Improving the Design Performance of Field Programmable Gate Array Devices with Efficient Approach to Measure Power Consumption

Esteve Hassan
Electrical and Computer Engineering Technology
Mohawk College of Applied Arts and Technology
Hamilton (ON), Canada
e-mail: esteve.hassan@mohawkcollege.ca

Bilal Al Momani
Electrical and Computer Engineering Technology
Mohawk College of Applied Arts and Technology
Hamilton (ON), Canada
e-mail: bilal.al-momani@mohawkcollege.ca

***Abstract*— The study in this paper is focused on the improvement of a Field Programmable Gate Arrays (FPGA) based design using a hierarchical analysis tool offered by XILINX PlanAhead™. During this work, PlanAhead software is used to address any problems on the physical side of our FPGA design flow to add more modules visibility and control. The design application is a telemetry system intended for health monitoring applications. FPGA is used as the brain control unit at both transmitter and receiver sides. The transmitter side is recording data packets through external interfaced sensors. Verilog Hardware Description Language (Verilog-HDL) has been used to implement the various functionalities required by the FPGA device. The system performance as shown in the results has been optimized using a recent comprehensive tool to reach and maintain the goals of the design. The power performance of the FPGA-based design will be assessed using the XILINX Xpower tool. A Modelsim Code coverage feature has been incorporated to make sure that the test bench will cover all the nets branch statements of the design and create the most accurate Value Change Dump (VCD) file for the power consumption assessment process.**

Keywords- FPGA; Telemetry system; FPGA design improvement; power assessment.

I. INTRODUCTION

The challenge is to raise or at least maintain the present level of health care providers without ending up in an uncontrolled cost explosion of services. The increasing number of researchers and manufacturers who are working on a new generation of wireless technology applications for the medical field has led to improved quality and reduced cost of patient care. One of the areas in healthcare that best lend itself to wireless technology is patient monitoring, also known as wireless telemetry. Due to system complexity, the use of FPGA for this vital application has become widely common.

Electronic systems development is becoming more and more complex, fast, powerful, and power-consuming. Indeed, transistor miniaturization dramatically increases the power consumed by a whole chip [1,2]. The main consequences of this trend are the addition of elaborated cooling circuits and the reduction of battery life for the embedded systems. As for timing and die area, power consumption becomes a critical constraint for electronic system design. A previous study [3] has demonstrated the beneficial effect of power optimization at high-level; it is then necessary to develop high-level estimation tools which use power models for all kinds of components (Application-Specific Application Circuit (ASIC), FPGA) in a system. Playing many important roles in recent applications, FPGAs devices are used in a wide scale of designs ranging from small glue logic replacement to System-on-Chip. The main advantage of FPGA compared to ASIC chips is the flexibility: a design can be reprogrammed partially or totally in-situ. This functionality is realized by a configuration plan and requires a large number of transistors for Static Random Access Memory (SRAM) FPGA; therefore, the drawback is important static power consumption. Moreover, FPGA builders are currently improving this circuit characteristic to facilitate their integration in System on Chip (SoC). The health care field became one of the most recent applications of the FPGA designers [4].

The PlanAhead software provides insight into the data flow of the design by displaying I/O interconnect as well as physical block net bundles [5]. Timing constraints can be then modified within the PlanAhead environment. These analysis results can help to determine what logic should be grouped and floor planned. Paths can be logically sorted, grouped, and selected for floorplanning. TimeAhead environment can also be leveraged with imported timing results from the timing analyzer tool within Xilinx Integrated Synthesis Environment (Vivado™) software [5]. TimeAhead is useful to validate and optimize the constraint set before running any Vivado implementation tools. In addition, it provides visual aids to comprehend the physical implementation results. Design rule checks (DRCs) are provided to catch errors early. It also flags designs that do not properly take advantage of certain device resources, such as the dedicated registers of the XtremeDSP™ slice.

Design solutions can be addressed quickly by visualizing area problems, either in the register transfer logic (RTL) or on the physical implementation side, without having to continue RTL and synthesis iterations. FPGA vendors are facing the difficult task to accurately specify the energy consumption information of their products on the device data sheets because the energy consumption of FPGAs is strongly

dependent on the target circuit including resource utilization, logic partitioning, mapping, placement, and route. While major Computer-Aided Design (CAD) tools have started to report average power consumption under given transition activities, energy optimal FPGA design demands more detailed energy estimation. This work aims to present a useful methodology for estimating the power consumption of an FPGA-based system designed for medical applications. Modelsim code coverage capability will be used to investigate the different styles of test bench coding on the overall power consumption estimation of the FPGA device.

In Section II, a system overview is presented. Sections III-VI are outlining the design methodology for FPGA devices using PlanAhead. ModelSim code coverage is explained in Section VII and Section VIII explains the accurate FPGA power estimation. Section IX is giving the merits of the new method to perform an accurate power consumption assessment. Finally, conclusions are drawn in Section X.

II. SYSTEM OVERVIEW

The main blocks of the transmitter side FPGA are shown in Fig. 1. The different units of the system were coded with Verilog HDL simulated with ModelSim and implemented with Vivado. The final implementation was targeting the Virtex-7 device since it provides the various features that solve the designer's challenge throughout the entire system. The transmitter FPGA consists mainly of an SPI (Serial Peripheral Interface), RLE (Run Length Encoding) compressor, and framer units. The operation of the system units and the flow of data through the system are controlled by the main FSM (Finite State Machine) controller.

On the receiver side, a data recovery unit is needed to extract the clock from the received bitstream. The de-framer and the RLE decompresser blocks are designed to reconstruct the original data bytes sent by the transmitter.

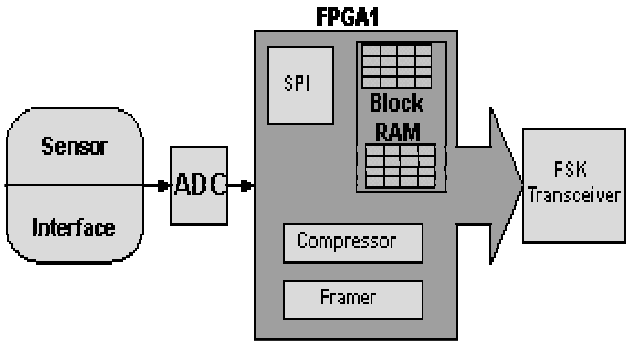

Figure 1. Building blocks of the transmitter FPGA

A. SPI main units

An efficient SPI unit has been modeled, as shown in Fig. 2. The Master out Slave in (MOSI) signal has been omitted from the design based on the hardware requirements where data only needed to be transferred from the ADC to the FPGA system. The main units of the SPI are functioning as follows:

1. Clock Divider Unit: Divides the system clock by a certain factor to generate the required SPI clock frequency.
2. Data out clock synchronizer: used to generate both the rising edge (dout7) and the falling edge (dout16) of the ADC clock.
3. ADC Enable unit: triggered on when the start_conv signal is asserted to generates the following signals:
   a. Capture signal to capture data transfer from the ADC to the SPI register after each byte transfer.
   b. Increment signal used to change the address inside the Block RAM unit.
4. Slave Chip Select (CS).
5. SPI Register Unit: contains the SPI serial in/parallel out register, which is enabled when the capture signal is asserted and receives input serial data through ADC_Din signal. Spiout (output) signal carries the information data bytes to the Block RAM unit.
6. Distributed Block RAM: stores the data bytes in locations determined by the increment signal.

Typically, test benches have become the standard method to verify HDL designs. Test benches invoke the functional design, and then simulate it. Accordingly, an efficient test bench has been written to mimic the behavior of the ADC and verify the operation of the SPI system units.

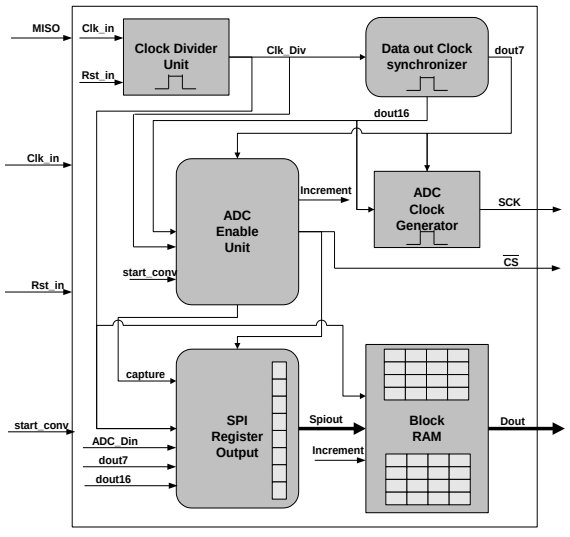

Figure 2. Block units of the developed SPI inside the FPGA

III. DESIGN FLOW USING PLANAHEAD (PA)

The PlanAhead tool sits between synthesis and the Vivado place and route (P&R) tools as shown in Fig. 3.

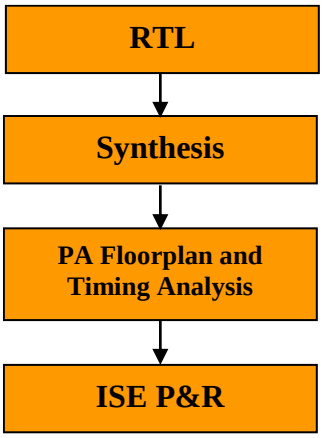

Figure 3. FPGA design flow using Plan Ahead

To increase the design performance, the HDL code has been written taking into consideration the following comments:

The design has been partitioned at the RTL level, such that critical timing paths are confined to individual modules. Critical paths that span large numbers of hierarchical modules can be difficult to floorplan. The outputs of all modules have been registered to help limit the number of modules involved in a critical path.

Dividing large hierarchical blocks into smaller RTL units to avoid the possibility of having long paths, which makes the floorplan a difficult task.

Through analysis and floorplanning, physical constraints are applied to help control the initial implementation of the design. PlanAhead is also used after implementation to analyze the placement and timing results in order to improve the floorplanning and complete the design.

A very optimized Verilog code has been written to describe the different block units of the design since it is based only on the instantiation of the basic units that can be invoked directly from the library. This is essential in the design process to remove any complexity from the model, which makes it easy to understand and debug. In addition, unnecessary resources are not added by the synthesize tool to the code, which leads to a power-efficient model design. An example of such code for the ADC chip selection is shown in Fig. 4.

IV. PLANAHEAD IMPLEMENTATION

In this section, the PlanAhead implementation on the transmitter side FPGA will be presented. The on-chip design partitions are referred as physical block (Pblocks).

With PlanAhead software, the utilisation estimates of the device resources viewed below in Fig. 5.

```
//==== Multiplexer of the ADC Chip Select====/

AND2 and2instance3 (.O (and3out),
            .I0 (fdeout5),
            .I1 (Mod_Fjkout)
);

AND2 andinstance4 (.O (and4out),
            .I0 (fdeout4),
            .I1 (inv3out)
);

INV invinstance3  (.O (inv3out),
            .I (Mod_Fjkout));

OR2 OR2_instance1  (.O (En_Adc),
             .I0 (and4out),
             .I1 (and3out));
```

Figure 4. Verilog_HDL sample code form the SPI design

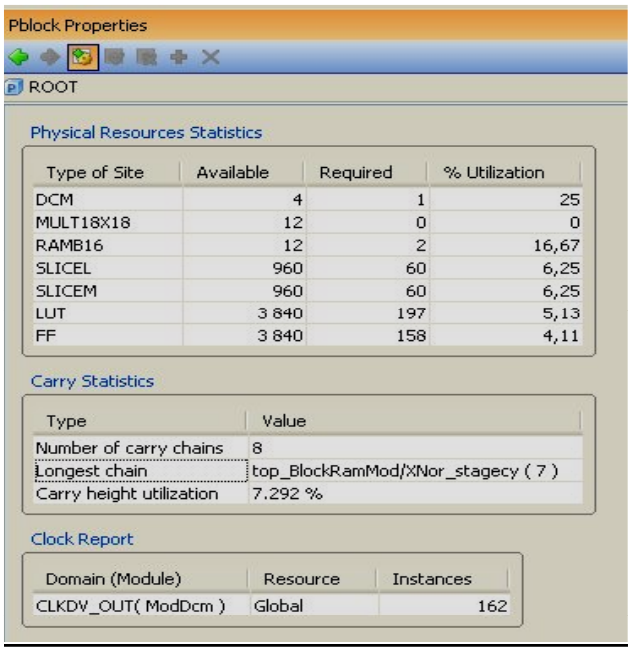

Figure 5. device resources used for the target system design

In order to get a good schematic-level view of the key portions of the design, PlanAhead software has been used for this purpose as well. The schematic of the design top-level is presented in Fig 6. Such views can be a valuable aid in understanding how the modules of the design are connected to each other. It is recommended with PlanAhead to view and analyze the hierarchy of the design. Such a view can be useful to implement the best floorplan and also may indicate

the location of the longest timing path (critical path). In Fig. 7, the hierarchy of the design is displayed.

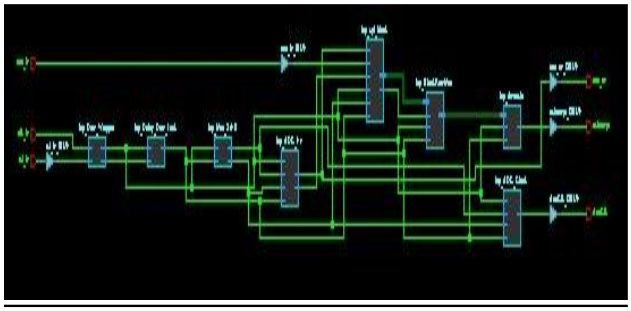

Figure 6. Design top-level schematic

PlanAhead software has an embedded static timing analysis engine and environment called TimeAhead. With this feature, timing estimations can be utilized at various stages of design implementation. The longest path has been explored and highlighted in Fig. 8. This step is necessary for the next coming stages to improve the floorplan for better performance. It is worth mentioning here that in most cases the longest path is associated with the module of the biggest size in a fully synchronous design. In the hierarchy view shown in Fig. 7, the module top_BlockRamMod is obviously the one, which has the longest path.

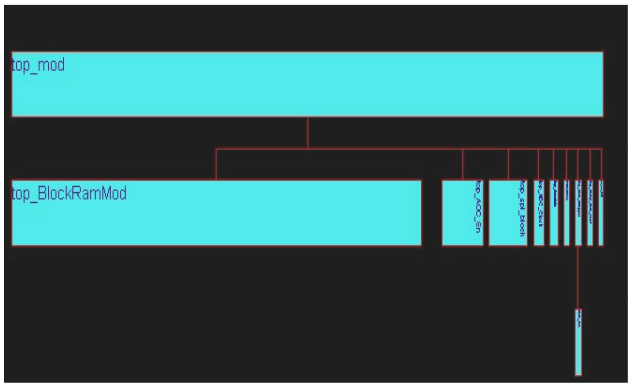

Figure 7. Module hierarchy view from PlanAhead

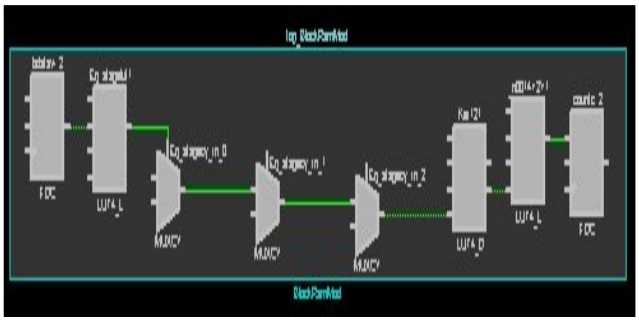

Figure 8. The longest logic delay path

V. FLOORPLAN STRATEGIES

A. Problem Description

Assume we are given a set of modules, each of them having an associated resource requirement vector ϕ = *(n1,n2,n3)*, which means this module requires *n1* CLBs, *n2* RAMS, and *n3* multipliers. The FPGA floorplanning problem is to place modules on the chip so that each rectangular region assigned to a module should satisfy its resource requirements.

For example, we have 6 modules, and their resource requirement vectors are ϕ_1 = (12,2, l), ϕ_2 = (30,4,4), ϕ_3 = (15,1,1), ϕ_4 = (24,4,4), ϕ_5 = (18,2,2), ϕ_6 = (30,2,2). Fig. 9 is a feasible floorplan for these modules, which shows a slicing structure.

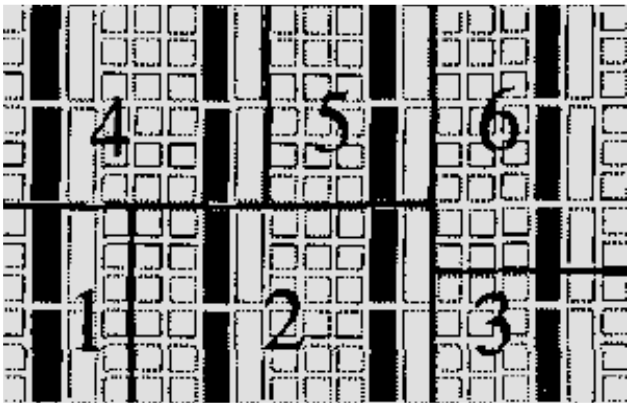

Figure 9. Example of Floorplan slicing structure

In the following sections, the use of PlanAhead tool to implement our floorplan design strategies will be explained.

B. Logic Placement Before Floorplanning

Fig. 10 shows how the logic modules of the design are distributed by the Vivado tool inside the chip. The FPGA I/O resources displayed as thin rectangles just outside the device are the Input/Output (I/O) pads. I/O banks are displayed as thin rectangles just outside the I/O pads. Digital clock managers (DCMs) are shown graphically as rectangles along the I/O ring. The clock I/O pins are shown as filled rectangles. The interior of the device is broken up into smaller rectangles called tiles. These tiles contain placement sites for the different types of logic primitives pertinent to the architecture being used.

In Table I, a summary of the device utilization before implementing floorplanning is given. The number of slices has been highlighted since it is important figure that indicates the device area covered by the design. Such figure will be used later on for performance comparisons.

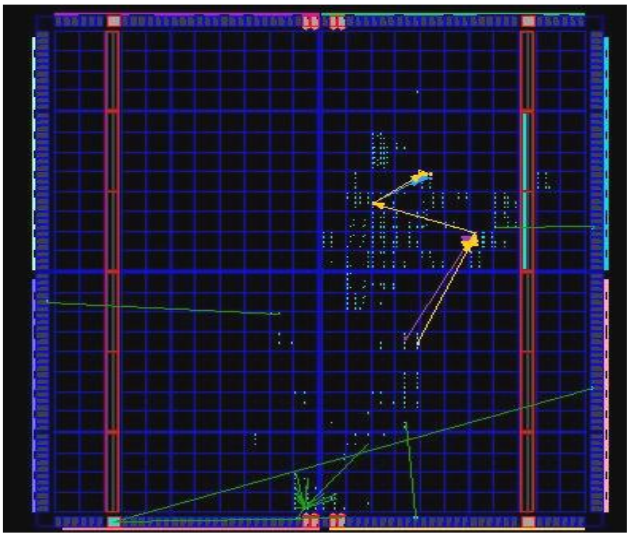

Figure 10. Logic placement before Floorplanning

TABLE I: DEVICE UTILIZATION BEFORE USING PLANAHEAD TOOL

Device Utilization Summary		
Number of BUFGMUXs	2 out of 8	25%
Number of DCMs	1 out of 4	25%
Number of External IOBs	6 out of 173	3%
Number of LOCed IOBs	2 out of 6	33%
Number of RAMB16s	2 out of 12	16%
Number of Slices	143 out of 1920	7%
Number of SLICEMs	1 out of 960	1%

To get a closer view of the timing properties of the current design, a timing report has been generated as in Fig. 11. The following is the flow used to floorplan our design. The netlist file is generated first inside the Vivado using XST (Xilinx Synthesis Technology). Then the netlist file is used to create a new project in PlanAhead. After floorplanning the design, the netlist and the User Constraint File (UCF) will be exported back into Vivado environment and the timing report can be read from the place and route (P&R) results.

```
------------------------------------------------------------------------------------
Constraint                                | Requested | Actual    | Logic
                                          |           |           | Levels
------------------------------------------------------------------------------------
TS_clk_in = PERIOD TIMEGRP "clk_in" 20 ns | N/A       | N/A       | N/A
 HIGH 50%                                 |           |           |
------------------------------------------------------------------------------------
TS_top_Dcm_Wrapper_Dcm_Ins_CLKDV_BUF = PE | 640.000ns | 7.834ns   | 4
RIOD TIMEGRP        "top_Dcm_Wrapper_Dcm |           |           |
_Ins_CLKDV_BUF" TS_clk_in * 32 HIGH 50%  |           |           |
------------------------------------------------------------------------------------
```

Figure 11. Timing report before implementing PlanAhead

From Fig. 11, it can be noticed the large difference between the requested time for running the design (640 nsec) and the actual time given by the P&R report (7.834nsec).

C. *Floorplanning Techniques Implementation*

In this section, PlanAhead software will be used to implement two different strategies of floorplanning. The output results of the implementation will be presented and discussed. The main goal is to show how PlanAhead can be employed successfully to optimize the placement area of our design logic.

The first strategy is based on pulling the whole design logic to be focused in one Pblock. This has been done as shown in Fig. 12, where all the modules have been placed in one rectangular at the bottom right corner of the device chip.

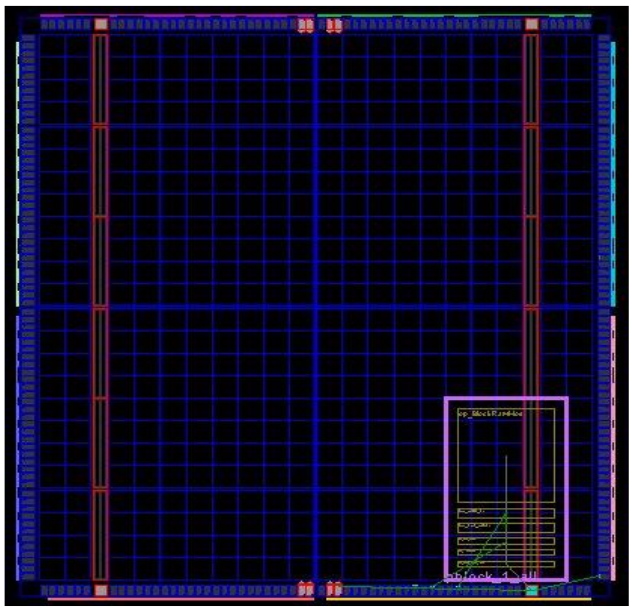

Figure 12. Logic placement after FloorPlanning using 1 Pblock strategy

The summary of the device utilization after implementing this strategy is given in Table II.

TABLE II: DEVICE UTILIZATION USING PLANAHEAD STRATEGY 1

Device Utilization Summary		
Number of BUFGMUXs	2 out of 8	25%
Number of DCMs	1 out of 4	25%
Number of External IOBs	6 out of 173	3%
Number of LOCed IOBs	6 out of 6	33%
Number of RAMB16s	2 out of 12	16%
Number of Slices	128 out of 1920	7%
Number of SLICEMs	1 out of 960	1%

It is clear from the number of slices that the design area has been compressed by more than 10%. To have a complete

picture, a timing report has been generated using the same procedure mentioned in the previous section and it is shown in Fig. 13.

Constraint	Requested	Actual	Logic Levels
TS_clk_in = PERIOD TIMEGRP "clk_in" 20 ns HIGH 50%	N/A	N/A	N/A
TS_top_Dcm_Wrapper_Dcm_Ins_CLKDV_BUF = PERIOD TIMEGRP "top_Dcm_Wrapper_Dcm_Ins_CLKDV_BUF" TS_clk_in * 32 HIGH 50%	640.000ns	9.117ns	3

Figure 13. Clock report after implementing the PlanAhead using 1Pblock strategy

One interesting comment that can be drawn from Fig. 13, is the actual time has been increased from the one before implementing this strategy. This can be explained as the design now become more compact in a smaller area, which leads to having data congestion. Despite this increase in time, it will not have any impact on our design performance since we are meeting the required time far below.

In the second strategy, two Pblocks will be used to place the design modules and the necessary connections between these two blocks can be visible in PlanAhead. The placement of the 2Pblock is shown in Fig. 14.

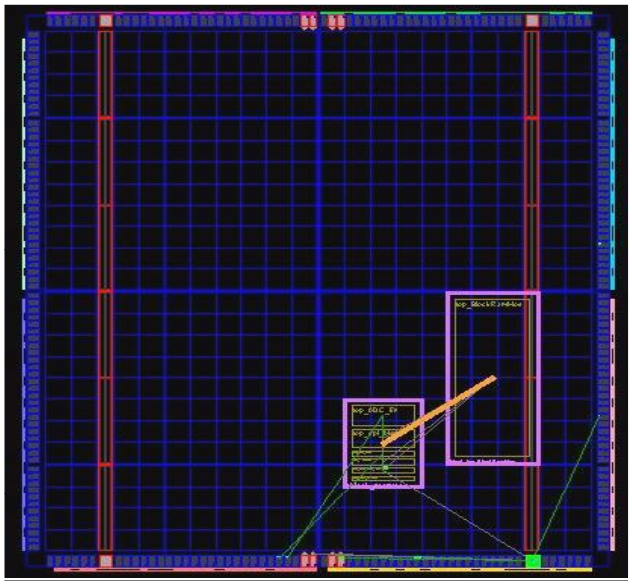

Figure14. Logic placement after FloorPlanning using 2 Pblocks strategy

In the same manner, Table III is showing the device utilization after implementing the second strategy. As expected the design area has been slightly compressed with this strategy.

TABLE III: DEVICE UTILIZATION USING PLANAHEAD STRATEGY 2

Device Utilization Summary		
Number of BUFGMUXs	2 out of 8	25%
Number of DCMs	1 out of 4	25%
Number of External IOBs	6 out of 173	3%
Number of LOCed IOBs	6 out of 6	33%
Number of RAMB16s	2 out of 12	16%
Number of Slices	140 out of 1920	7%
Number of SLICEMs	1 out of 960	1%

Finally, the timing report of this implementation is showing less value for the actual time as presented in Fig. 15. With this strategy, a larger area is provided for the design, which reduces the possibility of having a high data congestion rate.

Constraint	Requested	Actual	Logic Levels
TS_clk_in = PERIOD TIMEGRP "clk_in" 20 ns HIGH 50%	N/A	N/A	N/A
TS_top_Dcm_Wrapper_Dcm_Ins_CLKDV_BUF = PERIOD TIMEGRP "top_Dcm_Wrapper_Dcm_Ins_CLKDV_BUF" TS_clk_in * 32 HIGH 50%	640.000ns	8.134ns	4

Figure 15. Clock report after implementing the PlanAhead using the 2 Pblocks strategy.

Another highlighted field in the utilization summary tables is the one that shows the locked Input/Output Blocks (IOBs). Before implementing floorplanning, only two pins have been given positions in the UCF, these are the clock and reset pins. As the physical locations of these two pins were compulsory, the floorplanning was influenced by this fact. The closest physical location to the design has been chosen to lock the other pins after floorplanning. That is why the number of locked pins is complete in Tables II and III.

By doing a comparative study for the two suggested floorplanning techniques, it is obvious that the first strategy gives the optimum design area solution. On the other hand, the speed of data transfer might be affected due to the data congestion that occurs between the modules but this would not be a serious issue in our case. The second floorplanning strategy slightly reduces the area since more space will be provided for the design modules. The data speed has been improved in compared with the first strategy but still our design meets the requested timing requirements in all the cases.

VI. FPGA POWER ESTIMATION

Power consumption is mandatory information in modern digital system design.

Chip vendors are naturally in charge of supplying energy consumption information of their products on the device datasheets. However, it is not possible for vendors to specify power consumption information of SRAM-based FPGAs because it is not only dependent on the target device and operating frequency but is highly dependent on the design and operating conditions. Power consumption is strongly dependent on the target circuit including resource utilization, low-level features such as logic partition, mapping, placement, and route. Here a new power estimation methodology is introduced on the FPGA after the design has been optimized with PlanAhead tool as presented by previous sections.

A. Related Work

For FPGA, some methodologies and models have already been developed to estimate the power consumed specifically by the logic elements. For example, a probabilistic model is proposed by [6]; developed for a CAD tool, this model estimates, at the transistor level, the 0.18 μm Complementary Metal Oxide Semiconductor (CMOS) FPGA power consumption based on place and route. The switching activity used to calculate the dynamic power is determined by the transition density of the signal. The static power is evaluated by a sub-threshold current estimation. The resulting absolute error of this model is 23% compared to measurements. Some techniques are proposed in [7] to reduce both static and dynamic power consumption like drowsy mode, clock gating, guarded evaluation, counter and state machine encoding, but no estimation model is proposed. For a design in Virtex-7, [8] proposes an estimate of the dynamic power consumed by logical elements after routing. Lastly, [9] has presented a Register Transfer level power estimator based on the determination of wire length and switching activity with an average error of 16.2%. The first parameter is calculated by applying Rent's rule during high-level synthesis. The second parameter is evaluated by a fast switching activity calculation algorithm. The model developed by [10] allows estimating the power consumption of distributed memory (using logic elements) in past FPGA families with technical parameters. Another model proposed in [11] uses technical parameters such as effective capacitance of each resource which is hardly obtained. All these methodologies and models use low-level parameters and technical characteristics which are not available before place and route. More precise approaches to estimate FPGA power consumption were described in [12-14].

B. XPower Xilinx Tool

XPower is a commercial-off-the-shelf tool to estimate the power consumption of Xilinx SRAM-based FPGAs. The design flow of the XPower is shown in Fig. 16. In this part of the work, the implementation of the Xilinx XPower will be investigated. XPower reads in either pre-routed or post-routed design data, and then makes a power model either for a unit or for the overall design based on the power equation: $P=CVf$ where P is average power consumption, C is equivalent switching capacitance, V is supply voltage and f is operating clock frequency or toggle rate. It considers resource usage, toggle rates, input/output power, and many other factors in estimation. Because XPower is an estimation tool, results may not precisely match actual power consumption. The frequency, f, is determined by users or provided by simulation data from the ModelSim family of HDL simulators.

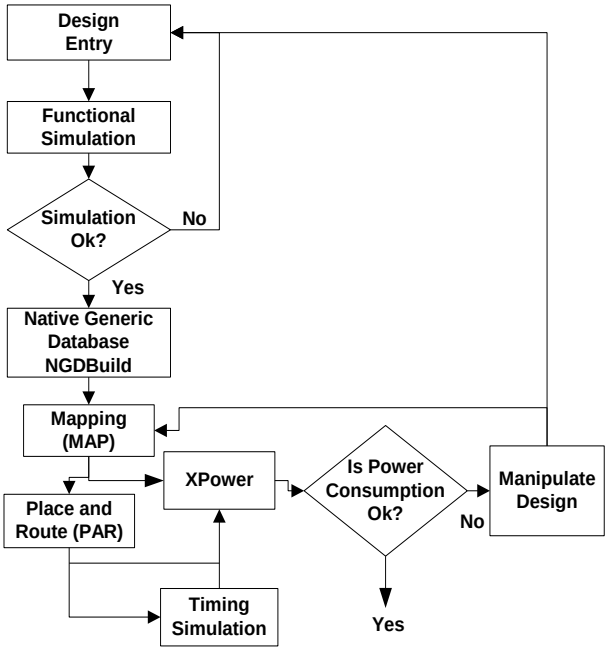

Figure 16. XPower tool design flow

XPower provides two types of information called data view and reports view. The data view shows the power consumption of individual parts of a design such as signals, clocks, logic, and outputs. The report viewer represents the total power consumed by a given design, which is again classified into the power consumption of clocks, logic and outputs, and static (leakage) power. The power consumption of clocks, logic, and outputs are calculated by equivalent switching capacitance models. The static power is based on the constant value quoted in a data book or calculated by an equation associated with temperature, device utilization, and supply voltage.

VII. MODELSIM CODE COVERAGE

ModelSim code coverage can display a graph and report file feedback on which statements, branches, conditions, and expressions in the source code have been executed. It also measures bits of logic that have been toggled during the execution. Therefore, it can be considered as a trace tool and probe at the same time that provides a history of the software execution.

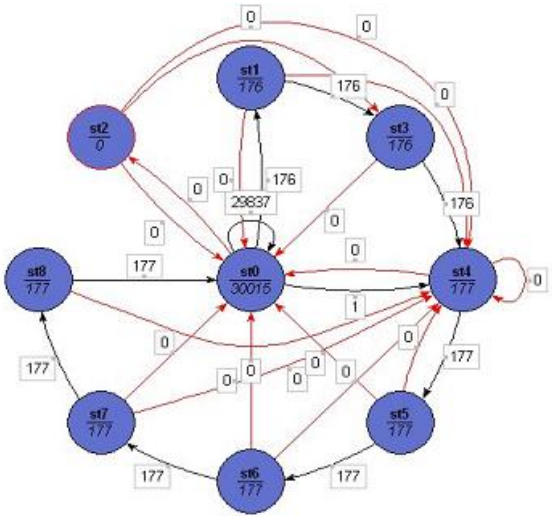

Figure 17. Transmitter FPGA-Compressor unit FSM

As code execution is almost invisible without an accurate trace tool, it is common for the entire blocks or modules of code to go unexecuted during test routines or redundant user-case suites. Coverage metrics showing which functions are not executed are useful for writing new, additional tests or identifying unused "dead" code. In applications where code size is critical, removing dead code reduces both wastes as well as risk in the targeted design. In many cases, code coverage can also be used to analyze errant behavior and the unexpected execution of specific branches or paths. The FSM can be extracted with code coverage as in Fig. 17 for the compressor unit.

The FSM figure clarifies the number of states involved in the design and the interaction between these states. A test bench has been written to examine the behavior of the HDL design. The code coverage is enabled to check the covered and uncovered parts of the design code by the test function which can lead to alter the design and consequently change the power consumption.

Fig. 18 shows some uncovered statements indicated by red X mark. In addition, an example of missed branches is shown in Fig. 19, where the X_T mark indicates that the true branch of the conditional statement was not covered.

```
BlockRamControl.v
  192                     equal    <= 1'b1    ;
  193                     nonequal <= 1'b0    ;
  228                        n_state = `incc             ;
  234   n_state = `tpinc    ;
  245         default   :       n_state = `init     ;
  266                   countc1  <=  8'h00           ;
  267                     bwritec1 <=  bwritec1 + 7'h01 ;
  270                     countc1  <= countc1 + 8'h01   ;
  271                     bwritec1 <=  bwritec1         ;
```

Figure 18. Not covered design statements is escaped

```
BlockRamControl.v
  XT        191             if (bdataw1 == bdataw)
+ X 2/10    225         case (c_state)
  XT        227                     if (equal)
+ X 1/4     263        case (c_state)
  X         265            if (countc1 == `incrc)
```

Figure 19. Not covered design branches

VIII. ACCURATE FPGA POWER ESTIMATION

The power characterization using XPower tool is done and based on the mapped/placed/routed design. In general, the total power consumption of a CMOS component is given by Equation.1. The dynamic power is due to the component activity while static power represents the power consumed by the leakage current.

$$P_{total}=P_{dynamic}+P_{static} \quad (1)$$

The static power given by the XPower is constant and calculated by the multiplication of maximum leakage current absorbed by the FPGA core and its supply voltage. On the other hand, dynamic power is varying according to the switching activity of the design. Therefore, two factors determine the accuracy of the XPower analyzer estimation: the accuracy of the data within Xpower analyzer and the stimulus provided by the user. It is necessary to mention that XPower relies upon stimulus data to estimate the power consumption for internal components. Valid input frequencies and toggle rates are necessary parameters to generate a proper power estimate. The main four important files that need to be invoked by the tool are the design (*.ncd), simulation (*.vcd), physical constraints (*.pcf), and setting (*.xml) files.

There are few strategies that can be implemented to reduce the power consumption of the FPGA device; these are:

1. Turn off clocks when they are not in use.
2. Make Block RAMs operate in "no read on write" mode. This reduces toggling of the output of the BRAM.
3. Use a clock enables the reduction of switching activity on the output of Flip Flops (FFs).
4. Partition Logic is driven by global clocks into clock regions and reduces the number to which each global clock is routed.
5. Reduce the total number of columns to which a clock is routed.
6. Reduce the total length of heavily loaded signals.

Mainly, we followed the recommendations in 3 &4 to reduce the power consumption of our design. Therefore, the global design has been partitioned into a lot of small blocks.

In order to investigate the impact of test bench writing style on the accuracy of the power estimate, two methods have been exercised. In the first one, the ADC_Din (line

carrying input data coming from the ADC has been stimulated by variable 8-bits data samples, as expected in the practical case. In the second method, only 0 data value is stimulating the ADC_Din. As an example, the control logic of the compressor BRAM has been considered to see the difference in the code coverage represented by the summary reports given in Figs. 20 and 21.

File: BlockRamControl.v

Enabled Coverage	Active	Hits	% Covered
Stmts	86	80	93.0
Branches	52	47	90.4
Conditions	6	6	100.0
States	9	8	88.9
Transitions	26	10	38.5

Figure 20. BRAM control coverage report without adc_in

File: BlockRamControl.v

Enabled Coverage	Active	Hits	% Covered
Stmts	86	83	96.5
Branches	52	50	96.2
Conditions	6	6	100.0
States	9	9	100.0
Transitions	26	12	46.2

Figure 21. BRAM control coverage report with adc_in

The hits count shows the number of times the indicated code part has been reached or executed. It is obvious that this count has been increased for all the design parts except the conditions. The states of the design have been fully covered in Fig 21 because the system is dealing with all the possible design options that are required in the verification stage. Sample ModelSim waveforms for the two adc_in conditions are shown in Figs. 22 and 23.

To investigate the effect of adc_in on the power consumption of the FPGA device, XPower tool has been used for this purpose. Table IV summarizes the total power and current estimates for both configurations.

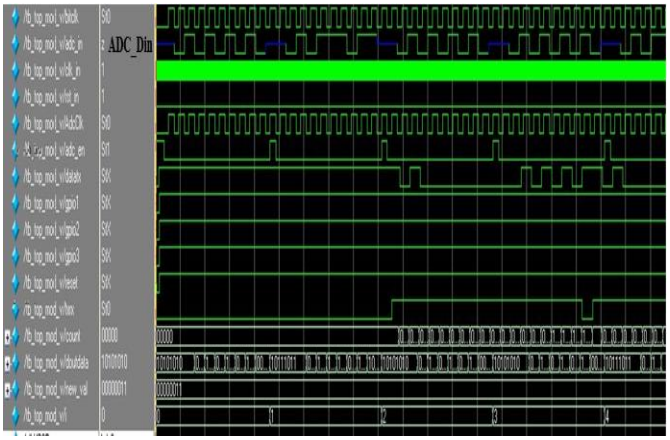

Figure 22. The simulation output waveform of the design with for variable adc_in

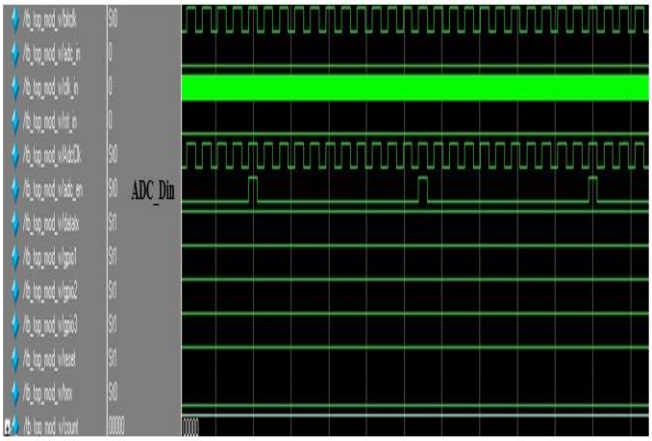

Figure 23. The simulation output waveform of the design with 0 data on adc_in

TABLE IV: TOTAL CURRENT AND POWER ESTIMATES

Total Power and Current estimates	I(mA) With adc_in	P(mW) With adc_in	I(mA) Without adc_in	P(mW) Without adc_in
Device		59		51
Vccint 1.20V	12	16	11	13
Vccaux 2.50V	16	41	15	38
Vcco25 2.50V	1	2	0	0
Quiescent Vccint 1.20V	10	12	10	12
Quiescent Vccaux 2.50V	15	38	15	38

As given above, more power is needed for the design when ADC_Din is clocking with different serial data. This leads to the conclusion that the FPGA device will consume higher power if the analog input signal to the ADC is rapidly changing. IN this case, the compressor unit of the design will be fully functioning with all the possible transition states.

The second case is assuming that the analog input has a steady value which is rare in practice, but useful to have power estimates for different working conditions. Thus total dynamic power is more dependent on the states of the input signals. In comparison, the power values in the two cases are different due to the code coverage analysis that has been discussed earlier. As more code has been covered with ADC_Din is varying, then we can consider that the obtained power estimate with such case has more credibility.

Quiescent power is the same for both configurations since it depends on the device itself using default conditions in moderate environments. To reduce the power consumption of the FPGA without losing accuracy, more work needs to be done outside than inside the device. For example, reduce the ADC resolution or the analog input can be good options. The test bench should be written in a very optimum way to provide a stimulus for all the inputs and read efficiently all the outputs. This is an important issue in the XPower tool since it provides one of the main files for the tool to estimate the power. The code coverage sometimes can help to extract the unnecessary parts from the

design which has a great benefit for the power consumption. As a final comment, obtaining the high code coverage can lead to a higher but more accurate power estimate using XPower.

IX. CONCLUSIONS

In this paper, PlanAhead tool has been employed as a hierarchical software environment after synthesis to analyze, modify, constrain and implement our design. It has been shown that a significant reduction in both the number and the length of design iterations can be obtained when using this tool.

A wireless system, which has been designed for sensor monitoring, required some optimization. Mainly, PlanAhead has been used to optimize the area occupied by the entire design, giving a better insight into the place and route process. Two strategies have been adopted in this work based on the number of Pblock placed to contain the different modules of the design. From the presented comparative results, using a single Pblock floorplanning design was representing the best scenario in terms of area compressing.

Also part of the work, the power consumption of the FPGA-based system has been investigated after passing through optimization placing with PlanAhead. The use of XPower from XILINX was the main focus of this work as an efficient tool to get good power estimates for the target FPGA device. The relation between the test bench coverage and power estimate accuracy was studied under different design conditions. It has been found that a good test bench with higher design code coverage capability can achieve more accurate power estimates. The presented results reflected clearly the efficiency of this method which can be applied with similar performance on any other Xilinx FPGA device.

REFERENCES

[1] E. Hassan and B. Al Momani, “New Approach to efficiently Assess the Power Consumption of Field Programmable Gate Array Devices”, The Sixteenth International Conference on Systems ICONS, Portugal, April 2021

[2] N. S. Sung et al., “Leakage Current: Moore’s Law Meets Static Power”, IEEE Computer Magazine, December 2003, pp 68 – 75.

[3] J. M. Rabaey,M. Pedram, Low Power Design Methodologies, Kluwer Academic Publisher, 1996, ISBN 0-7923-9630-8.

[4] P. Dillinger, J. F.Vogelbruch, J. Leinen, S. Suslov, R. Patzak, H. Winkler, and K .Schwan, “FPGA based real-time image segmentation for medical systems and data processing”, IEEE 14th NPSS Real Time Conf Proc., June 4-10 Sweden, 2005.

[5] K. Li, L. Lei, Q. Guang, J. Shi, and Y. Hao, “Improving the performance of an SOC design for network processing based on FPGA with PlanAhead”, IEEE International Conference on Electronics, Communications and Control (ICECC), Sept 9-11, 2011

[6] K. W. Poon, Power Estimation For Field Programmable Gate Arrays, Ph.D Thesis, department of Electrical and Computer Engineering, University of British Colombia, Vancouver BC, Canada 1999.

[7] X. Guo-Lin, “Power-sensitive design techniques on FPGA devices,” Electronics Quality Journal, 2002.

[8] L. Shang, A. S. Kaviani, K. Bathala, “Clock Power Analysis of Low Power Clock Gated Arithmetic Logic Unit on Different FPGA”, in IEEE International Conference on Computational Intelligence and Communication Networks, November 14-16, India, 2014.

[9] D. Chen, J. Cong, and Y. Fan, “Low-Power High-Level Synthesis for FPGA Architecture”, ISPLED’03, August 25-27 Seoul Korea, 2003.

[10] A. D. Garcia, Estimation et optimisation de la consommation de puissance des circuits logiques programmables du type FPGA [Estimation and Optimization of the Power Consumptionlogic circuits of the FPGA type], Ph.D Thesis, Ecole nationale suprieure des t´el´ecommunications de Paris, 2000.

[11] D. Elleouet, N. Julien, D. Houzet, J. Cousin, and M. Martin, “Power consumption characterization and modeling of embedded memories in XILINX”, IEEE Digital System Design Conf Proc., 2004, pp. 394-401.

[12] H. Gyu Lee, S. Nam, and N. Chang, “Cycle-accurate Energy Measurement and High-Level Energy Characterization of FPGAs”, IEEE Quality Electronic Design Symp Proc. 2003, pp.267-272.

[13] T. Fryza and M. Waldecker, "Precise Measurement of Power Consumption and Execution Time for Embedded Platforms," 25th International Conference on Systems, Signals and Image Processing (IWSSIP), Maribor, Slovenia, 2018, pp. 1-4.

[14] N. Lawal, F. Lateef, and M. Usman, "Power consumption measurement & configuration time of FPGA," Power Generation System and Renewable Energy Technologies (PGSRET), Islamabad, Pakistan, 2015, pp. 1-5.