Comprehensive Address Generator for Digital Signal Processing

Ramesh Kini M.†, Sumam David S., *Senior Member, IEEE*
Department of Electronics and Communication Engineering,
National Institute of Technology Karnataka, Surathkal, INDIA - 575025
†email: rameshkinim@gmail.com

***Abstract*—Computational efficiency of Signal Processing Algorithm implemented in hardware depends on efficiency of datapath, memory speed, and generation of addresses for data access. In case of signal processing applications, pattern of data access is complex in comparison with other applications. If implemented in a general purpose processor, address generation for signal processing applications will require execution of a series of instructions and use of datapath elements like adders, shifters etc. In general, considerable processor resources and time are utilized. It is desirable to execute one loop of a kernel per clock. This demands generation of typically three addresses per clock: two addresses for data sample/coefficient and one for storage of processed data. A set of dedicated, efficient Address Generator Units (AGU) will definitely enhance the performance. This paper focuses on design and implementation of Address Generators for complex addressing modes required by Multimedia Signal Processing algorithms. Among other addressing modes, a novel algorithm is developed for accessing data in a Bit-Reversed order for Fast Fourier Transforms (FFT), and Zig-zag order for Discrete Cosine Transforms (DCT). When mapped to hardware, this scales linearly in gate complexity with increase in the size and uses less components.**

I. Introduction

Offline data processing or stream data processing of large number of data points involves feeding the datapath with data in appropriate sequence; as decided by the algorithm or the dataflow diagram. Patterns of data access can be identified and these patterns can be called as *addressing modes*. As it is evident that addressing mode can be viewed as primitive process of generating the address of next element of data, with *a priori* knowledge of the address of the data that is being currently accessed. Address generation needs simple arithmetic operations like addition, subtraction, comparison etc. Older processors utilized the ALU for address generation, reducing the availability of the computational elements for data processing. As the processor and the VLSI technology improved, dedicated computational units are built for address generation, thus address generation is fast and concurrent with data processing.

Digital Signal Processors (DSP) may have support for address generation of complex addressing modes like circular or butterfly. The programmer needs to initialize and control the address generation at the beginning and end of loops. For example, butterfly address generation for 8-point FFT will have 3 loops and the programmer will have to initialize the values of registers at the beginning of each loop and monitor the termination of the loop. This requires some code to be executed by the processor as an overhead and resulting in a break in the flow of data processing.

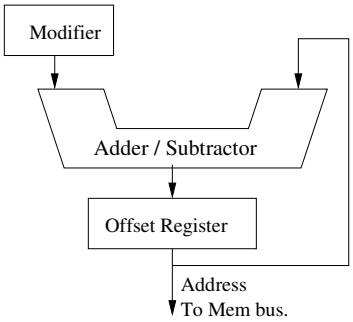

Fig. 1. Address generation scheme

Some of the most often used addressing sequences are *Sequential, Sequential with offset, Shuffled, Bit-reversed, Reflected.* Hulina et al. [1] discusses implementation of Coprocessor for generation of these address sequences and provides the host processor with few additional special addressing modes defined by signal processing algorithms, without any change in the host processor's instruction set architecture nor the external memory.

Efficient generation of Address sequences like *Zig-zag* for DCT, sequence of addresses to fetch the twiddle factors in FFT operation, and sequence of addresses to fetch the data in convolution operation are essential for Multimedia Processing.

This paper describes an Address Generation Unit suitable for a Dynamically Reconfigurable Datapath Processor which is capable of generating address sequences that can generate next address in one clock and can be synchronized with the datapath operations by using the *Address Generate Enable* signal. The following address sequences suitable for multimedia applications are supported:

- Bit-reversed: for data fetch and data store in case of a complete N-point FFT kernel.
- Fetching twiddle factors for a complete N-point FFT kernel.
- Data fetch operation for a Convolution kernel
- Zig-zag: suitable for fetching data for entropy coding.
- Other modes like linear, modulo N (circular), divide-by-N etc.

978-1-4244-4837-1/09/$25.00 ©2009 IEEE

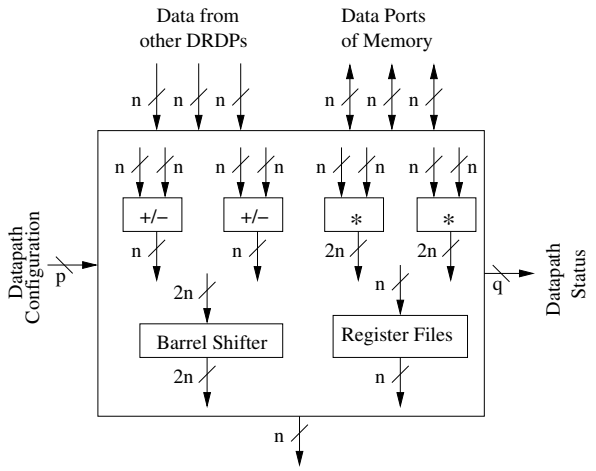

Fig. 2. Schematic of a DRDP

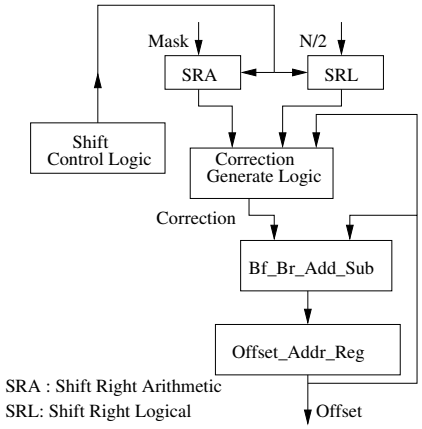

Fig. 3. Hardware schematic of bit-reversed address generator

The scheme of generation of sequences of addresses is as shown in Fig. 1. The next address in the sequence is generated by adding or subtracting a modifier to the current address. The resultant will be treated as an effective address where the required data sample/coefficient needs to be accessed.

Section II describes algorithms developed and the corresponding hardware schematic for some of the addressing modes listed above. It deals with AGUs for data access and twiddle factor fetch suitable for a complete N-point FFT kernel; implementation of N-point FFT kernel using Dynamically Reconfigurable Data Path (DRDP), AGUs for data, coefficient access for a complete Convolution kernel and implementation of kernel; and AGU for accessing data from *N*x*N* pixel array in a zig-zag order used in entropy coding after DCT. Section III discusses the results obtained.

II. Address Generation Unit

The AGU discussed here is tailored to work with a DRDP, though the concept can be used with any datapath unit with appropriate synchronization. AGU and DRDP are designed as parameterizable word length and address size using a Hardware Description Language in fully Structural style of coding. Thus the hardware synthesized will be identical to the description in the code. AGU and the DRDP provide necessary status signals back to the controller and can be controlled by a microprogrammed controller; hence reconfigurability can be achieved easily with change of the microprogram.

A. Overview of the Dynamically Reconfigurable Datapath

The DRDP under discussion supports signed integer and fixed point arithmetic; has two Adder/Subtractors, two Multipliers, sixteen Registers, a Barrel Shifter and status bits of various functional elements. The output of any of these functional units can be routed to at least one input of all the functional units. MAC units can also be formed with a register with guard bits. The DRDP unit will have three input ports and a output port other than two memory read ports and a memory write port. Schematic of a DRDP is shown in Fig. 2. The configuration of a datapath is defined by a control word that is stored in a configuration memory. This can be considered as a microinstruction. A sequence of such control words or microinstructions forms a microprogram, addressed by a microprogram counter. The sequencing of execution of microinstructions is controlled by state of conditional flags through a microprogram control unit. A datapath suitable to execute a signal processing kernel is formed by writing microprogram. Microprograms for various kernels are stored in the microprogram memory. By changing the base address of the microprogram we can switch between kernels in a single clock cycle.

The DRDPs can be cascaded by interconnecting the Input/Outputs (IO) appropriately to form a complex datapath with kernel operations spread over multiple DRDPs in a chained or pipelined fashion.

B. Address Generation for N-point FFT Kernel

1) Bit Reversed Address Generation for N-point FFT: In radix-2 FFT algorithms, the input or the output samples need to be re-ordered in bit-reverse fashion depending on whether Decimation-In-Time (DIT) or Decimation-In-Frequency (DIF) approach is employed. The samples are recursively partitioned into sub-partitions of even and odd samples within a sub-partition. The sequence of indices of the sub-partition accessed for processing or for storage of processed sample, resembles bit-reversed order of addresses of memory locations where the original sequence of samples are stored.

For example, consider a 8-point DIF FFT being computed on samples of data are stored in locations {0, 1, ...6, 7}. The pairs of data samples used for first stage of FFT computation are {0, 4}, {2, 6}, and {1, 5}, {3, 7}, the second stage FFT computation needs data sample pairs as {{0, 2}, {4, 6}} and {{1, 3}, {5, 7}}. For the third stage, the data access will contain sample pairs as {{0, 1}, {2, 3}, {4, 5}, {6, 7}}. As in-place computation is being used, within a given stage, it is the pairs of the data samples that are important and not the sequence of pairs of data.

Address generation for the first stage is simple, and can be

achieved by using a binary adder with propagation of carry in the reverse order i.e. from most significant bit towards least significant bit. For a N-point FFT address generation, the current address is to be added with $N/2$ using a reverse carry propagation adder. This does not hold good for the other stages. The modifier needs correction at the end of address generation of a sub-partition. A novel algorithm for generating Bit-Reversed Addresses for any N-point FFT with $log_2 N$ stages has been developed, hardware designed, simulated, tested and the block schematic is as shown in Fig. 3. The algorithm can be summarized as follows:

Reset: Reset SRA; Reset SRL;
Reset Offset Address Register;
Init: Load SRA with Mask; Load SRL with N/2;
Begin: If Bitwise_OR(SRA out, Cur_Offset)==All Ones
Then Correction = Bitwise_OR(N/2, SRL out)
Else Correction = SRL out;
If Bitwise_OR(Mask, Cur_Offset)==All Ones
Then {Enable Shift SRA, SRL;
Reset Offset Address Register;
}
Offset=Bit_Rev_Add(Cur_Offset, Correction);
Goto Begin;.

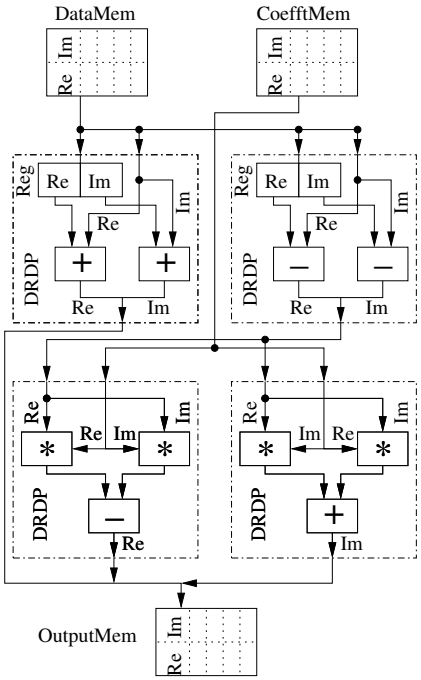

Fig. 4. Datapath used for DIF based FFT

Generation of addresses can be terminated on generating addresses for *log2(N)* stages of FFT; for which SRL content becoming zero is one of the indicators.

Many algorithms to compute bit-reversed address are available in literature. Many of them are best suited for coding using high level languages on microprocessor or digital signal processor [2], [3], [4], [5]. These algorithms can be classified as those based on heuristic [2] and algorithms using Seed-Table [3]. Address generation using these methods have a long delay as compared to the data-path latency and the memory access delay.

Hardware Address Generation Units (AGU) have been developed for array processors [6]. Nwachukwu [6], Hulina [1] implement the Bit-Reversed address generation using Counter-Multiplexer method. Counter-Multiplexer method can generate variety of patterns, but as the number of addresses increase, area increases exponentially and also results in increase in power dissipation and leakage. The method proposed in this paper can generate sequence of addresses suitable for many of multimedia algorithms; using adders, shifters, counters in the datapath; and very few gates and flipflops for the simple control logic. This translates to a linear increase in transistor count with increase in the number of address bits unlike counter-multiplexer method. Banerjee et al. [7] describe an algorithm for address generation for data access for N-point FFT. The hardware developed for implementing the algorithm uses 3 loadable down counters and allied control circuit. Hardware developed by us for the same functionality needs only 2 shift registers and allied control circuit as depicted in Fig. 3. The shifters hold data patterns like '11..1100..00' and '00..00100..00', and are shifted once after completion of each stage of FFT, and only 2 bits toggle in each of the shifters as compared to multiple bits toggling in each of the counters after every address generation as in Banerjee et al. [7]

FFT kernel can be executed in a single DRDP in a folded manner or can be executed in four DRDPs in a chained manner. A typical datapath for DIF is as shown in Fig. 4. In each of these datapaths 4 DRDP units have been cascaded and configured to form a single datapath capable of performing one butterfly computation for every two clocks so that N-point FFT operation is completed in $Nlog_2 N+4$ clock cycles where a constant of 4 clock cycles corresponds to initialization of the DRDP and write back of the last butterfly result. The setup assumes that the twiddle factors are precomputed and saved in memory.

The input or the output samples need to be re-ordered in bit-reverse fashion depending on whether DIT or DIF approach is employed. This reordering is done by exchanging data in memory locations pointed by pairs of address generated by Bit-reversed address generator for first stage. For the actual exchange the data elements of the pair are fetched from memory, stored in registers of the DRDP and written back in exchanged order from registers. For fetching the data and writing it back we use two AGUs appropriately synchronized.

Discussion on reducing the number of memory accesses for mapping the data array in bit-reversed order by avoiding self-reversed binary address patterns is given in [8]. This concept is useful in reducing the memory accesses required during pre or post processing of data in terms of reordering the data samples while computing FFT using DIT or DIF schemes. This is realized by comparing the bit-reversed address generated by AGU with an internal linear up counter. If the counters match,

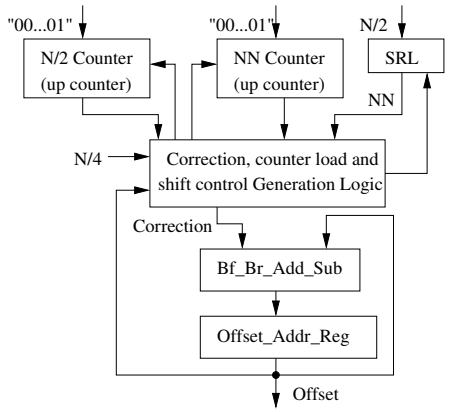

Fig. 5. Hardware schematic of FFT twiddle factor address generator

exchange is not required, hence no memory read or write operations; the AGU will continue to generate the next address in the sequence.

2) Address Generation for accessing Twiddle factors for N-point FFT: For a FFT butterfly computation, a pair of data operands with bit-reverse order address and a twiddle factor are needed.

An algorithm for generating sequence of addresses for fetching twiddle factors for any N-point FFT with $log_2(N)$ stages has been developed, hardware designed, simulated, tested and the block schematic is as shown in Fig. 5. The algorithm can be summarized as follows:

Reset: Reset Nby2 counter; Reset NN counter;
Reset Offset Address Register;
Init: Load Nby2 counter = 1; Load NN counter = 1;
Load SRL with N/2;
Begin: If (NN counter==NN and Nby2 counter==N/2)
Then shift SRL;
If (NN counter == NN)
Then load NN counter = 1;
If (Nby2 counter == N/2)
Then load Nby2 counter = 1;
If (Correction_select) Then
{Correction=Cur_offset;
Offset=Cur_offset-Correction; }
Else {Correction=N/4;
Offset=Cur_Offset+Correction;}
Goto Begin;

Correction_select()
{
(NN counter==NN) OR (NN_0) OR
(NN_1 AND (Nby2 counter==N/2)
AND (NN counter==NN));
}.

C. Address Generators for Convolution Kernel

When an input sequence of length N is convolved with an impulse response of length M, the output sequence is of length $N + M - 1$. The address generation scheme assumes that the given data is padded with $M - 1$ zeroes at both ends. For example we have 3 data points x(n) with pad data, 2 sample points of impulse response h(n); resulting in 4 output values y(n). The sample points of impulse response are stored in a reverse order in the memory. Table I summarizes the sequence of addresses to be generated for fetching the data, coefficients and storing the result.

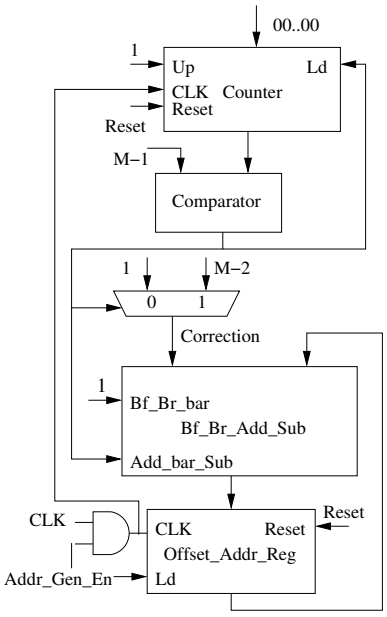

Fig. 6. Hardware schematic of AGU for data fetch of convolution kernel

TABLE I
SEQUENCE OF ADDRESS GENERATED FOR CONVOLUTION KERNEL

Address sequence			Corresponding values		
x(n)	h(n)	y(n)	x(n)	h(n)	y(n)
0, 1	0, 1	0	x_{-1}, x_0	h_1, h_0	y_0
1, 2	0, 1	1	x_0, x_1	h_1, h_0	y_1
2, 3	0, 1	2	x_1, x_2	h_1, h_0	y_2
3, 4	0, 1	3	x_2, x_3	h_1, h_0	y_3

The sequence of addresses for fetching coefficients follows a *Modulo-M* pattern and that for writing the convolution result *Divide-by-M*.

1) Address Generator for fetching data for convolution: The algorithm for generating the address can be summarized as follows:

Reset: Reset Counter; Reset Offset Address Register;
Begin: If Counter = $M - 1$
Then Correction = $-(M - 2)$; Counter = 0;
Else Correction = 1;

Offset=Current_Offset + Correction;
Goto Begin;.

The hardware schematic for the above AGU is shown in Fig. 6.

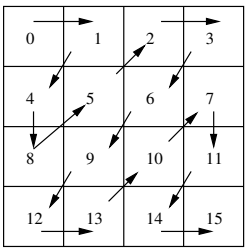

Fig. 7. Sequence of addresses generated in zig-zag addressing mode.

2) Address Generator for accessing filter coefficients in convolution: The algorithm for generating the addresses for fetching coefficients is of Modulo *N* type and is similar to that of data fetch for convolution except the correction is $-(N-1)$ whenever the counter reaches a value of $N-1$.

3) Address Generator for accessing convolved data write-back: The algorithm for generating the addresses for result storage is of Divide by *N* type and is similar to that of coefficient fetch for convolution except the correction is '0' when the counter value is less than $N-1$ and correction is '1' when counter value is equal to $N-1$.

D. Zig-zag Address Generation for accessing $N \times N$ pixel array

JPEG uses Entropy coding for compressing the data after performing DCT and Quantization. After computing the 2D - DCT of a $N \times N$ image, it can be seen that the significant coefficients are present in top-left corner of 2D matrix. For compressing the coefficients further, it is necessary to process only these coefficients.

Entropy coding requires the quantized data of *N*x*N* pixel array to be read in zig-zag fashion as shown by the sequence of arrows in Fig. 7. An algorithm has been developed to generate this address sequence for any value of *N*x*N* (*N* being even), hardware developed, simulated, tested and the block schematic is as shown in Fig. 8. The algorithm can be summarized as follows:

Reset: Reset Row counter; Reset Column counter;
Reset Offset Address Register;
Begin: If ((!cond1) AND (!cond2)) Then Correction=N-1;
If ((!cond1) AND (cond2)) Then Correction=N;
If (cond1) Then Correction=1;
If (cond3) Then Offset=Cur_Offset-Correction;
Else Offset=Cur_Offset+Correction;
If (cond4) Then Column counter up enable;
If (cond5) Then Column counter down enable;
If (cond6) Then Row counter up enable;
If (cond7) Then Row counter down enable;
Goto Begin;.

cond1 to *cond7* check if the counter value pair has reached the boundary of the pixel array map and hence a change of direction in scanning is required.

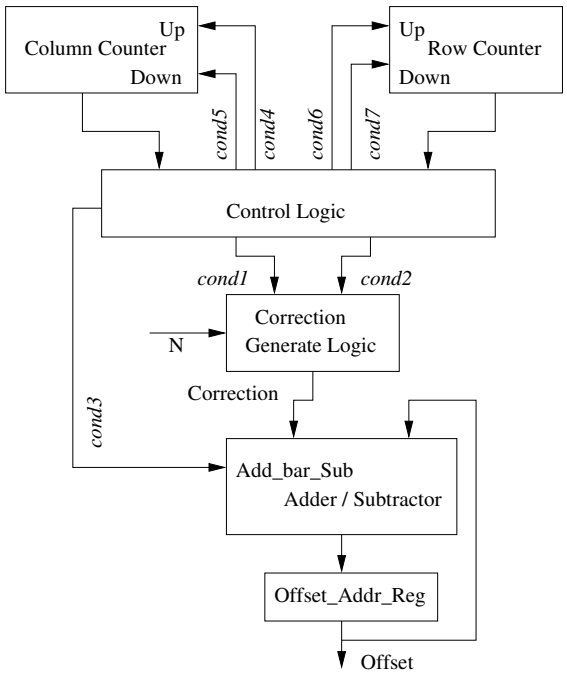

Fig. 8. Block schematic of hardware used for zig-zag address generation

III. Results and Conclusion

Efficient Address Generation Algorithms and hardware suitable for speeding up execution of DSP kernels using DRDPs have been developed. A novel algorithm for Bit Reversed Address Generation for N-point FFT capable of generating address sequence for Butterfly computations over $Nlog_2N$ stages in $Nlog_2N+4$ clock cycles has been developed. Algorithm for fetching the FFT twiddle factors in synchronization with the data fetch has also been developed. Sample simulation result of a 8-point FFT kernel with DIF approach, implemented on datapath using 4 DRDPs being chained; completes the operation in 28 clock cycles as shown in Fig. 9. Convolution kernel has been implemented using a single DRDP and Fig. 10 shows the simulation results of 5 data points convolved with impulse response of length 4; operation being completed in $((N + M - 1) \times M) + 3$ clock cycles. These prove the efficacy of the AGUs developed, the ability to synchronize the data access and computation of result using the DRDPs in case of a 8-point DIF FFT kernel and Convolution kernel. Efficient algorithm and hardware for address generation to fetch data in a zig-zag sequence as required by Entropy coding after DCT operation is also developed.

The algorithms have been implemented in VHDL in a fully structured coding style. The data width and the address width are parameterizable. The coding completely adheres to structural style and the algorithm is using components that scale linearly in terms of complexity of number of transistors in the hardware. All the algorithms have been simulated and tested. Due to a structured approach and shared common components in the AGU for various addressing modes, a microprogrammed controller that can handle various addressing

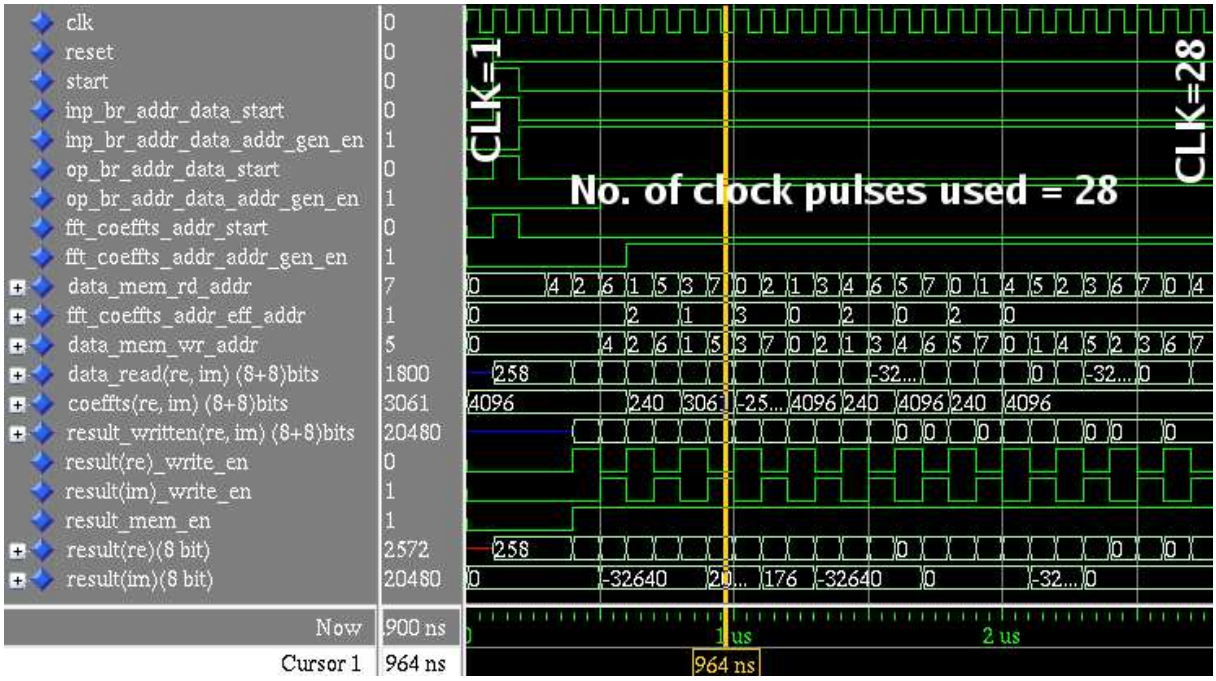

Fig. 9. Simulation result of a 8-point FFT kernel

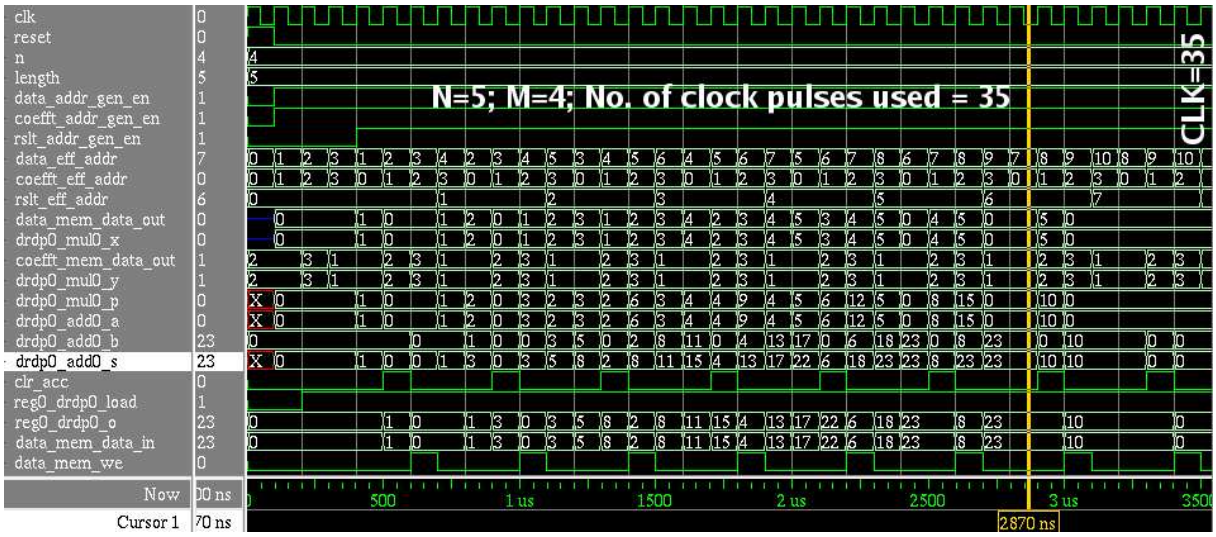

Fig. 10. Simulation result of a convolution kernel

modes required by DSP kernels can be implemented; leading to a Reconfigurable AGU.

REFERENCES

[1] Hulina P T, Coraor L D, Kurien L, and John E, "Design and VLSI Implementation of an Address Generation Coprocessor," *IEE Proceedings on Computers and Digital Techniques* vol. 142, no. 2, pp. 145-151, March 1995.

[2] Angelo A. Yong, "A Better FFT Bit-Reversal Algorithm Without Tables," *IEEE Transactions on Signal Processing*, vol. 39, no. 10, pp. 2365-2367, October 1991.

[3] James S Walker, "A New Bit Reversal Algorithm," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38. no. 8, pp: 1472-1473, August 1990.

[4] Evans David M W, "A Second Improved Digit-Reversal Permutation Algorithm for Fast Transforms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37. no. 8, pp: 1288-1291, August 1989.

[5] Jeffrey J Rodriguez, "An Improved FFT Digit-Reversal Algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37. no. 8, pp: 1298-1300, August 1989.

[6] E O Nwachukwu, "Address Generation in an Array Processor," *IEEE Transactions on Computers*, vol. c-34, no. 2, pp: 170-173, February 1985.

[7] Ayan Banerjee, Anindya Sundar Dhar, and Swapna Banerjee, "FPGA realization of a CORDIC based FFT processor for biomedical signal processing," *Elsevier Science - Microprocessors and Microsystems*, vol. 25, pp: 131-142, February, 2001

[8] Thomas R Harley and G P Maheshwaramurthy, "Address Generators for Mapping Arrays in Bit-Reversed Order," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp: 1693-1703, June 2004.