SERBIAN JOURNAL OF ELECTRICAL ENGINEERING
Vol. 12, No. 2, June 2015, 219-236

UDC: 621.377.624:796.342]:004.451.47 **DOI: 10.2298/SJEE1502219I**

Implementation of Control Logic in the Scoreboard of Tennis

Sandra Ilijin[1], Predrag Petković[2]

Abstract: This paper presents one original solution of control logic for scoreboard in tennis match. The main goal is to simplify the process of recording points. Instead of using six buttons the chair umpire (referee) will use only two control buttons or a joystick to assign a point to a player. The proposed system takes care of all other data processing. The system is designed in Application Specific Integrated Circuits (ASIC) and Standard Application Specific Integrated Circuits (SASIC) technology to demonstrate similarities and differences between two design technologies. Finally it is realized on FPGA type EP2C35F672C6 from Cyclone II Altera's family.

Keywords: Application Specific Integrated Circuit (ASIC), Standard Application Specific Integrated Circuit (SASIC), Tennis scoreboard.

1 Introduction

Tennis as a sport was found in 1873. and since then there is no doubt that it is one of the most popular sport in the world. It is difficult to find a country and a city where tennis is not being played, if not professionally, then at least for fun. If on the other hand people are not playing it, they are certainly following it on TV. Tennis courts have entered a long time ago as one of standard tourist offer, and they have become one of integral parts of sport centres, even in smaller towns. To allow players, and of course the audience to monitor the current result of the match it is necessary to provide adequate monitors – sports scoreboards.

Today there can be seen a lot of different types of scoreboards on tennis courts, from paper ones where chair umpire must continually rotate lists in order to change the result, to digital ones in which it is possible to change the result with pressing on one of the buttons. Currently, mainly digital systems are in use, typically different only in size and material of which they are made, adjusted to the basic functions of strict rules. One interesting solution is the score-tape players wear around their wrist like a wristwatch [1]. Regardless the size of the

[1]HDL Design House, Golsvortijeva 35, 11000 Belgrade, Serbia; E-mail: sandrailijin@gmail.com
[2]Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia; E-mail: predrag.petkovic@elfak.ni.ac.rs

display, basic function is the same and requires a controller operated by a referee. It controls the state points, games and sets of both players. There also must be one reset button which will set all the values to their initial values and to define a start of the match.

This paper describes implementation of an integrated tennis scoreboard, which has a unique controller in order to reduce the number of controls to manage the scoreboard. Aim of this implementation is to facilitate access to the chair umpire that instead of six keys for controlling points, gems and sets use only two by using only two buttons or a joystick. His task is just to assign points to players, and controller automatically updates values of games and sets. This approach significantly helps the umpire, and reduces the risk of displaying wrong result by pressing the wrong button. Implementation of integrated tennis scoreboard is done using Application Specific Integrated Circuits (ASIC) and Standard Application Specific Integrated Circuits (SASIC) technology.

Description of individual blocks is performed using VHDL hardware description language and verified by simulation before implementation. Thereafter it is realized on DE2 board with FPGA chipset EP2C35F672C6 which belongs to the Cyclone II series manufactured by Altera [2]. The same design is implemented in standard cells design flow using CMOS AMI05 technology provided by Alcatel Microelectronics. This was done to illustrate similarities and differences of design flows based on ASIC and SASIC approaches within the first author's master thesis.

The paper is organized in six sections. The following section defines ASIC and SASIC design styles and lists their advantages and drawbacks. The third section describes the design flow in context of ASIC and SASIC design styles. Two subsequent sections describe function and structure of the tennis scoreboard at system and subsystem levels, respectively. The sixth section presents conclusion.

2 ASIC and SASIC Technology

ASIC circuits are manufactured for specific purpose for a specific user. Most commonly the end-user designs a circuit and sends the necessary data to the manufacturer for production. The main characteristic of this style is that only designer knows function and purpose of IC. Thus, it is mostly useful for protecting intellectual property [3]. ASIC circuits can be designed on full-custom or semi-custom manner. Full-custom ASIC considers design process from system down to transistor level. Therefore it is optimised for area and is very time consuming and expensive process that is justified for massive volume IC production. The semi-custom design uses predefined structures so that the design price and time are reduced. The predefined structures are predesigned logic gates. Therefore a designer operates at gate level instead to design transistor by transistor. In order to further reduce time-to-market SASIC design style appeared.

SASIC design style has been created out of ASIC and standard ICs. A typical SASIC circuit consists of an array of disconnected logical structures and programmable connections in the same chip. These logical structures can be of varying degrees of complexity, ranging from the transistor level up to the level of basic combinational and sequential blocks. The produced SASICs are encapsulated and ready to be sold. Therefore they are standard from the perspective of the manufacturer (can be purchased at the store) but do not contain built-in function. The chip receives function after the end-user programmes it. In the category of SASIC circuits field programmable integrated circuits (Field Programmable Gate Array FPGA) stand out.

It is important for a designer to know what of the two design styles to use. One of very, important criteria for comparing ASIC and SASIC is definitely the total cost. The design and prototyping costs of ASIC circuits are much higher than for SASIC circuits. However ASICs are more optimised for area, timing and power consumption than SASICs. Consequently fixed costs are higher for ASIC while costs proportional to production volume are higher for SASIC. Therefore ASIC circuits are mostly used for high volume applications while the SASIC circuits are used for low volume applications. In addition ASICs are using fixed number of the logic gates, and thus very little material is actually wasted compared with SASIC. But, since ASIC circuits don’t have ability to be reprogrammed the costs of fabrication are much higher than in SASIC, and this is the main reason why SASIC circuits are used for low volume applications. Fig. 1 [4] shows a chart on comparing ASIC and SASIC costs after each year of production. It can be seen that after one year of production using ASIC circuits will decrease the costs, while using SASIC is cost effective only in first year after which the costs are much higher.

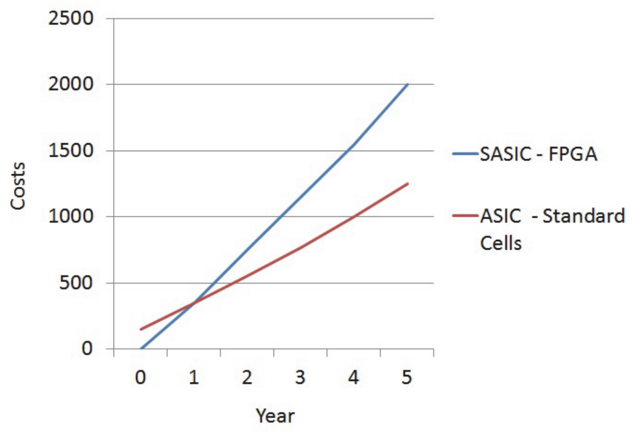

Fig. 1 – *Costs per year for ASIC and SASIC* [4].

Second main advantage ASIC over SASIC circuits is that they can be fully optimized regarding number of gates and logics. SASIC circuits can contain millions of gates for programmable logic design, depending on the type of the chip. As SASICs are not designed for the specific purpose it is hard to imagine an application that will use all resources on the chip: all configurable logic blocks (CLB) and all connections. In order to connect as much CLBs as needed, the design tools can assign some CLBs to enable connection for the entire design. Consequently the design is not fully optimized. Optimization has direct influence on power consumption, the better the optimization the less power consumption. Besides, optimization is in inversely proportional frequency and in directly proportional to power consumption. Fig. 2 [5] illustrates that SASICs will dissipate more power for the same operating frequency than ASICs. The feature to work on higher frequencies with lower power consumption gives advantages to ASIC circuits for complex designs.

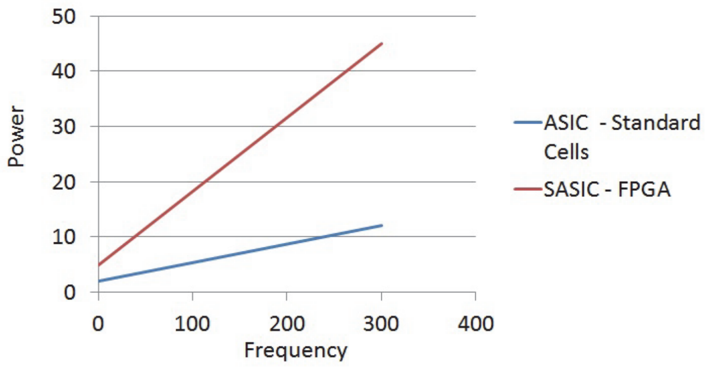

Fig. 2 – *Power consumption compared with Frequency for ASIC and SASIC* [5].

The main disadvantage of ASIC circuits is the need to send the design for production after design and to wait couple of weeks for prototypes. Mask production is very expensive and unsuccessful tries are not desirable and often not afforded. To avoid these additional costs, design must be fully verified before sent to manufacturer. Besides, once fabricated ASIC, cannot be reused by reprograming as FPGAs can.

SASIC circuits can be reprogrammed and rapidly implemented in the lab. This functionality allows easy testing of new logic that, for example, should be added to existing hardware. SASIC circuits allows fully verification of ASIC circuit before it is produced in the factory, or just to verify a part of the design, so reducing costs and time to market.

In conclusion of this section we can say that SASIC are suitable for rapid test of some idea and for low volume production. Their application is limited with complexity, speed and power consumption. In contrary ASICs are favourable for massive production. Their limits in complexity, speed and power consumption are higher when compared to SASIC. Consequently, one should check for cost effectiveness before makes decision about proper design style.

3 Integrated Circuit Design Flow

Typical ASIC design flow requires several general steps that are illustrated on the flow chart in Fig. 3 [3].

A designing process starts with project description using hardware description language (HDL). Today the most popular are VHDL [6] and Verilog [7] languages. VHDL allows simple description of the project at various levels of design abstraction [3].

Pre–layout simulation verifies if the design fully provides the desired functionality. The functional verification of the design that occurs at this point must be as complete and through as possible, before the synthesis process. If all requirements are met, one can proceed to the next step in the design, otherwise the modification must be done in design to meet all the requirements [3].

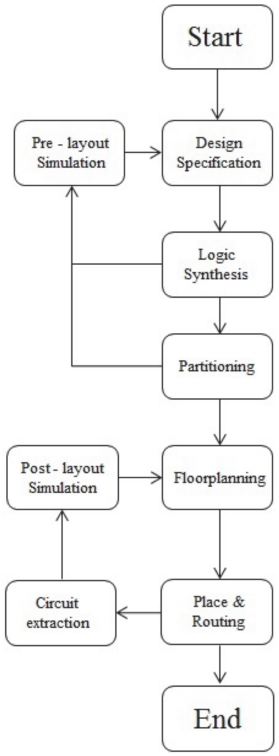

Fig. 3 – *Design flow diagram.*

Logic synthesis translates VHDL code from functional to structural level. Synthesis tools recognizes characteristic parts of code and assigns them appropriate generic logic modules, like NOR circuit with *m* inputs. After synthesis one gets structure, or logic schematic of the circuit but the generic logic blocks are not assigned to a specific technology. Therefore there are no information regarding delays of individual blocks. However these schematics should be verified by simulation before the generic modules are replaced with available logic blocks (cells) in desired technology. The result of synthesis process is so called netlist which represents the network of lines that inter connects primitive elements of the design. With this step the realization of hardware design is completely described. During the synthesis process, the optimization of the design is also included, which tries to summarize certain parts of design if possible, or to minimise combinational logic delay [6].

Partitioning process interconnects generic logic modules with available logic cells that exist in cell library of the chosen technology. For instance, if in the cell library NOR with *m* inputs do not exist, then the circuit is decomposed into required *m*-1 two-input NOR cells [8]. After partitioning the delays of all logic cells are known and the design should be verified by timing analysis.

The following phase is physical design. This phase translates design to silicon. First step in this phase is floor planning.

Floor planning is very important process in the design flow because it affects performance and the chip area [9]. Decisions taken in this step have impact to interconnections lengths and critical delay paths in a circuit. It is very important that each block should be close to appropriate pads on the chip to avoid problems with timing violations caused by signal delay due long connection lines. Modules should be organized so that logic elements clocked with the same frequency are grouped together. In ASIC technology it is necessary to make an outline plan (rectangle area) where the standard cells will be placed. When using SASIC FPGA technology placement depends on available chip architecture and chosen global pins.

Placement process includes mapping the modules on the particular place on the chip. It is very important that modules are well organized on the chip because of the routing process that comes afterwards. Placement has direct influence on area usage and overall timings [9]. Routing is done by tracing connections between cells. After this activity we get precise information about the both length and delay on connections [5]. If the modules are not properly placed, or they are too far from each other, then the interconnection will be too long and could violate for timing constraints. In ASIC, the tracing represents physical connecting of cells, whereas in FPGA circuits it represents assigning particular interconnections to internal buses of the FPGA chip. The dedicated tools, help to minimise area and delays. Place and routing is in most cases done automatically by design tools, not by the designer itself. But there are cases where the tool cannot improve timing, so it is necessary to manually change the place of some module on the chip.

After routing, a complete layout of the circuit is known. Thus, all lengths of connections are defined as well as parameters that describe delays due to parasitic effects. Before the final realization of the circuit, it is necessary to check whether and how these effects affect the basic function of the circuit. Therefore another, so called post-layout verification is needed. To make it more realistic, it is necessary to extract all relevant parameters of the circuit, from the layout. This includes information about parasitic capacitance but also information about mutual influence of certain connections (crosstalk). The result of circuits' parameters extraction from the layout is a net list that allows post-layout circuit simulation.

Post layout simulation represents final verification of complete circuit [3] before prototyping. If the results meet the requirements of the project, then the design of this project is completed and ready for prototyping or programming. If this is not the case, it is necessary to adjust the project. The most common adjustments are related to the addition of a buffer and/or troubleshooting antenna problems that are characteristic of long connections in CMOS circuits.

4 System Description

This chapter describes the complete system for tennis scoreboard, starting from project's specification to description of every sub-module in the system.

This integrated circuit should set and display result in points, gems and sets for both players, with additional option to reset the system. The idea is to simplify the control board for the user (the referee in tennis match) by reducing it to just three buttons. One is reset button while two other serve to increment points to each player. At one moment it is possible to increment score only to one of these two players. (It is recommended to replace these buttons in final realisation with a joystick). Values for points that scoreboard can display are 00, 15, 30, 40 and A (for advantage) in a regular phase of the game. A player wins a game in case of:

- winning four points in a row;
- winning one point after A (advantage);
- winning seven points in Tie Break phase of the game and minimum two points more than other player.

If the player won the game, the score will automatically be incremented by one, and the values for points are reset to 0. Values for games can be 0-6 and 7 if it is a Tie Break (TB) phase. TB phase will occur if both of the players are tied at 6-6 in games, where the points are scored 0,1,2,...,15. TB is won by the player who first won at least seven points and two points of advantage over other player. Set changes are also performed automatically and the score is changed when one of players wins 6 games in regular phase and 7 in TB phase of the game. At the moment of set incrementation, values for both, points and games are reset to 0 and a new set can start. Game ends when one of two players wins two sets. The system is then blocked and the new match can start only after reset.

System in Fig. 4 consists of three control buttons, counters for points, games, sets, points in TB phase and four 7-segment displays grouped in one two-digits units for points, and two 1-digit displays for games and sets respectively.

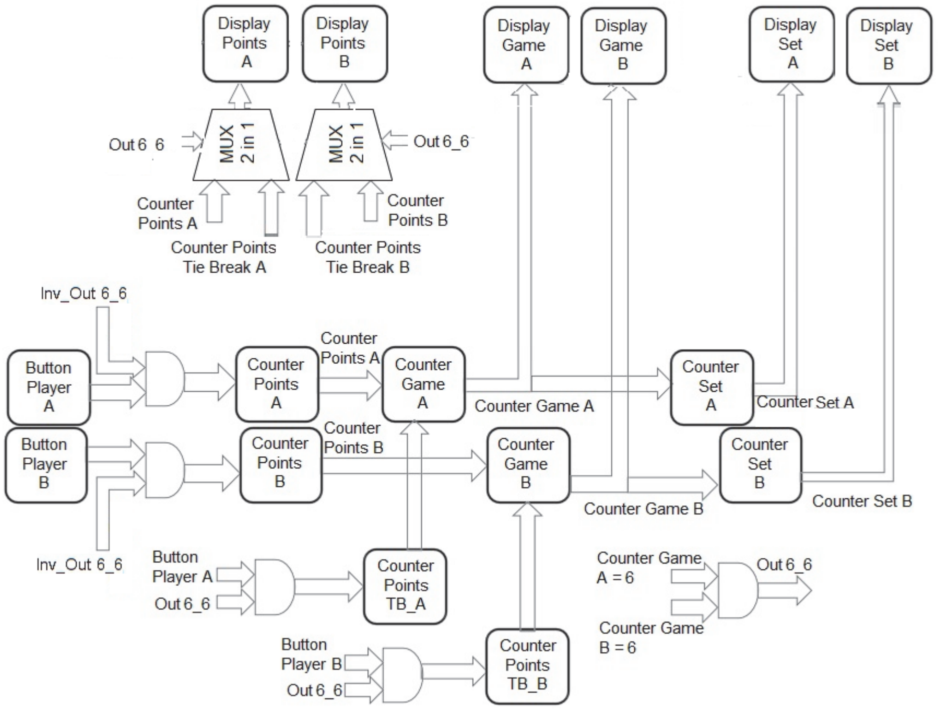

Fig. 4 – *Block diagram of the system.*

When one of the buttons is pressed an impulse is driven to the output of the block Button Player. Depending on the part of the game this signal is sent either to the input of block Counter Point in regular part or to the input of Counter Point_TB in TB phase of the game. This choice depends on the signal "*Out_6_6*" which indicates that both players have the same value in games equal to 6. If this is the case then the points are counted by Counter Points TB.

If not then the points are counted by Counter Points. Independently on the phase of the game – TB phase or regular phase, the points are displayed on Display Point. Impulse coming from point counter indicates that a player won a gem and this signal is passed to block Counter Game. The value of points is passed through MUX 2in1 to block Display Point. The value that will be passed through MUX 2in1 depends on the signal which indicates the phase of the game (TB phase or regular phase). The value from Counter Game goes to blocks Counter Set and Display Game. Output from block Counter Set is passed to block Display Set. Display Point has two, while Display Game and Display Set have one 7-segment digit.

5 Design of Sub-module Counter Points

The next step after defining structure of the system is design of each sub-module. Therefore every block in Fig. 4 is described at functional level using VHDL language. This phase will be illustrated on design of the first block connected directly to control buttons, named Control Point block. It has function to count points for each player. Therefore there are two identical blocks, denoted in Fig. 4 as Counter Point A and Counter Point B. It is a 3-bit counter which counts from 0 to 4. Inputs into these counters are signals coming from buttons *swA*, or *swB*, reset signal *(rst),* clock signal *(clk)*, signal that represents the advantage of other player *(a_in)* and signal that indicates if the other player has value 40 in points *(p40_in).*

Function of block Counter Point is described using two flowcharts presented in Figs. 5 and 6.

Project is completely synchronized with clock signal *Clk* even if this could be done totally asynchronously.

Process starts with pressing one of two buttons which assigns points to one of the players synchronously on rising edge of the clock signal. On rising clock edge the system first checks if the reset signal is active (reset button pressed). In that case, result is set to zero *(Res=0),* otherwise it is in incrementing mode. When player A scores a point, A button is pressed (*sw_A*). The decision of the output depends on the current result:

- If the current result is *Res=3* (the player's A score is 40; Code 40) and player B does not have score 40 (signal *p40_in=0*) or player A has advantage (signal *Res=4*) then, player A wins the game (signal *Game* set to 1) while Counter Points are reset to zero (Res=0)
- If it is not the case it means that *Res*<3 then point value for player A is incremented *Res=Res+1.*
- If player B wins a point (signal *swB=1*) and *Res=4,* then result is changed from "A" to 40 (*Res=Res-1*)

The process finishes, assigning one of the following values to signals:

– when *Res=3*, then player has 40 points, signal *p40_out=1* (otherwise *p40_out = 0*)

– when *Res=4*, then player has Advantage, and signal *a_out=1* (otherwise *a_out=0*)

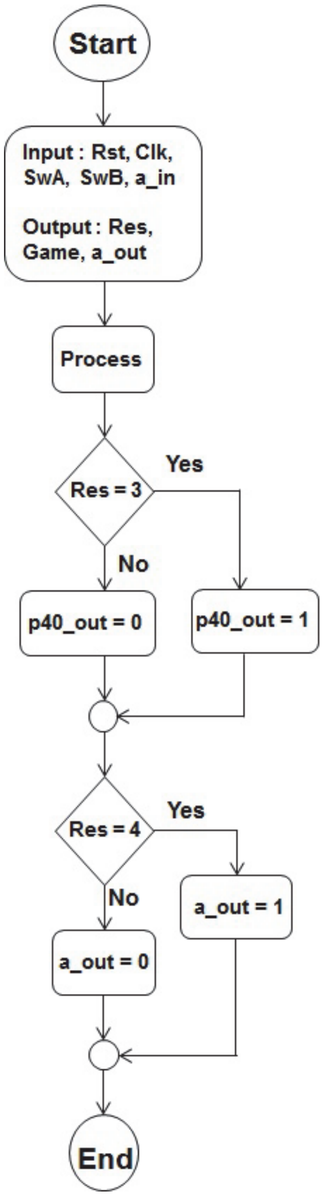

Fig. 5 – *Algorithm of Counter Point Block.*

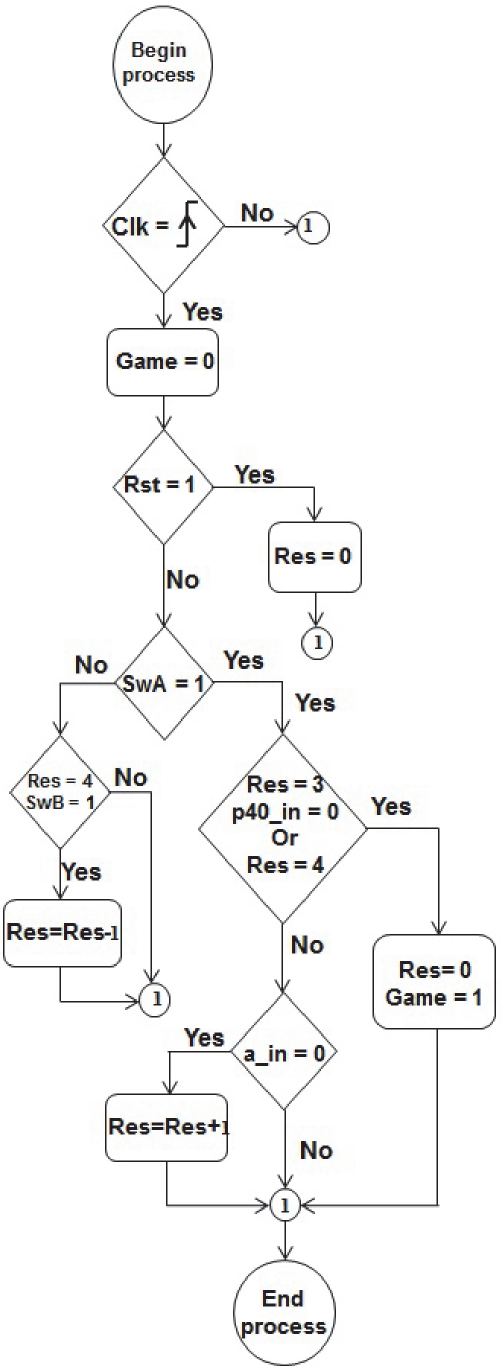

Fig. 6 – *Algorithm of process inside of the Counter Point Block.*

Table 1 shows the coding values of points.

Table 1
Coded point values.

$(Res)_{10}$	$(Res)_2$	Point coded value
0	000	00
1	001	15
2	010	30
3	011	40
4	100	A

5.1 System verification

The functional description in VHDL language is verified by simulation. Test bench is simulated with input signals that cover all critical situations in circuit operation [10]. These situations are increments of points, games and sets. Two enlarged details in Fig. 7 illustrate the moment of changing value of the game after the score of winning game point changes in set values after value 6 is reached in games.

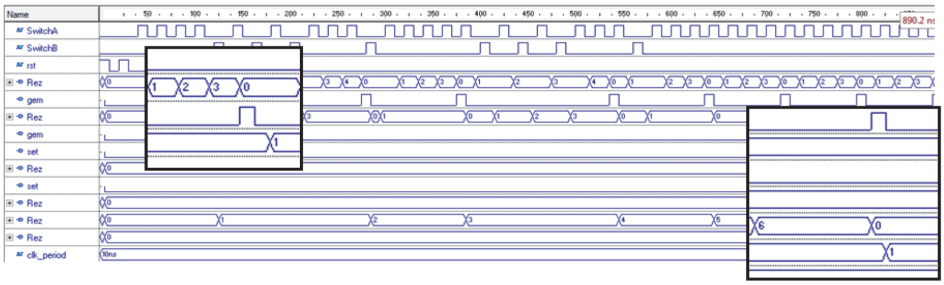

Fig. 7 – *Verification results.*

5.2 ASIC design

Upon successful completion of the system verification on functional level, the next step in design process is synthesis. In order to synthesize ASIC circuit Mentor Graphics' Leonardo Spectrum tool was used [11]. CMOS AMI05 technology provided by Alcatel Microelectronics was chosen for the target technology. Supply voltage of 3.3 V and temperature of 27°C is configured. Timing violations related to the data delay between two registers, delay from input to output, delay from input to register, delay from register to output, clock frequency and etc. **Table 2** summarises selected values.

Table 2
Time delay.

Register – register	20 ns
Input – register	10 ns
Register – output	10 ns
Input– output	10 ns
Clock	20 ns -> 50 MHz

Before synthesis it was necessary to define criteria for area optimization and critical paths. The synthesis resulted with the report shown in **Table 3** and Verilog code describing the system in terms of standard cells. It is used for re – verification, taking into account real delays of all used standard cell. Fig. 8 presents verification results by ModelSim [12] verification tool.

Table 3
Synthesis results in ASIC.

Technology	AMI05
Pin number	60
Connection number	397
Instance number	365

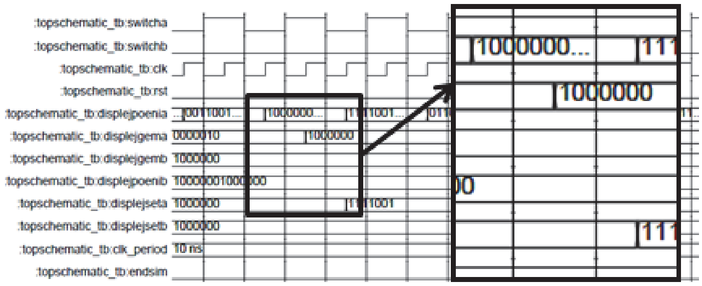
:topschematic_tb:switcha
:topschematic_tb:switchb
:topschematic_tb:clk
:topschematic_tb:rst
:topschematic_tb:displejpoenia
:topschematic_tb:displejgema
:topschematic_tb:displejgemb
:topschematic_tb:displejpoenib
:topschematic_tb:displejseta
:topschematic_tb:displejsetb
:topschematic_tb:clk_period
:topschematic_tb:endsim
10 ns

Fig. 8 – *System verification results after synthesis.*

After the synthesis and verification are carried out, the next step is physical design of the chip.

We want to implement the whole system as a macro cell. Therefore it was necessary to define approximate dimensions and aspect ratio of the cell. Physical design was completed automatically using IC Station design tool [11]. Fig. 9 illustrates layout of the designed macrocell that occupies area of 6.22 mm^2.

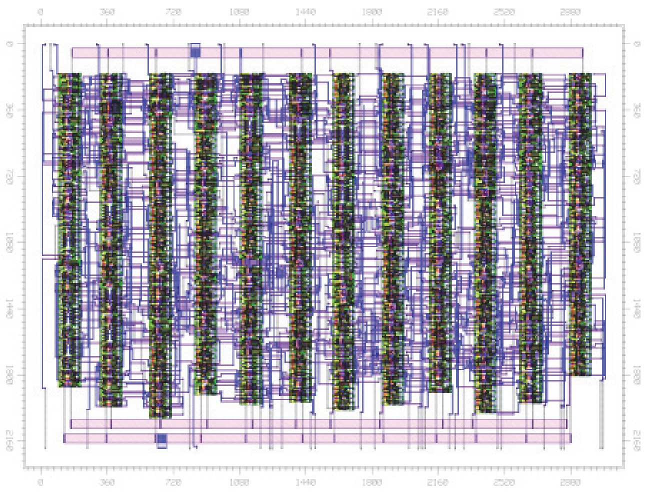

Fig. 9 – *System layout.*

5.3 SASIC design

Synthesis process for SASIC circuit differs significantly from implementation based on ASIC design technology.

We used FPGA chip EP2C35F672C6 which belongs to the Cyclone II Altera manufacturer to illustrate the application of SASIC design. FPGA consists of two-dimensional fields of logic blocks interconnected by programmable connections [13]. FPGA implementation requires converting logic function in form of truth table. Afterwards, its content is written to SRAM cells [9]. Function generator implemented on SRAM basis is known as LUT (Look Up Table) [14]. Using this approach delay of any function that fits in LUT is the same, regardless of its complexity.

For a functional level's system description, we used the same VHDL code as in ASIC design style. Physical verification of SASIC based integrated circuit is possible by programming the chip on-board. Therefore the VHDL code was loaded in Quartus II tool, for circuit synthesis. Similarly in ASIC design, certain parameters are set for optimization during the synthesis process. The user can choose optimization by delay, by resources or balancing between both criteria. In case of optimization by delay, the goal is to minimize the delay on the longest path interconnecting registers or register and FPGA pin. In case of the area optimization the goal of synthesis is to occupy as small number of resources (CLB and wires) as possible. **Table 4** summarizes synthesis results.

Table 5 shows the values corresponding to the number of combinatorial functions in the system. Fig. 10 illustrates part of the scheme obtained after the synthesis process.

Table 4
Synthesis results in SASIC.

Family	Cyclone II
Pin number	60
Register number	32
Combinatorial function	169

Table 5
Number of combinatorial functions.

LUT inputs	169
4-input functions	93
3-input functions	57
2-input functions	19

Table 6
Number of combinatorial functions.

Chip	EP2C35F672C6
Pin number	60/475 (13%)
Register number	32/33126 (<1%)
Combinatorial function number	169/33126 (<1%)
Logic element number	170/33126 (<1%)

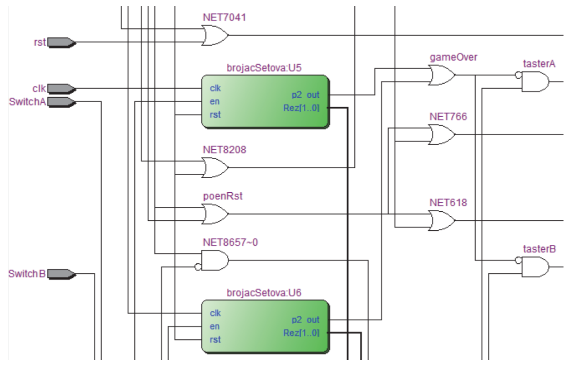

Fig. 10 – *Part of the diagram resulted after synthesis.*

After synthesis the next step is placement phase. It is aimed to allocate CLB block and/or special internal parts (multipliers, PLLs, internal memories) to particular functional blocks obtained from synthesis. The final designing phase is routing. It assigns internal bus of FPGA chip to individual routes in order to obtain the smallest possible delay. After routing the user knows all information about occupancy of the chip. **Table 6** presents the report corresponding to the described project.

Once placement and routing are completed all data related to timing constraints are known. Therefore it is possible to accomplish final verification. The difference in comparison to the functional simulation is that now the information about delays on routes is available. Eventually, the FPGA chip needs to be programmed. The most common technique for chip programming is based on SRAM cells, which means that the configuration is kept externally. FPGA can be programmed when mounted on printed circuit board that contains PROM or flash memory. After obtaining power or on demand, it will load appropriate program with desired configuration. For real time verification we used Altera DE2 [15] evaluation board.

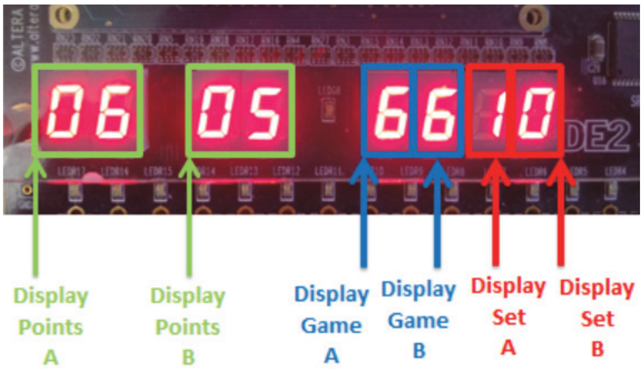

Fig. 11 – *Display of system on* DE2 *Altera board.*

6 Conclusion

This paper described designing of control logic for monitoring and displaying result in tennis match. The aim was to provide the chair umpire with a simple two-button tool so to be able to assign points by pushing a button corresponding to the player who scores a point. Additional goal was to compare effectiveness of ASIC and SASIC design styles. Firstly, the project was realised using ASIC design flow assigned to CMOS AMI05 technology. This, little bit obsolete technology was the only one available with the student version of the accessible design kit. The whole design occupied area of 6.22 mm^2.

The design was repeated on FPGA design flow using the same VHDL description that was the entry into ASIC design. Final implementation was on EP2C35F672C6 chip from Cyclone II Altera's family that was mounted on Altera DE2 evaluation board.

One of the key design features is total cost of the produced device. The total cost depends on fixed and costs proportional to the production volume. The proportional costs are much higher for FPGA, while the fixed costs are enormously higher for ASIC. The highest share in ASIC price has cost of the masks. The best small volume multi-project cost for ASIC prototyping is 500 €/mm^2. (This stands for universities). So, one ASIC sample would cost 3,110 €! Simultaneously price of one FPGA is around 100€ [16].

Therefore it comes out that with volume of only 32 units the total price for ASIC and SASIC will be the same. Of course, this is not true because prices for commercial customers are much (more than ten times) higher than for universities. Nevertheless total cost will depend on the production volume.

Therefore it is essential to evaluate potential market size before the final decision about the used design style.

However, needs of the regional market can reach 1000 units. With this figure in mind one may, conclude that ASIC design style would be cost-effective application.

The next steps in the development of this system would be to reduce the hardware resources in order to make the system as cheep as possible. With this approach it would be affordable for small tennis courts where people usually play tennis for recreation and wants to track the current result of the play. In the case where there is no chair umpire, which is the case with the amateurs' players, it is necessary to allow players themselves to control the result on the scoreboard using wireless controls. One of possible solution is to design the system to fit into a bracelet (bracelet with buttons) [1]. Players (at least one player) would wear the bracelet that would have communication with the display.

7 References

[1] http://scoreboard.net

[2] http://www.datasheetarchive.com/EP2C35F672C6-datasheet.html

[3] P.P. Petkovic: Design of CMOS Integrated Circuits with Mixed Signals, Faculty of Electronic Engineering, University of Nis, Nis, Serbia, 2009. (In Serbian).

[4] Keith: ASICs vs FPGAs, Nov. 2010. Available at: http://blog.customsiliconsolutions.com/uncategorized/asics-vs-fpgas.

[5] R. Mosher: FPGA Prototyping to Structured ASIC Production to Reduce Costs, Risk and TTM, 2005. Available at: http://www.design-reuse.com/articles/13550/fpga-prototyping-to-structured-asic-production-to-reduce-cost-risk-ttm.html.

[6] P.J. Ashenden: The Designer Guide to VHDL, Elsevier, NY, USA, 2008.

[7] S. Palnitkar: Verilog HDL: A Guide to Digital Design and Synthesis, Prentice-Hall, NY, USA, 2003.

[8] G.Lj. Đorđević: Microsystem Architecture, Faculty of Electronic Engineering, University of Nis, Nis, Serbia, 2010. (In Serbian).

[9] K. Golshan: Physical Design Essentials: An ASIC Design Implementation Perspective, Sringer, NY, USA, 2007.

[10] https://www.doulos.com/knowhow/perl/testbench_creation/

[11] Manual for ADK Design Kit Version 1.6: Designing ASICs with the ADK Design Kit and Mentor Graphics Tools, 2005.

[12] ModelSim Advanced Verification and Debugging SE Tutorial, Version 6.0.c, 2005.

[13] http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm.

[14] I. Stojanović: ASIC Design Circuit for Testing VGA Display using Standard Cells in AMI05 Techonolgy, Faculty of Electronic Engineering, University of Nis, Nis, Serbia, 2013. (In Serbian).

[15] ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf.

[16] http://www.europractice-ic.com/.