

MIPS based MCU Architecture

FINAL PROJECT

MIPS BASED MCU ARCHITECTURE

2024 SEM B

Guy Cohen 207881004

Liav Ben Or 315909390

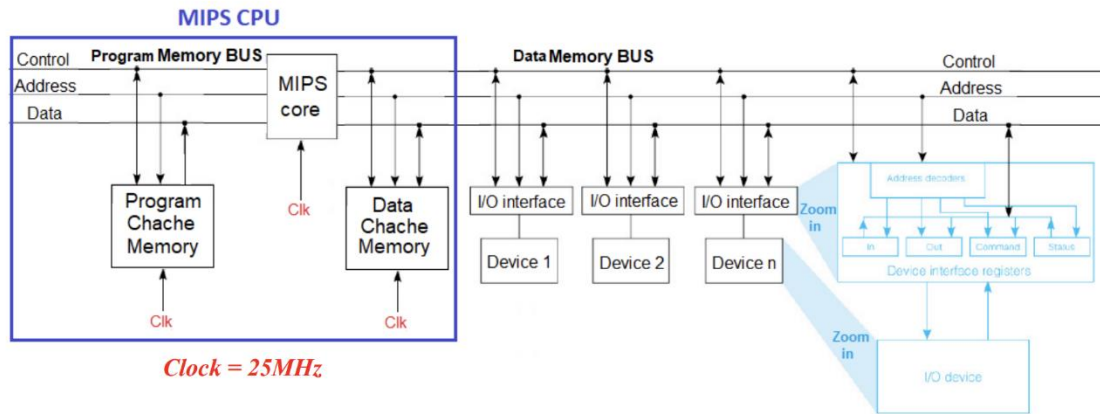
תוכן עניינים

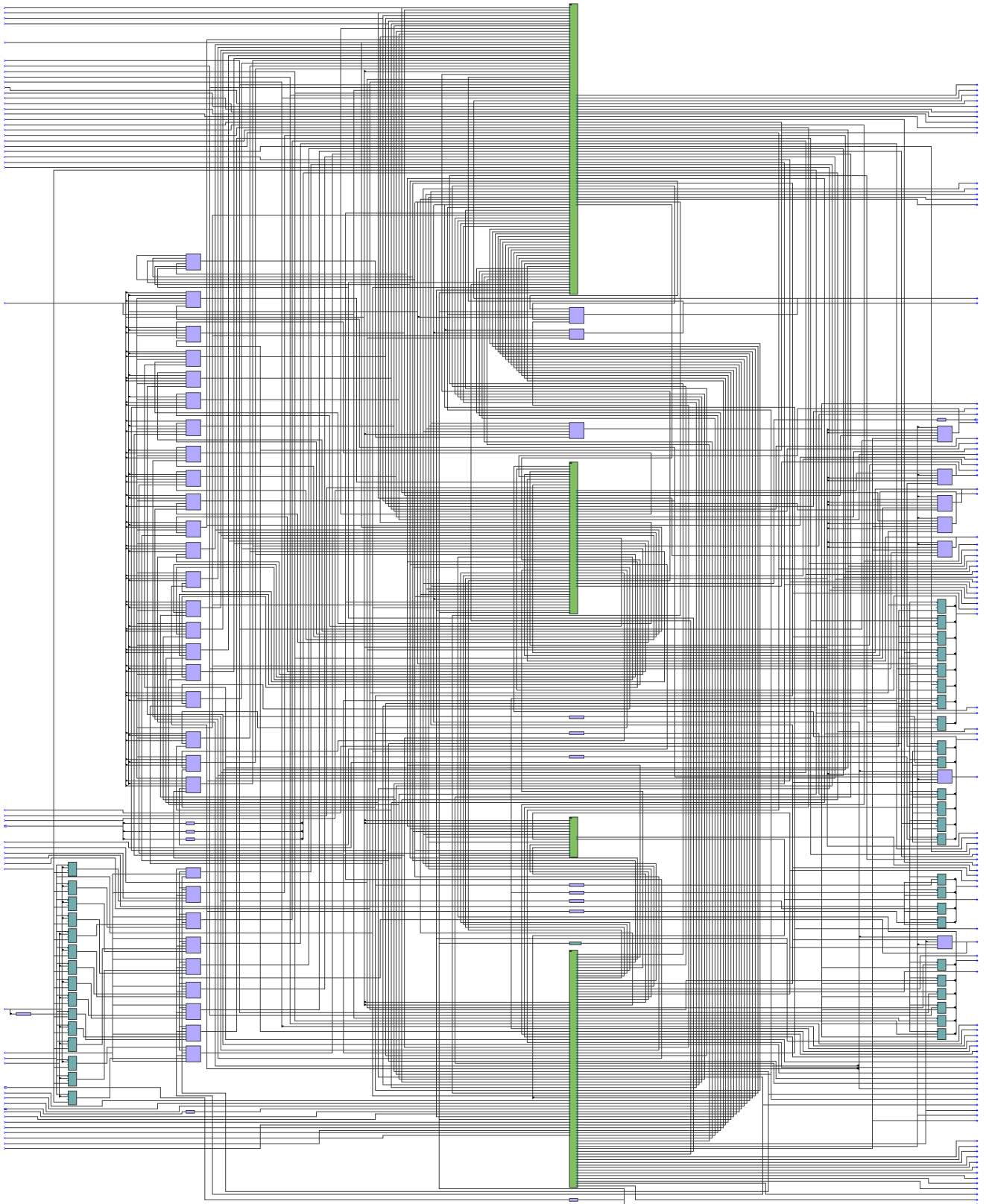
3MCU
5MIPS
9 Instruction Fetch
9 Instruction Decode
9 Execute
9 Data Memory
9 Write Back
10Divider
11Basic Timer
12 Interrupt Controller
13 GPIO
14נתיב קריטי
16ניתוח תוצאות
16 ניתוח גלים ב-Model SIM
18 ניתוח גלים ב-Signal Tap

תיאור המערכת הכללית

בפרויקט זה נרצה לבנות MCU שמכיל מעבד MIPS מסוג Single Cycle המורכב מ-5 שלבים – IF, ID, EX, MEM, WB. התבקשנו להוסיף רכיבי חומרה (Basic Timer, Unsigned Division accelerator, GPIO, Interrupt Controller) שיעבדו בשיתוף פעולה בניהם בהתאם לצורך. התקשורת בין רכיבי החומרה השונים תהיה באמצעות 3 קווי BUS שיעבירו מידע רלוונטי בין הצרכים לצרכנים.

להלן שרטוט המערכת בדיאגרמת בלוקים –





להלן השימוש בקומבינטוריקה הלוגית במודול זה :

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimate of Log...n (ALMs needed)	1961
2		
3	▼ Combinational A...usage for logic	2320
1	-- 7 input functions	62
2	-- 6 input functions	1296
3	-- 5 input functions	248
4	-- 4 input functions	366
5	-- <=3 input functions	348
4		
5	Dedicated logic registers	1780
6		
7	I/O pins	65
8	Total MLAB memory bits	0
9	Total block memory bits	557056
10		
11	Total DSP Blocks	2
12		
13	▼ Total PLLs	1
1	-- PLLs	1
14		
15	Maximum fan-out node	PLL:m1 altpl:atp...eneric_pll1_outclk
16	Maximum fan-out	1599
17	Total fan-out	21595
18	Average fan-out	4.98

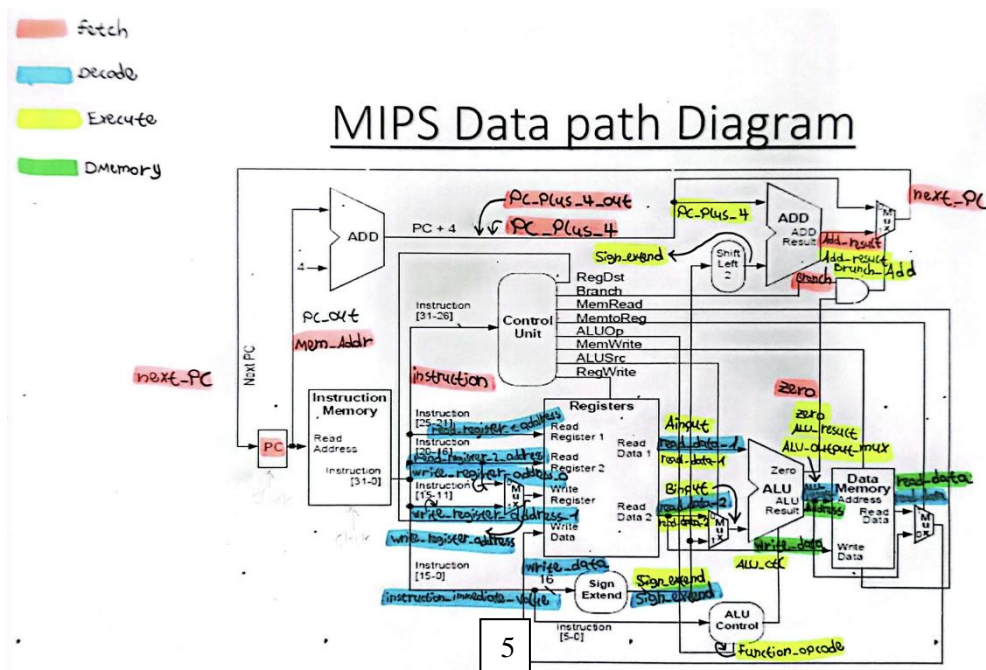
Analysis & Synthesis Summary	
<<Filter>>	
Analysis & Synthesis Status	Successful - Mon ... 2 18:50:13 2024
Quartus Prime Version	21.1.0 Build 842 1...21 SJ Lite Edition
Revision Name	MCU
Top-level Entity Name	MCU
Family	Cyclone V
Logic utilization (in ALMs)	N/A
Total registers	1780
Total pins	65
Total virtual pins	0
Total block memory bits	557,056
Total DSP Blocks	2
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1
Total DLLs	0

להלן השימוש בקומבינטוריקה הלוגית עבור כל יתר הרכיבים:

Analysis & Synthesis Resource Utilization by Entity				
<<Filter>>				
	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Block Mem
1	▼ MCU	2320 (138)	1780 (200)	557056
1	BTIMER:Basic_Timer	75 (75)	102 (102)	0
2	Divider:div_acc	113 (113)	105 (105)	0
3	▶ GPIO:IO_interface	89 (0)	56 (0)	0
4	INTERRUPT:Intr_Controller	132 (132)	14 (14)	0
5	▶ MIPS:CPU	1660 (36)	1191 (21)	557056
6	OptAddrDecoder:OAD	4 (4)	0 (0)	0
7	▶ PLL:m1	0 (0)	0 (0)	0

MIPS

רכיב זה הוא CPU של הMCU, כאשר הוא מסוג Single Cycle. המעבד יודע לתמוך בהוראות המצורפות לתרגיל כך שהוא יודע לבצע את כל התוכניות שמשתמשות אך ורק בהוראות אלו, להלן המעבד עם שמות האותות:



Arithmetic Instructions			
Instruction	Example	Meaning	Comments
add	add \$1,\$2,\$3	$\$1 = \$2 + \$3$	
subtract	sub \$1,\$2,\$3	$\$1 = \$2 - \$3$	
add immediate	addi \$1,\$2,100	$\$1 = \$2 + 100$	
Multiply (without overflow)	mul \$1,\$2,\$3	$\$1 = \$2 * \$3$	Result is only 32 bits!

Logical Instructions			
Instruction	Example	Meaning	Comments
and	and \$1,\$2,\$3	$\$1 = \$2 \& \$3$	Bitwise AND
or	or \$1,\$2,\$3	$\$1 = \$2 \$3$	Bitwise OR
xor	xor \$1,\$2,\$3	$\$1 = \$2 \wedge \$3$	Bitwise XOR
and immediate	andi \$1,\$2,100	$\$1 = \$2 \& 100$	Bitwise AND with immediate value
or immediate	ori \$1,\$2,100	$\$1 = \$2 100$	Bitwise OR with immediate value
xor immediate	xori \$1,\$2,100	$\$1 = \$2 \wedge 100$	Bitwise XOR with immediate value
shift left logical	sll \$1,\$2,10	$\$1 = \$2 \ll 10$	Shift left by constant number of bits
shift right logical	srl \$1,\$2,10	$\$1 = \$2 \gg 10$	Shift right by constant number of bits

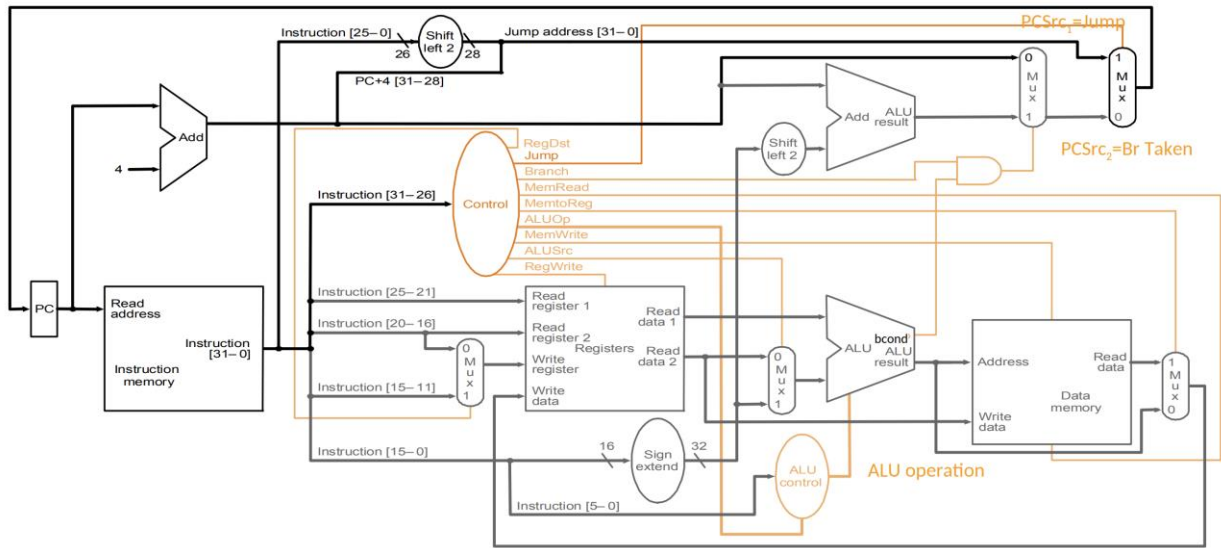
Data Transfer Instructions			
Instruction	Example	Meaning	Comments
move	move \$1,\$2	$\$1 = \2	Pseudo-instruction (provided by MARS Assembler, not processor!) Copy from register to register.
load word	Lw \$1,100(\$2)	$\$1 = \text{Memory}[\$2 + 100]$	
store word	Sw \$1,100(\$2)	$\text{Memory}[\$2 + 100] = \1	
load upper immediate	lui \$1,100	$\$1 = 100 \times 2^{16}$	Load constant into upper 16 bits. Lower 16 bits are set to zero.

Conditional Branch Instructions			
Instruction	Example	Meaning	Comments
branch on equal	Beq \$1,\$2,100	if($\$1 == \2) go to PC+4+100	Test if registers are equal
branch on not equal	Bne \$1,\$2,100	if($\$1 \neq \2) go to PC+4+100	Test if registers are not equal

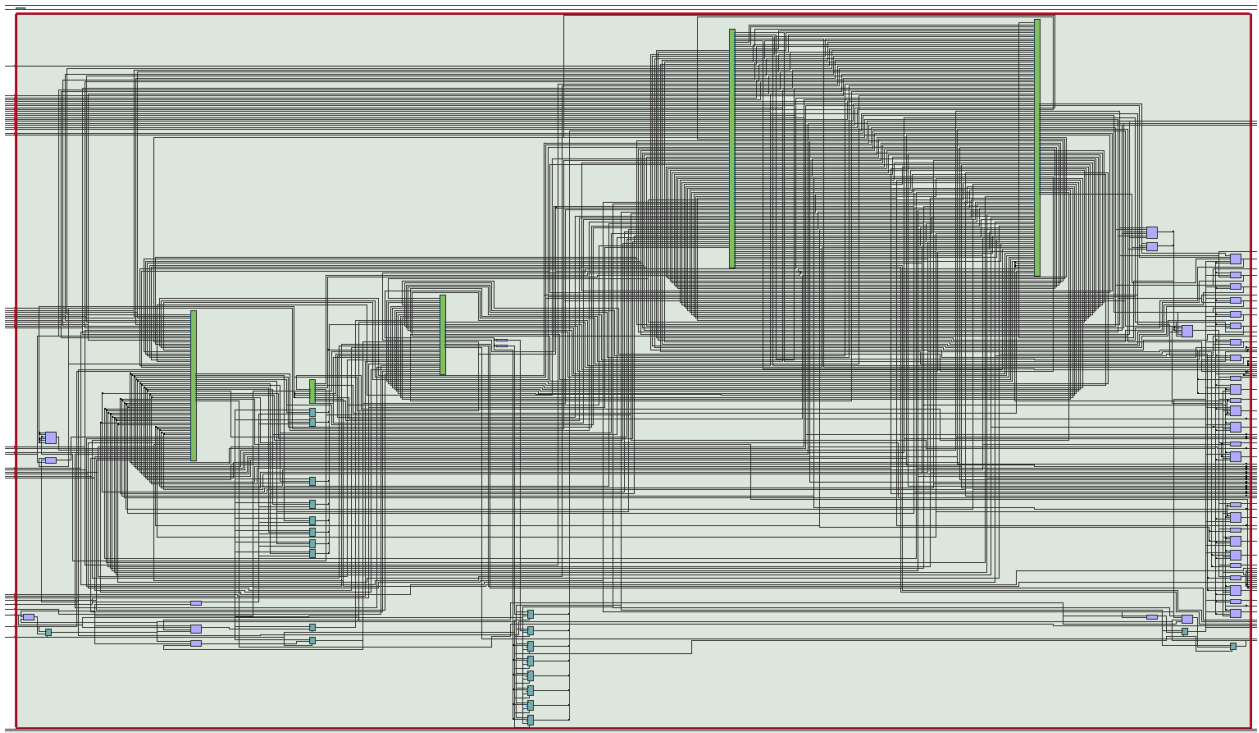
Comparison Instructions			
Instruction	Example	Meaning	Comments
set on less than	slt \$1,\$2,\$3	if($\$2 < \3) $\$1 = 1$; else $\$1 = 0$	Test if less than. If true, set \$1 to 1. Otherwise, set \$1 to 0.
set on less than immediate	slti \$1,\$2,100	if($\$2 < 100$) $\$1 = 1$; else $\$1 = 0$	Test if less than. If true, set \$1 to 1. Otherwise, set \$1 to 0.

Unconditional Jump Instructions			
Instruction	Example	Meaning	Comments
jump	j 1000	go to address 1000	Jump to target address
jump register	jr \$ra	go to return address stored in \$ra	procedure return
jump and link	jal 1000	\$ra=PC+4; go to procedure call which starts in address 1000	Use when making procedure call. This saves the return address in \$ra

להלן שרטוט המעבד –



להלן דיאגרמת RTL כללית במודול זה:



MIPS:CPU

	ALU_result[1]-20
	ALU_result[2]-53
	ALU_result[3]-57
	ALU_result[4]-21
	ALU_result[5]-25
	ALU_result[6]-42
	ALU_result[7]-46
	ALU_result[8]-40
	ALU_result[9]-32
	ALU_result[10]-35
	ALU_result[11]-38
	AddressBus[0]-0
	AddressBus[1]-3
	AddressBus[2]-6
	AddressBus[3]-7
	AddressBus[4]-1
	AddressBus[5]-2
	AddressBus[9]-4
	AddressBus[10]-5
	AddressBus-8
BTIMER:Basic_Timer:Equal4-0	ControlBus[0]-6
BTIMER:Basic_Timer:Equal4-2	ControlBus[1]-5
DataBus[0]-22	ControlBus[2]-4
DataBus[1]-28	ControlBus[3]-1
DataBus[2]-35	ControlBus[4]-2
DataBus[3]-41	ControlBus[5]-3
DataBus[4]-52	ControlBus[6]-0
DataBus[5]-58	DataBus[0]-1
DataBus[6]-64	DataBus[1]-2
DataBus[7]-75	DataBus[2]-3
INTERRUPT:Intr_Controller:DataBus[8]-85	DataBus[3]-4
INTERRUPT:Intr_Controller:DataBus[9]-82	DataBus[4]-5
INTERRUPT:Intr_Controller:DataBus[10]-79	DataBus[5]-6
INTERRUPT:Intr_Controller:DataBus[11]-76	DataBus[6]-7
INTERRUPT:Intr_Controller:DataBus[12]-73	DataBus[7]-8
INTERRUPT:Intr_Controller:DataBus[13]-70	DataBus-0
INTERRUPT:Intr_Controller:DataBus[14]-67	Equal1-0
INTERRUPT:Intr_Controller:DataBus[15]-52	Equal5-[1..2]
INTERRUPT:Intr_Controller:DataBus[16]-55	INTA_sig
INTERRUPT:Intr_Controller:DataBus[17]-58	INTR_STATE[1]
INTERRUPT:Intr_Controller:DataBus[18]-61	ldcode:ID:Equal3-1
INTERRUPT:Intr_Controller:DataBus[19]-64	ldcode:ID:Equal4-1
INTERRUPT:Intr_Controller:DataBus[20]-46	ldcode:ID:Equal5-0
INTERRUPT:Intr_Controller:DataBus[21]-49	ldcode:ID:Equal5-3
INTERRUPT:Intr_Controller:DataBus[22]-103	ldcode:ID:Equal6-0
INTERRUPT:Intr_Controller:DataBus[23]-106	ldcode:ID:Equal7-1
INTERRUPT:Intr_Controller:DataBus[24]-88	ldcode:ID:write_data-0
INTERRUPT:Intr_Controller:DataBus[25]-91	Mux32-11
INTERRUPT:Intr_Controller:DataBus[26]-94	Mux33-11
INTERRUPT:Intr_Controller:DataBus[27]-97	Mux34-11
INTERRUPT:Intr_Controller:DataBus[28]-100	Mux35-12
INTERRUPT:Intr_Controller:DataBus[29]-112	Mux36-11
INTERRUPT:Intr_Controller:DataBus[30]-115	Mux37-11
INTERRUPT:Intr_Controller:DataBus[31]-109	Mux38-11
INTERRUPT:Intr_Controller:INTR	Mux39-11
IN62	Mux40-11
OptAddrDecoder:OAD:CS_LED6-0	Mux41-11
a_internal_jtagaltera_internal_jtag-TCKUTAP	Mux42-10
ra_internal_jtagaltera_internal_jtag-TDIUTAP	Mux43-10
dr_reg	Mux44-10
ena-input	Mux45-10
lrf_reg[1][0..5]	Mux46-10
lrf_reg[2][0..5]	Mux47-10
reset-inputCLKENA0	Mux48-10
reset-input	Mux49-10
splitter_nodes_receive_0[3]	Mux50-10
splitter_nodes_receive_1[3]	Mux51-10
state[3..5]	Mux52-11
state[8]	Mux53-10
virtual_lr_scan_reg	Mux54-10
-GND	Mux55-10
	OUT[49..50]
	OUT[53..54]
	OUT30
	PC[2..9]
	altynoram_dh33:altsynoram:q_a[29]
	register_array[26][0]
	slid_mod_ram_romcmgl_prim2:adapted_id0-0
	slid_mod_ram_romcmgl_prim2:adapted_id0-0
	slid_mod_ram_romcmgl_prim2:lr_loaded_address_reg[0..1]
	slid_mod_ram_romcmgl_prim2:lr_loaded_address_reg[0]
	slid_mod_ram_romcmgl_prim2:lr_loaded_address_reg[1..2]
	slid_mod_ram_romcmgl_prim2:lr_loaded_address_reg[2..3]
	slid_mod_ram_romcmgl_prim2:lr_loaded_address_reg[3..4]
	slid_mod_ram_romcmgl_prim2:lr_loaded_address_reg[4..5]
	slid_mod_ram_romcmgl_prim2:lr_loaded_address_reg[5..6]
	slid_mod_ram_romcmgl_prim2:lr_loaded_address_reg[6]
	slid_mod_ram_romcmgl_prim2:is_in_use_reg
	slid_mod_ram_romcmgl_prim2:is_in_use_reg

Instruction Fetch

בשלב זה נרצה להביא את הפקודה בכתובת ה-PC מהזיכרון של ההוראות. כמו כן, בחלק זה ישב הטיפול ב-PC הנבחר, מפני שבארכיטקטורה המצורפת קיימים תמיד מספר אופציות לשינוי ה-PC –

- נרצה להכניס ל-PC ערך של כתובת אליה נרצה לבצע `jump`
- נרצה להכניס ל-PC ערך של כתובת אליה נרצה לבצע `branch`
- $PC \leftarrow PC + 4$ כאשר נרצה לקדם את הפקודה הבאה

```
Next_PC <= X"00" WHEN Reset = '1' ELSE
  ISRAAddr(9 DOWNT0 2) WHEN Read_ISR_PC = '1' ELSE -- Interrupt!
  jump_address WHEN ((jump = '1' or jump_register = '1') and Read_ISR_PC = '0') ELSE --jump\jal
  Add_result WHEN ( ( Branch = '1' ) AND ( Zero = '1' ) ) ELSE --beq
  Add_result WHEN ( ( Branch_not_equal = '1' ) AND ( Zero = '0' ) ) ELSE --bne
  PC_plus_4( 9 DOWNT0 2 );
```

Instruction Decode

בשלב זה נרצה לקחת את הפקודה שהוראה משלב ה-`fetch` ולהמיר אותה לאחד מסוגי הפקודות שלהלן:

Type	-31- format (bits) -0-					
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rt (5)	immediate (16)		
J	opcode (6)	address (26)				

כך נכניס את כתובת הרגיסטרים המתאימים ל-`RF` שנמצא בשלב ה-`decode` ונוכל להוציא את המידע מהרגיסטרים הדרושים.

Execute

בשלב זה מתבצע השלב החישובי של המעבד, עבור הוראות מסוג `Rtype`, ה-`ALU` לוקח את הנתונים משני רגיסטרים מבוקשים ומעביר את התוצאה לקו הבקרה `Alu_result`. עבור פקודות `Itype` ה-`ALU` מבצע פקולה אריתמטית עבור נתונים מרגיסטר אחד ביחד עם קבוע המאוחסן ב-`instruction`.

Data Memory

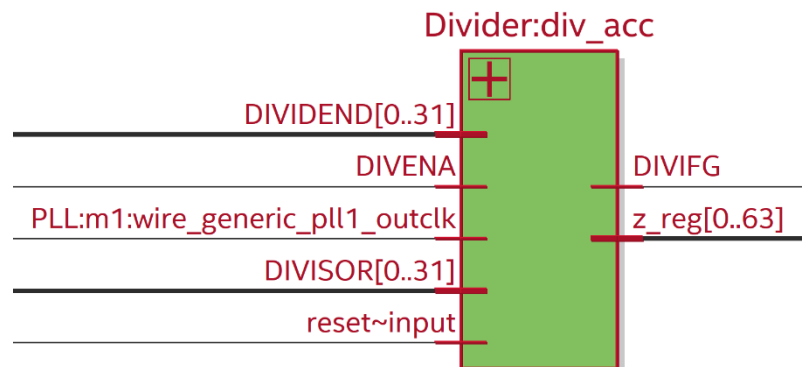
מודול זה אחראי על הכתיבה והקריאה מתוכן ה-`Data Memory` שזה זיכרון ה-`RAM` של המערכת שלנו. אנחנו במשימה השתמשנו בגודל `RAM` של 2^{12} .

Write Back

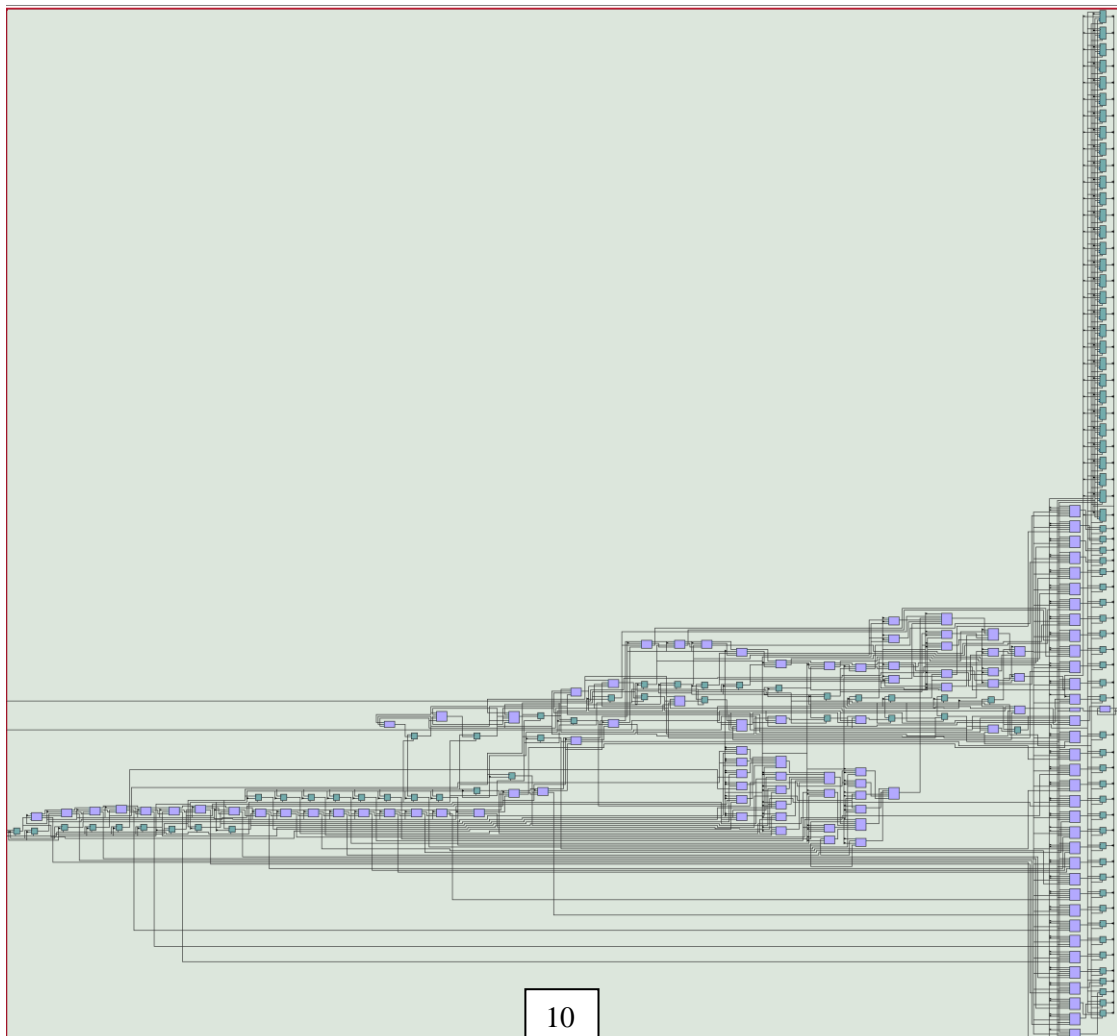
שלב זה מוכל במימוש שלי בתוך ה-`Decode`, אשר מקבל את המידע מה-`ALU` עבור פעולות `Rtype` ומה-`RAM` עבור פעולת `LW`.

Unsigned Binary Division Multicycle Accelerator

רכיב חומרה זה אחראי על חלוקת מספר המכיל 32 ביטים במספר בעל 32 ביטים, בסיום הפעולה רכיב החומרה טוען לזכרון את תוצאת החלוקה כמספר שלם ואת השארית במקומות המתאימים בזכרון. לאחר 32 מחזורי שעון, רכיב החומרה מעלה דגל DIVIFG אשר מודיע לבקר הפסיקות כי הוא סיים את פעולתו ושיש לבצע את רוטינת הפסיקה בהקדם האפשרי. מימשנו את הרכיב כך שהוא יעבוד רק כאשר הוכנס מספר מחלק אשר שונה מ0 כדי להימנע מexceptions.



RTL:

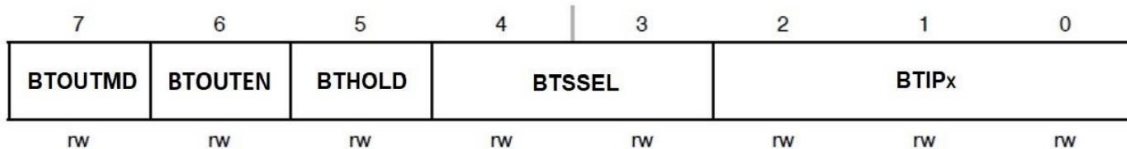


Basic Timer

תפקידי רכיב חומרה זה:

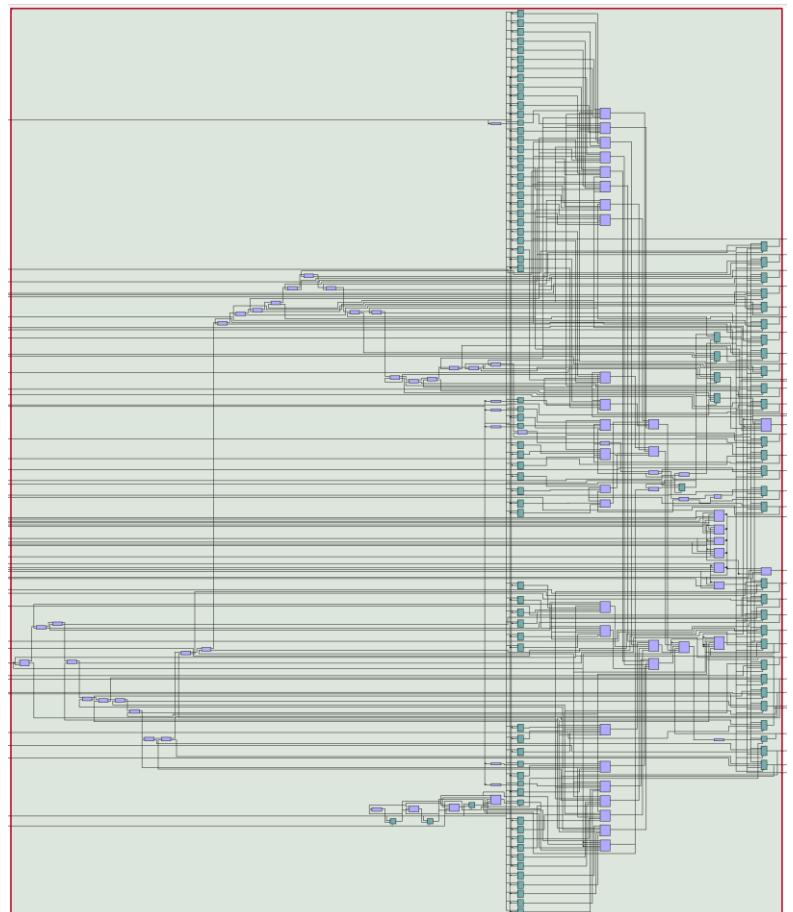
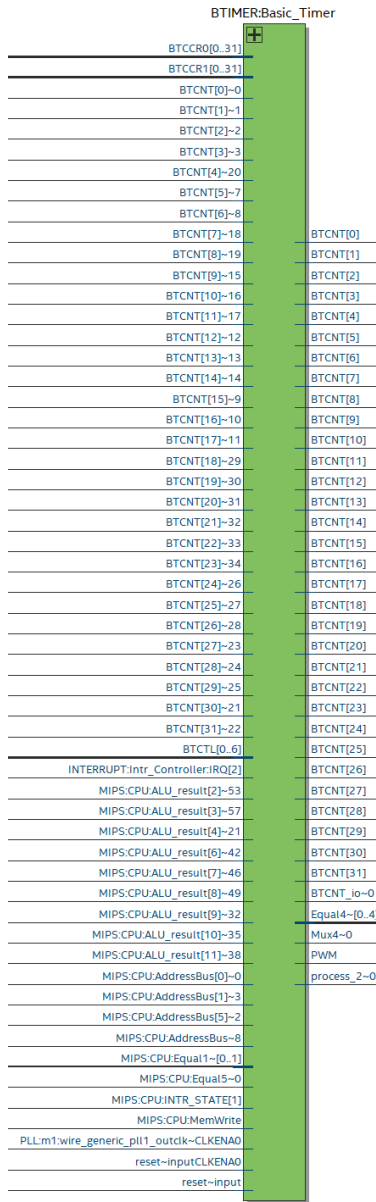
- רכיב חומרה זה מכיל בין היתר רגיסטר בקרה BTCTL הקובע את תכונות השעון ורגיסטר BTCNT שהוא מייצג את ערך הספירה הנוכחי של הטיימר. כמו כן, קיים ברגיסטר הבקרה ערך BTIP שלפיו הרכיב מעלה דגל כאשר ערך רגיסטר הספירה מגיע לערך מתאים. ניתן להשתמש בדגל זה כדי להפריע לתוכנית הראשית של המיקרו בקר ולבצע ISR מתאים.

BTCTL, Basic Timer Control Register



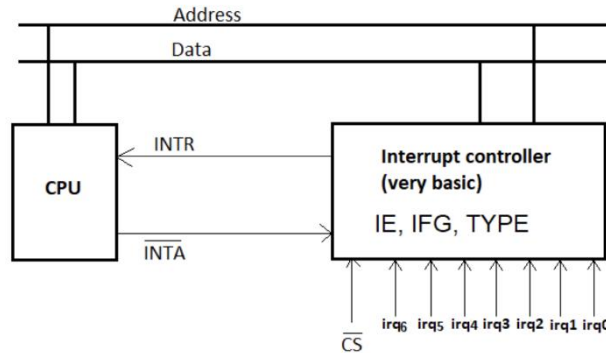
- ייצור אות PWM באמצעות הטיימר הבסיסי הזה כך שהטיימר סופר ובאמצעות CCRx ניתן לשלוט על הזמן מחזור של האות ועל Duty Cycle שלו באמצעות רגיסטרים CCR0 וCCR1 בהתאמה.

דיאגרמת RTL כללית במודול זה:



Interrupt Controller

פסיקה מתבצעת כאשר אחד מרכיבי החומרה השונים מבקשים לבצע פסיקה אזי הם מעלים סיגנל בשם IRQ שמבקש לבצע פסיקה, signal בשם intrsrc מכיל את המידע על הרכיבים המבקשים לקפוץ לרוטינת הפסיקה שלהם. פסיקה מתרחשת כאשר קיים אפשרות גלובאלי לפסיקות (GIE) וגם אין פסיקה מעדיפות גבוהה יותר לפניה.



דיאגרמת הRTL של רכיב זה:

INTERRUPT Intr. Controller	
BTONT0-10	
BTONT0-11	
BTONT0-12	
BTONT0-13	
BTONT0-14	
BTONT0-15	
BTONT0-16	
BTONT0-17	
BTONT0-18	
BTONT0-19	
BTONT0-20	
BTONT0-21	
BTONT0-22	
BTONT0-23	
BTONT0-24	
BTONT0-25	
BTONT0-26	
BTONT0-27	
BTONT0-28	
BTONT0-29	
BTONT0-30	
BTONT0-31	
BTONT0-32	
BTONT0-33	
BTONT0-34	
BTONT0-35	
BTONT0-36	
BTONT0-37	
BTONT0-38	
BTONT0-39	
BTONT0-40	
BTONT0-41	
BTONT0-42	
BTONT0-43	
BTONT0-44	
BTONT0-45	
BTONT0-46	
BTONT0-47	
BTONT0-48	
BTONT0-49	
BTONT0-50	
BTONT0-51	
BTONT0-52	
BTONT0-53	
BTONT0-54	
BTONT0-55	
BTONT0-56	
BTONT0-57	
BTONT0-58	
BTONT0-59	
BTONT0-60	
BTONT0-61	
BTONT0-62	
BTONT0-63	
BTONT0-64	
BTONT0-65	
BTONT0-66	
BTONT0-67	
BTONT0-68	
BTONT0-69	
BTONT0-70	
BTONT0-71	
BTONT0-72	
BTONT0-73	
BTONT0-74	
BTONT0-75	
BTONT0-76	
BTONT0-77	
BTONT0-78	
BTONT0-79	
BTONT0-80	
BTONT0-81	
BTONT0-82	
BTONT0-83	
BTONT0-84	
BTONT0-85	
BTONT0-86	
BTONT0-87	
BTONT0-88	
BTONT0-89	
BTONT0-90	
BTONT0-91	
BTONT0-92	
BTONT0-93	
BTONT0-94	
BTONT0-95	
BTONT0-96	
BTONT0-97	
BTONT0-98	
BTONT0-99	
BTONT0-100	
BTONT0-101	
BTONT0-102	
BTONT0-103	
BTONT0-104	
BTONT0-105	
BTONT0-106	
BTONT0-107	
BTONT0-108	
BTONT0-109	
BTONT0-110	
BTONT0-111	
BTONT0-112	
BTONT0-113	
BTONT0-114	
BTONT0-115	
BTONT0-116	
BTONT0-117	
BTONT0-118	
BTONT0-119	
BTONT0-120	
BTONT0-121	
BTONT0-122	
BTONT0-123	
BTONT0-124	
BTONT0-125	
BTONT0-126	
BTONT0-127	
BTONT0-128	
BTONT0-129	
BTONT0-130	
BTONT0-131	
BTONT0-132	
BTONT0-133	
BTONT0-134	
BTONT0-135	
BTONT0-136	
BTONT0-137	
BTONT0-138	
BTONT0-139	
BTONT0-140	
BTONT0-141	
BTONT0-142	
BTONT0-143	
BTONT0-144	
BTONT0-145	
BTONT0-146	
BTONT0-147	
BTONT0-148	
BTONT0-149	
BTONT0-150	
BTONT0-151	
BTONT0-152	
BTONT0-153	
BTONT0-154	
BTONT0-155	
BTONT0-156	
BTONT0-157	
BTONT0-158	
BTONT0-159	
BTONT0-160	
BTONT0-161	
BTONT0-162	
BTONT0-163	
BTONT0-164	
BTONT0-165	
BTONT0-166	
BTONT0-167	
BTONT0-168	
BTONT0-169	
BTONT0-170	
BTONT0-171	
BTONT0-172	
BTONT0-173	
BTONT0-174	
BTONT0-175	
BTONT0-176	
BTONT0-177	
BTONT0-178	
BTONT0-179	
BTONT0-180	
BTONT0-181	
BTONT0-182	
BTONT0-183	
BTONT0-184	
BTONT0-185	
BTONT0-186	
BTONT0-187	
BTONT0-188	
BTONT0-189	
BTONT0-190	
BTONT0-191	
BTONT0-192	
BTONT0-193	
BTONT0-194	
BTONT0-195	
BTONT0-196	
BTONT0-197	
BTONT0-198	
BTONT0-199	
BTONT0-200	
BTONT0-201	
BTONT0-202	
BTONT0-203	
BTONT0-204	
BTONT0-205	
BTONT0-206	
BTONT0-207	
BTONT0-208	
BTONT0-209	
BTONT0-210	
BTONT0-211	
BTONT0-212	
BTONT0-213	
BTONT0-214	
BTONT0-215	
BTONT0-216	
BTONT0-217	
BTONT0-218	
BTONT0-219	
BTONT0-220	
BTONT0-221	
BTONT0-222	
BTONT0-223	
BTONT0-224	
BTONT0-225	
BTONT0-226	
BTONT0-227	
BTONT0-228	
BTONT0-229	
BTONT0-230	
BTONT0-231	
BTONT0-232	
BTONT0-233	
BTONT0-234	
BTONT0-235	
BTONT0-236	
BTONT0-237	
BTONT0-238	
BTONT0-239	
BTONT0-240	
BTONT0-241	
BTONT0-242	
BTONT0-243	
BTONT0-244	
BTONT0-245	
BTONT0-246	
BTONT0-247	
BTONT0-248	
BTONT0-249	
BTONT0-250	
BTONT0-251	
BTONT0-252	
BTONT0-253	
BTONT0-254	
BTONT0-255	
BTONT0-256	
BTONT0-257	
BTONT0-258	
BTONT0-259	
BTONT0-260	
BTONT0-261	
BTONT0-262	
BTONT0-263	
BTONT0-264	
BTONT0-265	
BTONT0-266	
BTONT0-267	
BTONT0-268	
BTONT0-269	
BTONT0-270	
BTONT0-271	
BTONT0-272	
BTONT0-273	
BTONT0-274	
BTONT0-275	
BTONT0-276	
BTONT0-277	
BTONT0-278	
BTONT0-279	
BTONT0-280	
BTONT0-281	
BTONT0-282	
BTONT0-283	
BTONT0-284	
BTONT0-285	
BTONT0-286	
BTONT0-287	
BTONT0-288	
BTONT0-289	
BTONT0-290	
BTONT0-291	
BTONT0-292	
BTONT0-293	
BTONT0-294	
BTONT0-295	
BTONT0-296	
BTONT0-297	
BTONT0-298	
BTONT0-299	
BTONT0-300	
BTONT0-301	
BTONT0-302	
BTONT0-303	
BTONT0-304	
BTONT0-305	
BTONT0-306	
BTONT0-307	
BTONT0-308	
BTONT0-309	
BTONT0-310	
BTONT0-311	
BTONT0-312	
BTONT0-313	
BTONT0-314	
BTONT0-315	
BTONT0-316	
BTONT0-317	
BTONT0-318	
BTONT0-319	
BTONT0-320	
BTONT0-321	
BTONT0-322	
BTONT0-323	
BTONT0-324	
BTONT0-325	
BTONT0-326	
BTONT0-327	
BTONT0-328	
BTONT0-329	
BTONT0-330	
BTONT0-331	
BTONT0-332	
BTONT0-333	
BTONT0-334	
BTONT0-335	
BTONT0-336	
BTONT0-337	
BTONT0-338	
BTONT0-339	
BTONT0-340	
BTONT0-341	
BTONT0-342	
BTONT0-343	
BTONT0-344	
BTONT0-345	
BTONT0-346	
BTONT0-347	
BTONT0-348	
BTONT0-349	
BTONT0-350	
BTONT0-351	
BTONT0-352	
BTONT0-353	
BTONT0-354	
BTONT0-355	
BTONT0-356	
BTONT0-357	
BTONT0-358	
BTONT0-359	
BTONT0-360	
BTONT0-361	
BTONT0-362	
BTONT0-363	
BTONT0-364	
BTONT0-365	
BTONT0-366	
BTONT0-367	
BTONT0-368	
BTONT0-369	
BTONT0-370	
BTONT0-371	
BTONT0-372	
BTONT0-373	
BTONT0-374	
BTONT0-375	
BTONT0-376	
BTONT0-377	
BTONT0-378	
BTONT0-379	
BTONT0-380	
BTONT0-381	
BTONT0-382	
BTONT0-383	
BTONT0-384	
BTONT0-385	
BTONT0-386	
BTONT0-387	
BTONT0-388	
BTONT0-389	
BTONT0-390	
BTONT0-391	
BTONT0-392	
BTONT0-393	
BTONT0-394	
BTONT0-395	
BTONT0-396	
BTONT0-397	
BTONT0-398	
BTONT0-399	
BTONT0-400	
BTONT0-401	
BTONT0-402	
BTONT0-403	
BTONT0-404	
BTONT0-405	
BTONT0-406	
BTONT0-407	
BTONT0-408	
BTONT0-409	
BTONT0-410	
BTONT0-411	
BTONT0-412	
BTONT0-413	
BTONT0-414	
BTONT0-415	
BTONT0-416	
BTONT0-417	
BTONT0-418	
BTONT0-419	
BTONT0-420	
BTONT0-421	
BTONT0-422	
BTONT0-423	
BTONT0-424	
BTONT0-425	
BTONT0-426	
BTONT0-427	
BTONT0-428	
BTONT0-429	
BTONT0-430	
BTONT0-431	
BTONT0-432	
BTONT0-433	
BTONT0-434	
BTONT0-435	
BTONT0-436	
BTONT0-437	
BTONT0-438	
BTONT0-439	
BTONT0-440	
BTONT0-441	
BTONT0-442	
BTONT0-443	
BTONT0-444	
BTONT0-445	
BTONT0-446	
BTONT0-447	
BTONT0-448	
BTONT0-449	
BTONT0-450	
BTONT0-451	
BTONT0-452	
BTONT0-453	
BTONT0-454	
BTONT0-455	
BTONT0-456	
BTONT0-457	
BTONT0-458	
BTONT0-459	
BTONT0-460	
BTONT0-461	
BTONT0-462	
BTONT0-463	
BTONT0-464	
BTONT0-465	
BTONT0-466	
BTONT0-467	
BTONT0-468	
BTONT0-469	
BTONT0-470	
BTONT0-471	
BTONT0-472	
BTONT0-473	
BTONT0-474	
BTONT0-475	
BTONT0-476	
BTONT0-477	
BTONT0-478	
BTONT0-479	
BTONT0-480	
BTONT0-481	
BTONT0-482	
BTONT0-483	
BTONT0-484	
BTONT0-485	
BTONT0-486	
BTONT0-487	
BTONT0-488	
BTONT0-489	
BTONT0-490	
BTONT0-491	
BTONT0-492	
BTONT0-493	
BTONT0-494	
BTONT0-495	
BTONT0-496	
BTONT0-497	
BTONT0-498	
BTONT0-499	
BTONT0-500	
BTONT0-501	
BTONT0-502	
BTONT0-503	
BTONT0-504	
BTONT0-505	
BTONT0-506	
BTONT0-507	
BTONT0-508	
BTONT0-509	
BTONT0-510	
BTONT0-511	
BTONT0-512	
BTONT0-513	
BTONT0-514	
BTONT0-515	
BTONT0-516	
BTONT0-517	
BTONT0-518	
BTONT0-519	
BTONT0-520	
BTONT0-521	
BTONT0-522	
BTONT0-523	
BTONT0-524	
BTONT0-525	
BTONT0-526	
BTONT0-527	
BTONT0-528	
BTONT0-529	
BTONT0-530	
BTONT0-531	
BTONT0-532	
BTONT0-533	
BTONT0-534	
BTONT0-535	
BTONT0-536	
BTONT0-537	
BTONT0-538	
BTONT0-539	
BTONT0-540	
BTONT0-541	
BTONT0-542	
BTONT0-543	
BTONT0-544	
BTONT0-545	
BTONT0-546	
BTONT0-547	
BTONT0-548	
BTONT0-549	
BTONT0-550	
BTONT0-551	
BTONT0-552	
BTONT0-553	
BTONT0-554	
BTONT0-555	
BTONT0-556	
BTONT0-557	
BTONT0-558	
BTONT0-559	
BTONT0-560	
BTONT0-561	
BTONT0-562	
BTONT0-563	
BTONT0-564	
BTONT0-565	
BTONT0-566	
BTONT0-567	
BTONT0-568	
BTONT0-569	
BTONT0-570	
BTONT0-571	
BTONT0-572	
BTONT0-573	
BTONT0-574	
BTONT0-575	
BTONT0-576	
BTONT0-577	
BTONT0-578	
BTONT0-579	
BTONT0-580	
BTONT0-581	
BTONT0-582	

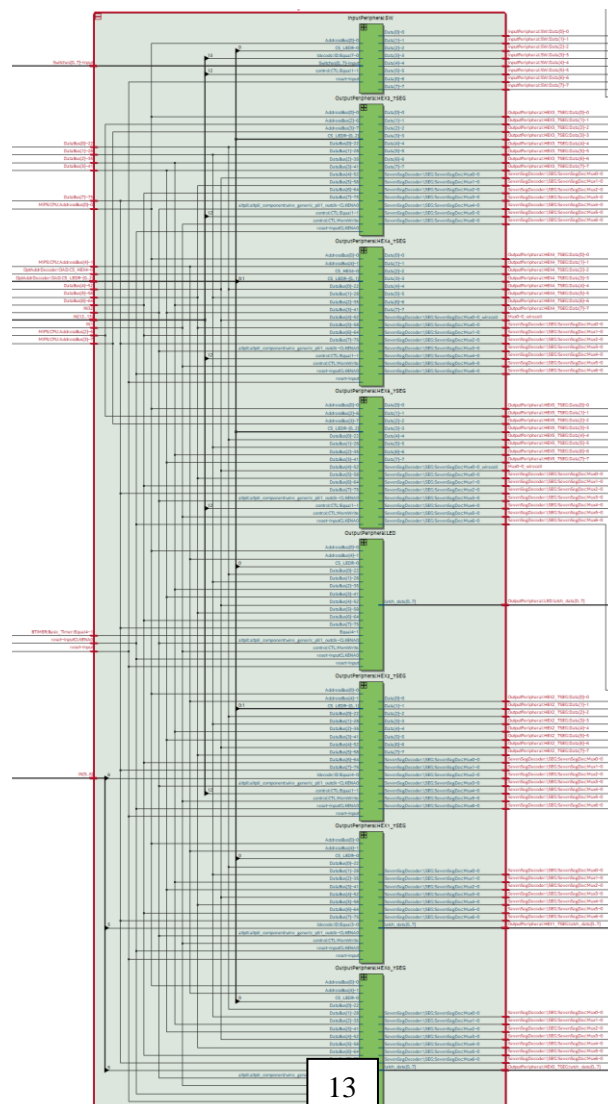
GPIO

ממש אשר אחראי על התמיכה ברכיבי IO השונים שנמצאים על גבי שבב הFPGA, רכיבי IO הם למעשה Memory Mapped IO, להלן הכתובות של הרכיבים בזכרון:

*GPIO
without
interrupt
capability*

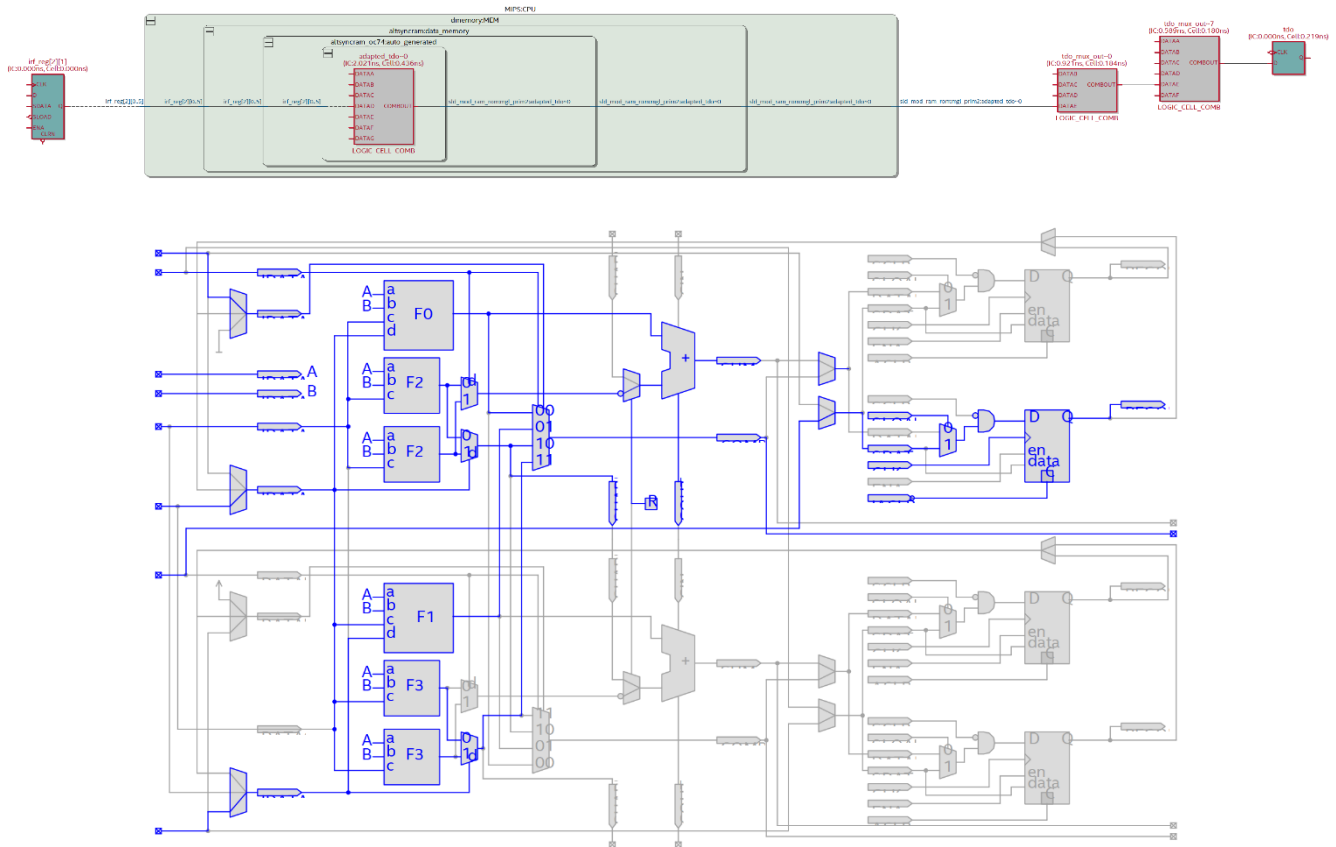
```
#-----
#          MEMORY Mapped I/O
#-----
#define PORT_LEDR[7-0] 0x800 - LSB byte (Output Mode)
#----- PORT_HEX0_HEX1 -----
#define PORT_HEX0[7-0] 0x804 - LSB byte (Output Mode)
#define PORT_HEX1[7-0] 0x805 - LSB byte (Output Mode)
#----- PORT_HEX2_HEX3 -----
#define PORT_HEX2[7-0] 0x808 - LSB byte (Output Mode)
#define PORT_HEX3[7-0] 0x809 - LSB byte (Output Mode)
#----- PORT_HEX4_HEX5 -----
#define PORT_HEX4[7-0] 0x80C - LSB byte (Output Mode)
#define PORT_HEX5[7-0] 0x80D - LSB byte (Output Mode)
#-----
#define PORT_SW[7-0] 0x810 - LSB byte (Input Mode)
#-----
#define PORT_KEY[3-1] 0x814 - LSB nibble (3 push-buttons - Input Mode)
#-----
```

למעשה, בעת לחיצה על אחד הלחצנים, רכיב חומרה זה אחראי ע"י קפיצה לרושנית פסיקה של הלחצנים להעביר מידע לאחד מרכיבים אלו או קבלת מידע מרכיבי קלט כגון מתגים.



נתיב קריטי

הנתיב הקריטי הוא הנתיב הארוך ביותר של התפשטות הלוגיקה בתוך המעגל הדיגיטלי שבנינו. הוא מייצג את הנתיב האיטי ביותר במערכת, כך שנצטרך לקבוע לפיו את תדר השעון המירבי שניתן להשיג עבור תכנון הMCU. הנתיב הקריטי בפרויקט זה:



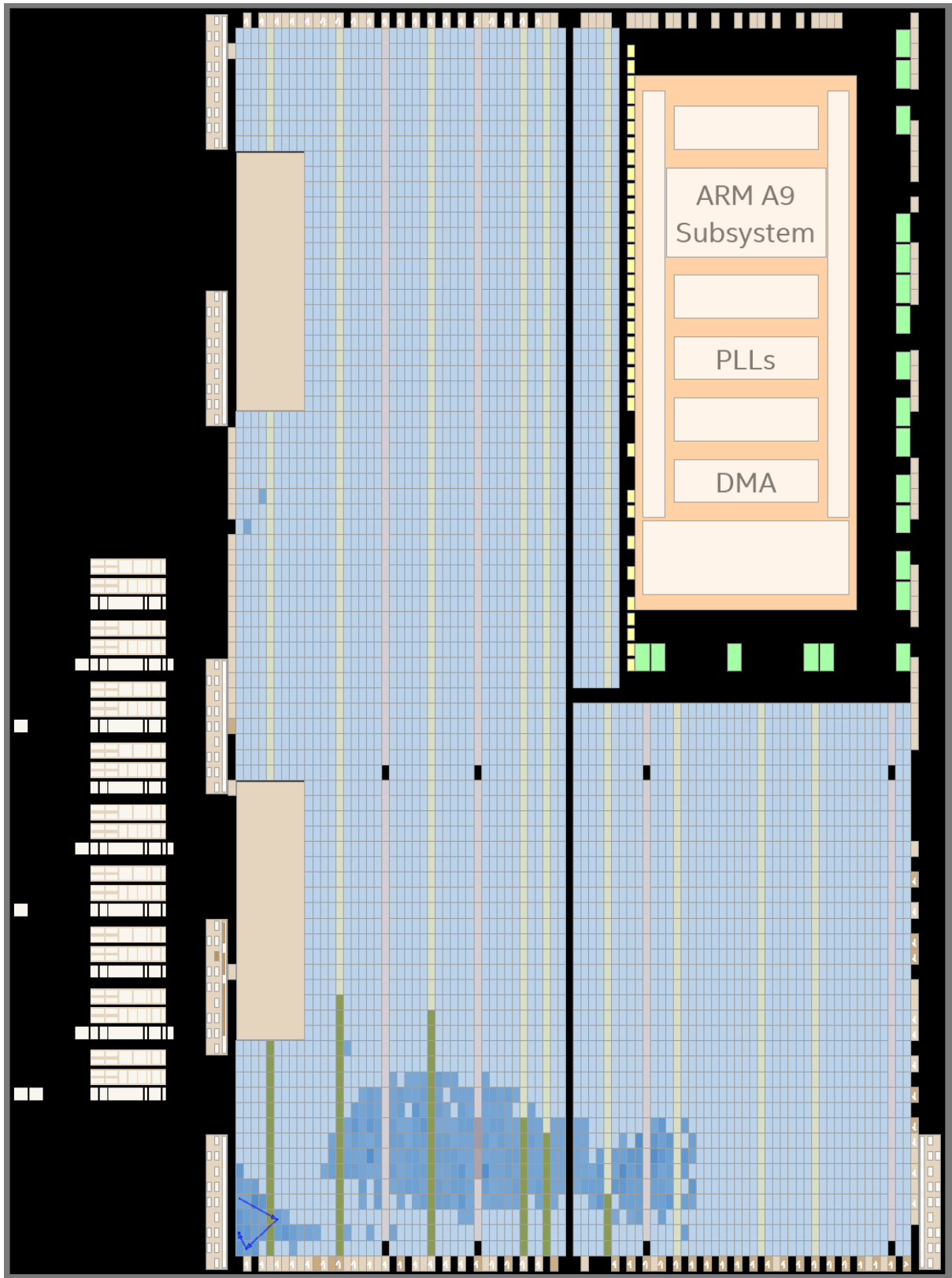
בנוסף על כך, מצאנו את תדר השעון המקסימלי עבור המודל שבנינו והוא:

Slow 1100mV OC Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	91.92 MHz	91.92 MHz	alter...d_tck	

הנתיב הקריטי היכן שממוקם על גבי השבב הינו:



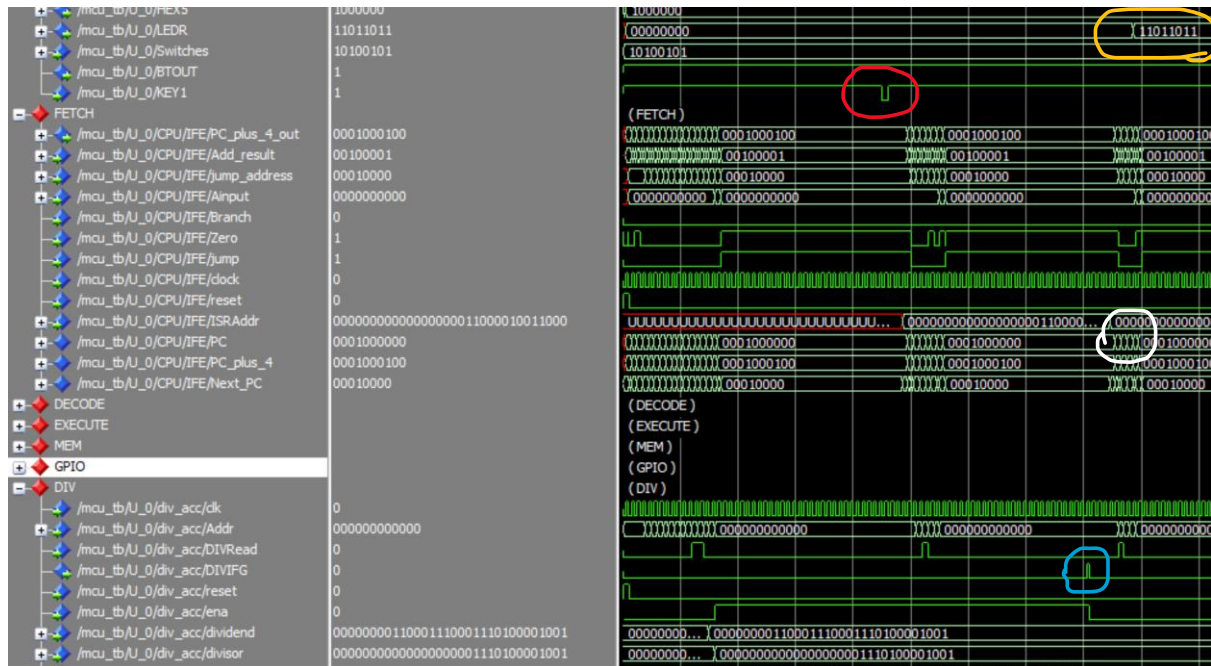
ניתוח תוצאות

בפרק זה נדון בתוצאות המערכת שלנו גם מבחינת ניתוח הזמנים, גם מבחינת סימולציות ותיעוד הגלים בFPGA באמצעות Signal Tap.

ניתוח גלים בModel SIM

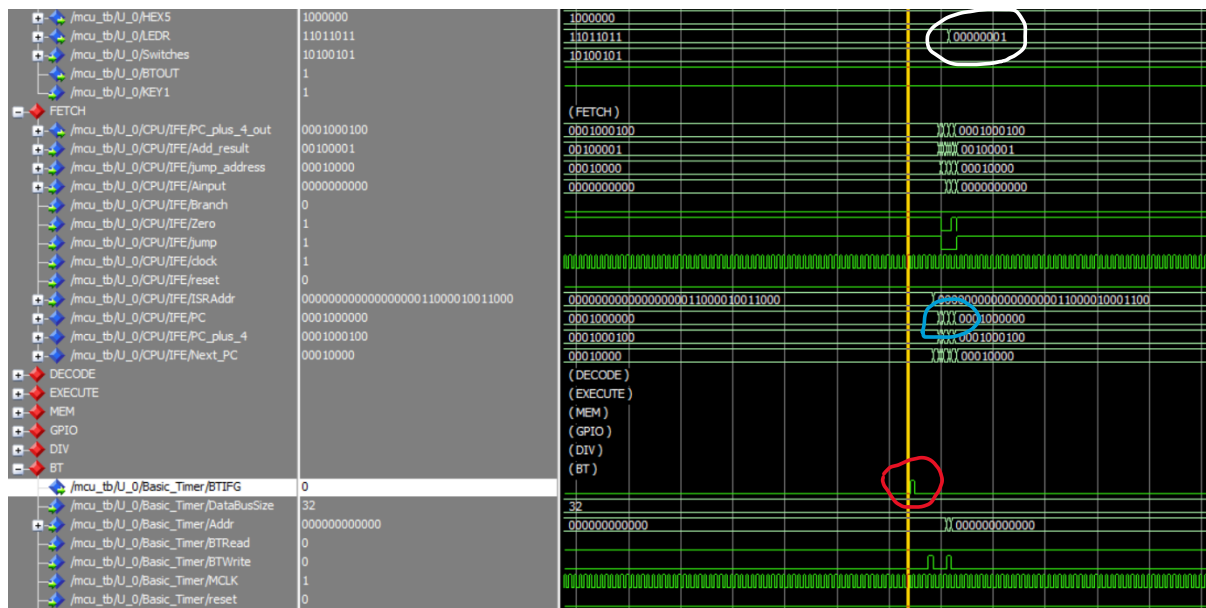
בחלק זה אציג את פעולת הרכיבים כפי שנראים בסימולציית הגלים:

להלן הדוגמא הראשונה:



- בסימון האדום ניתן לראות לחיצה על לחצן מספר 1.
- לחצן אחד טוען שני מספרים למחלק הבינארי (מספר מחלק ומספר מחולק), לאחר 32 מחזורי שעון ניתן לראות בסימון הכחול כי רכיב החומרה מעלה דגל המסמל שהתוצאות זמינות כעת.
- בסימון הלבן ניתן לראות את השינוי בPC כתוצאה מקפיצה לISR של המחלק.
- בסימון הצהוב ניתן לראות את תוצאת החלוקה נכתבת לLEDS.

להלן הדוגמא השנייה:



- בסימון האדום ניתן לראות בקשת ה BT לפסיקה בעקבות ערך המנייה אשר הגיעה לערך המתאים לפי BTIP .
- בסימון הכחול ניתן לראות את רגיסטר ה PC משתנה לכתובת ה ISR של השעון החומרתי.
- בסימון הלבן ניתן לראות את ערך הלדים מעודכן לפי ערך המנייה (בבדיקה זו הקוד מבצע מנייה כלפי מעלה על גבי הלדים ע"י פסיקת השעון החומרתי).

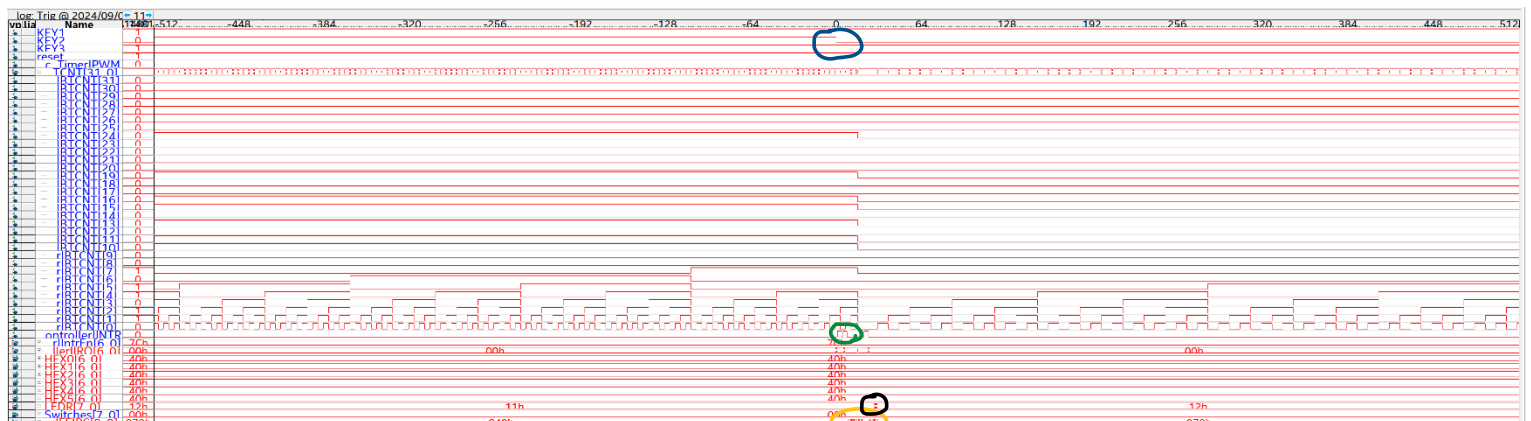
ניתוח גלים Signal Tap

ראשית אציג את רשימת האותות אשר בחרתי לבחון תחת בדיקה זו:

trigger: 2024/09/03 1		Lock mode:		Allow	
Node	Name	Enabler	Enabler	Enabler	Condi
KEY1		✓	✓	✓	✓
KEY2		✓	✓	✓	✓
KEY3		✓	✓	✓	✓
reset		✓	✓	✓	✓
c_TimerIPWM		✓	✓	✓	✓
TCNTI31 01		✓	✓	✓	XXXXh
ontrollerINT		✓	✓	✓	✓
rlIntrEnI6 01		✓	✓	✓	XXh (OR
llrIIROI6 01		✓	✓	✓	XXh (OR
HEX0I6 01		✓	✓	✓	XXh (OR
HEX1I6 01		✓	✓	✓	XXh (OR
HEX2I6 01		✓	✓	✓	XXh (OR
HEX3I6 01		✓	✓	✓	XXh (OR
HEX4I6 01		✓	✓	✓	XXh (OR
HEX5I6 01		✓	✓	✓	XXh (OR
LDRI7 01		✓	✓	✓	XXh (OR
SwitchesI7 01		✓	✓	✓	XXh (OR
IEFIPCI9 01		✓	✓	✓	XXh (OR

נשים לב כי ארבעת הלחצנים מקבלים טריגר ב falling edge ב Basic Or.

הקוד הצרוב על הבקר בעת הבדיקה מבצע מנייה כלפי מעלה ע"י פסיקת שעון, להלן תוצאות הדוגמא הראשונה:

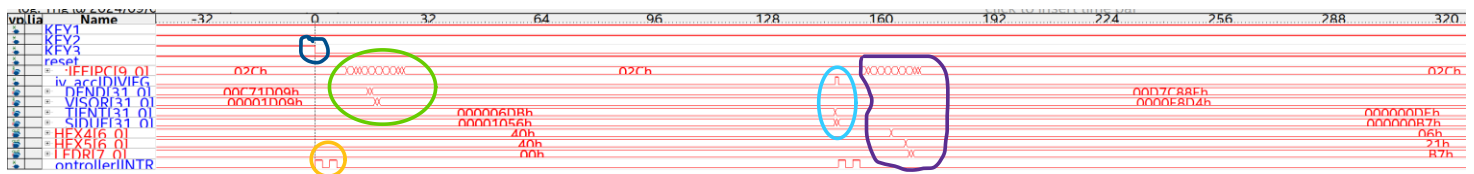


- בסימון הכחול ניתן לראות את הטריגר של לחיצה על לחצן מספר 2.
- בסימון הירוק ניתן לראות את בקשת הפסיקה מועברת לרכיב int controller על מנת לבצע רוטינת פסיקה של הלחצן.
- בסימון הצהוב ניתן לראות את ביצוע רוטינות השירות של הלחצן והשעון.
- בסימון השחור ניתן לראות לאחר פסיקת השעון את ערך המניה על גבי הלדים עולה ב1.

```
On KEY3 pushing:
-----
loading DIVIDEND with 8-bit value of y=0x00D7C88F=14,141,583
loading DIVISOR with 8-bit value of x=0x0000F8D4=63,700
Note: y/x={QUOTIENT=0xDE=222 ,RESIDUE=0xB7=183}

On Divider Accelerator interrupt:
-----
write QUOTIENT low nibble to PORT_HEX4[7-0]
write QUOTIENT high nibble to PORT_HEX5[7-0]
write RESIDUE to PORT_LEDR[7-0]
```

תוצאות הסימולציה:



- בסימון הכחול ניתן לראות לחיצה על לחצן מספר 3.
- בסימון הצהוב מועברת ל int controller בקשת פסיקה מהלחצן.
- בסימון הירוק ניתן לראות ביצוע רוטינת השירות של לחצן 3, במהלכה נטענים ערכי המחלק והמחולק עם הערכים שציינתי מעלה.
- בסימון התכלת, מאיץ החומרה מסיים את פעולת החילוק לאחר 32 מחזורי שעון ומעלה דגל בקשת פסיקה המועבר ל int controller.
- בסימון הסגול, רואים קפיצה לרוטינת השירות של מאיץ החומרה, במהלכה תוצאות החילוק נטענים לרכיבי ה IO המתאימים, ניתן לראות שתוצאות החישוב תואמות למצופה כפי שכתבתי מעלה.