# Independent Source Architecture for Developing FPGA-based Physical Layer Security Techniques

James Chacko, Marko Jacovic, Cem Sahin, Nagarajan Kandasamy, Kapil R. Dandekar
ECE Department, Drexel University, 3141 Chestnut St, Philadelphia, PA, 19104
{jjc652, mj355, cs486, nk78, krd26}@drexel.edu

***Abstract*—This paper describes and verifies a method of implementing bit error rate (BER) calculation for FPGA-based physical layer security techniques for Software Defined Radio (SDR). Specifically, we describe an independent source signal processing architecture for an efficient calculation of BER for wireless communication modules across the transmitter and receiver nodes. The source components at the transmitter and the receiver both generate identical random bits independently from each other, allowing for the received data to be compared to the original bit stream to calculate BER completely on hardware. The described method is implemented on a Xilinx Virtex-6 ML605 FPGA and reduces processing time by more than four orders of magnitude less than hardware simulation techniques in regression testing and validation over billions of bits, shortening design turn around times and accelerating Physical layer based security development for wireless communication research. The described independent source approach utilizes a minimal amount of board resources, allowing it to be integrated seamlessly into SDR hardware designs. Experimental validation of the independent source based BER calculation is performed for an Orthogonal Frequency Division Multiplexing signal, and a comparison between different stages of hardware design for the execution time required for BER testing of a large number of bits is provided.**

***Index Terms*—Software defined radio, Wireless communication, Communication Systems Security, Encryption, Field programmable gate arrays, Random number generation**

## I. Introduction

The research and development of baseband physical (PHY) layer security techniques involve first creating specific sets of baseband modules inherent to the wireless communication standard being tested. Often such development starts off in script implementations on general purpose processors and then moves towards a FPGA or ASIC hardware implementation with one or more intermediate steps distributed across software and hardware designs. Software Defined Radio (SDR) allows for rapid prototyping of communications system based research, given the flexibility provided by software and the speed of hardware, making it widely used in the development of wireless testbeds.

While generating data to be transmitted during scripting and its intermediate stages can be easily sourced off a host PC, this process becomes much slower and complicated when the target is a complete hardware implementation. Simulations of hardware modules are very time consuming, depending on the complexity of the design, and encouraged only for shorter evaluation runs. Simulations of hardware involving larger datasets weigh down the sourcing/simulating host PC for a significant time while generating useful data. For such cases

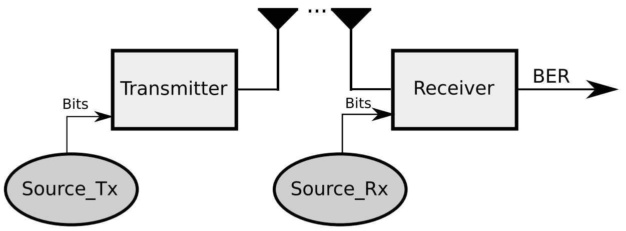

Fig. 1: Independent Source Architecture Overview. *Source_Tx* component generates a random bit stream for transmitter module, transmitter sends signal to receiver, receiver processes signal and recovers transmitted bits, recovered bits are compared to the bits generated by the *Source_Rx* component to calculate BER.

a hardware-based implementation [1] is much more practical for faster experimentation, verification and post processing. Scripting languages like MATLAB and Python are capable of sending and capturing data directly from hardware for post processing; however this requires knowledge of the interface bandwidth constraints that become the bottleneck.

This paper develops an on-board data source that generates data on both the transmit and receive side independently in a packet based approach, not requiring a backbone connection to the source generator as shown in Fig. 1. As our system does not require a source generator to be directly attached to multiple nodes, we reduce physical testing limitations related to wired connections and testing environments. In addition, our approach avoids applying large latencies to data transferred from the source to the receiver, which reduces FPGA resource utilization. Our system is capable of adjusting the total number of bits generated on board at run-time, while other shared seed approaches [2] are unable to change the number of bits generated without rebuilding their FPGA design. The implemented hardware design described in this paper makes an excellent tool for physical layer security development within wireless communication testbeds due to its ability to generate data in a much shorter period of time in comparison to a software-based SDR approach, which allows for a quicker verification of system performance.

Hardware development for SDR physical layer security research involves stages of script simulation, hardware simulation, hardware co-simulation, and hardware implementation as shown in Fig. 2. Development begins with a script implementation due to its high design flexibility and debugging attributes. Verification of a principal concept is easily obtained

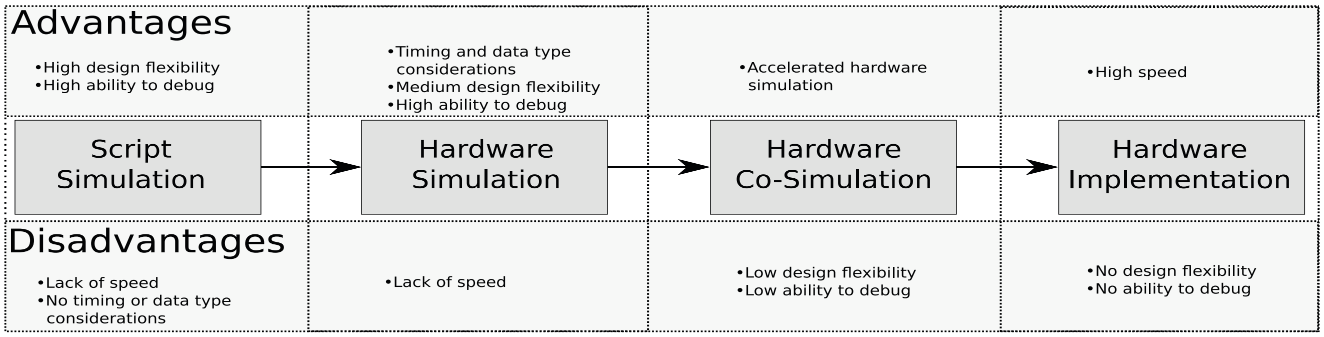

Fig. 2: Hardware-based SDR Design Process. Stages are often implemented sequentially, and are shown with associated advantages and disadvantages. Earlier stages focus on design, while the focal point of hardware implementation is performance.

in this stage; however, script simulation suffers from a lack of speed and does not consider aspects of timing and data type propagation (floating/fixed-point) that are inherent design constraints for hardware. The next step involves implementing the same design in a hardware development environment for simulation, which is an essential task since it involves timing and data type considerations while maintaining design flexibility and debugging capabilities. Hardware co-simulation involves exporting subsets of the design into hardware in phases prior to the entire system being ported into hardware, allowing for portions of the design to still be run on software. Co-simulation allows for a hardware accelerated simulation to be performed, with tradeoffs of lower flexibility in design and debug capabilities than in pure simulation. In the stages of script simulation, hardware simulation, and hardware co-simulation previously described, the dependence on a software-based source throttles the speed capabilities of the system. A purely hardware-based approach achieves speeds that are magnitudes higher than the other stages discussed previously, which will be examined in detail in Section IV. A hardware implementation does not allow for design changes or debugging; however, since the design process is sequential, implementation would not be completed until the prior three stages were thoroughly performed and evaluated, therefore the features are not required at this stage.

The independent source approach for bit error rate (BER) calculation in this work is designed with the primary goal of implementing an efficient method of testing physical layer security techniques within wireless SDRs. The final hardware implementation of the independent source architecture is an IP core that is capable of being targeted to several FPGA-based SDR platforms, and may also be used to test subsystems of an SDR for validation. The Drexel Software Defined Communication (SDC) testbed is a hardware based SDR with a software driven interface, which allows for rapid prototyping of wireless communication systems as described in [3]. The independent source design is developed on the SDC platform and shares the development flow shown in Fig. 2. Though pure software-based research provides design-time and run-time flexibility with the ease of debugging, experiments in the area of wireless communications need to be validated over a wide range of data that can be best achieved only with hardware solutions not bottlenecked by software processing speeds.

This paper describes our considerations and challenges in implementing such a signal independent source architecture along with discussing its significance and relevance for PHY layer security development. The key contributions are in providing researchers working in hardware-based wireless security with the ability to $i)$ rapidly run and validate FPGA-targeted security modules for large data sets, $ii)$ save resources on-board due to its low profile implementation, and $iii)$ generate data on-demand and eliminate the need for the original uncorrupted bits to be connected directly from the transmitter source to the receiver output for comparison.

## II. Related Work

A key aspect of the design in this paper is based on random number generation, as bits are generated on both the transmitter and receiver sides for BER calculation. Random number generation for communications is primarily focused on coding and cryptology techniques. A method of developing codes for Direct Sequence Spread Spectrum signals on FPGA was developed using LFSRs in [4]. A combined decimal sequence pseudo random number generation approach was implemented on hardware in [5] to develop spreading sequences for Code Division Multiple Access. In both scenarios, the contributions were focused on code generation, and were not used as a source of data for the communication systems. A true random number generation method was performed for the purpose of cryptology in [6]; however, a true random number generator is not desirable in this contribution, as it is necessary for both sources to generate an identical pseudo-random bit sequence. Bit generation has been done previously in [7], which implements a Cyclic Redundancy Check to determine error rates but was not explicitly used in SDR. An implementation of bit generation was presented in [8], but was only considered at the transmitter, and was not extended for an efficient BER calculation to be used in testing SDRs. BER calculation has been implemented for wireless applications previously in [2]; however, in their design, the source bits are wired directly to the receiver for comparison to the received bits, which requires additional latency to be introduced to the connected source bits for analysis. Adding latency to data streams on hardware is a costly operation pertaining to

TABLE I: Resource estimation targeted to a Xilinx Virtex-6 ML605 Board using 32 Fibonacci generators.

| **Resource Name** | **Count** | **% Utilization** |
|---|---|---|
| LUT's | 4166 | 2.76 |
| FF's | 5988 | 1.89 |
| BRAM's | 0 | 0 |
| Mult/DSP48 | 0 | 0 |

resource utilization [9]. The design also requires additional infrastructure which may result in limited testing capabilities with regards to the distance between the radios and experimental setups. Another implementation of BER calculation on-board an FPGA is described in [10], which also does identical data generation both on the receiver and transmitter side based on multiple pseudorandom binary sequences. However, [10] does not provide any details on implementation complexity and required hardware resources. The independent source architecture developed in this paper follows a combination of simple Xilinx IP based modules in implementing our Bit Error Rate Tester (BERT) with minimal resource utilization. This paper is focused on describing our step by step approach designing our BERT system for SDR based physical layer security applications. Our design keeps resource utilization low which is vital to the SDC [1] [11] research testbed enhanced through this design. SDC is used for rapid prototyping within PHY layer modules in Orthogonal Frequency Division Multiplexing (OFDM) based wireless standards.

## III. Independent Source System Architecture

The independent source system described in this paper follows the main design concept of the SDC testbed, by being insensitive to latency changes in baseband modules across the transmitter and receiver. SDC achieves this design goal using modules built with intelligent controllers that avoid data overflow or starvation during its operation. A key motivation of the independent source design emerged from the requirement of SDC to have adjustable data rates that are capable of changing at run-time [11]. The transmitter source module, *Source_Tx*, implements this on-demand concept by being able to generate and throttle data, while the receiver source module, *Source_Rx*, independently generates original data at the receiver and calculates BER upon the arrival of propagated data. Next we will describe specific implementation details of the source components in our system.

***Source_Tx***: The detailed architecture of the *Source_Tx* module described in this paper is shown in Fig. 3. The inputs of the module consist of the number of *bits_per_cycle*, a *reset* signal, and an *enable* signal, while it outputs the *generated bits*. Individual Fibonacci generator blocks, which are LFSR based series generators from the Xilinx IP library [12], were uniquely seeded based on the number of *bits_per_cycle*. The *reset* signal was then toggled low to initialize the system. Upon the reception of an *enable* signal, that represents the ready state of the block accepting the bits, each unique Fibonacci generator output an unsigned 1-bit wide (UFix_1_0) fixed-point data type. These data bits were then bit-bashed/concatenated through a parallel to serial block. The serialized data were then truncated using bit slicers that provided different input ports to a multiplexer block that selected the appropriate slice to output the *generated bits* based on the *bits_per_cycle* signal input.

***Source_Rx***: The *Source_Rx* module of the independent source architecture has a very similar internal structure to the *Source_Tx* block. The inputs consist of the *received_bits*, a *received_bits_valid* signal, the number of *bits_per_cycle*, and a *reset* signal, while it outputs the *number of bits received* and the *number of bit errors*. Upon reception of a bit stream at the receiver, the *received_bits_valid* signal showed logical HIGH when the data were valid, which allowed for the valid bits to be differentiated from the invalid bits. The *received_bits_valid* signal was used to enable the uniquely seeded Fibonacci generator blocks, and the number of *bits_per_cycle* was used to select the appropriate bit sliced output. The generated bits at the *Source_Rx* were then compared to the *received_bits* using a comparator. The *number of bits received* was obtained by accumulating the samples of the *received_bits_valid* signal, while the *number of bit errors* was determined by adding the comparator output. Both the outputs were also stored in shared memory registers that were accessible to the on-board microblaze processor on the Virtex 6 ML605 board to enable querying the values from the host.

***Resource Utilization***: A summary of the number of estimated resources required to implement the hardware design of the independent source architecture for 32 Fibonacci Generators is provided in Table I. The percentages shown are based on the ML605 Virtex-6 evaluation board. A small number of look-up tables and flip-flops are required for the design, as only 2.76 and 1.89 percent of the total respective resources of the board are used. Neither Block RAMs nor multipliers are used in the design since each resource type is considered scarce. As an example, the target FPGA used in this paper has 1100 total Block RAMs and 768 total multipliers. Due to the resource efficient independent source architecture that was implemented, the system may be integrated into hardware SDR designs without causing resource limitation issues that would hinder SDR design capabilities. In addition, it may be noted that the estimated resource utilization shown in Table I is based on using 32 Fibonacci Generators, which may be scaled down to further reduce the number of resources required.

A key benefit of the system developed in this paper is that the generated bits from the transmitter are not required to be linked directly to the receiver for a loop-back BER calculation, as is the requirement of a single source approach for a full hardware implementation. For example, the independent source system enables testing the developed physical layer security techniques on nodes without having the need to share predetermined data for comparison, saving storage and time. Block RAMs are required for storage, which are scarce resources as previously described in this section. With regards to timing, complex transmitter and receiver designs require a greater number of clock cycles to complete the loop-back process, which results in

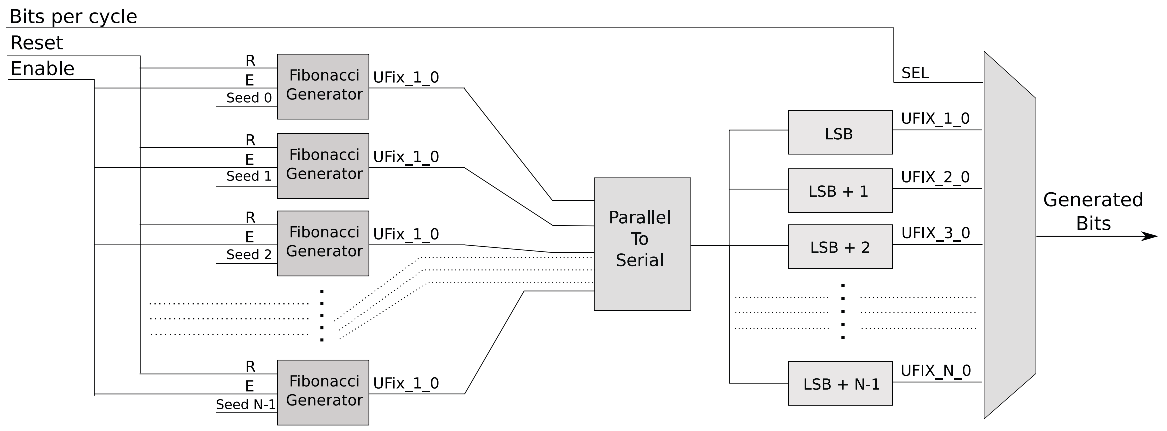

Fig. 3: Detailed *Source_Tx* Architecture. The *Source_Tx* block consists of individually seeded Xilinx Fibonacci series blocks that generate data on demand based on an enable signal and the number of bits required per cycle.

the generated bits being delayed substantially for synchronized BER calculation. Large delays of data streams on hardware result in a substantial increase in resource utilization in terms of memory [9]. Each delay used requires a flip-flop and a shift register consisting of a look-up table as listed in [12]. The independent source architecture implemented in this paper reduces the number of FPGA resources utilized, allowing for a greater number of resources to be dedicated to other aspects of the physical layer design.

## IV. Experimental Procedure

The independent source architecture is capable of being used with any communication module that takes serial or parallel data, ranging from 0 to 32 bits, and gives data out in the same format. For our experiment we selected to secure an OFDM signal, implemented using the SDC testbed. OFDM is widely researched in wireless communications due to its benefits of spectral efficiency, robustness to severe channel changes, and other advantages as described in [13]. The PHY security technique being tested consists of a simple pseudo-random sequence uniquely seeded based on a key, known only to the transmitter and receiver, being used to convolute the data line before the QPSK mapper on the transmitter and after the QPSK decoder at the receiver. It was observed that a mis-match in the key always resulted in a BER of around 50% which is close to random suggested the success of the technique. This paper focuses more on the testbed data sourcing that enables the validation of the PHY security techniques as the one described above. A simplified version of an OFDM signal chain consisting of a transmitter, receiver, and Additive White Gaussian Noise (AWGN) channel is shown in Fig. 4. The transmitter generates bits using the *Source_Tx* architecture described in Section III, maps the bits to Quadrature Phase Shift Keying (QPSK) symbols, and performs multi-carrier modulation using an Inverse Fast Fourier Transform (IFFT) to obtain the time domain transmitted signal. QPSK was selected instead of higher orders of QAM due to its lower theoretical BER in AWGN channels. The ability to generate a large amount of bits very rapidly allows for accurate evaluation of low values of BER, which is a key advantage of our system. For example, to compute a BER of $10^{-8}$, the theoretical minimum of bits needed to be generated is $10 \times 10^6$, while an accurate approximation would require the number of bits considered to be magnitudes greater than the minimum. The resulting signal from the transmitter is passed through an AWGN channel with a targeted level of energy per bit to noise power density ratio ($E_b/N_0$) to introduce distortion. At the receiver, a Fast Fourier Transform (FFT) is performed on the resulting signal from the channel to recover the frequency domain QPSK symbols, a maximum likelihood decoder is used to obtain the received bits, and the BER is calculated by comparing the received bits to the bits generated by a *Source_Rx* described in Section III. As the first part of our experiment we validate the BER measurements and motivate the independent source architecture we have developed in this paper by showing the ease and accuracy with which it produces a close to theoretical BER curve over an $E_b/N_0$ sweep of 10 billion bits through an AWGN channel. A second experiment is performed to further motivate the independent source architecture by showing the execution timing results running the loop-back system described in Fig. 4 across its implementations on hardware simulation, hardware co-simulation and hardware implementation by transmitting and receiving $1 \times 10^6$ up to $5 \times 10^6$ bits. The number of bits analyzed in the second experiment was substantially decreased in comparison to the first experiment due to the large length of time required for bits to be generated in hardware simulation and hardware co-simulation approaches, demonstrating a benefit of our full hardware implementation.

### A. Experiment 1: BER Validation

Validation of the independent source design was performed by comparing the BER calculated on hardware to the ideal values for a QPSK signal over an AWGN channel with a varying range of $E_b/N_0$ values. The theoretical BER for QPSK as a function of $E_b/N_0$ is given in [13] as

$$BER_{QPSK} = Q\left(\sqrt{\frac{2E_b}{N_0}}\right), \tag{1}$$

where Q(.) is the Q-Function. In the experiment, $10 \times 10^9$ bits generated by the *Source_Tx* were sent from the transmitter to

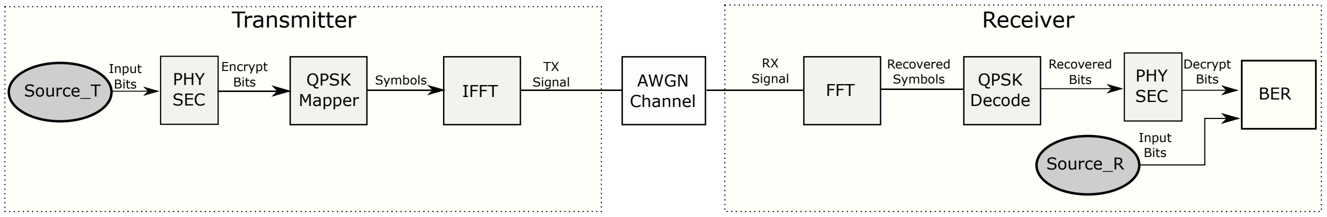

Fig. 4: Experimental Layout. Design consists of a simplified OFDM signal.

the receiver for each level of $E_b/N_0$ and the BER was calculated by comparing the recovered bits to the bits generated on the *Source_Rx* at the receiver as

$$BER = \frac{1}{N}\sum_{i=0}^{N-1} d_i \oplus \hat{d}_i, \qquad (2)$$

where $N$ is the number of bits transmitted, $d$ are the generated bits, $\hat{d}$ are the recovered bits, and $\oplus$ is the exclusive OR operation. A comparison of the hardware implementation results to the ideal BER given by (1) is shown in Fig. 5. The percent error of the hardware results from the theoretical BER values over a range of $E_b/N_0$ from 0 to 13 dB was determined to be 7.03% relative to the theoretical value. An increase in the number of bits considered in the range from 10 to 13 dB would further decrease the error as expected from a probabilistic standpoint. The entire experiment took 17.33 minutes to perform on hardware, processing a total of 260 Gbits over the sweep (inclusive of print and status interrupts), demonstrating the efficiency of the test method. Overall the purpose of this experiment was to affirm that the BER testing system performed as expected prior to analyzing benefits regarding execution time. A loopback design was selected to provide a controlled testing environment with an accurate AWGN channel for verification purposes. The source blocks did not share data, and were independent of each other, as the system is designed to work for wireless scenarios.

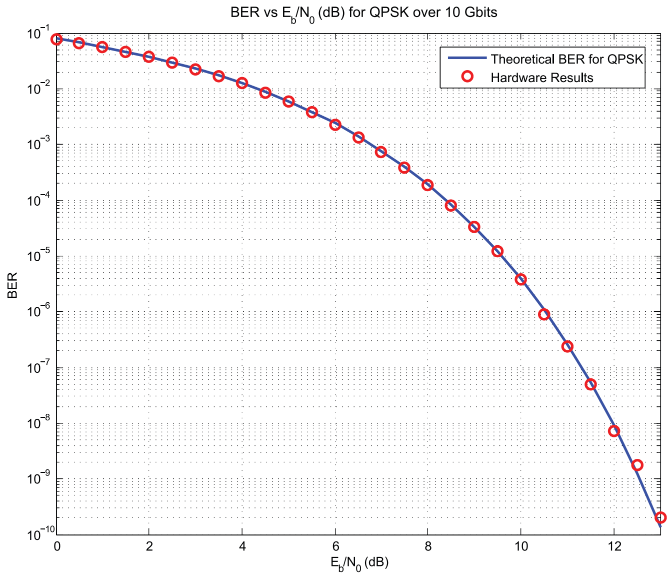

Fig. 5: BER of hardware experiment over AWGN Channel. Theoretical value of BER calculated by (1) shown by solid line, hardware results displayed by circular markers. Hardware BER at each level of $E_b/N_0$ is determined using (2) on-board using 10 Gbits. Comparison of hardware results to theoretical values validates accurate BER calculation.

## B. Experiment 2: Timing Analysis

To represent a valid workload we used SDC to build a simplistic OFDM based physical layer consisting of QPSK modulation and IFFT modules at the transmitter side along with the corresponding blocks performing the QPSK de-modulation and FFT at the receiver side. The source blocks are natively built using Xilinx SysGen modules within the MATLAB Simulink environment. For the pure software-based simulation and co-simulation setup, this implementation enables ports from the *Source_Rx* module to output the *'number of bits'* received and the *'error count'* directly to the MATLAB workspace, from which they may be processed both quantitatively and qualitatively across a given workload.

*Hardware Simulation*: In order to run a pure simulation based timing analysis, a MATLAB script was used to sweep the source module to generate datasets consisting of from 1 Mbits up to 5 Mbits at the transmitter and measure the time taken to completely send and receive the generated dataset across the workload described earlier in this section. This is a software-based simulation setup useful for researchers who would like verify radio components in MATLAB across a large number of bits using the independent source modules.

*Co-Simulation*: The co-simulation framework consists of having either the entire or part of the model under test implemented on hardware for accelerated performance. For this timing comparison, we synthesized the independent source blocks and the workload onto the Virtex6 ML605 FPGA, and left the control to start the system as part of the MATLAB software. In this setup MATLAB was used to control the system's execution for a set number of bits and provide the execution time and BER upon its completion. MATLAB does this operation by sending control signals over its Gigabit Ethernet connection to the Virtex6 ML605 board to start the processing and also grab the data from the *Source_Rx* block into its workspace.

*Hardware Implementation*: In this section the entire project consisting of the source modules, the workload and the control was synthesized on to the Virtex6 ML605 board unlike the co-simulation framework where the control was from software. Control of the system was done through the on-board microblaze processor that handled starting and recording the *'number of bits'* received and the *'error count'* on to shared

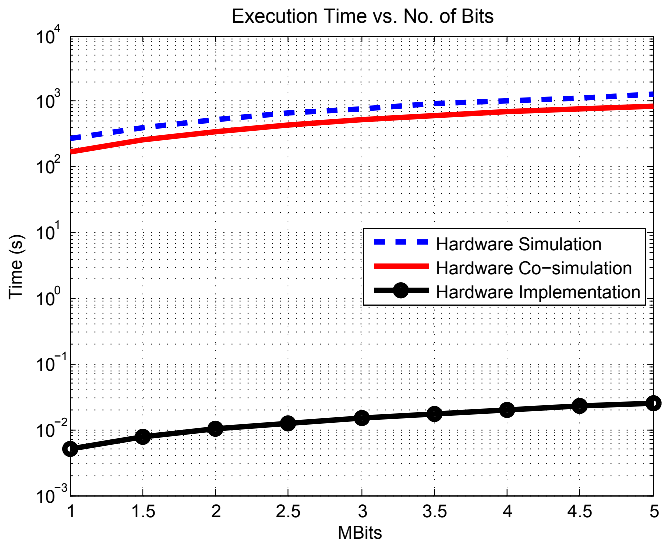

Fig. 6: Execution time over three stages of hardware design. The effect of the number of bits processed by the independent source architecture on elapsed time is analyzed for 1 to 5 Mbits. Hardware co-simulation required less time than hardware simulation, however both stages require greater than 4 orders of magnitude of time than for hardware implementation.

register space that could be accessed through a register read from the SDK workspace. Timing experiments consisted of writing C scripts in the SDK environment to set the number of bits to be sent and starting the transmitter side and reading the processed data from the receiver side on hardware.

*Timing Results*: A comparison of execution time required to process the bits through the independent source architecture is shown in Fig. 6, where the dashed line represents the hardware simulation results, the solid line is based on the hardware co-simulation, and the circular marker represents the hardware implementation. The execution time required was analyzed for each stage of the hardware design process for the transfer of 1 Mbits up to 5 Mbits. Hardware simulation was determined to take the longest amount of time as expected, while hardware co-simulation was slightly quicker than simulation. Hardware acceleration in co-simulation accounted for the decrease in time required; however, the software-based control caused the execution time to be much longer than for a pure hardware implementation. Over the course of this experiment hardware co-simulation required an average factor of over 33,000 times the amount of time required for hardware implementation, while hardware simulation required a factor of over 51,000. In both cases the stages required time greater than four orders of magnitude than the pure hardware implementation. The large difference in execution time required in this experiment highlights the benefit of the independent source architecture presented in this paper.

## V. Conclusions

A method for calculating BER independently across a transmitter and receiver based on an independent source architecture was introduced for the purpose of efficiently developing and testing the PHY layer security techniques for FPGA-based wireless testbeds. Identical pseudo-random bits were generated by both the source blocks to be used in the calculation of the BER on hardware, without directly linking the source data of the transmitter to the input of the BER component. Analysis of the resource utilization demonstrated that the independent source architecture uses minimal board resources, and is capable of being integrated into SDR testbeds. Experimental results obtained on a Xilinx Virtex-6 ML605 FPGA board using an OFDM signal design over an AWGN channel verified that the BER system performed as expected. The hardware implementation was able to process bits faster than hardware simulation and co-simulation by a factor of four orders of magnitude. This paper thus provided and validated an efficient hardware-based BER calculation architecture to be used for FPGA-based PHY layer security technique research.

## VI. Acknowledgments

This project is supported by the National Science Foundation through grants 1457306 and 1422964.

## References

[1] B. Shishkin, D. Pfeil, D. Nguyen, K. Wanuga, J. Chacko, J. Johnson, N. Kandasamy, T. Kurzweg, and K. Dandekar, “SDC testbed: Software defined communications testbed for wireless radio and optical networking,” in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2011 Int. Symp. on*, May 2011, pp. 300–306.
[2] A. Alimohammad and S. Fard, “FPGA-based bit error rate performance measurement of wireless systems,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 22, no. 7, pp. 1583–1592, July 2014.
[3] J. Chacko, C. Sahin, D. Pfiel, N. Kandasamy, and K. Dandekar, “Rapid prototyping of wireless physical layer modules using flexible software/hardware design flow,” in *Proc. of the 2015 ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays*. ACM, 2015.
[4] R. Stepien and J. Walczak, “Application of the DLFSR generators in spread spectrum communication,” in *Mixed Design of Integrated Circuits and Systems (MIXDES), 2012 Proc. of the 19th Int. Conf.*, May 2012.
[5] A. Mahfouz, K. Shehata, and M. Hanna, “A secure spreader/despreader for code division multiple access applications,” in *Signals, Circuits and Systems, 2008. SCS 2008. 2nd Int. Conf. on*, Nov 2008, pp. 1–6.
[6] S. Kwok and E. Lam, “FPGA-based high-speed true random number generator for cryptographic applications,” in *TENCON 2006. 2006 IEEE Region 10 Conf.*, Nov 2006, pp. 1–4.
[7] K. Siozios, G. Koutroumpezis, K. Tatas, D. Soudris, and A. Thanailakis, “Dagger: A novel generic methodology for FPGA bitstream generation and its software tool implementation,” in *Parallel and Distributed Processing Symp., 2005. Proc.. 19th IEEE Int.*, April 2005, pp. 165–165.
[8] R. Soni, N. Steiner, and M. French, “Open-source bitstream generation,” in *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annu. Int. Symp. on*, April 2013, pp. 105–112.
[9] S. M. Qasim, S. A. Abbasi, and B. Almashary, “A review of FPGA-based design methodology and optimization techniques for efficient hardware realization of computation intensive algorithms,” in *Multimedia, Signal Processing and Communication Technologies, 2009. IMPACT ’09. Int.*, March 2009, pp. 313–316.
[10] P. Sastry, D. Rao, S. Vathsal, and A. Rajaiah, “HDL design architecture for compatible multichannel multi-frequency rate serial bit error rate tester (BERT) ASIC IP core for testing of high speed wireless system products/applications,” in *Communication Systems and Network Technologies (CSNT), 2015 5th Int. Conf. on*, April 2015, pp. 839–843.
[11] J. Chacko, C. Sahin, D. Nguyen, D. Pfeil, N. Kandasamy, and K. Dandekar, “FPGA-based latency-insensitive OFDM pipeline for wireless research,” in *High Performance Extreme Computing Conf. (HPEC), 2014 IEEE*, Sept 2014, pp. 1–6.
[12] *System Generator for DSP Reference Guide*, 14th ed., Xilinx, Inc., October 2012.
[13] J. G. Proakis and M. Salehi, *Digital communications*, 5th ed. Boston, Mass: McGraw-Hill, 2007.