FPGA-Based Sound Synthesis by Digital Waveguide

Siti Aisyah
Department of Electrical Engineering
Politeknik Negeri Batam
Batam, Indonesia

***Abstract*— This paper address the issues involved in physical modelling synthesis using digital waveguides and its implementation in the hardware. Physical modelling of a one-dimensional musical resonator such as a string can be derived using digital waveguide modelling. The digital waveguide model itself comprises of pair of delay lines. This research models the string digital waveguide using an IIR filter to model the loss parameter and an all-pass filter to handle the fractional number of delays.**

This paper covers the implementation of digital waveguide string model in FPGA board. Applying a scaling and saturation methods, the design approach ensure that the bit overflow would not occur. Applied to the Cyclone II EP2C35F672C6 FPGA on Altera's DE2 board, it uses a total of 1243 registers which are 4% of the total register found on the FPGA.

Keywords—digital waveguide, string model, FPGA

I. INTRODUCTION

Physical modelling of musical instruments has been studied for more than 20 years [1]. Sound synthesis by physical modelling is a technique that produces more realistic sound than other synthesis methods. Some of the physical models are based on finite difference, digital waveguide, mass-spring, modal, wave guide, source-filter models [2]. Among physical modelling approaches, digital waveguide can be implemented at minimal cost because the core model consists only of pair of delay-lines arranged in a circular buffer using pointer arithmetic [1].

Digital waveguide modelling is suited to audio synthesis especially for one-dimensional music resonators such as strings and tubes [3]. A digital waveguide string model comprises of a pair of delay lines [1]. For handling fractional delay, Smith [1] proposed the all-pass filter. To model the propagation losses of delay line, Valimaki [2] proposed a single loop filer. FIR filters can also be applied to delay lines to model the losses parameters for efficiency design [4].

Other music resonators such as membranes can be modelled by using digital waveguides. Membrane model can be obtained by parallelizing the digital waveguide string model [4,5].

An audio signal processing needs a high performance processor to be implemented in real-time. Many studies has been publicized the capability of a Field-Programmable Gate Array (FPGA) as a high speed processor. It has been known that the FPGA is the most capable device to fulfil highly task in many fields of signal processing [6]. FPGA are also used in real-time sound synthesis [6, 7, 8].

This research covers the digital waveguide string model and its implementation on FPGA board which can be explored to be a membrane model in future study.

II. RELATED WORK

A. String Model

There are many physical modelling approaches that can be used for simulating musical instruments [2]. The digital waveguide model proposed by J.O. Smith [1] is one of those approaches that has often used in sound synthesis applications. This kind of physical modelling can be applied to acoustic instruments such strings and membranes. The waveguide model of an ideal string is represented in Fig. 1.

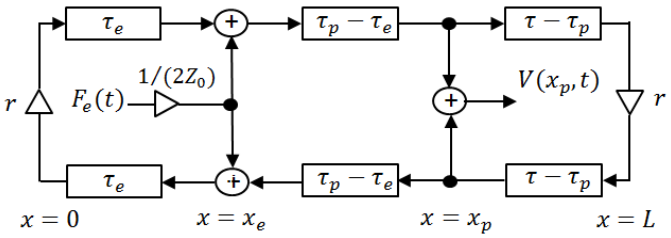

Fig. 1. Waveguide model of an ideal string. Physical model of a string driven by external force (F_e) at position $x = x_e$, while the velocity (V) observed at x_p. The reflection coefficients ($r = -1$) at both ends represent that the wave travel along the model will be totally reflected at the boundaries.

The velocity waves can be derived by summing the velocities at the pick-up point due to the right and left-propagating waves in agreement with d'Alembert solution [5]. Giving a force in the string, the velocity waves can be observed in the pick-up point. The relation between the force in the string and the velocity waves is defined as

$$f^+(x,t) = Z_0 \frac{dw^+}{dt}(x,t) \quad (1)$$

where the characteristic impedance, Z_0 , defined by the formula $Z_0 = \rho Ac$.

978-1-4673-6601-4/15/$31.00 ©2015 IEEE

The reflection coefficient (r) defines the string model in term of boundary whether it is fixed-boundary or free-boundary. We can consider that the reflection coefficient is minus one (as it is suited to the early study of the project) which means a fixed-boundary string. The velocity can be observed at the pick-up point $x = x_p$ by driving force (F_e) at position $x = x_e$. The delay parameters (τ, τ_e, τ_p) represent the wave travel along the waveguide with respect to delay time in certain position. Those delay parameters depend on the wave speed (c) and its position which can be expressed as $\tau = L/c, \tau_e = x_e/c, \tau_p = x_p/c$.

In digital domain, the waveguide modelling of an ideal string can be modelled as shown in Fig. 2. The model consists of pair of delay lines. In this digital model, the injection and observation point can be simply changed by calculating the delay address pointers arithmetically.

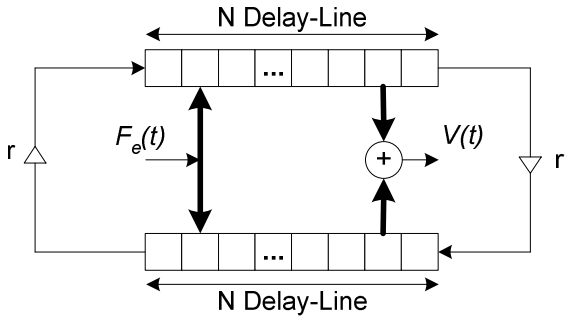

Fig. 2. Digital waveguide string model is realised by employing a pair of delay-lines.

B. Digital Waveguide String Model

The all-pass filter can be implemented to the string digital waveguide by combining the fractional delays within the model. To model the propagation losses within the model, the IIR loss filter can be employed as a single loop filter. Applying the all-pass filter and the IIR loss filter, the digital waveguide string model can be illustrated as shown in Fig. 3.

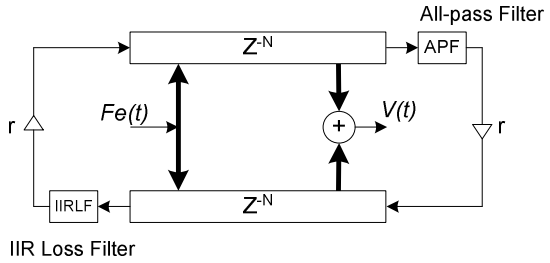

Fig. 3. Digital waveguide string model consisting of pair of delay lines, all-pass filter and IIR loss filter.

In this string model, the integer number of delays which depends on the length of string L, the propagation distance over time, X, and the sample period T can be calculated using (2). The fractional delay can also be calculated using the formula in (3). As string model has 2 delay lines, the fractional number of delay should be multiplied by 2 and it is represented in (4). The fractional number of delays D then can be applied to the all-pass filter coefficient formula.

$$N_d = \lfloor (L/X) \rfloor \tag{2}$$

$$d = L/X - (N_d);\ 0 < d < 1 \tag{3}$$

$$D = 2d;\ 0 < D < 2 \tag{4}$$

There is a possibility that the injection and extraction point fall in between the sampling point. To deal with this, Interpolation (I) and de-interpolation (DI) can be applied as an approximation method [5]. Fig. 4 illustrated I/DI scheme of a digital waveguide string model.

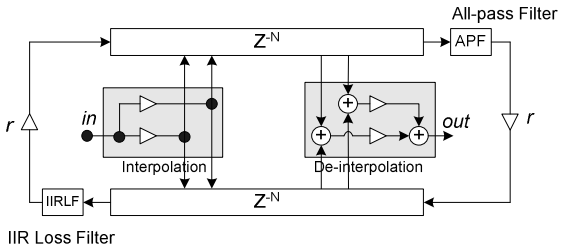

Fig. 4. Digital waveguide string model with two multiplication interpolation and de-interpolation.

Injection and extraction points can be changed easily using pointer arithmetic. Applying I/DI methods on model will automatically arrange the number of input and output. In this model, there will be 4 points input and 4 points output along delay lines.

Fig. 5 showed that there are two delay lines which have an opposite direction, to the right and to the left, within a string model. The delay width from the writing point w_p to the first excitation point x_e is denoted with N, while M is a delay width from w_p to the first pick-up point x_p. Adding 1 to the address of the first excitation and pick up point will obtain the second excitation and pick-up point address respectively.

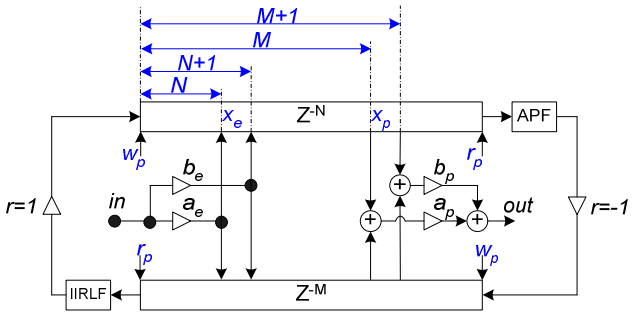

Fig. 5. The delay width of excitation point x_e and extraction point x_p, N and M respectively, within the delay lines in digital waveguide string model.

Defined the relative point of string length, the excitation and pick-up point can be arranged flexibly along the delay line. Assumed that the relative excitation point is r_e, the excitation point can be defined using the formula $x_e = r_e L$.The same holds to the pick-up point, $x_p = r_p L$ then can be used for the relative pick-up point r_p.

The integer number of delay width of the excitation point (N) can be defined as $N = \lfloor (x_e/X) \rfloor$ and $M = \lfloor (x_p/X) \rfloor$ is the

integer number of delay width of the pick-up point, where $X = cT$, in which c is wave velocity and T denotes time of a single delay. The coefficient of excitation and pick-up point in digital waveguide string model is defined as (5) and (6) respectively.

$$b_e = (x_e/X) - N;\ a_e = 1 - b_e \quad (5)$$

$$b_p = (x_p/X) - M;\ a_p = 1 - b_p \quad (6)$$

III. Design Approach

The 16 bits input in_string, the filter coefficients, aAP, gLP and aLP, and the fractional delays coefficients (a_e, b_e, a_p, and b_p) are injected to the entity to yield an output out_string. The output has been set up on 54 bits to reduce the possibility of data losses but it is still can be arranged flexibly to another number of output using the scaling method. The entity contained of delay blocks, all-pass filter and IIR loss filter is shown in Fig. 6.

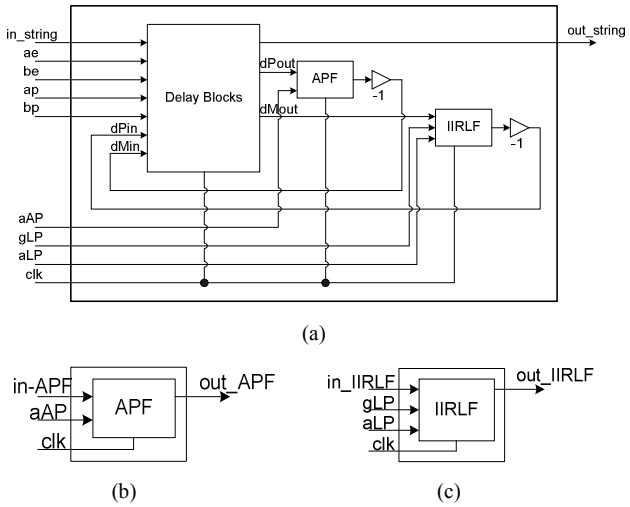

Fig. 6. (a). String digital wavQ1111111eguide design entity; (b). sub entity / sub program all pass filter; (c). sub entity / sub program IIR loss filter.

The data flow is described in the following condition:

- The 16 bits data input in_string and the coefficients of digital waveguide a_e, b_e, a_p, and b_p are processed within a certain number of delay blocks. The process yields the 32 bits output cal led dPout.
- The dPout will become an input for the all-pass filter. The All-pass filter which controlled by the coefficient aAP will yield an output denotes as out_APF in the subprogram figure 28.b. Being multiplied by "-1", the out_APF will become an input, dMin, of delay blocks.
- The 32 bit dMout will be obtained after the dMin is being processed within a certain number of delay lines. This process also involved the data input in_string and the coefficient a_e, b_e, a_p, and b_p.

The data from dMout will be fed into the IIR loss filter. Being controlled by the coefficient gLP and aLP, the IIR loss filter will yield an output out_IIRLF which depicted in sub-entity/sub-program Fig. 6.c. Multiplied by "-1", the out_IIRLF will become an input, dPin, of the delay blocks described in point 1. The whole process keeps continuing in looping condition until being terminated. The detail process has depicted in flowchart in Fig. 7.

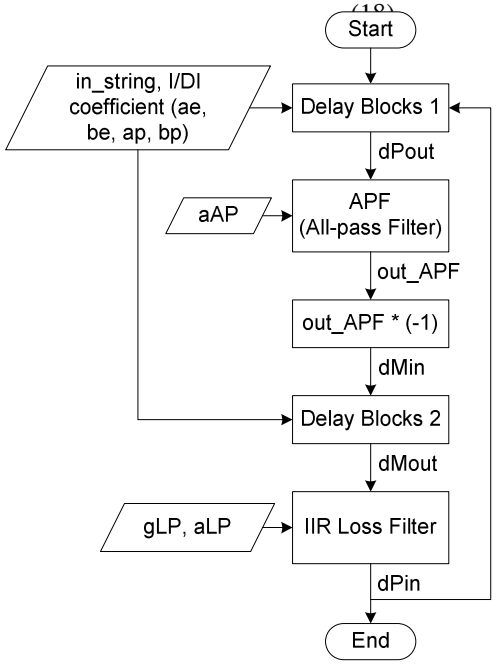

Fig. 7. Data flow of string digital waveguide model in recursive structure.

The all-pass filter and the IIR loss filter itself are also being processed in feedback circuit. This characteristic may deal with the overflow condition which the bit data is too big to be represented by the data width.

IV. Simulation Result

A. Scaling and Saturation Methods

A series of simulations in Modelsim has been carried out to implement the string model in hardware. The first simulation is started with the first order all-pass filter design. The design of first order all-pass filter consists of 2 delay blocks, 2 adder/subtraction and 2 multipliers. It is also illustrated in the final hardware design shown in Fig. 8. The all-pass filter has a feedback circuit. Applying 32 bits input/output and 16 bits coefficient aAP, the design deal with an overflow issue within the feedback circuit. In this design, it has applied the scaling and saturation methods to process bit overflow.

The scaling method means that a certain signal has a certain bit width. For example, by defining max_val(31 downto 0), the max_value will be ranged as 32 bits.

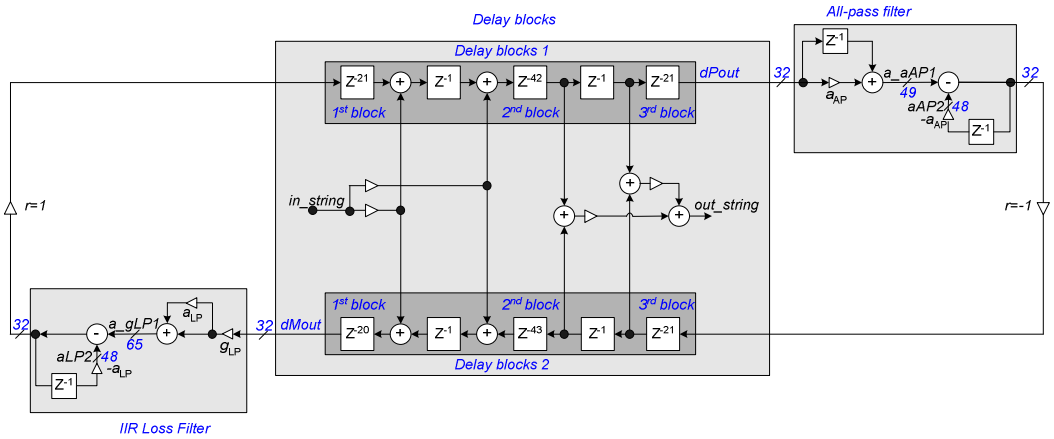

Fig. 8. The implementation of digital waveguide string model to the hardware design which consists of three main entity, namely all-pass filter, IIR loss filter and delay blocks.

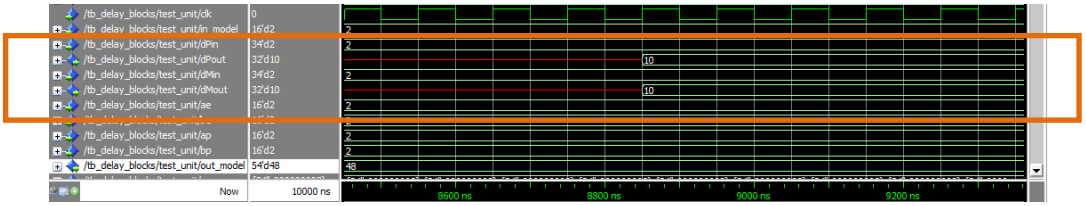

Fig. 9. The delay blocks ModelSim Trace, the output $dPout$ and $dMout$ come out in 88 clock cycles.

The scaling and saturation process has been implemented to the a_aAP1 and the aAP2 which have been scaled to 31 bits and saturated to the maximum value of 1073741823 and minimum value of -1073741824 since the test using the data type signed. The out_APF has also been scaled to 32 bits and saturated to the maximum and minimum value as aAP2.

The same condition holds to the IIR loss filter. The a_gLP and the $aLP2$ have been scaled to 31 bits and the out_IIRLF1 has been scaled to 32 bits.

The whole model design has applied a looping structure, hence the bit overflow would become an issue to the model. The same methods, scaling and saturation, have also been implemented to the model design to handle bit overflow. No matter how big the data width in the output of delay blocks, $dPout$ and $dMout$, the value has to be scaled to 32 bits and applied to the saturated value. This condition would ensure that data width between the output of delay block and the input of each filter would match.

In the delay blocks design, the 88 delays has been applied for each delay block 1 and delay block 2. Because the delay blocks used any adder and the adder itself take 1 delay to complete its operation within the clocked process, the number of delays applied should be calculated correctly. In the design depicted in Fig. 8, block delays 1 has been determined to have a sequential 21 delays in the first block, 42 delays in the second block and 21 delays in the last block. To have a match injection point and extraction point between the two delay blocks, the block delays 2 should have a sequential of 20, 43 and 21 delays for the first, second and last blocks respectively.

Simulating the delay blocks in ModelSim, the output $dPout$ and $dMout$ have been generated in 88 clock cycles which is shown in Fig. 9.

In order to injecting in the right point within the delay lines, the delay block between the two adder components should be consider to be deleted. It is because the adder component itself needs 1 delay to derive an output. This consideration leads the another design model for the delay blocks as depicted in Fig. 10.

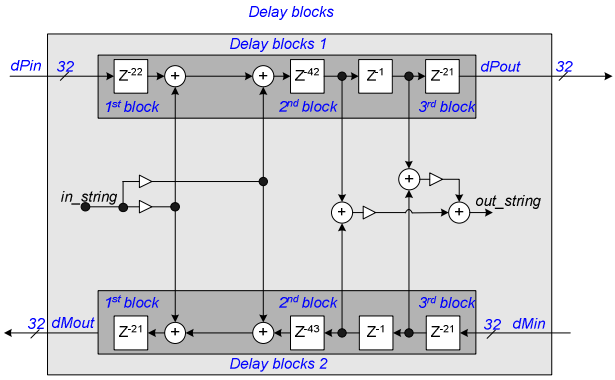

Fig. 10. Another design model of the delay blocks within the delay lines by deleting a delay block between the two adders in each delay lines for pointing the right injection points.

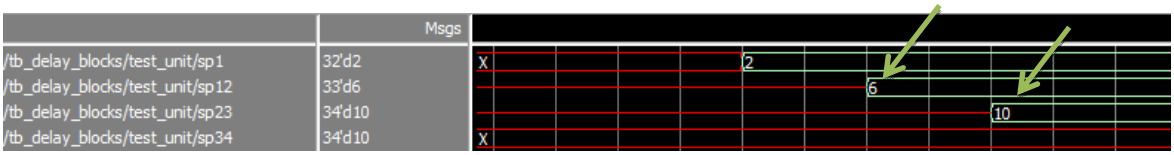

Fig. 11. The timing trace in ModelSim to show that the two injection points differ by one delay.

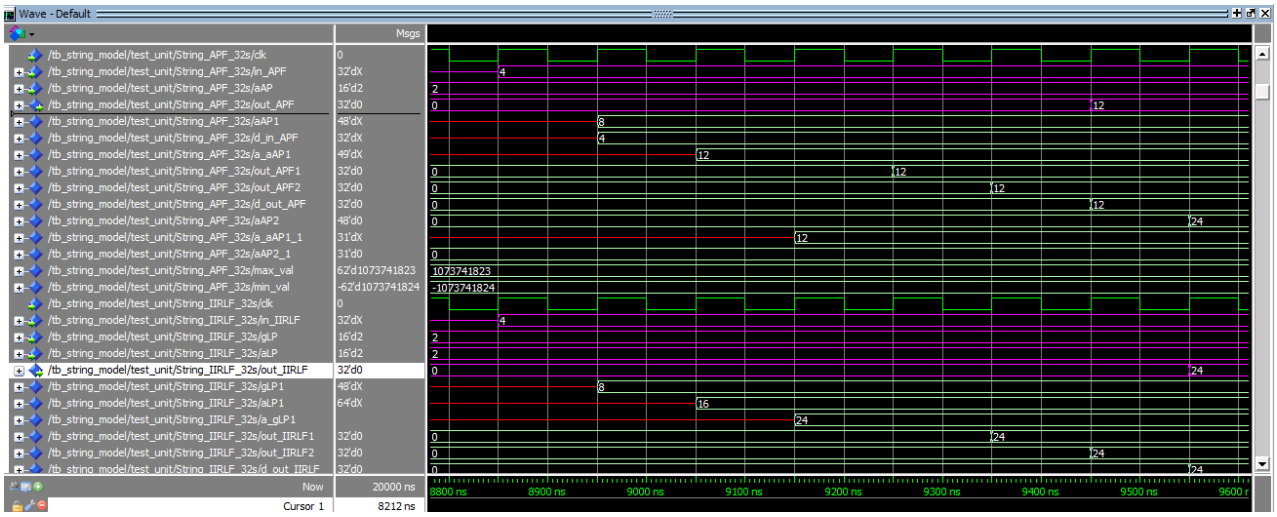

Fig. 12. The output out_APF (magenta-line) of 12 (in decimal) was generated by the all-pass filter and the IIR loss filter yielded the output out_IIRLF (magenta-line) of 24 (in decimal), where the logic input string is 1 (in decimal) and all the coefficients of 2 (in decimal).

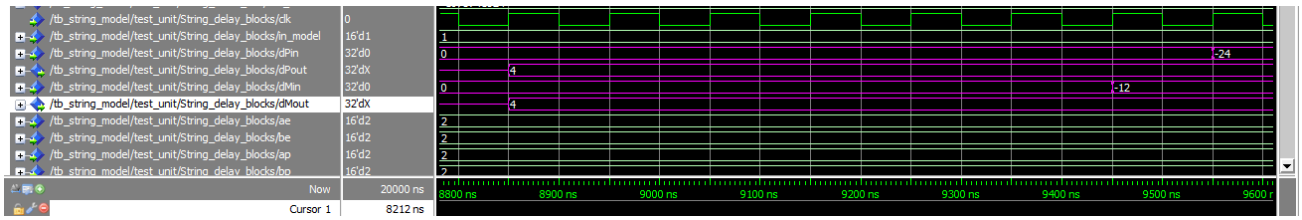

Fig. 13. $dMin$ (magenta-line) is equal to -12 (in decimal) after the output of the all-pass filter, out_APF, was multiplied by '-1', while $dPin$ (magenta-line) is the IIR loss filter output, out_IIRLF, multiplied by '-1' that equals to -24 (in decimal).

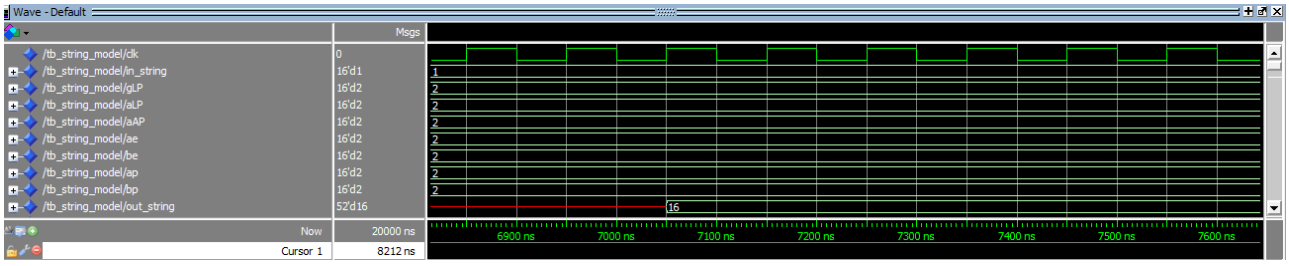

Fig. 14. The first output signal of 16 (in decimal) in 70 clock cycles.

The simulation in ModelSim, illustrated in Fig. 11, has been shown that the delay was one between the two injection points.

B. String Model Response

The string model has been simulated in ModelSim by feeding in the logic to the input in_APF of "0000000000000001" and to the all coefficients, a_{AP}, g_{LP}, a_{LP}, a_{e}, b_{e}, a_{p}, and b_{p} of "0000000000000010". The first signal generated as the input of each filter is "0000000000000100" or 4 in decimal as shown in Fig. 12. The all-pass filter outputs the signal of 12 (in decimal) and the signal of 24 (in decimal) was generated by the IIR loss filter.

As mention earlier, being multiplied by "-1", the output of the all-pass filter, out_APF, will become an input, $dMin$, of

delay blocks. It is clearly shown in Fig. 13 that $dMin$ is -12 (in decimal). The same holds to the output of the IIR loss filter, out_IIRLF, which will become the input of delay blocks, $dPin$, that have a value of -24 (in decimal) after being multiplied by '-1'.

Setting up the input signal as mentioned earlier, the simulation results a stream of data. The first output signal which generated by the string model is 16 (in decimal) of 70 clocks as shown in Fig. 14.

The resources used for synthesising the string model which is set to target the Cyclone II EP2C35F672C6 FPGA on Altera's DE2 board has shown in Fig. 15. It uses a total of 1508 logic elements which is less than 5% of the logic elements found on the FPGA mentioned above. Among the logic elements, a total of 1243 registers were used and 181 pins were being assigned as I/O ports. The utilisation of the pins is at 38% of the total pins available on the FPGA. This string model system is able to operate at a maximum clock rate of 126.63MHz (7.89 ns), which has shown in Fig. 16.

Flow Summary	
Flow Status	Successful - Tue Sep 17 09:15:55 2013
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	String_Model
Top-level Entity Name	String_Model
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	1,508 / 33,216 (5 %)
Total combinational functions	1,231 / 33,216 (4 %)
Dedicated logic registers	1,243 / 33,216 (4 %)
Total registers	1243
Total pins	181 / 475 (38 %)
Total virtual pins	0
Total memory bits	5,052 / 483,840 (1 %)
Embedded Multiplier 9-bit elements	34 / 70 (49 %)
Total PLLs	0 / 4 (0 %)

Fig. 15. Area utilisation summary of the string model system

Slow Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	126.63 MHz	126.63 MHz	clk	

Fig. 16. Speed analysis of the string model system

V. CONCLUSION

Implementation of a recursive structure on hardware will generate bit overflow. The string model that implements a closed loop structure and feedback circuit within the filters design needs to deal with this overflow issue. To ensure that bit overflows do not occur, the scaling and saturation method seems to be an appropriate method to be applied to the model.

The hardware approach in this research shows that the digital waveguide string model required many registers. In hardware perspective, it will consume memory and time.

ACKNOWLEDGEMENT

This study was accomplished in the Queens University of Belfast under supervision of Dr. Maarten van Walstijn.

REFERENCES

[1] J.O. Smith, "Physical audio signal processing", Standford University, (http://ccrma.stanford.edu/jos/pasp/, online book).

[2] Vesa Valimaki, Jyra Pakarinen, Cumhur Erkut and Matti Karjalainen, "Discrete-time modelling of musical instruments", Reports on Progress in Physics, volume 69, number 1,2006, pP 1-78.

[3] S. Bilbao, "Fast modal synthesis by digital waveguide extraction", IEEE Signal Processing Letters 13, pp.1-4, 2006.

[4] M. van Walstijn and Eoin Mullan, "Time-domain simulation of rectangular membrane vibrations with 1-D digital waveguide", Aalborg, Forum Acusticum 2011.

[5] Eoin Mullan, "Physical modelling sound synthesis by digital waveguide extraction application to computer games and virtual environments, PhD Thesis, Queen's University Belfast, 2011.

[6] F. Pfeifle and R. Bader, "Real-time finite difference physical models of musical instruments on a field programmable gate array (FPGA)", Proc. of the 15th Int. Conference on Digital Audio Effects (DAFx-12), York, UK, 2012.

[7] K. Chuchacz, S. O'Modhrain and R. Woods, "Physical models and musical controllers – designing a novel electronic percussion instrument", NIME, 2007.

[8] E. Motuk, R. Woods, and S. Bilbao, "FPGA-based hardware for physical modelling sound synthesis by finite difference schemes", IEEE ICFPT, 2005.