# An Introduction to an Array Memory Processor for Application Specific Acceleration

Gerald G. Pechanek* Nikos Pitsianis†‡

*Independent Consultant
107 Stoneleigh Drive
Cary, NC 2751

†Department of Electrical and Computer Engineering
Aristotle University of Thessaloniki
Thessaloniki 54124, Greece

‡Department of Computer Science
Duke University
Durham, NC 27708, USA

***Abstract*—In this paper, we introduce an Array Memory (AM) processor. The AM processor uses a shared memory network amenable to on-chip 3D stacking. Node couplings use a 1 to $K$ adjacency of connections in each dimension of communication of an array of nodes, such as an $R \times C$ array where $R \geq K$ and $C \geq K$ and $K$ is a positive odd integer. This design also provides data sharing between processors within sub-arrays of the $R \times C$ array to support high-performance programmable application specific processing. A new instruction set architecture is proposed that has arithmetic instructions that do not require the specification of any source or target operand addresses. The source operands and target values are provided by separate load, store, and arithmetic instructions that are appropriately scheduled with the arithmetic instruction to be executed to reduce the storage of temporary variables for lower power implementations.**

***Keywords*—network organization and topology; adjacency of connections; 3D physical layout; parallel architectures; application specific processors (ASP).**

## I. Introduction

Two significant performance problems in multi-processor architectures involve accessing data from memory and the sharing of data between processors. Having adequate memory bandwidth is related to the organization of the processors, memory modules, and interconnection network. In the past, a wide variety of interconnection networks have been studied and deployed in multiprocessing systems. Examples include, rings, bus connected, tree, shuffle, omega, butterfly, Banyan, Clos, mesh, crossbar, and hypercube as described in a number of surveys, such as [1] and [2], folded array [3], and ManArray networks [4]. Selecting a network has generally been driven by application-specific performance and cost requirements.

In order to provide high performance interconnects, many general purpose on-chip multiprocessor systems sacrifice network connectivity in order to implement a scalable network with a simple connection structure such as used in mesh, torus, bus-based, or hypercube networks. In many cases, as the number of on-chip processors increase, such on-chip networks are being replaced by advanced networks on chip (NOCs) with network management units to provide adequate bandwidth and address scalability issues [5], [6]. Such NOCs introduce great complexity and utilize a significant amount of chip area [7]. This paper introduces a novel scalable interconnection network, the Wings network, and processor architecture that shows potential for on-chip 3D stacking, high performance, and low implementation cost for use in application specific processors (ASPs). The Array Memory (AM) processor and its use in accelerating an FFT example is also briefly described.

The structure of this paper is as follows. Section II provides a brief review of nodal topologies and their important characteristics. Next, Section III gives an overview of the Wings network while Section IV highlights its network layout properties. Section V describes the Wings architecture and Section VI provides a Fast Fourier Transform (FFT) example program operative on the AM processor. Section VII concludes the paper and suggests areas for future research.

## II. Background

A shared memory network for an array of processors $P$ and an array of memories $M$ includes a store network ($P \rightarrow M$) and a load network ($M \rightarrow P$) to support parallel store and load operations between the $P$ and $M$ endpoints. The load network and the store network each have a set of internal nodes distributed among $S$ stages with unidirectional links (L) between nodes in each stage. Stages of a network are made up of distributors, crossbar switches, or multiplexer nodes. The number of stages varies depending on number of inputs and the topology of the network. The network cost $C$ is presented in terms of the number of connection paths in the network. For example, a ring network of $N$ nodes provides connections between adjacent nodes in a linear organization with $L = 4$, due to each node having 2 separate transmit links and 2 separate receive links, $C$ =O($2N$), since a transmit link from one node becomes a receive link at the other end. A four neighborhood regular torus $G$ by $G$ network of $N = G^2$ nodes provides unidirectional connections between North, East, West, and South (NEWS) nodes with $L = 8$, due to each node having 4 separate transmit and 4 separate receive links, $C$ =O($4G^2$). However, the ring and torus networks are not generally considered shared memory networks where a memory node is an endpoint of a store network and a source point for a load network. In another example, a crossbar switch network provides complete connectivity between the processors and the memories, with P*M nodes, for simplicity $P = M = N$, with each network path having $L = N$, and for combined Load + Store networks, $C$ =O($2N^2$). An $N$ by $N$ crossbar implemented with 2-to-1 multiplexers

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

(mpxs) requires, for $N$ a power of 2, a path length of $\log_2 N$ stages [8].

Multistage interconnection networks in which input nodes have a single input and multiple outputs are considered processor nodes. Also, output nodes which have multiple inputs and a single output are considered memory nodes. In other multistage interconnection networks in which the input nodes have two or more inputs and multiple outputs, these input nodes are considered part of the network and processor nodes are added as a first stage of the shared memory network. For example, a unidirectional $r$-dimension butterfly network [9] is configured with $(r+1)2^r$ 2 by 2 nodes, $r$ stages of connections between the $2 \times 2$ nodes, $2^r$ $2 \times 2$ nodes per each stage, and $r2^{r+1}$ edges. The total butterfly unidirectional network cost with $P = M = 2^{r+1}$, is $P + M + r2^{r+1} = (r+2)2^{r+1}$ edges.

A Clos network [10] is configured with at least three stages of crossbar switches. For a three stage Clos network, the ingress stage has $C$ crossbars each having $A$ inputs and $B$ outputs. There are $CA$ processor nodes, with each processor node providing one of the $CA$ network inputs to the first stage of the Clos network. The second (middle) stage has $B$ crossbars each having $C$ inputs and $C$ outputs. The egress stage has $C$ crossbars each having $B$ inputs and $A$ outputs. There are $CA$ memory nodes, with each memory node receiving one of the $CA$ network outputs of the egress stage. For a strict sense nonblocking Clos network, $B \geq 2A - 1$.

A Benes network [11] is a rearrangeably non-blocking Clos network configured with $I = CA$ network inputs and $O = CB$ network outputs, $A = B = 2$, $C$ $2 \times 2$ crossbars in each stage, and having $2\log_2(2C) - 1$ stages. There are $CA$ processor nodes, with each processor node providing one of the $CA$ inputs. There are $CA$ memory nodes, with each memory node receiving one of the $CA$ outputs of the egress stage.

A Banyan network [12] has $CA$ network inputs, $CA$ network outputs, $J$ columns of $CA/D$ $D \times D$ crossbar switches, where $CA = D^J$ and $D \geq 2$, and the crossbar switches supporting all $D!$ permutations [12]. There are $A$ processor nodes, with each processor node providing one of the $A$ inputs to the first stage of the Banyan network. There are $B = A$ memory nodes, with each memory node receiving one of the $B$ outputs of the egress stage.

## III. Wings Network

Fig. 1 illustrates a linear arrangement of a 2-dimensional (2D) row by column $(R = 4) \times (C = 4)$ Wings Store (WS) network having $3(16)$ nodes made up of 16 processors, 16 $K \times K$ crossbars, with $K = 3$ in Fig. 1, and 16 memories. Links between nodes in a first stage of the WS network are made according to a 1 to $K$ adjacency of nodes, which includes wrap around adjacent nodes, in a first dimension of an $R \times C$ matrix of nodes, where $K$ is an odd integer, $K > 1$, $R \geq K$ and $C \geq K$. In the second dimension, links are made according to the 1 to $K$ adjacency of nodes, including wrap around links in the second dimension. For $K = 3$, the links in the first dimension represent the East/West links

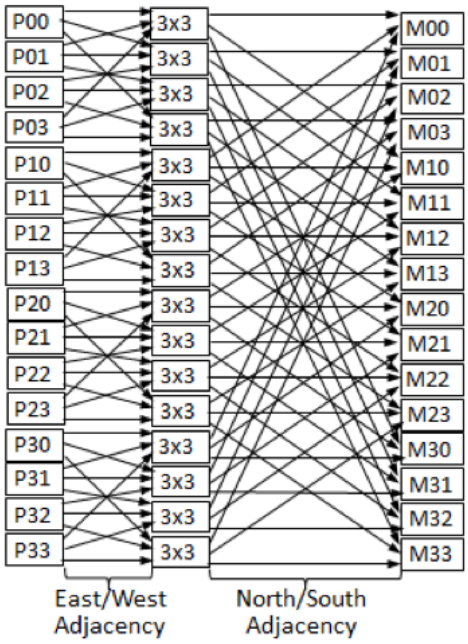


Fig. 1. Wings Store (WS) network.

and links in the second dimension represent the North/south links of an $R \times C$ torus connected array. Links are also included for direct feedthrough paths between each processor node and its corresponding memory node. A Wings Load (WL) network is symmetrical to the WS network of Fig. 1, with changed connection directions for loading data from a memory to a processor. For $K = 3$, each Prc is connected by three unidirectional links to three $3 \times 3$ crossbars and each $3 \times 3$ crossbar is connected to three Mrcs. The 2D WS network of Fig. 1 uses 2 stages of $KRC$ links/stage requiring $2KRC$ links. A 3D WS network, such as a $4 \times 4 \times 4$, uses 4 planes of 3 stages of $KRC$ links/stage requiring $12KRC$ links. As defined above, a Wings network is not a butterfly, Clos, Benes, or Banyan network. Table I compares shared memory networks implemented with $2 \times 2$ crossbars with the Wings networks implemented with $3 \times 3$ crossbars. Each shared memory network consists of a first network having 64 processors (Ps) connected to 64 memories (Ms) and a second network consisting of the 64Ms connected to the 64Ps, the Ps and Ms in both networks being the same units.

For $K = 3$, a minimum network size, as defined, is a $3 \times 3$ network. For $K = 3$, a processor executing a store operation can write data to a single memory node or to combinations of up to nine memory nodes. Using a store and a load operation, a processor can communicate to $5 \times 5$=twenty five surrounding processors. For $K = 5$ the minimum network size, as defined, is a $5 \times 5$ network. In a Wings array memory network with 1 to 5 adjacency connectivity, a processor node executing a store operation can write data to a single memory node or to combinations of up to twenty five memory nodes. Using a store and a load operation, a processor can communicate to $9 \times 9$=eighty one surrounding processors.

While the complexity of a $3 \times 3$ crossbar is greater than the

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

TABLE I
EXAMPLE FOR 64P TO 64M NETWORKS.

| Network | $\#Z \times Z$s L or S network, | #delay stages | $\#Z \times Z$s in L&S network | #Links in L&S network |
|---|---|---|---|---|
| Crossbar | $64P \times 64M$ | 6-2to1 mpxs | Add a $2^{nd}$ $64M \times 64P$ | $2 \cdot 4096 = 8192$ |
| Butterfly | 32 $2 \times 2$s/stage, $6 \cdot 32 = 192$ $2 \times 2$s | 7 | $2 \cdot 192 = 384$ | $2(r+2)2^{r+1} = 896$ |
| Benes | 32 $2 \times 2$s/stage, $11 \cdot 32 = 352$ $2 \times 2$s | 12 | $2 \cdot 352 = 704$ | 1536 |
| Banyan | 32 $2 \times 2$s/stage, $6 \cdot 32 = 192$ $2 \times 2$s | 7 | $2 \cdot 192 = 384$ | 2(384+64) = 896 |
| Wings $8 \times 8$ | 64 $3 \times 3$s | 2 | 128 $3 \times 3$s | $2 \cdot 384 = 768$ |

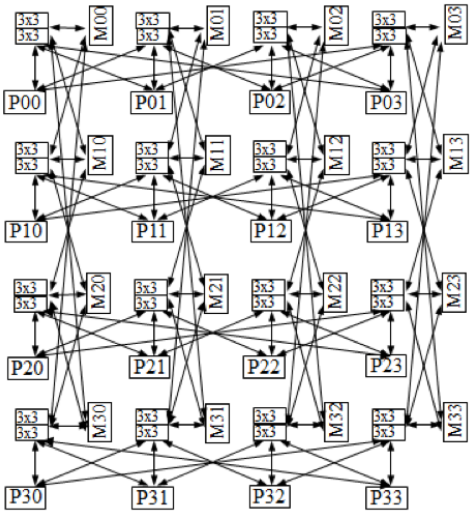


Fig. 2. 3D Array Memory Processor.

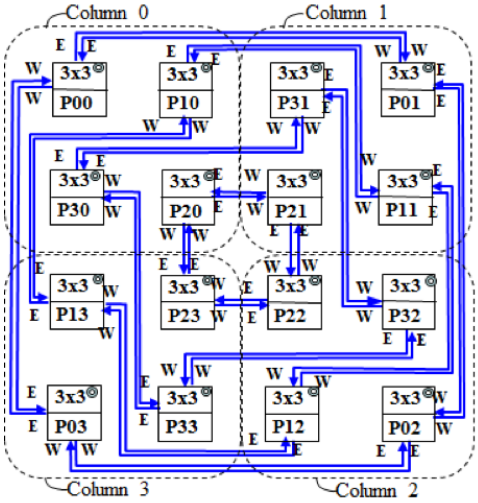


Fig. 3. 3D Cluster Grouping and East/West Wiring.

complexity of a $2 \times 2$ crossbar, the $3 \times 3$ crossbar is less than twice as complex. In an implementation, the area of 128 $3 \times 3$s of the Wings $8 \times 8$ configuration would be less than the area of 384 $2 \times 2$s of a comparable butterfly or Banyan network. Also, the Wings $8 \times 8$ configuration uses 768 links as compared to the 896 links of the butterfly and of the Banyan networks. In Fig. 2, the memory elements (M00-M33) are overlaid on corresponding processor elements (P00-P33) and organized as a 3D array memory processor. The load and store networks are two independent networks, each having a set of crossbars that are individually controlled and operated in parallel. Due to the symmetric nature of the load and store networks, the crossbar units of both networks are able to also overlap the processor and memory nodes. Further details of the Wings network, 3D layout and stacking, the proposed architecture, and application to neural and fast Fourier transpose (FFT) processing can be found in multiple awarded U.S. Patents, including 7,581,079 [13], 7,886,128 [14], 8,156,311 [15], 8,443,169 [16], 9,460,048 [17], 9,507,603 [18], and a pending U.S. Patent Application [19].

## IV. WINGS NETWORK 3D LAYOUT

Problems associated with a large memory array include connectivity and wire length between memory elements and the communicating processing elements and the corresponding impact of capacitance on the memory wiring. Such problems increase power usage and lower performance. The Wings network lends itself to a 3D implementation minimizing wiring length which, generally, lowers capacitance, reduces power usage, and increases performance. For example, a most highly used interconnection is between a processor element and its corresponding memory element, which in the Wings array memory processor is of minimum length due to the stacking of the memories over the corresponding processing elements. Fig. 3 illustrates an example layout of East/West plane wiring in the WS network of Fig. 1 with the processors P organized in groups by column, see columns in Fig. 2, in order to equalize the wire lengths. The memory hierarchy, made up of an L1 instruction cache, L1 data cache, L2, and L3 caches, that is application dependent, take up a significant portion of on-chip area in current processing chips [20]. By placing the memory, having a capacity required by an ASP, on a plane separate from the processors, the processor core area can be minimized and also benefit from use of shorter wiring lengths.

Each processor provides an E output to a first $3 \times 3$, a W output to a second $3 \times 3$, and a direct connection to third $3 \times 3$ crossbar that is associated with the processor. In a separate plane, Ms corresponding to the Ps and overlayed on the Ps are wired locally in the column groups. In this separate plane, each $3 \times 3$ crossbar switch of Fig. 3 provides a North (N) output to a first M, a South (S) output to a second M, and a direct connection to a third M that is associated with the processor. For example, for column 3, the N/S wires are provided between crossbars and memories M03, M13, M23, and M33.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

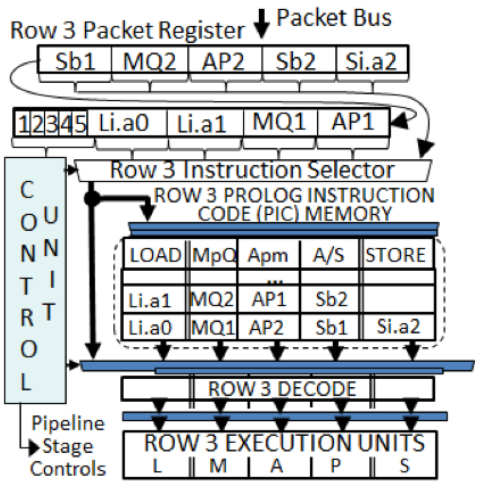


Fig. 4. Example Processor Pipeline Control.

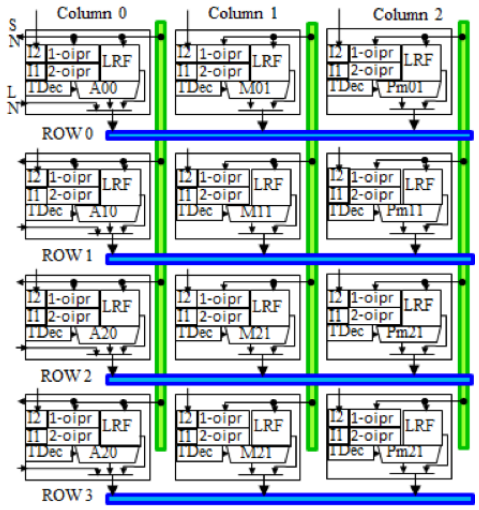


Fig. 5. Example Processor Execution Array.

## V. Processor Architecture

Fig. 4 illustrates an example processor pipeline control that receives a chained execution packet (CEP) having a header 12345, which contains control parameters for an identified execution row, such as row 3, as described in more detail with regards to Figs. 5, 8, and 9. Upon receiving a fetched CEP, the control parameters are loaded into the programmable control unit to control the pipeline stage operations. Instructions are selected from a row 3 packet register and loaded into a prolog instruction control (PIC) memory. The PIC memory loads up the instructions during execution of the prolog and are accessible in combinations of up to five instructions for parallel decode and parallel execution. An FFT example is provided to further describe the processor architecture and operation. The row 3 execution units are one row of an arithmetic execution array shown in Fig. 5.

Fig. 5 illustrates the arithmetic execution array for a single processor node having row0-row3 execution elements (REs).

| 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rv | Instr Type 000, 001 | | | ALU/ MPU Opcode | | | | LxF R0-6, R7= 1oipr | | | LyF R0-6, R7= 2oipr | | | s/us | 3-bit Data type | | | 4th dst I4 | | | 3rd dst I3 | | | 2nd dst I2 | | | 1st dst I1 | | |

Fig. 6. Exemplary ALU or MPY Instruction.

Rows 0-3 have the same functionality in each column of the execution array. The store network (SN) and the load network (LN) are each partitioned into four data unit paths, such as four 32-bit paths, to support parallel operations in the four rows of REs. The REs are each configured with operand input pipe registers (OIPRs) and a distributed local register file, (LRFs) connected by a Wings array local network (WALN) allowing an RE to write to an OIPR (RE $\rightarrow$ oipr), RE $\rightarrow$ WALN $\rightarrow$ a distributed (oipr), RE $\rightarrow$ LRF, and RE $\rightarrow$ WALN $\rightarrow$ a distributed LRF. Each LRF is accessible by an execution element in the array and each LRF has at least one read port and one write port. With two register entries in each LRF, the single processor node distributed LRF would have a capacity of twenty four entries with twelve read ports and twelve write ports. The delay due to the one stage of $3 \times 3$ crossbar switches used in the execution array is minimal. Wiring delays are also minimized by equalizing path length between nodes including wrap around connections. Also, since the AM networks are scalable, higher capacity register files with even a greater number of parallel locally accessible ports can be implemented.

To support chaining of instructions across the internal processor execution array, an arithmetic logic (AL) instruction does not need to specify a target storage address in a central register file or in a memory unit where execution results are stored. In order to provide operands to an AL unit and save results from the AL unit, an AL instruction is paired with a load and a store instruction or with another instruction or instructions that when executed provide source operands and take result operands for further processing or storage.

Fig. 6 illustrates an exemplary ALU or MPY instruction format. While 32-bit instruction format sizes are not precluded, a 26-bit instruction format size is described. Bits 22-24 encode an instruction type as 000 to indicate an arithmetic logic (AL) type instruction. A 4-bit function opcode in bits 18-21 allows the encoding of sixteen different functions. Bits 15-17 Rx encode a first source register address 0-6 selected from a local file (LRF) associated with the execution unit for this particular instruction. Bits 15-17 Rx encoded with a binary seven (111) indicates the input source data is to be selected from a first OIPR. Bits 12-14 Ry encode a second source register address 0-6 selected from a local file (LRF) associated with the execution unit for this particular instruction. Bits 12-14 Ry encoded with a binary seven (111) indicate the input source data is to be selected from a second OIPR. Bit 11 s/us is used to indicate a signed or unsigned data type. A 3-bit data type (Dtype) in bits 8-10 specifies various data type formats, such as packed 8-bit, 16-bit, 32-bit and the like formats and

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

| 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R v | Instr Type 101 | | | Load or Store Opcode | | | | Shared Memory Block | | | | | Load or Store Specific | | | | | | | | | 2nd dst I2 | | 1st dst I1 | |

Fig. 7. Exemplary Load or Store Instruction.

floating point formats.

The result of executing a function instruction may be directed forward to a 1st OIPR or a 2nd OIPR of 1 or 2 or 3 or 4 following instructions or combinations of two, three, or four instructions in a chained execution packet (CEP). This is referred to as result forwarding and specified by the destination (dst) bits0-7 which identifies instructions I1-I4 following the source instruction. Each destination specification, such as bits 0 and 1, may be encoded to a not used indication (NU=00), to use a 1st OIPR=01, to use a 2nd OIPR=10, or to use both OIPRs=11. The hardware determines an available execution unit having the OIPRs or LRFs associated with the destination instruction.

Fig. 7 illustrates an exemplary load or store instruction format indicated as 101 encoded in bits 22-24. Bits 18-21 are encoded to indicate the type of load or store operation to be executed, such as a load PCrelative, load indirect, load indirect with increment, and the like. For loads, the shared memory block select field, bits 13-17, is encoded to select a shared memory as the source of data to be loaded. For a 1to3 adjacency connectivity network, each processor node may connect by means of the Wings network to nine memory blocks. By using a 1to5 adjacency connectivity network, each processor node may connect by means of the Wings network to twenty five memory block nodes. The load operation fetches a data element and forwards the result of the fetch to a first destination and/or a second destination.

A permute/move instruction is encoded similarly to an ALU/Mpy instruction of Fig. 6, but with a bit field that indicates a destination row number of an execution array where data is to be moved and with dst bits 0-7 indicating a destination instruction for the specified destination row number.

From a system perspective, the AM processor is controlled by a separate control processor that initializes the AM processor, provides a co-processor interface to a host system to specify signal flow graph parameters, access to array data, and controls direct memory access transfers to the array of memories. Each processor in the array is a multiple instruction streams, multiple data streams (MIMD) processor that fetches its own instructions and data. The instruction memories have a direct path to the processor to provide access to the fetched CEPs and are located on the same plane as the data memories.

## VI. Wings FFT Example

Fig. 8 illustrates an example signal flow graph for a Fourier transform of length 4. Such a cell can be used in the implementation of the fast Fourier transform (FFT) algorithm,

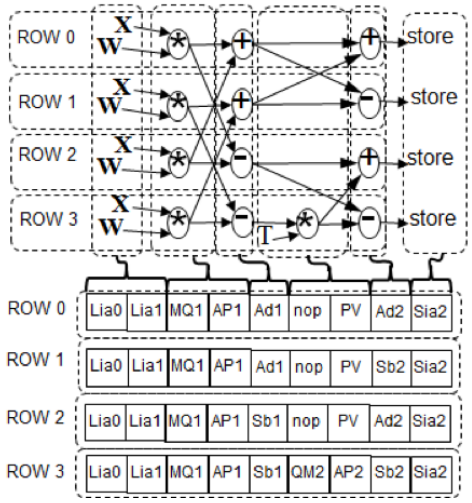


Fig. 8. FFT Signal Flow Graph with CEPs.

similar to the hypercube FFT [21], [22]. Marked with ROWs 0-3 are mappings to chained execution packets (CEPs). Column functions and transfers in the signal flow graph are enclosed in dashed boxes, corresponding to Wings instructions in each CEP.

The FFT signal flow graph requires a vector of twiddle factors W, a signal vector X and a complex constant value T. Complex arithmetic operations multiplication, addition, and subtraction produce results that are stored to memory at the final stage. The execution of the CEPs instructions is controlled by one or more finite state machines, included in pipeline control of Fig. 4, that are programmed at the time the CEPs are initially loaded. Each CEP also includes a control header, 12345 of Fig. 4, that is used to identify the processing element, signal graph opcode, packet length, and additional programmatic information, such as loop prolog and epilogue length, loop body repeat count, and related information for compact code representation and efficient execution [18].

The use of CEPs, generally without register file operand specifications, saves power. For example, the execution of the rows 0-3 CEPs saves 28 register file write operations and 36 register file read operations for the execution of the four CEPs as compared to standard RISC type processor execution.

Fig. 9 illustrates an FFT CEP pipeline. The transfer of data across rows is through the 4x3 Wings array local network (WALN) used in Fig. 5. The instructions enclosed with dashed boxes in Fig. 9 are described in this section.

The first instruction in the row 3 CEP, Fig. 4 and 8, is a load indirect (Li) instruction that loads a 32-bit complex data value $W_3$ which is comprised of a real 16-bit portion $W_{3r}$ and an imaginary 16-bit portion $W_{3i}$. The execution of the Li instruction accesses the complex word at a memory address based on an address register 0 (a0) that is auto incremented each execution for delivery through the Wings load network to a destination execution unit associated with the destination instruction, cycle 1 Fig. 9.

As part of initialization, the input $T$ is loaded into a local file associated with a Row 3 multiplier. The Li.a0 instruction,

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

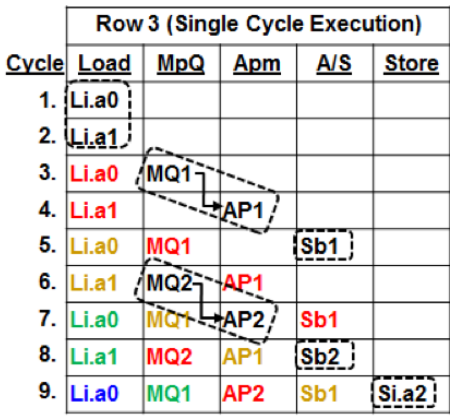


Fig. 9. FFT CEP Pipeline.

referenced as instruction zero, is coded to indicate a chained linkage to the MpQ1 instruction, which is instruction 2 in the row 3 CEP counting from the source Li.a0 instruction zero. The Li.a0 instruction also indicates that the fetched data is to be directed to the 1st OIPR of the execution unit the MpQ1 instruction is dispatched to. In a similar manner, the next instruction in the row 3 CEP after Lia0 is another load indirect instruction Lia1 which operates in a similar manner to the first Li instruction, but uses address register 1 (a1). The Li.a1 instruction, instruction zero, indicates that the fetched data $X_3$ is to be forwarded to the 2nd OIPR of the execution unit the MpQ1 instruction is dispatched to, cycle 2 Fig. 9. The MpQ1 instruction causes the four loaded 16-bit operands to be operated on and produce four 32-bit results which are directed to the 1st OIPR, 2nd OIPR, 3rd OIPR, and 4th OIPR of an execution unit that the next instruction is dispatched to, cycle 3 Fig. 9. This next instruction is an Add permute move word to Row 1 (AP1) instruction which produces a rounded $(X_{3r}W_{3r} - X_{3i}W_{3i})$ 16-bit $T_r = ((X_3W_3)_r$ result and a rounded $(X_{3r}W_{3i} + X_{3i}W_{3r})$ 16-bit $T_i = (X_3W_3)_i$ result, cycle 4 Fig. 9. The AP1, Apm1.Row1, instruction is encoded with an indication that the 32-bit combined $(T_r, T_i)$ result is directed to the 1st OIPR of the subtract Sb1 in Row 3 and 2nd OIPR of the add Ad1 in Row 1. The 2nd OIPR of the Sb1 in row 3 comes from execution of an AP1 instruction in the Row 1 CEP.

The hardware then directs a result of executing the Row 3 Sb1 instruction to the 1 st OIPR of the MQ2 execution unit, cycle 5 Fig. 9. Using the $T$ value previously loaded, the MQ2 execution generates four 32-bit results which are directed to four OIPRs of the AP2 execution unit, cycle 6 Fig. 9. The AP2 unit directs a 32-bit combined $(T_r, T_i)$ result to the 1 st OIPR of the subtract Sb2 in Row 3 and 2nd OIPR of the add Ad2 in Row 2, cycle 7 Fig. 9. The 2nd OIPR of the Sb2 unit in Row 3 comes from the execution of the permute move (PV) in Row 2. A NOP instruction in rows 0-2 causes no operation in those rows for one cycle. The Sb2 execution generates a result which is directed to the 1 st OIPR of a store instruction (Si.a2) execution unit, cycle 8 Fig. 9. The execution of the store instruction stores the result in processor memory at a specified address, cycle 9 Fig. 9. In steady state, the instruction execution sequence of cycles 8 and 9 can be repeated as needed to compute length 4 FFTs every 2 cycles. Cycle 8 corresponds to the second row and cycle 9 corresponds to the first row shown in the PIC memory of Fig. 4.

TABLE II
ADJACENCY AND DIAMETER COMPARISON

| Adjacency | diameter 1 | diameter 2 | diameter 2 |
|---|---|---|---|
| 1 to 3 Torus | **3x3** | **5x5** | **5x5x5** |
| 1 to 5 Torus | **5x5** | **9x9** | **9x9x9** |
| 1 to 7 Torus | **7x7** | **13x13** | **13x13x13** |

In general, an FFT of length $N$ that is a multiple of $P$ is computed as $P$ FFTs of length $N/P$ in parallel, followed by $N(P-1)/P$ complex multiplications, and finally $N/P$ FFTs of length $P$ [23], [24], where $P$ represents a row CEP processor, as shown in Figs. 5 and 8 having 4 rows. The number of cycles for the length $N/P$ FFT is approximately, $4.5N/P\log_2(N/P)/10$, assuming that each processor can complete one complex multiplication and two complex additions per pipeline cycle. The $N(P-1)/P$ complex multiplications can be completed in $N(P-1)/P^2$ cycles. The final $N/P$ FFTs of length $P$, are completed in $N/P\log_2(P)$ cycles. For example, when $N = 256$, and $P = 4$, the AM processor requires $(172 + 48 + 128) = 348$ cycles in steady state.

Other algorithms map similarly to CEP execution, including bitonic sort, reductions, and prefix scans.

## VII. CONCLUSION

Our experienced engineering team has previously led the design and development of an advanced ManArray processor architecture, a corresponding simulator, FPGA model, and award-winning functional silicon (Manta) [25], [26]. The designs presented here illustrate a multi-dimensional memory network that can be embedded within a multi-processor architecture suitable for implementation using 3D techniques. A listing of a number of network adjacency organizations using the same adjacency in each dimension and associated properties is shown in Table II. Diameter 1 is for a network having direct $P$ node to $P$ node paths, while the diameter 2 columns are directed to the array memory network and uses store and load instructions to communicate. Ongoing research and development work, includes further architecture definitions of the memory nodes, network nodes, execution nodes, and array memory network modeling. For example, the 3D WS network, such as a $4\times4\times4$, may be folded according to 3D planes with the processor and memory elements overlaid to produce a $4 \times 4$ organization of quad core nodes [18]. Each quad core node comprises four processors rows, each as shown in Figs. 4 and 5, and their corresponding memory nodes. Another aspect of our research is to focus on further parallelization of FFTs, having each P in Fig. 2 implemented as the 4-row by 3-column execution array of Fig. 5, and computing the FFT example of Figs. 8 and 9 in parallel. Functional simulations are under construction using HDL and FPGA technologies.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

## REFERENCES


[1] T. yun Feng, "A survey of interconnection networks," *Computer*, vol. 14, no. 12, pp. 12–27, Dec 1981.

[2] K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*. McGraw-Hill, 1984, ch. Structures and Algorithms for Array processors, pp. 332–354.

[3] G. G. Pechanek, S. Vassiliadis, and J. G. Delgado-Frias, "Digital neural emulators using tree accumulation and communication structures," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 934–950, Nov 1992.

[4] G. G. Pechanek, S. Vassiliadis, and N. P. Pitsianis, "ManArray Interconnection Network: An Introduction," in *Proceedings of EuroPar '99 Parallel Processing*, ser. Lecture Notes in Computer Science, vol. 1685, Toulouse, 1999, pp. 761–765.

[5] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Comput. Surv.*, vol. 38, no. 1, Jun. 2006.

[6] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L. S. Peh, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, Sept 2007.

[7] E. Salminen, A. Kulmala, and T. D. Hamalainen, "Survey of network-on-chip proposals," 2008, white Paper.

[8] K. Choi and W. S. Adams, "Vlsi implementation of a 256 times;256 crossbar interconnection network," in *Proceedings Sixth International Parallel Processing Symposium*, Mar 1992, pp. 289–293.

[9] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Array, Trees, Hypercubes*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992, ch. Section 3.2 The Buterfly, Cube-Connected Cycles.

[10] C. Clos, "A study of non-blocking switching networks," *The Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, March 1953.

[11] V. E. Beneš, Ed., *Mathematical Theory of Connecxting Networks and Telephone Traffic*. Academic Press, 1965.

[12] A. Youssef and B. Arden, "Topology of efficiently controllable Banyan multistage networks," in *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing 1990*, Dec 1990, pp. 79–86.

[13] G. G. Pechanek, "Processor composed of memory nodes that execute memory access instructions and cooperate with execution nodes that execute function instructions," US Patent 7,581,079, 08 25, 2009.

[14] ——, "Interconnection network and method of construction thereof for efficiently sharing memory and processing in a multi-processor where connections are made according to adjacency of nodes in a dimension," US Patent 7,886,128, 02 8, 2011.

[15] ——, "Interconnection networks and methods of construction thereof for efficiently sharing memory and processing in a multi-processor where connections are made according to adjacency of nodes in a dimension," US Patent 8,156,311, 04 10, 2012.

[16] ——, "Interconnection network connecting operation-configurable nodes according to one or more levels of adjacency in multiple dimensions of communication in a multi-processor and a neural processor," US Patent 8,443,169, 05 14, 2013.

[17] ——, "Methods and apparatus for creating and executing a packet of instructions organized according to data dependencies between adjacent instructions and utilizing networks based on adjacencies to transport data in response to execution of the instructions," US Patent 9,460,048, 10 04, 2016.

[18] ——, "Methods and apparatus for signal flow graph pipelining that reduce storage of temporary variables," US Patent 9,507,603, 11 29, 2016.

[19] ——, "Methods and apparatus for signal flow graph pipelining in an array processing unit that reduces storage of temporary variables," US Patent Application Serial No. 15/238,429,, 08 16, 2016.

[20] J. S. Miguel, J. Albericio, A. Moshovos, and N. E. Jerger, "Doppelgänger: A cache for approximate computing," in *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2015, pp. 50–61.

[21] S. R. Walton, "Fast Fourier Transforms on the Hypercube," Ametek Computer Research Division, 610 N. Santa Anita Ave., Arcadia, CA 91006, Tech. Rep., September 1986.

[22] P. Swarztrauber, "Multiprocessor FFTs," *Parallel Computing*, vol. 5, no. 1-2, pp. 197–210, 1987.

[23] C. F. Van Loan, *Computational frameworks for the Fast Fourier Transform*. SIAM, 1992.

[24] N. P. Pitsianis, "The Kronecker product in approximation and fast transform generation," Ph.D. dissertation, Cornell University, 1997.

[25] "New Halo C Compiler from BOPS gets excellent results from EEMBC Benchmarking," http://www.design-reuse.com/news/630/new-halo-c-compiler-bops-gets-excellent-eembc-benchmarking.html, 2001.

[26] "In-Stat/MDR Announces BOPS, Inc. and Intel as Recipients of Microprocessor Reports Analysts Choice Award," https://www.thefreelibrary.com/Cahners+In-Stat\%2fMDR+Announces+BOPS+Inc.+and+Intel+as+Recipients+of...-a083526393, 2001.


978-1-5386-3472-1/17/$31.00 ©2017 IEEE