International Journal of Industrial Engineering & Production Research

March 2012, Volume 23, Number 1
pp. 13-21

http://IJIEPR.iust.ac.ir/

ISSN: 2008-4889

GPS Time Reception Using Altera SOPC Builder and Nios II: Application in Train Positioning

S. Moslehpour[*], K. Jenab & S. Valiveti

***Saeid Moslehpour**, Chair, Dept of ECE., Hartford University, USA,*
***Kouroush Jenab**, Education Chair, Society of Reliability Engineering-Ottawa, Canada,*
***Srikar Valiveti**, Dept of ECE., Hartford University, USA,*

KEYWORDS

Train positioning, GIS, GPS, SOPC, Nios II

ABSTRACT

As functional integration has increased in hand-held consumer devices features such as Global Positioning System (GPS) receivers have been embedded in increasingly more devices in recent years. For example, the train positioning system based on GPS provides an integrated positioning solution which can be used in many rail applications without a cost intensive infrastructure. The network built in the GPS receiver has the advantage of determining the exact location and time of the train. The objective of this research was to develop a system which accepts the location from the GPS receiver mounted on the train and extracts its local time. This is implemented using Altera SOPC builder in the NIOS – II environment. Nios II is a 32 bit soft-core embedded-processor architecture designed specifically for the Altera family of FPGAs. The signal received using the GPS receiver is given to the DE2 board through the UART port and converted it in to local time and displayed on the NIOS II console. A working system was developed, which accepts the location from the GPS receiver and extracted its local time.

1. Introduction

The wide application of the global positioning system (GPS) results in further development of the geographic information systems and vice versa. By receiving the geometric and geographic information, the geographic information system technology (GIS) can produce GIS database and thematic maps. As a result, the GPS allows the accurate positioning of a train based on satellite signals [12].
The main objective of this project is to develop a system which accepts the location from the GPS receiver and extracts its local time. This is implemented using Altera SOPC Builder in the Nios II environment. As successor of the CPLD (Complex Programmable Logic Device), the FPGA (Field Programmable Gate Array) is a digital hardware-programmable component that offers lots of new perspectives for the digital signal processing, which has been growing into the power electronics area. Based on configurable connections between its basic blocks, the logic elements (LE), the user can design a system with multiple functions operating in parallel inside one chip. This property, and others, gives some advantages of the FPGA over DSP, as the possibility of hardware optimization and the parallel processing of data. Furthermore, the FPGA design made for one specific component can be easily fitted into another with similar capabilities, and can be modified whenever necessary [10].
The growth in size and performance of field programmable gate arrays (FPGAs) has compelled system-on-a-programmable-chip (SoPC) designers to use soft processors for controlling systems with large numbers of intellectual property (IP) blocks. Soft processors control IP blocks, which are accessed by the

* **Corresponding author: Saeid Moslehpour**
Email: *moslehpou@hartford.edu*
Paper first received July 20, 2011, and in revised form Nov. 04, 2011.

processor either as peripheral devices or/and by using custom instructions (CIs) [3].
Nios II is a 32 bit soft-core embedded-processor architecture designed specifically for the Altera family of FPGAs. FPGA is one of the most successful of today’s technologies for developing the systems which require a real time operation. Semi-custom and full custom application specific integrated circuit (ASIC) devices are also used for this purpose but FPGA provide additional flexibility as they can be used with tighter time to-market schedules. FPGAs provide the ability to upgrade architectures quickly to meet evolving requirements, while scalability allows use of FPGAs in low-cost and high-performance systems. FPGA places fixed logic cells on the Wafer and the FPGA designer construct more complex functions from these cells [4]. The signal received using the GPS receiver is sent to the DE2 board through the UART port and converted it in to local time and displayed on the console.

1-1. System-On-a-Programmable Chip (SOPC)

A new technology has emerged that enables designers to utilize a large Field Programmable Gate Arrays (FPGA) that contains both memory and logic elements, along with a processor core to implement a computer and custom hardware for system-on-a-chip (SOC) applications.
This approach has been termed as system-on-a-programmable chip (SOPC). SOPC Builder is a powerful system development tool which enables one to define and generate a complete system-on-a-programmable-chip (SOPC) in much less time than using traditional, manual integration methods. The designing technology of SOPC is the products of the modern computer-aided design technology, the EDA technology and the great development of large scale integrated circuit technology.
SOPC technology is a completed electronic system, including the embedded processor system, the port system and the hardware acceleration or co-processors systems, the DSP systems, digital communication systems, the storage and general digital circuit system. It is embedded in a single FPGA to achieve the design of the circuit. SOPC Builder automates the task of integrating hardware components [9]. It reduces the task of manually writing HDL modules to wire the components of the system. Once the system components are specified in a Graphical user Interface (GUI), the SOPC Builder automatically generates the interconnect logic [1].

1-2. Soft Processor

To increase the flexibility of single-chip evolvable hardware systems, we explore possibilities of systems with the evolutionary algorithm implemented in software on an on-chip processor. This gives higher flexibility compared to implementing an evolutionary algorithm directly in hardware, since the parameters and behavior of the algorithm can easily be changed, and complex operators are more feasible to implement [11].
Soft processors have become an increasingly common component of systems that use Field-Programmable Gate Arrays (FPGAs). Soft Processors are used to implement a wide variety of control and data processing functionality. Often, some additional functionality needs to be added to a system when there is very little space left on the physical device. This functionality may not be performance critical, and so could be implemented on a slow soft processor. For this reason it may be useful to have a processor that is as small as possible yet similar to other commonly-used processors [2].

1-3. Nios II Processor

A Nios II processor system is equivalent to a microcontroller or computer on a chip that includes a CPU and a combination of peripherals and memory on a single chip. The term Nios II processor system refers to a Nios II processor core, a set of on-chip peripherals, on chip memory, off-chip memory, which are all implemented on a single Altera chip. Like a microcontroller family, all Nios II processor systems use a consistent instruction set and programming model.

1-4. SOPC Design Flow

The traditional flow of Computer Aided Tools (CAD) typically follows a path from hardware description language (HDL) or schematic design entry. The design is synthesized and necessary tools are used to program the design into an FPGA. A processor core configuration tool block is provided in the SOPC Builder, which is a user friendly GUI interface that allows the designer to customize the processor for a particular application. The configurable parameters include the datapath width, memory, address space and peripherals such as general purpose I/0, UART’s, Ethernet controllers, memory controllers. Once the processor parameters are specified in the GUI interface, the processor core is generated in the form of an HDL file or netlist file.
The full hardware system designed is then compiled and the FPGA can be programmed with the resulting file using the standard tools included in the SOPC Builder [1].
When the processor core configuration tool generates the HDL or netlist files, it also creates a number of library files and their associated C header files, that are customized for the specific processor core generated. A C/ C++ compiler targeted at this processor is also provided which allows the designer to program the application on the processor.
Once a program file has been generated, it must be loaded into the processor’s program and data memories. This loading can be done in several ways depending on the memory configurations of the

processor. In general, processor cores are classified as either hard or soft.
This designation refers to the flexibility or configurability of the core. Hard cores are less configurable, but they tend to have higher performance characteristics than soft cores. Hard processor cores use an embedded processor core in addition to FPGA's normal logic elements. Soft cores use existing programmable logic elements from the FPGA to implement the processor logic. Soft core processors are feature rich and flexible, allowing the designer to specify the memory width, the ALU functionality, number and types of peripherals and memory address space parameters. Soft cores have slower clock rates and use more power than an equivalent hard processor core.
For projects requiring a hardware implementation, the FPGA based SOPC approach is easier, faster and more economical [1].

1-5. SOPC Component

An SOPC Builder component is a hardware design block available within SOPC Builder that can be instantiated in an SOPC Builder system. These are the following types of components in an SOPC Builder system:

1. Components that include their associated logic inside the SOPC Builder system.
2. Components that interface to logic outside the SOPC Builder system [5].

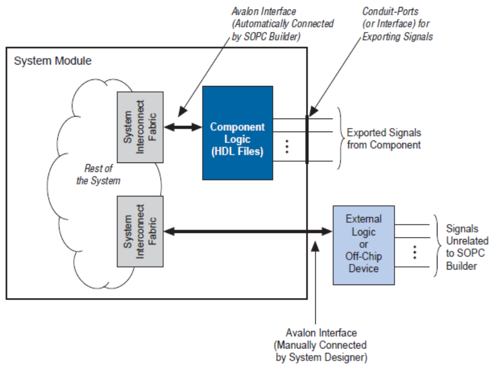

Avalon Interface (Automatically Connected by SOPC Builder)
Conduit-Ports (or Interface) for Exporting Signals
System Module
System Interconnect Fabric
Component Logic (HDL Files)
Exported Signals from Component
Rest of the System
System Interconnect Fabric
External Logic or Off-Chip Device
Signals Unrelated to SOPC Builder
Avalon Interface (Manually Connected by System Designer)

Fig. 1. SOPC Components

Fig 1, represents components that are instantiated inside the SOPC Builder system. The component defines its logic in an associated HDL file. During system generation, SOPC Builder instantiates the component and connects it to the rest of the system. The component can include exported signals in conduit interfaces. Conduit interfaces become ports on the system, so they can be connected to logic outside the SOPC Builder system in the board-level schematic [5].
In general, components are connected to the system interconnect fabric using the Avalon Memory-Mapped (Avalon-MM) interface or the Avalon Streaming (Avalon-ST) interface. A single component can provide more than one Avalon port i.e. a component might provide an Avalon-ST source port, in addition to an Avalon-MM slave for control.

1-6. Avalon Switch Fabric

The system interconnect fabric is the collection of interconnect and logic resources that connects Avalon-MM master and slaves on components in a system. SOPC Builder generates the system interconnect fabric to match the needs of the components in a system. The system interconnect fabric implements the connection details of a system. It guarantees that signals are routed correctly between master and slaves, as long as the ports hold to the rules of the Avalon Interface Specifications.
System interconnect fabric for memory-mapped interfaces supports the following:

1. Any number of master and slave components. The master-to-slave relationship can be one-to-one, one-to-many, many-to-one, or many-to-many.
2. On-chip components.
3. Interfaces to off-chip devices.
4. Master and slaves of different data widths.
5. Components operating in different clock domains.
6. Components using multiple Avalon-MM ports [5].

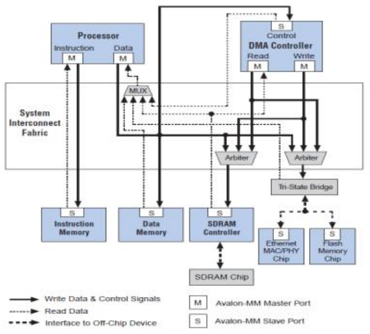


Fig. 2. Avalon Switch Fabric

SOPC Builder supports components with multiple Avalon-MM interfaces, such as the processor component shown in Fig 2. Because SOPC Builder can create system interconnect fabric to connect components with multiple interfaces, one can create complex interfaces that provide more functionality than a single Avalon-MM interface.
System interconnect fabric can connect any combination of components, as long as each interface conforms to the Avalon Interface Specifications. It can connect a system comprised of only two components with unidirectional dataflow between them. Avalon-MM interfaces are suitable for random address transactions, such as to memories or embedded peripherals. Generating system interconnect fabric is SOPC Builder's primary purpose [5].

1-7. Assigning IRQs in SOPC Builder

One can specify IRQ settings on the System Contents tab of SOPC Builder. After adding all components to the system, one can make IRQ settings for all interrupt senders, with respect to each interrupt receiver. For each slave, one can either specify an IRQ number, or specify not to connect the IRQ.

2. Nios II Processor

As shown in Fig 3, Nios II processor system is equivalent to a microcontroller or computer on a chip that includes a CPU and a combination of peripherals and memory on a single chip. Like a microcontroller family, all Nios II processor systems use a consistent instruction set and programming model.

The Nios II processor is a general-purpose RISC processor core, providing:

1. Full 32-bit instruction set, data path, and address space.
2. 32 general-purpose registers and 32 external interrupt sources.
3. Single-instruction 32 × 32 multiply and divide producing a 32-bit result.
4. Dedicated instructions for computing 64-bit and 128-bit products of multiplication.
5. Single-instruction barrel shifter.
6. Access to a variety of on-chip peripherals, and interfaces to off-chip memories and peripherals.
7. Hardware-assisted debug module enabling processor start, stop, step and trace under integrated development environment (IDE) control.
8. Software development environment based on the GNU C/C++ tool chain and Eclipse IDE.
9. Instruction set architecture (ISA) compatible across all Nios II processor systems [6].

Fig. 3. Nios II Processor

2-1. Customizing Nios II Processor Designs

Altera FPGAs provide flexibility to add features and enhance performance of the processor system. Unnecessary processor features and peripherals can be eliminated and made to fit the design of a smaller, lower-cost device.

The pins on the chip can be rearranged to make board design easier. For example, address and data pins for external SDRAM memory can be moved to any side of the chip to shorten board traces [6].

Extra pins and logic resources on the chip can be used for functions unrelated to the processor. Extra resources can provide a few extra gates and registers as glue logic for the board design; or extra resources can implement entire systems. For example, a Nios II processor system consumes only 5% of a large Altera FPGA, leaving the rest of the chip's resources available to implement other functions.

Extra pins and logic on the chip can be used to implement additional peripherals for the Nios II processor system. Altera offers a growing library of peripherals that can be easily connected to Nios II processor systems.

In practice, most FPGA designs do implement some extra logic in addition to the Nios II processor system. Additional logic has no effect on the programmer's view of the Nios II processor.

2-2. Configurable Soft-Core Processor

The Nios II processor is a configurable soft-core processor, as opposed to a fixed, off-the-shelf microcontroller. Configurable means that features can be added or removed on a system-by-system basis to meet performance or price goals. "Soft-core" means the CPU core is offered in "soft" design form (i.e., not fixed in silicon), and can be targeted to any Altera FPGA family.

It is the users that configure the Nios II processor and peripherals to meet their specifications, and then program the system into an Altera FPGA [7].

Configurability does not mean that designers must create a new Nios II processor configuration for every new design. Altera provides ready-made Nios II system designs that system designers can use asis. If these designs meet the system requirements, there is no need to configure the design further.

In addition, software designers can use the Nios II instruction set simulator to begin writing and debugging Nios II applications before the final hardware configuration is determined.

2-3. Flexible Peripheral Set & Address Map

A flexible peripheral set is one of the most notable differences between Nios II processor systems and fixed microcontrollers. Because of the soft core nature of the Nios II processor, designers can easily build made-to-order Nios II processor systems with the exact peripheral set required for the target applications [6].

A result of flexible peripherals is a flexible address map. Software constructs are provided to access memory and peripherals generically, independently of address location. Therefore, the flexible peripheral set and address map does not affect application developers.

Peripherals can be categorized into two broad classes:

1. Standard peripherals.
2. Custom peripherals.

a. Standard Peripherals

Altera provides a set of peripherals commonly used in microcontrollers, such as timers, serial communication interfaces, general-purpose I/O, SDRAM controllers, and other memory interfaces as shown in Fig 4. The list of available peripherals continues to grow as Altera and third-party vendors release new soft peripheral cores [6].

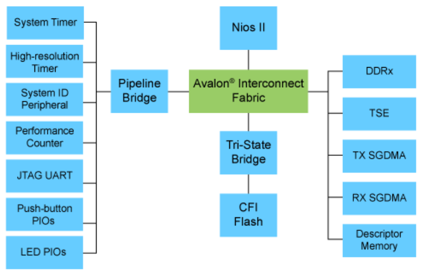


Fig. 4. Standard Peripherals

b. Custom Peripherals

As shown in Fig 5, designers can also create their own custom peripherals and integrate them into Nios II processor systems. For performance, critical systems that spend most CPU cycles executing a specific section of code, it is a common technique to create a custom peripheral that implements the same function in hardware. This approach offers a double performance benefit: the hardware implementation is faster than software; and the processor is free to perform other functions in parallel while the custom peripheral operates on data [6].

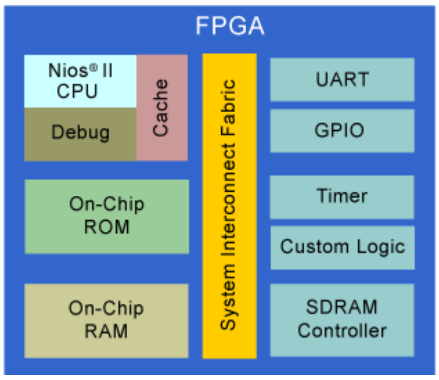


Fig. 5. Custom Peripherals

c. Custom Instructions

Like custom peripherals, custom instructions are a method to increase system performance by augmenting the processor with custom hardware. The soft-core nature of the Nios II processor enables designers to integrate custom logic into the arithmetic logic unit (ALU). Similar to native Nios II instructions, custom instruction logic can take values from up to two source registers and optionally write back a result to a destination register [6].

By using custom instructions, designers can tune the system hardware to meet performance goals. Because the processor is implemented on reprogrammable Altera FPGAs, software and hardware engineers can work together to iteratively optimize the hardware and test the results of software executing on real hardware.

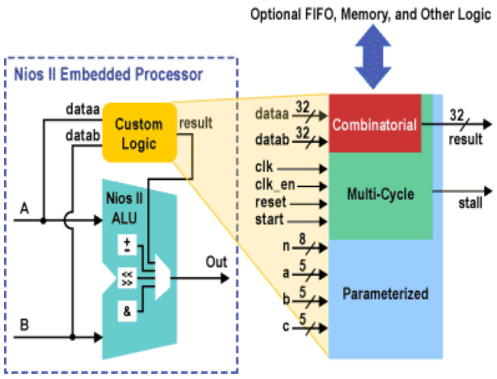


Fig. 6. Custom Instructions

From the software perspective, custom instructions appear as machine generated assembly macros or C functions, so programmers do not need to know assembly in order to use custom instructions.

SOPC Builder design tool fully automates the process of configuring processor features and generating a hardware design that can be programmed into an FPGA.

Fig 6, represents the SOPC Builder graphical user interface (GUI) which enables hardware designers to configure Nios II processor systems with any number of peripherals and memory interfaces. Entire processor systems can be created without requiring the designer to perform any schematic or hardware description-language (HDL) design entry. SOPC Builder can also import a designer's HDL design files, providing an easy mechanism to integrate custom logic into a Nios II processor system.

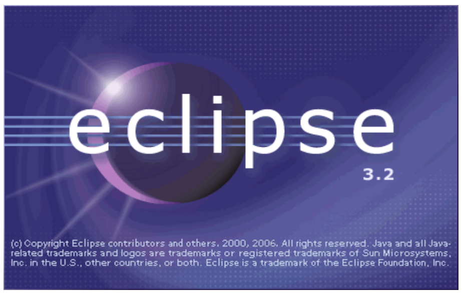
After system generation, the design can be programmed onto a board, and software can be debugged while program is being executed on the board. Once the design is programmed into a board, the processor architecture is fixed. Software development proceeds in the same manner as for traditional, non-configurable processors [6].

3. Proposed Design of the Project

The main objective of this research is to develop a system which accepts the location from the GPS receiver and extracts its local time. This is implemented using Altera SOPC Builder in the NIOS II environment. The previous chapters explain in details about the SOPC Builder and the NIOS II processor. This chapter discusses in detail about the working of the proposed design.

As the idea of using a soft-core processor was something new, our initial plan was to develop a simpler design using the SOPC Builder and NIOS II processor. But after the training (organized by Altera corporation) on the above mentioned tools, the project was developed on a complex design.

A personal computer (PC) provided with Quartus II software including the SOPC Builder environment and the Nios II soft-core processor are used for developing the proposed system. The figure shown above explains the system in detail. The GPS antenna receives the signals from the satellite and these signals are transmitted to the system using the GPS module. The GPS receiver and the GPS transmitter together form a GPS Module.
The extracted signals are transmitted to the development board via RS 232 which is used for serial communication.
The software built will be executed by a Nios II processor-based system in an FPGA. Therefore, the initial step is to configure the FPGA on your development board with the pre-generated Nios II standard hardware system. One has to download the FPGA configuration file, that is, the SRAM Object File (**.sof**) that contains the Nios II standard system to the board.
As the main idea of the project was to extract the GPS time, the software part was developed to eliminate the rest of the content received by the Antenna. Fig 7, represents Eclipse IDE environment where the software part was developed in C language. The eclipse environment has a C/C++ compiler and a set of powerful commands, utilities, and scripts to build options for applications, board support packages, and software libraries. Nios II Software Build Tools for Eclipse focuses on improving software productivity for large software applications and team-based software design.



(c) Copyright Eclipse contributors and others. 2000, 2006. All rights reserved. Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S., other countries, or both. Eclipse is a trademark of the Eclipse Foundation, Inc.

Fig. 7. Developing of software part in Eclipse environment

3-1. Implementation

The block diagram provided in Fig 8, explains about the implementation of the proposed application.

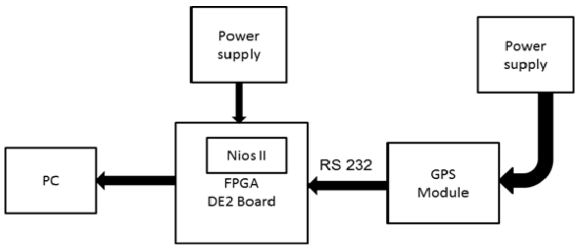


Fig. 8. Block Diagram of the Proposed Application

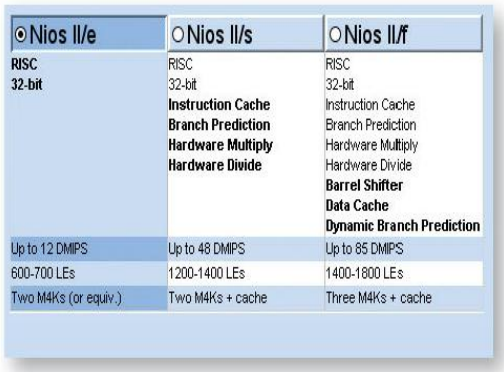
3-2. Nios II standard hardware system

The hardware system developed has four main parts:

1. Nios II Processor core.
2. On-chip memory.
3. JTAG UART.
4. UART(RS 232 for serial port)

Fig 9, represents the core configuration of Nios II processor Nios II processor core has three cores:

- Nios II/f core (Fast)
- Nios II/s core (Standard)
- Nios II/e core (Economy)



⦿Nios II/e	○Nios II/s	○Nios II/f
RISC **32-bit**	RISC 32-bit **Instruction Cache** **Branch Prediction** **Hardware Multiply** **Hardware Divide**	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide **Barrel Shifter** **Data Cache** **Dynamic Branch Prediction**
Up to 12 DMIPS	Up to 48 DMIPS	Up to 85 DMIPS
600-700 LEs	1200-1400 LEs	1400-1800 LEs
Two M4Ks (or equiv.)	Two M4Ks + cache	Three M4Ks + cache

Fig. 9. Nios II Core Configuration

a. Nios II/f Core (Fast)

The Nios II/f fast core is designed for high execution performance. Performance is gained at the expense of core size.
The base Nios II/f core, without the memory management unit (MMU) or memory protection unit (MPU) is approximately 25% larger than the Nios II/s core. "Nios II/f is designed to maximize the instructions-per-cycle execution efficiency, to optimize interrupt latency, to maximize fMAX performance of the processor core" [6].
The Nios II/f core has separate optional instruction and data caches. It provides optional MMU to support operating systems that require an MMU. It also provides optional MPU to support operating systems and runtime environments that desire memory protection but do not need virtual memory management.
Nios II/f core can access up to 2 GB of external address space when no MMU is present and 4 GB when the MMU is present. It supports optional external interrupt controller (EIC) interface to provide customizable interrupt prioritization. It also supports optional shadow register sets to improve interrupt latency, and supports optional tightly-coupled memory for instructions and data.
Nios II/f core provides optional hardware multiply, divide, and shift options to improve arithmetic performance and supports the addition of custom instructions.

b. Nios II/s core (Standard)

The Nios II/s standard core is designed for small core size. On-chip logic and memory resources are conserved at the expense of execution performance. The Nios II/s core uses approximately 20% less logic than the Nios II/f core, but execution performance also drops by roughly 40%.

Designed Nios II/s does not cripple performance for the sake of size. This core can be used for medium-performance applications [6].

The Nios II/s core has an instruction cache, but no data cache, and can access up to 2 GB of external address space.

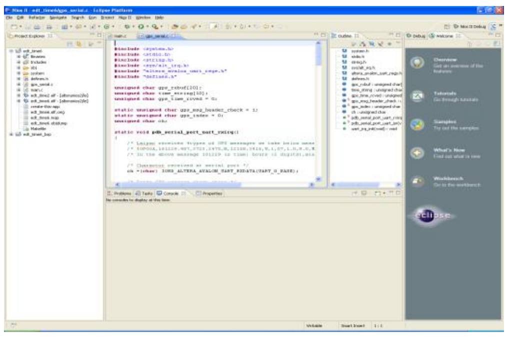
It supports optional tightly-coupled memory for instructions and employs a 5-stage pipeline. It provides hardware multiply, divide, and shift options to improve arithmetic performance and also supports the addition of custom instructions and JTAG debug module.

c. Nios II/e Core (Economy)

The Nios II/e economy core is designed to achieve the smallest possible core size. Nios II/e core is designed with a singular design goal; i.e. to reduce resource utilization any way possible, while still maintaining compatibility with the Nios II instruction set architecture.

Hardware resources are conserved at the expense of execution performance. The Nios II/e core is roughly half the size of the Nios II/s core, but the execution performance is substantially lower.

The Nios II/e core executes at most one instruction per six clock cycles and can access up to 2 GB of external address space. Like other cores it supports the addition of custom instructions and JTAG debug module. But it does not provide hardware support for potential unimplemented instructions. It does not have instruction cache or data cache and cannot perform branch prediction [6].



```
#include <system.h>
#include <stdio.h>
#include <string.h>
#include <sys/alt_irq.h>
#include "altera_avalon_uart_regs.h"
#include "defines.h"

unsigned char gps_rxbuf[10];
unsigned char time_string[10];
unsigned char gps_time_rcved = 0;

static unsigned char gps_msg_header_check = 1;
static unsigned char gps_index = 0;
unsigned char ch;

static void pdb_serial_port_uart_rxirq()
{
    /* [illegible] */
    /* [illegible] */
    /* [illegible] */

    /* Character received at serial port */
    ch =(char) IORD_ALTERA_AVALON_UART_RXDATA(UART_0_BASE);
```

Fig. 10. Developing the Nios II Application in Eclipse Environment

Fig 10, represents the development of the application (Nios II/S core) in eclipse environment. The on-chip memory is used to store the date from the GPS receiver.

3-3. On-Chip Memory.

The on-chip Static Random Access Memory (SRAM) is used for data memory residing on-chip that is mapped into an address space which is put out of place from the off-chip memory but connected to the same address and data buses. If the application is small and can fit into memory blocks available on the FPGA, then the program can be initialized in the on-chip memory when the hardware configuration is programmed [6].

As the application program has to be modified a number of times before the final program is complete, a boot loader is provided to download the application code from the PC to the memory on an FPGA. A software boot loader is comprised of code that is loaded into an on-chip memory.

A hardware boot loader provides functionality very similar to a software boot loader, but it is implemented in dedicated logic within the processor core. The boot loader hardware can start and stop the processor and can control the downloading of a program over the JTAG or serial interface to the desired memory locations.

3-4. RS 232 Interface

RS-232 (Recommended Standard 232) is a standard for serial binary single-ended data and control signals connecting between a DTE (Data Terminal Equipment) and a DCE (Data Circuit-terminating Equipment).

It is commonly used in computer serial ports. The standard defines the electrical characteristics and timing of signals, the meaning of signals, and the physical size and pin out of connectors. As a result, the voltage values for the data bits and the control signals are opposed to each other.

3-5. UART

The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes.

Serial transmission is commonly used with modems and for non-networked communication between computers, terminals and other devices [9].

3-6. System Interconnect Fabric for Streaming Interfaces

Avalon-ST interconnect fabric is logic generated by SOPC Builder. SOPC Builder can specify how Avalon-ST source and sink ports connect. SOPC Builder creates a high performance point-to-point interconnect between the two components. The Avalon-ST interconnect is flexible and can be used to implement on-chip interfaces for industry standard telecommunications and data communications cores.

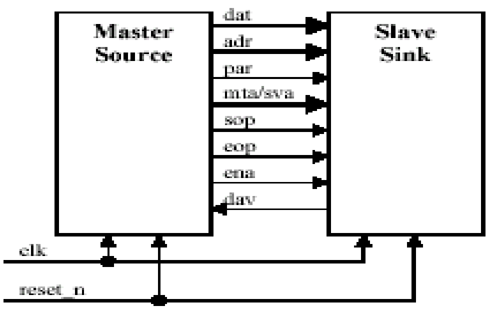


Fig. 11. Interconnect between source and sink

Fig 11, illustrates the simplest system, which generates interconnect between the source and sink. This source-sink pair includes only the data and valid signals.
All data transfers using Avalon-ST interconnect occur synchronously to the rising edge of the associated clock interface.
All outputs from the source interface, including the data, channel, and error signals, must be registered on the rising edge of the clock. Registers are not required for inputs at the sink interface [7].

4. Conclusions

As the main objective of this project is to extract the time from the GPS receiver and extract its local time on the train, the results are mainly based on two factors, i.e, extracting the signal from satellite using GPS Module and transferring the extracted signal from GPS module to the DE2 board.
The main function of the GPS module is to extract the signal from the satellite using antenna. The figure shown below represents the GPS antenna, which extracts the signal from the satellite.
The primary function of the LNA is to set the receivers noise figures and eliminate out of band interference. The GPS module is provided with a 5V external power supply.
The extracted signal is sent to the GPS module using an SMA connector.

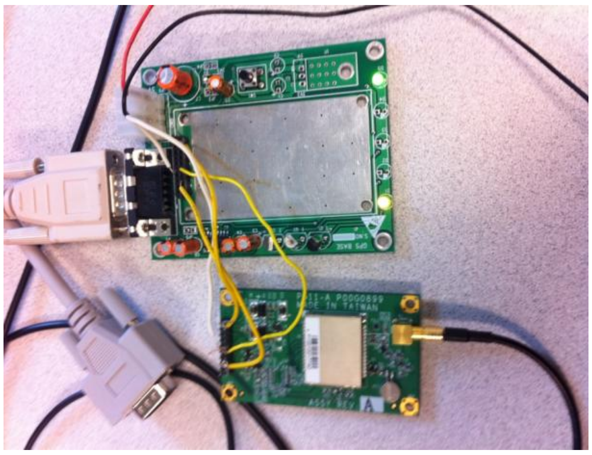
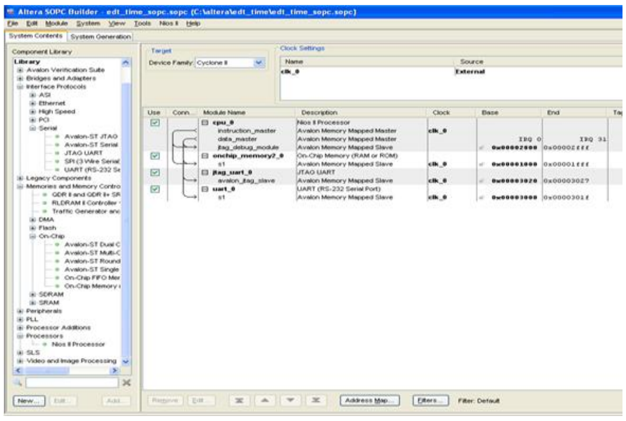


Fig. 12. Working Results of the GPS Module

Fig 12, shows the working results of the GPS module. The Green LED glowing represents the power supply to the GPS module and the yellow LED goes blinking ON and OFF representing the data transmitting and receiving. The data received is sent to the DE2 board using and RS 232 serial cable.

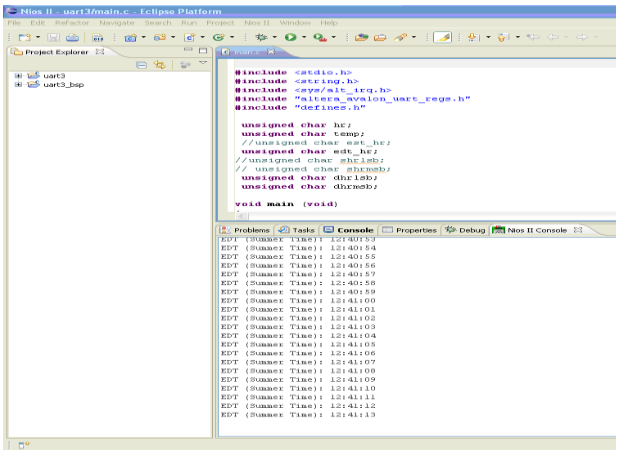

Altera SOPC Builder - edt_time_sopc.sopc (C:\altera\edt_time\edt_time_sopc.sopc)
File Edit Module System View Tools Nios II Help
System Contents
System Generation
Component Library
Target
Device Family: Cyclone II
Clock Settings

Name	Source
clk_0	External

Use	Conn.	Module Name	Description	Clock	Base	End
✓		cpu_0	Nios II Processor			
		instruction_master	Avalon Memory Mapped Master	clk_0		
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31
		jtag_debug_module	Avalon Memory Mapped Slave		0x00002000	0x00002fff
✓		onchip_memory2_0	On-Chip Memory (RAM or ROM)			
		s1	Avalon Memory Mapped Slave	clk_0	0x00001000	0x00001fff
✓		jtag_uart_0	JTAG UART			
		avalon_jtag_slave	Avalon Memory Mapped Slave	clk_0	0x00003020	0x00003027
✓		uart_0	UART (RS-232 Serial Port)			
		s1	Avalon Memory Mapped Slave	clk_0	0x00003000	0x0000301f

Fig. 13. Working Results of Developing System Using SOPC Builder

Fig 13, represents the results of successfully generating a Nios II processor system using SOPC builder.
A working system was developed, which accepts the location from the GPS receiver and extracted its local time. Fig 14 and Fig 15 represents the working results of the project.


Nios II - uart3/main.c - Eclipse Platform
File Edit Refactor Navigate Search Run Project Nios II Window Help
Project Explorer
uart3
uart3_bsp
main.c
Problems
Tasks
Console
Properties
Debug
Nios II Console

```
#include <stdio.h>
#include <string.h>
#include <sys/alt_irq.h>
#include "altera_avalon_uart_regs.h"
#include "defines.h"

 unsigned char hr;
 unsigned char temp;
 //unsigned char est_hr;
 unsigned char edt_hr;
//unsigned char shrlsb;
//  unsigned char shrmsb;
 unsigned char dhrlsb;
 unsigned char dhrmsb;

void main (void)
```

```
EDT (Summer Time): 12:40:53
EDT (Summer Time): 12:40:54
EDT (Summer Time): 12:40:55
EDT (Summer Time): 12:40:56
EDT (Summer Time): 12:40:57
EDT (Summer Time): 12:40:58
EDT (Summer Time): 12:40:59
EDT (Summer Time): 12:41:00
EDT (Summer Time): 12:41:01
EDT (Summer Time): 12:41:02
EDT (Summer Time): 12:41:03
EDT (Summer Time): 12:41:04
EDT (Summer Time): 12:41:05
EDT (Summer Time): 12:41:06
EDT (Summer Time): 12:41:07
EDT (Summer Time): 12:41:08
EDT (Summer Time): 12:41:09
EDT (Summer Time): 12:41:10
EDT (Summer Time): 12:41:11
EDT (Summer Time): 12:41:12
EDT (Summer Time): 12:41:13
```

Fig. 14. Displaying Time in Nios II Console

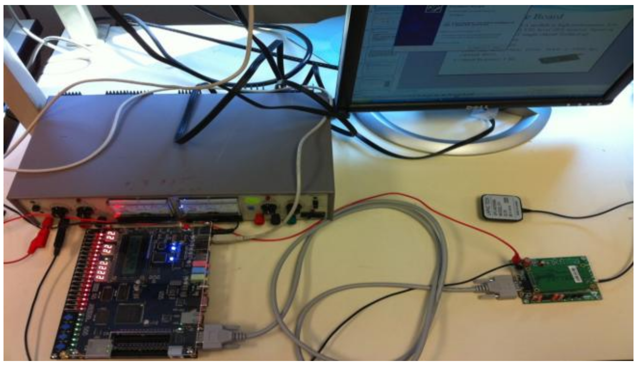


Fig. 15. Working Results of Hardware

References

[1] Hamblen, J.O., Hall, T.S., Furman, M.D., *Rapid Prototyping of Digital Systems*, SOPC Edition Springer Science+Business Media, LLC, New York, USA, 2008.

[2] Robinson, J.; Vafaee, S.; Scobbie, J.; Ritche, M.; Rose, J., *The Supersoft Small Processor*. IEEE Transactiojns on Programmable Logic Conference (SPL), 2010 VI Southern, 2010, pp:3-8.

[3] Fort, B., Capalija, D., Vranesic, Z.G., Brown, S.D., *A Multithreaded Soft Processor for SoPC Area Reduction*. 14th Annual IEEE Symposium and IEEE Transactions on Field-Programmable Custom Computing Machines, FCCM '06. 2006, pp:131-142.

[4] Manjurkha, M., Pathan, Deshmukh, A.Y., *FPGA Implementation of Real Time Processor*. International Journal of VLSI and Signal Processing Applications, Vol.1, No.2, 2011, pp:2231-3133.

[5] Altera Quartus II version 7.2 Handbook, Vol.5: SOPC Builder, http://www.cs.columbia.edu/~sedwards/classes /2010/4840/n2cpu_nii5v3.pdf (accessed Sept. 2011).

[6] Altera Embedded Design Handbook. Ver.2.9, July 2011, http://www.altera.com/literature/hb/nios2/edh_ed_ha ndbook.pdf (accessed Sept. 2011).

[7] Altera Nios II Processor Reference Handbook. Ver.11.0, May 2011, http://www.altera.com/literature/hb/nios2/ n2cpu_nii51002.pdf (accessed Sept. 2011).

[8] Durda, F., Serial and UART Tutorial, FreeBSD: doc/en_US.ISO8859-1/articles/serial-uart/article, Vol.1.14.(accessed Oct. 2010).

[9] Zhang, M., Liu, H.T., *The Design of the Displaying System Based on the SOPC Embedded Chips*. The 2011 Proceedings International Conference on Electric Information and Control Engineering, ICEICE 2011, art. no. 5777471, 2011, pp. 5477-5480.

[10] Alcalde, A.L.P., Ortmann, M.S., Mussa, S.A., *NIOS II Processor Implemented in FPGA: An Application on Control of a PFC Converter*. Power Electronics Conference, COBEP '09. Brazil. 2009, pp:4446-4451.

[11] Glette, K., Torresen, J., Yasunaga, M., Yamaguchi, Y., *On-Chip Evolution Using a Soft Processor Core Applied to Image Recognition*. The First NASA/ESA Conference on Adaptive Hardware and Systems, 2006, pp:373-380.

[12] Mintsis, G., Basbas, S., Papaioannou, P., Taxiltaris, C., Tziavos, I.N., *Applications of GPS Technology in the Land Transportation System*, European Journal of Operational Research, Vol.152, No.2, 2004, pp:399-409.