Self-Timed Single Circular Pipeline for Multiple FFTs

Ryuichi TAGUCHI, Hajime OHISO, Keizo MENDORI, Kei MIYAGI, Makoto IWATA
Graduate School of Engineering, Kochi University of Technology,
Kami, Kochi, 782-8502 Japan

Abstract—Future wireless ad hoc network should accommodate different types of mobile terminals equipped with different wireless communication schemes. Especially when disaster will happen, to guarantee dependable connectivity among mobile terminals will be indispensable for delivering emergent information by using available wireless links. In order to realize such heterogeneous wireless communication systems, one of the key technologies is adaptive fast Fourier transform (FFT) engine to accept multiple wireless signal sequences with different sampling rate and different FFT point.

This paper discusses a basic idea of novel FFT engine based on the self-timed (clockless) pipeline circuit to compute multiple FFT’s in parallel. After that, the potential performance of the proposed circuit is evaluated through its FPGA implementation. Preliminary results indicate the proposed circuit could process two 4096-point FFT’s at 276 M sample/s per each FFT.

Keywords: heterogeneous wireless communication, ad hoc network, FFT, self-timed pipeline

1. Introduction

Diverse wireless communication devices have permeated throughout our daily lives in modern information society. Therefore, wireless networks supporting higher throughput and wider coverage area are increasingly demanded. Since the modern homogeneous wireless networks are facing severe limitation of transmit power level affecting coverage range and the amount of interference, heterogeneous wireless networks are becoming part of the mainstream wireless communication infrastructures [1], [2], [3]. Those heterogeneous wireless communication devices can be utilized to configure more dependable and flexible ad hoc networks, especially in emergent conditions such as natural or artificial disasters.

Heterogeneous wireless communication devices should be equipped with a multimode and multiband receiver module so as to select optimum modulation, channel, and network dynamically depending on its individual wireless communication condition. To realize such intelligent multimode devices, sophisticated radio link management decision among available wireless connections is required. Frequency domain equalizer (FDE) [4] with channel estimation is one of the most important functions for the intelligently dependable mobile terminal, because its channel estimation result can be utilized to decide an appropriate radio link and FDE itself improves the bit error rate (BER) to mitigate interference within both air and RF devices.

In our prior research project, ultra-low power data-driven networking system for ad hoc wireless network has been investigated. The final results of the project demonstrated that our ultra-low-power data-driven chip-multiprocessor LSI fabricated in 65 nm CMOS process can perform at less than a few-tenth of power of conventional embedded microprocessors [5], [6]. Since the target protocols in the project were over layer 3, for further investigation of low-power wireless networking technologies, it is necessary to study layer 2 and baseband process.

Our research project therefore aims to establish a self-timed pipeline (STP) implementation for the dependable wireless systems (DWS) supporting multimode and multiband interfaces. Since the STP circuit inherently has clockless passive operation mode [7], [8], it can flexibly process any combination of signal sequences even if they are sampled at different frequencies. In this paper, fast Fourier transform (FFT), one of the heaviest functions in the DWS, is focused on and its STP design is proposed. Finally, its feasibility is discussed through a preliminary field programmable gate array (FPGA) design of STP-based FFT circuit.

2. Pipeline Parallelism of Multichannel FFT

The single carrier FDE module performs on the receiver side after the FFT calculation to combat frequency-selective fading and phase distortion [4]. To equalize the transmitted data in frequency domain, a pilot signal is used for estimating the transfer function and the noise power in the air channel. Therefore, after the received data are transformed from time domain to frequency domain by FFT, they are equalized based on estimated results and then retransformed to time domain by IFFT.

In case of orthogonal frequency division multiplexing (OFDM), FFT is also used for modulating data onto each subcarrier and IFFT is for demodulating data on each subcarrier. Furthermore, in multiple-input multiple-output (MIMO) antenna configuration, a multichannel FFT/IFFT processor is necessary in a transmitter/receiver.

Therefore, we aim to implement a multichannel FFT processor in which multiple FFT operations of variable...
sizes are simultaneously performed for multiple input signal sequences sampled in variable frequencies.

Originally FFT is a fast version of discrete Fourier transform (DFT). N point DFT is defined by the equation (1).

\[ X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn} \]  

where the input sequence of N complex data \( x(0), x(1), \ldots, x(N-1) \) is transformed into an N-periodic sequence of complex data. In the Cooley-Tukey FFT algorithm, radix-\( r \) butterfly operations are recursively applied to \( N \) input signals, and the depth of recursion is \( \log_r N \). In each recursion, the number of butterfly operations (i.e., \( r \)-point FFT) is \( N/r \) and they can be calculated in parallel, because there is no data dependency among them. An example of a decimation-in-time FFT (\( N=8, r=2 \)) is shown in Figure 1.

As seen in this dataflow diagram, four butterfly operations can be concurrently executed in each recursion step.

![Dataflow diagram of radix-2 decimation-in-time FFT (N=8)](image)

Fig. 1: Dataflow diagram of radix-2 decimation-in-time FFT (N=8).

If the dataflow diagram of FFT shown in Figure 1 is interpreted based on the dynamic dataflow model [7], multiple instances of the same FFT diagram can be allowed to be executed by introducing a channel identifier, which differentiates between them. In the same way, the dataflow diagram of a butterfly operation can be interpreted for multiprocessing of the butterfly if every data flowing the diagram have a set of identifiers \( ID \), which is composed of channel identifier \( ch \), step identifier \( step \), and butterfly instance identifier within the step \( btf \). In this case, it is necessary to provide a function supplying appropriate set of operands with those identifiers and storing intermediate data in the memory buffer. This parallel execution scheme in case of radix-2 butterfly is illustrated in Figure 2. In the figure, every operand and result of butterfly are identified by \( ID(ch, step, btf) \) and multiple sets of operands are issued from the commutator consecutively. The commutator manages the number of operand sets for butterfly instances, which represents the degree of parallelism \( P_{ch} \). At the same time, the commutator attaches appropriate identifiers to those issued operands.

![Parallel execution scheme of multiple butterfly instances](image)

Fig. 2: Parallel execution scheme of multiple butterfly instances.

As long as the butterfly operation with a correct set of identifiers is executed under dynamic dataflow model, valid execution of multiple FFT calculations is guaranteed even if the size of an FFT \( N_{ch} \) and the sampling frequency of its input data sequence are different from others.

However, the connectivity in each step is different from that in others, as seen in Figure 1 so that commutator might be complex. Therefore, in our design, the original FFT structure is modified to uniform FFT structure shown in Figure 3. By adopting this FFT structure, the ID handling function in the commutator is simply defined as shown in Figure 4.

![Uniformity in radix-2 FFT structure (N=8)](image)

Fig. 3: Uniformity in radix-2 FFT structure (N=8).

Furthermore, this uniform type of FFT structure allows the commutator to fetch \( r \) operands from the buffer memory in parallel. Because the memory access pattern is invariant at all steps, the buffer memory can be composed of \( r \) single-port memory banks.

Figure 5 shows a dataflow graph representation of the proposed parallel execution scheme of multichannel FFT. Firstly, the input complex data is stored in multi-bank buffer memory consecutively. If a set of operands for the first


```latex
\begin{align*}
\text{do} & \left\{ \\
& \quad btf += P_{ch}; \\
& \quad \text{if} (btf \geq (N_{ch}/r)) \{ \\
& \quad \quad \quad \text{step}++; \\
& \quad \quad \quad btf \%= (N_{ch}/r); \\
& \quad \} \right. \\
& \left. \text{while} (\text{step} < \log(N_{ch})); \right. \\
\end{align*}
```

Fig. 4: ID handling function in commutator.

3. STP Implementation of Multichannel FFT

The parallel execution scheme proposed in the previous section is an abstract model and that does not deal with temporal information such as sampling frequency. In order to process multiple input sequences with different frequencies, its circuit implementation should have a passive operation mode enabling to accept input data adaptively along with outside conditions. Therefore the self-timed pipeline (STP) circuit is employed for implementing multichannel FFT.

In this section, the passive and autonomous behavior of the STP is briefly introduced and then its natural contribution to multichannel FFT implementation is discussed.

3.1 Self-Timed Pipeline

Each pipeline stage of the STP consists of a data latch as a pipeline register, function logic, and transfer control unit named C-element. The basic structure of the STP is shown in Figure 6. The data latch, function logic, and C-element are denoted by DL, Logic, and C, respectively. The data is packed with tag into packet form, and the packet is transferred between the pipeline stages as a result of the communication between the C’s in the adjacent stages. The communication is performed stage-by-stage according to the 4-phase handshake protocol [9] by using transfer request and acknowledge signals which are called send signal and ack signal respectively. The stage-by-stage transfer control channel changes the states of each pipeline stage independently, and the states of the stages are defined below according to the handshake protocol. Here, the C-element in the i-th stage is denoted by $C_i$.

- **Reset state**: The send and ack signals are negated after the assertion of the reset signal.
- **Idle state**: The $C_i$ waits until the $send_{i-1}$ is asserted.
- **Busy state**: The $send_{i-1}$ is asserted at the beginning of the transfer of the packet from the precedent ($i-1$)-th stage. After the assertion of the $send_{i-1}$, the $C_i$ asserts its ack signal (ack$_{i-1}$). In response to the assertion, the $C_{i-1}$ negates the $send_{i-1}$. After that, if and only when both the $send_{i-1}$ and ack$_{i-1}$ are negated, the $C_i$ asserts the ToDL$_i$ to open the DL$_i$, and it asserts send$_i$ at the same time. As a consequence, the packet is latched in the $i$-th stage, and the $i$-th stage goes to idle state. Otherwise, the $C_i$ waits until the ack$_i$ is negated while it keeps its send and ack signals.
The successive stages receiving the assertion of the send signal go to busy state and their C’s repeat the same transfer control sequence individually. During the handshakes, the send signals are delayed to assure the completion of the primitive logic function and ack signals are delayed to assure the setup-hold timing of the DL’s.

This stage-by-stage transfer control of the STP suggests the timing of the power controls. That is, in the idle stages, the circuit of the DL and combinational logic can be powered-off, i.e., the supply-voltage can be cut, while that of the C and sequential logic can be powered-down, i.e., the supply voltage can be lowered enough to keep the circuit’s states [10]. Moreover, in the busy stages, those circuits should be powered-down enough to assure the switching of the transistors, i.e., the supply-voltage can be lowered as long as the required switching speed is achieved [11].

3.2 Multichannel FFT Implementation

In order to realize the dataflow shown in Figure 5, it is essential to maintain stable dataflow in the STP without any pipeline bottleneck as well as to guarantee atomic (i.e., read-after-write) accesses of intermediate data stored in the buffer memory during execution. Therefore, the buffer memory accesses must be integrated at the single STP stage. Moreover, intermediate resultant data of radix-$r$ butterfly are written in different buffer addresses from that of operands when the uniform type of FFT structure is employed. Thus, in our design, we adopt dual buffer memory modules each of which is used for butterfly operations in either even step or odd step of the FFT. It requires $2N_{ch}$ words SRAM for $N_{ch}$-point FFT. As a result, multichannel FFT engine is designed as shown in Figure 7 to utilize the passive operation mode of STP. This FFT engine operates as follows.

All data flowing in the pipeline has an operation code $op$ as well as $ID(ch, step, btf)$. The $op$ is assigned one of operations, i.e., $in$, $read$, $write$, or $out$. Every stage in the STP-based FFT engine changes its operation depending on $op$ of the packet.

- **input phase**: $N_{ch}$ input data from a channel $ch$ consecutively arrive at one of input ports of the merge stage. At that time, each input data is composed as a packet form including a complex number, $ch$ identifier, index $i(=0,\ldots,N_{ch}-1)$ and $op (= in)$. The input data reaching to the buffer memory stage is written in a place associated with index $i$ of the packet.

- **instantiation phase**: If $op$ of a packet arriving at the ID handler stage is $in$, an FFT for the channel $ch$ may be instantiated. Only when $r$ operands necessary for the first butterfly are stored in the buffer memory, an FFT instance for the channel $ch$ is initiated. $ID(ch, 0, 0)$ for the first butterfly is issued with $op (= read)$. After that, succeeded butterfly instances are instantiated with $ID(ch, 0, 1),\ldots, ID(ch, 0, P_{ch}-1)$ within the allowable degree of parallelism $P_{ch}$.

- **read phase**: If $op$ of a packet arriving at the buffer memory stage is $read$, $r$ operands for the $btf$-th butterfly instance are read out from the buffer memory in parallel. Their addresses can be calculated from $ID(ch, step, btf)$. To allow those parallel accesses, the buffer memory is composed of dual $r$-way memory banks. If the $step$ is even, the operands are read from the first set of $r$-way memory banks. If odd, the second one is accessed for operand fetches. At the same time, $(r-1)$ twiddle factors necessary for the butterfly are fetched from the twiddle factor lookup table in parallel. Since the lookup table holds twiddle factors only in the fourth quadrant, each lookup data needs the change of quadrant by swapping the real and imaginary number, or changing on one (or both) sign(s) of the number(s).

- **butterfly phase**: If $op$ of a packet arriving at the butterfly stage is $read$, a butterfly instance is executed using $r$ operands and $r-1$ twiddle factors.

- **prerelease ID phase**: If $op$ of a packet arriving at the ID handler stage is $read$, this stage prepares to write $r$ resultant data to the buffer memory. In the uniform type of FFT, all results of a butterfly should be stored to one of memory banks. Therefore, word length of each memory bank is expanded to $r \times (\text{length of a complex word})$. By this expansion, all results are written at the same time. In this prerelease ID phase, $r$ results are packed into one word with $op (= write)$ to prepare for the next writing phase.

- **write phase**: If $op$ of a packet arriving at the buffer memory stage is $write$, a packed result of the $btf$-th butterfly instance is written in the buffer memory. Its address can be calculated from $ID(ch, step, btf)$. If the $step$ is even, the intermediate result of the FFT is written in the second set of $r$-way memory banks. If odd, the first one is accessed for the result storing. After writing the result, the ID packet is transferred to the ID handler stage. In this stage, $ID(ch, step, btf)$ is updated based on the function defined in Figure 7 and then $op$ is changed to $read$. If $step$ exceeds $log_{2} N_{ch}$, the FFT operation is finished and buffered data are output. In this case, $op$ is changed to $out$.

4. Evaluation

In order to evaluate the performance of the stream-driven FFT processing, its STP circuit is designed and implemented on FPGA (Stratix II, Altera Corp.). Table 1 shows the specifications of the implemented self-timed FFT circuit on the FPGA. This FPGA design was conducted for confirming that parallel butterfly instances works well in pipelined parallel processing of the single circular STP circuit. In the preliminary result of non-optimized circuit, the processing time of 1024-point FFT was 424 us. This means that the acceptable sampling rate is 2.4 M sample/s. In this case, the maximum pipeline tact in 13 pipeline stages was 20.6
ns. The number of logic cell (LC) Combinational, LC Registers, block memory bits, and DSP slices required for the designed FFT circuit are 1200, 2008, 40062, and 12 respectively.

Table 1: Specifications of the self-timed FFT circuit on FPGA

<table>
<thead>
<tr>
<th># of FFT points $N_{ch}$</th>
<th>16 ~ 1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radix $r$</td>
<td>4</td>
</tr>
<tr>
<td>Degree of parallelism $P_{ch}$</td>
<td>2</td>
</tr>
<tr>
<td>Complex data (real, imaginary)</td>
<td>32 bit, fixed point integer (16bit, 16bit)</td>
</tr>
<tr>
<td># of STP stages</td>
<td>13</td>
</tr>
</tbody>
</table>

The processing time $T_{FFT}$ of the implemented STP circuit can be approximated by the equation (1).

$$T_{FFT} = \left( \frac{r - 1}{r} N_{ch} + 2 \frac{N_{ch} \log_2 N_{ch}}{r P_{ch}} S \right) * T_{fmax}$$

where $T_{fmax}$ denotes the maximum pipeline tact in the STP, $S$ denotes the number of pipeline stages, and $P_{ch}$ denotes the number of parallel instances of butterfly. The first term in this equation expresses the initiation delay to wait the arrival of a set of operands for the first butterfly. The second term expresses the parallel execution time of all butterfly instances. As understood from this equation, $r$ and $P_{ch}$ should be increased and $T_{fmax}$ should be decreased in order to shorten the FFT processing time $T_{FFT}$.

Based on our ASIC design experiences for the STP, $T_{fmax}$ can be reduced to about 1 ns if 65 nm CMOS process will be used to implement the proposed FFT circuit. Through those considerations, the performance of the proposed self-timed FFT circuit can be estimated as shown in Figure 8. For example, two 4096-point radix-8 FFTs will be performed at 276 M samples/s in case of 8 parallel instances, i.e., the degree of pipeline parallelism for radix-8 butterfly operations.

5. Conclusion

In order to establish low-power and dependable wireless networks, mobile devices should be equipped with a multimode and multiband transceiver/receiver module for both cellular networks and ad hoc networks. Such mobile terminals would be useful and sustainable in case of emergent situation as well as normal situation.

This paper focuses on a multichannel FFT engine used for MIMO OFDM and SC-FDE and proposes a basic idea of its self-timed pipeline implementation based on the dynamic data-driven multiprocessing model. The preliminary results indicated that the proposed STP implementation will be feasible to the required performance for heterogeneous wireless communication environment involving mobile broadband wireless access (MBWA), wireless local area network (WLAN), wireless personal area network (WPAN).
Acknowledgement

Although it is impossible to give credit individually to all those who organized and supported our project, the authors would like to express their sincere appreciation to all the colleagues in the project.

This research work was supported in part by Core Research for Evolutional Science and Technology (CREST), Japan Science and Technology Agency (JST). The circuit design work was supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc. and Cadence Design Systems, Inc.

References