Real-Time Subtraction Procedure for Power-Line Interference Removing from ECG: High Level Synthesis with Compaan

Tsvetan Shoshkov and Georgy Mihov

Abstract - **Power-line interference (PLI) is often present in the Electrocardiogram (ECG). The subtraction procedure has proved its efficiency in removing the PLI from ECG signals.**

Development process of such device is difficult and time consuming. High level synthesis (HLS) automates and speeds up the development process. In this paper HLS tools are used to develop a device for PLI removing. FPGA is used as a target platform. Automatic output generation is used to make the reconfiguration and further development easier and faster.

Keywords – **ECG, PLI, FPGA, HLS, Compaan**

I. Introduction

The Electrocardiogram (ECG) is often contaminated by power-line interference (PLI). The subtraction procedure shows good results in removing PLI from ECG signals [2]. The subtraction procedure algorithm is modeled in Matlab. Based on these algorithms is developed a hardware design for PLI removing form ECG [1]. This design realizes the subtraction procedure for real time operation. It is manually developed using VHDL and FPGA as a target platform. This standard manual development process using VHDL is error prone and time consuming. The design often must be reconfigured which is also a difficult process.

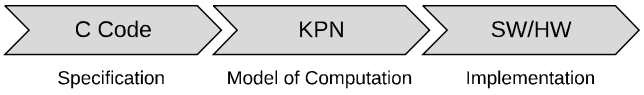

Fig. 1. Compaan Design Flow Basic Stages

In [5] are presented several system level development technologies. Automated high level system generation helps the development process and makes it easier and faster. In the present work we choose Compaan to develop a design for PLI removing from ECG using the Subtraction Procedure. Compaan is a high level synthesis tool [4]. Compaan is used in the current work because it can provide full system integration. It can implement not only the hardware and software implementation but also the communication and integration between them.

Compaan design flow basic stages are shown on fig. 1. Compaan works with simple C code input specification. Khan Process Network (KPN) model of computation is used [4]. KPN specification is automatically generated based on the input. After that target platform and mapping specification are defined. Then software and hardware implementation can be automatically generated. Quick changes in the input specification are enough to automatically generate a new design and reconfigure the target platform.

II. Subtraction Procedure for Power-Line Interference Removing

The process flow of the Subtraction Procedure is shown on fig. 2. Its structure contains three main stages:

– *Linear segments detection*. Each ECG sample is checked weather it belongs to a linear segment by using appropriate linearity criterion and a dynamic threshold M.

– *Interference calculation*. In linear segments the interference is calculated using a digital filter. It is stored in a temporal FIFO buffer and subtracted from the signal;

– *Interference extrapolation*. In nonlinear segments the value of the interference is calculated using the data stored in the temporal buffer and it is subtracted from the signal.

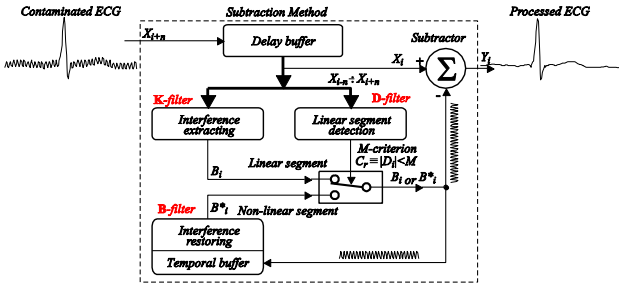

Fig. 2. Basic Structure of the Subtraction Procedure for PLI Removing.

More details about the subtraction procedure for PLI removing are given in [2], [3]. The present work is focused on the implementation of the subtraction procedure for PLI removing in real time using Compaan design flow. This paper evaluates the used design flow and the generated implementation.

III. Compaan Design Flow

Compaan design flow is centered on Compaan compiler and ESPAM tools [4]. The application must be specified as parameterized static affine nested loop which is a subset of the C language. Fig. 3 presents a detailed block diagram of the used Compaan design flow.

Compaan compiler automatically generates a KPN model of computation based on the input C specification. Processes are further processed by Compaan to obtain hierarchical sub networks. Target platform is selected to generate a platform specification. Processes of the KPN specification are mapped on HW (FPGA) and mapping specification is automatically generated. HW is

automatically generated by ESPAM based on the specifications generated in the first step. [3]

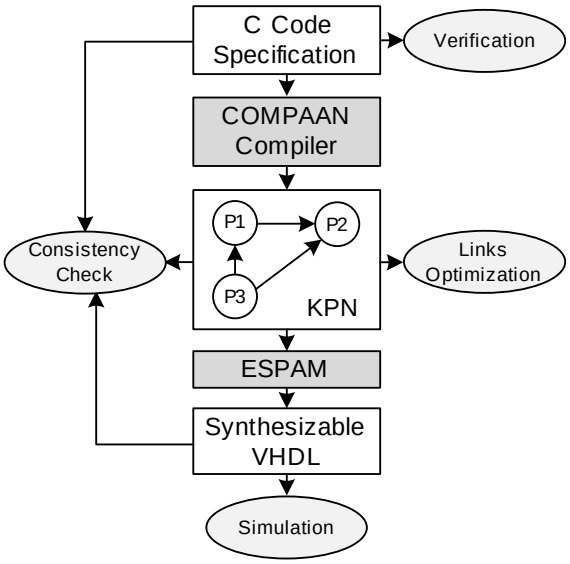

Fig. 3. Compaan Design Flow.

Nodes functions execution can be implemented by processors or HW accelerators. The HW nodes can be automatically generated with third party software or generated by Compaan. In our current implementation we use Compaan to generate the HW nodes. They are composed of three separate units - read, execute and write.

A deep pipeline can be integrated in the execution block. In this case the HW accelerators execution part must be manually filled. Though, Compaan generates a shell which includes input and output signals and the structure of pipeline stages.

The read block waits until there are tokens ready to be read. When there are processed tokens ready to be sent the write block writes them into the corresponding communication link if there is space available. This structure provides the nodes with the ability to work independently of each other. Inside each node there is a constant track of the data, using iteration counters. Compaan tool allows simulation and verification of the system at each stage of development. It can optimize the size of the communication links based on the current application. [3]

Compaan design flow is producing low power consuming and fault tolerant designs. Fault tolerance is based on the simplicity and higher abstraction of the input specification. Constant track of the data inside each node also increases the system fault tolerance. The asynchronous work inside the design makes it low power consuming since each node is working only when there is data to be processed. Memory is main source of power consumption and communication links size optimization is also a very important feature.

The reconfiguration of the system using Compaan design flow is easy. It requires modification of the input C code and following the Compaan design flow to generate bit stream that can be used to reconfigure the hardware.

IV. Implementing the Subtraction Procedure for PLI Removing with Compaan

The basic structure and elements of the Subtraction Procedure are shown on fig. 2. We use the Compaan design flow which is shown on fig 3 to implement the subtraction procedure for PLI removing. All development flow steps are presented in this section. First input specification is created, after that KPN is automatically generated. Finally synthesizable VHDL code is automatically generated. More details for each step are listed in the following subsections.

A. *Input Specification*

As a first step of the development it is created the input SW specification. Short fraction of the C code input specification is present to picture its main concepts:

```
  #pragma compaan_procedure ecg_pli
  void filter(int data_in[WIDTH], int data_out[WIDTH],
int pli_out[WIDTH], int cr_out[WIDTH]) {
    int ecg[WIDTH], cr[WIDTH], ecg_out_linear[WIDTH];
    int ecg_out_non_linear[WIDTH], ecg_pli[WIDTH];
    int ecg_filtered[WIDTH], kf[WIDTH], ku[WIDTH];
    int i, j, x;
    ecg_out_linear[0] = 0;
    ecg_pli[0] = 0;
    // Stream data into the design
    for (i = 1; i <= WIDTH; i = i + 1) {
        ecg[i] = data_in[i];
    }
    // Data processing
    for (j = 1; j <= WIDTH; j = j + 1) {
      // Linearity criterion calculation
      linearity_criterion(ecg[j], &cr[j]);
      // Frequency deviation tracking
      deviation_tracking(cr[j], ecg_pli[j-1],
ecg_out_linear[j-1], &kf[j], &ku[j]);
      // Interference extrapolation
      non_linear(ecg[j], kf[j], ku[j], ecg_pli[j-1],
&ecg_out_non_linear[j]);
      // Interference calculation in linear segments
      linear(ecg[j], kf[j], ecg_pli[j-1],
&ecg_out_linear[j]);
      // Switch between linear and non-linear segment
      cr_switch(cr[j], ecg_out_linear[j],
ecg_out_non_linear[j], &ecg_pli[j]);
      // Subtraction of the interference from the ECG
      subtract(ecg[j], ecg_pli[j], &ecg_filtered[j]);
    }
    // Stream data out
    for (x = 1; x <= WIDTH; x = x + 1) {
      // Filtered data
      data_out[x] = ecg_filtered[x];
      // Filtered Power-Line Interference
      pli_out[x] = ecg_pli[x];
      // Linearity criterion
      cr_out[x] = cr[x];
    }
  }
```

Functions are called inside static affine nested loops to process the data. Inputs and outputs of the functions are given as parameters. Output parameters are specified as addresses to variables where the result is stored. Feedbacks from the output PLI to the linear and non-linear segment calculations are realized.

First step in the specification is to stream in data into the design. After the actual processing starts with the first block which is linearity criterion calculation. Then deviation tracking is processed. Functions for linear and nonlinear segment filtrations are present. Another function switches between the nonlinear and linear output based on

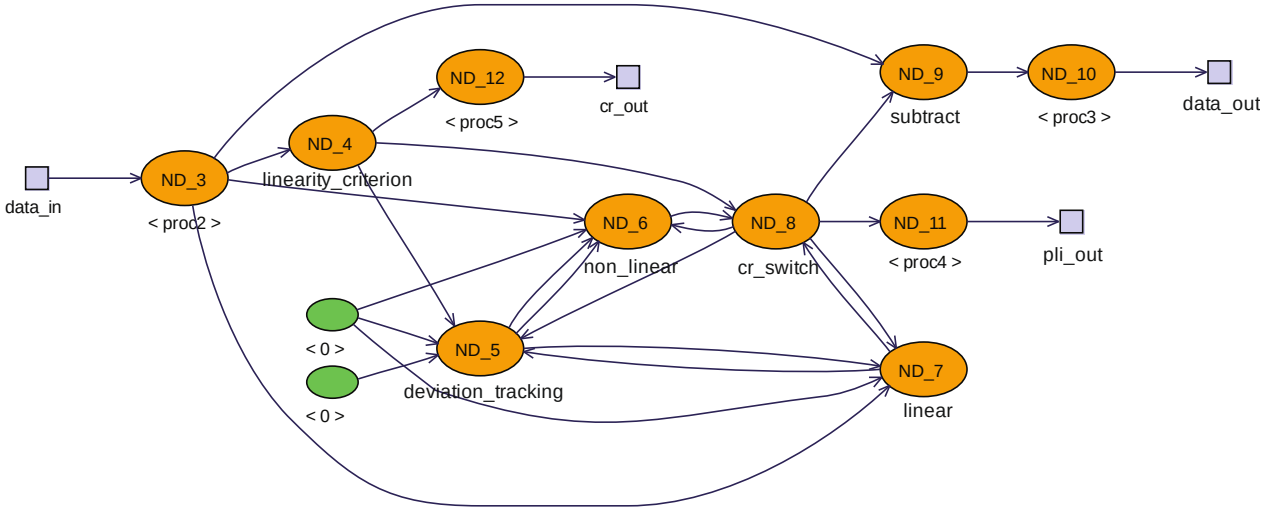

Fig. 4. Subtraction Procedure for PLI Removing KPN.

the linearity criterion. The PLI interference is subtracted by the original ECG signal. In the end the data is streamed out.

A. *KPN Model of Computation*

Compaan compiler is used to automatically generate KPN specification. This KPN specification can be used as a basis to generate the actual implementation of the device. It defines the dependencies between the nodes and connections between them.

The KPN is simulated to automatically optimize the communication link sizes. Fig. 4 shows the produced KPN for PLI removing from ECG using the Subtraction Procedure. Each function from the input specification is realized using a separate processing node. Additional processing nodes are used to stream data in and out from the design.

B. *Implementation*

Next step of the Compaan design flow is selecting target platform and mapping creation. In our case we use default target and mappings, since we want to generate a project to simulate. After we have the KPN, platform and mapping specifications ready, we can generate synthesizable VHDL code. Each node is mapped to a hardware processing core. It consists of read, execute and write section. Each execution section must be additionally filled with the actual processing equations. The manually filled code realizes the relation between the inputs and the outputs in each block. Here is a fraction of the code from the process that realizes the dynamic threshold calculation:

```
DYN_TR : process(CLK) begin
  if rising_edge(CLK) then
    if (RST='1') then
      mmax <= (others => '0');
      mmin <= (others => '0');
      mpp <= (others => '0');
      mmu <= (others => '0');
      md_wide <= (others => '0');
      sum15 <= (others => '0');
      sum16 <= (others => '0');
      sum17 <= (others => '0');
      sum18 <= (others => '0');
      sum19 <= (others => '0');
    else
      -- Pipeline Depth: 1 stages
      -- STAGE_0
      if( EN(0) = '1' ) then
        if mmax < XB(ni + 1) then
          mmax <= XB(ni + 1);
        else
          sum15 <= mmax - XB(ni + 1);
          sum16 <= abs(sum15 * d500);
          mmax <= mmax - sum16(D_2Width -
(D_2Width+1)/4 downto (D_2Width+1)/4);
        end if;
        if mmin > XB(ni + 1) then
          mmin <= XB(ni + 1);
        else
          sum17 <= mmin + XB(ni + 1);
          sum18 <= abs(sum17 * d500);
          mmin <= mmin + sum18(D_2Width -
(D_2Width+1)/4 downto (D_2Width+1)/4);
        end if;
        mpp <= mmax - mmin;
        if mmu > mpp then
          mmu <= mpp;
        else
          sum19 <= abs(mpp * d500);
          mmu <= mmu + sum19(D_2Width -
(D_2Width+1)/4 downto (D_2Width+1)/4);
        end if;
        md_wide <= mmu * d01;
      end if;
    end if;
  end if;
end process; -- DYN_TR
```

The dynamic threshold is being calculated for each new token received. It takes into account the difference between the minimum and maximum values of the ECG signal. Thus the result of the linearity criterion comparison with the given threshold is more precise.

At each step of the development flow we perform consistency check with the other stages to define if there

are any errors. Thus errors are easy to be identified and fixed at earlier stages of the development.

V. Evaluation and Results

Main target of our evaluation is to check if the generated design satisfies the application requirements. The development time is also a major aspect in this research. The system design must meet the functional and performance requirements and at the same time must be low power consuming and fault tolerant.

We use Xilinx ISE Design Suite 14.7 to simulate and verify the design. Virtex 6 FPGA is used as a target platform. ML605 evaluation kit with 200 MHz system clock is used. Compaan automatically generates a test bench that streams in and out data from the design. We use ECG signal provided by the American Health Association to test our design. Finally we use Matlab to plot the signals in graphics.

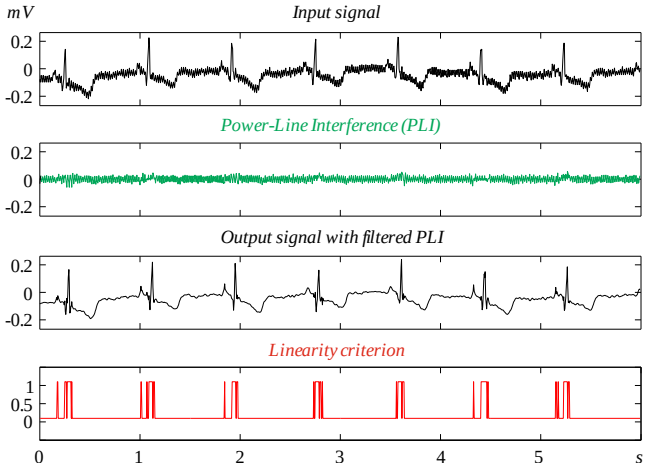

Fig. 5. PLI Removing Result Signals.

Fig. 5 shows two samples with input and output result signals obtained by the simulation of the design: The results obtained prove that the integrated design realizes the subtraction procedure for PLI removing according to the modeled algorithms.

The design implementation of Compaan suggests that the processes inside are working asynchronously. The HW IP cores are working independently of each other, so they can work only when there is input data to be processed. This way it is achieved low power consumption. Another main point sufficient for low power consumption is the link sizes optimization that is automatically performed.

A fully manual VHDL implementation of the subtraction procedure for PLI removing is presented in [1]. In the current work we use Compaan design flow to develop the same PLI removing processing algorithm. Table 1 presents a comparison between the development times consumed in the two cases. The comparison points that using Compaan the development time required is considerably decreased. The number of occupied slices in the FPGA chips is relatively the same.

Development Method	Target Platform	Slices Occupied	Man-Months
Manual	Spartan 3	5016 (44%)	1
Compaan	Virtex 6	4159 (2%)	0.5

Table 1. Development Methods Comparison.

Using third party software to fully automates the HW accelerators generations would additionally speed up the development. Automation of the generation and optimization benefit the development process. Using Compaan errors can be detected in earlier stages of the development.

VI. Conclusion

In this paper we present a high level synthesis of the subtraction procedure for PLI removing from ECG. We use Compaan design flow to automatically generate an output design based on simple input specification. This considerably decreases the development time required. Compaan further optimizes the design and makes it fault tolerant and low power consuming.

The automatic generation of the output makes the development faster and easier. Slight changes in the input specification can be made and a new design can be easily regenerated. This benefits the research and development process. Quicker further investigations in this field can be achieved.

VII. Acknowledgements

This paper is a part of a project under contract № 142ПД0040-03/2014, which is funded by the research program of the TU – Sofia, Bulgaria.

References

[1] S. Mihov, R. Ivanov, A. Popov, “Real-Time Subtraction Procedure for Eliminating Power-Line Interference from ECG”, ET 2008, Sozopol

[2] G. Mihov, I. Dotsinsky, T. Georgieva, “Subtraction Procedure for Powerline Interference Removing from ECG: Improvement for Non-Multiple Sampling”, Journal of Med. Engineering & Technology, 29, No 5, pp. 238-243, 2005.

[3] G. Mihov, I. Dotsinsky, C. Levkov, R. Ivanov, “Generalised Equations and Algorithm of the Subtraction Procedure for Removing Power-line Interference from ECG”, Proceedings of the Technical University of Sofia, Vol. 58, b. 2, Sofia, pp. 31-38, 2008.

[4] T. Stefanov, C. Zissulescu, A. Turjan, B. Kienhuis, Ed Deprettere. “System Design using Khan Process Networks: The Compaan/Laura Approach”, DATE, Paris, 2004

[5] A. Gerstlauer, C. Haubelt, A. Pimentel, T. Stefanov, D. Gajski, J. Teich, “Electronic System-Level Synthesis Methodologies”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 28, No. 10, pp. 1517-1530, 2009