Address Generation for DSP Kernels

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

***Abstract*—Performance of Signal Processing Algorithms implemented in hardware depend on efficiency of datapath, memory speed, and address computation. Pattern of data access in signal processing applications is complex and it is desirable to execute the innermost loop of a kernel every clock. This demands generation of typically three addresses per clock: two addresses for data sample/coefficient and one for storage of processed data. Presence of a set of dedicated, efficient Address Generator Units (AGU) helps in better utilization of the datapath elements by using them only for *kernel operations*; and will certainly enhance the performance. This paper focuses on design and implementation of Comprehensive Address Generator Unit (CAGU) for complex addressing modes required by DSP Kernels used in Multimedia Signal Processing. An 8 bit CAGU has been implemented using UMC 0.18 micron, 6 metal layers process, that occupies 21802 sq microns, consuming 2.95 mW and works with a clock period of 6 ns.**

***Index Terms*—Address generation, Bit-reversed address, Dynamically Reconfigurable Datapath, Fast Fourier Transform, Sum of Absolute Difference, Zig-zag address generation.**

I. Introduction

DSP implementation is driven by 3 major goals: data parallelism, application specific specialization and functional flexibility. These have led to a variety of hardware implementations [1]. Performance of a signal processing system can be measured using parameters like speed of execution or throughput, energy consumed in performing the task, flexibility in terms of programmability or modifying the system to perform any other function and cost. Various reconfigurable approaches, hardware implementation issues, types of coupling in a reconfigurable system, partitioning between hardware and software, context memories and reconfiguration issues are discussed in literature [2].

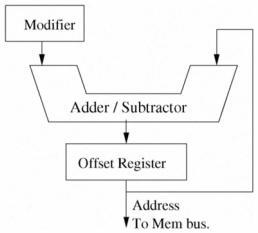

Fig. 1. Address Generation Scheme

In signal processing applications, data access can be characterized as indexed access of vectors stored in memory; referred to as pointers in higher level languages. These indices are data dependent and the sequence of access depends on the kernel operation being performed on the data. *Stride* can be defined as the distance between the addresses of consecutively accessed vectors in the memory. Strides can be linear or non linear and can be characterized by an algebraic equation. Thus the address of the next location to be accessed can be expressed as an algebraic expression in terms of the current address and viewed as a addition of a modifier to the current address. This algebraic expression can be synthesized into an *Address Generation Unit* using arithmetic operators and a controller. The scheme of generation of sequences of addresses is shown in Fig. 1.

In signal processing ASICs, Programmable Digital Signal Processors (PDSP) and present generation General Purpose Processor (GPP), address generation is performed by dedicated AGUs; are application specific, support few special addressing modes and support only trivial addressing modes respectively. For other addressing modes, datapath elements/instructions are used, resulting in sharing of datapath for kernel execution and address generation and hence non optimal use of datapath. In case of PDSP, with higher onchip memory bandwidth, time-budget for address generation shrinks. Types of sequences of address to be generated is very wide and complex for DSP kernels. Moreover these sequences also depend on factors like architecture of processor (pipelined/non-pipelined etc.), memory (stored/streaming data, interleaved etc.) and the kernel (Fast Fourier Transform (FFT)/Sum of Absolute Difference (SAD)/Entropy computation 1D/2D computation etc.)

Section II describes algorithms and hardware developed for various AGUs; Section III discusses the performance of these AGUs, development of a Comprehensive AGU, ASIC implementation and performance of CAGU.

II. Address Generators for Dynamically Reconfigurable Processors for DSP Kernels

Address Generation Units suitable for a Dynamically Reconfigurable Data Path (DRDP); capable of generating one address per clock in required sequence and synchronized with the datapath operations is discussed in this section. The following addressing modes 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.

978-1-4244-9799-7/11/$26.00 ©2011 IEEE

- data fetch operation for a Convolution kernel (Stored data - any values N and M)
- data fetch operation for a Convolution kernel (Streaming data - any values N and M)
- impulse response coefficient fetch operation for a Convolution kernel (Stored/Streaming data) - Modulo M (circular) addressing mode.
- result store operation for a Convolution kernel (Stored/Streaming data) - Divide by N addressing mode.
- data fetch operation for a Linear Phase FIR filter (Streaming data).
- data fetch operation from macroblocks for SAD computation in Motion Estimation kernel.
- Zigzag - suitable for fetching data for entropy coding.
- other modes like increment and decrement etc.

A. Address Generation for 'N' point FFT Kernel [3]

1) Bit Reversed Address Generation for 'N' point FFT: 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 [4], [5], [6], [7]. These algorithms can be classified as those based on heuristic [4] and algorithms using Seed-Table [5]. 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 [8]. Nwachukwu [8], Hulina [9] 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 can generate sequence of addresses suitable for many of multimedia algorithms; use adders, shifters, counters in the datapath; and very few gates for the simple control logic [3]. This translates to a linear increase in transistor count with increase in the number of address bits unlike counter-multiplexer method. Banerjee et al. [10] describe an algorithm for address generation for data access for N point FFT. Proposed architecture uses 3 loadable down counters and associated control circuit. Our architecture for the same functionality needs only 2 shift registers and allied control circuit [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 [10]. N point FFT operation is completed in $Nlog_2(N)+4$ clock cycles [3]. The setup assumes that the twiddle factors are precomputed and saved in memory.

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

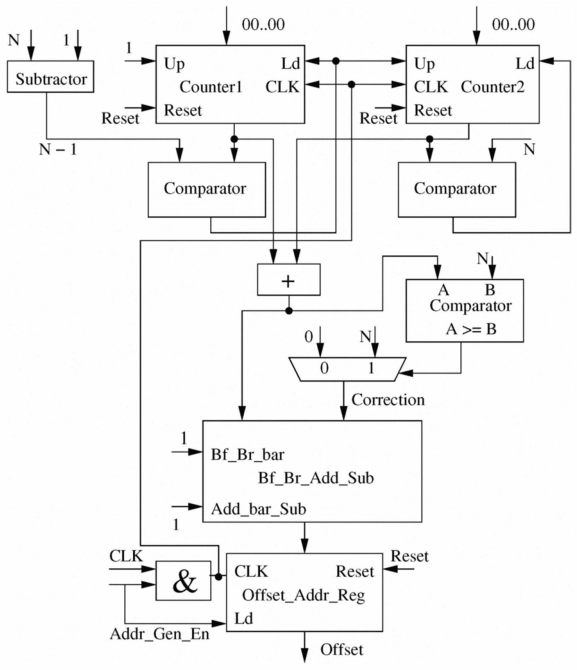

Fig. 2. Hardware Schematic of AGU for Data Fetch of Convolution Kernel - streaming data.

An algorithm for generating sequence of addresses for fetching twiddle factors for any N point FFT with log_2N stages has been designed, simulated, and tested [3].

B. Address Generators for Convolution Kernel

Convolution is used in implementing Digital Filters like Finite Impulse Response (FIR) filter. Linear Phase FIR filters are typically used where filter coefficients are symmetric or antisymmetric. When an input sequence of length N is convolved with an impulse response of length M, the output sequence is of length $N + M - 1$. Convolution is defined as

$$y_n = \sum_{k=0}^{M-1} x_k h_{n-k} \tag{1}$$

The address generation scheme assumes that the given data is padded with *M-1* zeroes at both ends and the sample points of impulse response are stored in a reverse order in the memory. The sequence of addresses for fetching coefficients follows a *Modulo M* pattern and that for writing the convolution result *Divide by M*. The convolution operation may be performed on stored data or streaming data. Each of these operations need a different type of AGU. AGU suitable for both applications have been developed. The algorithm for fetching the coefficients will remain the same irrespective of the kernel is working on stored or streaming data. The AGU used for storing the convolved data will use linear or circular addressing mode depending on whether the kernel is working on stored or streaming input.

1) AGU for data fetch for convolution kernel - streaming data: In case of streaming data, the data samples are stored in a circular buffer. An efficient algorithm and hardware has been

Reset: Reset Counter1, Counter2, Offset Register;
Begin: If $Counter1 == N-1$ Then
$Counter1 = 0; Counter2 = Counter2 + 1;$
Else $Counter1 = Counter1 + 1;$
If $Counter2 == N$ Then
$Counter2 = 0;$
If $(Counter1 + Counter2) < N;$ Then
$Correction = 0;$
Else $Correction = N;$
$Offset = (Counter1 + Counter2) - Correction;$
Goto Begin;

Fig. 3. Algorithm for Data Fetch of Convolution Kernel AGU - streaming data.

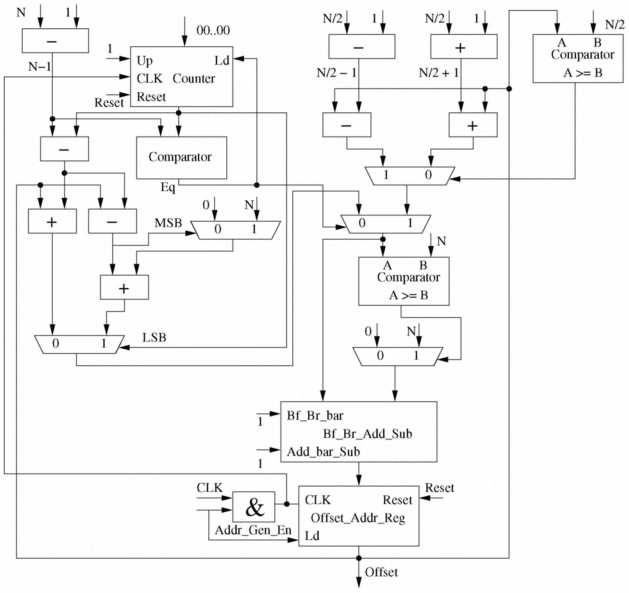

Fig. 4. Hardware Schematic of AGU for accessing data for a Linear Phase FIR Kernel.

developed and tested. The schematic of the AGU hardware is as shown in Fig. 2 and the algorithm for generating the address for fetching streaming data can be summarized as shown in Fig. 3.

C. Address Generator for accessing data for a Linear Phase FIR Kernel

Consider a case of a Linear phase FIR filter with even number of coefficients h_n and they are symmetric.
For example: let $N = 6$; then

$$h_0 = h_5, h_1 = h_4, h_2 = h_3. \tag{2}$$

Hence y_n can be written as

$$y_n = (x_n + x_{n-5})h_0 + (x_{n-1} + x_{n-4})h_1 + (x_{n-2} + x_{n-3})h_2 \tag{3}$$

A novel algorithm for generating appropriate sequence of addresses to fetch the data has been developed, implemented and tested. The schematic of the hardware is shown in Fig. 4.

Reset: Reset Counter, Offset Address Register;
Begin: If $Counter == N-1$ Then $Counter = 0;$
Else $Counter = Counter + 1;$
$s = (N-1) - Counter;$
If $(Counter)_0$ Then $t = |(Current\_Offset - s)|_N;$
Else $t = (Current\_Offset + s);$
If $Current\_Offset >= N/2$
Then $u = Current\_Offset - (N/2 - 1);$
Else $u = Current\_Offset + (N/2 + 1);$
If $Counter = N - 1$ Then $v = u$
Else $v = t;$
$Next\_Offset = |v|_N;$
Goto Begin;

Fig. 5. Algorithm for accessing data for a Linear Phase FIR Kernel AGU - streaming data.

and the corresponding algorithm can be summarized as in Fig. 5.

D. Address Generators for Motion Estimation using Block Matching Technique Kernel

Between adjacent frames of normal video, there will be little motion of the object. An object in the previous frame tends to get displaced/rotated in the next frame by a small amount. Slice is a consecutive series of smaller blocks called *Macro Blocks* (MB), a frame may contain multiple slices. If a macroblock in the given slice in current frame is compared with the corresponding macroblock or in the neighborhood of the corresponding macroblock in the prior frame the difference will normally be very small. The search area for macro block match is restricted to *search parameter p* pixels around the macro block in a slice in the previous frame. Searching for such a block for which the cost function is the least and within a prescribed limit; and computing the displacement of the block from previous to current frame is the goal of Motion Estimation (ME). This operation needs to be done for all blocks in the frame and is a computationally expensive operation; motion detection over longer distance need larger value of *search parameter* and results in exponential increase of computational complexity. Popular cost functions are Sum of Absolute Difference (SAD), Mean Absolute Difference (MAD) and Mean Square Error (MSE).

The datapath for computing SAD is as shown in Fig. 6, where N is the side of the macroblock; C_{ij} and R_{ij} are the pixels being compared in the current and the reference slice.

1) Address Generator for fetching data for Motion Estimation using Block Matching Technique: The AGU assumes that the data of the current and the reference slices are kept in the memory as rows; the width, height of the macroblock (mb_wd, mb_ht) and the width of slice (sl_wd) are given. The AGU uses 2 up-counters to maintain the row and column counts; block schematic of the hardware implementation and the corresponding algorithm is as shown in Fig. 7 and Fig. 8.

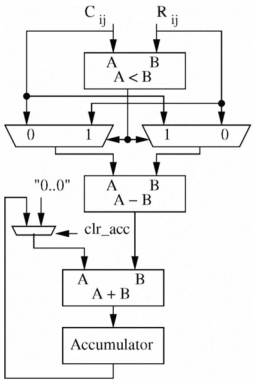

Fig. 6. Datapath for SAD computation.

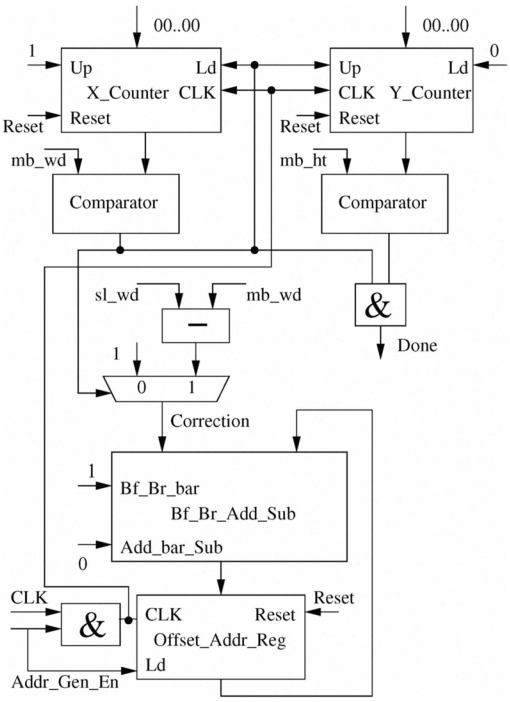

Fig. 7. Hardware Schematic of AGU for data fetch for motion estimation kernel.

E. Zigzag Address Generation for accessing $N \times N$ pixel array [3]

JPEG uses Entropy coding for compressing the data after performing DCT and Quantization. Entropy coding requires the quantized $N \times N$ pixel array to be read in Zigzag fashion. An algorithm has been developed to generate this address sequence for any value of NxN ('N' being even), hardware developed, simulated and tested [3].

III. Results

Efficient Address Generation Algorithms and hardware suitable for speeding up execution of DSP Kernels using DRDPs have been developed. Convolution kernel has been implemented using a single DRDP and convolution kernel operation is completed in $((N + M - 1) \times N) + 3$ clock cycles [3]. An AGU for Data fetch for Convolution for streaming data

```
Reset: Reset X_Counter;
       Reset Y_Counter; Reset Offset Address Register;
Begin: If X_Counter == mb_wd
       Then X_Counter=0;
            Y_Counter = Y_Counter + 1;
            Correction = sl_wd - mb_wd;
       Else X_Counter= X_Counter + 1
            Correction = 1
       Offset = Offset + Correction
       Goto Begin
```

Fig. 8. Algorithm for data fetch for motion estimation kernel.

has been developed; this along with convolution coefficient fetch AGU (Modulo M) and convolution result write AGU (Divide by M) are used to build and test the Convolution of streaming data Kernel. Data fetch AGU for Linear phase FIR with data being streamed has been developed; used with convolution coefficient fetch and result write AGUs to implement Linear Phase FIR Filter Kernel. An AGU for Data fetch for Motion Estimation using block matching technique has been developed; a Kernel to compute the SAD has been implemented and tested; the simulation results are shown in Fig. 9. AGU for Data fetch for Zigzag sequence used in case of entropy coding after DCT has been developed and tested [3]. FFT kernel, filtering using convolution kernel on stored data are designed and simulated [3]. These prove the efficacy of the AGUs developed, the ability to synchronize the data access and computation of result using the DRDPs. The algorithms have been implemented in VHDL in a fully structured coding style. The data width and the address width are parameterizable. Structured approach of the coding helps in identifying common components used across various addressing modes like counters, shifters, adder and comparators etc. Using Up/Down Counters cum Logical/Arithmetic Right Shifters, an Accumulator, Comparators and little glue logic, a configurable *Comprehensive Address Generator Unit* (CAGU) that can support multiple addressing modes listed earlier has been designed. Table I summarizes the timing details of simulation of execution of some DSP Kernels implemented using CAGU developed along with proposed DRDPs.

TABLE I
Timing details of various Kernel execcution

Kernel	Number of clocks	Overhead (Initialization and latency)
FFT N point	$Nlog_2N$	4
Convolution (i) Stored data (ii) Streaming data	 $(N+M-1)N$ $(N+M-1)N$	 3 3
Computation of SAD (for $N \times M$ macroblock)	$N \times M$	2

Since most of the kernels datapath would involve at least one MAC unit whose datapath delay is going to be considerable; the timing of the CAGU need not be very aggressive

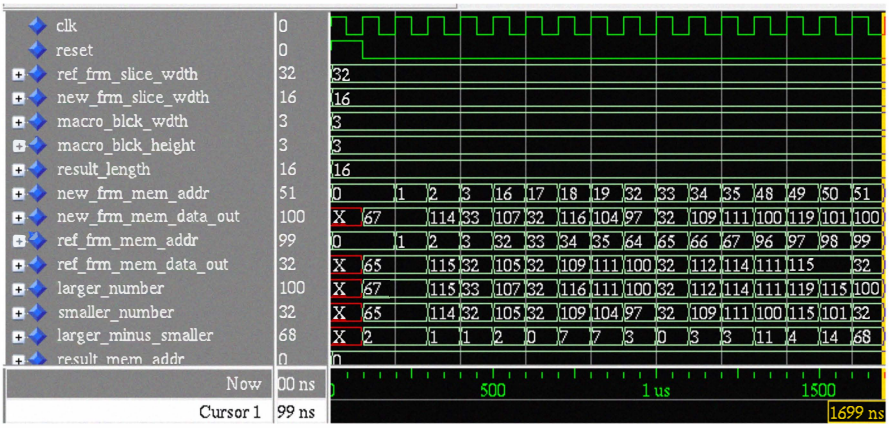

Fig. 9. Simulation result of Kernel to compute the SAD for Motion Estimation - streaming data

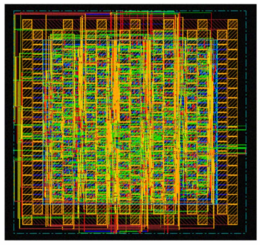

Fig. 10. Snapshot of the layout of a 8 bit wide CAGU

and a clock of 6 ns period was assumed; keeping this in mind, Ripple Carry Adders were used in the CAGU; they are slower but occupy less space. Standard cell library from Faraday based on UMC 0.18 micron with 6 metal layers process was used for implementation.

TABLE II
SYNTHESIS REPORT OF CAGU FOR VARIOUS ADDRESS WIDTHS.

Criteria/Address width (clk period)	8 bit (6ns)	16 bit (8ns)	24 bit (10ns)
Cells used	1016	1975	2948
Area in μ^2	21802	43019	64202
Leakage Power in nW	39.46	77.84	117.46
Dynamic Power in mW	2.95	5.43	8.65
Slack in ps	25	296	238

The CAGU was synthesized, placed and routed. The post synthesis report is summarized in Table II and demonstrates that the algorithm developed for the AGUs and design of the units scale linearly with address width. The snapshot of the layout is shown in Figure 10. It was observed that for a given address width, reduction in clock frequency results in reduction in both area and power.

The functional verification was carried out on the extracted netlist and the CAGU is functioning as desired. The design has been sent for fabrication with *Europractice* and the chip will be tested upon arrival.

The entire implementation process was carried out using Cadence Tool suite; functional simulation and post extraction behavioral simulation were carried out with ModelSim tool.

IV. CONCLUSION

The concept proposed demonstrates the utility of dedicated AGUs, and proves the following: (i) Computation of one address per clock per AGU, (ii) Comprehensive AGU can be configured to generate address sequence over the entire DSP Kernel without any intervention of any kind during the execution of the kernel, (iii) With the help of these AGUs and suitable reconfigurable datapath the innermost loop of the chosen DSP Kernel can be executed in one clock period. (iv) Algorithm and hardware designed scale linearly in terms of complexity of gates with address width.

REFERENCES

[1] Russell Tessier and Wayne Burleson, "Reconfigurable Computing for Digital Signal Processing: A Survey," *The Journal of VLSI Signal Processing*, Springer Netherlands, Vol. 28, No. 1-2, pp: 7-77, 2001.

[2] K. Compton and S. Hauck, "Reconfigurable Computing: A Survey of Systems and Software", *ACM Computing Surveys*, Vol. 34, No. 2, June 2002, pp. 171-210.

[3] Ramesh Kini. M. and Sumam David S, "Comprehensive Address Generator for Digital Signal Processing," *Proceedings of International Conference on Industrial and Information Systems (ICIIS 2009), SriLanka, 28-31 December 2009.* pp:325-330, 2009.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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, 2001, pp: 131-142.