Low-Cost and Portable Interactive Sinusoidal Digital Signal Generator by Using FPGA

Aiman Zakwan Jidin[1,2], Irna Nadira Mahzan[1], Nurulhalim Hassim[1], Ahmad Fauzan Kadmin[1]
[1]*Faculty of Engineering Technology, Universiti Teknikal Malaysia Melaka, Melaka, Malaysia.*
[2]*Center for Telecommunication Research and Innovation, Universiti Teknikal Malaysia Melaka, Melaka, Malaysia.*
aimanzakwan@utem.edu.my

***Abstract*—This paper presents the development of a low-cost and portable interactive Sinusoidal signal generator which has been implemented on FPGA device. The sine wave is generated by using a Lookup Table method, where the sine values are pre-calculated and stored in the onboard memory. The frequency of the generated signal is modified by changing the value of the memory address incremental step. In addition, the implemented signal generator is serially connected to a graphical user interface (GUI) on a PC, which can be used to select the type of the desired signal to be generated and to set the signal frequency. The proposed design was successfully implemented in ALTERA Cyclone II DE0 FPGA Development Board, where the sine wave can be generated within the range of 1 kHz to 1 MHz, with 1 kHz frequency resolution.**

***Index Terms*—FPGA; Function Generator; Sinusoidal; User Interface.**

I. Introduction

Waveform generator or usually referred as function generator is an important tool which is very popularly utilized as the input signal generator for tests and experiments in various applications, such as telecommunication, control, measurement and teaching field [1]. A function generator is typically used to generate the signal with capabilities to accurately control the frequency and the amplitude characteristic to replicate the input signal of the circuit begin tested. It produced repetitive signals in waveforms like sinusoidal or sine wave, square or pulse wave, triangular wave, and sawtooth wave [2].

A function generator can be implemented on programmable devices such as a microcontroller or a Field Programmable Gate Array (FPGA). The former is widely used, owing to the simplicity of the system development. By using a high-level programming language like C language, users can use the predefined `sin()` function for example, in order to generate the sine wave. However, the execution time of microcontrollers are generally quite slow since all the instruction sets are executed sequentially and in addition, only one instruction can be executed at a time. Therefore, FPGA is an adequate solution for high-performance computations and it is widely used many high-speed applications, owing to its low cost, its ability to implement pipelined and parallel computations, and its capability to operate at high-frequency clocks [3,4].

There are several research which had been proposed in order to implement the functional or waveform generator on FPGA. Some had proposed the use of Direct Digital Synthesis (DDS), a popular technique which can produce outputs with high-frequency resolution and accurate frequency adjustment. DDS produces the analog signal by generating the time-varying signal in a digital form, then converted into the analog signal via digital-to-analog conversion. The principle of DDS is to vary the frequency of the clock which is used to read the waveform amplitude, which is digitally stored in a memory. Then, read data is converted to analog signal by using the digital-to-analog converter (DAC) [1,5,6,7].

On the other hand, research in [8] had implemented the waveform generator in Xilinx Virtex II FPGA, by using the embedded microprocessor. In this research, a soft processor called MicroBlaze, which control the system is interfaced to peripherals such as memories and DAC. The hardware configuration was done by using Xilinx Embedded Development Kit, whereas the software, which was written in C, was developed in Xilinx Software Development Kit. However, to achieve high-bandwidth signal generator, it is required to use high-end FPGA such as Virtex FPGA which cost very expensive.

Typical function generators in the market can be very expensive, depending on the performance and the features they provide. They are normally equipped with several knobs or the keypads or both, as the inputs to configure the desired signals to be generated. Meanwhile, research in [9] proposed the utilization of GUI as the control medium. In this research, a GUI was developed by using Visual Basic and an auxiliary USB controller is used to communicate the PC with FPGA.

This paper presents the implementation of an interactive function generator in FPGA which is controllable by using a GUI in a computer. In this paper, it will only focus on the sine wave generation which frequency can be varied within the range of 1 kHz to 1 MHz, with 1 kHz of frequency resolution. For this research purpose, no DAC is involved and thus, the proposed research produced the digital sine wave with accurate frequency. Sine wave signal is generated by adopting a technique which is quite similar to the DDS, but the memory address incremental step is tuned instead of the memory clock frequency. The GUI was developed by using an open source software called Processing.

II. Lookup Table For Sine Wave Generation

In order to construct a lookup table by using the onboard ROM, a specific memory initialization file is created. This file contains the information like the total number of data, the address and its corresponding data, and also the data format, either in binary or hexadecimal. This file is then used to initialize the contents of the ROM for the sine lookup table. In this research, it will be filled with 20000 samples from one full cycle of the proposed system base signal, which is 1 kHz.

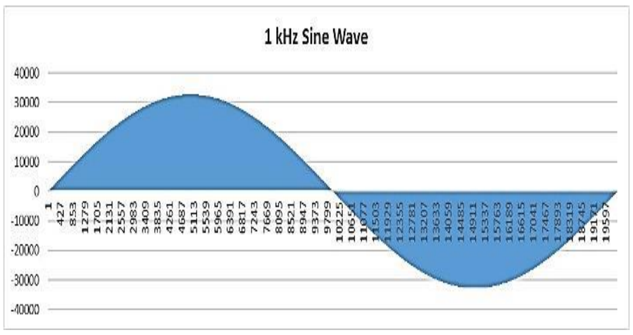
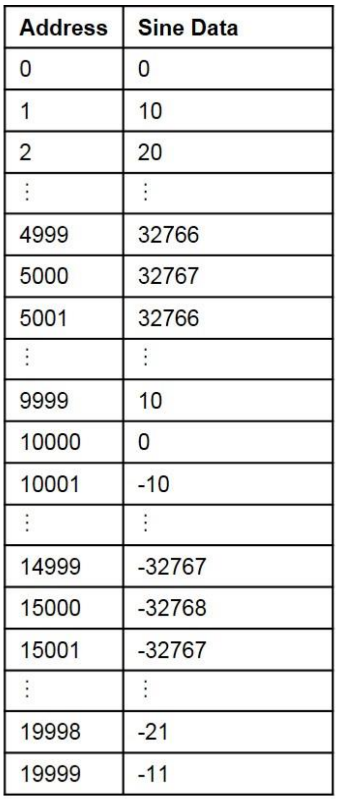


Figure 1: 1 kHz sine wave composed of 20000 samples.

This table has the address bus and the clock signal as the inputs, plus the sine data as the output. Since it contains 20000 samples of data, the address bus width is set to 15 bits. For this research purpose, the output sine data is set to 16 bits. The input clock will serve as the sampling clock. At every clock cycle, the address of the table will be increment by an incremental step value, which is equal to the desired sine wave frequency to be generated in kHz unit. For example, for a 20 kHz sine wave, the address will be increased by 20. This process will be repeated until it reaches the end of the table, before restarting back from the beginning.



Address	Sine Data
0	0
1	10
2	20
⋮	⋮
4999	32766
5000	32767
5001	32766
⋮	⋮
9999	10
10000	0
10001	-10
⋮	⋮
14999	-32767
15000	-32768
15001	-32767
⋮	⋮
19998	-21
19999	-11

Figure 2: Lookup table for sine wave generation.

In this case, the choice of the table clock frequency is substantial. As previously mentioned, the base signal frequency F_{base} = 1 kHz. Hence, the base period T_{base} = 1 ms. Therefore, in order to determine the sampling clock period:

$$T_{sampling} = \frac{T_{base}}{n} \tag{1}$$

where n the number of samples. From Equation (1), the period of the sampling clock is equal to 50 ns. In other words, the address of the table will be increased at every 50 ns.

III. RS-232 COMMUNICATION PROTOCOL

RS-232 is a standard for serial communication which is used for data transmission and reception. It normally connects a data terminal equipment (DTE) such as a computer terminal or a graphical user interface (GUI) to a data circuit-terminating equipment (DCE) like modems or any controllable devices which equipped with the RS-232 interface.

The data transmission through RS-232 can be set to various baud rate: 9600 bps, 19200 bps, 38400 bps, 57600 bps or 115200 bps. The baud rate is the rate at which data is being transferred through a communication channel and it is usually measured in bits per seconds (bps).

In order to connect a DTE to a DCE by using RS-232 communication, a cable equipped with DB9 connector is required. This connector contains 9 pins and the functionality of each DB9 connector pin is described in Table 1 [10].

Table 1
Function Description of DB9 Connector Pins

Pin	Function
1	Carrier Detect
2	Receive Data (rxD)
3	Transmit Data (txD)
4	Data Terminal Ready
5	Ground
6	Data Set Ready
7	Request to Send
8	Clear to Send
9	Ring Indicator

In the proposed system, the RS-232 is used by FPGA only for reception purposed. Thus, only the Transmit Data pin (pin 3) at the computer is used and connected to the Receive Data pin (pin 2) of the FPGA.

In RS-232 serial communication, each data will be transmitted in a packet of 10 bits, which is consisted by 1 start bit (set at low voltage), 8 bits of data (starting from LSB to MSB) and 1 stop bit (set at high voltage). Figure 3 shows the example of a data 01010011b is being transmitted from a DTE to a DCE.

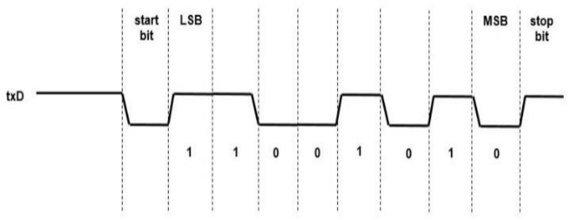


Figure 3: RS-232 communication waveform.

The FPGA, which acts as the DCE, will receive the data and store it in an 8-bit register. It will start receiving when the received data becomes low (start bit) and then a counter inside the FPGA will count up from 0 to 7. After that, the transmission of 8-bit data is completed and FPGA shall receive high signal (stop bit). Each time a new bit is received, the content of the register will be shifted to the right first and the new bit is stored at the MSB of the register since the least significant bit (LSB) is transmitted first.

IV. GUI Development

For this research purpose, a simple and basic GUI on the computer is developed in order to configure the type and the frequency of the signal to be generated by the FPGA board. It contains a drop-down menu which is used to select the type of signal among several options: sine wave, triangular wave, sawtooth wave, and pulse wave. Besides, a textbox is also available in the GUI, where users can use to set the desired signal frequency. A clickable button is also added and therefore, the new signal configuration will be sent to the FPGA board via RS-232 communication by clicking it.

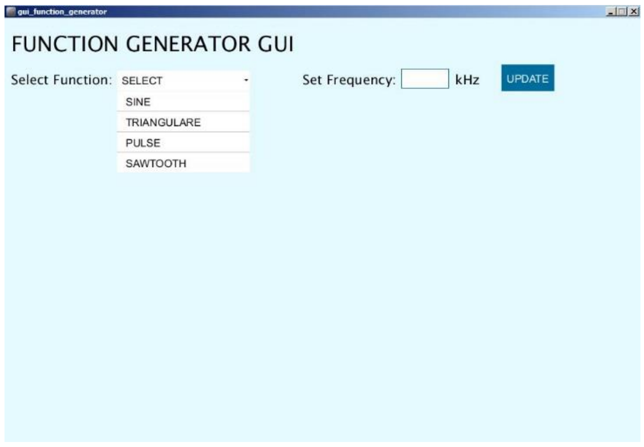


Figure 4: GUI for proposed function generator.

The GUI which has been developed for this research is shown in Figure 4. The process to set the new configuration of the signal to be generated is depicted in Figure 5. For the moment, only the sine wave can be generated by the proposed system, whereas the other options are kept for future works.

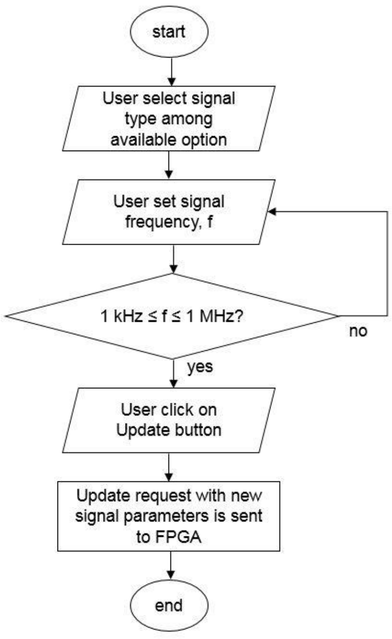

start
User select signal type among available option
User set signal frequency, f
1 kHz ≤ f ≤ 1 MHz?
no
yes
User click on Update button
Update request with new signal parameters is sent to FPGA
end

Figure 5: GUI process flowchart.

The GUI has been developed by using the Processing Software, which is a widely used tool within the context of the visual art [11]. By using it, the GUI can be designed and configured by using the Processing language, which is based on Java language. By consequences, it shall contain all the Java libraries, in addition to the user-defined libraries.

One of the most important libraries used during this GUI development is called controlP5. This user-defined library is utilized to add and configure the buttons, the text box and the drop-down menu to the GUI [12].

V. Proposed System Architecture

Figure 6 presents the block diagram of the proposed system architecture. There are 5 subcomponents inside the FPGA: PLL, Controller Finite State Machine (FSM), RS232 Controller, Clock Divider and Sine Lookup Table. The PLL and the clock divider are needed in order to obtain a sampling clock (*sampling_clk*) with period 50 ns. Since the onboard oscillator generates 20 ns clock signal, the PLL is needed in order to produce a faster clock (*pll_clk*) with the period equal to 10 ns. Then, the *pll_clk* will be divided to 5 in order to produce a 50 ns clock signal.

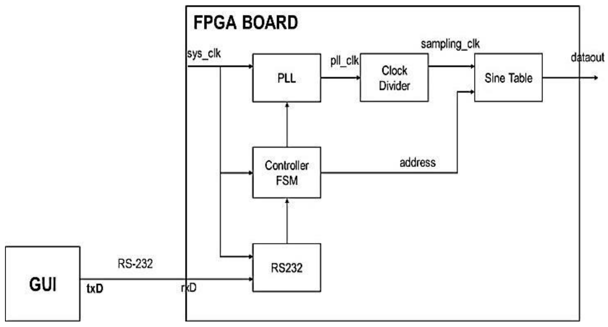
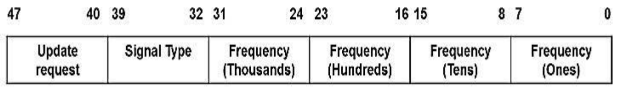


Figure 6: Block diagram of the proposed system architecture in FPGA.

The RS232 controller is the block which manages the reception of the data transmitted from the GUI, via RS-232 serial communication. It will receive a total of 48 bits data (excluding start and stop bits) which are stored in a 48-bit register called *data_buffer*. It is composed of six 8-bit data which represent the update request flag, the requested signal type flag and also requested signal frequency. The composition of this register is shown in Figure 7.



47 40	39 32	31 24	23 16	15 8	7 0
Update request	Signal Type	Frequency (Thousands)	Frequency (Hundreds)	Frequency (Tens)	Frequency (Ones)

Figure 7: Composition of data_buffer register in RS232 Controller.

Once received the request for an update, the Controller FSM will identify the requested signal type and its frequency, before initiating the reset of the system. Since the PLL is also being reset, the system needs to wait for *the pll_locked* flag before proceeding to the sine wave generation. Next, the new address incremental step will be provided to the Sine Lookup Table, where the incremental step is equal to the frequency value in kHz unit (i.e. step = 500 for f = 500 kHz). The lookup table will then increase the address of the memory by the step value at every *sampling_clk* cycle, in order to produce the sine wave output.

The process starting from receiving the update request from the GUI to the generation of new output sine wave is depicted in Figure 8. For the moment, only the sine wave can be generated by the proposed system. For future works, the different type of signal can be selected simply by using the requested signal type flag in the *data_buffer*.

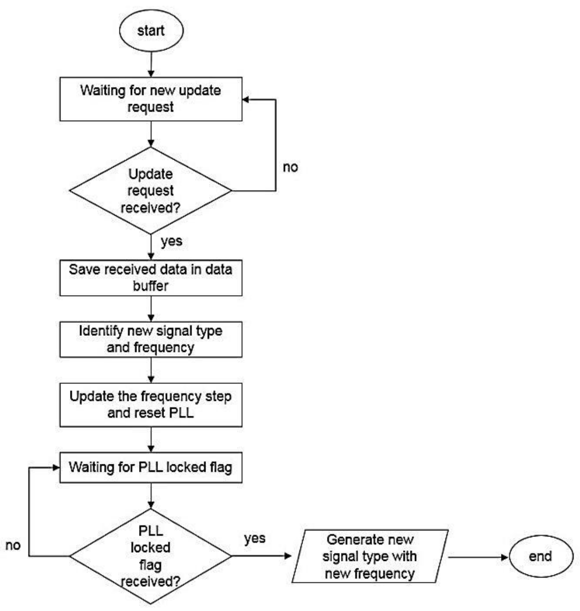

start
Waiting for new update request
Update request received?
no
yes
Save received data in data buffer
Identify new signal type and frequency
Update the frequency step and reset PLL
Waiting for PLL locked flag
PLL locked flag received?
no
yes
Generate new signal type with new frequency
end

Figure 8: Proposed functional generator process flowchart.

VI. Results and Discussions

For this research purpose, a simple and basic GUI on the computer is developed in order to configure the type and the frequency of the signal to be generated by the FPGA board. It contains a drop-down menu which is used to select the type of signal among several options: sine wave, triangular wave, sawtooth wave and pulse wave. Besides, a textbox is also available in the GUI, where users can use to set the desired signal frequency. A clickable button is also added and therefore, the new signal configuration will be sent to the FPGA board via RS-232 communication by clicking it.

The proposed system was designed by using the Verilog HDL code in the Altera Quartus II Design Software. It has been successfully implemented in Altera Cyclone III DE0 FPGA Development Board. To validate the proposed system functionality, functional simulations were conducted by using Mentor Graphic ModelSim-Altera Edition. Then, the experimental tests were performed in FPGA hardware and the results were visualized and observed in Altera SignalTap Logic Analyzer.

A. *Functional Simulations*

In this research, the functional simulations were performed in order to verify the correct functionality of the proposed signal generator. For this research paper purpose, five different frequency values are set during the simulations: 25 kHz, 75 kHz, 150 kHz, 667 kHz and 1 MHz.

Figure 9 presents the simulated sine wave output which has been generated by the proposed design. As can be seen, the signal frequencies observed are the approximately the same as the desired frequency, with very little errors. The frequencies were obtained by measuring the period or the time interval between two cursors in the simulation waveform, as shown in Table 2.

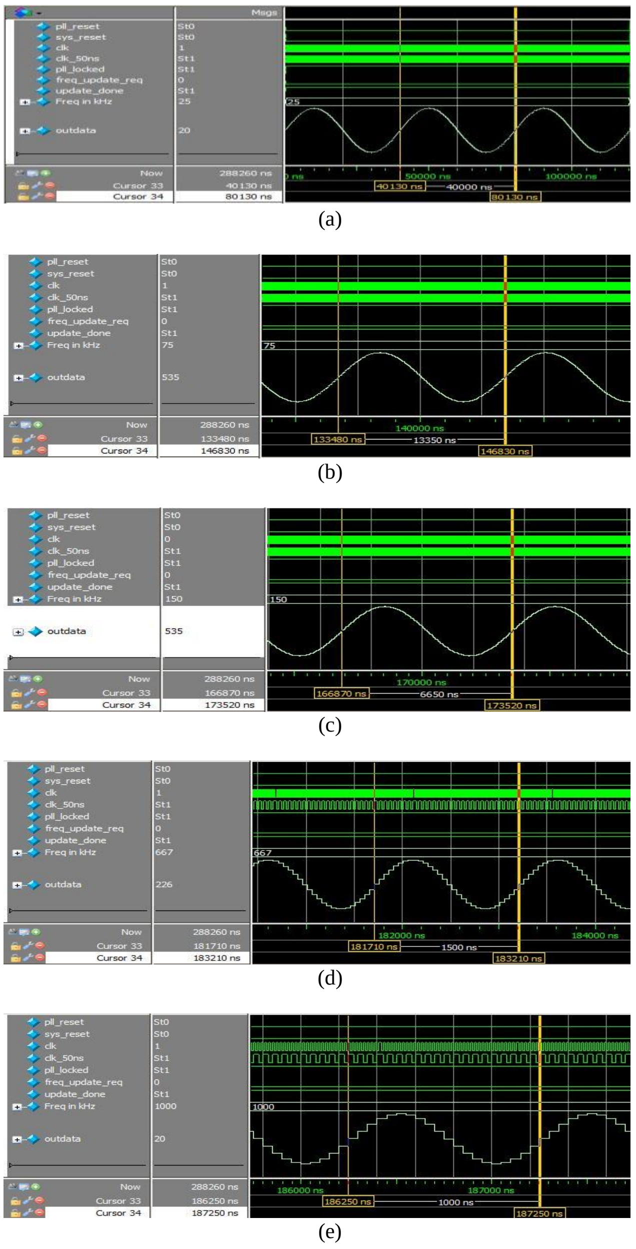


Figure 9: The simulation results for the generated signals with various frequencies: (a) 25 kHz, (b) 75 kHz, (c) 150 kHz, (d) 667 kHz, (e) 1 MHz.

Table 2
Comparison Between Desired Frequency vs. Measured Frequency

Desired Frequency	Measured Period	Measured Frequency
25 kHz	40 000 ns	25.0 kHz
75 kHz	13 350 ns	74.9 kHz
150 kHz	6 650 ns	150.3 kHz
667 kHz	1 500 ns	666.7 kHz
1 MHz	1 000 ns	1.0 MHz

B. *Hardware Experimental Tests*

The proposed system hardware test setup is presented in Figure 10. The DE0 FPGA Board contains the pushbutton which is used as the system reset button. Then, the rxD, txD and gnd pin of the RS-232 interface on FPGA are connected to their respective pins on the RS-232 connector, which is directly connected to the GUI on the PC. The results of the experimental tests were observed in the SignalTap Logic Analyzer windows [13].

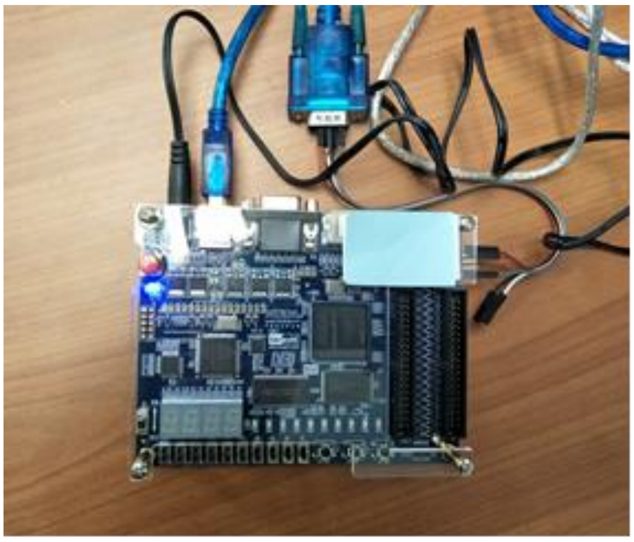
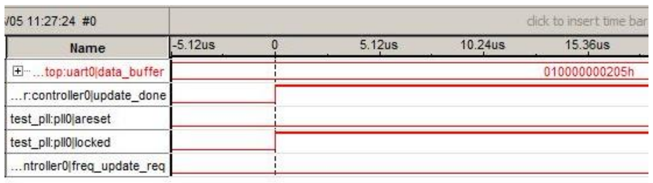


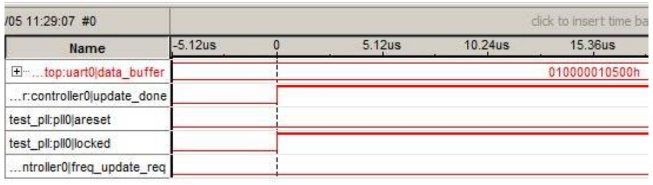
Figure 10: Hardware experimental setup.

Figure 11 shows the updated value of the *data_buffer*, which was observed in the SignalTap, when the signal frequency value is set to 25 kHz, 150 kHz, and 667 kHz, respectively, from the GUI. In Figure 11(a) for example, the value of the *data_buffer* is updated to 010000060607 h, which can be detailed as follows:

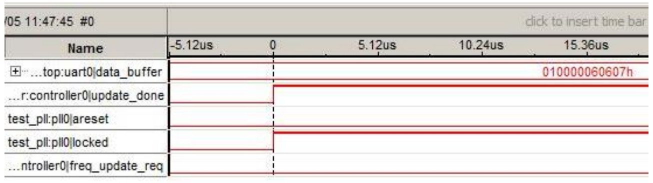
- 01 – to indicate a signal update request from GUI
- 00 – the selected signal type is sine wave
- 00 – the value of thousands is 0
- 06 – the value of hundreds is 6
- 06 – the value of tens is 6
- 07 – the value of ones is 7



(a)



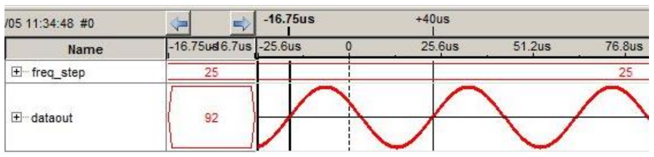
(b)



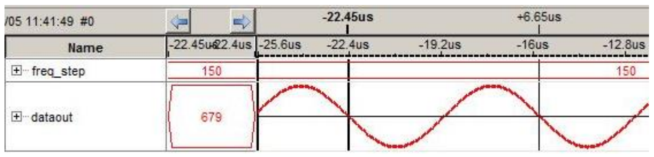
(c)

Figure 11: Observation of the data_buffer value update in hardware tests with different frequencies: (a) 25 kHz, (b) 150 kHz, (c) 667 kHz.

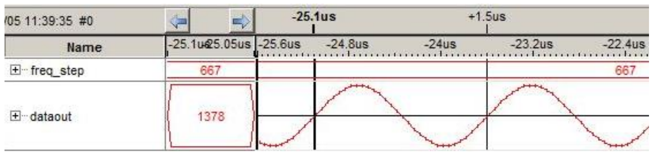
Next, Figure 12 shows the generated sine wave signals which are observed in the SignalTap. From these images, the generated signals' periods observed are 40 µs, 6.65 µs, and 1.5 µs, respectively. Therefore, the frequency obtained from these generated signals are equal to 25 kHz, 150.3 kHz, and 666.7 kHz, respectively.



(a)



(b)



(c)

Figure 12: Observation of the generated sine wave in hardware tests with different frequencies: (a) 25 kHz, (b) 150 kHz, (c) 667 kHz.

C. Design Performance Analysis

Once the design is compiled in Altera Quartus II Design Software, the synthesis and fitter reports are generated, containing the information on the number of logic elements (LE), registers, memory bits and PLL blocks used in the design. Table 3 shows that the proposed design used 240 LEs, 111 registers, 320000 onboard memory bits and 1 PLL block. However, when integrating the SignalTap logic analyzer to the design, those numbers have been increased, especially for the number of onboard memory bits used (from 320000 bits to 426469 bits). This is due to the fact that SignalTap requires some memory allocations in order to store the resulting data which are to be observed.

Table 3
Statistic of FPGA Hardware Resources Utilization for Proposed System Implementation

Design	Hardware Resource Usage			
	Logic Elements	Register	Onboard Memory	PLL Block
Proposed design without SignalTap	240 (1.5%)	111 (0.7 %)	320000 (62.0%)	1 (25.0%)
Proposed design with SignalTap	1387 (9.0%)	1110 (7.0%)	426469 (83.0%)	1 (25.0%)

For further improvements, the system could produce signals with more accurate frequency by increasing the frequency of the sampling clock and also by adding more pre-calculated data inside the lookup table. The latter is only possible to be done by removing the integrated SignalTap logic analyzer since the number of available memory bits is limited. Instead, a digital-to-analog converter (DAC) could be added and thus, the output signal can be visualized by using the oscilloscope. Otherwise, a larger FPGA with more memory spaces can be used, but it will come with greater costs.

Moreover, more features need to be added to this proposed design to make it more useful to users. For example, common signal types like triangle, sawtooth and pulse should be added as an option. Besides, functionalities such as amplitude and phase adjustment will be very useful in many applications. Furthermore, it is also possible to have a multi-channel function generator, where two or more signals can be generated simultaneously. But, this one may depend on the FPGA device capabilities in term of hardware resources.

VII. Conclusion

This paper has discussed on the development of an interactive digital sine wave function generator based on FPGA device. The frequency of the generated sine wave can be adjustable simply by increasing or decreasing the incremental step of the address of a memory, which contains 20000 sine wave sample data. Furthermore, the generation of the signal can be configured from a GUI on a PC which is connected to the FPGA via RS-232 serial communication. The proposed system has been successfully implemented in FPGA and the GUI was developed by using Processing language. The observation of the simulation results and the hardware experimental results shows the proposed system has produced the correct signals as the output, with accurate frequencies.

Acknowledgment

The authors wish to acknowledge Universiti Teknikal Malaysia Melaka (UTeM) for the financial funding under Grant No. PJP/2015/FTK(28D)/S01452 and providing instrumentation devices support for this project.

References

[1] S. Ding, A. An, and X. Gou, "Digital waveform generator based on FPGA," *Res. J. Appl. Sci. Eng. Technol.*, vol. 4, no. 14, pp. 2160–2166, 2012.

[2] A. H. Tirmare, M. S. R. Mohite, V. A. Suryavanshi, T. C. Department, B. Vidyapeeth, and E. Kolhapur, "FPGA Based Function Generator," pp. 2394–2399, 2015.

[3] W. Zheng, R. Liu, M. Zhang, G. Zhuang, and T. Yuan, "Design of FPGA based high-speed data acquisition and real-time data processing system on J-TEXT tokamak," Fusion Eng. Des., vol. 89, no. 5, pp. 698–701, 2014.

[4] G. Brebner and W. Jiang, "High-speed packet processing using reconfigurable computing," *IEEE Micro*, vol. 34, no. 1, pp. 8–18, 2014.

[5] X. Ye, M. Gao, and J. Huang, "12 -Way High Accuracy Sine Signal Generator System Based on FPGA," 2015, pp. 833–836.

[6] S. Yanbin, G. Jian, and C. Ning, "High Precision Digital Frequency Signal Source Based on FPGA," in *Physics Procedia*, 2012, vol. 25, pp. 1342–1347.

[7] M. Herrero, J. J. Rodríguez-Andina, and J. Fariña, "FPGA-based design, implementation, and evaluation of digital sinusoidal generators," in *IECON Proceedings (Industrial Electronics Conference)*, 2008, pp. 2459–2464.

[8] "Waveform Generator Implemented in FPGA with an Embedded Processor by Anna Goman," 2003.

[9] J. W. Hsieh, G. R. Tsai, and M. C. Lin, "Using FPGA to implement a N-channel arbitrary waveform generator with various add-on functions," in *Proceedings - 2003 IEEE International Conference on Field-Programmable Technology, FPT 2003*, 2003, pp. 296–298.

[10] Christopher E. Strangio, "The RS232 Standard," *CAMI Research Inc., Acton, Massachusetts*, 2015. [Online]. Available: http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html.

[11] C. Reas and B. Fry, *Getting Started with Processing*. Sebastopol, CA: O'reilly Media, 2010.

[12] A. Schlegel, "controlP5," 2015. [Online]. Available: http://www.sojamo.de/libraries/controlP5/.

[13] "DE0 Development and Education Board User Manual," *Terasic*, 2009. [Online]. Available: https://www.altera.com/content/dam/altera-www/global/en_US/portal/dsn/42/doc-us-dsnbk-42-5804152209-de0-user-manual.pdf. [Accessed: 23-May-2017].