CONFIGURABLE 2k/4k/8k FFT-IFFT CORE FOR DVB-T AND DVB-H

Trio Adiono, Andyes Fourman D.A.S, Amy Hamidah Salman

Electrical Engineering Department, School of Electrical Engineering and Informatics

Institut Teknologi Bandung, Jl Ganesha 10 Bandung 40132, Indonesi

tadiono@paume.itb.ac.id

andyes04@students.ee.itb.ac.id

amy_setiadi @ yahoo.com

Abstract

Modulation technique uses a modifier module IFFT signal data from frequency domain to time domain. While at the demodulation part, FFT module is used to change the return signal from the output of the IFFT and converted them from the time domain into the frequency domain. FFT-IFFT modules are made to support 2k/4k/8k FFT and IFFT algorithms. FFT-IFFT 2k/4k/8k Core are built using the radix 2, radix 4 and radix 8. Core is designed to be able to receive data continuously, without buffer (temporary data container). The FFT-IFFT 2k/4k/8k module designs started with the functional description in model. Then the design of hardware architecture is made based on functional design in model. Then the architecture design will be used in making model bit precision. Furthermore the model bit precision design is used as a foundation in designing RTL. The result of FFT-IFFT modules meet the standard specified by the DVB consortium, with a maximum test frequency of FFT-IFFT 2k/4k/8k Core is 69.36 MHz using **FPGA STRATIX II EP2S60-F1020C3** that surpass the requirements in the standard DVB-T/DVB-H (40 MHz). In addition, the module has a high throughput with the average of 39.82 M sym / s.

1. INTRODUCTION

DVB-H and DVB-T use OFDM modulation technique (Orthogonal Frequency Divison Multiplexing). This modulation technique uses many sub-carriers of each other orthogonal, enabling sub-carrier modulation in OFDM are close together without any interference **[1].** DVB-T using the 2 mode modulation FFT - IFFT the 2k mode and 8k mode. While DVB-H modulation mode using only FFT-IFFT 4k. 2k mode using the 2048-point algorithm for the modulation and demodulation, 4k mode using a 4096-point algorithm for the modulation and demodulation, while for the 8k mode using 8192-point algorithm for the modulation and demodulation. IFFT processing is one of the final stage of the phase modulation and change from time to frequency domain, while the FFT is one of the first stage of the demodulation.

2. FFT –IFFT ALGORITHM

DFT for N-point sequence data {x(n)} with n=0,1,..,N-1 is defined in equation (1), with k=0,1,..,N-1 and $W_N = e^{-\frac{2\pi}{N}}$. The value N often also referred to as the length of the transformation (N-point) and index n and k are index in time domain and frequency domain.

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad (1)$$

On the other hand, *inverse* DFT for data sequence $\{X(k)\}$ with $k = 0,1,\dots,N-1$ defined in equation (2).

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk} \quad (2)$$

Equation 2 above can be generate by using equation (1) with the real exchange value of the input with the input imaginer (so also for the value of output)

Table 1 Radix that chosen

Stage	Radix
1	Radix-2
2	Radix-4
3	Radix-8
4	Radix-8
5	Radix-8
6	Radix-8

Table 2 Table value of n in the FFT-IFFT 2k/4k/8k point[2/3]

	Nilai n
FFT-IFFT 8k *point*	$n5+8n4+64n3+512n2+2048n1+4096n0$
FFT-IFFT 4k *point*	$n5+8n4+64n3+512n2+2048n1$
FFT-IFFT 2k *point*	$n5+8n4+64n3+512n2$

Meanwhile, for the k values

Table 3 Table value of k in the FFT-IFFT 2k/4k/8k point[2/3]

	Nilai k
FFT-IFFT 8k *point*	$k0+2k1+4k2+16k3+128k4+1024k5$
FFT-IFFT 4k *point*	$k1+2k2+8k3+64k4+512k5$
FFT-IFFT 2k *point*	$k2+4k3+32k4+256k5$

In the table above, can be observed that for the FFT-IFFT module 4k-point FFT module of 8k-point IFFT, we need to eliminate the value of n0 and k0. In addition, we need to divide the value of k in the FFT-IFFT module 8k with point 2. This divider is come from the radix value (radix-2) of FFT-IFFT 8k point that removed to get the FFT-IFFT module 4k point. Once the module is to get the FFT-IFFT 2k point. Here is the decomposition 8k-point FFT module of equality DFT (Equation 1).

$$X(k)=\sum_{n5=0}^{7}\sum_{n4=0}^{7}\sum_{n3=0}^{7}\sum_{n2=0}^{3}\sum_{n1=0}^{1}\sum_{n0=0}^{1} x(n5+8n4+64n3+512n2+2048n1+4096n0)W_{8192}^{nk} \quad (3)$$

As described previously, that to create a 4k-point FFT-IFFT from 8k-point FFT-IFFT, we need to turn off the value of n0 (the control signal) and k0 on 8k-point FFT-IFFT, so the decomposition will be obtained as follows [4].

$$X(k)=\sum_{n5}^{7}\sum_{n4}^{7}\sum_{n3}^{7}\sum_{n2}^{3}\sum_{n1}^{1}\sum_{n0}^{1} x\left(\begin{matrix}n5+8n4+64n3\\+512n2+2048n1\end{matrix}\right)* \underbrace{W_{4096}^{(n_5+8n_4+64n_3+512n_2+2048n_1)(k_1+2k_2+8k_3+64k_4+512k_5)}}_{Twiddle\ Factor\ for\ 4k\ point\ FFT-IFFT} \quad (4)$$

After that, bypass stage 1, so that does not produce constant x (N5 +8 n4 +64 n3 +512 n2 +2048 n1) are repeated (as many as 2 times for removal radix-2, 4 times for waste radix-4 and 8 times for the removal radix -8) So that will be generated equality 4k-point FFT-IFFT as follows

$$X(k)=\sum_{n5}^{7}\sum_{n4}^{7}\sum_{n3}^{7}\sum_{n2}^{3}\sum_{n1}^{1} x(n5+8n4+64n3+512n2+2048n1)* \underbrace{W_{4096}^{(n_5+8n_4+64n_3+512n_2+2048n_1)(k_1+2k_2+8k_3+64k_4+512k_5)}}_{Twiddle\ Factor\ for\ 4k\ point\ FFT-IFFT} \quad (5)$$

It also for 2k-point FFT-IFFT, when we want to generate from 4k-point FFT-IFFT. As described above, disabled value of n1 and k1 so that the equation becomes as follows

$$X(k)=\sum_{n5=0}^{7}\sum_{n4=0}^{7}\sum_{n3=0}^{7}\sum_{n2=0}^{3} x\left(\begin{matrix}n5+8n4+64n3\\+512n2\end{matrix}\right) \underbrace{W_{2048}^{(n_5+8n_4+64n_3+512n_2)(k_2+4k_3+32k_4+256k_5)}}_{Twiddle\ Factor\ for\ 2k\ point\ FFT-IFFT} \quad (6)$$

Comparison of proposed algorithm with Zero Padding Algorithm (Common algorithm for configurable 2k/4k/8k FFT-IFFT Core)

There is also Zero Padding algorithm, which is an algorithm with algorithm that control of incoming input. FFT-IFFT that used is a modular architecture with the biggest FFT-IFFT point (FFT-IFFT 8k point) . For example, when we use the FFT-IFFT 8k point of input 4k point, we simply add a zero input (dummy input) as illustrated below

Figure 1 Enlargement input (symbol) from the 4k point 8k point with a dummy input

Here is a comparison between the algorithms with Zero Padding Multiplexing algorithm (Algorithm offered).

Table 4 Comparisons between algorithms Zero Padding and Multiplexing

FFT Mode	**Zero Padding (#clk)**	**Multiplexing (#clk)**	**ΔUSELESS CLOCK (#clk)**
FFT 2K	16384	4096	12288
FFT 4K	16384	8192	8192
FFT 8K	16384	16384	0

3. DESIGN OF 2K/4K/8K FFT-IFFT CORE

Architecture design of FFT-IFFT 2k/4k/8k Core model based on model have been made previously. Architecture that used is pipeline data path blocks using multiple core computing memory and split into several smaller blocks. Each stage consists of a block of core memory and computing. FFT-IFFT 2k/4k/8 on the design of FFT-IFFT Core architecture is chosen because the type of pipeline data path architecture is able to receive data continuously so that it does not require additional memory buffer outside the system so that gives a higher throughput. With top level system block FFT-IFFT 2k/4k/8k overall is as follows

Figure 2 Top-level datapath diagram FFT-IFFT 2k/4k/8k Core

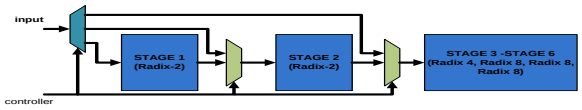
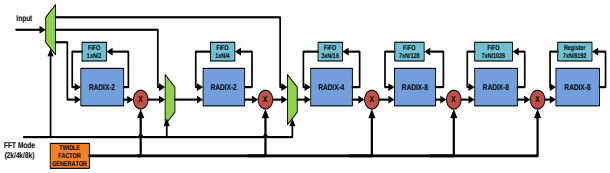
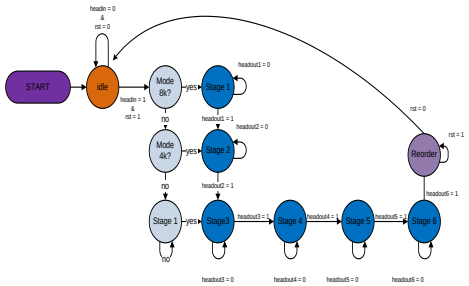


Figure 3 block Top-level diagram FFT-IFFT 2k/4k/8k Core



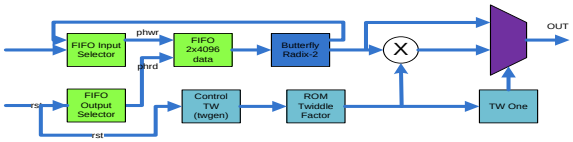
Architecture that selected is the architecture of the Single Path Delay Feedback. This architecture uses a FIFO buffer to store data temporarily before it can be computed by butterfly unit. In the Radix-2 required 1 unit FIFO buffer Radix-4 required 3 unit FIFO buffer with the length of the complex each N/4.On the other hand, Radix-8 required the FIFO buffer 7 complex with each length N / 8. Thus the length FIFO buffer in each phase will be shorter than FIFO buffer on previous stages (Figure 3).

Figure 4 FSM FFT-IFFT 2k/4k/8k Core



FSM above illustrates the datapath module of FFT-IFFT 2k/4k/8k. At the time data is inserted, initially will be checked the value of **headin (**start flag**)** signal and **rst (**reset flag**)** signal. When the **rst** and **headin** has zero value, then the module will be in idle state when the module is in idle condition, or does not do the computing. While at **rst** and **headin** have a value of one, then the process will move to the next state. In the next process, the module that used will be checked. In 8k mode, the processing stage will involve all the FFT-IFFT 2k/4k/8k stages (stage 1 to stage 6 with the reorder module), while the 4k mode will involve stage 2 up to stage 6 (the reorder module), and 2k mode will involve only stage 3 to 6 (with the reorder module). Reorder module is used as a commutator value data, so that the data that has shuffled has become the right data (data position)

Figure 5 Blok datapath radix-2 stage 1

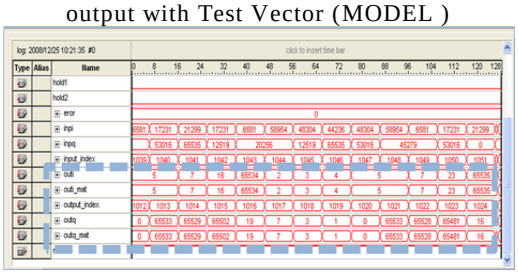


For example, we take the example datapath blocks stage 1. Writing FIFO (write phase) will be generated by 'FIFO Input Selector' and read signal (phase read) from the FIFO be generated by 'FIFO Output Selector'. After the data processed by a butterfly, the data will be multiplied by twiddle factor. Twiddle factor value will be checked by the module 'TW one'. This module will proceed directly from the output value when the twiddle factors that generated is '1 '.

4. VERIFICATION AND MEASUREMENT 2K/4K/8K FFT-IFFT CORE SYSTEM ON DVB (INTEGRATED)

The process of verification includes functional simulation, gate-level simulation, and visual verification with FPGA. Test vector for IFFT is one frame (68 symbol) OFDM with 2048 long for 2k-point FFT-IFFT Core, 4096 point for the 4k point FFT and IFFT-8k for the 8192 point FFT-IFFT Core. Data 2x16 bit complex input-output. Verification is considered successful if the output of the functional simulation (before synthesis), and the output gate-level simulation (after synthesis) exactly match the output of model.

Figure 6 Verification results of FPGA output with Test Vector (MODEL)

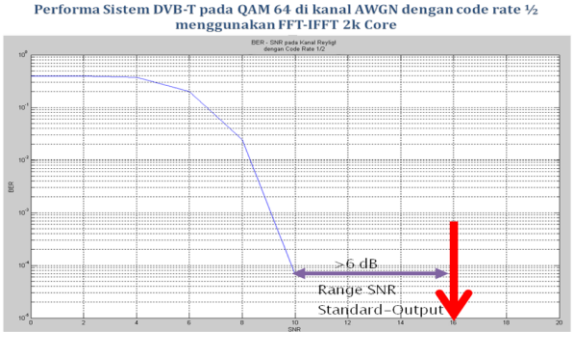


Visual verification using FPGA STRATIX II S60 F1020C3 results that verification of the of MODEL results with the RTL has been the same. This is shown by **outi** and **outq** (output FPGA) with **outi_mat** and **outq_mat** (output model), **outi** is real output and the output outq is imaginer

Table 5. Table measurements FFT-IFFT 2k/4k/8k point on the AWGN channel

Modulation	Code Rate	AWGN Channel (SNR)	BER Standard ETSI (1 frame)	Hasil simulasi (1 frame)
QPSK	1/2	3,1	2x10-4	0
QPSK	2/3	4,9	2x10-4	0
QPSK	3/4	5,9	2x10-4	0
QPSK	5/6	6,9	2x10-4	0
QPSK	7/8	7,7	2x10-4	0
16-QAM	1/2	8,8	2x10-4	0
16-QAM	2/3	11,1	2x10-4	0
16-QAM	3/4	12,5	2x10-4	0
16-QAM	5/6	13,5	2x10-4	0
16-QAM	7/8	13,9	2x10-4	0
64-QAM	1/2	14,4	2x10-4	0
64-QAM	2/3	16,5	2x10-4	0
64-QAM	3/4	18,0	2x10-4	0
64-QAM	5/6	19,3	2x10-4	0
64-QAM	7/8	20,1	2x10-4	0

Figure 7. Results measurement system with DVB hardware FFT mode 8k Code Rate ½ in the Rayleigh channel model.



In the Table 5, can be observed that the FFT 2k/4k/8k model which has been integrated with the DVB system established for the particular channel (AWGN) with the value of BER (Bit Error Rate) and SNR (Signal to Noise Ratio) is defined by ETSI. BER value obtained, for the simulation results of FFT 2k/4k/8k the SNR value zero on the same value standard by ETSI that BER value is $2x10^{-4}$. While in Figure 7 can be observed the improvised BER is better (more than 6dB) compared with the ETSI value.

5. CONCLUSION

A configurable FFT-IFFT Core has successfully designed and implemented on the FPGA. This core can perform FFT and IFFT computing 2048/4096/8192 points. Verification was done on the level MODEL model, functional RTL, RTL gate level, and visual verification on the FPGA. All the verification shows that 2k/4k/8k FFT-IFFT Core can work with good performance. Synthesis results show that the core can work at a frequency of 69.36 MHz with troughtput 39.82 M symbols / s at a frequency of 40MHz using FPGA STRATIX II EP2S60-F1020C3.

REFERENCE

[1]. **ETSI TR 101 190 V1.2.1(2004-07).** *Digital Video Broadcasting (DVB) Framing structure, channel coding and modulation for digital terrestrial television.* s.l.:European Broadcasting Union, 2004.

[2]. **W.Li.** *STUDIES ON IMPLEMENTATION OF LOW POWER FFT PROCESSOR.* Linköping, Sweden : Department of Electrical Engineering Linköpings universitet, 2003. LiU-Tek-Lic-2003:29.

[3]. **T.Wada.** The 10th International LSI Design Contest. *The University of The Ryukyus.* [Online] OctoBER 2007. http://www.ie.u-ryukyu.ac.jp/~wada/design07/spec_e.html.

[4]. **R.Chidambaram,.** *A scalable and high-performance FFT processor, optimized for UWB-OFDM.* Technishe Universiteit Delft : Microelectronics, Department of Electrical Engineering, Faculty of Electrical Engineering, Mathematics and Computer Science, 2005