Embedded Hilbert Transform-Based Algorithm within a Field Programmable Gate Array to Classify Nonlinear SDOF Systems

Jonathan D. Jones  
Graduate Student  
School of Electrical & Computer Engineering  
University of Oklahoma  
Norman, OK 73019-1024

Jin-Song Pei  
Assistant Professor  
School of Civil Engineering & Environmental Science  
University of Oklahoma  
Norman, OK 73019-1024

Monte Tull  
Associate Professor  
School of Electrical & Computer Engineering  
University of Oklahoma  
Norman, OK 73019-1024

ABSTRACT

This paper presents some preliminary results of an ongoing project. A pattern classifier to identify types of nonlinearities for single-degree-of-freedom (SDOF) systems is being developed and embedded into a Field-Programmable Gate Array (FPGA) and microprocessor-based data processing core. Reported here in this paper is part of an embedded design developed for a full implementation within FPGA hardware. The goal of this study is to enable and optimize the functionality of onboard data processing of nonlinear, nonstationary data in smart wireless sensing for structural health monitoring applications. As a fundamental “building block” in processing nonlinear time histories and extracting instantaneous features of nonstationary dynamic data, an established algorithm based on the Hilbert Transform developed by Feldman [3, 5] is being implemented into this embedded hardware design with some modifications at this stage of the project. Compared with traditional microprocessor-based systems, fast growing FPGA technology offers a more powerful, efficient, and flexible hardware platform including on-site (field-programmable) reconfiguration capability of the hardware. A proof-of-concept work that this project is built upon is reviewed in this paper. An off-the-shelf high-level abstraction tool rather than hardware description language (HDL) is adopted to program the FPGA. Various design options are considered in this study to optimize computational efficiency and/or resources. Simulation results using the embedded hardware design are presented and compared with their counterparts using a general purpose computer in a Matlab environment. Future work includes making an FPGA and a microprocessor operate together to embed a further developed algorithm that yields better computational and power efficiency.
1 INTRODUCTION

This paper presents some preliminary results of an ongoing project, which aims at (1) the development of a pattern classifier that can directly operate on measured time histories and extract the underlying nonlinear characteristics for damage detection purpose, and (2) embedding this classifier to a Field-Programmable Gate Array (FPGA) and microprocessor-based data processing core at a wireless sensor node. The work presented in this paper is part of an embedded design to be used entirely within FPGA hardware. This design will be used for future integration with a microprocessor core.

Dynamic measurements have been used as an important resource to assess the structural integrity for structural health monitoring (see summaries in [17, 18, 10] and many other references). The ability to derive meaningful patterns and delineate other information from nonlinear and non-stationary data sets using truly nonlinear techniques rather than entirely relying on equivalent linearization is being sought in this study. This is because nonlinear behavior and rapidly changing processes are often associated with possible damage. For example, permanent deformations and cracks of a reinforced concrete structure during strong earthquake shaking lead to nonlinear hysteresis with memory effect and degradation. As a fundamental “building block” in processing nonlinear time histories and extracting instantaneous features of nonstationary dynamic data, an established algorithm based on the Hilbert Transform [9, 20] developed by Feldman [3, 4, 5, 7, 6] is being studied by the authors and their co-authors [16] and is the focus of this proposed hardware implementation.

The development of embedded systems for civil infrastructures, especially for wireless communication, has seen a surge in the past decade with a large of amount noteworthy work accomplished (see [14, 8] for summaries). This study is built upon a proof-of-concept study accomplished by this research team [12, 13]; it will be continued to explore the application of fast growing Field-Programmable Gate Array (FPGA) technologies in wireless monitoring systems. The goal is to optimize the functionality of onboard data processing capacities by selecting proper hardware platforms and proper combinations of them to implement a wide range of system identification and damage detection algorithms for local data interrogation.

In the proof-of-concept study [12, 13], several Fast Fourier Transform (FFT)-related digital signal processing (DSP) algorithms were implemented to an FPGA-based smart wireless sensing unit. In this study, the embedded algorithms will be drastically extended to handle nonlinear and nonstationary data sets onboard. To do so, the Hilbert Transform and backbone curve are the key components to start with.

For a monocomponent signal $y(t)$ (which can be displacement, velocity, acceleration and others), its Hilbert Transform $\tilde{y}(t)$ is defined as follows:

$$\tilde{y}(t) = \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{y(\tau)}{t - \tau} d\tau$$

(1)
Two other basic governing functions essential to this study are:

\[ A(t) = \sqrt{y^2(t) + \tilde{y}^2(t)} \]  
\[ \omega(t) = \frac{d}{dt} \left( \arctan \frac{\tilde{y}(t)}{y(t)} \right) \]

where \( A(t), \psi(t) \) and \( \omega(t) \) are the instantaneous envelope or amplitude, phase and angular frequency, respectively, of the analytic signal \( y(t) + j\tilde{y}(t) \). The plot \( A(t) vs. f(t) \) is called a system backbone (or skeleton) curve [3, 4, 5, 7, 6], where \( f(t) = \frac{\omega(t)}{2\pi} \). A closely related plot \( A(t) vs. \omega(t) \) is implemented into the hardware design in this study.

Continuing with the proof-of-concept work, a Matlab Simulink-based high-abstraction tool, System Generator [24], developed by Xilinx Inc [23], for programming FPGAs, is utilized to develop the design rather than using hardware description language (HDL) alone. System Generator extends Simulink to enable hardware design by providing high level abstractions that can be automatically compiled into an FPGA [24]. This introduces transparency and efficiency into a hardware implementation of a complex algorithm being developed within this cross-disciplinary team. System Generator does not fully replace HDL-based design, but does allow the user to focus attention only on the critical portions of the design, which is the case in this study.

## 2 DESIGN IMPLEMENTATION

While detailed implementation can be referred to [11], an overall design flow of this study is illustrated in Fig. 1. At this stage of the project, the design implementation is limited solely to an FPGA-based platform with focus given to Stages II, III and IV. Stage I Preprocessing will be carried out at a later stage of this project to produce monocomponent signals suitable for the Hilbert Transform. Within Stages II to IV, seven specific steps are identified to form a design flow chart for this hardware implementation. This design flow in Fig. 1 results from careful planning and selection of stages and steps that are suitable for an embedded system. Overall, Stages II and III produce basic results that form a backbone curve useful for pattern classification in Stage IV. The steps implemented in Stage IV primarily simplify and then extract the dominating features of the backbone curve, which enables an automation of the identification of different types of nonlinearities. Currently, three basic linear/nonlinear types (i.e., linear, and hardening and softening nonlinearities) are considered in the identification, while more types of nonlinearities will be implemented in the future based on this design.

The heart of the proposed design shown in Fig. 1 is the Hilbert Transform (HT). This transform can be implemented in many different ways (e.g., [20, 9]). One such algorithm that is suitable for an embedded design is: (1) perform the Fast Fourier Transform (FFT) on the signal, (2) make the components of negative frequencies zero, and (3) calculate the inverse Fast Fourier Transform (iFFT) to obtain the analytic signal, and (4) take the imaginary portion as the Hilbert Transform of the original signal [20]. Item (2) above is equivalent to multiplying the FFT output by a step function, which explains the shorthand notation of “Step” used later in Fig. 2. To implement this algorithm, two options are executed via hardware simulation by use of Xilinx System Generator [24]
and are illustrated in Fig. 2. The first option shown in Fig. 2(a) implements two System Generator FFT blocks in sequential order to produce the desired HT output (referred to as “two-FFT” option). The second design option shown in Fig. 2(b) implements one FFT block to perform both the forward and inverse FFT operations via a feedback loop configuration (referred to as “one-FFT” option). While hardware resources are conserved by the use of only one FFT block with this configuration, new issues arise from the critical control timing for both feedback and FFT/iFFT operations. Both design configurations produce identical Hilbert Transform results, and both possess tradeoffs that will be elaborated hereafter.

Figure 1: Overall design block diagram.

Figure 2: Two implementation schematics for Step 1 Hilbert Transform shown in Fig. 1 with the options of (a) two System Generator FFT blocks, and (b) one System Generator FFT block, respectively. The numbers in Panel (b) indicate the execution sequence.

All block operations within System Generator consume a specific amount of FPGA resources depending upon the complexity of the operation. To minimize the hardware resources required for the FFT operation, Radix-2 mode was adopted throughout this study. In addition to resource consumption, operations within System Generator could posses a latency contributing to the overall computation time of the process. Keeping this in mind, the two HT options (i.e., two-FFT and one-FFT options) are analyzed to examine both resource utilization and computation time which are shown in Fig. 3(a1,2) and (b), respectively. The resource utilization includes slices, flip-flops (FFs), look-up-tables (LUTs), block random access memory (BRAMs) and embedded multipliers (Mults). It can be seen that the resources consumed by the two-FFT option exceed those obtained
from the one-FFT option, despite the addition of necessary feedback and control components. This is due to the majority of FPGA resources being allocated to the FFT operation. Although hardware resources are conserved with the one-FFT option, it comes at the price of a slightly increased computational time of 13732 clock cycles, compared to 12605 for the two-FFT option. This increase in computation time could be considered negligible in many possible applications in health monitoring of civil infrastructures due to the high clock frequencies available in today’s FPGA technologies. For instance, assuming an FPGA clock frequency of 100 MHz, the difference in computation time is \((13732 - 12605) \times 10\text{ns} \approx 0.11\mu\text{s}\).

As seen in Fig. 1, after the Hilbert Transform the analytic signal is sent to **Step 2**, which introduces the CORDIC (COordinate Rotation DIgital Computer) algorithm [21] [19], used to perform the arctangent of the analytic signal, yielding the amplitude \(A(t)\) and instantaneous phase \(\psi(t)\) values to be utilized further in the design. CORDIC is a hardware-efficient algorithm used for iterative solutions of trigonometric and other transcendental functions via shifts and additions only. Usually the CORDIC rotator is carried out in one of two modes - rotation and vectoring [2]. Xilinx provides various CORDIC algorithms in the System Generator blockset [24]. The precision of the CORDIC block’s output is dependent upon the number of processing elements implemented in the algorithm. An increase in processing elements in turn increases the number of iterative stages used in fine angle rotation, resulting in an increased precision at the output, generally one additional bit of precision per iteration [2]. As processing elements are increased, however, more FPGA resources are consumed, introducing a need to optimize the design for both precision and device utilization. This is another challenge to overcome in this study.

To better understand the CORDIC algorithm, an exercise was carried out using sine and cosine functions. The performance of the System Generator SINCOS CORDIC block was compared with Matlab fixed-point sine/cosine signals. Fig. 4(a) presents the mean squared errors (MSE) for a certain range of System Generator CORDIC block configurations, each utilizing an increased number of processing elements. The Matlab values of sine and cosine are considered the baseline in this comparison. Presented in Fig. 4(b) are the respective resource utilization values according to each CORDIC configuration. This exercise indicates that even a small number of processing elements (e.g., 4 to 10) would result in a balanced design for both accuracy and hardware efficiency. More extensive investigations into this topic for CORDIC operations relevant to this study will be performed later.
Figure 4: Preliminary validation of System Generator SINCOS CORDIC operation via comparison to Matlab fixed-point results. Mean squared error (MSE) is seen in (a), while resource consumption figures are seen in (b). Note the discontinuity of the x axis due to the nonuniform increase in processing elements.

The CORDIC arctangent algorithm in Step 2 of the design yields the magnitude and phase of the analytic signal. The phase output representation is defaulted to a repeating wrapped phase. This behavior is undesirable since the discontinuity of the phase is not differentiable, preventing the subsequent first-order derivative option (Step 4). To overcome this problem, Step 3 was implemented to “unwrap” the instantaneous phase using System Generator. This is accomplished via a “Shift & Subtract” subsystem designed in System Generator. In this system, a first-order numerical differentiation technique is implemented into hardware by implementing single-port RAM to store the data along with its shifted counterpart. Simple subtraction is then carried out between the two RAM vectors to yield the desired derivative.

At the end of Stage III (i.e., Line “e” in Fig. 1), two signals, $A$ and $\omega$, are obtained necessitating a backbone curve (i.e., $A$ vs. $\omega$). It will be shown later in the simulation that this curve, however, is not be suitable for a direct submission via a wireless link (the ultimate goal of this study described in Section 1). The information needs to be condensed, and the main features of the backbone extracted. This is achieved in Stage IV of the design. Eventually, the output is represented as a simple number that classifies the type of nonlinearities according to the main features of the backbone. In this study, the three basic linear/nonlinear types are conveniently classified using three numbers, $-1$, $0$ and $+1$ for softening nonlinearity, linearity and hardening nonlinearity, respectively. These values are produced via the “Signum” operation devised in Step 7, which is performed on the slope of the signal $\frac{d\omega}{dA}$. Steps 5 and 6 are intermediate steps to obtain a smooth curve $\frac{d\omega}{dA}$, which are currently under development.
Numerous simulations were performed in this study to validate every step and stage proposed in Fig. 1; some typical ones will be presented herein using the “Line” markers indicated in the same figure. Validation of the developed hardware design was carried out by comparing both hardware (System Generator) and software (Matlab) results using the same numerical procedure and input data. Though it is not strictly a light-to-light comparison as it involves the comparison between a fixed and floating-point scheme (and even single with double-precision option), this validation reveals how the performance of an embedded system compared with its general purpose computer counterpart.

Given the nature of an embedded hardware design, it is always critical to bear in mind the design criteria that produces the best performance. Knowing this, the proposed design is limited to a 1024-point Hilbert Transform. To ensure perfect signal reconstruction, 1024-point signals each of approximately ten cycles were adopted throughout the simulation. The data format in the hardware design used to represent the signal is signed fixed-point 16-bit, 15 binary point type, which indicates the need of using normalized input signals, i.e., the data within the range of \((-1, +1)\). For the purpose of testing Stages II to IV, monocomponent testing signals were used throughout to guarantee well-behaved Hilbert Transform.

Figure 5 presents the result of an initial test of the embedded Hilbert Transform (Stage II or Step 1 in Fig. 1) using a continuous sine wave as the input signal \(y(t)\). It can be seen that the hardware implementation produces near identical results to those from Matlab. The beginning of both signals (shown left in Fig. 5) deviate from the exact solution of the Hilbert Transform of a sine wave (which is a negative cosine wave). This behavior occurs since no anti-aliasing filter is implemented into both the FPGA design and Matlab counterpart at this preliminary stage of this study. The inaccurate results at both ends will be excluded from further analysis in Stage IV using a procedure named “Truncate” shown in Step 5 of Fig. 1. This is one of the tasks for future improvement together with Stage I Preprocessing.

![Figure 5: Validation of the embedded Hilbert Transform using the proposed one-FFT option. Shown here is the output of the proposed Stage II (Step 1) Hilbert Transform shown in Fig. 1 generated with System Generator overlayed with Matlab results. A sine wave of 1024 data points is used as the input signal, while a zoomed view of the first 256 points is presented here.](image_url)
Both Stages II and III in this design were validated using the displacement time histories from three distinct single-degree-of-freedom (SDOF) systems under free vibration subjected to nonzero initial displacement alone. The equation of motion is as follows:

$$m \ddot{x}(t) + r(t) = 0 \quad \text{with} \quad x(0) \neq 0$$

(4)

where mass $m = 1$ in this study, $x(t), \dot{x}(t)$ and $\ddot{x}(t)$ denote displacement, velocity and acceleration, respectively. $r(t)$ stands for restoring force. Linear, Duffing and softening Duffing restoring forces [1, 15] were used in this study to generate simulated data for validation:

- linear system: $r = 0.04 \dot{x} + x$
- hardening system: $r = 0.04 \dot{x} + x + 5x^3$
- softening system: $r = 0.0159 \dot{x} + x - 0.5x^3$

(5) (6) (7)

Figure 6(a) displays the simulated free vibration time histories, while Fig. 6(b) shows the corresponding restoring force versus displacement plots. Note that the total of 1024 points for each system is interpolated uniformly from the original simulated results. The characteristics of linearity and hardening and softening nonlinearities are prominent using both the restoring force and displacement as shown in Fig. 6(b). In contrast, the displacement time histories (shown in Fig. 6(a)) do not show obvious characteristics. The goal of this hardware design is to merely use the displacement signal and extract the underlying characteristics utilizing backbone curves following [3, 5].
Figure 7: Validation of the proposed **Stage II** and **III** values along **Lines b, c and d** shown in Fig. 1 using both System Generator and Matlab. The three input data sets are presented in Fig. 6(a).
curves for the three selected SDOF systems are in consistency with those introduced in [3]. Clearly, there is a need to extract smooth trend lines from these plots - the focus of the proposed Step 5 (see Fig. 1).

![Figure 8: Validation of the proposed Stage II and III together with the truncation operation in Step 5. The Truncation operation removes 12.5% of the beginning and end of the data input, respectively. The three input data sets are presented in Fig. 6(a).](image)

While further research and development is currently underway on Steps 5 Extraction and Smoothing and 6 First-Order Derivative and CORDIC Division, Step 7 Sign Identification was completed and validated. In short, the function of Step 7 is equivalent to a signum operation. The output of this step is a sign identification (+1, 0, or -1 for positive, zero, or negative input data, respectively) at Line h. A straightforward validation using a periodic signal is presented in Fig. 9 showing proper functioning of this step.

![Figure 9: Validation of Step 7 Sign Identification. The operation of this step can be compared to that of the signum function, with the green “dots” representing the outputs of the Sign Identification operation.](image)
4 DISCUSSION

To fully implement the proposed design within an FPGA device, it is useful to understand the amount of resources consumed by the design. Hardware resource consumption was estimated respectively for Stages II to IV of the design and is listed in Table 1. There are multiple choices of available Xilinx FPGA devices to support the current design’s total resource consumption. For example, Table 2 lists hardware resource capacities from the smallest to largest FPGAs in the Xilinx Virtex-II Pro family [22]. It is important to keep in mind that the design’s resource utilization will vary as the design progresses from a full hardware implementation to a combined FPGA-microprocessor implementation. It is desired to develop an optimal design utilizing the lowest possible amount of hardware resources by working with low-end yet capable devices.

<table>
<thead>
<tr>
<th>Design Portion</th>
<th>Slices</th>
<th>FFs</th>
<th>LUTs</th>
<th>BRAMs</th>
<th>Mults</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage II*</td>
<td>1,206</td>
<td>1,927</td>
<td>1,834</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Stage III*</td>
<td>1,457</td>
<td>1,660</td>
<td>2,733</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>Stage IV†</td>
<td>2,431</td>
<td>3,491</td>
<td>3,944</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>5,294</td>
<td>7,078</td>
<td>8,511</td>
<td>26</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 1: Resource estimations for Stages II - IV. *Utilizing one-FFT option. †CORDIC arctangent of 10 processing elements. †Resources estimated on preliminary Stage IV design. CORDIC division of 10 processing elements.

<table>
<thead>
<tr>
<th>Design Portion</th>
<th>Slices</th>
<th>FFs</th>
<th>LUTs</th>
<th>BRAMs</th>
<th>Mults</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2VP2</td>
<td>1,408</td>
<td>2,816</td>
<td>2,816</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>XC2VP4</td>
<td>3,008</td>
<td>6,016</td>
<td>6,016</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>XC2VP30</td>
<td>13,696</td>
<td>27,392</td>
<td>27,392</td>
<td>136</td>
<td>136</td>
</tr>
<tr>
<td>XC2VP100</td>
<td>44,096</td>
<td>88,192</td>
<td>88,192</td>
<td>444</td>
<td>444</td>
</tr>
</tbody>
</table>

Table 2: Hardware resources available in various Xilinx Virtex-II Pro devices (Virtex-II Pro 2, 4, 30, and 100) [22].

5 CONCLUSION

This paper presents an initial design of a nonlinear pattern classifier to eventually be embedded within an FPGA and microprocessor-based system. The Hilbert Transform and the backbone identification technique are embedded completely into an FPGA-based design at this development stage. The Xilinx System Generator environment has been adopted throughout the hardware design, with Matlab results used for validation of the hardware simulations. Various design options have been explored to achieve an optimal hardware design. Capabilities presented by new and emerging FPGA technology is researched and utilized to achieve higher processing speed and efficiency with the design. A portion of this hardware design will be further implemented into a microprocessor system. This future configuration will be developed to yield the highest optimization in power consumption, processing speed, and precision.
6 ACKNOWLEDGEMENTS

This study is supported by the National Science Foundation under CMMI-0626401 and the program officer Dr. S.-C. Liu. The generous donation from the university program at Xilinx Inc. (XUP) is gratefully acknowledged. The first author wants to acknowledge the Palace Acquire Program of United States Air Force for the support of his graduate study. The second author would like to thank Professor Joseph P. Wright from Weidlinger Associates Inc. on teaching her Michael Feldman's work as early as in 2000. Insights offered by Professor Joseph P. Wright from Weidlinger Associates Inc. on some numerical challenges in this study are greatly appreciated. Several helpful discussions offered by Mr. Jeff Weintraub from Xilinx University Program are greatly acknowledged. The authors would also like to thank Chetan Kapoor for his help with the initiation of this study and recognize his previous work that offered a solid foundation for this study to be built upon. The assistance offered by Krisda Piyawat in understanding the Hilbert Transform and backbone curve as well as developing some Matlab codes for cross validation is greatly appreciated. An initial discussion with Professor John. K. Antonio on FPGA technologies from School of Computer Science at the University of Oklahoma is also acknowledged.

References


