Signal Processing and Waves Graphing Acceleration using FPGA

Holguer A. Becerra
UPB Professor
Electronics Engineer
Universidad Pontificia Bolivariana
Bucaramanga, Colombia
Email: holguer.becerra@upb.edu.co

Jose P. Pinilla
IEEE Member
Electronics Engineer
Universidad Pontificia Bolivariana
Bucaramanga, Colombia
Email: jose.pinilla@upb.edu.co

***Abstract*—FPGAs (Field Programmable Gate Arrays) and the description of custom HW (Hardware) are tools used for the enhancement of embedded applications, by abstracting exigent iterative tasks out of a processor and towards dedicated HW modules. This project contemplates the development of custom HWD (Hardware Description) in Verilog HDL (Hardware Description Language) for the acceleration of a signal wave graphing embedded system that performs the acquisition, processing and rasterization algorithms.**

A graphing module is described which provides the capability to intercept video interfaces, such as VGA or digital RGB, and overlay multiple wave graphs from raw data, with configurable characteristics. The development process of this project allowed the comparison of a wave graphing system based on a soft-processor, with accelerated designs using custom HWD for data acquisition and graphing. Results show that in addition to performance, the HWD modules offer lower power consumption and ease of implementation.

I. Introduction

Signal wave graphing is a common requirement in embedded instrumentation systems dedicated to the monitoring and logging of external variables. However, this functionality requires a lot more resources than the ones available in simple signal acquisition systems bound to strict power and economic budgets, as they may require the use of external GPUs that integrate a variety of graphing resources unnecessary for logging-and-graphing dedicated applications[1][2], such as Digital Oscilloscopes, Industrial Monitoring, Vital Signs Monitors and others.

A Verilog HDL module for the acceleration of signal wave graphing using FPGAs has been developed as part of an ongoing process carried by the ADT (Advanced Digital Technologies) academic group at UPB (Universidad Pontificia Bolivariana) Bucaramanga. This paper presents its justification, specifications, development methodology, comparison test results, performance and future work. ADT develops resources related to its fields of study and leaves them available and open for other students and the engineering industry to use in embedded systems solutions.

An advantage provided by the use of FPGAs is the possibility of performing signal wave graphing using the same integrated circuit used for signal processing and peripheral interfaces, without the high cost related to custom ASICs (Application Specific Integrated Circuits). FPGAs are a fast way of getting to market with highly integrated devices before or without moving to ASICs. For instance, Global Information and Communication Technology service provider Huawei was known for being the single largest buyer of FPGAs before changing their commercial solutions to ASICs[3], and recent news stated that Intel will be manufacturing Altera's high-end FPGAs[4], giving them an edge in technology that should boost the use of their devices in faster design-to-market solutions.

Although the algorithms for signal wave graphing can be carried out by a processor with the use of graphing libraries and frame dedicated modules, these systems result in lower performance for any other task, as video processing is a very demanding task by itself due to the speeds required to perform stable and optimal frame rates. However, the iterative nature of rasterization and other wave related algorithms makes them highly translatable to efficient HWD modules, where the inputs are points X and Y, in which X could be time and Y is the input data, while the output is the wave pixels setting to be plotted in the display. This project makes use of custom HWD for the acquisition and processing of incoming data from multiple signal waves, as well as rasterization algorithms and the graphing of pixels directly into the video frame.

II. Methodology

Video applications using FPGAs portray thorough examples of signal processing, clock synchronization and custom HW development. Therefore, UPB Bucaramanga's Electronics Engineering courses and the ADT group use video modules for the teaching and in the design of embedded systems, starting with the implementation of HW video synchronization using VGA on a Xilinx's Spartan 3A Starter Kit and focusing on graphing modules for the acceleration of this process as an addition to embedded soft processors and custom HWD.

A. Graph Module

A following step is to develop a set of modules that are capable of graphing any input data. For this purpose there is a peripheral device that delivers multiple-wave continuous data through a UART (Universal Asynchronous Receiver Transmitter) transmission that has to be decoded and arranged. The

978-1-4799-1121-9/13/$31.00 2013 IEEE

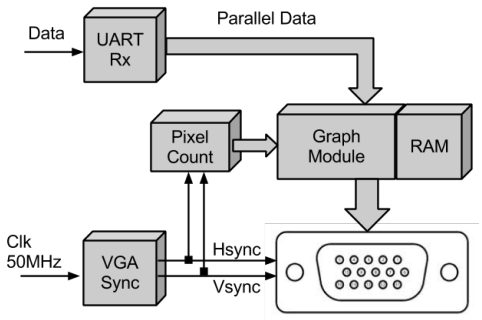

Fig. 1. Block Diagram of video synchronization and frame drawing module

graphing modules store a complete trace of the wave and create the graph to be displayed through VGA.

The first module required for this purpose takes the synchronization signals, that in the case of VGA are Horizontal and Vertical Sync, and counts them to obtain the location (X,Y) of the pixel being set in the frame. It resets in every frame and is configurable for any resolution. This count is pipelined into the main Graph Module, which uses the X axis count to address a RAM, where it stores every new data given by a UART receiver module. Figure 1 is a block diagram of the three mentioned modules (UART Receiver, Pixel Count and Graph Module), along with a VGA synchronization block that generates the corresponding H-Sync and V-Sync signals according to the resolution and frame rate to be displayed.

Every new data stored by the Graph Module is the raw amplitude of an acquired signal. These are cyclically written into the RAM which depth is equal to the width of the wave graphing area in the interface, i.e. 640 in a 640x480 display where the wave graph takes the whole screen. This size and location of the graph are configurable, e.g. A wave graph area of 360x80 can have its origin at pixel 280,400 to make it occupy the right-most bottom corner of a 640x480 display.

Every data in RAM matches a pixel in the graphing area, where its X location is the corresponding RAM address shifted by the X origin offset of the wave graph area. The Y axis operation takes into account that the pixel count starts at the left-most upper corner of the display and the wave should increase towards that direction, therefore the value in RAM has to be subtracted from the Y offset of the wave graph plus its height to obtain the Y axis pixel location. The X pixel count is constantly reading this RAM, and the Graph Module determines whether the current pixel should display or not a dot when pixel count Y matches the Y axis location resulting from the data read. If the RAM is initialized with zeros, a line will be drawn on the bottom of the graph, and as soon as data is received it will begin to be displayed.

The Graph Module generates the RGB colors configured by the user and a bit signal called Graph ON that is high when the pixel being set matches the data to be graphed. "Graph ON" is the selector of a multiplexing circuit that chooses between the VGA background color (e.g. Black) and the wave signal color, which results in a cyclical drawing of dots along the axes of the graph like the ones pictured in image (a) of Figure 2, which does not represent a continuous graph. A rasterization algorithm is required in order to obtain continuous lines by

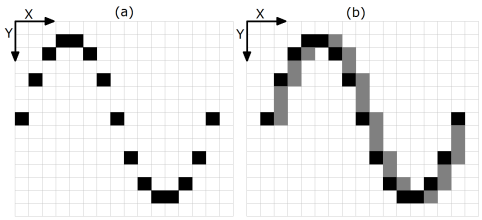

Fig. 2. Data Rasterization. (a) Data as pixels (b) Data as lines

connecting the dots. This process is also carried out by a logical HWD inside the Graph Module.

When a new amplitude data is read from RAM, the previous data is saved in another RAM (RAM-1) and is read to determine if a line of pixels should be drawn above or under the new data pixel in order to connect both dots. It is a simple rasterization algorithm that allows a continuous line to be drawn from a vector of data, these pixels are going to have a Graph ON signal when the following condition is true:

```
wire SlopeFX=

((CountY+W) >= ((RAM_1[7:0]/H) + Pos_Y))?
((CountY <= ((RAM[7:0]/H) + Pos_Y))&
((CountY+W) >= ((RAM_1[7:0]/H) + Pos_Y))):

((CountY) <= ((RAM_1[7:0]/H) + Pos_Y))?
(((CountY+W) >= ((RAM[7:0]/H) + Pos_Y))&
(CountY <= ((RAM_1[7:0]/H) + Pos_Y))):

1'b0;
//Where: W=Width and H=Height
```

Figure 2 (b) is a demonstration of how lines are drawn using this module, gray pixels are drawn by the rasterization conditions while black ones are read directly from RAM.

Additionally, there are configuration inputs that allow the user to change any graph parameter, like color, size, position, visibility, speed and mode (scroll or redraw). The configuration for a scroll display mode is done by incrementing the address register for RAM, giving the appearance that data is being shifted through RAM. When RAM is overflowed the address offset register resets and continues the process. The full I/O port list of the graph module is as follows:

```
module plot_graph(
 //Inputs
 CountX, //X-position of current pixel
 CountY, //Y-position of current pixel
 data_graph, //Input data to store in RAM
 data_graph_rdy, //Input data ready
 display_clk, //Graph module clock
 color_graph, //RGB format graph color
 scroll_en, //Mode: Scroll or Redraw
 Pos_X, //X-axis position for graph origin
 Pos_Y, //Y-axis position for graph origin
 Witdh, //Ammount of data to be graphed
 Height, //Maximum wave amplitude
 line_width, //Pixel width of the wave
 Graph_en, //View graph or make invisible
 reset_n, //Active-low reset
```

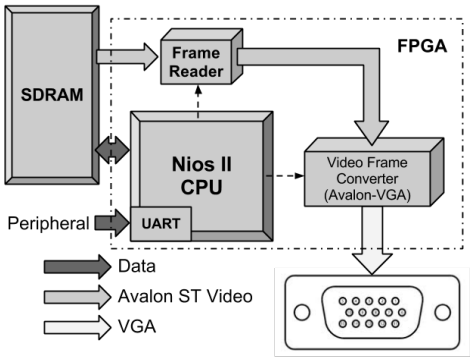

Fig. 3. Block Diagram of the First Test

```
 pause_graph, //Stop graphing new data
 //Outputs
 Graph_on, //Graph is in current pixel
 R, //Red component
 G, //Green component
 B//Blue component
);
```

B. First Test

The first testing stage of these modules involves a comparison with a wave graphing system based solely on software, for which the Nios II soft-processor from Altera[5] was chosen, along with the DE0-Nano Development board with external RAM(Random Access Memory) and Altera's VIP Cores[6]. As shown in Figure 3, the processor takes the UART input, graphing libraries and control of the video cores to perform the process. Wave data is written to RAM with the video formatting functions supported by the VIP Frame Reader Core according to design examples and conforming a custom SoPC (System on-a Programmable Chip)[7]. However, when processing multiple signals, this system presents stuttering and lost values caused by the sequential processing, rasterization and graphing of data with insufficient clock speeds required to display an optimal frame rate. This design is used as a base to create a second test by merging it with the previous HWD design.

C. Second Test

In order to broaden the applications of the first HWD modules, an additional processing system can create a video source to work as a background and for user interaction, on top of which the waves will be graphed. Figure 4 is an example of the video output used for the system tests, in which the graphs are implemented with SW(1st Test) or HWD(2nd and 3rd Tests), while the picture and the labels are always drawn using SW. This background, labels and pictures are set by a system similar to the one used in the first test: a Nios II processor, VIP Cores and a user input (Touch, Keyboard or Mouse). The system uses a Frame Reader module to acquire sprites and drawings from RAM to use as buttons, text boxes, lines and pictures; these form a GUI (Graphic User Interface) with a dedicated area for wave graphing.

Frames are sent through a video converter or CVO (Clocked Video Output) that turns them into VGA or Digital

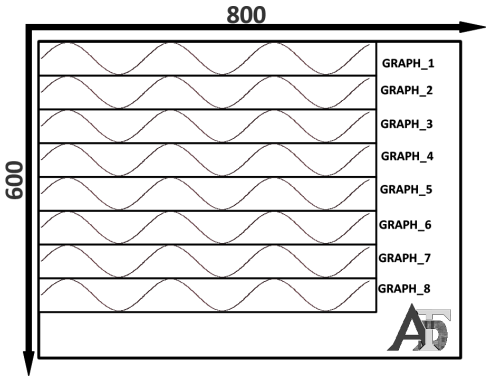

Fig. 4. Video Output Example

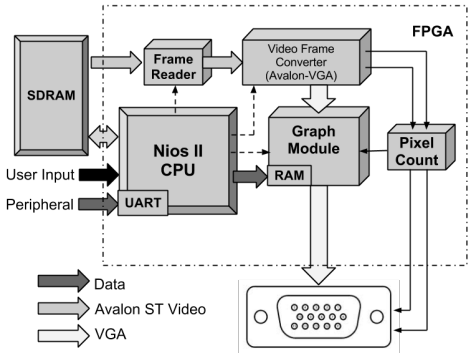

Fig. 5. Block Diagram of the Second Test

RGB format so they can be run through the previously mentioned multiplexing circuit in the Graph Module. This results in a processing system dedicated to the GUI that is going to be updated with every interaction of the user, at very low frequencies, while a HWD system will be dedicated to signal graphing which requires higher frequencies. Data acquired through a UART module attached to the Nios II processor is decoded and forwarded to the Graph Module where it rasterizes the data into waves to be overlayed on the background. The resulting system is pictured by the block diagram in Figure 5 where control signals for parameters like color and location are connected from the Nios II processor to the graphing module. This approach shows a performance upgrade from the previous system but the task for data acquisition and decoding is still demanding processor resources that can be outsourced.

D. Third Test

For the final design, the UART data decoding and arranging is taken out of the processor and into a custom HWD module, distributing the design and releasing more processing tasks out of the Nios II, as shown in Figure 6. Similar to the first HWD, this design shows a wave graph on the display if no other video has been set by the Nios II and VIP modules. It keeps the configuration ports attached to the Nios II allowing the customization of the graphs through interaction of the user, whether it is by using a touch display or through other input devices attached to the Nios II. The processor uses the VIP modules to lay a background and GUI (Graphical User Interface), resulting in a complete system for the visualization of signal graphs with the possibility for interaction and custom

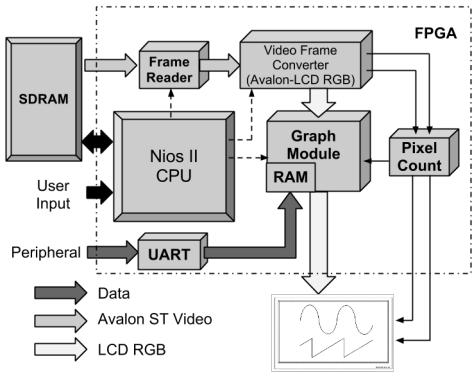

Fig. 6. Block Diagram of the Third Test

TABLE I. CHARACTERISTICS OF EACH GRAPH MODULE

Parameter	Value
Combinational	99 LC
Registers	27 LC
Memory	3840 bits
Power	0.85 mW
Module Frequency	40 MHz
Screen Resolution	800x600 pixels
Wave Resolution	640x64

interface designs.

Additional development involved the use of LCDs instead of VGA monitors, which requires a different Video Frame Converter and connection of the video signals to the Pixel Count module, without regarding the other components of the design as RGB data is kept unchanged. This serves as an example of the flexibility achieved with this development, by performing equally on both Xilinx Spartan 3A and Altera's Cyclone IV, as well as with different display technologies through simple modifications due to a pure Verilog description of the modules without depending on device primitives or additional software.

III. RESULTS

Each of the three testing systems was run with different parameters, varying frequencies and all versions of the Nios II processor (Fast, Standard and Economic) that can achieve 170, 96 and 23 DMIPS (Dhrystone Million Instructions Per Second) correspondingly at 150 MHz. The results acquired from these tests can be divided in three main characteristics: Each Graph Module's implementation, Frames Per Second (FPS) and power dissipation.

Table 1 is a detailed description of how each Graph Module is implemented in the FPGA for every test. Although the resolution values and frequency are variable, the remaining values are exactly the same for every instance that is added to the design. Therefore, a system with 8 Graph modules will occupy 8 times 99 Logic Cells (LC) of combinational logic, 8 times 27 LC of register logic and 8 times 3840 bits of memory.

FPS are the most important unit of measure in these tests and the results shown in Figure 7 are the justification for this project. This graph shows the number of FPS that every test system can update using different parameters and with an

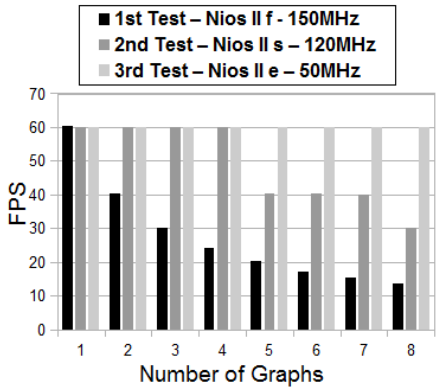

Fig. 7. FPS Comparison

increasing number of waves to be graphed. The first test system is run with the highest DMIPS processor, the second test is done with a Standard Nios II while the third test is run with the lowest performance processor. This however shows that while the first test is unable to keep a FPS rate of 60 Hz with more than one wave being graphed, the second test system is capable of graphing four, despite of having a lower performance Nios II at a lower frequency. Furthermore, the third system is the only one capable of maintaining a stable 60 Hz FPS rate at the video output, while having the lowest performance Nios II, at 8 DMIPS (Nios II economic at 50 Hz). The same results can be achieved by the third test system using any other operational frequency for the Nios II processor, because the data acquisition and graphing are done independently.

The use of HWD modules for demanding data processing tasks, called Accelerators, can result in a reduction of the dynamic power consumption of a system. Their objective is to release tasks from a processor and perform them more efficiently by reducing the number of required clock cycles through custom design and parallelization. Converting software to HWD is a task that can be automated by tools such as C2H (C to Hardware) translator, but not every task can be easily parallelized or successfully converted to HWD. Video processing however, is a good example of an application that can be accelerated with significant results[8].

An approximation of the power dissipation of every HWD module can be derived from the device and HWD characteristics, such as frequency, LCs and operating temperature (25 °C room temperature by default), the Quartus II software from Altera gives an approximate Total Thermal Power that is a sum of the Dynamic, Static and I/O power estimated in the design[9]. Total Thermal Power is used in this project to compare each of the evaluated systems in Figure 8, by taking the minimum value at which each system is capable of updating one wave at 60 Hz.

The decrease in power that can be seen between the first and second tests relates only to the Graph Module as it was shown in the corresponding (first and second tests) block diagrams. However, when the third system is tested at 120 MHz the power reduction relates to the UART receiver and decoder that is now a HWD, allowing the system to update a 60 HZ FPS video output using a lower performance Nios II. Further reduction of power in the latter examples can be

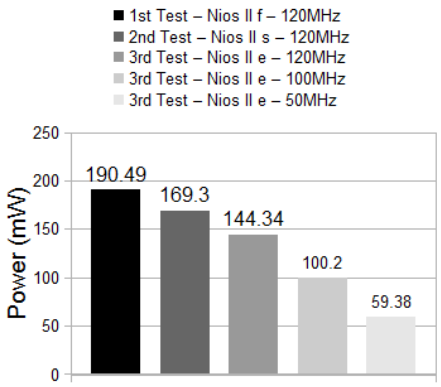

Fig. 8. Minimum power at 60 FPS

expected due to the relation between frequency and power in digital circuits, the Nios II processor frequency can be reduced, whereas the Graph module's frequency remains at 40 MHz.

IV. Conclusion

The final design, used as the third testing system, shows a liable behavior under demanding conditions like multiple graphs and complex data frames, making it appropriate for FPGA-based embedded systems or SoC (System on-a Chip) implementations where a high level of reliability is required in the continuity and stability of real-time multiple wave graphing (i.e Digital Oscilloscopes, Spectrum Analyzers, Vital Signs Monitors or Audio players). The system offers ease of scalability and real-time processing while keeping a lower memory usage due to the use of raw data vectors and the re-processing to obtain the wave graph on every frame read, instead of storing the processed matrices or graphical objects.

In regard to power consumption the module showed a noticeable upgrade from a typical application using the Nios II to write the video frames in memory, specially when the system is configured with more than one wave graph, and noting that only the final design was capable of displaying eight wave graphs while still keeping the frame rate. Furthermore, there is always a decrease in power related to the implementation of UART data reception and decoding with a custom HWD module instead of the Avalon and software based interface used in the previous implementations (e.g 0.64 mW for the UART receiver and decoder).

Future work on this system aims towards the standardization of the modules by using a more complex rasterization algorithm (e.g. Bresenham's Algorithm[10]), the implementation of lines and markers, and Avalon or Wishbone bus standards to improve its applicability in commercial FPGA systems. As a result the Graph Module should be able to perform real-time processing of the data points with configurable parameters and extend its application to dynamic X axis data.

References

[1] Kahraman Serdar Ay and Atakan Doan, *Hardware/Software Co-Design of a 2D Graphics System on FPGA*, International Journal of Embedded Systems and Applications (IJESA) Vol.3, No.1, March 2013, Available: http://airccse.org/journal/ijesa/papers/3113ijesa02.pdf

[2] Brahim Betkaoui, David B. Thomas, Wayne Luk and Natasa Przulj, *A Framework for FPGA Acceleration of Large Graph Problems: Graphlet Counting Case Study*, 2011 International Conference on Field-Programmable Technology. December 2011, Available: http://www.doc.ic.ac.uk/ natasha/fpt11bb.pdf

[3] Dylan McGrath, *Huawei using ASICs for First Time*, EETimes, 14 Oct 2012, Available: http://www.eetimes.com/electronics-blogs/semi-conscious/4399283/Huawei-using-ASICs-for-first-time

[4] Eric Savitz, *Intel Agrees To Serve As Foundry For Altera FPGAs*, Forbes, 25 Feb 2013, Available: http://www.forbes.com/sites/ericsavitz/2013/02/25/intel-agrees-to-serve-as-foundry-for-altera-fpgas/

[5] Altera Corporation, *Nios II Processor: The World's Most Versatile Embedded Processor*, 2011, Available: http://www.altera.com/devices/processor/nios2/ni2-index.html

[6] Altera Corporation, *Video and Image Processing Suite MegaCore Functions*, 2010, Available: http://www.altera.com/products/ip/dsp/image_video_processing/m-alt-vipsuite.html

[7] Pong P. Chu, *Embedded SoPC Design with Nios II Processor and Verilog Examples*, Wiley, 2012

[8] Altera Corporation, *AN 531: Reducing Power with Hardware Accelerators*, 2008, Available: http://www.altera.com/literature/an/an531.pdf

[9] Altera Corporation, *PowerPlay Early Power Estimator: User Guide*, 2012, Available: http://www.altera.com/literature/ug/ug_epe.pdf

[10] Andrs Iglesias, *Computer-Aided Geometric Design and Computer Graphics: Line Drawing Algorithms* Department of Applied Mathematics and Computational Sciences, University of Cantabria, 2001, Available: http://ocw.unican.es/ensenanzas-tecnicas/visualizacion-e-interaccion-grafica/material-de-clase-2/03-LineAlgorithms.pdf

[11] Pong P. Chu, *FPGA Prototyping By Verilog Examples: Xilinx Spartan-3 Version*, Wiley, 2008

[12] Altera Corporation, *Adding Hardware Accelerators to Reduce Power in Embedded Systems*, 2009, Available: http://www.altera.com/literature/wp/wp-01112-hw-reduce-power.pdf