International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 12, December-2014

Data Processing Platform with DSP Slice Based Processor

Supriya Unnikrishnan K[1], Augustine Abraham[1]
[1] PG Student,
TocH Institute of Science and Technology,
Kochi, Kerala

Sudheesh Madhavan[2]
[2]Associate Professor,
TocH Institute of Science and Technology
Kochi, Kerala

Miny G[3], Jayamma T M [3]
[3]Scientist F,
Naval Physical and Oceanographic Laboratory (NPOL),
Kochi, Kerala

Ralph D Kappithan[4]
[4]Scientist,
Naval Physical and Oceanographic Laboratory (NPOL),
Kochi, Kerala

***Abstract*-Computation and Communication are the two major requirements of any VLSI design unit. Many applications in Defense, Robotics, and Industrial Automation etc. demand a reconfigurable platform that incorporates high speed Computation and Communication. All they need is Support for high speed interface for data capture, Memory Storage, Computation fabric for data processing and Communication interface to send processed data to remote terminal. Data Acquisition Control Cards for Sonar (Sound Navigation and Ranging) is an example. Modern day FPGAs (Field Programmable Gate Arrays) are promising integrated solution with all these resources.**

This paper presents a report on how we can effectively use these FPGAs for such application citing example of Xilinx Virtex5 FPGA.With embedded Ethernet MAC (Medium Access Control) and embedded high speed transceivers high speed data communication task is made fairly easy by the vendors.DSP48E1 slices of the high end FPGA are designed specifically for high speed computation. These slices can be combined to form Single Instruction Multiple Data (SIMD) processor with computation power sufficient for front end processing applications. The SIMD processor can be efficiently utilized to make computation parallel. Such a soft processor will combine efficiency introduced by DSP Slices and flexibility introduced by Microprogramming.

I. INTRODUCTION

The data received from sensors, antenna arrays, microphone arrays (sonar) etc. needs a front end controller card the functions of which includes control and configuration of ADC cards, storage of acquired data , data pre-processing like filtering , compression ,AGC(Automatic Gain Control), Encryption etc. Figure 1 shows a data acquisition system framework.

FPGA based cards are promising due to the flexibility, re-configurability, time to market and cost factors. Most of these cards are not required in high volume and hence developing an ASIC (Application Specific Integrated Circuits) will have huge cost impact compared to FPGA.

Virtex5LXT FPGAs have the communication and computational resources embedded in it which can be used effectively for the front end processing.Virtex5 is rich with Embedded Hard Ethernet MAC, High Speed Serial I/O Links, System Monitor and DSP (Digital Signal Processing) Slices.

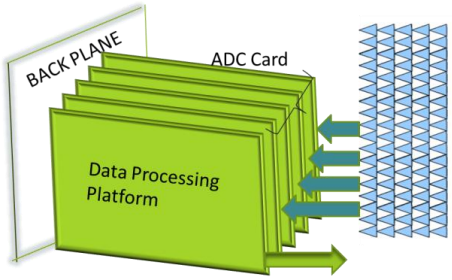


Figure 1 Frontend Data Processing

II. DATA COMMUNICATION AND STORAGE

Modern FPGA devices provide specialized I/O, Hard macros and Soft IPs (Intellectual Property) for supporting all of the cutting edge protocols of the day. Gigabit Ethernet with Serial Gigabit Media Independent Interface (SGMII) and Reduced Gigabit Media Independent Interface (RGMII) support, Gigabit serial backplanes like Rocket I/O Serial Backplane Interface (RSBI), 3GIO, RapidIO, ATA, on chip block memories and specialized circuitry for accessing DDRn (Dual Data Rate) memories are available for user applications.

As shown in figure 2, one example user scenario is Gigabit Ethernet MAC for remote communication, RSBI for backplane interaction with ADC card for data acquisition and Memory Interface Generator for interfacing to external DDR memory storage. ML505/506/507 Evaluation board can be used for controller card development. Embedded microcontroller Micro blaze can be used for control operation through human machine interface (HMI) which is not detailed in this work. If control through soft processor is not required, we can have default configuration running from BRAM (Block Random Access Memory) which can be partially reconfigured in case we

need a different reconfiguration. This control can come through the Gigabit Ethernet.

a) 1 Gb Ethernet

Gigabit Ethernet is an evolutionary step in the advance of networking technology[26][29].Xilinx Virtex5 Embedded Ethernet MAC along with RocketIO GTP transceivers enables interfacing to wide a number of network devices. The Ethernet MAC block is integrated into the FPGA as a hard block in Virtex-5 devices.

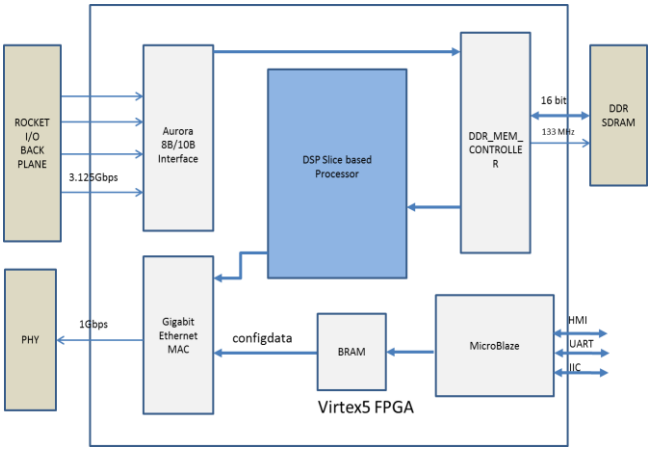


Figure 1 Data Processing Card using Xilinx Virtex5 FPGA

The physical interface of each Ethernet MAC can be configured to operate as one of five different Ethernet interfaces. The Media Independent Interface (MII), Gigabit Media Independent Interface (GMII), and Reduced GMII (RGMII) are parallel interfaces. Serial GMII (SGMII) and 1000 BASE-X are serial interfaces that use the physical coding sub layer (PCS) and physical medium attachment (PMA) sections of the Ethernet MAC. These interface to the Virtex-5 Rocket IO GTP (Gigabit Transceiver) serial transceivers. SGMII provides 10/100/1000 Mbps full-duplex BASE-T functionality.

b) RSBI

Parallel back plane architectures are moving serial to cope with demand for high data rate. Ex –PCI moving to 3GIO[35] .FPGAs can be used for implementing serial back planes using high speed standard serial interfaces like rocket IO , rapid IO ,XAUI,3GIO etc. . Rocket I/O Serial back plane I/F (RSBI) is a flexible light weight serial back plane Interface developed by Xilinx and is now industry standard. It supports 16 channels up to data rate of 3.125Gbps.

c) Memory Interface Generator

MIG is a tool used to generate memory interfaces for Xilinx FPGAs [28].Supports a wide variety of vendors and different types of memories like DDR, DDR2, DDR3, and QDRAM packed as components, Unbuffered Dual in Line Memory Module (UDIMMS), Small Outline Dual in Line Memory Module (SODIMMS) or Registered Dual in Line Memory Module (RDIMMS).

III. PARALLEL COMPUTING IN DSP

The inherent data parallelism found in DSP functions has made DSP Algorithms a very good candidate for hardware implementation. Reconfigurable Hardware provides flexibility of software programmable processors and performance of fixed functionality hardware. The core processing element of most reconfigurable platforms are FPGA which contains large resources of registers, block RAMs and DSP slices that can be effectively used for fine/coarse grain parallelism of DSP application. A lot of research is being done to use FPGA resources for parallel computation. [11]

A. PARALLEL COMPUTATION DESIGN ASPECTS

The general design aspects of parallel computation are discussed below. [1][11][23]

a) Appropriate Granularity for Reconfigurable fabric

Each unit of computation, or logic block, can be as simple as a 3-input look up table (LUT), or as complex as a 4-bit ALU

- Fine Grained : Manipulates data at bit level
- Coarse Grained: Manipulates group of bits via complex function units (Ex: PipeRench).In coarse grained again, the computational granularity is also important aspect i.e. whether to do 8/16/32 bit data processing.

b) Interconnect topology of Logic blocks

- Nearest Neighbor : Each logic block communicates directly with its immediate neighbor
- Segmented: short wires accommodate local communications traffic. These short wires can be connected together using switchboxes to emulate longer wires.
- Hierarchical: local routing within a cluster and longer wires at the boundaries connect the different clusters together.

c) Memory management

An increase in the operation parallelism results in a respective increase in the rate by which data are fetched from memory - called data memory bandwidth. It is a major bottleneck in exploiting the inherent parallelism. Floating point operations per second (FLOPs) is directly proportional to memory bandwidth. Inside FPGA, each Processing Element (PE) unit can have a memory associated with it or can use centralized RAM system. Scratch pad, Register File or Buffer I/F is used between External and Internal memory.

d) Configuration Scheme

- Single context: Needs complete reconfiguration to change any of the programming bits.
- Muticontext: Has multiple layers of programming bits .Each layer can be active at different point of time.
- Partially Reconfigurable: Can be selectively programmed without complete reconfiguration

e) Programming model

Parallel software development is not as mature as hardware .OpenCL and Open MP are promising development towards direction of single platform for parallel programming. For Single Instruction Multiple Data (SIMD) machines**,** popular and successful data parallel languages include CMF, *Lisp, C*, Ruby etc.

B. CASE STUDIES

Different architectures are developed for parallel computation in academy and industry. There are many aspects to differentiate between them.

- Number of cores
- Processing Element Granularity – Complexity of PE(1/8/16/32 bit)
- Homogeneous/Heterogeneous
- Reconfiguration Strategy
- Processing Element Interconnection
- Memory Hierarchy
- SIMD(Single Instruction Multiple Data)/MIMD(Multiple Instruction Multiple Data)
- Target Applications/Optimizations

Below are some of the architectures developed in last three decades

PipeRench [8]: ALU based coarse grain system with linear topology. Use Pipelined configuration where reconfigurable fabric is divided to physical pipeline stages. A row of processing elements forms a pipeline. Each such pipeline can be configured individually. Uses a set of CAD tools to synthesize a stripe based on parameters No of PE's in the stripe, Width in bits of each PE and Number of registers in PE's register file. Dataflow intermediate Language (DIL) is used for programming.

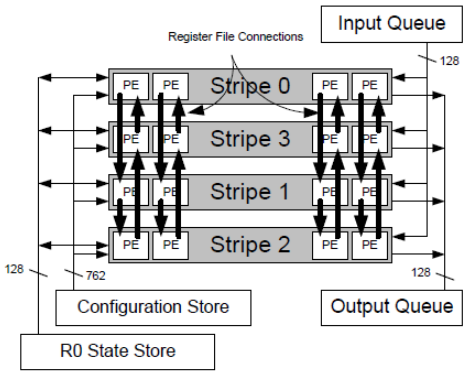


Figure 2 PipeRench Architecture [8]

ILLIAC [2] is an array processor with a two dimensionally arranged 64 PEs controlled by a Control Unit. Each PE consists of sophisticated ALU together with six programmable registers used for computation, memory addressing and intercommunication. A memory is associated with each PE. High speed routing lines run between routing registers of PE. Nearest neighbor and nearest eighth neighbor interconnection is enabled. The command is broadcasted to all processing elements .Illiac IV FORTRAN and Tranquil are some of the programming languages attempted.

MORA [19] is a two Dimensional mesh based CGRA system and consists of array of reconfigurable cells arranged in four 4x4 quadrants. Reconfigurable Cell (RC) consists of 8-bit Processing Element, 256*8 dual port data memory and a central controller. MORA does not use a centralized RAM system. Instead each RC is provided with a 256 8 bits data memory bank. Depending on the instruction word, multiplexers write the result of the current PE operation or data from external RC into the specified memory address, through the specified port. Programming language is done in low level MORA specific assembly language.

SYSCORE [20] is proposed for low power bio signal processing where ASICS are costly and general purpose processers cannot meet high performance and low power requirements. Configurable function units (CFU) and Round about Interconnect (RAI) are the basic processing elements. CFU are interconnected to East and West neighbor and RAI columns after every two columns of CFU facilitate interconnection between any East and any West CFUs. RAM accesses are restricted so as to minimize power. Data reuse is applied and intermediate results are pipelined till we get the

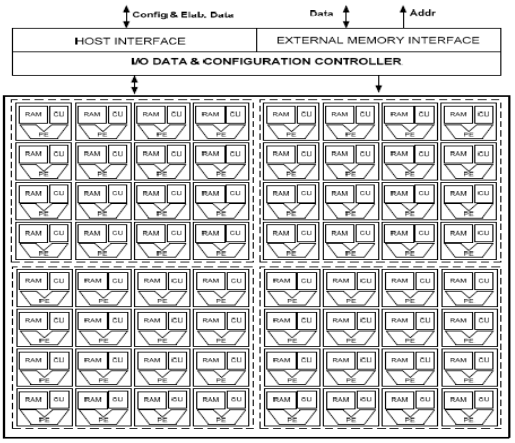


Figure 3 MORA Architecture [19]

final result. DMA unit in the left and top drives the data sequence and output is connected by another DMA unit. Execution is spread over three modes: Configuration mode, Execution mode and Flush Mode.

Morphosys [7] is a two dimensional reconfigurable cell array processor meant for DSP platforms, image processing etc. Interconnection is hierarchical where 8x8 array is divided in to four quadrants of 4x4 arrays. Inside quadrants, there is full connectivity between rows and columns. Between the quadrants, connectivity is through express lane where anyone PE in the row can access to any other PE in the same row. Tiny RISC processor is used for the control and sequencing operations. Frame Buffer holds the streaming data and connected to external SDRAM

memory through DMA Controller. Dynamic reconfiguration is achieved with context memory the content of which is stored in to context registers in the reconfigurable cell .These registers decides the operation/configuration of Reconfigurable Cell. A graphical user interface tool mView generates assembly code for RC array from the user inputs.

FLORA [17] is specialized for floating point operations.PE s are grouped together to handle mantissa and exponent part of the floating point arithmetic. Configuration memory contains information on the operation to be performed and interconnection for each PE. Data memory is accessible by Reduced Instruction Set (RISC) processor though DMA unit. External RISC processor executes control intensive code units. A small size finite state machine is implemented in each PE for supporting floating point operations. It supports Add/Sub, Sqrt, Multiplication and Division.

Nano processor [6] employs principle of data flow mapping. This implies distributed control using basic nodes like SWITCH, SELECT. While loops are build using these basic dataflow operators. Computational unit is 16 bit ALU with Booth multiplication support. Memory nodes store multiple internal states. This can be considered a cluster of nodes. Communication through two levels, Level 1 connects four Nano processors .A semi crossbar interconnect used for Level 2 interconnection.

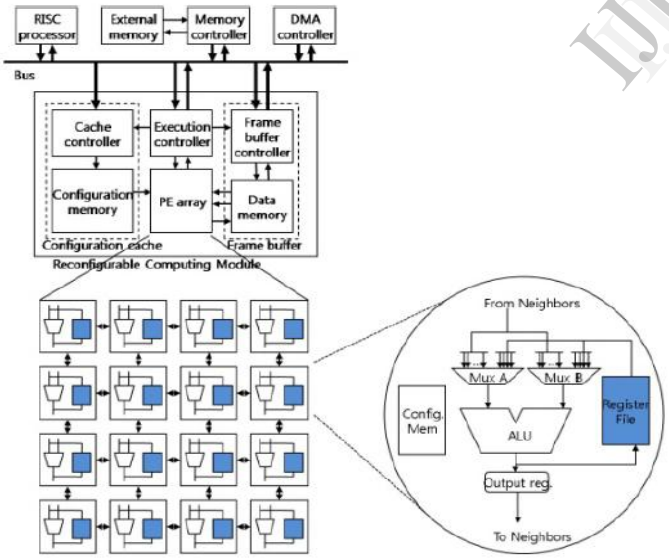


Figure 4 FLORA Architecture [17]

PACT-XPP [12] also uses dataflow mapping algorithm for implementing DSP applications. Reconfiguration and partial reconfiguration is used for sequencing of operations. Meant for data streaming and hence doesn't use on chip storage memory .Consists of I/O array, RAM array and ALU array. Registers are used for routing from top to bottom and vice versa. Interconnection through packet network. Native Mapping Language (NML) used for dataflow graph mapping.

RAPID [10] provides a simple computational platform with no overhead of Cache and register file .Meant for data streaming applications where data is either stored in external memory or acquired directly from sensors. It is designed as Linear Interconnection of coarse grain function units. Memory Interface programmed to load /read internal FIFOs. This data stream is passed through pipelined data path which is controlled by instruction stream.

A 167 processor [16] platform uses Globally Asynchronous Locally Synchronous Concept for clocking and though individual clock halting achieve power efficiency. Dynamic voltage scaling and frequency scaling is used for low power. Specialized Circuits are introduced for common functions like FFT, Viterbi decoding etc. Uses On chip large memory for large data sharing across the processors. Has 16 bit data path and use Asynchronous FIFO for synchronization between processors. It is arranged as mesh and use circuit based switching for intercommunication.

Smart Cell [15] is designed for data streaming applications .Use 64 coarse grain processing element which are interconnected using three layer of interconnection. 4 PEs are connected together in cross bar switch which is termed as tile. These tiles are connected together in mesh like architecture with static nearest neighbor connection. In the third level of connection, Mesh network enables non neighbor PE communication. Cells can be configured in SIMD, MIMD or systolic array fashion. The instruction memory of each PE is connected as linear array using SPI chain and can be dynamically configured. Smart C is being developed for programming. Use 16 bit granular operations.

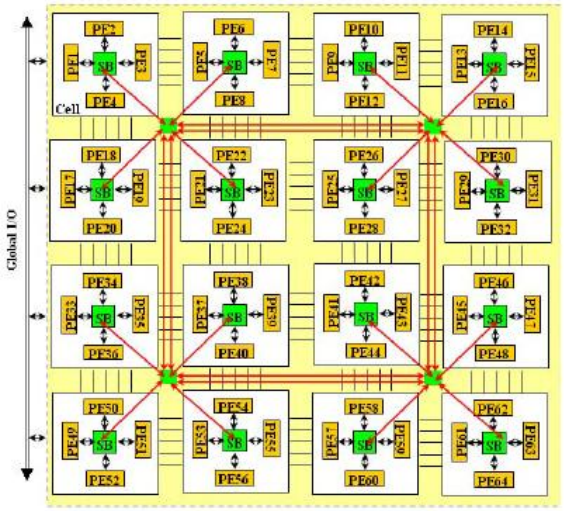


Figure 5 SMARTCELL Architecture [15]

SODA [14] is one multi-core DSP architecture designed specifically for software-defined radio (SDR) applications. System consists of four data processing units; one control processor and one global scratch pad memory connected

though a shared bus. SODA PE consists of SIMD pipeline for vector processing and scalar pipeline for control operations. Two local memories are used Scalar and SIMD pipeline .DMA unit is used for memory access. Intra-PE communication is achieved using specialized network called SIMD Shuffle Network. Data processing is done in 16 bit granularity.

Commercial Multi-Core DSP Platforms

Connection Machine [5]: Pioneered SIMD machine with CM-1, CM-2 and CM-5 Huge Array Processor with Flexible Interconnection mechanism like broadcast, global OR, Hypercube and NEWs grid for NN communication .CM1 consists of up to 64K data processors, one to four sequencers, and one to four front end computer interfaces. Each data processor includes ALU, 4Kbits of bit addressable memory, flag registers and router interface. Used standard languages with extensions for parallel computation support.*Lisp is an example.

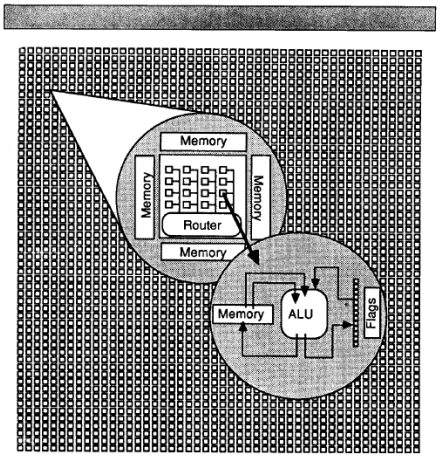


Figure 6 Connection Machine Diagram [5]

Transputer [4] is one of the first commercially available designs made to facilitate MIMD system. Transputer's unique feature was point to point link which makes the multi core connection easy .Each transputer is a microcomputer with local memory and communication link to connect to another transputer. It implements a hardware scheme for process scheduling. The programming model for transputers is defined by Occam.

Ambric [30] is massively parallel MIMD system which is differentiated by the usage of Asynchronous processing elements. Ambric processors are interconnected by Ambric channels. These channels are formed by Ambric registers and permits input and output to run at different clock rates.

Freescale's MSC8256FS [34] is six core DSP processor designed for medical imaging, Aerospace and defense applications built on Starcore technology. Based on homogeneous hierarchical interconnect with chip level arbitration and switching system (CLASS)

Pico Chip PC205 [22] is mesh interconnection of 248 VLIW DSP processors.ARM926EJ is used for performing control functions. Each processor has instruction and data memory. Use custom programming model.

TILERA's TILEPro64 [31]: Homogeneous mesh interconnection architecture. The cache coherence is directory based and it employs three way VLIW CPU. Each tile consists of CPU, L1, L2 cache and a switch for interconnection. The architecture exploits the massive parallelism inherent in DSP algorithms. Not suitable for general purpose applications. Use TILERA's gentle slope programming model.

NVIDIA G200 [33] specialized for data intensive applications 240 cores. These are arranged in 10 clusters of 24 cores. Each cluster is controlled in a SIMD manner and employs noncoherent memory system. Every 8 core shares 16Kb local memory. Use CUDA as programming model.

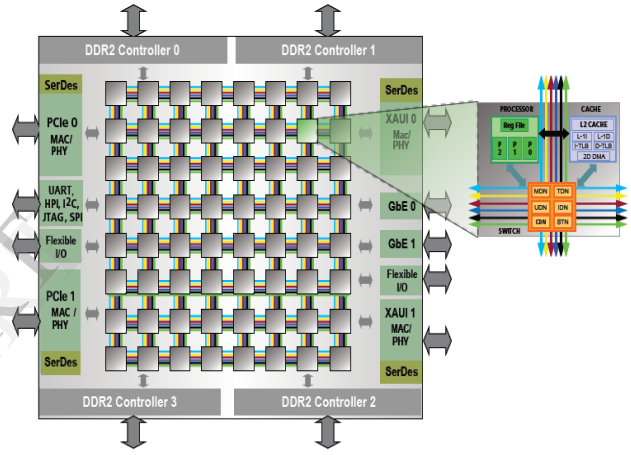


Figure 7 TILERA TILEPro64 [31]

TI Keystone [32] architecture supports integration of homogeneous cores of fixed point and floating point C66XX DSP Core as well as heterogeneous system which integrates ARM Cortex and C66XX cores. In homogeneous systems up to 8 C66X CorePac DSP Cores are connected with Tera Net high bandwidth switching fabric. Each core has 32KB of L1 data and instruction cache, 512 KB L2 Cache .Multicore Shared Memory Controller arbitrates access to shared memory of 4 MB .OpenMP, OpenCL programming model is supported in Keystone architecture

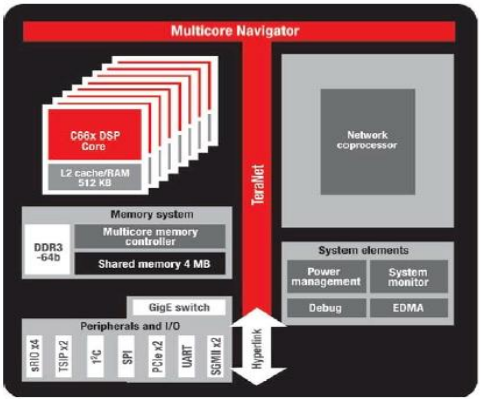


Figure 8 Key Stone Architecture[32]

IV. DSP SLICE

The math operation of the DSP Slice consists of 25bit by 18 bit two's complement multiplier followed by three 48 bit three path multiplexers[27][18] .This is followed by three input adder/subtract or two input logic unit.

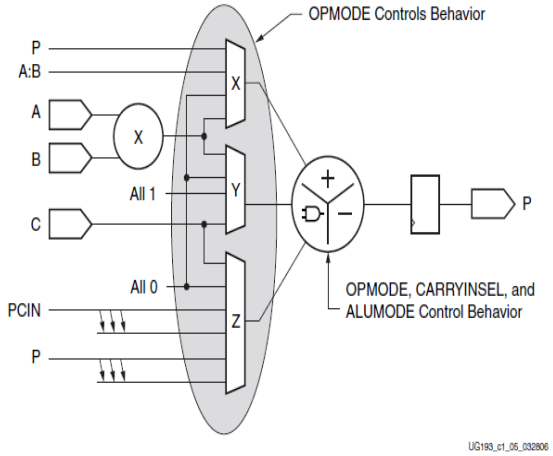


Figure 9 DSP Slice Simplified Diagram [27]

DSP Slice Configuration is through Core Generator, a tool integrated with Xilinx ISE.DSP Slice can be configured for 64 unique instructions selected from a list of available instructions. Clock gating using CE signal can be used for power sensitive applications. Different pipeline options are available for the optimization between speed and area.

V. DSP IMPLEMENTATION REQUIREMENTS

Digital Signal Processors, unlike general purpose processors are optimized for real time math operations [25]. While general purpose processors are used for applications where I/O processing and control operations are more important, DSP s are optimized for fast repetitive math for real time processing.

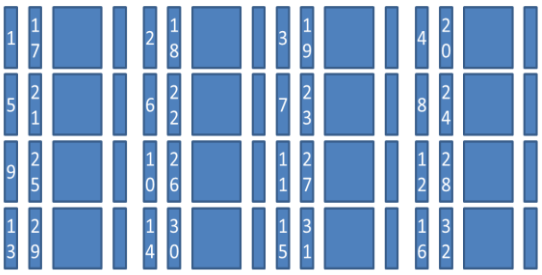
Basic requirements of DSPs include fast arithmetic, extended precision, dual operand fetch, circular buffering and Zero Overhead looping. Below are the analysis of steps required for some basic computation Data Average, Matrix Multiplication and FIR filter. The DSP Slices are used as processing elements and the kind of intercommunication and memory access requirements are studied.

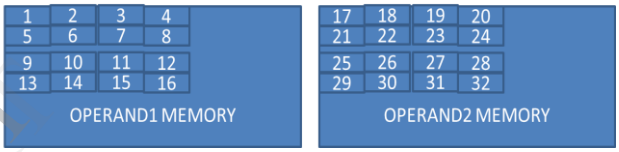
a) DATA SUMMATION

Since FPGAs are rich in block RAMs, two separate memories are used for operand 1 and operand 2 for every 4x4 tile of DSP Slices.16 bit operation is assumed and hence BRAM is of 4x16 width and depth of 1024. If we are using streaming applications, we can capture the data in input registers.

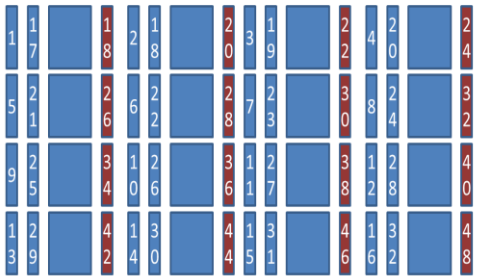
Assuming, we have the data to be summated stored in two Operand memories, the steps needed for summation is illustrated below.

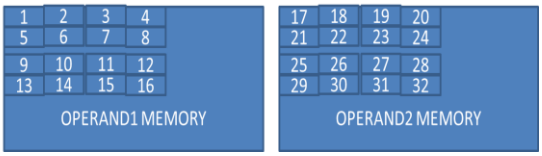
1) Row Read instruction will populate row of the DSP Slice array .To populate four rows this instruction need to be iterated four times. If we have enough resource, we can include matrix read instruction which can populate rows and columns of DSP Array in single instruction. For this to be fast, we need to prefetch and keep this data in memory registers. Each DSP slice need to have two input registers and one o/p registers to hold data. In case of streaming applications, these registers can hold the input samples.





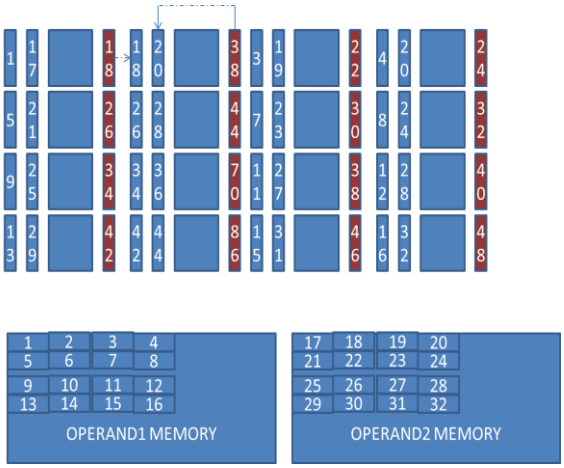
2) Input register 1 is connected to input A of DSP Slice and Input Reg 2 to input B of DSP Slice. Add instruction to DSP Slice will give summation of contents of registers.



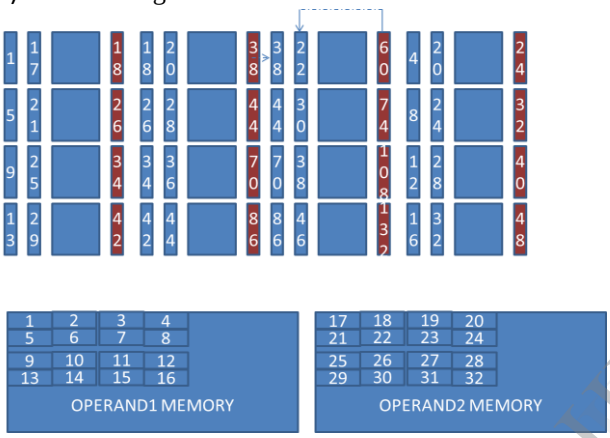


3) Mesh based interconnected is simple and suited for structured DSP computation.8 Nearest Neighbour Connection can be provided. Instructions for right, left, top, bottom shifts are required. Do a right shift and

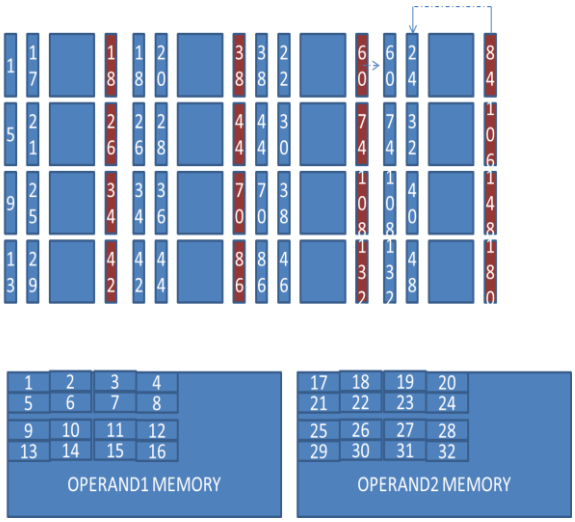
Accumulate ADD i.e. P+A



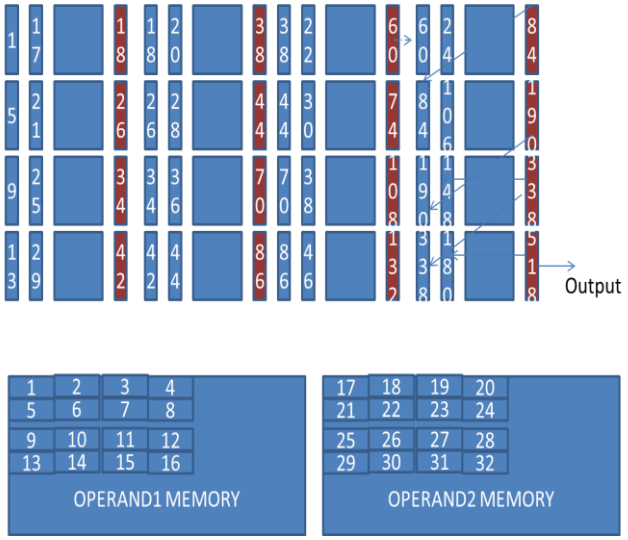
4) Do a right shift and Accumulate ADD i.e. P+A



5) Do a right shift and Accumulate ADD i.e. P+A. These repetitive actions can be formulated as Row Add instruction.



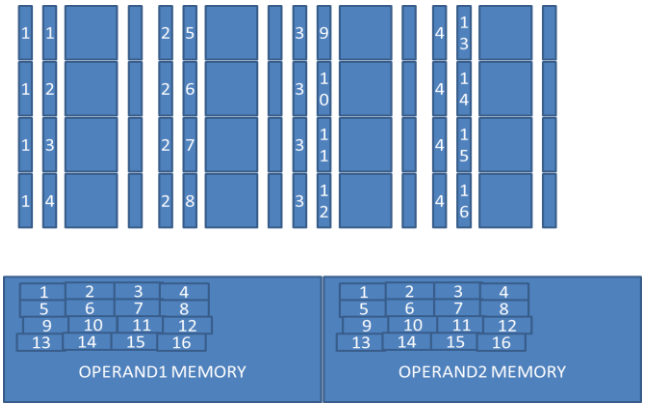
6) Similarly do a Column wise Accumulate and add. A column add instruction will give final o/p and this need to be written to output memory.



Cascading interconnection between DSP slices is available in the FPGA fabric which can be used for summation over a row.

b) MATRIX MULTIPLICATION

Let us say we have the 4X4matrix1 stored in operand 1 and 4X4matrix2 in operand 2 memory.



1)Do operand1 row read and do shift down four times so that input register1 of each row of DSP Slice array will have first row of matrix1 (This operation can be formulated as Row Read Broadcast) .Operand2 memory matrix transpose read will populate input register 2 of DSP Array with matrix2 transpose. If resources allow, we can have matrix transpose read instruction or else we can built this using column read and row read instructions.

2) Execute DSP Slice instruction A*B

3) Do Row Add .This gives first row elements of result matrix.

4) Do row write to output memory.

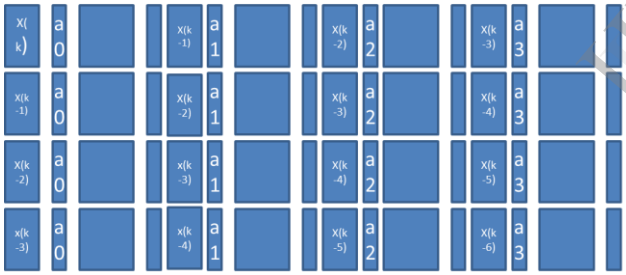
5) Now execute Operand1 memory row read broadcast. DSP Slice array will have second row of matrix1

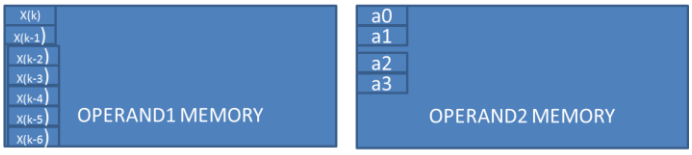
6) Execute DSP Slice instruction A*B

7) Do Row Add .This gives second row elements of result matrix
8) Do row write to output memory.
9) Now execute Operand1 memory row read broadcast. DSP Slice array will have third row of matrix1.
10) Execute DSP Slice instruction A*B.
11) Do Row Add .This gives third row elements of result matrix.
12) Do row write to output memory.
13) Now execute Operand1 memory row read broadcast. DSP Slice array will have third row of matrix1
14) Execute DSP Slice instruction A*B
15) Do Row Add .This gives fourth row elements of result matrix
16) Do row write to output memory.
The above operations can be looped if we introduce loop instruction in the array processor.

c) FIR FILTER

Let input samples be loaded in operand 1 memory as column. Coefficients of 4 tap filter in operand2 memory
Read filter coefficients into I/P Register 2.Operand2 column read broadcast will populate DSP array rows with the coefficient.
2) Do Column read of operand1 memory (4 elements x (k), x (k-1), x (k-2), x (k-3) loaded in first row)
3) Increment Row Address
4) Do Column read of operand1 memory (4 elements x (k-1), x (k-2), x (k-3), x (k-4) loaded in second row)





5) Increment Row Address
6) Do Column read of operand1 memory (4 elements x (k-2), x (k-3), x (k-4), x (k-5) loaded in third row)
7) Increment Row Address
8) Do Column read of operand1 memory (4 elements x (k-3), x (k-4), x (k-5), x (k-6) loaded in fourth row)
9) Execute A*B DSP Instruction and do a Row add. (Instructions 2 and 3 can be put in a loop of count 4). We get 4 output values of Filter.
Repeat this for no of samples

a) Online data processing

In some cases we might have to process streaming data. In that case, we can have 4 streams of data coming in .Latch the data in to first row element of DSP array. Shift right the sample and latch in new sample. Repeat this for number of taps. Execute Row add Instruction. In parallel, we should be taking in data sample.

VI. DSP SLICE ARRAY PROCESSOR

An array processor for DSP applications can be developed using mesh based array of DSP Slices readily available in high end FPGAs.
The idea is to configure this array for

- SIMD or MIMD operations,
- The number of slices in SIMD tile.
- Number of SIMD tiles configured for MIMD processing
- Streaming data or Stored Data
- Computation Granularity (BIT_WIDTH Default 16 bit)
- Interconnection strategy – Mesh or Crossbar

A. SIMD

While configured for SIMD operations, array can have (assuming bit_width = 16)

- 8x8 DSP Slices
- Each DSP slice is associated with 32 bit input register 1/2 and output register
- Interconnection is done using 8NN network
- Every 4x4 array will have three instances of 64 bit Wide Block RAM which is stitched up using 16 bit wide BRAMs .A data matrix sufficient to populate the elements of a SIMD tile is always buffered in MemBuf registers.
- These are used as operand1 memory, operand 2 memory and output data memory.
- The instruction fetching, decoding and sequencing and controlling of operation is done by Finite State Machine which has direct access to instruction memory.
- A shared memory controller provides access to external storage device DDR.Can be configured for number of lanes.

B. MIMD

Two or more SIMD tiles can be used for enabling MIMD operations. Each tile will be associated with Operand1/Operand2/Instruction/Output Memory, FSM for sequencing

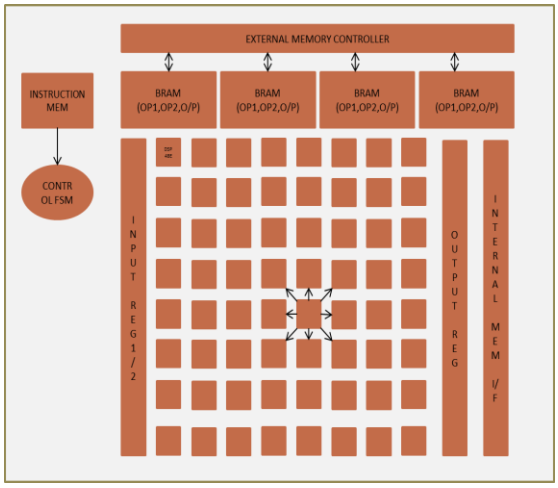


Figure 13 DSP Slices Configured in SIMD Mode

and control .Data Exchange between these tiles can be done through the memory. This is useful when pipelined functions to be done on data like DCT, Quantization, and Huffman Encoding etc. in a JPEG Algorithm.

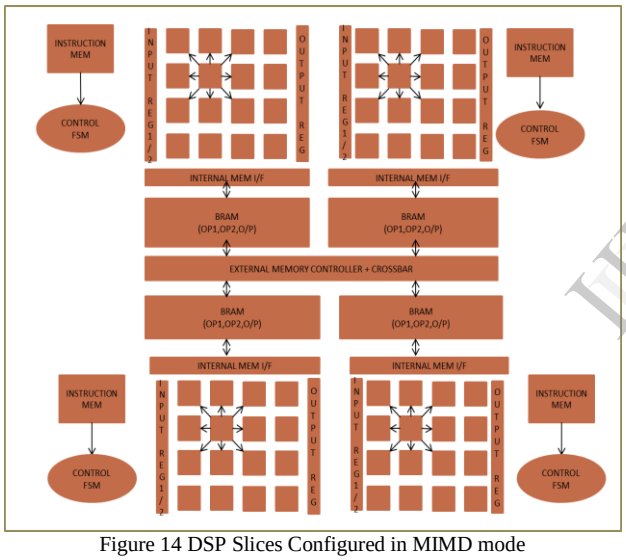


Figure 14 DSP Slices Configured in MIMD mode

C. *Instruction Set*

Custom Instruction Set is developed depending on the control and storage access requirements. Assembler parsers based on Perl, python etc. can be developed for conversion to binary.

DSP Slice Instructions	
64 instructions selected from the available list	
Control Instructions	
Shift TP/BT/RT/LT/TC/BC/RC/LC	Move o/p to i/p of any one neighbor
I/O Latch	Latching the I/O data
Bypass TP/BT/RT/LT/TC/BC/RC/LC	Move from i/p reg to i/p reg of any one neighbor
Memory Access Instructions	
Matrix read	Memory read of 4 rows to populate all elements of DSP array
Matrix transpose read	Memory transpose read of 4 rows to populate all elements of DSP array. Row data in memory written to column of DSP array
Row read	Row read from memory
Column Read	Row read from memory and data assigned to column of array processor
Row Write <row no>	Write the o/p register contents of a particular row to memory
Column Write <Column no>	Write the o/p register contents of a particular column to memory

VII. CONCLUSION

FPGAs based controller cards are easy to design with the vendors providing almost all the required industry standard communication interfaces and specialized computation macros. Monitoring utilities like System Monitors facilitates debugging of the system. With an extra care in placement and utilization of logic resources, power and performance can also be made well within the standards required by the specialized applications like front end sonar, radar processing etc.

REFERENCES

[1] Compton, C., Hauck, S. et Al. An Introduction to Reconfigurable Computing. IEEE Computer (April 2000).
[2] BOUKNIGHT, W. J., ET AL. The Illiac IV system. Proc. IEEE 60, 4 (Apr. 1972), 369-388. (1972)
[3] S. Y. Kung, *VLSI Array Processors*, 1988 :Prentice-Hall
[4] Transputer architecture Reference Manual (1987)
[5] D. Hillis *The Connection Machine*, 1985 :MIT Press
[6] A. K. W. Yeung, J.M. Rabaey: A Reconfigurable Data-driven Multiprocessor Architecture for Rapid Prototyping of High Throughput DSP Algorithms; Proc. HICSS-26, Kauai, Hawaii, Jan. 1993.
[7] H. Singh , M.-H. Lee , G. Lu , F. Kurdahi , N. Bagherzadeh , E. Chaves . MorphoSys: an integrated reconfigurable system for data-parallel and compute intensive applications. IEEE Trans. Comput. , 5 , 465 – 481
[8] H. Schmit, D. Whelihan, A. Tsai, M. Moe, B. Levine, and R. R. Taylor. PipeRench: A virtualized programmable datapath in 0.18 micron technology. In Proc. of IEEE Custom Integrated Circuits Conference, 2002.
[9] Kees A. Vissers, Parallel Processing Architectures for Reconfigurable Systems, Proceedings of the conference on Design, Automation and Test in Europe, p.10396, March 03-07, 2003
[10] Carl Ebeling , Darren C. Cronquist , Paul Franklin, RaPiD - Reconfigurable Pipelined Datapath, Proceedings of the 6th International Workshop on Field-Programmable Logic, Smart Applications, New Paradigms and Compilers, p.126-135, September 23-25, 1996

[11] R. Tessier, W. Burleson, "Reconfigurable Computing and Digital Signal Processing: Past, Present, and Future", Programmable Digital Signal Processors, Yu Wen Hu, ed., Marcel Dekker, pp. 147--186, 2002.
[12] Syed MA, Schueler E (2006) Reconfigurable parallel computing architecture for on-board data processing. In: Proceedings of the 1st NASA/ESA conference on adaptive hardware and systems. IEEE Press, New York, pp 229–236
[13] G. Blake, R.G. Dreslinski and T. Mudge, &ldquo,A Survey of Multicore Processors,&rdquo, *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 26-37, Nov. 2009.
[14] Y. Lin, H. Lee , M. Who , Y. Harel , S. Mahlke , T. Mudge , C. Chakrabarti and K. Flautner "SODA: A high-performance DSP architecture for software-defined radio", *IEEE Micro*, vol. 27, no. 1, pp.114 -123 2007
[15] C. Liang and X. Huang,"Smartcell: A power-efficient reconfigurable architecture for data streaming applications," in Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on, oct. 2008, pp. 257-262
[16] D. N. Truong , W. H. Cheng , T. Mohsenin , Z. Yu , A. T. Jacobson , G. Landge , M. J. Meeuwsen , A. T. Tran , Xiao , E. W. Work , J. W. Webb , P. V. Mejia and M. Baas "A 167-processor computational platform in 65nm", *IEEE J. Solid-State Circuits*, vol. 44, no. 4, pp.1130 -1144 2009
[17] Lee, D., Jo, M., Han, K. and Choi, K.: FloRA: Coarse-Grained Reconfigurable Architecture with Floating-Point Operation Capability, *International Conference on Field-Programmable Technology*, pp.376-379 (2009).
[18] H. Y. Cheah, S. A. Fahmy, D. L. Maskell, and C. Kulkarni, "A lean FPGA soft processor built using a DSP block," in Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), 2012, pp. 237-240.
[19] Sai Rahul Chalamalasetti , Sohan Purohit , Martin Margala , Wim Vanderbauwhede, MORA - An Architecture and Programming Model for a Resource Efficient Coarse Grained Reconfigurable Processor, Proceedings of the 2009 NASA/ESA Conference on Adaptive Hardware and Systems, p.389-396, July 29-August 01, 2009 [doi>10.1109/AHS.2009.37]
[20] S. M. Kunjan Patel and C. J. Bleakley, "Syscore: A coarse grainedreconfigurable array architecture for low energy biosignalprocessing, " IEEE International Symposium on Field-ProgrammableCustom Com-puting Machines, vol. 978-0-7695-4301-7/11, 2011IEEE
[21] R. Hartenstein, "A Decade of Reconfigurable Computing: A Visionary Retrospective," *Proc. Design, Automation and Test in Europe* (DATE 01), IEEE CS Press, 2001, pp. 642-649.
[22] L. J. Karam , I. AlKamal , A. Gatherer , G. A. Frantz , D. V. Anderson and B. L. Evans "Trends in multicore DSP platforms", *IEEE Signal Process. Mag.*, vol. 26, no. 6, pp.38 -49 2009
[23] Yu, Z. (2010) 'Towards High-Performance and Energy-Efficient Multi-core Processors', in K. Iniewski (ed.), C*MOS Processors and Memories*. Dordrecht et al.: Springer.
[24] Mahendra Pratap Singh and Manoj Kumar Jain. Article: A Survey of Reconfigurable Architectures. *International Journal of Computer Applications* 98(14):35-40, July 2014.
[25] Mixed-Signal and DSP Design Techniques, edited by Walt Kester (Newnes, 2003)
[26] http://www.design-reuse.com/articles/15175/designing-with-virtex-5-embedded-tri-mode-ethernet-macs.html
[27] www.xilinx.com/support/documentation/user_guides/ug193.pdf
[28] www.xilinx.com/support/documentation/ip_documentation/ug086.pdf
[29] http://www.xilinx.com/support/documentation/ip_documentation/tri_mode_eth_mac_ug138.pdf
[30] Ambric : TeraOPS Hardware : A new massively parallel MIMD computing fabric IC
[31] http://www.tilera.com/sites/default/files/productbriefs/TILEPro64_Processor_PB019_v4.pdf
[32] http://www.ti.com/lsds/ti/dsp/keystone/overview.page
[33] www.nvidia.in
[34] www.freescale.com
[35] www.xilinx.com/aurora/ 1_xilinx_backplanes_v5.pdf