Tutorial 2 - Discrete Walsh Transform Processor (solution)

Philip Leong

July 10, 2017

1 INTRODUCTION

Complex waveforms can be decomposed into harmonic components using the discrete Fourier transform. Similarly, they can be decomposed into a sum of Walsh functions via a discrete Wavelet transform (DWT). In this laboratory, we will develop a DWT accelerator. The explanation of the FWT algorithm below follows that of Beauchamp [1].

1.1 DEFINITIONS

A real series $S_n(t), n = (0, 1, \ldots)$, is *orthogonal* with weight K over the interval $t \in [0, T]$ if

$$\int_0^T K S_n(t) S_m(t) dt = \begin{cases} K & \text{if } n = m \\ 0 & \text{if } n \neq m \end{cases} \tag{1}$$

where n and m are integers, and K is a non-negative constant. If $K = 1$, it is an *orthonormal* series.
For an orthogonal $S_n(t)$, a time-varying function, $f(t), t \in (0, T)$ can be represented as:

$$f(t) = \sum_{n=0}^{\infty} C_n S_n(t) \tag{2}$$

where $C_n \in \mathscr{R}$.
In a practical implementation, the sum in Equation 2 must be finite, and the mean-squared approximation error is

$$E = \int_0^T [f(t) - \sum_{n=0}^{N} C_n S_n(t)] dt \tag{3}$$

which is minimised by setting

$$C_n = \frac{1}{T}\int_0^T f(t)S_n(t)dt. \tag{4}$$

A function set is *complete* if Equation 3 converges monotonically to zero with N. Examples of complete function sets include the Fourier series and the Walsh series. An example of an orthogonal function set which is not complete are the Rademacher functions.

1.2 WALSH FUNCTIONS

An alternative to expressing the Walsh functions in terms of Rademacher functions as described previously, is in the form of the continued product:

$$\begin{aligned} WAL(n,t) &= W_n(t/T) && (5)\\ &= WAL(n_{p-1}, n_{p-2}, \ldots, n_0; t_{p-1}, t_{p-2}, \ldots, t_0) && (6)\\ &= \prod_{r=0}^{p-1} (-1)^{n_{p-1-r}(t_r + t_{r+1})} && (7) \end{aligned}$$

where $t \in \{0, 1, \ldots, T\}$; and $(n_{p-1}n_{p-2}\ldots n_0)_2$ and $(t_{p-1}t_{p-2}\ldots t_0)_2$ are the binary representations of the p-bit unsigned integers n and t.

1.3 DISCRETE WALSH TRANSFORM

A Lebesque integrable function $f(t), t \in (0,1)$ can be represented by a Walsh series over $(0,1)$ as

$$x(t) = a_0 + a_1 WAL(1,t)) + a_2 WAL(2,t)) + \ldots \tag{8}$$

where $a_k = \int_0^1 f(t)WAL(k,t)dt$. The Walsh transform pair can then be defined as:

$$\begin{aligned} f(t) &= \sum_{k=0}^{\infty} F(k)WAL(k,t) && (9)\\ F(k) &= \int_0^1 f(t)WAL(k,t)dt. && (10) \end{aligned}$$

For the discrete case, the trapezium rule can be applied with $N = 2^p$ (where p is a non-negative integer) sampling points x_i to give the finite linear DWT transform pair:

$$\begin{aligned} X_n &= \frac{1}{N}\sum_{i=0}^{N-1} WAL(n,i)x_i,\ i = 0,1,\ldots,N-1 && (11)\\ x_i &= \sum_{n=0}^{N-1} WAL(n,i)X_n,\ i = 0,1,\ldots,N-1. && (12) \end{aligned}$$

where $WAL(n,x)$ is the Walsh function defined in the previous tutorial. It is important to note that:

- The computation involved is simply an $N \times N$ matrix-vector multiplication with the Walsh matrix $W_N = [WAL(0,t); WAL(1,t), \ldots, WAL(N-1,t)]$ $t = 0,1,\ldots,N-1$.

- Since $WAL(n,t)$ only takes the values $\{-1,+1\}$, transforms can be computed using only subtraction and addition.
- Both the forward and inverse transform are the same computation with different scaling factors.

1.4 Examples

The 8-point DWT of $x = (0,1,2,3,4,5,6,7)^T$ is

$$X = W_8 x \tag{13}$$

$$= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{pmatrix} \tag{14}$$

$$= \begin{pmatrix} 3.5 & -2.0 & 0.0 & -1.0 & 0.0 & 0.0 & 0.0 & -0.5 \end{pmatrix}^T \tag{15}$$

The inverse DWT of $X = \begin{pmatrix} (3.5 & -2.0 & 0.0 & -1.0 & 0.0 & 0.0 & 0.0 & -0.5)^T \end{pmatrix}$ is

$$W_8 X = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{pmatrix} \times \begin{pmatrix} 3.5 \\ -2.0 \\ 0.0 \\ -1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ -0.5 \end{pmatrix} \tag{16}$$

$$= \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{pmatrix}^T \tag{17}$$

$$= x \tag{18}$$

2 Laboratory Questions

1. Discrete Walsh transform processor (30%). Make a combinatorial, parallel implementation of an $N = 64$ DWT processor for the Altera Cyclone V 5CSEMA5 FPGA used in the DE1-SoC board. Your inputs should be 16-bit integers in two's complement form, and your output represented as a two's complement fraction with sufficiently large wordlength that overflow cannot occur.

   Create a set of random test vectors and verify that your design is correct via simulation. The FPGA design tools report the maximum clock rate, f_{max} which can be achieved by

your design. What is this value? The maximum throughput is thus $2Nf_{max}$ bytes/sec. Calculate the throughput of your design.

2. Pipelined DWT processor (30%). Modify your DWT processor so that it is pipelined and verify via simulation. What is the new design's maximum throughput? What is the speedup compared with the non-pipelined design?

3. Multicycle execution (40%). It may not be feasible to supply the DWT processor with 64 high-speed, parallel inputs. Develop a modified version of the DWT processor which takes 2 inputs samples per cycle, i.e. it takes $N/2$ cycles to obtain a complete input vector. Redesign your processor so that it minimises the area-delay product (area being measured in LUTs) and can process streaming input data without stalling. What is the maximum performance in bytes/sec?

4. Comparison (bonus 20%). Integrate the DWT processor with an linear feedback shift register based random number generator which is used as the source of the input data. Implement the design in hardware and verify its correct operation. Identify the sources of bottlenecks in your design.

3 SOLUTIONS

REFERENCES

[1] K.G. Beauchamp. *Walsh functions and their applications.* Academic Press, 1975.