- 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.

💫 V	īvado HLS				-	×
File	Edit Project Solution Window	v Help				
₽ ₩ ₩	VIVADO.	4		E XILINX ALI PROGRAMAZE		
	Quick Start			Recent Projects		
	Create New Project	Open Project	Open Example Project			
	Documentation	User Guide	Release Notes Guide			

Zobrazí sa úvodná strana programu.

Obrázok 1 Úvodný obraz po spustení Vivado HLS

- 1.2. Na úvodnej strane kliknite na **Create New Project** alebo choď te 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.

💫 New Vivado HLS Project –	-		×
Project Configuration Create Vivado HLS project of selected type		A	G
Project name: matrixmul.prj			
Location: C:\Vivado_projects\lab1	E	Browse	
< Back Next > Finish		Cancel	

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.
- 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** 

	ion Dialog										)
Select: 💊 Par	rts 📕 Boards										
RTL Tool	Filter										
Auto 🗸	Product Categor	y: All				¥	Package:	clg484			¥
	Family:					~	Speed grad	e: -1			¥
	Sub-Family:	zynq				~	Temp grade	: All			¥
Part	Famil	y Pa	ackage	Speed	SLICE	LUT	FF		DSP	BRAM	
Part	Famil 84-1 zynq	y Pa cl	ackage g484	Speed -1	SLICE 13300	LUT 5320	FF 0 10	5400	DSP 220	BRAM 280	
Part	Famil 84-1 zynq	y Pa cl	ackage g484	Speed -1	SLICE 13300	LUT 5320	0 10	5400	DSP 220	BRAM 280	
Part	Famil 84-1 zynq	y Pa cl	ackage g484	Speed -1	SLICE 13300	LUT 5320	0 10	5400	DSP 220	BRAM 280	
Part	Famil 84-1 zynq	y Pa cl	ackage g484	Speed -1	SLICE 13300	LUT 5320	0 FF	5400	DSP 220	BRAM 280	
Part	Famil 84-1 zyng	y Pa	ackage g484	Speed -1	SLICE 13300	LUT 5320	0 10	5400	DSP 220	BRAM 280	

Obrázok 3 Výber súčastí pre ZedBoard

RTI Tool	Filter									
Auto 🗸	Product Categ	ory:	All			~	Package:	clg400	)	~
Family:			zynq			~	Speed grade	: -1		~
	Sub-Family:	Ē	zvna			~	Temp grade	All		~
earch: 🔻				Re	set All Filters	]				]
earch: 🔻 🗌	Fan	ily	Package	Re	set All Filters SLICE	LUT	FF		DSP	BRAM
earch: ▼ Part ≫xc7z020clg40	Fan 10-1 zyn	ily 1	Package clg400	Re Speed -1	set All Filters SLICE 13300	LUT 5320	0 10	400	DSP 220	BRAM 280
earch: ▼ <sup>y</sup> art	Fan 10-1 zyn 10-1 zyn	ily 1 1	Package clg400 clg400	Re Speed -1 -1	SLICE 13300 4400	LUT 5320 1760	FF 0 100 0 352	400	DSP 220 80	BRAM 280 120

Obrázok 4 Výber súčastí pre Zybo

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

Select: 🔊 Parts						×
	Boards					
RTL Tool	ilter					
Auto V	endor:	All				~
	Namber Manager	All				
L	isplay Name:	A.				•
				-		
			Reset All Filters			
Search: 🔻						
Display Name			Part	Family	Vendor	^
Zyng ZC702 Eva	luation Board		xc7z020clg484-1	zyng	xilinx.com	
		Development Kit	xc7z020clg494-1	70/00	and humat cans	
🧱 ZedBoard Zynq	Evaluation and	Development Kit	XC72020Clg404-1	Lynng	em.avnet.com	
ZedBoard Zynq	Evaluation and valuation Plat	form	xc6vlx240tff1156-1	virtex6	xilinx.com	
ZedBoard Zynq Virtex 6 ML605 E Virtex 5 ML510 E	Evaluation and Evaluation Plat Evaluation Plat	form form	xc6vbx240tff1156-1 xc5vfx130tff1738-2	virtex6 virtex5	xilinx.com xilinx.com	
ZedBoard Zynq Virtex 6 ML605 E Virtex 5 ML510 E Virtex 5 ML510 F	Evaluation and Valuation Plat Valuation Plat	form form form	xc5vfx130tff1136-1 xc5vfx130tff1138-2 xc5vfx70tff1136-1	virtex6 virtex5 virtex5	xilinx.com xilinx.com xilinx.com	

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<sup>©</sup> void matrixmul(
            mat_a_t a[MAT_A_ROWS][MAT_A_COLS],
mat_b_t b[MAT_B_ROWS][MAT_B_COLS],
            result_t res[MAT_A_ROWS][MAT_B_COLS])
73 {
      // Iterate over the rows of the A matrix
Row: for(int i = 0; i < MAT_A_ROWS; i++) {</pre>
74
75
76
          // Iterate over the columns of the B matrix
77
78
          Col: for(int j = 0; j < MAT_B_COLS; j++) {
            // Do the inner product of a row of A and col of B
res[i][j] = 0;
79
80
            Product: for(int k = 0; k < MAT_B_ROWS; k++) {
  res[i][j] += a[i][k] * b[k][j];</pre>
81
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.





2.3. Rozbal'te 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 ina 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.

Vivado HLS - matrixmul.prj (C:\Vivado_projects\lab1\matrixmul.prj)					– 🗆 X
File Edit Project Solution Run Window Help					
🕨 II 🛢 🕺 3. 🥱 IR 🗮 XI 😭 🖬 🖓 🖹 🖻 🖨	💀 🗎 🔳 667 🍳	I		🕸 Debug 🏊 Sy	nthesis & Analysis
🎋 Debug 🖾 🔓 Explorer	🧏 🕅 🖬 🔻 🗖 🗖	(x)= Variables 🙁	● Breakpoints 👭 Register	s 🐼 Expressions 🛋 Modules	
c matrixmul.prj.Debug [C/C++ Application]				🏝 📲 📄 🕤	× 🖗 📑 🖻 🔻
<ul> <li>Csim.exe [8468]</li> <li>Thread #1.0 (Supremoded - Prophysics)</li> </ul>		Name	Туре		Value
<ul> <li>Thread #10 (suspended : Breakpoint)</li> <li>main() at matrixmul test and 77 0.401304</li> </ul>		(×)= argc	int		1
<ul> <li>Thread #2.0 (Supported as Contained)</li> </ul>		> 🔿 argv	char *	ŧ	0x754ea8
Thread #2 0 (Suspended : Container)		🔉 🏉 in_mat_a	char [3	3][3]	0x61fedf
Thread #4 0 (Suspended : Container)		> 🍃 hw_result	short [	3][3]	0x61fec4
adb (7.8)		(×)= err_cnt	int		4201726
Jac (10)		> 🌔 in_mat_b	char [3	3][3]	0x61fed6
		> 🥭 sw_result	short [	3][3]	0x61feb2
		<			>
					A
			1		· · · · ·
Synthesis(solution1)	- 0	🗄 Outline 🛿 🔪		🗊 🖻 🖓 🕅 💱	● # ▽ □ □
71 <sup>©</sup> int main (int argc, char **argv)	^	iostream	1		
72 { 72 } mat a t in mat a[3][3] = (		a matrixm	iul.h		
$73  \text{mat}_a t \text{ in } \text{mat}_a[3][3] = 1$		≣ std			
75 {14, 15, 16},		<ul> <li>main(int</li> </ul>	t, charnn) : int		
76 {17, 18, 19}					
♦ 77 };					
78 mat_b_t in_mat_b[3][3] = {					
79 {21, 22, 23},					
80 {24, 25, 26},					
81 {27, 28, 29}	¥				
<	>				
📮 Console 🛛 🧔 Tasks 🖹 Problems 📀 Executables 🚺 Memory			<b>X</b>	🔆 🗟 🚮 🖨 🚝 📑 🗳	• 📬 • 🖉 🗖 🗖
matrixmul.prj.Debug [C/C++ Application] csim.exe					
					^
					¥
<					>

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).



- 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.

Vivado HLS - matrixmul.prj (C:\Vivado_projects\lab1\matrixmul.prj)			– 🗆 X
File Edit Project Solution Run Window Help			
	r   😜		🕸 Debug 💫 Synthesis 🔗 Analysis
🎋 Debug 🛛 🔁 Explorer 🛛 🐐 😰 🗖 🗖	(x)= Variables 🖾 🤇	• Breakpoints 1919 Registers	🙀 Expressions 🛋 Modules 🛛 🖓 🗖
✓			顓 📲 📄 🌮 💥 💥 📑 🖻 🗵
✓	Name	Туре	Value
Thread #10 (Suspended : Step)	(x)= argc	int	1
main() at matrixmul_test.cpp:94 0x40140a	> 🔿 argv	char **	0x24ea8
Thread #2 0 (Suspended : Container)	(×)= k	int	1
Thread #40 (Suspended : Container)	(×)= j	int	0
📕 gdb (7.8)	(x)= i	int	0
	> 🔚 in_mat_a	char [3][3]	Uxb1tedt
	(x)= err. cnt	int	0
	> (= in mat b	char [3][3]	0x61fed6
	> 🏉 sw_result	short [3][3]	0x61feb2
			÷
	<		>
Synthesis(solution1)		Outline 🛿	🝃 🖻 🎼 🌶 🖋 🖉 🖝 🗶 🗆 🗆
<pre>88 for(int i = 0; i &lt; MAT_A_ROWS; i++) {</pre>	^	iostream	
<pre>89 for(int j = 0; j &lt; MAT_B_COLS; j++) {</pre>		🛀 matrixmul.h	
90 // Iterate over the columns of the B matrix		🚔 std	
91 sw_result[1][]] = 0; 92 // Do the inner product of a row of A and col of B		<ul> <li>main(int, char**): int</li> </ul>	
93 for (int $k = 0$ ; $k < MAT B ROWS; k++)$ {			
94 sw result[i][j] += in mat a[i][k] * in mat b[k][j	];		
95 }			
96 }			
97 }			
99 #ifdef HW COSIM	~		
<	>		
📮 Console 🕱 🖉 Tasks 🔝 Problems 🕥 Executables 🔋 Memory		<b>= x</b> %	<b>}</b>
matrixmul.prj.Debug [C/C++ Application] csim.exe			
			ů.
<			>

Obrázok 10 Krokovanie zdrojového kódu

- 3.6. Kliknite na **Resume** (<sup>III</sup>) 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.

Vivado HLS - matrixmul.prj (C:\Vivado_projects\lab1\matrixmul.prj)		– 🗆 X
File Edit Project Solution Run Window Help		
ID II - N 3. 9. 12 = X   1   2   2   4   1   1   2   2   4   1   1   2   2   2   2   2   2   2   2	tr 🖏	🏇 Debug 📐 Synthesis 🔗 Analysis
🗱 Debug 🕴 🎦 Explorer 🛛 💥 🕼 🖬 🗢 🖓 🗖	🗱 🖉 🖉 🖉 🖉 🖉 (x)= Variables 🕺	🛠 Expressions 🛋 Modules 🛛 🖓 🗖
C matrixmul.prj.Debug [C/C++ Application]     C im.exe [10228]     P foread #10 (Suspended : Breakpoint)     main[0 at matrixmul test.cpp:101 Dx4014aa     P Thread #20 (Suspended : Container)     P foread #20 (Suspended : Container)     P gdb (7.8)	Name         Type           ♥ @ sw_result[0]         short [3]           (0+ sw_result[0][1]         short           (0+ sw_result[0][1]         short           (0+ sw_result[0][2]         short           (0+ sw_result[1]         short [3]           (0+ sw_result[1][2]         short           (0+ sw_result[1][2]         short           (0+ sw_result[1][2]         short           (0+ sw_result[2][2]         short           (0+ sw_result[2][1]         short           (0+ sw_result[2][1]         short           (0+ sw_result[2][2]         short	Image: Second secon
<pre>98 #ifdef HW_COSIM 98 99 #ifdef HW_COSIM 100 // Run the AutoESL matrix multiply block 80101 matrixmul(in_mat_a, in_mat_b, hw_result); 102 #endif 103 104 // Print result matrix </pre>	C Utiline 23 i ostream i matrixmul.h i std o main(int, char**): int	e, ⊡ †5 od K e ±€ ,
📮 Console 🕴 🧔 Tasks 🖹 Problems 🚺 Executables 🚺 Memory	🔳 🗶 💥	🔍 🛃 💭 😾 🖃 🕶 🔂 🕶 🗋 🗋
matrixmul.prj.Debug [C/C++ Application] csim.exe		
<		,

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:	

h	nterface					
	Summary					
	RTL Ports	Dir	Bits	Protocol	Source Object	С Туре
	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

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

#### 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

 <sup>™</sup> Debug Synthesis & Analysis 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.

- l'avý 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

💫 Vivado HLS - r	natrixmul.	prj (C:\	Vivac	lo_proje	ects\lab1\n	natrixmul.p	orj)											-		×
File Edit Projec	t Solutio	on W	indov	v Helj	р															
🔛 🕼 🗶 🕒	5	) - 1	1 60	6	fer 🔇											轸	Debug 💫	Synthesis	66° A	nalysis
🔞 Module Hierard	:hy						E	8 - 0	🗐 Synt	thesis(solution1)	🖶 Performanc	e(solution1)	×							
<ul> <li>matrixmul</li> </ul>	BRAM 0	DSP 1	FF 61	LUT 68	Latency 106	Interval 107	Pipeline ty none	pe	Cur	rent Module :	matrixmul									
									1-23	Operation\C ± Row	ontrol Step	CO	C1	C2	C3	C4	C5			
Performance P	rofile 🛛 Pipeliner -	d Lat	Reso ency	urce Pro Initial 107	ofile tion Interva	al Iteratio	• Latency	Trip count												
> @ Row	no	105		-		35		3	Perform	mance Resource						 				
									Pro	perties 🛛							EI	» 🗔 🗖	~	
									Proper	rty			Value							
									<											>

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.

cu	Operation\Control Step	C0	C1	C2	63	C4	C5
1	-Row		<u> </u>	02	0.5	01	0.5
2	i(phi mux)						
3	exitcond2(icmp)						
4	i 1(+)						
5	tmp s(-)			Goto Sourc	e		
6	-Col			Goto Verilo	g 🛔		
7	j(phi_mux)			Goto VHDL			
8	exitcond1(icmp)		_		,		
9	j_1(+)						
10	tmp_2(+)						
11	- Product						
12	res_load(phi_mux)						
13	k(phi_mux)						
14	node_40(write)						
15	exitcond(icmp)						
16 Perfor	k 1 (+) mance Resource						
🔲 Pro	operties 🖻 C Source 🛛						
File: (	C:\Vivado_projects\lab1\matrixmul.cp	р					
75 F	Row: for(int i = 0; i < MAT_A_ROV	VS; i++) {					

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.

🚰 Performance Profile 🛛 🔄 Resource Profile 🛛 🕞 🗖 🗖										
	Pipelined	Latency	Initiation Interval	Iteration Latency	Trip count					
v o matrixmul	-	106	107	-	-					
V O Row	no	105	-	35	3					
V O 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).

6	⊡Col		
7	j(phi_mux)	Property	Value
8	exitcond1(icmp)	Pipelined:	no
9	j_1(+)	Latency:	33
10	tmp_2(+)	Initiation Interval:	-
11	- Product	Iteration Latency:	11
12	res_load(phi_mux)	Trip count:	3
13	k(phi mux)		

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.

f"	Perf	ormance Profile 📃 F	lesource Profi	le 🖾								
			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						
		Image: Memories(0)	0		0	0	0			0	0	0
	>	Expressions(13)	0	1	0	44	61	58	0			
	>	1010 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.

🗐 Syr	🗊 Synthesis(solution1) 🗧 Resource(solution1) 🛛							
Current Module : matrixmul								
	Resource\Control Step	C0	C1	C2	C3	C4	C5	
1	-I/O Ports							
2	a(p0)				re	ad		
3	res(p0)				write			
4	b(p0)				re	ad		
5	-Memory Ports							
6	a(p0)				re	ad		
7	res(p0)				write			
8	b(p0)				re	ad		
9	Expressions							
10	1_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						*	
Perfor	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.

🔁 Co-simulation Dialog	×
C/RTL Co-simulation	
Verilog/VHDL Simulator Selection	
Auto ~	
RTL Selection	
○ Verilog	
Options	
Setup Only	
Dump Trace none 🗸	
Optimizing Compile	
Reduce Diskspace	
Compiled Library Location Browse	
Input Arguments	
Do not show this dialog box agai	n.
OK Cancel	

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 ...
  /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)
  **** SN 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
## guit
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 ***
```



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.

💫 Co-simulation Dialog	×
C/RTL Co-simulation	V
Verilog/VHDL Simulator Selection	
RTL Selection VHDL	
Options	
Dump Trace all V	

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**.

	Untitled 1* ×																								Ľ	×
																						1,	245.	000	ns	^
	Name	Value	10 ns		1200 r	IS		4	400 ns			1600	ns			1800	ີກຮ			11	.000	ns			1.2	
0	🕼 ap_start	1								1.																
0	👍 ap_done	0																								
0	🖓 ap_idle	0																								
8?	🖟 ap_ready	0																								
\$		0	×		1)2)(			X)		3)	3)(4)(5	00	3)(4)	50	3/4	X\$X	6	)7)	<u>e)()</u>	<u>e</u> k	7)(8)	06	)(7)(6	X		
M	₩ а_се0	1			ШШ			I.				ЛП		ПП						Ш		ΠΠ	ЛЛ	<u>Γ</u>		
	<u>∎</u> • 📲 a_q0[7:0]	xx	00					0		( 0e		<b>\$</b> 0				0	(11		00	ø		$\mathbf{)}0($		▣∕₽	x	
Ľ		0	X	$\odot$	<u>3)EX</u>	ХIX	<u>XZX</u>	æ	XSXBX	ьХ	) <u>(3)(</u>	х	)(4)	<b>7</b> 0	(2)(5	X®X	ь Хо	X3X	εXX	<u>a</u> x	<u>ŧ)(7</u> )	02	XSX	Хф	_X	
2	₩ b_ce0	1			ШΠ			Д				ЛП								Ц		ΠΠ	ЛЛ	Л		
al.	⊞ 📲 b_q0[7:0]	XX	00	_)(0		00		De				фO		(0)		90	0(	)(0)	00	þ		$\mathbf{X}$		⊡)(þ	x	
E		0	×	Х_	0	Х	1	XL.	2	_X	3	ĻХ	4		<u> </u>	\$	_X_	6	X	9	7	<u>_X</u> _	8		_X	
19	res_ce0	1																		Ц.						
쁘	k res_we0	1												Ш						Ц						
		0000	XXXX		₽X®X	XoX	))OX	XQ.			))O)(	ιXXe	ı)(D)		0)0	ж		XoX	<u>XX</u> e		))O)	)e	XeX	IX q	<u>• X</u>	
3		00000001	K								000	0000	0													
1000		00000001	K								000	000	0											<u> </u>		
	🖽 📲 ready_cnt[31:0]	0000001	00000000					<u> </u>					000	0000	1											
	🚡 ready_initial	0										<u> </u>														
																										¥
		< >	<																				_			>

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 <sup>1</sup>/<sub>1</sub>.
Otvorí sa dialógové okno Export RTL Dialog.

💫 Export RTL Dialog	×
Export RTL	#
Format Selection	
IP Catalog	<ul> <li>✓ Configuration</li> </ul>
Options Evaluate Verilog	
	Do not show this dialog box again.

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
  - o prechádzajúce cez syntézu
  - o 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. Rozbal'te 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:1Celková hodnota FF:61 (ZedBoard) 78 (Zybo)Celková hodnota LUT:68 (ZedBoard) 69 (Zybo)