FPGA Implementation of Musical Notes Recognition for Automatic Transcription System

S. C. Sugiarto and C. Ortega-Sanchez
Department of Electrical and Computer Engineering
Curtin University of Technology
Kent St, Bentley 6102, Western Australia
s.sugiarto@student.curtin.edu.au
c.ortega@curtin.edu.au

ABSTRACT

Musical transcription from musical data into score is commonly performed by software running on a PC. This involves significant delay, which can be considerably reduced by hardware implementation. This paper presents a hardware design and implementation of the frequency identification system, one of the most important function in musical transcription, accomplished through the use of Fast Fourier Transform (FFT) algorithm implemented in a Field Programmable Gate Array (FPGA).

1. INTRODUCTION

Frequency identification is at the heart of automatic transcription system. Real-time automatic transcription system can be used in many applications such as preserving impulses of music talents by translating it to score or as a means of testing musical ability by comparing played and original musical score. Frequency identification system can be implemented in real-time using the widely known FFT algorithm performed on an FPGA.

1.1. MUSIC

Two aspects of music are integral in the development of the musical transcription and frequency identification: notes and score. The knowledge of musical notes is necessary to determine the sampling frequency of the FFT inputs, which is essential in interpreting the outputs of the FFT. The understanding on music representation in the form of musical score is crucial to be able to display the output of musical transcription system.

1.1.1. MUSIC NOTES

A musical note is used to represent the duration and pitch of a particular sound. The chromatic scale (shown in Figure 1), consisting of thirteen notes (including both ends), is the main scale in music, with all other scales being a subset of this particular scale[1]. In the chromatic scale the frequencies of any two notes are separated by the interval of semitone, which is $2^{1/12}$ of the frequency of the lower note[1]. The first and last notes of this scale are C's with the second C twice the frequency of the first one as shown in Figure 1 and 2. This separation is called an octave.

The circled notes in Figure 1 form the diatonic scale in music. These notes are represented by white keys on a piano, as shown in Figure 2, and can be combined to create simple music as shown in Figure 3.

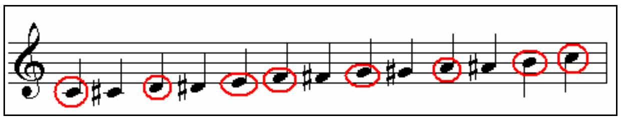
Figure 1: Chromatic scale of 13 notes[1]

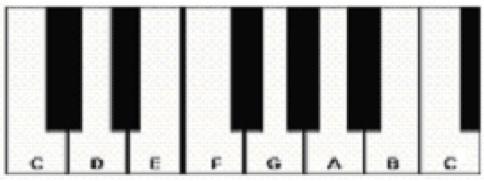

Figure 2: Middle octave of a piano

For the practicality of testing and implementation, diatonic scale is used in the investigation. The frequencies of the notes in the middle octave with the central A of 440 Hz are depicted in Table 1.

Table 1: Frequencies of notes in middle octave

Notes	Frequencies (in Hz)
C	261.63
D	293.66
E	329.63
F	349.23
G	392
A	440
B	493.88
C	523.25

1.1.2. MUSIC SCORE

Music has been, actively and passively, part of human lives. Passive music, where the user utilise the music by listening to it, is usually in the form of recorded sounds. Active music, on the other hand, needs to have a more explicit representation than recording. The most common representation of active music is in the form of musical score, an example of which is shown in Figure 3.

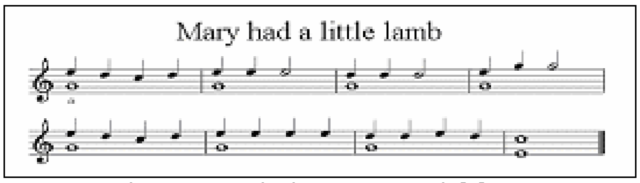

Figure 3: Musical score example[2]

Musical score elucidates aspects of a piece of music that are not obvious from mere listening. Musical score is used so people, other than the original author or original performer, can perform a particular piece of music and reproduce a musical performance. It is used to preserve and communicate a musical piece to oneself and others.

1.2. FAST FOURIER TRANSFORM

The Fourier Transform is used to translate a time-domain signal into a frequency-domain signal. The Fast Fourier Transform algorithm is widely used in computer implementations as it reduces the execution time compared to the conventional Fourier Transform. The FFT algorithm is simple and efficient as it only involves multiple additions and constant multiplications. The effectiveness of FFT algorithm can be further exploited by utilizing its parallel nature.

The FFT is an algorithm to compute complex Discrete Fourier Transform, for complex signals that are discrete and periodic. In the context of musical note identification, the input signals are real, discrete and periodic signals. Due to this reason, only the real DFT needs to be computed by the FFT and it is necessary to translate the real DFT data into the complex DFT format. Complex DFT introduces the terms real part and imaginary part to the algorithm. For the purpose of real DFT, these terms simply means the amplitude of cosine wave amplitude and sine wave amplitude respectively[3]. An example of the cosine waves is shown in Figure 4. From Figure 4, k is defined as the number of full cycle a signal makes in the duration of 2π and n is the sample number that goes from 0 to 15. Sample 16 is not taken into consideration as it is a repeat of sample 0.

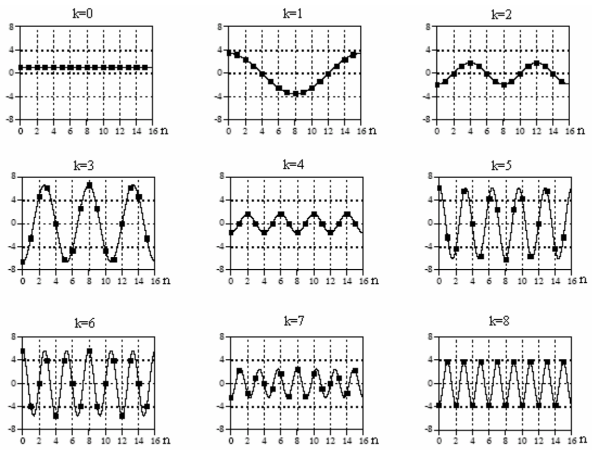

Figure 4: Cosine waves of 16-point FFT decomposition[3]

The algorithm used in this investigation is the decimation-in-time, radix-2, 16-point FFT algorithm. Bit reversal is performed on the input data before the computation begins. The computation consisted of four stages with each stage computed in one clock cycle. Eight constants are necessary to perform the 16-point FFT algorithm. These constants are represented by ω_N where N is the number of points in the FFT. For N equals to 16, these constants are derived from equation (1).

$$\omega_{16}{}^{nk} = \exp(-j\tfrac{2\pi}{16}nk) = \cos(\tfrac{2\pi}{16}nk) - j\sin(\tfrac{2\pi}{16}nk) \tag{1}$$

where nk = (0, 1, 2, 3, 4, 5, 6, 7). Only these values of nk are considered as this is a periodic variable.

The output of the FFT consists of two parts that are the mirror image of one another. An example of this, with two spikes at frequency $\dfrac{3\omega_s}{16}$ and $\dfrac{13\omega_s}{16}$, is shown in Figure 5.

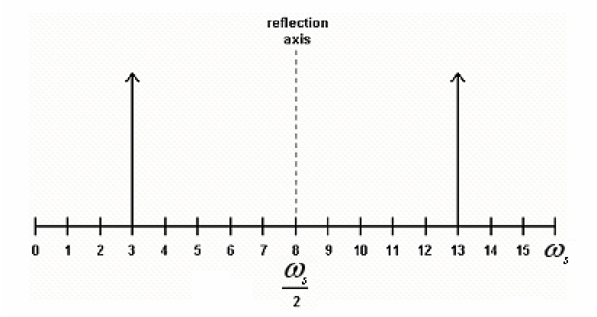

Figure 5: An example of output of an FFT

ω_s represents the sampling frequency of the FFT and $\dfrac{\omega_s}{2}$ is half of the sampling frequency. A signal must be sampled two times as fast as its frequency to be detected by the FFT. For a 16-point FFT with particular sampling frequency ω_s, only eight discrete signals with frequency between 0 and $\dfrac{\omega_s}{2}$ can be distinguished. Any two neighbouring outputs of the FFT have equal separation in frequency with the first having 0 Hz frequency (DC) and the last having $\dfrac{\omega_s}{2}$ Hz frequency. However, for the purpose of the frequency identifier in the automatic musical transcription, another interpretation of the output was utilised. In this interpretation, notes with frequency between $\dfrac{\omega_s}{2}$ and ω_s are distinguished by sampling the input signals at ω_s. This is possible as long as it can be ensured that inputs are within a fixed range of frequency of $\dfrac{\omega_s}{2}$ to ω_s .

1.3. FPGA AND VHDL

FPGA implementation was chosen because it has the capability to compute highly parallelized algorithms and exploit the high speed and efficiency of hardware implementation. This is especially useful in implementing Digital Signal Processing (DSP) algorithms such as the FFT. Very High Speed Integrated Circuit (VHSIC) Hardware Description

Language (VHDL) is one of the standard languages that can be used to implement designs in FPGAs. As a Hardware Description Language (HDL), it possesses characteristics that can handle the complexity of hardware circuitry that is needed in design implementation. Beside that, it has the re-programmable capability which is convenient in the prototyping phase of a design.

1.4. XILINX ISE DEVELOPMENT SOFTWARE

In this investigation, the development software platform ISE Foundation from Xilinx is used to simulate, verify and download the design into and FPGA board. ISE is a very powerful tool that does synthesis and implementation of the design, including place and route[4].

2. AUTOMATIC MUSIC TRANSCRIPTION

Musical transcription from audio signals is the process of taking a samples from a digitised sound waveform and extracting from it the symbolic information related to the high-level musical structure that are seen on a score[5]. The two most important features that need to be extracted from the audio signal are the pitch, corresponding to the frequency of the sound and the duration of the sound. Figure 6 shows the proposed automatic transcription system. The input to the system is music in its audio format. This input will be amplified and fed into an analogue to digital converter (ADC). This ADC determines the sampling frequency of the input to the frequency identifier system, which is crucial in the interpretation of the output.

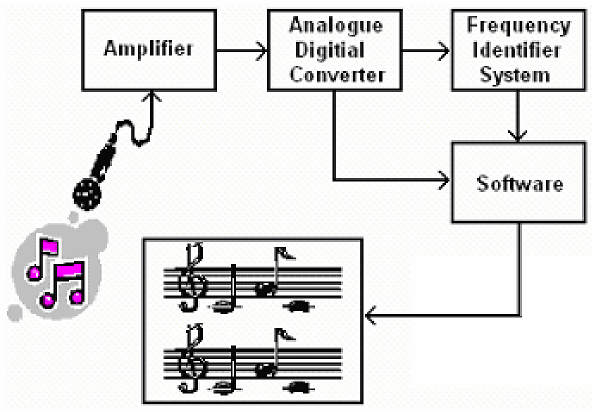

Figure 6: Automatic Transcription System

The frequency identifier system or musical notes recognition system implemented in FPGA provides the frequency of each individual note as well as other information needed for a musical score to be produced such as the duration of the note. The duration of a note can be found by detecting the silence between notes. To detect notes from different octaves multiple FFT should be utilised, each one running at different sampling rates. In this way the level of complexity of the system can be kept to a minimal level. The duration of the fastest note in music is significantly longer than the execution time of the FFT on the FPGA. Due to this, it is possible for a note to be sampled at various sampling frequencies until the required frequency range is found. The sampling frequency has to be sent to the software together with the particular frequency found by the identifier system. The software in this system executed on a PC will then convert all the required information into a musical score. For future optimization, by performing simple modification of the model, it should be possible to fully implement this automatic transcription system in hardware, eliminating the PC. However, further investigation needs to be carried out to verify this assumption.

3. FREQUENCY IDENTIFIER SYSTEM

The frequency identifier system is an important function block automatic transcription system. The block diagram of this system is shown in Figure 7. For verification and testing purpose, the ADC in Figure 6, which provides inputs to the frequency identifier system, was replaced by file stored in PC containing all input values representing a particular note. In other words, instead of the digitized input from the ADC, a hexadecimal text file is used as an input and communication between the FPGA and PC is done through the HyperTerminal.

This system consists of the clock divider, UART receiver and shift registers to prepare the inputs of the FFT. The FFT is the main algorithm used to detect frequency of a note. The output block modifies the outputs of the FFT into the required format of the rest of the automatic transcription system. However, for testing purposes, the output block receives the FFT outputs and generates signals to control eight LEDs, combination of which represents each note in the diatonic scale. To achieve this purpose, the output block in Figure 7 converts the FFT outputs into binary values of either 0 or 1 to control the LEDs.

3.1. MUSICAL NOTES AS INPUTS

Input selection depends solely on the sampling frequency that is used to obtain the input data. For the FFT to generate perfect spikes with only two non-zero outputs, inputs need to follow equation (2). Two spikes, instead of one, are generated due to the mirror image property of FFT outputs as shown in Figure 5.

$$input = \sin\frac{2\pi nk}{N} \qquad (2)$$

where n is the sample number, k is the number of full cycle a particular signal can do in 2 π and N is the total number of samples in the FFT. However, musical notes are not characterised by k and N. A note can only be distinguished from another note by its frequency. The relationship between frequency, k and N is stated in equation (3).

$$f = \frac{k}{N} f_s \qquad (3)$$

By combining equations (2) and (3), the inputs to the frequency identifier system are obtained by equation (4).

$$input = \sin\frac{2\pi nf}{f_s} \qquad (4)$$

These inputs are scaled up by a factor of 127 to magnify the spikes given out by the FFT. The sampling frequency used in this investigation was 523.26 Hz. 16 inputs are generated for each of the eight notes specified in Table 1.

3.2. CLOCK DIVIDER AND UART RECEIVER

The UART receiver is used to convert serial input to 8-bit parallel outputs. The clock divider is used to generate 16 times the baud rate of the UART receiver from the 100 MHz oscillator on the Virtex 4 board. Depending on the required baud rate, the clock divider can be controlled to produce clock with the desired frequency. The baud rate used in the testing of the system is 19200. To achieve the requirement of 16 times the baud rate, a clock of period 3.26 μs was generated by the clock divider. This is done through the use of a counter to count 324 times the main oscillator of the board.

3.3. SHIFT REGISTERS

The shift registers are controlled by the data ready signal from the UART receiver. On the rising edge of this signal, a new parallel data from the UART is latched into the first register and the current data in the rest of the registers are latched ready to be shifted into the next register on the falling edge of the data ready signal. The main purpose of this block is to prepare the inputs to the FFT. The output of the UART is one 8-bit signed integer whereas to be meaningful, the FFT needs the full set of 16 8-bit signed integers as inputs. The shift registers act as buffers to store the output of the UART so 16 numbers of integers can be entered in parallel to the FFT. The implication of this design is that the first 15 outputs sets of the FFT have to be discarded and only the 16th output set onward carries meaningful output from the system.

3.4. FFT

The FFT used in the investigation takes its inputs from the shift register on the rising edge of the 100 MHz clock of the FPGA board. The FFT computation itself requires 4 clock cycles to be executed. One additional clock cycle is needed to compute the magnitude square of the outputs as they have real and imaginary part. Magnitude square instead of magnate is utilised to simplify the computation. The bit reversal stage is done on the top level VHDL code which maps the output of the shift registers in bit reversal order to the input of the FFT. The inputs from the shift registers are taken as the real part of the input and the imaginary parts are all set to zero. This is one of the requirements of the FFT as shown in Figure 7. As the actual input from the music only represent the real part, the imaginary parts are manually assigned to zero values.

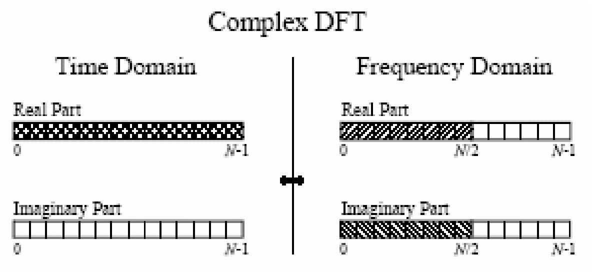

Figure 8: Input – output format of complex DFT[3]

As shown in Figure 8, the complex DFT transforms two N points time domain signals into tow N points frequency domain signals, which are called the real part and the imaginary part. This is in contrast to the real DFT that transforms an N points time domain signal into two $\frac{N}{2}$ + 1 points frequency domain signals.

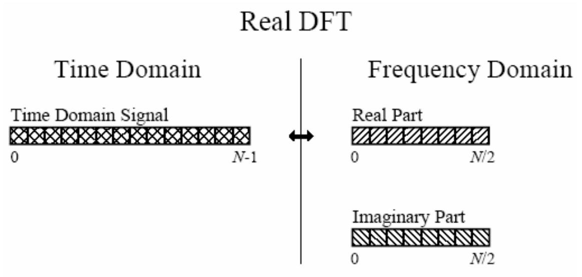

Figure 9: Input – output format of real DFT[3]

To transform time-domain music signals into frequency domain using the FFT algorithm, it is necessary to translate the complex DFT into real DFT format. To do so, it is necessary to move the N points time domain real signal to the real part of the time domain the complex DFT and set all of the samples in the imaginary part to zero[3].

The 16-point, decimation in time FFT takes four stages to generate output values. Each stage is completed in one clock cycle. As the input-output mechanism is much slower (controlled by the data ready signal from the UART) than the FFT computation (controlled by the board 100 MHz clock), it can be ensured that by the time the outputs are collected, the FFT computation has reached completion and correct outputs are propagated to the rest of the system. The FFT is controlled by the falling edge of the clock.

3.5. OUTPUTS INTERPRETATION AND LEDS

The interpretation of the outputs is one of the most important parts of the design and analysis process. The FFT produces an output as depicted in Figure 5. With a sampling frequency of 523.25 Hz, only inputs with the frequency between 0 and 261.625 Hz could be distinguished. However, looking again at Figure 4, it can be noted that the outputs are two parts mirrored at 261.625 Hz and, due to this reason, the frequencies

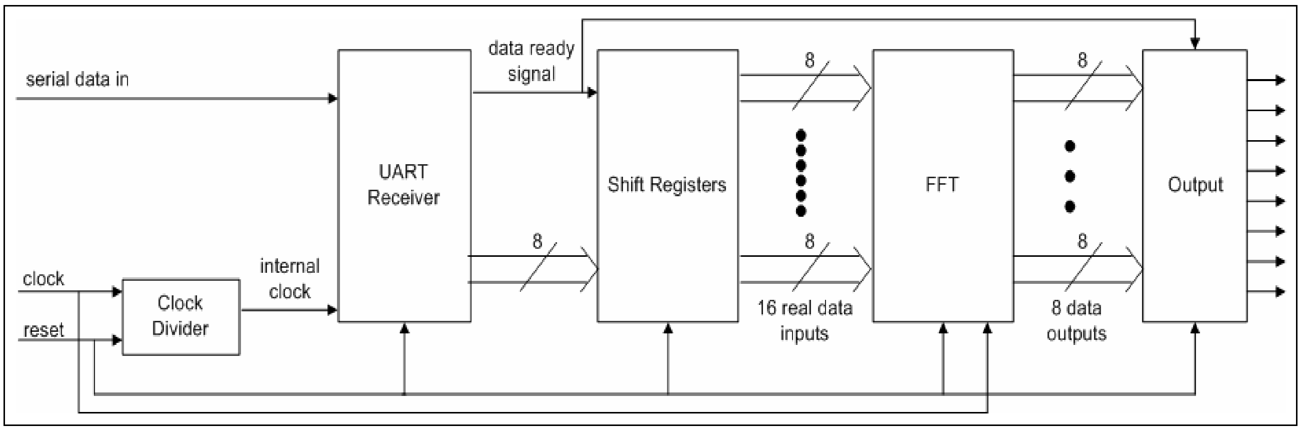

Figure 7: Frequency Identifier System

between 261.63 Hz and 523.25 Hz can be implicitly distinguished by using the conversion presented in Table 2.

One condition of this implicit recognition is that, despite the frequency of the inputs, it should have been known to the system that this frequency is limited within the range of 261.63 Hz and 523.25 Hz. For simulation purposes, the input values are manually typed into the system. For implementation, samples were saved into a text file and fed to the FPGA using Windows Hyper Terminal on a PC.

Table 2: Representation of FFT Outputs

FPGA outputs	FPGA outputs frequencies (Hz)	LEDs
0 = ω_s	261.63	9
1 =15	294.33	8
2 = 14	327.04	7
3 = 13	359.74	6
4 = 12	392.44	5
5 = 11	425.15	4
6 = 10	457.85	3
7 = 9	490.55	2
8 = $\omega_s/2$	523.25	1

Therefore, the signals between 261.625 Hz and 523.25 Hz can be represented as shown in table 3. For six out of eight, or 75% of the musical notes that want to be differentiated, one LED lighted up indicating that particular note. However, for two of the notes, instead of one, two LEDs lighted up for each note. This is because the FFT outputs are evenly spaced; whereas music notes follow a logarithmic scale. The frequencies of musical notes do not match exactly with the FFT output frequencies and two of them are located between two FFT output frequencies, giving two non-zero outputs instead of one. For the purpose of frequency identification, this representation of outputs is acceptable. Pattern of LEDs output can be used to distinguish one note from another. The LED pattern for each musical note is presented in Table 3.

Table 3: Outputs of the Frequency Identification System

Notes	Frequency (Hz)	LEDs
C	261.63	1
D	293.66	2
E	329.63	3
F	349.23	3 and 4
G	392	5
A	440	6 and 7
B	493.88	8
C’	523.25	9

4. TESTING AND VERIFICATION

The testing and verification of the system are carried out through simulation of test benches in the ISE software. The test benches are created to verify how the system responds to selected inputs that represent different musical notes.

The simulation of the frequency identification system was done part by part. The UART and clock divider were verified and tested prior to this investigation. The shift registers were tested through simulation using Xilinx ISE software. The screenshot of this simulation is not included as shift registers are reasonably simple design and they also are common components of system design.

Most simulation efforts were dedicated to verify the functionality of the FFT and LEDs output algorithm. Due to the specification of the FPGA board, the design assumes that an LED is turned on when the signal controlling it is low and turned off when the signal is high. Figure 10 shows a screenshot of the simulation generated with the ISE software. In Figure 10, the first set of input values between 0 and 160 ns represents the note D with amplitude of 127. The output corresponding to this set of inputs is propagated to the LED on the first falling edge of the signal clk_ready, which is the ready signal generated by the serial receiver. For note D, xled_2 is turned on when the output propagates to it. Before this first output is ready to be propagated to the LED, all the LEDs have undefined values. The next set of inputs between 160 ns to 310 ns represents the note G, which is shown by xled_5 taking the value of 0 in the output on the next falling edge of signal clk_ready. When all of the inputs are set to zero, all the LEDs in the outputs are turned off on the next time the outputs are propagated to the LED or on the next falling edge of the clk_ready signal.

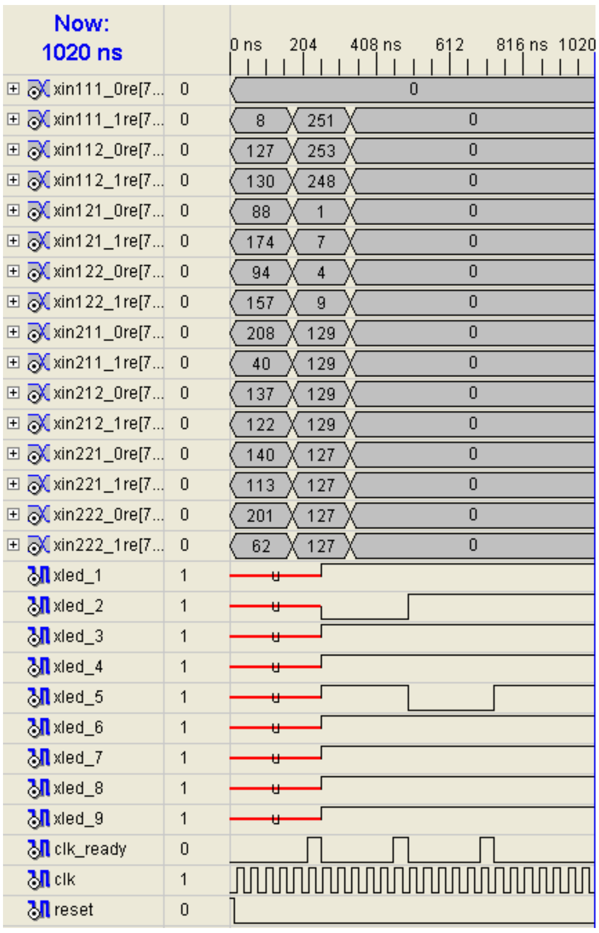

Figure 10: Simulation result of D and G

One point worth noting from the simulation is regarding the signal clk_ready. For the purpose of simulation, this signal is set as an input that can be manually modified to emulate the data ready signal generated by the serial receiver. Implemented as a full frequency identifier system as pictured in Figure 8, the data ready signal takes a significantly longer time than the 100 MHz clock of the FPGA board used to control the FFT algorithm. The difference in the frequency of the ready signal from the UART with the main clock of the board is significantly larger than the difference between clk and clk_ready in the screenshot captured in Figure 10.

5. Optimization and Implementation

For the purpose of the musical notes recognition system, resolution of the FFT computation is not a main factor. As long as the FFT provides distinguishable spikes at the correct output location, the magnitude of the spike itself is less of an issue. Due to this reason, optimization can be achieved by scaling the value of inputs, outputs and intermediate values of the FFT computation down. In this investigation the inputs and outputs were chosen to be 8-bit signed integer to ease interfacing of this block with others such as the UART, that provides 8-bit outputs. Moreover, the intermediate values are scaled to be 17-bit signed integer due to the realisation that the multiplication of two 8-bit values will result in a maximum value of 16-bit and the addition of two 16-bit values will result in a maximum value of 17-bit.

The design was implemented in the Xilinx Virtex 4 SX35 development board with the XC4VSX35-10FF668C FPGA. The SX family of the Virtex 4 is especially designed to perform DPS applications effectively. The SX 35 contains 192 Xtreme DSP slices, each containing one 18x18 multiplier, an adder and an accumulator [6]. As the design is optimised and with the powerful capability of the FPGA the utilisation of the board can be as low as 10%.

6. Conclusions

The purpose of this paper is to investigate the musical notes recognition system design and implementation on an FPGA board. This system will eventually be implemented as part of the automatic transcription system; however the integration of the two systems is subject to further investigation. From this investigation, it was found that the musical notes recognition system can be implemented for an octave of musical notes through the simple 16-point FFT algorithm with significantly low resolution of 8-bit signed integer inputs and outputs. This investigation also suggests that it is possible to implement multiple FFT to recognise all notes in the full scale of music. However, further investigation is needed to verify this prospect.

7. Acknowledgments

The authors wish to thank Kim Bourne Ng and Dr. Kelvin Tan for their supports in the writing of this paper.

References

[1] "Wikipedia, The free encyclopedia", 2006. Retreived: 1st September, 2006, from: http://en.wikipedia.org/wiki/Main_Page
[2] "Mary Had A Little Lamb sheet music", 2000. Retreived: 20th August, 2006, from: http://www.8notes.com/scores/572.asp?ftype=gif
[3] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, 2nd ed. San Diego, California: California Technical Publishing, 1999.
[4] "ISE Data Sheet", 2006. Retreived: 1 September, 2006, from: http://www.xilinx.com/publications/prod_mktg/pn0010867.pdf
[5] J. P. Bello, G. Monti, and M. Sandler, "Techniques for Automatic Music Transcription," n.d.
[6] "Virtex-4 MB Development Board User's Guide," 2005.

[7] E. Scheirer, "Extracting expressive performance information from recorded music," MIT, 1995.

[8] "Virtex-4 Family Overview", 2006. Retreived: 1st September 2006, 2006, from: http://direct.xilinx.com/bvdocs/publications/ds112.pdf

[9] "A Framework for Hardware-Software Co-Design of Embedded Systems", 2006. Retreived: 2nd August 2006, from: http://embedded.eecs.berkeley.edu/Respep/Research/hsc/abstract.html#motivation

[10] J. C. Brown and B. Zhang, "Musical Frequency Tracking using the methids of conventional and narrowed autocorrelation," *Acoustic Society of America*, vol. 5, 1991.