www.icgst.com PDCS

ASIC Implementation of Address Generation Unit for Digital Signal Processing Kernel-Processor

Ramesh Kini. M, Sumam David
Department of Electronics and Communication Engineering, National Institute of Technology Karnataka, Surathkal, Mangalore, India – 575025.
rameshkinim@gmail.com, sumam@ieee.org

Abstract

Multimedia applications are written in high level languages and are implemented on Reconfigurable-Processors (RP) through High Level Synthesis. Such implementation style tends to use the same data-path elements for both Address Generation and Kernel processing. This results in inconsistent and non-optimal use of data-path elements. This paper discusses design and implementation of dedicated Address Generation Unit (AGU) capable of generating complex sequences of addresses required by typical Multimedia Digital Signal Processing (DSP) Kernels like Fast Fourier Transform (FFT), Convolution, Linear Phase Finite-Impulse-Response (FIR) Filters and Sum-of-Absolute-Difference (SAD) computation for Motion Estimation (ME). Use of such a dedicated AGU with the proposed Reconfigurable-Data-path separates the data-paths of address computation and Kernel computation and paves way for efficient high level synthesis. The AGU designed by us helps to realize execution of one innermost loop of a DSP kernel every clock with minimal hardware. A comprehensive AGU (CAGU) that can be configured to generate sequences of addresses of data stored in memory for the computation of DSP kernels listed earlier has been implemented as an ASIC using 0.18μ, 6 metal, single poly technology from UMC.

Keywords: *Address generation, Bit-reversed address, Convolution, Digital Signal Processing Kernel, Dynamically Reconfigurable Data-path, Fast Fourier Transform (FFT), Finite Impulse Response (FIR) Filters, Motion Estimation, Zig-zag address generation.*

1. Introduction

Offline data processing or stream data processing of large number of data points, as in the case of multimedia applications, requires data to be accessed from memory at high rates. The sequence of this data access is non-linear, complex and varies with the type of signal processing kernel that is being executed. Observations made during this research show that using kernel execution data-path for address computation leads to ineffective utilization of resources. Hence it is desirable to have a dedicated Comprehensive AGU (CAGU). This paper deals with the design and implementation of such a CAGU as an Application Specific Integrated Circuit (ASIC).

Study of topics like Reconfigurability, Synthesis, Placement, Routing strategies, Configuration loading and management etc. for reconfigurable devices and the reflection of the same on design of reconfigurable devices is described in [1]. Algebraic operators basically have regular structures; coarse grained architectures can provide configurable word length operators with area efficient data-paths created with programmable interconnects [2].

Concept of Reconfigurable Processor (RP) comes from the idea of having a general purpose processor coupled with some reconfigurable resources that allow execution of custom application specific instructions. Dynamically Reconfigurable Processor (DRP) achieves higher speed of computation with lower cost of silicon area for computationally intensive applications in the domains like multimedia.

Section 2 deals with design issues that highlight the need for a CAGU and provides an overview of the setup in which the designed CAGU can be used. Section 3 describes the algorithms and hardware developed for AGUs supporting data/coefficient fetches and result updates. It also discusses 3 main DSP kernel implementations namely FFT, Convolution and SAD computation used in motion estimation using the AGUs developed. Integration of various AGUs into CAGU, simulation results of various kernels, ASIC implementation of the CAGU and verification results are discussed in Section 4.

2. Overview of the proposed DRP

RPs find extensive applications in networking and multimedia domains. Multimedia applications like Audio/Video Encoders/Decoders, Trans-coders, FFT in OFDM used in Software Defined Radios etc can exploit the features of a RP to minimize the hardware requirements and at the same time improve the throughput.

The proposed processor has an array of four dynamically Reconfigurable Functional Units (RFU) sharing a common memory. The array of RFUs and the memory are controlled by an Application Specific Instruction set

Processor (ASIP). The RFU array with a control processor can also be mapped onto a FPGA based embedded system for multimedia applications.

The overall operation of the proposed processor can be summarized as follows: Upon receiving a service request for executing a kernel operation, the ASIP controller facilitates the storage of data to be processed in appropriate memory block, initializes a idle RFU with appropriate control settings and schedules the kernel on that RFU. Upon completion of kernel operation controller frees the RFU for executing any other kernel. Basically ASIP coordinates the job of kernel execution. Advantages of the architecture are:

- The innermost loop of a kernel can be executed in one clock and hence is faster compared to execution on a GPP or DSP processor.
- Since the kernel itself is implemented as a micro-program, there is no need for the code (program) memory; no instruction fetches and decodes. This results in reduction of memory and power requirements.
- The application can be written in terms of function calls and these function calls can be mapped to kernels executed on the RFU by the ASIP. As a result application program tends to be compact.

RFU contains a Dynamically Reconfigurable Data Path (DRDP), 3 AGUs for generation of memory addresses and Configuration Memory with multiple contexts as shown in Figure 1. The DRDP consists of 2 Adders (Adder/Subtractor), 2 Multipliers, a Comparator, Barrel Shifter and 4 Register files for temporary storage. The DRDP under discussion supports signed integer and fixed point arithmetic. The output of any of these functional units can be routed to at least one input of all the functional units. The DRDP unit will have three input ports and an output port other than two memory read ports and a memory write port. The configuration of a data-path is defined by a control word that is stored in a configuration memory. The DRDPs can be cascaded by interconnecting the Inputs/Outputs (IO) appropriately to form a complex data-path with kernel operations spread over multiple DRDPs in a chained or pipelined fashion. The processor design is targeted at acceleration of execution of these often used kernels under the control of the ASIP.

Goal of executing the innermost loop of any of the selected kernel in one clock with few finite arithmetic units is satisfied if the address generation units are separated from data-path of the kernel. This leads to use of dedicated AGUs.

This is also supported by the observation made in [3] that in case of application implementation through high level synthesis, inclusion of AGU minimizes the circuit complexity and helps achieving higher speeds of execution.

HDL code in structured style has been developed, tested for each of the address update functions required by data fetch, coefficient fetch and result-write operations of each kernel, such that one address is generated per clock in the required sequence. Common functional units used in the address generation process for these kernels were identified; and a Comprehensive AGU that can support various addressing modes with a shared data-path elements and control structure was developed.

Though Coarse grain reconfigurable architectures provide significant speedups, ability to compile from imperative high level languages like ‘C’, Java®, Matlab® etc to achieve noticeable speedup is not proved due to reasons like reduced focus on compilation phase and mapping computational structures effectively on reconfigurable architectures [4]. Mapping of innermost loop to a dynamically reconfigurable data-path coprocessor is discussed in [5].

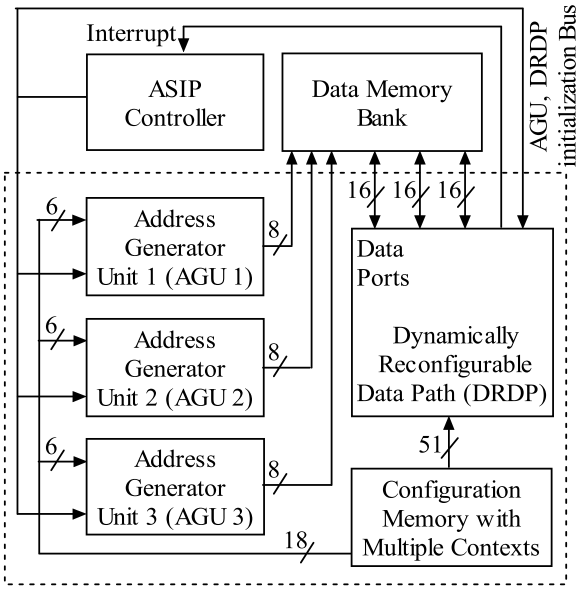
Interrupt
ASIP Controller
Data Memory Bank
AGU, DRDP initialization Bus
6
Address Generator Unit 1 (AGU 1)
8
16
16
16
Data Ports
6
Address Generator Unit 2 (AGU 2)
8
Dynamically Reconfigurable Data Path (DRDP)
51
6
Address Generator Unit 3 (AGU 3)
8
Configuration Memory with Multiple Contexts
18

Figure 1. Block schematic of Reconfigurable Functional Unit.

System described in this paper can map a high-level DSP kernel with recurring loop like N-point FFT, FIR filter implementations and others onto the dynamically reconfigurable data-path easily. The CAGU supports address generation across multiple levels of recursive loops for these kernels without changing the data-path.

To map a multimedia application to most of the reconfigurable processors developed till now, a specialized compiler or a high level synthesis tool is required. This research targets at reducing the complexity of mapping by designing a coarse grained architecture; such that the proposed design aids in mapping the multimedia applications more effectively to the proposed architecture. The multimedia applications can be written in any imperative high level language using two kinds of function calls. The first type being basic multimedia file header processing and system IO handling, these are processed by the controller processor. The second type of function calls being the DSP kernels that can be mapped directly on to the RFUs, can be executed in an accelerated fashion. This eliminates the need for re-synthesis and special efforts required in writing compilers/applications with effective design space explored to suit the reconfigurable hardware. The interaction between the Controller processor and the RFUs has been described earlier in this article. A set of dedicated, efficient Address Generator Units (AGU) will definitely enhance the performance. Next section discusses the AGU in detail.

3. CAGU for DRP for DSP Kernels

In signal processing applications, data access can be characterized as indexed access of vectors stored in memory. These indices are 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. The strides could 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 can be viewed as an addition of a modifier to the current address. Thus address generation process can be seen as a sequence of algebraic operations performed on current address and this algebraic expression can be synthesized into an Address Generation Unit using arithmetic operators and a controller.

Extracting the Address Expression (AE), applying high level optimizing methods like address expression splitting/clustering, induction variable analysis, target architecture selection and global-scope algebraic optimization are explored in [6]; it also aims at reduction of cost of time-multiplexed address unit at system level. This approach is more suited for high level synthesis of typical multimedia applications.

In a signal processing ASIC, AGU caters to generation of a pre-determined type of sequence of addresses. In case of GPP, the AGU may support a few simple sequence types or addressing modes. A Programmable Digital Signal Processor (PDSP) will have dedicated AGUs that support few additional addressing modes like bit-reversed, circular etc. In PDSPs, AGUs have their own dedicated computational logic and do not use the data-path resources leading to concurrency of address generation and data-path operations. If any other sequence of addresses is required, then it needs to be generated by execution of processor code and use data-path computational units, resulting in non-optimal use of resources like data-path elements and time.

Signal processing applications require varied types of addressing modes. For a given application, each type of data access may need a different type of addressing mode and these depend on the architecture and data-path of the processing system, the type of memory used and the way the data is stored in the memory. For example FFT computation needs the data to be fetched in bit-reversed order, the twiddle-factors in a linear fashion and the result to be written back in a bit-reversed order; FIR filter implementation of stored data requires the data to be fetched in a linear fashion, but the same filter processing streaming data needs to fetch the data from a circular buffer and needs circular addressing mode.

Most of the signal processing applications can be written in terms of DSP kernels and the kernels themselves could be function calls. To achieve speedy execution, the execution of kernels may be offloaded to a hardware block with a fixed or reconfigurable data-path under the control of a supervisor processor. The job of the supervisor processor is to schedule the kernels on the hardware, initialize the hardware with data pointers and other kernel control variables. The interaction between the hardware and the controller could be through interrupts, DMA and setting of control words. Such hardware could be a coprocessor or a reconfigurable data-path capable of executing various kernels and having ability to switch between specified kernel operations by reconfiguring its data-path. The reconfiguration of the data-path may be achieved by simply changing the content of micro-program-counter to the base address of the memory bank that holds the required micro-control-program. This hardware accelerator unit may be capable of accessing the data from a virtual memory for processing and result storing purposes. The data-path may be pipelined and the memory being accessed could be multi-ported. High throughput demands high memory bandwidth and puts very high constraints on timing budget for address generation. Hence, it is desirable to have multiple reconfigurable AGUs capable of concurrent generation of various special address sequences required by variety of DSP kernels, with every reconfigurable data-path. For example, there can be one AGU each for data fetch, coefficient fetch and result write operations. The reconfigurable AGU may have its own local micro-program or a finite state machine for generating sequences of addresses and may function under the supervisory control of the micro-program for that kernel.

Some of the most often used address sequences are Sequential, Sequential with offset, Shuffled, Bit-reversed, Reflected etc. Hulina et al. [7] discuss implementation of Coprocessor for generation of these address sequences and provide 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 or the external memory.

Melis et al. [8] describe the implementation of an AGU that supports different types of circular addressing and striped addressing modes. The AGU designed is concatenated with memory block and an Autonomous Memory Block (AMB) is created. AMB is a memory design abstraction for structured data access. This AGU implemented both as custom design and also on an FPGA, and their performances are compared. They claim reduction in routing resources and required area when implemented on an FPGA.

Efficient generation of Address sequences like Zigzag for entropy coding after DCT operation, 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. Address sequence generation for kernel operations on both stored data and streaming data are also necessary.

This paper describes an Address Generation Unit suitable for a Dynamically Reconfigurable Data-path Processor; capable of generating one address per clock in required sequence; and can be synchronized with the data-path operations by using the *Address-Generate- Enable* signal. The following address sequences suitable for DSP kernels are supported:

- Data, twiddle-factor fetch and result write for FFT kernel.
- Data, impulse response fetch and result write for Convolution Kernel - for both stored and streaming data.

- Data, coefficient fetch and result write for Linear Phase FIR filter – for streaming data input.
- Data fetch from macro-blocks for Motion Estimation kernel.
- Zigzag - suitable for fetching data for entropy coding.
- Other modes like increment and decrement etc.

Hardware for each of these addressing modes was developed separately and later a Comprehensive Address Generator Unit (CAGU) that can generate all the address sequences listed above was designed. The AGU developed is tailored to work with a DRDP, though the concept can be used with any data-path unit with appropriate synchronization. As an example of application of this CAGU, consider an array of reconfigurable processor data-paths that support execution of Single Kernel Multiple Data (SKMD) implying concurrent fetch of data from multiple databanks/data-streams for executing the same kernel operation on these data. The proposed CAGU can be used in such applications as well.

A set of CAGUs pertaining to access of data for a specific kernel can be configured by selecting suitable addressing modes and may be used to generate address sequences required by multiple data-paths.

AGU and DRDP are designed as parameterized 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 are controllable by a micro-programmed controller. Hence reconfigurability is possible with just change of the micro-program.

[9] reports AGUs developed 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, impulse response access for a complete Convolution kernel with implementation of kernel and AGU for accessing data from *N* x *N* pixel array in a Zigzag order used in entropy coding after DCT.

3.1. Address Generation for *N* point FFT Kernel

Jacobson et al. [10] discuss an implementation of dedicated address generator unit that supports reconfigurable radix-4 FFT processor. It uses two counters namely group counter and butterfly counter. The primary address is decomposed into group and butterfly counters. The addresses rotate in groups of 4 after first stage.

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 data-path that implements a single Butterfly operation in Decimation-In-Frequency (DIF) is given in [9] and on the same lines data-path for Decimation-In-Time (DIT) scheme is also implemented.

In each of these data-paths 4 DRDP units have been cascaded and configured to form a single data-path capable of performing one butterfly operation for every two clocks so that *N* point FFT operation is completed in $N\log_2(N)+4$ 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 pre-computed and saved in memory.

3.1.1. 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 [11, 12 and 13]. These algorithms can be classified as those based on heuristic [11] and algorithms using Seed-Table [13]. 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 (AGUs) have also been developed for array processors [14]. Nwachukwu [14], Hulina [7] implemented 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 methods proposed in [9] can generate sequence of addresses suitable for many of multimedia algorithms. They use adders, shifters, counters in the data-path and very few gates 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. [15] 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 [9] for the same functionality needs only 2 shift registers and allied control circuit as depicted in Figure 2. 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. [15].

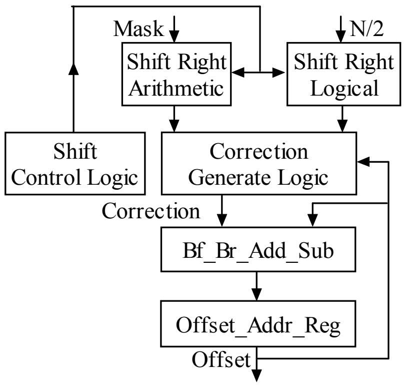

Figure 2. Hardware schematic of Bit Reversed AGU

The input or the output samples need to be re-ordered in bit-reversed 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 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.

3.1.2. Address Generation for accessing Twiddle-factors for *N* point FFT:

For a FFT butterfly operation, a pair of data operands i.e., one fetched from bit-reverse ordered address and the other being the 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 Figure 3.

3.2. Address Generators for Convolution Kernel

Convolution approach 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 anti-symmetric. 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 developed assumes that the given data is padded with $M-1$ zeroes at both ends. 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 kernel with *N* data points and *M* impulse response points is executed in $((N+M-1) \times N + 3)$ clock cycles where the 3 clock cycles correspond to initialization and write latency of last result.

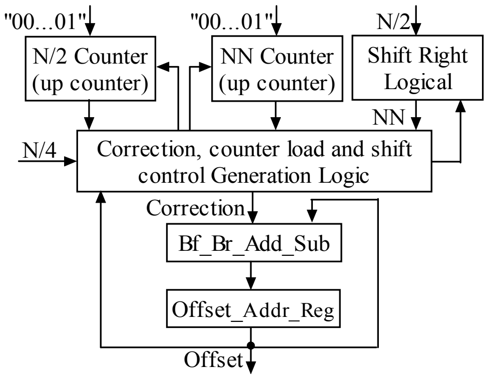

Figure 3. Block schematic of AGU for FFT Twiddle-Factor fetches

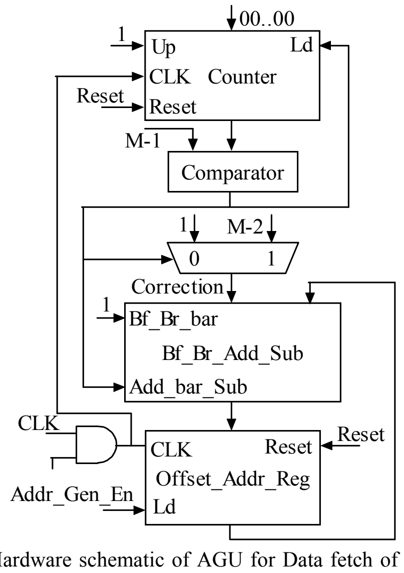

Figure 4. Hardware schematic of AGU for Data fetch of Convolution kernel - stored data

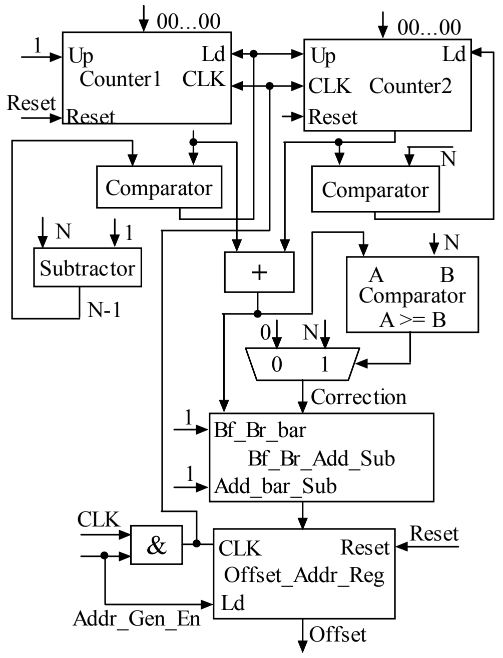

Figure 5. Hardware schematic AGU for data fetch for Convolution kernel - streaming data

3.2.1 Address Generator for fetching data for convolution - stored data and streaming data:

The convolution operation may be performed on stored data or streaming data. Each of these operations needs 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 whether 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.

The hardware schematic of AGU for data fetch in case of stored data is shown in Figure 4. In case of streaming data, the data samples are stored in a circular buffer. Schematic of the AGU hardware is as shown in Figure 5 and the algorithm for generating the address for fetching streaming data can be summarized as shown in Figure 6.

```
Reset: Reset Counter1, Counter2, Offset;
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;
       Go to Begin;
```

Figure 6. Algorithm of AGU for data fetching for Convolution kernel - streaming data

3.2.2. Address Generator for accessing impulse response samples in convolution:

The algorithm for generating the addresses for fetching impulse response is of Modulo *N* type and is similar to that of data fetch for convolution for stored data except for the correction '-(N-1)' whenever the counter reaches a value of N-1; and is as shown in Figure 7.

3.2.3. 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; and is as shown in Figure 8.

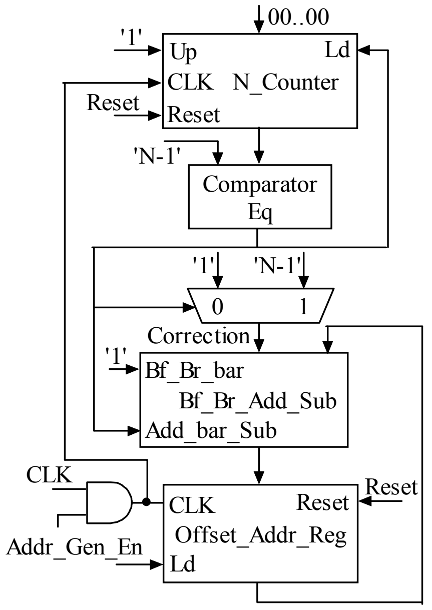

Figure 7. Hardware schematic of AGU for impulse response samples of Convolution kernel

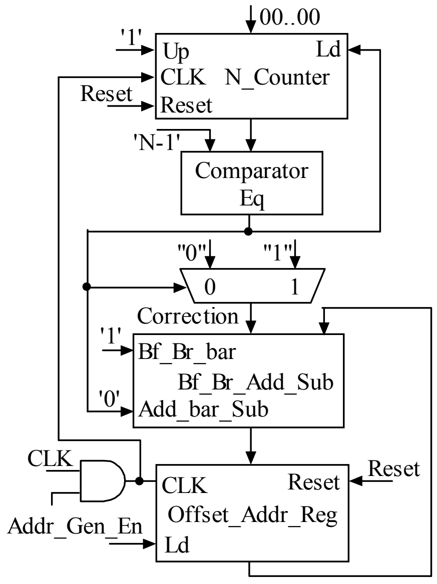

Figure 8. Hardware schematic of AGU for result storage of Convolution kernel

3.3. Address Generator for accessing streaming data for a Linear Phase FIR Kernel

Consider a case of Linear phase FIR filter with even number of coefficients h(n) and they are symmetric. For example: let *N*=4; then h(0)=h(3) and h(1)=h(2). Hence y(n) can be written as

$$y(n) = [x(n)+x(n-3)]h(0) + [x(n-1)+x(n-2)]h(1)$$

A novel algorithm for generating appropriate sequence of addresses to fetch the data has been developed, hardware implemented and tested. The schematic of the hardware is shown in Figure 9 and the corresponding algorithm can be summarized as shown in Figure 10. The address generation scheme for fetching filter-coefficients and storing the results are similar to that of convolution kernel with streaming data.

3.4. AGU for data fetch for Motion Estimation using Block Matching Technique Kernel

Between adjacent frames of video there will be little movement of objects including the background. An object in the previous frame tends to get displaced/rotated in the next frame by a small amount. Practically there may not be much of a difference between the frames.

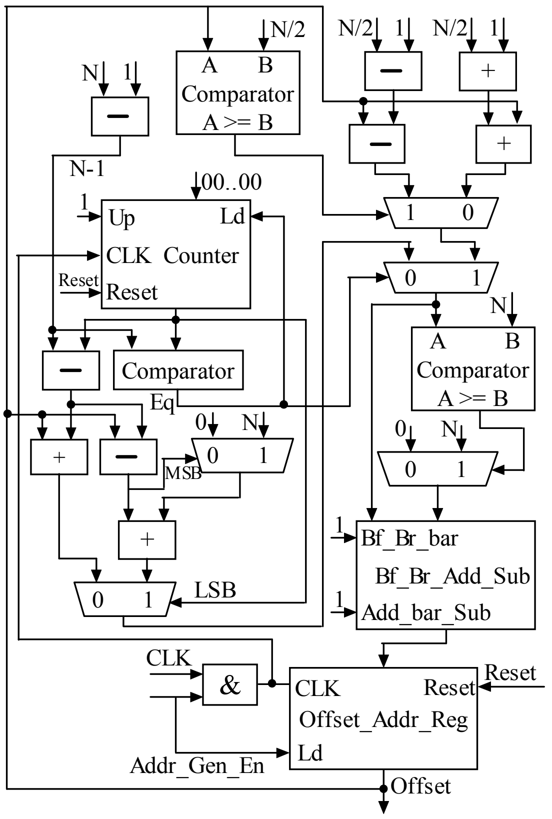

Figure 9. Hardware schematic of AGU for Data fetch for Linear Phase FIR Kernel - streaming data

Frames may contain multiple slices and the slices can be divided into smaller blocks called Macro-Blocks (MB). If a macro-block in the given slice in current frame is compared with the corresponding macro-block or in the neighborhood of the corresponding macro-block 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. Goal of Motion Estimation (ME) is to search for such a block for which the cost function is the least, within a prescribed limit and computing the displacement of the block from the previous to the current frame. This operation needs to be done for all blocks in the frame and is a computationally expensive operation; motion detection over longer distance needs 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). MSE as a cost function is computationally more intensive compared to MAD.

```
Reset: Reset Counter, Offset;
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 (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;
       Go to Begin;
```

Figure 10. Algorithm of AGU for fetching data for Linear Phase FIR kernel - streaming data

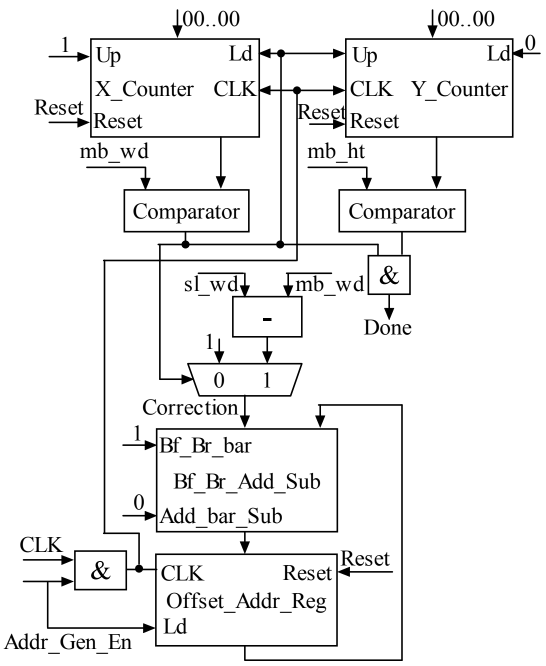

Figure 11. Hardware schematic of AGU for data fetch for Motion Estimation kernel

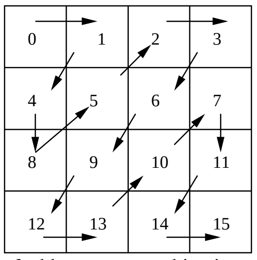

Figure 12. Sequence of address generated in zigzag addressing mode

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 macro-block (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 is as shown in Figure 11.

3.5. Zigzag Address Generation for accessing *N* x *N* pixel array

JPEG uses Entropy coding for compressing the data after performing DCT and Quantization. After computing the 2D-DCT of an *N* x *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 zigzag fashion as shown by the sequence of arrows in Figure 12. An algorithm has been developed to generate this address sequence for any even value of *N* x *N*, hardware has been developed, simulated, and tested. The block schematic is as shown in Figure 13. *Cond*1 to *Cond*7 are conditions that 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.

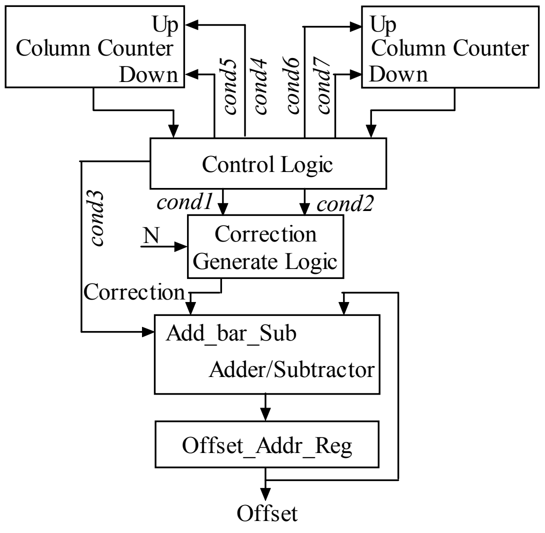

Figure 13. Hardware schematic for Zigzag address generation

4. Results and Conclusions

Efficient Address Generation Algorithms and hardware suitable for speeding up execution of DSP kernels using DRDPs have been developed and tested. The tests prove the efficacy of the AGUs developed, the ability to synchronize the data access and computation of result using the DRDPs as demonstrated in cases of a 8 point DIF FFT kernel, Convolution kernel and SAD computation in Motion Estimation kernel. Timing details of the various kernel executions are shown in Table 1.

More simulation results of these DSP kernel execution using the CAGU and the DRDP are published in our earlier papers [9, 16].

The algorithms have been implemented in VHDL in a fully structured coding style. The data width and the address width are parametered. 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.

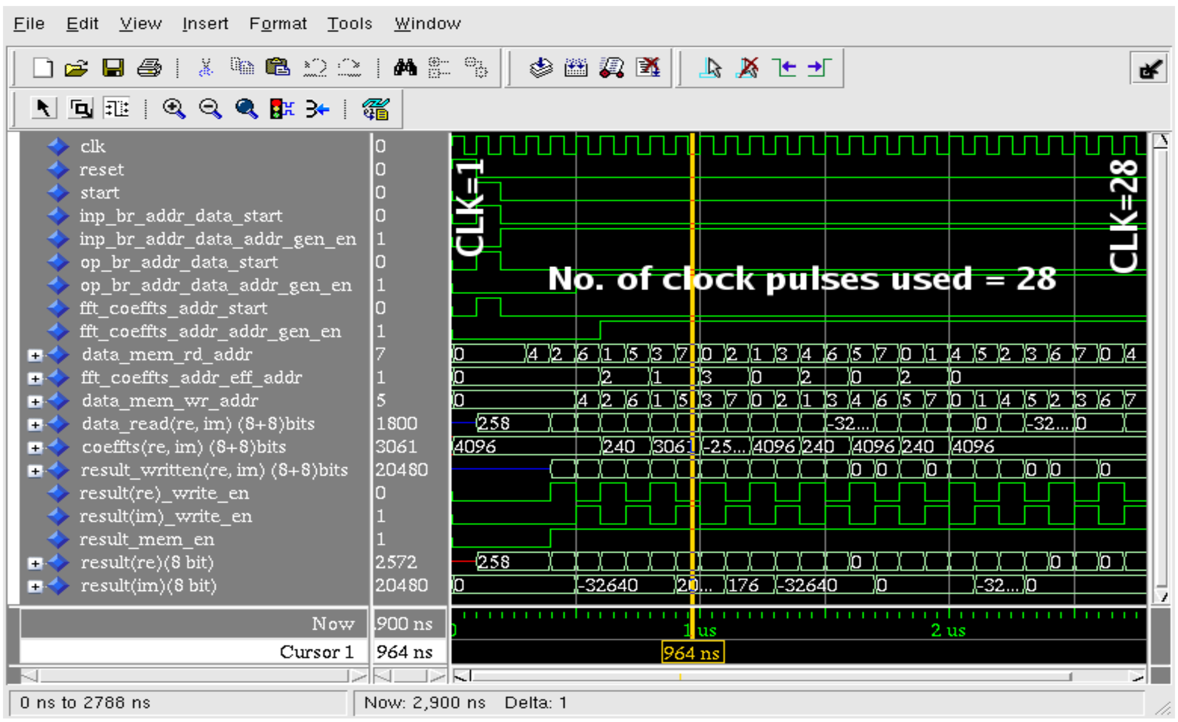

Figure 14. Snapshot of the simulation result of FFT kernel

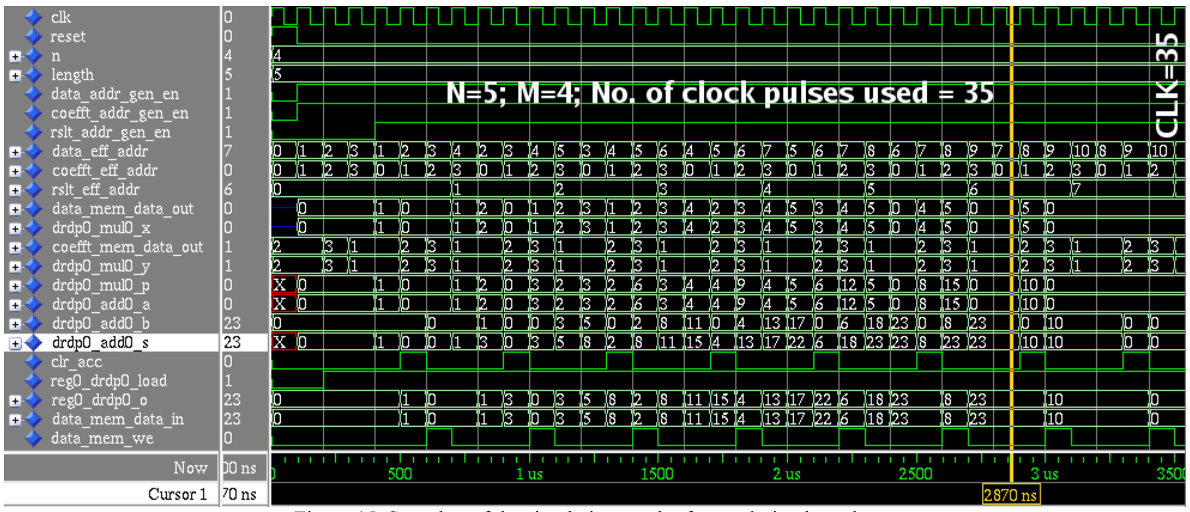

Figure 15. Snapshot of the simulation result of convolution kernel

Structured approach of the coding helps in identifying common components like counters, shifters, adder and comparators etc used in implementing various addressing modes. A Comprehensive Address Generator Unit (CAGU) that can support 11 addressing modes listed earlier has been designed using Up/Down Counters cum Logical/Arithmetic Right Shifters, an Accumulator, Comparators and little glue logic. The CAGU has been tested. The simulation results of FFT kernel and convolution kernel are shown in Figure 14 and Figure 15 respectively.

Table 1. Timing details of various kernel executions

Kernel	Number of clocks	Overhead (initialization and latency)
FFT (*N* point)	*N* $\log_2 N$	4
Convolution	(*N*+*M*-1)*N*	3
ME: SAD (*N* x *M*)	*N* x *M*	2

4.1. ASIC Implementation of the CAGU

The CAGU was synthesized, placed and routed. The post synthesis report is summarized in Table 2. The snapshot of the layout is shown in Figure 14.

Table 2. 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 µ2	21802	43019	64202
Leakage power in nW	39.46	77.84	117.46
Dynamic power in mW	2.95	5.43	8.65

The functional verification was carried out on the extracted net-list and the CAGU is functioning as desired. The design uses Faraday standard cell library based on UMC 0.18µ technology and has been fabricated with Europractice. The chip is currently being tested. The entire implementation process was carried out using Cadence Tool suite; functional simulation and post extraction behavioral simulation were carried out with ModelSim tool. Tools and fabrication were supported by Special Manpower Development Program of DIT, Government of India.

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 data-path the innermost loop of the chosen DSP kernel can be executed in one clock period and (iv) Circuit complexity and power consumption of the circuit increases linearly with increasing address width.

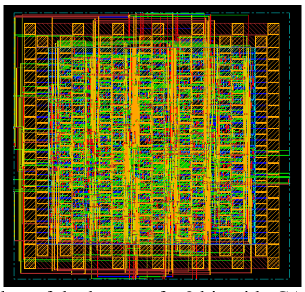
Figure 16. Snapshot of the layout of a 8 bit wide CAGU

5. References

[1] S. Hauck and A. DeHon, Reconfigurable Computing - The Theory and Practice of FPGA Based Computation, Morgan Kauffmann Publishers, 2008.

[2] R. Hartenstein, “Coarse grain reconfigurable architecture (embedded tutorial),” ASP-DAC '01: Proceedings of the 2001 Asia and South Pacific Design Automation Conference, Yokohama, Japan, New York, NY, USA: ACM, 2001, pp. 564-570.

[3] R. Lysecky, G. Stitt, and F. Vahid, “Warp processors,” ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 11, 2006, pp. 659-681.

[4] J.M.P. Cardoso, “Self Loop Pipelining and Reconfigurable Dataflow Arrays,” Workshop on Systems, Architectures, Modeling, and Simulation (SAMOS IV), Samos, Springer Verlag, 2005, pp. 234-243.

[5] Z. Huang, and S. Malik, “Exploiting operation level parallelism through dynamically reconfigurable data-paths,” DAC’02 Proceedings of the 39th annual Design Automation Conference, New Orleans, Louisiana, USA, NewYork, NY, USA: ACM, 2002, pp. 337-342.

[6] M.A. Miranda, F.V.M. Catthoor, M. Janssen, and H.J. De Man, “High-level address optimization and synthesis techniques for data-transfer-intensive applications,” IEEE Trans. Very Large Scale Integr. Syst., vol. 6(4), 1998, pp. 677-686.

[7] P. Hulina, L. Coraor, L. Kurian, and E. John, E, “Design and VLSI implementation of an address generation coprocessor,” Computers and Digital Techniques, IEE Proceedings, Vol.142(2), 1995, pp. 145-151.

[8] W.J.C.Melis, P.Y.K.Cheung, W. Luk, “Scalable structured data access by combining autonomous memory blocks,” IEEE International Conference on Field-Programmable Technology, 2004, pp.457-460.

[9] Ramesh M. Kini, and S. David, “Comprehensive address generator for digital Signal Processing,” International Conference on Industrial and Information Systems (ICIIS), Dec.2009, pp. 325-330.

[10] A.T.Jcobson, D.N.Truong, and B.M.Baas, “The design of a reconfigurable continuous-flow mixed-radix {FFT} processor,” IEEE International Symposium on Circuits and Systems, 2009. ISCAS 2009, pp.1133-1136.

[11] A. Yong, “A better FFT bit-reversal algorithm without tables,” IEEE Transactions on Signal Processing, vol. 39(10), 1991, pp. 2365-2367.

[12] D. Evans, “A second improved digit-reversal permutation algorithm for fast transforms,” IEEE Transactions on Acoustics, Speech and Signal Processing, 1989, vol.37(8), pp. 1288-1291.

[13] J. Walker, “A new bit reversal algorithm,” IEEE Transactions on Acoustics, Speech and Signal Processing, vol.38(8), 1990, pp. 1472-1473.

[14] E. Nwachukwu, “Address Generation in an Array Processor,” IEEE Transactions on Computers, vol.C-34(2), 1985, pp. 170-173.

[15] A. Banerjee, A.S. Dhar, and S. Banerjee, “FPGA realization of a CORDIC based FFT processor for biomedical signal processing,” Microprocessors and Microsystems, vol.25(3), 2001, pp. 131-142.

[16] Ramesh M. Kini and S. David, “Address Genration for DSP Kernels,” International Conference on Communications and Signal Processing (ICCSP), 2011, pp.112-116.

Biographies:

Ramesh Kini. M, obtained his B E degree from Mysore University, in 1984. He obtained his M Tech Degree from Mangalore University in 1997. Currently he is an Associate Professor at National Institute of Technology Karnataka, Surathkal, INDIA.
Email: rameshkinim@gmail.com

Prof. Sumam David obtained her BTech degree from University of Kerala in 1985, and the MTech and PhD degree from Department of Electrical Engineering, Indian Institute of Technology, Madras, India in 1986 and 1992 respectively. Currently she is a Professor of Electronics and Communication at National Institute of Technology Karnataka, Surathkal, INDIA.
Email: sumam@ieee.org