknite na **Finish**

Následne sa vytvorí projekt, ktorý môžete vidieť v okne **Explorer**.



Obrázok 6 Okno Explorer

Rozbalením jednotlivých priečinkov môžete vidieť položky patriace k danému priečinku/podpriečinku.

1.16. Dvojitým klikom na *matrixmul.cpp* sa otvorí zdrojový kód projektu matrixmul.prj.

```
67  #include "matrixmul.h"
68
69  void matrixmul(
70        mat_a_t a[MAT_A_ROWS][MAT_A_COLS],
71        mat_b_t b[MAT_B_ROWS][MAT_B_COLS],
72        result_t res[MAT_A_ROWS][MAT_B_COLS])
73  {
74    // Iterate over the rows of the A matrix
75    Row: for(int i = 0; i < MAT_A_ROWS; i++) {
76      // Iterate over the columns of the B matrix
77      Col: for(int j = 0; j < MAT_B_COLS; j++) {
78        // Do the inner product of a row of A and col of B
79        res[i][j] = 0;
80        Product: for(int k = 0; k < MAT_B_ROWS; k++) {
81          res[i][j] += a[i][k] * b[k][j];
82        }
83      }
84    }
85  }
```

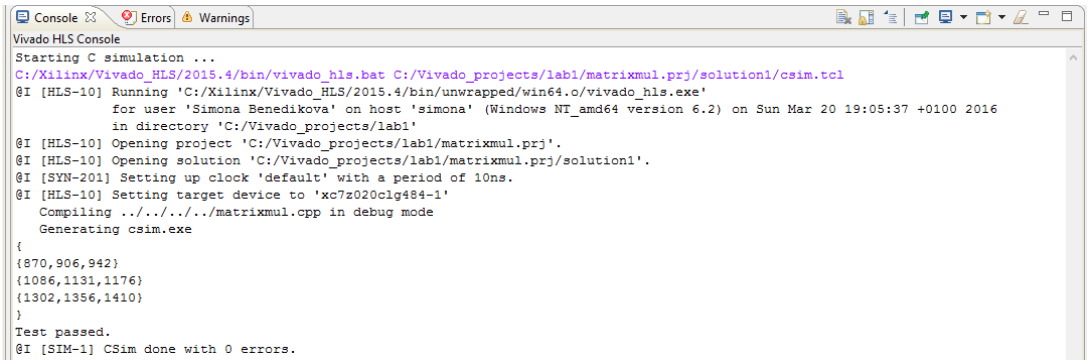
Obrázok 7 Zdrojový kód projektu matrixmul.prj

Ako môžete vidieť, program je zameraný na násobenie matice. Skladá sa z troch cyklov. Cyklus Product je najvnútornejší cyklus, ktorý vykonáva násobenie a následné sčítavanie jednotlivých prvkov matice. Výsledok sa uloží na dané súradnice výslednej matice, podľa toho, ako máme určené [i] a [j]. Stredný cyklus Col vykonáva posúvanie sa po stĺpcoch matice. Taktiež sa tu vykonáva vynulovanie res[i][j], keď cyklus Product prešiel všetkými riadkami daného stĺpca a prechádza sa na nový stĺpec. Cyklus Row je najvrchnejší cyklus a vykonáva posúvanie sa po riadkoch matice.

2. Spustenie C simulácie (Run C Simulation)

2.1. C simuláciu je možné spustiť cez **Project > Run C Simulation**, alebo kliknutím na  na paneli nástrojov. Zobrazí sa nové dialógové okno C Simulation, v ktorom kliknite na **OK**.

2.2. Po skončení simulačného behu môžete v spodnej časti programu v okne *Console* vidieť výstup simulácie.



Obrázok 8 Výstup simulácie

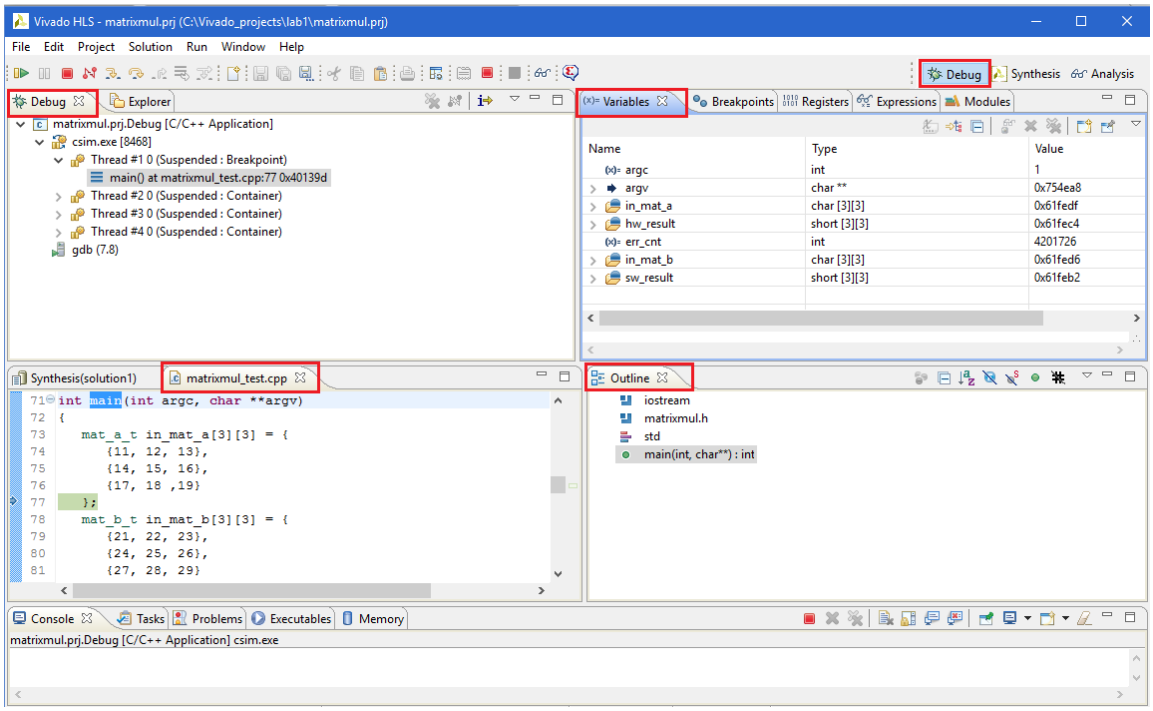
2.3. Rozbaľte priečinok *Test Bench* v okne *Explorer* a dvakrát kliknite na **matrixmul_test.cpp.**

V úvode zdrojového kódu môžete vidieť dve inicializované matice so vstupnými údajmi a ďalej algoritmy, ktoré sa v rámci kódu vykonávajú. Ak je definované HW_COSIM, funkcia matrixmul je volaná a výsledok, ktorý je vypočítaný sa porovnáva s hodnotou vrátenou z volajúcej funkcie. Ak sa výsledky zhodujú, na konci simulácie sa okrem výsledku vypíše aj *Test passed* (Testom potvrdené). Ak HW_COSIM definované nie je, funkcia matrixmul nie je volaná.

3. Spustenie debugger-a

3.1. Vyberte **Project > Run C Simulation** alebo kliknite na  na paneli nástrojov. Na zobrazenom dialógovom okne, je potrebné označiť **Launch Debugger** a následne kliknúť na **OK**.
Aplikácia sa skompiluje a automaticky sa otvorí pohľad debuggera.

3.2. V debugger-i môžete matrixmul_test.cpp vidieť zo zdrojového pohľadu. Premenné argc a argv sú definované v okne Variables, okno Outline zobrazuje objekty v aktuálnom rozsahu.



Obrázok 9 Okno debugger-a

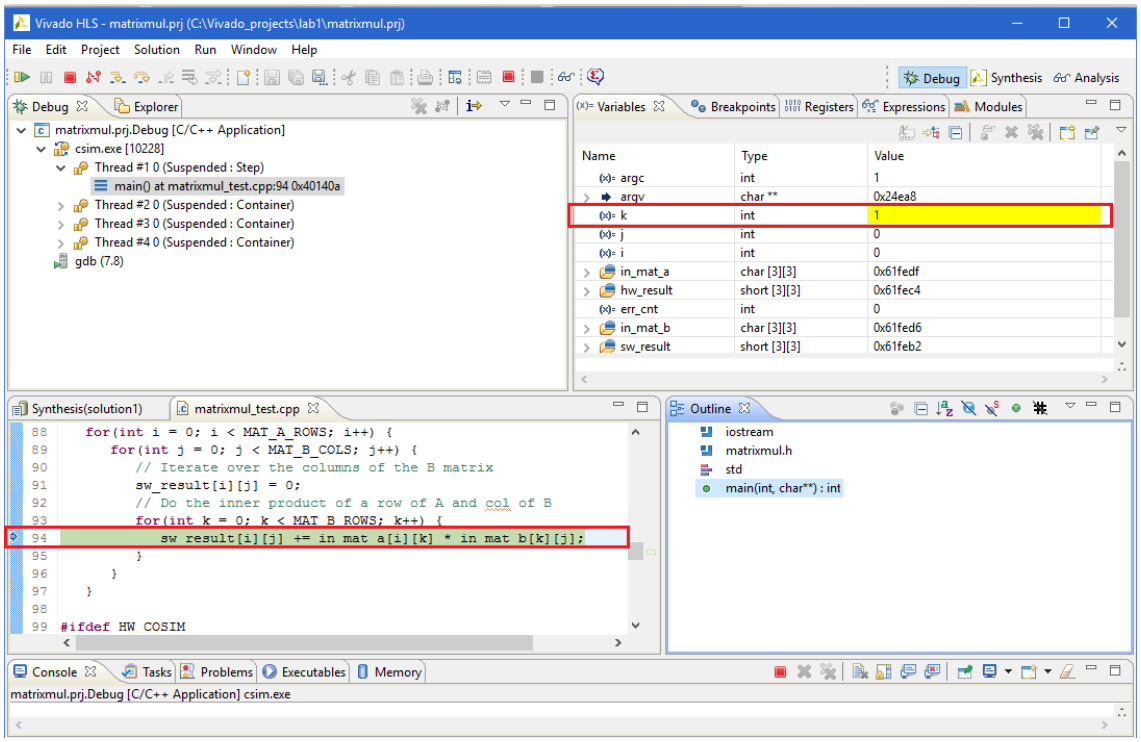
3.3. Presuňte sa v zdrojovom kóde matrixmul_test.cpp na riadok 105 a dvakrát kliknite na modro sfarbený stĺpec pri tomto riadku.

Po tomto kliku sa na danom mieste objaví bod prerušenia (modrý krúžok s fajkou).

```
104    // Print result matrix
105    cout << "{" << endl;
```

3.4. Rovnako kliknite aj na riadok 101, kde je definovaná funkcia matrixmul().

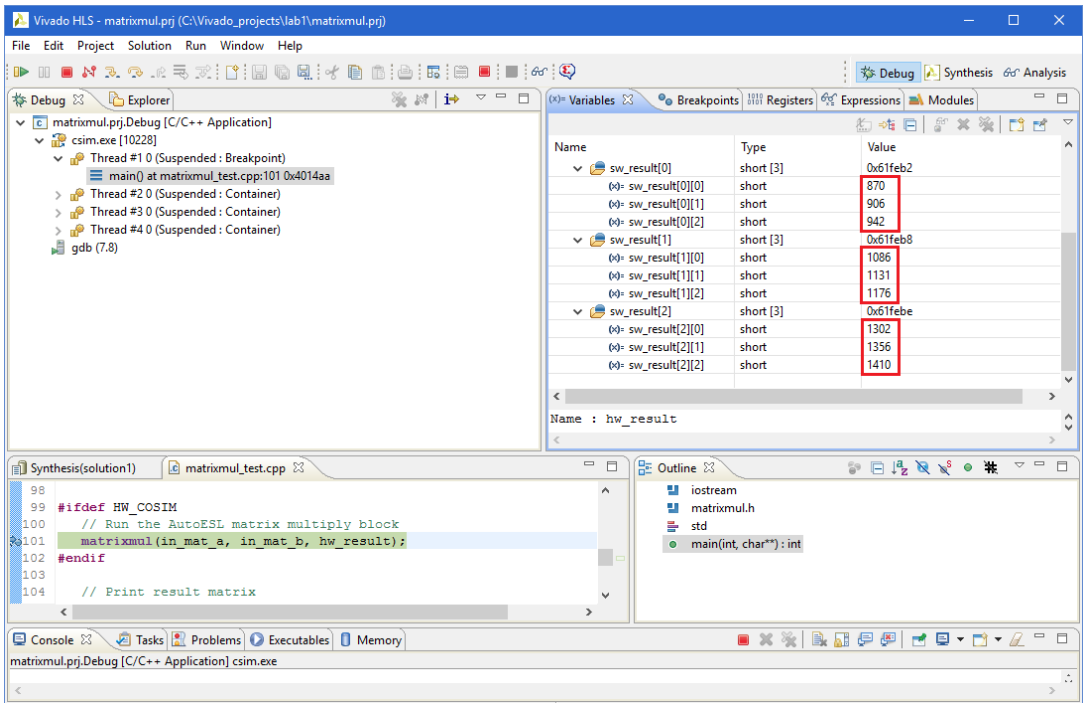
3.5. Niekoľkokrát kliknite na **Step Over (F6)** ( ) a sledujte ako sa menia hodnoty jednotlivých premenných v okne Variables. Zároveň budete vidieť, na ktorom riadku zdrojového kódu sa momentálne nachádzate.



Obrázok 10 Krokovanie zdrojového kódu

3.6. Kliknite na **Resume** ( ) alebo stlačte F8 pre dokončenie výpočtu a program sa zastaví na riadku 101, ktorý sme si v predchádzajúcom kroku označili.

3.7. Pozorujte vypočítaný výsledok v okne Variables.

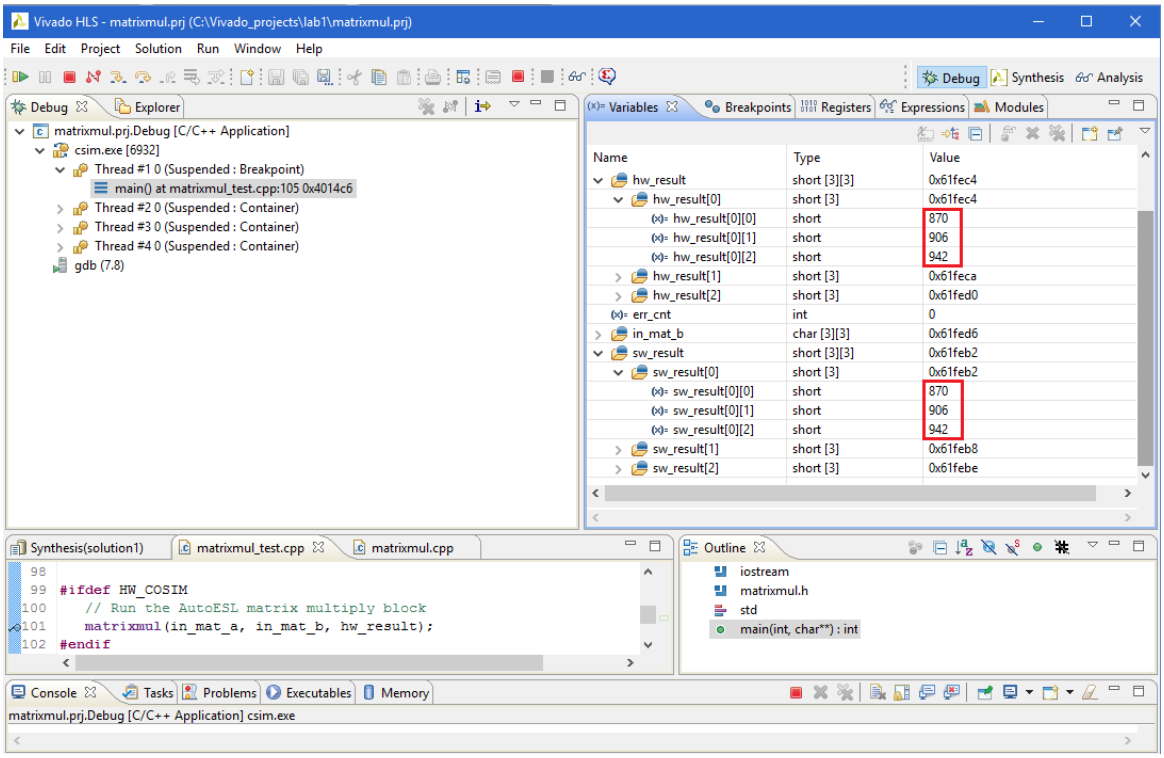


Obrázok 11 Softvérový výsledok programu

3.8. Kliknite na **Step Into (F5)** ( ) na prejdenie do matrixmul modulu a sledujte, či sa program zastavil na riadku 75.

3.9. Niekoľkokrát kliknite na **Step Over (F6)** a sledujte ako sa menia hodnoty premenných v okne Variables. Potom kliknite na **Step Return (F7)** pre návrat z funkcie.

3.10. Program bude pokračovať na riadku 105, ktorý sme si označili v predchádzajúcom kroku bodom prerušenia. Softvérové a hardvérové výsledky môžete sledovať v okne Variables.



Obrázok 12 Softvérový a hardvérový výsledok programu

3.11. Vložte bod prerušenia na riadok 134 (return err_cnt) a kliknite na **Resume**. Program bude pokračovať až po riadok 134. V okne Console sa zobrazí výsledok, ktorý ste mohli vidieť v predchádzajúcom kroku (krok 2.2, Obrázok 8).

3.12. Cez tlačítko **Resume** alebo **Terminate** je debagger možné kedykoľvek vypnúť.

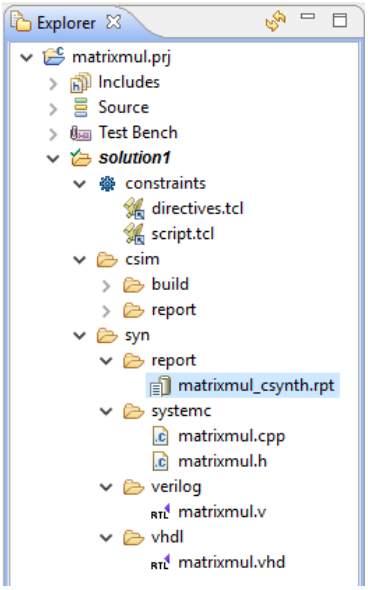
4. Spustenie syntézy (Run C Synthesis)

4.1. Prepnite pohľad z Debug na Synthesis kliknutím na Synthesis na paneli nástrojov.

4.2. Vyberte **Solution > Run C Synthesis > Active Solution** alebo kliknite na  .

4.3. Po spustení syntézy sa zobrazí súhrnná správa (Synthesis Report) spolu s Outline oknom. Okno Outline vás bude navigovať zobrazeným výsledkom jednoduchým kliknutím na jednotlivé časti.

4.4. Ak rozbalíte priečinok **solution1**, môžete vidieť, že boli automaticky vygenerované a vložené súbory do podpriečinka syn > report.



Obrázok 13 Okno Explorer po vykonaní syntézy

Všimnite si, že ak rozbalíte podpriečinok **syn** v priečinku **solution1** v okne **Explorer**, nachádzajú sa tam ďalšie podpriečinky report, systemc, verilog a vhdl v rámci ktorých sú generované (vhdl, verilog, hlavičkový a cpp) súbory. Dvojitým klikom na tieto súbory sa vám zobrazí ich obsah v informačnom okne.

4.5. Súhrnná správa zobrazuje odhad výkonu a využitia ako aj odhad latencie v navrhnutom programe.

4.6. Prechádzajte informačným oknom a odpovedzte na nasledujúce otázky

Odhadovaná perióda taktu: _____

Najhorší prípad latencie: _____

Celková hodnota DSP48E: _____

Celková hodnota FF: _____

Celková hodnota LUT: _____

4.7. Správa tiež zobrazuje najvyššie úrovne signálov rozhrania vytvorené nástrojmi.

Interface

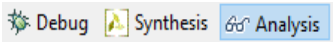
Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	matrixmul	return value
ap_rst	in	1	ap_ctrl_hs	matrixmul	return value
ap_start	in	1	ap_ctrl_hs	matrixmul	return value
ap_done	out	1	ap_ctrl_hs	matrixmul	return value
ap_idle	out	1	ap_ctrl_hs	matrixmul	return value
ap_ready	out	1	ap_ctrl_hs	matrixmul	return value
a_address0	out	4	ap_memory	a	array
a_ce0	out	1	ap_memory	a	array
a_q0	in	8	ap_memory	a	array
b_address0	out	4	ap_memory	b	array
b_ce0	out	1	ap_memory	b	array
b_q0	in	8	ap_memory	b	array
res_address0	out	4	ap_memory	res	array
res_ce0	out	1	ap_memory	res	array
res_we0	out	1	ap_memory	res	array
res_d0	out	16	ap_memory	res	array

Obrázok 14 Generované signály rozhrania

Môžete si všimnúť, že riadiace signály ap_clk, ap_rst, ap_idle a ap_ready boli automaticky dané do dizajnu. Tieto signály sú používané ako tzv. handshaking signály, ktoré ukazujú, kedy je dizajn pripravený začať vykonávať ďalší príkaz výpočtu (ap_ready), kedy ďalší výpočet začal (ap_start) a kedy sú výpočty dokončené (ap_done). Ostatné signály sa generujú na základe vstupných a výstupných signálov v dizajne a ich predvolených alebo zadaných rozhraní.

5. Používanie Pohľadu analýzy (Analysis Perspective)

5.1. Vyber **Solution > Open Analysis Perspective** alebo klikni na panely nástrojov na 

pre otvorenie pohľadu analýzy.

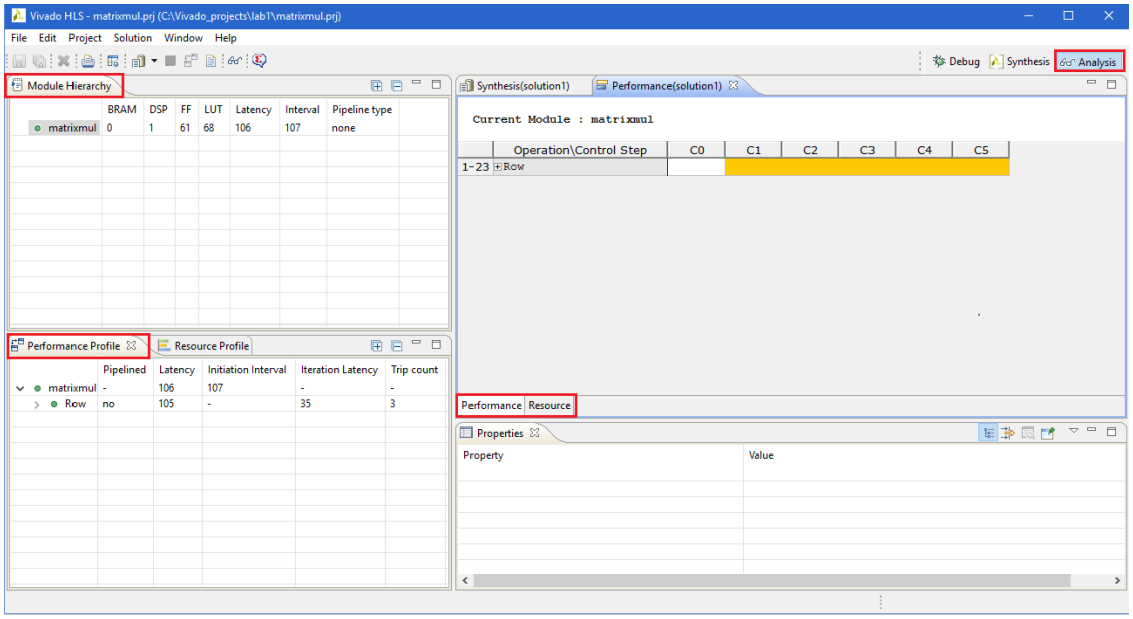
V nasledujúcom pohľade môžete vidieť niekoľko okien.

Okno Module Hierarchy zobrazuje výkon a informácie celého dizajnu a je možné ho použiť na navigáciu hierarchiou.

Okno Performance Profile zobrazuje podrobnosti o výkone pre danú úroveň hierarchie. Informácie v týchto dvoch oknách sú podobné informáciám, ktoré boli zobrazované v súhrnnej správe.

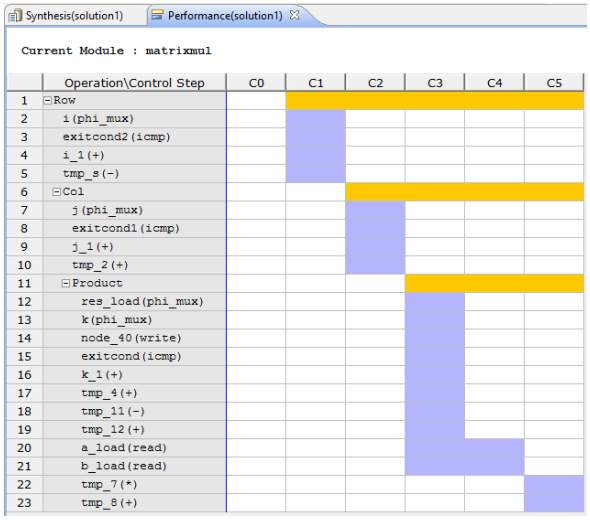
Okno Performance zobrazuje ako sú operácie v tomto bloku naplánované do hodinových cyklov.

- ľavý stĺpec zobrazuje prostriedky
- horný riadok zobrazuje stavy c0 až c5. Sú to vnútorné stavy používané na vysokej úrovni syntézy na naplánovanie operácií v hodinových cykloch



Obrázok 15 Pohľad analýzy

5.2. Rozbaľte jednotlivé cykly Row, Col a Product kliknutím na + pri ich názve.



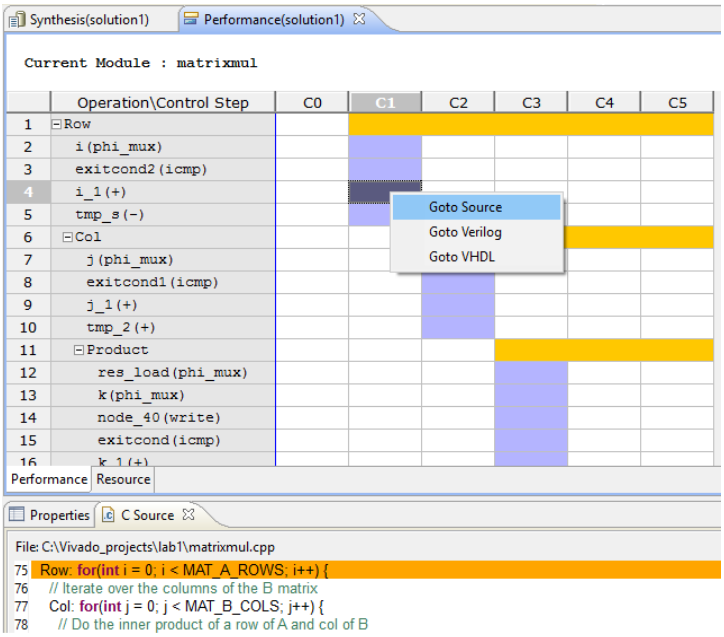
Obrázok 16 Zobrazenie jednotlivých cyklov rozbalením cyklu Row

Z toho môžete vidieť, že v prvom stave C1 cyklu Row sa kontroluje ukončovacia podmienka (exit condition) a vykonávajú sa tu pridávacie operácie. Okrem toho je to počítadlo na sčítavanie počtu iterácií cyklu.

Operácie vyplývajúce z cyklu sú zafarbené na žlto, štandardné operácie sú zafarbené na fialovo a ostatné čiastkové bloky sú zafarbené na zeleno (v našom prípade nemáme žiadne funkcie na nižšej úrovni).

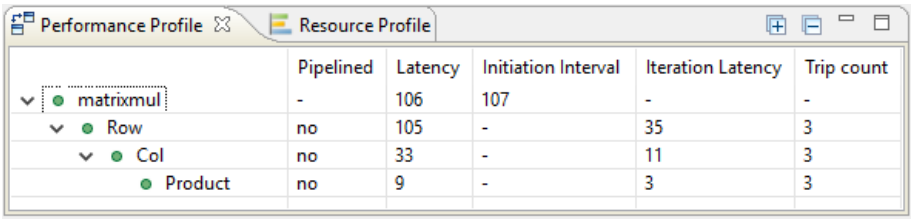
5.3. Kliknite pravým tlačidlom myši na fialové políčko stavu C1 riadku 4 a vyberte možnosť Goto Source.

Panel so zdrojovým kódom sa zobrazí v spodnej časti okna so zvýrazneným 75. riadkom, kde sa začína cyklus Row svoju činnosť. V ďalšom stave (C2) začína svoju činnosť cyklus Col.



Obrázok 17 Zobrazenie zdrojového kódu cyklu Row

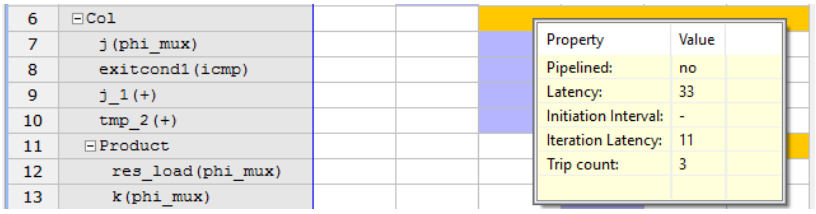
5.4. Rozbaľte všetky položky v okne **Performance Profile** a všimnite si hodnoty, ktoré obsahujú stĺpce Latency, Iteration Latencies a Trip count jednotlivých cyklov.



	Pipelined	Latency	Initiation Interval	Iteration Latency	Trip count
matrixmul	-	106	107	-	-
Row	no	105	-	35	3
Col	no	33	-	11	3
Product	no	9	-	3	3

Obrázok 18 Okno Performance Profile

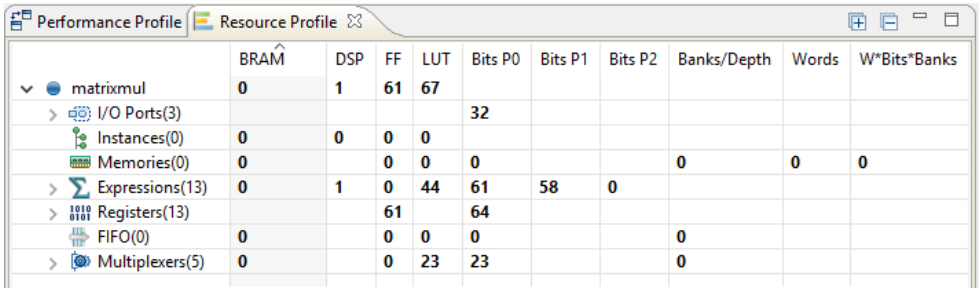
Počet iterácií je možné vidieť aj podržaním myši na žlté riadky jednotlivých cyklov v okne Performance (zobrazené dialógové okno zobrazuje štatistiku cyklu).



Obrázok 19 Štatistika cyklu

5.5. Kliknite na *matrixmul* v okne **Modul Hierarchy** a všimnite si, že položka nie je rozšírená, pretože v dizajne nie sú definované žiadne funkcie na nižšej úrovni.

5.6. Prepnite sa na okno Resource Profile a pozorujte jednotlivé zdroje, kde boli použité. Rozbalením sekcií Expressions a Registers môžete vidieť ako boli zdroje využité, v ktorých operáciách.

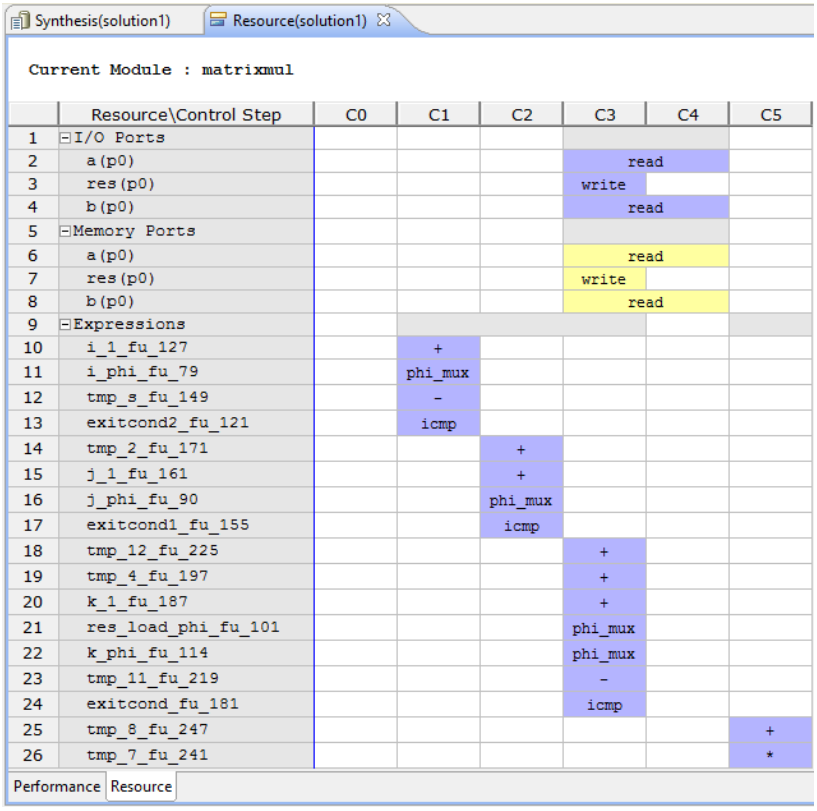


Performance Profile | Resource Profile

	BRAM	DSP	FF	LUT	Bits P0	Bits P1	Bits P2	Banks/Depth	Words	W*Bits*Banks
matrixmul	0	1	61	67						
I/O Ports(3)					32					
Instances(0)	0	0	0	0						
Memories(0)	0		0	0	0			0	0	0
Expressions(13)	0	1	0	44	61	58	0			
Registers(13)			61		64					
FIFO(0)	0		0	0	0			0		
Multiplexers(5)	0		0	23	23			0		

Obrázok 20 Okno Resource Profile

5.7. V okne **Resource(solution1)** sa prepnite na kartu **Resource** a rozbaľte položky I/O Ports a Memory Ports. Pozorujte, ktoré operácie a prostriedky boli použité, a v ktorom stave sa používali.



Synthesis(solution1) | Resource(solution1)

Current Module : matrixmul

	Resource\Control Step	C0	C1	C2	C3	C4	C5
1	I/O Ports						
2	a(p0)				read		
3	res(p0)				write		
4	b(p0)				read		
5	Memory Ports						
6	a(p0)				read		
7	res(p0)				write		
8	b(p0)				read		
9	Expressions						
10	i_1_fu_127		+				
11	i_phi_fu_79		phi_mux				
12	tmp_s_fu_149		-				
13	exitcond2_fu_121		icmp				
14	tmp_2_fu_171			+			
15	j_1_fu_161			+			
16	j_phi_fu_90			phi_mux			
17	exitcond1_fu_155			icmp			
18	tmp_12_fu_225				+		
19	tmp_4_fu_197				+		
20	k_1_fu_187				+		
21	res_load_phi_fu_101				phi_mux		
22	k_phi_fu_114				phi_mux		
23	tmp_11_fu_219				-		
24	exitcond_fu_181				icmp		
25	tmp_8_fu_247						+
26	tmp_7_fu_241						*

Performance | Resource

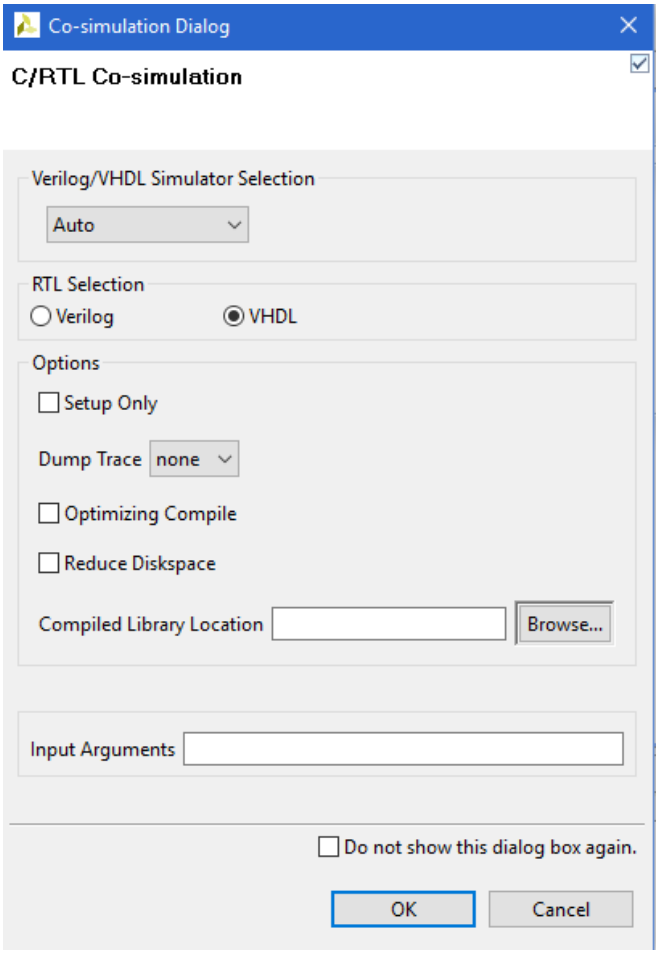
Obrázok 21 Karta Resource

5.8. Kliknite na Synthesis na paneli nástrojov pre návrat na pohľad Synthesis.

6. Spustenie C/RTL Co-simulácie

6.1. Vyberte **Solution > Run C/RTL Cosimulation** alebo kliknite na na paneli nástrojov, ak sa nachádzate v pohľade Synthesis. Následne sa zobrazí dialógové okno.

6.2. V časti RTL Selection vyberte možnosť **VHDL**.



Obrázok 22 C/RTL Co-simulation dialógové okno

6.3. Kliknite na **OK** pre spustenie VHDL simulácie.

C/RTL Co-simulácia sa spustí, vygeneruje sa a skompiluje niekoľko súborov a následne sa nasimuluje dizajn. Prechádza tromi fázami.

- Najprv sa VHDL test vykonáva pre generovanie vstupných podnetov pre RTL dizajn
- Potom je vytvorený RTL test s novo vygenerovanými vstupnými podnetmi a následne prebieha RTL simulácia
- Nakoniec sa výstup z RTL porovná s výsledok z VHDL testu.

V konzolovom okne môžete vidieť postup a správu o schválenom výsledku (Test passed).

```
Starting C/RTL cosimulation ...
C:/Xilinx/Vivado_HLS/2015.4/bin/vivado_hls.bat C:/Vivado_projects/lab1/matrixmul.prj/solution1/cosim.tcl
@I [HLS-10] Running 'C:/Xilinx/Vivado_HLS/2015.4/bin/unwrapped/win64.o/vivado_hls.exe'
            for user 'Simona Benedikova' on host 'simona' (Windows NT_amd64 version 6.2) on Sat Mar 26
13:14:48 +0100 2016
            in directory 'C:/Vivado_projects/lab1'
@I [HLS-10] Opening project 'C:/Vivado_projects/lab1/matrixmul.prj'.
@I [HLS-10] Opening solution 'C:/Vivado_projects/lab1/matrixmul.prj/solution1'.
@I [SYN-201] Setting up clock 'default' with a period of 10ns.
@I [HLS-10] Setting target device to 'xc7z020clg484-1'
@I [SIM-47] Using XSIM for RTL simulation.
@I [SIM-14] Instrumenting C test bench ...
   Build using "C:/Xilinx/Vivado_HLS/2015.4/msys/bin/g++.exe"
   Compiling apatb_matrixmul.cpp
   Compiling matrixmul.cpp_pre.cpp.tb.cpp
   Compiling matrixmul_test.cpp_pre.cpp.tb.cpp
   Generating cosim.tv.exe
@I [SIM-302] Starting C TB testing ...
{
{870,906,942}
{1086,1131,1176}
{1302,1356,1410}
}
Test passed.
@I [SIM-333] Generating C post check test bench ...
@I [SIM-12] Generating RTL test bench ...
@I [SIM-322] Starting VHDL simulation.
@I [SIM-15] Starting XSIM ...

****** xsim v2015.4 (64-bit)
  **** SW Build 1412921 on Wed Nov 18 09:43:45 MST 2015
  **** IP Build 1412160 on Tue Nov 17 13:47:24 MST 2015
    ** Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.

source xsim.dir/matrixmul/xsim_script.tcl
# xsim {matrixmul} -maxdeltaid 10000 -autoloadwcfg -tclbatch {matrixmul.tcl}
Vivado Simulator 2015.4
Time resolution is 1 ps
source matrixmul.tcl
## run all
Note: simulation done!
Time: 1245 ns  Iteration: 1  Process: /apatb_matrixmul_top/generate_sim_done_proc  File:
C:/Vivado_projects/lab1/matrixmul.prj/solution1/sim/vhdl/matrixmul.autotb.vhd
Failure: NORMAL EXIT (note: failure is to force the simulator to stop)
Time: 1245 ns  Iteration: 1  Process: /apatb_matrixmul_top/generate_sim_done_proc  File:
C:/Vivado_projects/lab1/matrixmul.prj/solution1/sim/vhdl/matrixmul.autotb.vhd
$finish called at time : 1245 ns
## quit
INFO: [Common 17-206] Exiting xsim at Sat Mar 26 13:15:37 2016...
@I [SIM-316] Starting C post checking ...
{
{870,906,942}
{1086,1131,1176}
{1302,1356,1410}
}
Test passed.
@I [SIM-1000] *** C/RTL co-simulation finished: PASS ***
```

Obrázok 23 Konzolové okno po ukončení co-simulácie

6.4. Akonáhle je simulácia dokončená, otvorí sa správa o co-simulácii v konzolovom okne. Správa hovorí, či bol výsledok simulácie schválený alebo nie. Okrem toho správa zobrazuje namerané latencie a interval.

Vzhľadom k tomu, že sme vybrali len VHDL, výsledok zobrazuje latencie a interval, ktoré označujú po koľkých hodinových cykloch bude možné poskytnúť ďalší vstup.

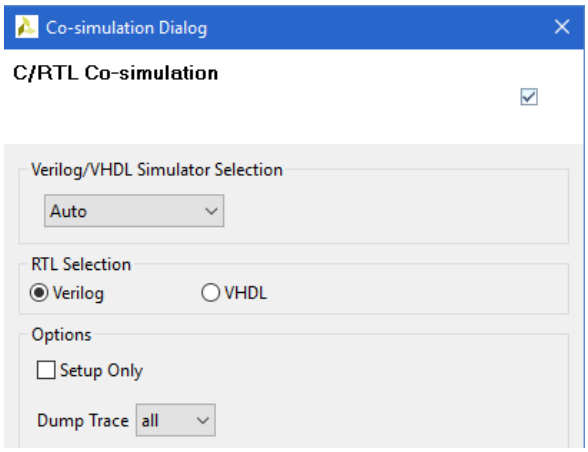
7. Zobrazenie výsledkov simulácie vo Vivado

7.1. Spustenie Verilog simulácie so zvolenými nastaveniami Dump Trace

7.1.1. Vyberte **Solution > Run C/RTL Cosimulation** alebo kliknite na ☑ na paneli nástrojov.

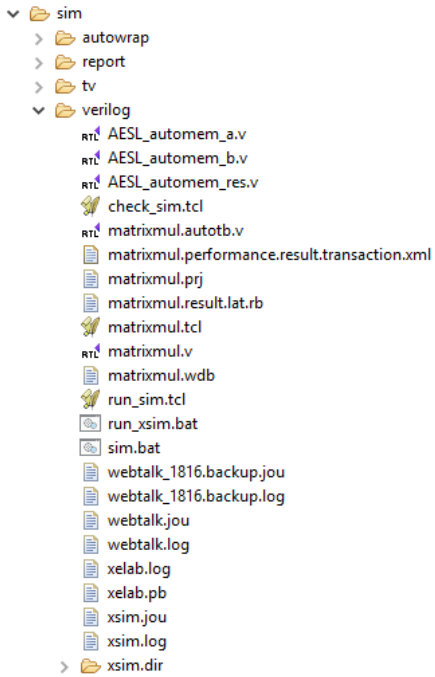
7.1.2. V zobrazenom dialógovom okne kliknite v časti RTL Selection vyberte možnosť **Verilog** a ponechajte možnosť Auto v časti Verilog/VHDL Simulator.

7.1.3. Vyberte možnosť **All** v časti **Dump Trace** a kliknite na **OK.**



Obrázok 24 Nastavenie pre Veriloc co-simuláciu

Akonáhle je co-simulácia dokončená, automaticky sa zobrazí v konzolovom okne potvrdzovacia správa o schválenom výsledku. Navyše preto, že boli použité nastavenia Dump Trace a Verilog, v adresári simulácie Verilog môžete vidieť dve súborové položky.



Obrázok 25 Okno Explorer po skončení Verilog co-simulácie

Správa co-simulácie ukazuje, že test pre Verilog bol schválený spolu s výsledkami pre latencie a interval. Taktiež môžete vidieť výsledky z predchádzajúceho spustenia co-simulácie.

7.2. Spustenie Vivado 2015.4 a vloženie príkazov do príkazového riadku

7.2.1. Spustite Vivado 2015.4 cez Štart > Všetky programy > Xilinx Design Tools > Vivado 2015.4

7.2.2. Po spustení programu napíšte nasledujúce príkazy do príkazového riadku okna Tcl Console

cd c:/Vivado_projects/lab1/matrixmul.prj/solution1/sim/Verilog

current_fileset

open_wave_database matrixmul.wdb

create_wave_config

add_wave /

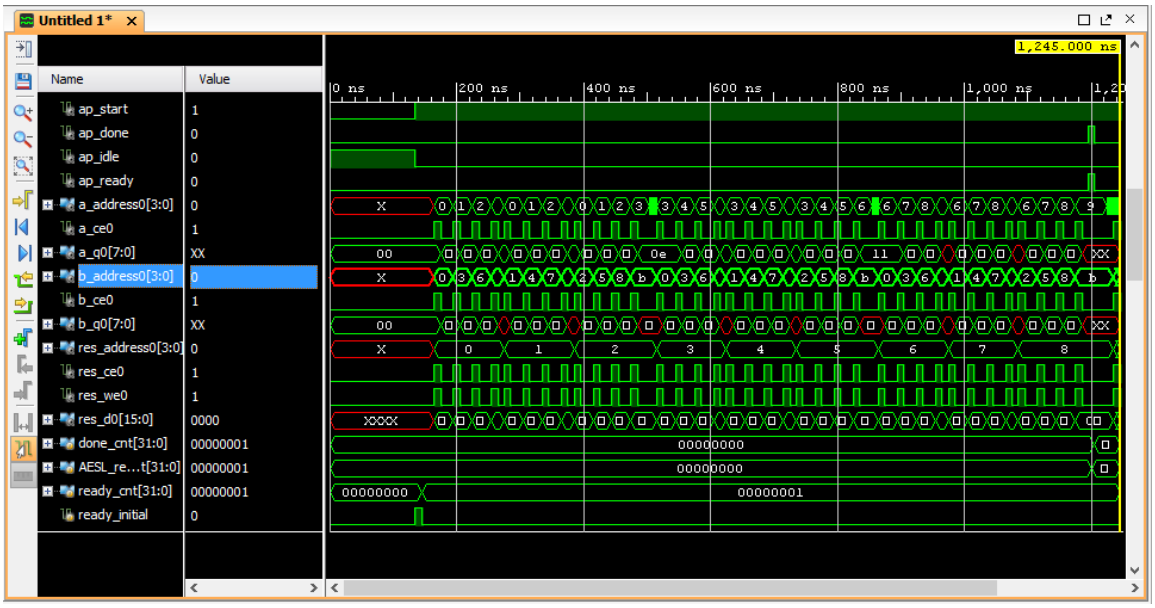
Po zadaní predchádzajúcich príkazov sa načíta projekt, výsledky a priebeh simulácie.

7.2.3. V okne kde sa nachádza priebeh simulácie kliknite na  pre plné priblíženie aby ste v rámci simulácie videli jednu iteráciu.

7.2.4. V okne **Objects** vyhľadajte **a_address0**. Na nájdený výsledok kliknite pravým tlačidlom myši a vyberte možnosť **Radix > Unsigned Decimal**. To isté spravte aj pre **b_address0** a **res_address0**.

7.2.5. Tak isto vyhľadajte **a_q0**, **b_q0**, a **res_d0** a vybrte možnosť **Radix > Signed Decimal**.

7.2.6. V priebehu simulácie sa mierne posuňte nižšie aby ste videli signály rozhrania ap_*.



Obrázok 26 Priebeh simulácie

Všimnite si, že akonáhle sa začal vykonávať ap_start, ap_idle skončil svoju činnosť čo značí, že dizajn je v režime počítania. Signál ap_idle je vypnutý pokiaľ sa nevykoná signál ap_done, ktorý značí dokončenie procesu. To znamená 106 hodinových cyklov latencie.

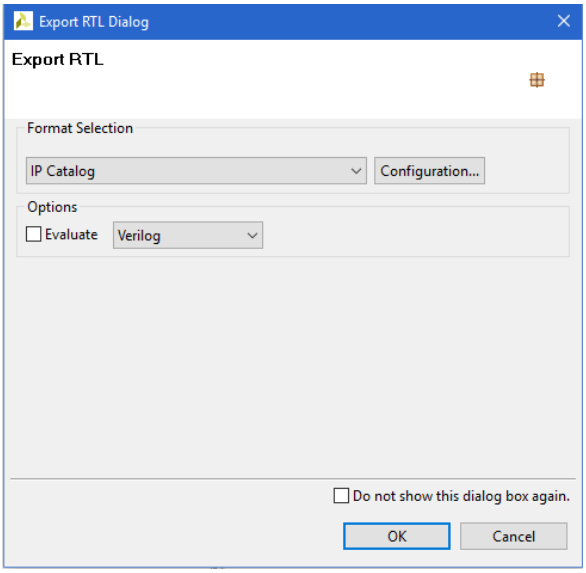
7.2.7. Použite tlačidlo priblíženia tak, aby ste videli priebeh od ~120 do ~550 ns.
Všimnite si, že dizajn očakáva prvky, ktoré poskytujú signály a_address0, a_ceo, b_address0, b_ceo a výsledok na výstupe používa res_d0, res_we0 a res_ce0.

7.2.8. Prezrite si aj ostatné časti simulácie a pokúste sa pochopiť, ako dizajn funguje.

7.2.9. Po prezretí, vyplite Vivado cez **File** > **Exit**. Kliknite na **OK** a vypnite program bez uloženia (možnosť **Discard**).

8. Export RTL a implementácia

8.1. Vo Vivado HLS vyberte **Solution** > **Export RTL** alebo kliknite na  .
Otvorí sa dialógové okno Export RTL Dialog.



Obrázok 27 Dialógové okno pre export RTL

S predvolenými nastaveniami (uvedené vyššie), spustí sa tzv. IP balíčkovací proces a vytvorí balík pre Vivado IP katalóg. S ďalšími možnosťami z rozbaľovacieho menu je možné vytvoriť IP balíčky pre Systém Generátor DSP/Systém Generátor využívajúci ISE, vytvoriť pcore pre Xilinx Platform Studio alebo vytvoriť Syntetizačný kontrolný bod.

8.2. V časti **Option** rozbaľte menu a vyberte možnosť **VHDL** a zaškrtnite možnosť **Evaluate** (vyhodnotiť).

8.3. Kliknite na **OK** spustí sa implementácia.

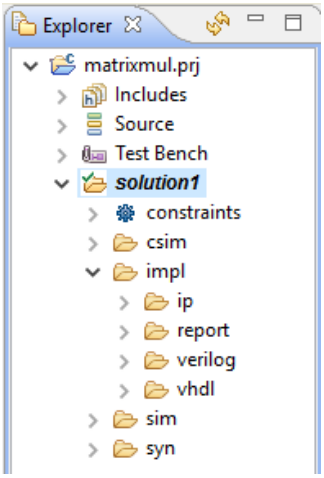
Môžete sledovať jednotlivé kroky v konzolovom okne Vivado HLS. Implementácia prechádza niekoľkými fázami:

- Exportovanie RTL ako IP v IP-XACT formáte
- Vyhodnotenie RTL, pretože ste zvolili možnosť Evaluate
  - prechádzajúce cez syntézu
  - prechádzajúce cez Umiestnenie a Smerovanie

Po skončení exportu sa v informačnom okne zobrazí správa.

Môžete si všimnúť, že sa načasovania stretli, ďalej si všimnite dosiahnuté periódy (7.522 [ZedBoard], 5,943 [Zybo]) a typ s množstvom prostriedkov, ktoré boli použité.

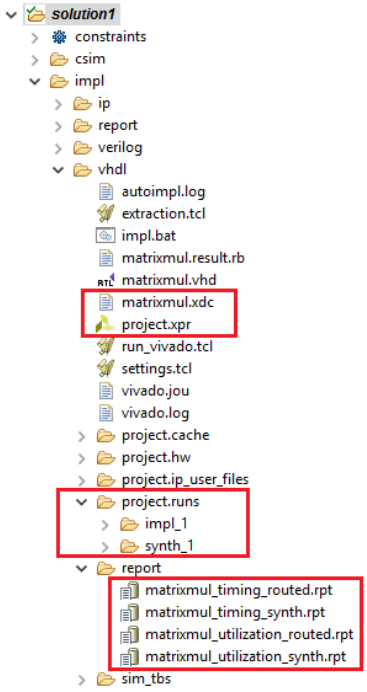
8.4. V okne Explorer pozorujte, či bol vytvorený súbor impl v rámci ktorého sa vytvorili podpriečinky ip, report, verilog a vhdl.



Obrázok 28 Okno Explorer po dokončení exportu

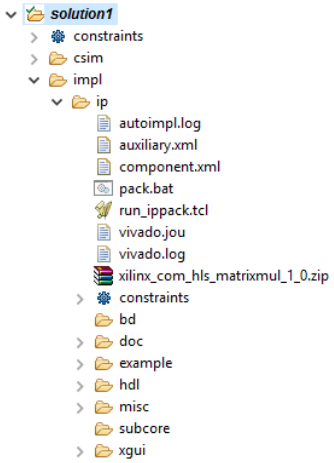
8.5. Rozbaľte podpriečinok verilog a vhdl a všimnite si, že podpriečinok verilog má len rtl súbor, zatiaľ čo podpriečinok vhdl má súborov niekoľko.

Okrem iného sa tu nachádzajú aj súbory ako **project.xpr** (súbor projektu Vivado), **matrixmul.xdc** (súbor na časové obmedzenie), priečinok project.runs (obsahuje podpriečinky synth_1 a impl_1, ktoré sa vytvorili pri behu syntézy a implementácie).



Obrázok 29 Adresáre vyplývajúce z implementácie

8.6. Rozbaľte priečinok ip a všimnite si IP balíček ako súbor formátu zip (xilinx_com_hls_matrixmul_1_0.zip). Je pripravený na pridanie do Vivado katalógu.



Obrázok 30 Obsah priečinku ip

8.7. Zatvorte Vivado HLS výberom **File > Exit.**

Odpovede

Odhadovaná perióda taktu:	8.34 ns (ZedBoard) / 6.38 ns (Zybo)
Najhorší prípad latencie:	106 (ZedBoard) 133 (Zybo)
Celková hodnota DSP48E:	1
Celková hodnota FF:	61 (ZedBoard) 78 (Zybo)
Celková hodnota LUT:	68 (ZedBoard) 69 (Zybo)