Nios® II and the Interval Timers' Alarm and Timestamp Functionality on the Intel® MAX® 10-10M08 Evaluation Kit

By Sean D. Liming and John R. Malin Annabooks, LLC. – <u>www.annabooks.com</u>

January 2023

Timing is everything. The Interval Time IP that comes with Quartus not only provides the date and time for a Nios II processor but also supports alarms and timestamp functionality. The paper walks through a couple of applications that test both.

Please see the article Intel® Quartus® Prime Lite and Nios® II SBT for Eclipse Installation Instructions on Annabooks.com to install the software needed for this hands-on exercise.

The Project Requirements:

- Intel Quartus Prime Lite Edition V21.0 and Nios® II SBT for Eclipse are already installed.
- Intel® MAX® 10 10M08 Evaluation Kit and the schematic for the evaluation board are required. The schematic PDF file can be downloaded from the Intel FPGA website.
- Intel FPGA Programming cable USB Blaster II or EthernetBlaster II. The Intel® MAX® 10 - 10M08 Evaluation Kit doesn't have a built-in USB Blaster II onboard.
- <u>Intel® Quartus® Prime Lite and NIOS® II SBT for Eclipse Installation Instructions</u> on Annabooks.com

Note: There are equivalent MAX 10 development and evaluation boards available. These boards can also be used as the target, but you will have to adjust to the available features on the board. Please make sure that you have the board's schematic files as these will be needed to identify pins.

1.1 Nios II Timer Project

The custom MCU will comprise the following IP blocks:

- Nios II processor
- Onchip RAM
- Interval Timer
- Parallel IO for LEDs
- Sys ID
- JTAG UART

1.1.1 Create the Project

The first step is to create a design project.

- 1. Open Quartus
- 2. Click on the New Project Wizard



- 3. Click Next to the Introduction dialog
- 4. Select or create a project directory \NIOS2_Timer (Do not use the Quartus installation directory) and name of the project: "NIOS2timer". Click Next.

Note: By default, the root directory is the Quartus installation directory. Make sure the root project directory is a separate path from the Quartus installation files. Also, there can be no spaces in the name of the folders or projects.

- 5. Project Type: Empty project, click Next
- 6. Add File no files to add, click Next.
- 7. Family, Device & Board Settings, click the Board tab and select: MAX 10 FPGA 10M08 Evaluation Kit, and click Next.

Family, Device & Board Device Board Select the board/development kit you want to target for compilation. Family: MAX 10 Development Kit: Any	New	Project Wizard					
Device Board Select the board/development kit you want to target for compilation. Family: MAX 10 Yealiable boards: Agailable boards: Arrow MAX 10 DECA 0.9 MAX 10 10M50DAF484C6GES Arrow MAX 10 DECA 0.9 MAX 10 10M08DAF484C6GES Arrow MAX 10 FPGA Evaluation Kit 1.0 MAX 10 DE10 - Lite 1.0 MAX 10 PFGA 10M08 Evaluation Kit 1.0 MAX 10 NEEK 1.0 MAX 10 IOM50DAF484C6GES Altera V Create top-level design file. Can't find your board? Check the Design Store for additions and search for baseline under Design Examples.	ami	ly, Device & Board Settings					
Select the board/development kit you want to target for compilation. Family MAX 10 Pevelopment Kit Any Agailable boards: Image: Colspan="2">Image: Colspan="2">Version Family Device Vendor Image: Colspan="2">Arrow MAX 10 DECA 0.9 MAX 10 10M050AF484C6GES Arrow 497 Image: Colspan="2">MAX 10 FPGA Evaluation Kit 1.0 MAX 10 10M080AF484C6GES Altera 497 Image: Colspan="2">MAX 10 FPGA 10M08 Evaluation Kit 1.0 MAX 10 10M080AF484C6GES Altera 497 Image: Colspan="2">MAX 10 FPGA 10M08 Evaluation Kit 1.0 MAX 10 10M080AF484C6GES Altera 497 Image: Colspan="2">MAX 10 FPGA ToM08 Evaluation Kit 1.0 MAX 10 10M080AF484C6GES Altera 497 Image: Colspan= ToMAX 10 FPGA Kit 1.0 MAX 10 10M080AF484C6GES Altera 497 Image: Colspan= ToMAX 10 FPGA Kit	Devi	ce Board					
Family: MAX 10 Development Kit: Any Agailable boards: Arrow MAX 10 DECA 0.9 MAX 10 10M50DAF484C6GES Arrow 497 BeMicro MAX 10 FPGA Evaluation Kit 1.0 MAX 10 10M08DAF484C6GES Arrow 806 MAX 10 DE10 - Lite 1.0 MAX 10 10M08SAE144C8GES Altera 497 MAX 10 FPGA 10M08 Evaluation Kit 1.0 MAX 10 10M08SAE144C8GES Altera 497 MAX 10 FPGA Development Kit 1.0 MAX 10 10M50DAF484C6GES Altera 497 MAX 10 FPGA Development Kit 1.0 MAX 10 10M08SAE144C8GES Altera 497 MAX 10 FPGA Development Kit 1.0 MAX 10 10M50DAF256C7G Altera 497 idvalue to the the the theoremotion of the theo	Selec	t the board/development kit you want to targ	et for compila	ation.			
Family: MAX 10 Device Vendor Agailable boards: Image: State of the state of				Development Kite	A		
Name Version Family Device Vendor Image: Arrow MAX 10 DECA 0.9 MAX 10 10M50DAF484C6GES Arrow 497 Image: BeMicro MAX 10 FPGA Evaluation Kit 1.0 MAX 10 10M08DAF484C6GES Arrow 806 Image: MAX 10 DE10 - Lite 1.0 MAX 10 10M08DAF484C6GES Altera 497 Image: MAX 10 FPGA 10M08 Evaluation Kit 1.0 MAX 10 10M085AE144C8GES Altera 497 Image: MAX 10 FPGA Development Kit 1.0 MAX 10 10M085AE144C8GES Altera 497 Image: MAX 10 FPGA Development Kit 1.0 MAX 10 10M085AE144C8GES Altera 497 Image: MAX 10 FPGA Development Kit 1.0 MAX 10 10M085AE144C8GES Altera 497 Image: MAX 10 FPGA Development Kit 1.0 MAX 10 10M50DAF48417G Terasic 497 Image: MAX 10 FPGA Kit 1.0 MAX 10 10M085AU169C8GES Macnica 806 Image: MAX 10 FPGA Kit 1.0 MAX 10 10M085AU169C8GES Macnica 806 Image: MAX 10 FPGA Kit 1.0 MAX 10 10M085AU16	Fami Avail	able boards:	· ·	Development Kit:	Any		•
Image: Second	. <u>.</u>	Name	Vers	ion Fam	ily Device	Vendor	
Image: Selection of the se	-	Arrow MAX 10 DECA	0.9	MAX 10	10M50DAF484C6GES	Arrow	4976
Image: Max 10 DE10 - Lite 1.0 Max 10 10M50DAF484C6GES Altera 497 Image: Max 10 FPGA 10M08 Evaluation Kit 1.0 MAX 10 10M08SAE144C8GES Altera 806 Image: Max 10 FPGA Development Kit 1.0 MAX 10 10M50DAF256C7G Altera 497 Image: Max 10 NEEK 1.0 MAX 10 10M50DAF484I7G Terasic 497 Image: Odyssey MAX 10 FPGA Kit 1.0 MAX 10 10M50DAF484I7G Terasic 497 Image: Odyssey MAX 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey MAX 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey MAX 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey MAX 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey MAX 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey MAX 10 FPGA Kit 1.0 Max 10 10M08SAU169C8GES Macnica 806 Image: Odyssey MAX 10 FP		BeMicro MAX 10 FPGA Evaluation Kit	1.0	MAX 10	10M08DAF484C8GE5	Arrow	8064
Image: Max 10 FPGA 10M08 Evaluation Kit 1.0 MAX 10 10M08SAE144C8GES Altera 806 Image: MAX 10 FPGA Development Kit 1.0 MAX 10 10M50DAF256C7G Altera 497 Image: MAX 10 NEEK 1.0 MAX 10 10M50DAF256C7G Altera 497 Image: MAX 10 NEEK 1.0 MAX 10 10M50DAF48417G Terasic 497 Image: Max 10 NEEK 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Max 10 FPGA term Image: Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Max 10 FPGA term Image: Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Max 10 FPGA term Image: Max 10 FPGA Kit 1.0 Max 10 10M08SAU169C8GES Macnica 806 Image: Max 10 FPGA term Image: Max 10 FPGA Kit 1.0 Max 10 FPGA Kit Image: Max 10 FPGA Kit Image: Max		MAX 10 DE10 - Lite	1.0	MAX 10	10M50DAF484C6GES	Altera	4976
Image: Max 10 FPGA Development Kit 1.0 MAX 10 10M50DAF256C7G Altera 497 Image: Max 10 NEEK 1.0 MAX 10 10M50DAF48417G Terasic 497 Image: Odyssey MAX 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey Max 10 FPGA Kit 1.0 Max 10 10M08SAU169C8GES Macnica 806 Image: Odyssey Max 10 FPGA Kit 1.0 Max 10 10M08SAU169C8GES Macnica 10 Image: Odyssey Max 10 FPGA Kit 1.0 Image: Odyssey Max 10 Image: Odyssey Max 10 Image: Odysey Max 10 Image: Odyssey Max 10<	-	MAX 10 FPGA 10M08 Evaluation Kit	1.0	MAX 10	10M08SAE144C8GES	Altera	8064
Image: Max 10 NEEK 1.0 MAX 10 10M50DAF48417G Terasic 497 Image: Odyssey MAX 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey MAX 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Odyssey Max 10 FPGA Kit Image: Odyssey Max 10 Image: Odysey Max 10 Image: Odyssey Max 10		MAX 10 FPGA Development Kit	1.0	MAX 10	10M50DAF256C7G	Altera	4976
Image: Max 10 FPGA Kit 1.0 MAX 10 10M08SAU169C8GES Macnica 806 Image: Macnica Macnic		MAX 10 NEEK	1.0	MAX 10	10M50DAF484I7G	Terasic	4976
 ✓ Create top-level design file. Can't find your board? Check the <u>Design Store</u> for additions and search for baseline under Design Examples. 		Odyssey MAX 10 FPGA Kit	1.0	MAX 10	10M08SAU169C8GE5	Macnica	8064
 ✓ Create top-level design file. Can't find your board? Check the <u>Design Store</u> for additions and search for baseline under Design Examples. 							
✓ Create top-level design file. Can't find your board? Check the <u>Design Store</u> for additions and search for baseline under Design Examples.	4						Þ
Can't find your board? Check the <u>Design Store</u> for additions and search for baseline under Design Examples.	√ c	reate top-level design file.					
	Can't	find your board? Check the <u>Design Store</u> for:	additions and	I search for baseline	under Design Examples.		
					- ,		
Help Cancel Next Sinish Cancel	Hel				< Back Next >	Finish	Concel

- 8. EDA Tools: click Next.
- 9. Summary: click Finish

Note: The actual MAX 10 on our board is the 10M08SAE144C8G, thus it is not an Engineering Sample (ES). The next two steps change the device to the production device. Your experience might be different. These next two optional steps change the device.

- 10. In the project navigation pane on the left, right-click on 10: 10M08SAE144C8GE, and select Device from the context menu.
- 11. In the Available devices, scroll down and select the 10M08SAE144C8G. Click OK.

Device Board							
Select the family ar You can install add To determine the v	d device you want to t tional device support v rsion of the Quartus F	arget for o with the Ir Prime soft	compilation. nstall Devices co ware in which yo	mmand on t our target dev	he Tools m ⁄ice is supp	enu. orted, r	refer to the <u>Device Support List</u> webpag
Device family				Show in 'A	vailable de	vices' li	st
Eamily: MAX 10	(DA/DF/DC/SA/SC/S	L)	•	Package	Pac <u>k</u> age:		÷
Dev <u>i</u> ce: All	•		Pin <u>c</u> ou	Pin <u>c</u> ount:		*	
Target device				Core sp <u>e</u> ed grade:		Any	
<u>A</u> uto device s <u>Specific device</u>	elected by the Fitter e selected in 'Available	e <mark>dev</mark> ices'	list	Name fi	lter: w advanced	device	25
Other: n/a				Device an	d Pin Optic	ons	
A <u>v</u> ailable devices:	-	1000-00		T. marine			
Name	Core Voltage	LES	Total I/Os	GPIOs	Memor	y Bits	Embedded multiplier 9-bit elem
10M085AE144C8	5.3V	8064	101	101	387072		48
10M08SAE144I7G	3.3V	8064	101	101	387072		48
	3 3V	8064	101	101	387072		48
10M0854F144I7P							

1.1.2 Create the Design in Platform Designer

Quartus supports many design types to create an FPGA design. The Platform Designer tool will be used for this hands-on exercise. Platform Designer makes it easy to add already-built IP blocks and interconnect them.

1. From the menu, select Tools->Platform Designer, or the Platform Designer icon from the toolbar.

The Platform Designer tool is launched. By default, a clock (clk_0) is added to the design. Platform Designer makes it easy to add IP blocks and make interconnections between the blocks.

2. The top left pane contains the IP Catalog with all the available IP blocks that come with Quartus Prime. In the search box, type NIOS.

NIOS X	📩 IP Catalog 🛛	- d	
Project		× 🛛	k
Basic Functions Simulation; Debug and Verification Simulation Nios II Custom Instruction Master BFM Intel FPGA II Nios II Custom Instruction Slave BFM Intel FPGA IP Orecessors Orecessors Output Output	Project Mew Comport Library Basic Functions Simulation; D Simulation; D Simulation Simulation Nios II Co Nios II Co Nios II Co Nios II Co Nios II Co 	ent bug and Verification os II Custom Instruction Master BFM Intel FPGA I os II Custom Instruction Slave BFM Intel FPGA IP ripherals stom Instructions tswap ustom Instruction Interconnect ustom Instruction Master Translator ustom Instruction Slave Translator oating Point Hardware oating Point Hardware 2 occessors I Processor	UF ,
< <p>New Edit</p>	Kew Edit	>	

- 3. Expand the Processors and Peripherals and Embedded Processors branches and doubleclick on the Nios II Processor.
- 4. This will open the Nios II Configuration page. The first tab is to select the type of core Nios II/e or Nios II/f. We will keep the defaults for now. Click Finish.

* Nios II Processor - nios2_gen2_0				×
Nios II Processor				
Megeocerer altera_nios2_gen2				Documentation
Block Diagram	Main Vectors	Caches and Memory Interfaces Arithmetic Instru	ctions MMU and MPU Settings JTAG Debug Ad-	vanced Features
Show signals	* Select an Ir	nplementation		
nios2_gen2_0	Nios II Core:	⊖ Nios II/e		
cik data_master_		Nos II/f		
reset reset avien instruction master.		Nios II/e	Nios II/f	
ing interrupt reset debug_reset_request	Summary	Resource-optimized 32-bit RISC	Performance-optimized 32-bit RISC	
debug_mem_slaveavalonnios_oustorm_instructionavaloninstructionavaloninstructionavaloninstructionavaloninstructionavaloninstructionavaloninstructionavaloninstructionavaloninstructionavaloninstructionavaloninstructionavaloninstructionavaloninstructionavaloninstructioninstructioninstructionavaloninstructioninstructioninstructionavaloninstructioninstruct	Features	JTAG Debug	JTAG Debug	
atera_nios2_gen2		ECC RAM Protection	Hardware Multiply/Divide Instruction/Data Caches	
			Tightly-Coupled Masters ECC RAM Protection	
			External Interrupt Controller Shadow Register Sets	
			MPU MMU	
	RAM Usage	2 + Options	2 + Options	
				1
Error: nios2 gen2 0: Instruction Cache is larger than the Instruction Address. Please reduce t	he Instruction Cachi	e Size. Current Tag Size is 0		
Error: nios2_gen2_0: Reset slave is not specified. Please select the reset slave				
Error: nios2_gen2_0: Exception slave is not specified. Please select the exception slave				
				Cancel Finish

- 1. The processor will be added to the design. Right-click on the name nios2_gen2_cpu, and rename it to nios2.
- 2. Now let's add the RAM IP block. In the IP Catalog enter RAM in the search box.
- 3. Double-click on On-chip Memory (RAM or ROM) in the Intel FPGA IP.



4. The configuration page will appear. Change the Total memory size to 16384. We need more memory to run the application.

* Size							
Enable different width for Dual-port a	Enable different width for Dual-port access						
Slave S1 Data width:	Slave S1 Data width: 32 🗸						
Total memory size:	16384	bytes					
Minimize memory block usage (may impact fmax)							
Read latency							

5. Uncheck the box for "Initialize memory content", and click Finish.

Memory initialization	
Initialize memory content	
Enable and default initialization file	
	ê.
Type the filename (e.g: my_ram	e hex) or select the hex file using the file browser button.
Type the filename (e.g: my_ram User created initialization file:	e n.hex) or select the hex file using the file browser button. onchip_mem.hex

- The On-chip Memory (RAM or ROM) in the Intel FPGA IP will be added to the design, rightclick on the name, and rename it to onchip_RAM.
- 7. In the IP Catalog search, enter timer.
- 8. Double-click on the Interval Timer Intel FPGA IP.

📩 IP Catalog 🛛	- ₫	
🔍 timer	×	I.
Project Wew Component Library		
Peripherals Interval Timer Intel FPGA IP		

- 9. Keep the settings as they are and click Finish.
- 10. In the IP Catalog search, enter system ID.
- 11. Double-click on the System ID Peripheral Intel FPGA IP.



Copyright © 2023 Annabooks, LLC. All rights reserved Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation All other copyrighted, registered, and trademarked material remains the property of the respective owners.

- 12. A configuration page will appear. There are no changes to be made. Click Finish.
- 13. In the IP Catalog search, enter uart.
- 14. Double-click on the JTAG UART Intel FPGA IP.

TP Catalog 🛛		
🔍 uart		×
Project 	nent ols 50 Compatible UART Intel FPGA IP <mark>5 UART Intel FPGA IP</mark> T (RS-232 Serial Port) Intel FPGA IP am ions I UART 32 UART	
New		

- 15. A configuration page will appear. There are no changes to be made. Click Finish.
- 16. In the IP Catalog enter pio in the search box.
- 17. Add the PIO (Parallel I/O) Intel FPGA IP to the design.



- In the configuration page, set the Width to 5, leave the Direction as Output, and set the Output Port Reset Value to 0x1f. Since the LEDs are active low, the value turns 5 LEDs off on startup.
- 19. Click Finish.

PIO (Parallel I/O) Intel FPGA IP - pio_0	
PIO (Parallel I/O) Intel FPGA IP	
Block Diagram	
Show signals	Basic settings Width (1-32 bits):
pio_0	Direction: O Bidir
	○ Input
clock	() InOut
reset	Output
s1avalon	Output Port Reset Value: 0.00000000000000000000000000000000000
external_connection	
conduit	Output Register
aitera_avalon_pio	Enable individual bit setting/dearing
	Edge capture register
	Synchronously capture
	Edge Type: RISING V
	Enable bit-dearing for edge capture register
	▼ Interrupt
	Generate IRQ
	IRQ Type:
	Level: Interrupt CPU when any unmasked I/O pin is logic true
	Edge: Interrupt CPU when any unmasked bit in the edge-capture
	register is register a der Available when synchronous capture is enabled

- 20. The PIO will be added to the design. Rename the PIO to pio_0.
- 21. For the PIO exernal_connection, double-click on the Export column and set the value to led5. This will provide a base name for connecting the signals to the PINs on the chip. The connection will be made in PIN Planner.
- 22. Now we need to wire the IP blocks together. The picture below shows all the writing connections for the design.

Rev	1.	.3

× 4	System: NIOS2tim	erMCU Path: dk_0			
Use	Connections	Name	Description	Export	Clo
		⊟ clk_0	Clock Source		
		⊏– dk_in	Clock Input	clk	ex
	•	⊏- dk_in_reset	Reset Input	reset	
		clk	Clock Output	Double-click to export	dk_
		→ dk_reset	Reset Output	Double-click to export	
\checkmark		回 喧 nios2	Nios II Processor		
	♦	→ clk	Clock Input	Double-click to export	clk
	+	→ reset	Reset Input	Double-click to export	[dk
		→ data_master	Avalon Memory Mapped Master	Double-click to export	[clk
		→ instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk
		→ irq	Interrupt Receiver	Double-click to export	[clk
	$ \succ$	debug_reset_requ	est Reset Output	Double-click to export	[clk
		→ debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk
	×	custom_instruction	_m Custom Instruction Master	Double-click to export	
		onchip_RAM	On-Chip Memory (RAM or ROM) Inte	1	
	♦ 	→ dk1	Clock Input	Double-click to export	clk
		→ s1	Avalon Memory Mapped Slave	Double-click to export	[clk
		→ reset1	Reset Input	Double-click to export	[clk
\checkmark		□ timer_0	Interval Timer Intel FPGA IP		
	♦ 	→ clk	Clock Input	Double-click to export	clk
		→ reset	Reset Input	Double-click to export	[clk
		→ s1	Avalon Memory Mapped Slave	Double-click to export	[clk
		— irq	Interrupt Sender	Double-click to export	[clk
		sysid_qsys_0	System ID Peripheral Intel FPGA IP		
	♦ 	→ clk	Clock Input	Double-click to export	clk
	• • • •	→ reset	Reset Input	Double-click to export	[clk
	• •	→ control_slave	Avalon Memory Mapped Slave	Double-click to export	[clk
		🖃 jtag_uart_0	JTAG UART Intel FPGA IP		
		→ dk	Clock Input	Double-click to export	clk
		→ reset	Reset Input	Double-click to export	[clk
		→ avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk
		—≺ irq	Interrupt Sender	Double-click to export	[clk
		🗆 pio_0	PIO (Parallel I/O) Intel FPGA IP		
	•	→ dk	Clock Input	Double-click to export	clk
	• •	→ reset	Reset Input	Double-click to export	[clk
	• •	→ s1	Avalon Memory Mapped Slave	Double-click to export	[clk
		 external_connection 	on Conduit	led5	

23. Let's assign a base address. From the menu, select System->Assign Base Address. This will remove a number of errors from the message box. You will see the base address values for each IP change in the System Contents tab.



- 24. Finally, let's set the reset and exception vector addresses. Double-click on the nios2 to open the configuration page.
- 25. Click on the Vectors tab.
- 26. Change the Reset vector memory drop-down to onchip_RAM.s1.
- 27. Change the Exception vector memory drop-down to onchip_RAM.s1.

Narameters 🛛	- 1							
System: NIOS2adcMCU Path: nios2								
Nios II Processor								
altera nios2 gen2	Deta							
Main Vectors Caches and Memory I	Interfaces Arithmetic Instructions MMU and MPU Settin							
Reset Vector								
Reset vector memory:	onchip_RAM.s1 \sim							
Reset vector offset:	0x0000000							
Reset vector:	0x00004000							
Evention Vector								
Exception vector								
Exception vector memory:	onchip_RAM.s1 v							
Exception vector offset:	0x0000020							
Exception vector:	0x00004020							
East TIP Miss Exception Vosta								
Fast TEB Pliss Exception vector	r							
Fast ILB Miss Exception vector memo	None v							
Fast TLB Miss Exception vector offse	et: 0x0000000							
Fast TLB Miss Exception vector:	0x0000000							

- 28. Click on Generate HDL...
- 29. Keep the defaults and click the Generate button.
- 30. A dialog will appear asking you to save the design, click Save.
- 31. Give the name as NIOS2timerMCU.qsys and click Save.
- 32. Once the save has been completed, click Close.
- 33. The generate process kicks off. The processes should succeed, click Close.

Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation All other copyrighted, registered, and trademarked material remains the property of the respective owners.

Copyright © 2023 Annabooks, LLC. All rights reserved

- 34. Click Finish to close the design.
- 35. Quartus then reminds you to add the new design to the project. Click Ok.
- 36. In the Project Navigator, click on the drop-down and select Files.
- 37. Right-click on Files and select Add/Remote Files in Project.



- A Settings NIOS2timer page appears with Files on the left highlighted. Click the three dots browse button for File name, and navigate to \NIOS2 Timer\NIOS2timerMCU\synthesis folder.
- 39. Click on NIOS2timerMCU.qip file and click open

Name	^
submodules submodules NIOS2timerMCU.o NIOS2timerMCU.v	цір /

40. Click OK to close the Settings- NIOS2timer page. The qip file is added to the Project navigator list. Underneath are all the Verilog files that were generated by Platform Designer.



Copyright © 2023 Annabooks, LLC. All rights reserved Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation All other copyrighted, registered, and trademarked material remains the property of the respective owners.

- 41. In the Project Navigator Right-click on the NIOS2timerMCU/synthesis/NIOS2timerMCU.v file, and select Set as Top-Level Entity from the context menu.
- 42. Save the project.
- 43. In the Task pane on the left, double-click on Fitter (Place & Route) to start the task. The analysis will take some time, and it should succeed in the end. This step helps to diagnose any errors and finds the Node Names for the pin assignments in the next step.
- 44. Once the process completes, the pin assignments need to be set, from the menu select



Rev 1.3

Assignments->Pin Planner or click on the icon from the toolbar. The analysis just run populated the Node Name list at the bottom of the Pin Planner dialog.

45. Using the board schematic, locate the pins for the SW1 and the 50MHz clock. Set the Location values for both node names. For the MAX 10 – 10M08 Evaluation Board, these values are as follows:

Node Name	Location
SW1	PIN_121
Clk_50MHz:	PIN_27
altera_reserved_tck	PIN_18
altera_reserved_tdi	PIN_19
altera_reserved_tdo	PIN_20
altera_reserved_tms	PIN_16
Led5_export[4]	PIN_141
Led5_export[3]	PIN_140
Led5_export[2]	PIN_135
Led5_export[1]	PIN_134
Led5_export[0]	PIN_132

46. Set the I/O Standard to 3.3V-LVTTL for all pins except JTAG. You can see from the schematic that the I/O are all tied to 3.3V.

×	Named: *	Edit: 🗙 ✔					
P	Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
	in altera_reserved_tck	Input	PIN_18	1B	B1_N0	PIN_18	2.5 V Sc Trigger
	in altera_reserved_tdi	Input	PIN_19	1B	B1_N0	PIN_19	2.5 V Sc Trigger
	altera_reserved_tdo	Output	PIN_20	1B	B1_N0	PIN_20	2.5 V
	in altera_reserved_tms	Input	PIN_16	1B	B1_N0	PIN_16	2.5 V Sc Trigger
	in_ clk_clk	Input	PIN_27	2	B2_N0	PIN_27	3.3-V LVTTL
	ut led5_export[4]	Output	PIN_141	8	B8_N0	PIN_141	3.3-V LVTTL
	ut led5_export[3]	Output	PIN_140	8	B8_N0	PIN_140	3.3-V LVTTL
	ut led5_export[2]	Output	PIN_135	8	B8_N0	PIN_135	3.3-V LVTTL
	ut led5_export[1]	Output	PIN_134	8	B8_N0	PIN_134	3.3-V LVTTL
	ut led5_export[0]	Output	PIN_132	8	B8_N0	PIN_132	3.3-V LVTTL
Pins	in_ reset_reset_n	Input	PIN_121	8	B8_N0	PIN_121	3.3-V LVTTL
	< <new node="">></new>						

47. Close the Pin Planner when finished. The diagram gets updated with the pin numbers.48. Save the project.

Note: Quartus can crash unexpectedly, which may be due to the fact that it was written in Java and is not a native Windows application based on .NET. Therefore, a best practice at this point is to make a backup of the project folder. Archiving is simple. From the menu, Project->Archive Project.

All other copyrighted, registered, and trademarked material remains the property of the respective owners.

49. Finally, compile the design. In the Task pane, right-click on Compile and Design and select

Start from the context menu, or you can click on the symbol in the toolbar. The design should compile successfully.

Flow Summary	
< <filter>></filter>	
Flow Status	Successful - Wed Jul 13 23:00:07 2022
Quartus Prime Version	21.1.0 Build 842 10/21/2021 SJ Lite Edition
Revision Name	NIOS2timer
Top-level Entity Name	NIOS2timerMCU
Family	MAX 10
Device	10M08SAE144C8G
Timing Models	Final
Total logic elements	3,635 / 8,064 (45 %)
Total registers	2180
Total pins	7 / 101 (7 %)
Total virtual pins	0
Total memory bits	194,240 / 387,072 (50 %)
Embedded Multiplier 9-bit elements	6 / 48 (13 %)
Total PLLs	0/1(0%)
UFM blocks	0/1(0%)
ADC blocks	0/1(0%)

1.1.3 Eclipse Application: Alarm

The first application will test the Alarm functionality.

- 1. In Quartus Prime, from the menu, select Tools->Nios II Software Build Tools for Eclipse.
- 2. Eclipse will open and ask for the root workspace directory. Set the workspace folder to something like \Documents\FPGA\Apps, and hit ok. It doesn't matter the location of the workspace, since the actual applications for the project will exist within the \NIOS2_Timer\software folder.
- 3. In Eclipse, from the menu, select File->New-> Nios II Application and BSP from Template.

0		cempse			- A				
File	Edit	Navigate	Search	Project	Run	Nios II	Wind	low	Help
	New					Alt+Shif	t+N>	C++	Nios II Application and BSP from Template
	Open	File						C++	Nios II Application
	Close					Ctr	I+W	C++	Nios II Board Support Package
	Close	ΔII				Ctrl+Shift	F+W	C++	Nios II Library
	ciose	- sit				cent onn			Project
	Save					Ct	rl+S		
	Save A	.s							Other Ctrl+N
	Save A	JI				Ctrl+Shi	ft+S		

- 4. The first step is to open the SOPC file that was generated for the hardware design. Click on the three dots button.
- 5. Navigate to the \NIOS2_Timer folder and open the NIOS2timerMCU.sopcinfo file. The CPU name will reflect the name we gave the CPU in Platform Designer.
- 6. Enter the project name: timerTest.
- 7. In the Project Template, select Blank Project

Copyright © 2023 Annabooks, LLC. All rights reserved

Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation

All other copyrighted, registered, and trademarked material remains the property of the respective owners.

8. Click Finish.

Nios II Application and BSP from	m Template — 🗆 X							
Nios II Software Examples								
Create a new application and boar template	rd support package based on a software example							
Target hardware information								
SOPC Information File name:	E:\FPGA\Intel_Max_10_FPGA_Evaluation_Kit\NIOS2_Timer\NIOS2tir							
CPU name: v								
Application project								
Project name: timerTest								
Use default location Project location: E:\FPGA Project template	\\Intel_Max_10_FPGA_Evaluation_Kit\NIOS2_Timer\software\timerTe:							
Templates	Template description							
Blank Project Board Diagnostics Count Binary Float2 Functionality Float2 GCC Float2 Performance Hello Freestanding Hello MicroC/OS-II Hello World Hello World Hello World Small Memory Test Memory Test Small	Blank Project creates an empty project to which you can add your code. For details, click Finish to create the project and refer to the readme.txt file in the project directory. The BSP for this template is based on the Altera HAL operating system. To use a BSP based on a different operating system, click Next and select the BSP from the BSP projects list. For information about how this software example relates to Nios II hardware design examples.							
(?)	< Back Next > Finish Cancel							
Memory Test Small	For information about how this software example relates to Nios II hardware design examples.							

Two projects will be generated. The timerTest _bsp is generated to give you the HAL drivers and API based on the hardware design. The timerTest is the application that will run on the hardware.

- We need to edit the BSP to use the small C library and drivers. The BSP Editor tool allows you to edit the settings.bsp file to make specific changes for the target. Right-click on timerTest_bsp and select Nios II->BSP Editor from the context menu.
- 10. The BSP Editor opens and opens the settings.bsp file automatically. If you started the BSP Editor from the main menu, you would have to manually navigate to open the file. In the BSP Editor, tick the box for enable_small_c_library and enable_reduced_device_drivers.

The interval timer can either be a sys_clk_timer or a timestamp_timer, but a single interval timer cannot be both. Two intervale timers could have been added to the design. One timer (timer_0) for the system clock timer (alarms / count down) and another timer (timer_1) for the timestamp timer. For this project, we will flip timer_0 between the two application projects. The first project will be the alarm so leave sys_clk_timer as is.

- 11. Click Generate to generate the changes.
- 12. Click Exit when finished.

The timerTest_bsp contains the key files that will help with filling in the code to access the timers and pio port. System.h contains the definitions that can be used for how the timer and PIO were set up in Platform Designer. Since we are using the small_C_library for space contain reasons, the standard C io calls cannot be used. Instead, we will be using the Nios II HAL API to access the timer, pio, and the standard output via Jtag uart. The header files for the timestamp.h and alarm.h contain the APIs needed for the application.

- 13. We need to add a main.c file to the project. Right-click on the timerTest project, and select New->File from the context menu.
- 14. Enter the file name, main.c, and click Finish.
- 15. Add the following code to the main.c file.

1.	#include	"sys/alt_stdio.h"
2.	<pre>#include</pre>	"system.h"
3.	<pre>#include</pre>	"priv/alt_busy_sleep.h"
4.	<pre>#include</pre>	"sys/alt_sys_wrappers.h"
5.	#include	"altera_avalon_pio_regs.h"
6.	<pre>#include</pre>	<sys alt_timestamp.h=""></sys>
7.	#include	<sys alt_alarm.h=""></sys>

Copyright © 2023 Annabooks, LLC. All rights reserved Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation

All other copyrighted, registered, and trademarked material remains the property of the respective owners.

8.			
9.			
10.	static	alt_u32	toggleFlag = 0;
11.			
12.	//Alarm	callba	ck will toggle the LEDs
13.	alt_u32	alarm_	callback(void* context) {
14.			
15.	Ė	if (togg]	leFlag) {
16.		t	coggleFlag = 0;
17.		I	COWR_ALTERA_AVALON_PIO_DATA(PIO_0_BASE, 0x15);
18.		}	
19.	e	else{	
20.		t	coggleFlag = 1;
21.		I	OWR ALTERA AVALON PIO DATA(PIO 0 BASE, 0xa);
22.		}	
23.	/	//return	n alt_ticks_per_second(); //periodic alarm
24.	3	return (0;
25.	}		
26.			
27.			
28.	int mai	n ()	
29.	{		
30.			
31.	á	alt puts	<pre>str("Alarm test\n");</pre>
32.	1	IOWR AL	TERA AVALON PIO DATA(PIO 0 BASE, 0x18);
33.		_	
34.	/	///Set t	the alarm
35.	5	static a	alt alarm countDownAlarm;
36.	Ė	if (alt a	alarm start(&countDownAlarm, 5000, alarm callback,
	NULL)<0) {	
37.		a	alt putstr("No System Clock Found\n");
38.		}	
39.			
40.	v	while(1)) {
41.			
42.		f	for(int i = 0; i < 100; i++) {
43.			alt printf("Application is running $x n$ ", i);
44.			usleep(50000);
45.		}	· · · · · · · · · · · · · · · · · · ·
46.		1	asleep(50000);
47.		}	<u>-</u>
48.			
49.	1	return (0;
50.	}		

The basic concept for programming on top of the provided HAL drivers is the HAL API Wrappers. The various driver header files contain the wrapper APIs that are used to access the timer and PIO.



The alarm needs a callback when the alarm triggers. More than one alarm and callback can be in the code. The callbacks are put into a linked list and managed by the interrupt handler. Lines 10-25 are the callback function for this exercise. The alarm will toggle the LEDs in a pattern based on

Copyright © 2023 Annabooks, LLC. All rights reserved

Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation

All other copyrighted, registered, and trademarked material remains the property of the respective owners.

the toggleFlag. There is a return value of 0. The value 0 tells the timer driver that this is a one-shot alarm and removes the callback from the alarm link list. If an integer value or alt_ticks_per_second() is in the return, the alarm will fire off over and over again at the periodic rate until the alt_alarm_stop() function is called.

Lines 35-38 create an instance of the alarm and start the alarm countdown with a time of 5 seconds and assign the alarm_callback to respond when the countdown reaches zero. The rest of the program keeps the application alive.

- 16. Save the file.
- 17. Right-click on timerTest project again, and select Build Project. The build should complete successfully, and the timerTest.elf file has been created.
- 18. Close Eclipse

Now, we are ready to program the board with the design and debug the application.

1.1.4 **Program the Board**

With the design compiled, application ready, and circuit connect, we can now test the design on the board.

1. Connect the board and the programming cable together per the cable instructions.

Note: The MAX 10 – 10M08 Evaluation Kit doesn't come with a programming cable or built-in JTAG USB Blaster II. You will have to use either the USB Blaster II or EthernetBlaster II external cables. The EthernetBlaster II was used for this example. DHCP setup was not working so a direct Ethernet cable connection was made between a PC and the EthernetBlaster II. The static IP was set for the PC network card to 198.162.0.1. The EthernetBlaster II was accessed via a browser and then the IP address was changed to a static IP that matched the network. The new IP address was used as the Server name.

- 2. Power on the board and the programming cable box.
- 3. In Quartus Prime, from the Task pane, right-click on Program Device (Open Programmer)



and select Open from the context menu or click on the **real select** icon on the toolbar.

- 4. The Programmer dialog appears, click on the "Hardware Setup" button.
- 5. Click the Add hardware button, select the Hardware type and fill in any remaining information, and click OK.

Hardware Settings	JTAG S	ettings		
Select a programming	g hardwa	re setup to use wh	nen programming devi rammer window	ces. This programming
ardware setup applit	is only to	, the current progr	annier window.	
Surrently selected ha	rdware:	No Hardware		
Hardware frequency Add Hardware	2		×	H
Hardware type:	Ethern	etBlaster	•	Add Hardware
Port:			Ŧ	Remove Hardware
Baud rate:			Ŧ	
Server name:			•	
Server port:	1309			
Server password:				
		011	Canad	

6. The tool allows you to connect to a number of programming cables. We need to select the one for our board. In the "Currently selected hardware", click the drop-down and select the hardware cable for the board, and click Close when finished

Hardware Settings JTAG S	ettings			
Select a programming hardwa hardware setup applies only to	re setup to u the current	se when pro programme	ogramming device r window.	es. This programming
Currently selected hardware:	EthernetBla	asterll on 19	2.168.1.198 [Ethe	ernetBlasterII] 🔹
Hardware frequency:				н
Available hardware items				
Hardware	Se	erver	Port	Add Hardware
EthernetBlasterII	19	192.168.1	EthernetBl	Remove Hardware

7. A NIOS2timer_time_limited.sof file gets created during the Compile Design flow. The file is automatically filled in. There is only one FPGA on the board and in the JTAG chain so the file already has the Program/Configure checkbox checked. Click the Start button to program the board. The process takes a few seconds and shows that the task was completed successfully.

Note: The reason for the "time_limited" in the name of the .sof file is that we chose a Nios II/f, which requires a license. The design must be connected to the JTAG cable or the system will shut off after an hour.

e <u>E</u> dit <u>V</u> iew	Processing Tools Wind	dow <u>H</u> elp					Bearch	Intel FPGA	
Hardware Setu	p on 192.168.1.198 [Et	hernetBlasterII]	Mode: JTAG		₹ P	rogress:			
Enable real-tim	e ISP to allow background p	rogramming when a	vailable						
▶ [™] Start	File	Device	Checksum	Usercode	Program/	Verify	Blank-	Examine	Secur
Stop	output_files/NIOS2adc	. 10M08SAE144	002F3207	002F3207	V		Check		
Auto Detect									
× Delete									
Add File									
Change File	•								
Save File									
Add Device	(1010)								
1 ¹ Up									
1% e									
↓ ^m Down	10M0854	144							
↓ [™] Down	TDO								
+ [∞] Down	TDO								

A dialog will appear that the design is time limited to one hour. The design can always be reloaded when the timeout occurs.

OpenCore Plus Status	×							
Design contains one or more time-limited OpenCore Pl								
Time remaining: 00	0:59:56							
<u>C</u> lose								

Important: This dialog acts as a tether to the time-limited IP. You must leave this dialog running while you are running applications.

1.1.5 Deploy the Application and Other Tests

With the design loaded and the connection to JTAG up and running, we can test the application.

- 1. From the Quartus menu, select Tools-> Nios II Software Build Tools for Eclipse.
- 2. Open the main.c application.
- 3. Right-click on timerTest and select Run As->Nios II Hardware. The program will load and start running.

The first 3 LEDs will turn on and after the alarm goes off, the LEDs will change to an on-off-onoff-on pattern. The application continues to run in the loop, but the alarm is done.

- 4. Edit main.c at line 24, and change the return from 0 to 3000.
- 5. Save the main.c file.
- 6. Rebuild the application.

Copyright © 2023 Annabooks, LLC. All rights reserved Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation All other copyrighted, registered, and trademarked material remains the property of the respective owners. 7. Right-click on timerTest and select Run As->Nios II Hardware. The program will load and start running.

The first 3 LEDs will turn on and after the alarm goes off, the LEDs start flashing the two patterns periodically every 3 seconds. If we had alt_arlam_stop() in the code somewhere, this would stop the alarm.

- 8. Edit main.c again at line 23, and change the return from 3000 to 0. The code is going back to a one-shot alarm.
- 9. At line 36 change the countdown value from 5000 to 300.
- 10. After line 38 add the following code:

```
for(int i = 0; i < 10; i++) {
    alt_printf("Time stamp running %x\n", i);
    usleep(50000);</pre>
```

}

}

- 11. Save the main.c file.
- 12. Rebuild the application.
- 13. Right-click on timerTest and select Run As->Nios II Hardware. The program will load and start running.

The alarm callback is set for a single-shot operation. Two alarms are set up for the same callback, but for the application to NOT crash, the first alarm has to finish before the second alarm can be set. Setting the second alarm just after the first for the same callback will crash the application. The loop between the two alarm_start() calls provides some delay. You will see the LEDs go through the pattern once and then the alarms are no longer set. This demonstrates that an alarm callback can be reused after it has been triggered.

If you want, you could create a different callback for the second alarm, that puts out a different LED pattern.

1.1.6 Eclipse Application: Timestamp

This second application will demonstrate the timestamp functionality of the Interval Timer.

- 1. From the menu in Eclipse, select File->Nios II Application.
- 2. Enter the project name timerTest2.
- 3. For the BSP Location, click on the 3 dot button.
- 4. Select timerTest_bsp and click OK.

← Nios II Application										
Nios II Application										
Create a new Nios II Software Build Tools application project										
Project name:	timer lest2									
BSP location: E:\FPGA\Intel_Max_10_FPGA_Evaluation_Kit\NIOS2_Timer\software\tin										
			Crea	ate						
✓ Use default	location									
$\label{eq:location} \mbox{Location} E:\mbox{FPGA}\mbox{Inte}\mbox{Max}\mbox{10}\mbox{FPGA}\mbox{Evaluation}\mbox{Kit}\mbox{NIOS2}\mbox{Timer}\mbox{software}\mbox{t} \ .$										
Additional arguments:										
	A									
Command:										
nios2-app-generate-makefile.exeapp-dirbsp-dir/timerTest_bspelf-name										
☑ Use relative path										
?		Finish	Cance	:I						

- 5. Click Finish.
- 6. Now, we need to edit the BSP to change timer_0 to be assigned as a timestamp_timer. Right-click on timerTest_bsp and select Nios II->BSP Editor.
- 7. Change sys_clk_timer to none.
- 8. Change timerstamp_timer to timer_0.

🁪 BSP Editor - settings.bsp		-		×			
File Edit Tools Help							
Main Software Packages Drivers Linker Script Enable File	Generation Target BSP Directory						
SOPC Information file: E:\FPGA\Intel_Max_10_FPGA_Evalue CPU name: nios2 Operating system: Altera HAL BSP target directory: E:\FPGA\Intel_Max_10_FPGA_Evalue	tion_Kit\vIOS2_Timer\vIOS2timerMCU.sopcinfo Version: default tion_Kit\vIOS2_Timer\software\timerTest_bsp						
Settings Common Common Settings Common Settings Setti	hal sys_clk_timer: none timestamp_timer: timer_0 ~ stdin: jtag_uart_0 ~ stdout: jtag_uart_0 ~ stderr: jtag_uart_0 ~ @ enable_small_c_library						
Information Decklasse December	L						
Loading drivers from ensemble report							
Mapped module: "hits2" to use the default driver version. Mapped module: "bits2" to use the default driver version. Mapped module: "bits2" to use the default driver version. Mapped module: "sysid_csys_0" to use the default driver version. Mapped module: "sysid_csys_0" to use the default driver version. Mapped module: "sysid_csys_0" to use the default driver version. Mapped module: "sysid_csys_0" to use the default driver version. Mapped module: "sysid_csys_0" to use the default driver version. Mapped module: "sysid_csys_0" to use the default driver version. Mapped module: "sysid_csys_0" to use the default driver version. Mapped module: "sysid_csys_0" to use the default driver version.							
Loading BSP settings from settings file.							
Finished loading SOPC Builder system info file "E:\FPGA\Intel	_Max_10_FPGA_Evaluation_Kit\VIOS2_Timer\VIOS2timerMCU.sopcinfo"			¥			
	Genera	te	Exit	t			

- 9. Click Generate.
- 10. Click Finish.
- 11. Right-click on timerTest2 and select New->File.
- 12. Name the file main.c and click finish.
- 13. Add the following code to main.c:

```
1.
       #include "sys/alt stdio.h"
2.
       #include "system.h"
3.
       #include "priv/alt busy sleep.h"
       #include "sys/alt_sys_wrappers.h"
4.
5.
       #include "altera_avalon_pio_regs.h"
6.
       #include <sys/alt_timestamp.h>
7.
       #include <sys/alt_alarm.h>
8.
9.
10.
       int main()
11.
       {
12.
13.
             alt putstr("Timer test\n");
14.
             IOWR_ALTERA_AVALON_PIO_DATA(PIO_0_BASE, 0x18);
15.
16.
             alt u8 timestampavailable = 0;
             alt u32 timeBefore, timeAfter, timerOverhead, totalTicks;
17.
18.
19.
             if(alt_timestamp_start() <0){</pre>
20.
                    alt putstr("No timestamp timer found\n");
21.
              }
22.
             else{
23.
                    timestampavailable = 1;
24.
                    timeBefore = alt_timestamp();
```

Copyright © 2023 Annabooks, LLC. All rights reserved

Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation

All other copyrighted, registered, and trademarked material remains the property of the respective owners.

~ -	
25.	<pre>timeAfter = alt_timestamp();</pre>
26.	timerOverhead = timeAfter - timeBefore;
27.	}
28.	
29.	//Test the time stamp functionality.
30.	<pre>if(timestampavailable){</pre>
31.	
32.	<pre>timeBefore = alt timestamp();</pre>
33.	for (int i = 0; i ⁻ < 10; i++) {
34.	alt printf("Time stamp running %x\n", i);
35.	alt busy sleep(6000);
36.	}
37.	<pre>timeAfter = alt timestamp();</pre>
38.	
39.	<pre>totalTicks = timeBefore-timeAfter-timerOverhead;</pre>
40.	
41.	alt printf("Total number of ticks in Hex to run the loop:
	<pre>%x\n", totalTicks);</pre>
42.	
43.	}
44.	while(1){
45.	
46.	usleep(5000);
47.	}
48.	
49.	return 0;
50.	}

The application tests the number of timer-ticks it takes to run through the for-loop at lines 33-36. The application sets up the variables to be used to get the time stamp before and after the loop runs. Before it can perform the test, the overhead of performing both timestamp operations is calculated. The test is performed and the resulting number of ticks is presented as a hex value.

- 14. Save the application.
- 15. Build the application.
- 16. With the design's sopc file programmed to the FPGA, run the application.

The application simply outputs from the loop and posts the result. Timestamps can be helpful when diagnosing code that is time critical.

	Pro	oblems	🤕 Tasks	₽	Console	📩 Nios	Il Console	ß	I I I	properties	🔗 Search	Progress	
tim	erTe	st2 Nios	II Hardware	conf	figuration -	cable: Et	hernetBlaste	erII o	n 192	. 168. 1. 198	[EthernetBlas	terII] device ID	: 1 instance
T	imeı	r test											
T	ime	stamp	running	0									
T	ime	stamp	running	1									
T:	ime	stamp	running	2									
T:	ime	stamp	running	3									
T	ime	stamp	running	4									
T:	ime	stamp	running	5									
T:	ime	stamp	running	6									
T:	ime	stamp	running	7									
T:	ime	stamp	running	8									
T:	ime	stamp	running	9									
T	otal	l numbe	er of tio	:ks	in Hex	to run	the loop	: f	£922	730			

17. When finished close Eclipse, the OpenCore Plus dialog, and the JTAG programming application.

1.2 Summary: It is About Time

As a test, you can go back to the design and add a second timer (timer_1), and then in the timerTest_bsp edit the BSP to have one timer be the sys_clk_timer and the other be the timestamp_timer. A single application can support both alarms and timestamps.

1.3 References

The following references were used for this article:

Nios® II Processor: Hardware Abstraction Layer Exercise Manual, Intel Corporation,

Nios® II Software Developer Handbook, V21.3, Intel Corporation, 10/4/21