THE VLSI IMPLEMENTATION OF 1D-DWT BASED ON GALS SYSTEMS

Shengxian Zhuang, Weidong Li, Jonas Carlsson, Kent Palmkvist, and Lars Wanhammar

Division of Electronics Systems
Department of Electrical Engineering, Linköping University
SE-581 83 Linköping, Sweden, zhuangsx@isy.liu.se

ABSTRACT

In this paper, we propose a VLSI implementation method for one-dimensional discrete wavelet transform (1D-DWT) based on the GALS systems approach. An asynchronous wrapper, which includes two data communication ports and a local clock controller, is designed for the asynchronous data communication between the locally synchronous filtering modules in the wavelet filter bank. The detailed design methodology for the GALS architecture of 1D DWT is given and the circuits are validated with VHDL and circuits simulation in standard CMOS technology.

1. INTRODUCTION

Discrete wavelet transform has become an important approach for the analysis of a transient signal since a connection was made between the wavelet transform and multirate filter bank trees by Mallat[1]. This inherent relation to the multiresolution decomposition of a signal gives rise to the intensive interests in the VLSI implementation for the DWT, although it is highly expensive. Many algorithms and schemes have been proposed for reducing the computation complexity and improving the hardware efficiency, such as recursive pyramid algorithm, the lifting scheme and various of systolic architectures[3-6]. However the early architectures for computing DWT was not well suited for VLSI since it used large multiplexors and memories for routing and storing the intermediate results. The lower bound for computing the DWT [2]shows that the small area and fast computation rate can’t be achieved at the same time. In some cases, the completely pipelined parallel architecture is more suited for the real time signal processing. But the complex connections among the filters and allocation and distribution of global clock for the decimation and interpolation may make it difficult to use the fully synchronous design method. [7] proposed the asynchronous filter banks for the DWT. It used a two-phase handshake protocol conversion to reduce the redundancy of the filter output downsampling. But it was limited to the specific wavelet form, i.e, the Haar due to complex control circuitry.

In this paper, with an aim to simplify the connections between the filtering modules, we propose an unified and modular architectures for the VLSI implementation of 1-D DWT based on globally asynchronous locally synchronous(GALS) approach[8][9]. In our proposed scheme, each filter in an octave band is constructed as locally synchronous module and equipped with asynchronous data communication ports and local clock controller. In addition, the downsampling and upsampling in the DWT and IDWT are dealt within the local synchronous module with an clock allocation technique. Thus the GALS system of 1D-DW behaves flexible and reliable with a high data throughput by a fully pipelined architectures. In the next section, the discrete wavelet transform is firstly introduced and it is followed by the detail design of the GALS system of 1D-DWT. In section 4, we discuss the circuit simulations both with VHDL and Cadence in standard CMOS technology. Lastly, the conclusion is given in section 5.

2. DISCRETE WAVELET TRANSFORM

From signal processing point of view, The wavelet transform of a sequential signal is to recursively decompose a sampled sequence of a signal into two components in octave bands, which are the approximation and detail of the output of the former decomposition. This recursively decomposition of input signal is expressed with convolutional computations of input signal sequences and the filtering coefficients as follow

\[ A_{j+1}(n) = \sum_{k=0}^{L-1} h(k)A_j(2n-k) \]  

\[ D_{j+1}(n) = \sum_{k=0}^{L-1} g(k)A_j(2n-k) \]

Where \( A_{j+1}(n), D_{j+1}(n), j = 0, 1, 2, 3, \ldots \) are respectively the approximation and detail at jth decomposition to the input sequence. \( A_0(n) \) represents input sequence of the original sampled signals. \( h(k), g(k) \) are the coefficients of the high pass and low pass filters in octave bands, which are relative to the specific wavelet basis selected in the wavelet transform. L is the number of taps of the filters. Obviously, each output of the convolutional computation is decimated by a factor of two. This algorithm of the 1D-DWT is also correspond to a down sampled filter bank which is shown in Fig.1.
The inverse discrete wavelet transform (IDWT) does the exactly opposite computation of the DWT, which is expressed as

\[ A_{j-1}(n) = \sum_{k=0}^{L-1} h'(n-2k) A_j(k) + \sum_{k=0}^{L-1} g(n-2k) D_j(k) \]  \hspace{1cm} (3)

Similar to the DWT, the computation of IDWT can be implemented with a upsampled filter bank. All the coefficients of the filters in DWT and IDWT are relative to the selected wavelets. However, for the decomposed signal to be reconstructed to the original signal stage by stage, the analysis filters and synthesis filters must satisfy the perfect reconstructible quadrature mirror filter (PRQMF) property.

Though the filters in octave bands can take various forms, for simplicity to the implementation of 1D-DWT filter bank, we employ the 4-tap FIR filters. Therefore the filters in the 1D-DWT filter bank has the following form

\[ h(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} \] \hspace{1cm} (4)

\[ g(z) = g_0 + g_1 z^{-1} + g_2 z^{-2} + g_3 z^{-3} \] \hspace{1cm} (5)

The basic architecture of the 4-tap filter can be configured in fig.2 which shows four multiplications and a summation of the products are required for each computation. In this paper, we use this filter structure as the prototype of each filtering module in the wavelet filter bank.

3. THE GALS ARCHITECTURE FOR 1D-DWT

3.1. Asynchronous Wrapper

In the filter bank implementation of 1D-DWT, the low and high pass filters feature the same hardware structure with only difference in coefficients. Thus the system design of 1D-DWT could be simplified if an unified module is configured for the hardware structure of each filtering block and an easily modular combination method is used. One of the modular systematic design approaches is to use GLAS architecture[9]. For building the GALS structure of the 1D-DWT, an asynchronous wrapper is needed to provide the filtering module with a local clock for internal computation and the interfaces for data communication. Fig.3 shows the 4-tap FIR filtering block encapsulated with the independent clock controller and data communication ports, which is a basic local synchronous module (FIR-LS) in the GALS system for 1D-DWT.

In our proposed asynchronous wrapper, the stretchable local clock controller, i.e., lclk in Fig.4, consists of a ring clock generator, a Müller C element and a NOR gate. The generated local clock signal is fed back to the inputs of Müller C element both with inversion, but with different delay. According to the properties of Müller C element, if its output, i.e., the local clock, is high, both of its inputs will go to low after some delay which makes the output return to low. Similarly, the output can’t maintain at the low state if no stretch request happens at which stretch request is initialized. Multiple stretch requests can be combined with an OR gate to generate a request. So any stretch request could cause the local clock be stretched at the low state.
The interface circuits include two ports for controlling the asynchronous data communication, named respectively with W-port and R-port. The developed W-port, which is shown in Fig. 5, is used for handling the data output to the filtering module at the next coarser level. It is activate with the internal request, i.e., \( W_r^+ \), when the computation is completed. The clock stretch request and data output request are generated sequentially with \( STRETCH_1^+ \) and \( REQ^+ \). The acknowledge with \( ACK^+ \) from the other modules will clear the requests for stretching the clock and the output of data with \( STRETCH_1^- \) and \( REQ^- \). Likewise, the R-port, which is shown in Fig.6 is used to read the data by giving the acknowledge to the request from the preceding filtering module. It is also independently activated by the internal request, i.e., \( R_d^+ \), which is followed by the \( STRETCH_2^+ \) and \( ACK^+ \) if a request arrives. The \( ACK^+ \) latches the input data and is used to initialize the request in the corresponding W-port. The most common used four-phase bundled data handshake signalling is employed for the asynchronous transmission between the W-port and R-port in the consecutive modules. That means every signal experiences two states for a complete data communication. Even though the W-port is the active port and the R-port is the passive in terms of handshake protocol, both the W-port and R-port are independently enabled by the internal demands from their own LS modules. Such constraints match most of the cases in GALS systems and make the asynchronous wrapper more flexible and robust.

3.2. The GALS System Architecture

With the asynchronous wrapper discussed above, all the filtering modules in the filter bank for 1D-DWT can be connected to construct a GALS system. Fig. 7 shows the GALS architecture for a 3-level 1D-DWT filter bank. Only the handshake signalling path is presented for the concisely explanation of the GALS architecture for 1-D DWT, because the data path is standard and the function of the FIR filtering module is isolated from the interface circuits. The data source is a LS module with a single R-port, which is used to provide the original sample signal sequence to be analyzed by means of the DWT. It can be a memory block equipped with handshake circuits or replaced by the software programming functioned as the output of data in sequence. Each filtering module, namely FIR-LS, functions as a 4-tap filter and a decimation by factor of two. The data transmission among the modules occur independently, but the sending modules must wait for the data has been fetched by both of the corresponding receiving modules. This is completed with an AND gate to synchronize the data output. A high processing frequency could be achieved due to pipelined independently operations of the filtering modules.

With a local clock, the computation in each FIR-LS module, which include 4 multiplications and a summation of the four products, can be schemed as synchronous paradigms which are synthesized with high level CAD tools. the design of each FIR-LS module is henceforth trivial. For downsampling the output of the FIR-LS, the input data is decimated by factor of two. It can be done by allocate the activations of the W-port and R-port with different frequencies. We determine these frequencies with \( f_{Wr} = 2f_{Rd} = 4f_{clk} \). It means each FIR-LS inputs a data for every two clock cycles, outputs a data for every four clock cycle. Due to the possible stretching of the clock for the activation of the R-port, two clock cycles is required for every data input in order to the computation be completed correctly.

As the 1D-IDWT, an exactly inverse computation is executed to reconstruct the original sequence, the operation of each filtering module is similar to that in DWT, but with upsampling the output. The interpolation of input data also can be dealt with scaling the local clock frequency with \( f_{Rd} = 2f_{Wr} = 4f_{clk} \).
4. SIMULATION

The GALS system for the implementation of 1D-DWT can be simulated with any VHDL simulation tools. We did the functional simulation with its behavioral model in FPGA advantage. Because the 4-tap FIR filtering computation is standard for synchronous design, The performance of the GASL systems is mainly dependent on the asynchronous wrapper. An asynchronous wrapper is built and simulated with Cadence in standard 0.35u CMOS technology in order to verify the performance. Fig. 8 shows the analog traces of the signals on the W-port and R-port in data communication. Both the activations of the W-port and R-port reliably stretch the low phase of the local clock. The computation results of the times that some critical signal transitions took are shown in table 1. We can see the least cycle for the W-port and the R-port is about 2.37ns and 2.15ns respectively. With the asynchronous wrapper, a 400 MHz data throughput can be achieved which is suitable for the computation of 1D-DWT.

Table 1: Timing for critical events.

<table>
<thead>
<tr>
<th>Events</th>
<th>Wr+ - Stretch+</th>
<th>Stretch+ -Req+</th>
<th>Ack+ - Req+</th>
<th>Req+ - Stretch+</th>
</tr>
</thead>
<tbody>
<tr>
<td>W-port</td>
<td>563 ps</td>
<td>215 ps</td>
<td>841 ps</td>
<td>635 ps</td>
</tr>
<tr>
<td>Events</td>
<td>Rd+ - Stretch+</td>
<td>Stretch+ -Ack+</td>
<td>Req+ - Ack+</td>
<td>Req+ - Stretch+</td>
</tr>
<tr>
<td>R-port</td>
<td>412 ps</td>
<td>129 ps</td>
<td>564 ps</td>
<td>771 ps</td>
</tr>
</tbody>
</table>

5. CONCLUSION

In this paper, we propose a VLSI implementation method for the 1D-DWT using the GALS systems approach. An asynchronous wrapper with novel handshake circuits including two data-ports and a stretchable clock controller is developed. The simulation results show a high performance can be achieved for the VLSI implementation of 1D-DWT based on the GALS systems.

REFERENCES