Digital Signal Processing at 1GHz in a Field-Programmable Object Array

Dirk R. Helgemo

***Abstract*—Autonomous MAC and ALU processors and register files (three types of Silicon Objects) are implemented with custom logic to achieve 1GHz fixed-point multiply and accumulate. Synchronous programmable interconnect and embedded storage reduces the need for difficult index calculation and the use of external memory for intermediate values. The flexibility of the objects and their interconnect allows the level of parallelism to be chosen freely based on performance requirements and resource constraints. Arraying hundreds of objects in parallel in a single chip enables incredible DSP performance from a flexible, in-circuit reprogrammable architecture.**

For example, a 1024-point FFT with (16+16)-bit complex samples can be completed every 160 clock cycles (i.e., every 160 nanoseconds) using 64 butterflies (128 MAC, 128 ALU, and 64 RF objects) assisted by 128 ALU and 64 RF objects for inter-stage data routing.

***Index Terms*—Digital Signal Processing (DSP), Application-Specific Programmable Product (ASPP), Reconfigurable Architecture, Field-Programmable Object Array (FPOA).**

I. INTRODUCTION

MathStar is offering a massively parallel high-performance computation fabric. Individual processing units, called Silicon Objects, are programmed individually and act autonomously. Each object is less than 400x400 micrometers square, implemented in custom logic, allowing hundreds of high-speed objects to be tiled on a single chip. Silicon Objects and their interconnect are programmed to construct computation macro blocks – composing simple scalar operations (addition, multiplication, logic, storage) into complex functions (e.g., 1024-point FFT). Interconnect and instructions are configured after fabrication via PROM, resulting in a field-programmable object array (FPOA). All communication and processing is synchronized to a global clock (up to 1GHz), removing the design issue of analog timing closure altogether.

II. SILICON OBJECT COMMUNICATION

Silicon Objects communicate via 21-bit buses composed of the following: sixteen bits of data, one bit indicating the validity of the data (e.g., for event-driven programming), and four bits of user-defined side-band control signals.

Communication proceeds synchronously and cooperatively. Buses are driven directly by registers (i.e., no intervening logic) for the most aggressive digital timing between objects. Values of interest are read by a cooperating receiving object; thus data is pulled rather than pushed through the architecture. Objects synchronize to the same digital clock cycle (phase) via user programming of control signals and/or data patterns.

The communication topology is a hybrid: objects can read registers from adjacent neighbors, or from any distance via pipelined "party lines." Neighbor registers in diagonal and Manhattan directions are observed with no latency (the same as local registers). Party lines can turn, pass, land, and/or launch at every object hop. The land/launch combination can be chosen to insert a pipeline delay and restore digital coherency, thereby enabling communication at any distance (at the expense of latency and party line landing registers). The communication topology thus facilitates the programming of high-speed computation kernels of arbitrary size and shape.

III. SILICON OBJECT TYPES

Whereas the communication infrastructure across a given fabricated Silicon Object array is uniform, the silicon implementation of each element can be unique, yielding a heterogeneous array. The following are available element implementations, known as Silicon Object types.

A. *Multiply-Accumulate Object (MAC)*

The MAC object type accepts two 16-bit signed integer inputs every clock cycle, multiplies them together, and adds or subtracts the product into the 32-bit accumulated result. The accumulator can be configured either to saturate, or to wrap into an 8-bit over-/underflow counter, tolerating up to 40-bit intermediate results. The entire accumulator is visible on object outputs and can be reset to zero or reloaded per control inputs, allowing either a new sequence to be started or a paused sequence to be resumed. The operation consumes fresh inputs and generates a result every clock cycle, with a processing latency of two clock cycles.

B. *Arithmetic-Logic Unit Object (ALU)*

The ALU is the most general-purpose object type. It employs a 16-bit add, shift, and logic operator controlled by an 8-instruction state machine. Each instruction selects up to three 16-bit input words and a carry input bit, configures the operator (a.k.a. opcode), selects one or more result destination registers, and specifies conditional execution and branching options. This object type contains nine working registers (four for neighbors, five for party lines); two programmable constant registers; and two wired constants. Thus there are twenty-one possible inputs and nine possible outputs. In a single clock cycle, the current instruction is fetched and decoded, the operator is executed, the result is stored (subject to conditional execution), and the next instruction is selected per branching.

Unclassified manuscript sent May 30, 2003, to the HPEC 2003 Conference.

Dirk R. Helgemo is Chief Architect of MathStar, Inc., 5900 Green Oak Drive, Minneapolis, MN 55343; phone 952-746-2200; fax 952-746-2201; email Dirk.Helgemo@MathStar.com.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
20 AUG 2004	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Digital Signal Processing at 1GHz in a Field-Programmable Object Array	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **MathStar, Inc.**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001694, HPEC-6-Vol 1 ESC-TR-2003-081; High Performance Embedded Computing (HPEC) Workshop (7th)., The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**33**	

C. Register File (RF)

The RF object type provides fast storage within the array. Up to two 20-bit values can be read and two 20-bit values written simultaneously every clock cycle, with an access latency of two clock cycles. Storage capacity is 64 20-bit words, also configurable as 32 40-bit words.

The read and write ports can each be configured for random or sequential access. Thus, an RF can be configured as a dual-port RAM, a FIFO, or random-write sequential-read. The last combination, also known as "sort" mode, allows values to be written in an arbitrary index order, but then retrieved as a sequence without the burden of address generation. That is, values can be written to arbitrary addresses in anticipation of the order in which they will be read out.

IV. Fast Fourier Transform (FFT) via Objects

A. Complex Multiplication

Two MAC objects can be efficiently ganged to multiply two complex numbers. Four products are generated, two of which are differenced, two of which are summed. Thus two MAC objects can generate a complex result every two clock cycles, with a latency of three clock cycles.

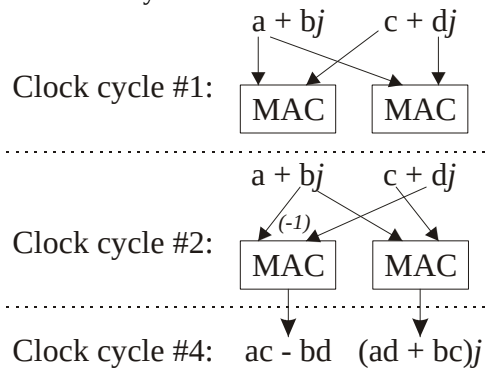


Figure 1: Complex Multiplication

B. Radix-2 Butterfly

The butterfly kernel within the FFT algorithm accepts two complex numbers from a previous FFT stage, multiplies one of the inputs with a twiddle factor (a complex constant), and performs a complex sum and difference, yielding two complex numbers for the next FFT stage.

Consider the implementation of a decimation-in-time butterfly: $out_1 = in_1+W^k in_2$, $out_2 = in_1-W^k in_2$, where W^k is the complex twiddle factor. Due to the predictability of the twiddle factors, they are precalculated and stored into an RF object configured for sequential read mode – and address generation is not required.

Thus the butterfly algorithm is as follows: 1. Fetch the precalculated complex W^k from the RF object. 2. Multiply $W^k in_2$ via two MAC objects (as described above). 3. Use two ALU objects to both sum and difference the complex product against in_1 to generate outputs out_1 and out_2, respectively, over the next two clock cycles. Thus, butterfly outputs can be calculated every two clock cycles, with a latency of five clock cycles.

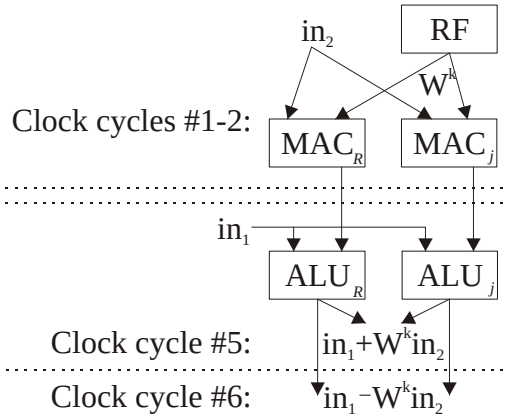


Figure 2: Decimation-In-Time Butterfly

C. Fast Fourier Transform (FFT)

Each stage of butterflies chooses a different pairing of the previous stage's results (as well as different twiddle factors) until all of the FFT inputs affect all of the FFT outputs (i.e., 2^n points require n stages). While a single butterfly can be leveraged to any size FFT, multiple parallel instantiations of the butterfly (in powers of two) increase the theoretical computational performance dramatically:

# Butterflies	MACs	ALUs	RFs	Rate	Latency
1	2	2	1	$1/(n2^{n+1})$	$3+n2^n$
2	4	4	2	$1/(n2^n)$	$3+n2^{n-1}$
$\mathbf{2^{n-2}}$	2^{n-1}	2^{n-1}	2^{n-2}	1/4n	3+4n
$\mathbf{2^{n-1}}$	2^n	2^n	2^{n-1}	1/2n	3+2n
$\mathbf{n2^{n-1}}$	$n2^n$	$n2^n$	$n2^{n-1}$	1/2	3+2n

Figure 3: Butterfly Parallelism for 2^n-point FFT (n stages) (Rate is results per clock cycle. Latency is clock cycles.)

Fortunately, practical performance does not substantially lag the theoretical ceiling. Butterflies are kept 100% utilized by providing two new complex inputs every two clock cycles. Either an RF object or two ALU objects can sustain this bandwidth indefinitely. The trick lies in efficient transitions between FFT stages.

Every butterfly result is used precisely twice as an input into the next stage. Therefore, the butterfly results (two complex result every two clock cycles) are routed via ALU objects (with stage-specific directions) toward the two butterflies for the next FFT stage. An RF object sorts the complex data values into the correct order for the next stage using a nearby ALU object to generate stage-specific write addresses into the RF object.

Performance is lost between stages only if the RF object cannot be loaded in time to start the next stage. In practice, index analysis of the data dependencies between stages allows the next stage to be started while the previous stage completes. (Ironically, fully parallelized butterflies cannot avoid stalling between FFT stages because none of them can start until the previous stage completes.)

V. Results

A 1024-point FFT with (16+16)-bit complex samples can be completed every 160 nanoseconds using 64 butterflies (128 MAC, 128 ALU, and 64 RF objects) assisted by 128 ALU and 64

RF objects for inter-stage data routing. An array of 25x25 objects provides the required number and arrangement (with over 100 objects remaining for control sequencing), yet fits within a 10x10 millimeter square of silicon. Note that the object commonality allows larger, smaller, and different mixes of object types, I/O, and on-chip RAM to be readily constructed according to specific application requirements.

Digital Signal Processing at 1GHz in a Field-Programmable Object Array

Dirk Helgemo
Chief Architect
MathStar, Inc.

Contents

- Driving Philosophy
- Architecture
  - Communication
  - Object Types
- DSP Algorithms in Objects
- Tools
- Applications
- Roadmap

Driving Philosophy

- **FPGA time to market**
  - Programmable/configurable silicon
- **Lower unit cost than FPGA**
  - Coarser programming ☪ higher density
- **ASIC-like performance (1GHz)**
  - Custom logic
- **Lower risk and easier design**
  - All analog problems are solved (timing, place & route)
  - Just digital design (program = resource allocation)
  - Use proven COTS chips with adequate resources or
  - Assemble custom chips with very low risk

Decisions

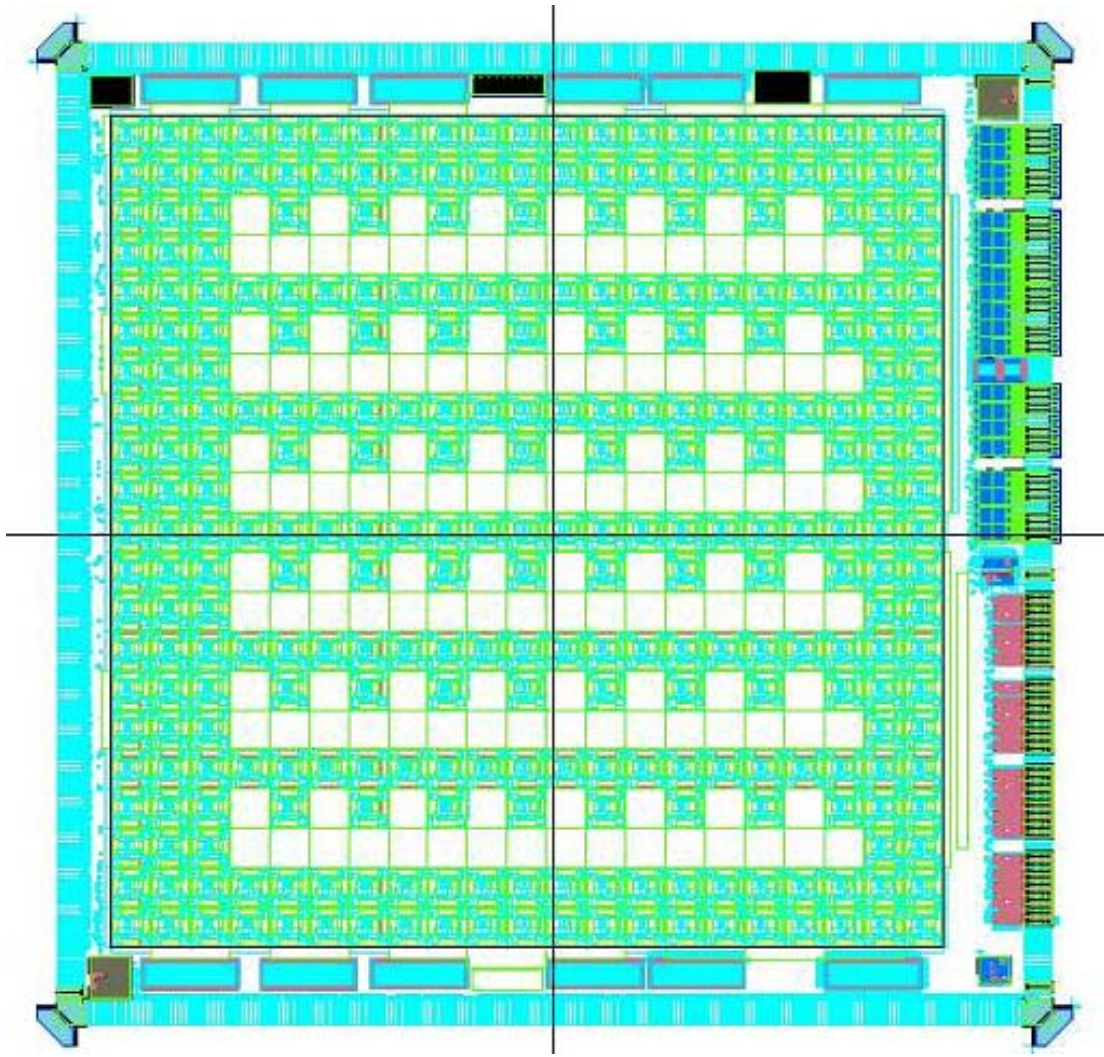
- **Everything is globally synchronized**
  - **No analog timing closure!**
- **Configured instructions (instead of streaming)**
  - **Massive parallelism without massive instruction buses**
- **Uniform interconnect and object size**
  - **Mix and match functions for different application spaces**
  - **Scripted object placement, power, clocking**

Architecture

- Package functions into Silicon Objects (SOs)
  - Homogeneous communication
  - Heterogeneous functions
    - Processors, memory, I/O
- Tile objects into an array
  - Choose the mix of functions (including I/O) to match the application space
    - Lots-o-multipliers for DSP FFT and FIR
    - Add high-speed I/O and CAM processors for networking
- Fabricate the object mix
- Program the application

Sample Mix

- 21*21 = 441 SOs
  - 6*16 = 96 MAC
  - 6*8 = 48 RF
  - rest = 297 ALU
- Periphery
  - 12*7KB int. RAM
  - 2*72b ext. RAM
  - 2*16b LVDS
  - 192 GPIO

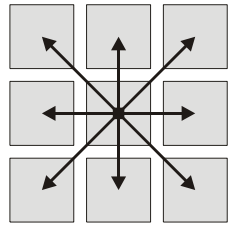


Communication

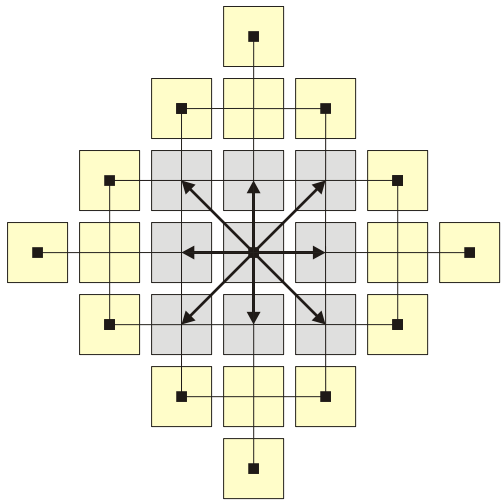
- **Uniform bus structure: 21 bits**
  - **16-bit data value (R)**
  - **1-bit "valid" indicator (V)**
  - **4 bits of control (C)**
- **Configuration granularity**
  - **R+V are handled as a unit**
  - **Each C bit is configured independently**
- **Usage**
  - **V can be used for event-driven (wave)**
  - **C provides arbitrary sideband control**
    - **Examples: sign, carry, start of packet**

Communication Routing

- **Nearest Neighbors (NN)**
  - Range = 1 (Manhattan + diagonals)
  - Same speed as local registers



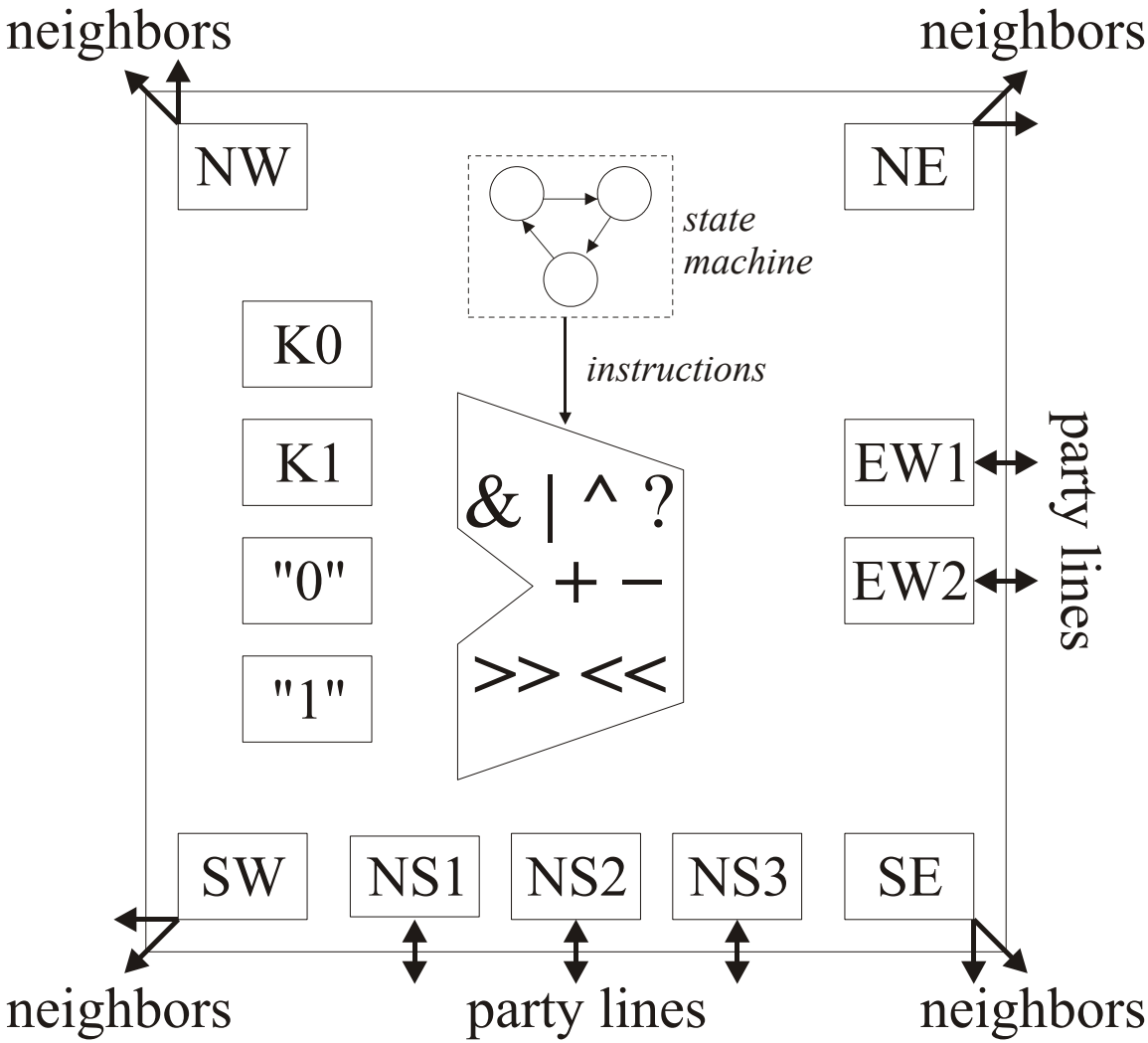
- **Party Lines (PL)**
  - Range = Manhattan hop to 3 (skip 2)
  - Extra clock cycles for digital retiming
    - 1 extra ☪ 25-object neighborhood
    - 2 extra ☪ 85-object neighborhood
    - More clock cycles ☪ entire chip



Silicon Object Types

- Arithmetic/Logic Unit (ALU)
- Multiply-Accumulate (MAC)
- Register File (RF)
- Truth Function (TF)
- CRC Generator (CRC)
- Pattern Processor (CAM)
- Internal RAM (IRAM)
- External RAM (XRAM)
- General-purpose I/O (GPIO)
- High-speed parallel I/O (Rx, Tx)

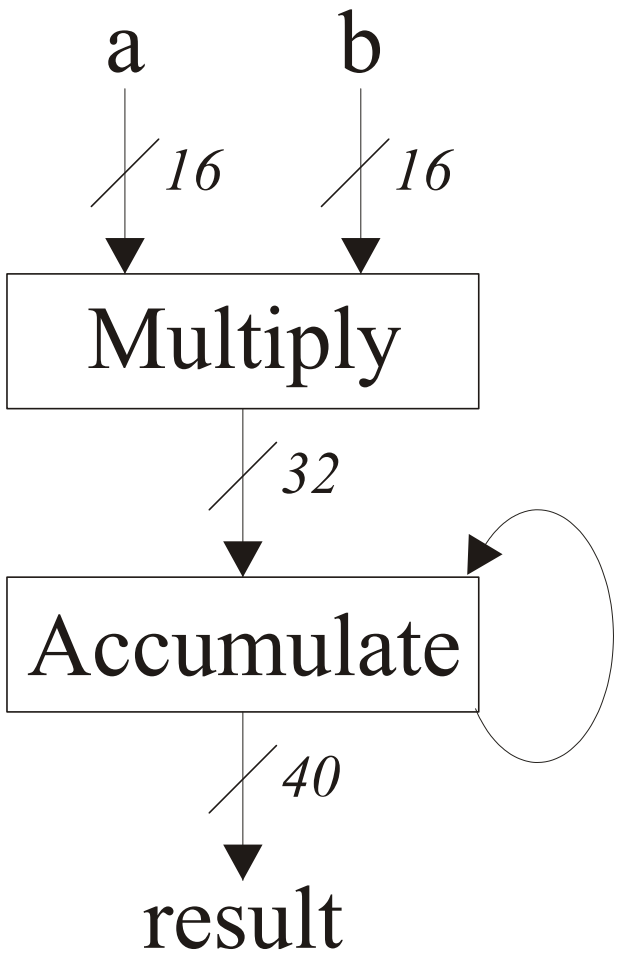
Object Type: ALU



ALU Details

- **Arithmetic-Logic Unit**
  - **16-bit data path**
    - **Add/subtract, shift/rotate, AND/OR/XOR/mux**
    - **Cascade larger words via status bit (SB)**
  - **Decode, execute, retire in 1 cycle (1 ns)**
  - **8 configured instructions per object**
  - **State is guided by control inputs**
    - **Expressions of up to four C/V/SB/R bits**
    - **Instruction offers four "next states"**
    - **Branch expression selects one of the four**
    - **Additional controls for conditional execution**

Object Type: MAC



MAC Details

- **Multiply-accumulate**
  - **16x16 fixed-point multiplication**
  - **40-bit accumulator (8-bit overflow)**
  - **Rate = every cycle, latency = 2 cycles**
    - **100 products in 101 cycles**
  - **Number formats: integer (16.0) and Q15 (1.15)**
  - **Signed and unsigned multiplication**
    - **Extended precision (32x32=64) in four MACs**
  - **Control bit inputs effect optional negation, accumulation, rounding**
  - **8-bit embedded counter (inner loop)**

Object Type: RF

- **Register File is a fast, small memory:**
  - **64 words of 20 bits (16R+4C)**
  - **Three modes of operation**
    - **Dual-ported RAM**
    - **FIFO**
    - **Sort: random write, sequential read**
  - **More control inputs to request read, request write**
  - **More control outputs indicate read valid, FIFO status**
  - **Rate = every cycle, latency = 2 cycles**

Object Type: TF

- **Truth Function generates four C bits**
  - **Four C/V/SB/R input bits per C bit output**
  - **Arbitrary functions via 4:1 lookup tables**
  - **Cascade large control expressions across multiple objects**
  - **Rate = every cycle, latency = 1 cycle**
- **Integrate TF with ALU object**
  - **ALU-TF is most general purpose**
  - **Fine-grained control for state machines and flow control (span clock domains, etc.)**

Object Type: CRC

- CRC = cyclic redundancy code generator
  - Single-cycle CRC-32 and CRC-16
  - Processes 8, 16, or 18 bits of data per clock
    - 18b for HyperTransport
  - Rate = every cycle, latency = 3 cycles
- Integrate with RF object
  - CRC is a very small circuit
  - Choose RF or CRC function
  - Span applications gracefully
    - Applications with no CRC are not impeded
    - Capacity for applications needing many CRCs (e.g., multichannel POS Ethernet)

Object Type: CAM

- CAM = pattern recognition
  - Input 20C or 16R+4C bits
  - Sixteen 20-bit patterns with wildcards
    - Each pattern bit is 0/1/x (x=wildcard)
  - On row match, indicate “hit” on V, update 20-bit result
  - Output 20C or 16R+4C bits
  - Rate = every cycle, latency = 2 cycles
  - Uses:
    - Bit-field parsing (variable- or fixed-width fields)
    - State machines (up to 16 transitions)

Object Types: IRAM, XRAM

- IRAM = Internal RAM
  - Single-ported block RAM
  - Spans two object columns, north or south
    - Address and control via pl_ns3
    - Data in/out via pl_ns1, pl_ns2
  - Capacity = 768 lines of 76 bits = 57Kb = 7.125KB
  - Rate = read or write at 500MHz, latency = 9 cycles
- XRAM = External RAM
  - Single-ported SRAM or DRAM memory controller
  - Same north/south object interface as IRAM (above)
  - 72-bit data path * 21-bit address = 144Mb = 18MB
  - Up to 250MHz DDR = 18Gb/s throughput

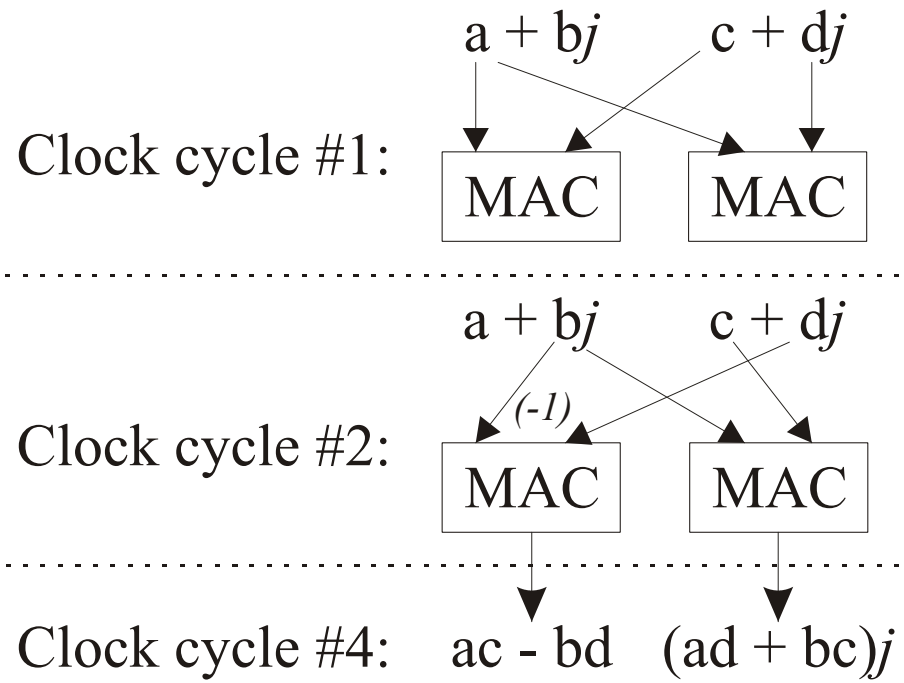
Object Types: GPIO, Rx/Tx

- GPIO = General-purpose I/O
  - 2.5V CMOS, up to 100MHz
  - Synchronized internally or externally
  - 48 read/write pins to 2 object columns (or rows)
    - 32 to R, 16 to C, configurable
- Rx,Tx = High-speed parallel I/O
  - Configurable for 16-bit LVDS or 32-bit HSTL
    - Up to 800MHz DDR LVDS (25Gb/s)
  - Receive into 2,4,8 object rows (configurable demux)
  - Transmit out of 2,4,8 object rows (configurable mux)

DSP Algorithms in Objects

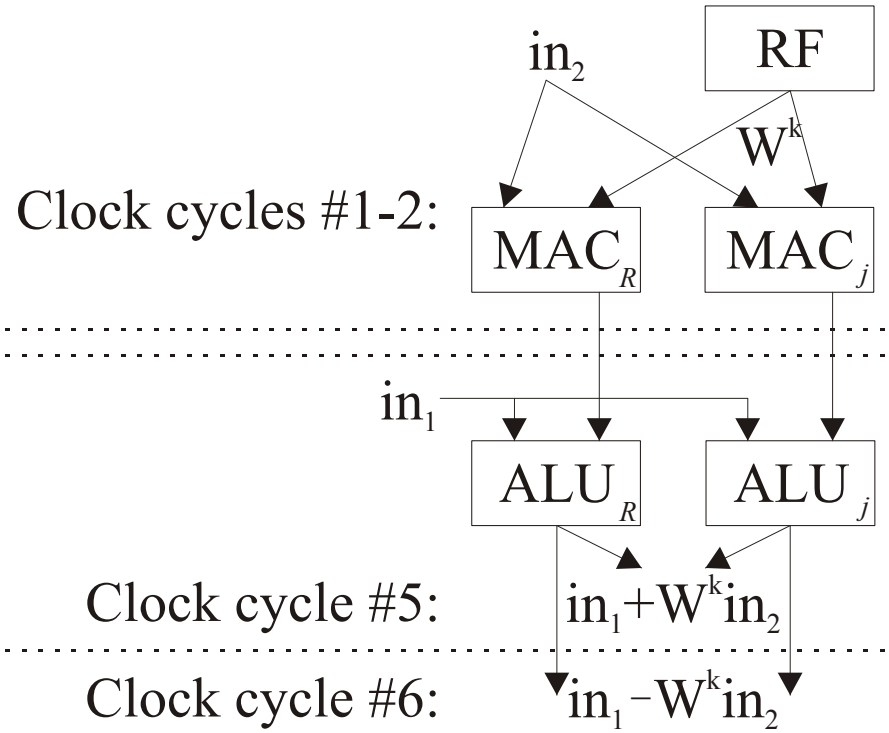
- Complex Multiplication
- Radix-2 DIT Butterfly
- Radix-4 DIF Dragonfly
- Fast Fourier Transform (FFT)

Complex Multiplication



- **Two MACs: one real, one imaginary**
- **Rate = every other cycle**
- **Latency = 3 cycles**

Radix-2 DIT Butterfly



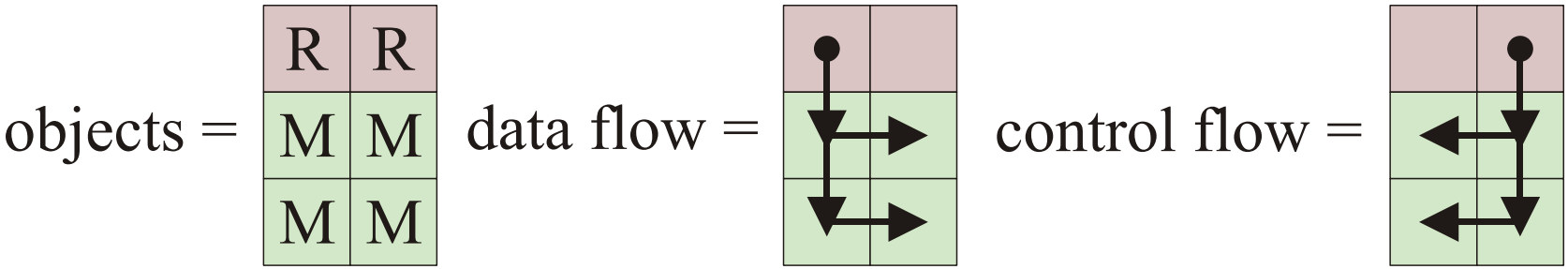
- **2 MACs, 2 ALU, 1 RF (W^k phase factors)**
- **Rate = every other cycle**
- **Latency = 5 cycles**

Radix-4 DIF Dragonfly

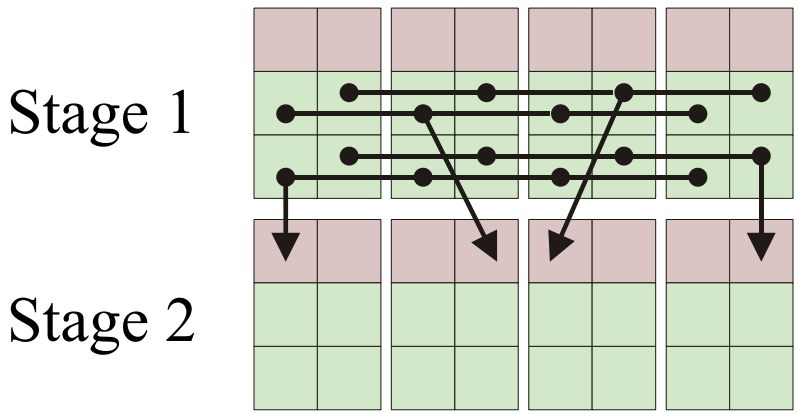
- Data = 3 sets of 4 complex numbers
  - Input values, phase factors (twiddle), output values
- Algorithm (roughly)
  - Output.r,i = Σ (+/- phase.r/i) * input.r,i = Σ 8 products
    - Sequence of sign and phase.r vs. phase.i varies for each output
- Processors = 4 MACs (one per output), 2 RFs
  - Each MAC calculates out.real then out.imaginary
    - Route the complex output value to RF in next stage
  - One RF streams the 4 complex inputs twice (8 integers)
  - Other RF sends control sequence (16 clock cycles)
    - Start (zero), choose positive/negative, choose phase.r/phase.i

Dragonfly in Pictures

- **Structure of one dragonfly tile**

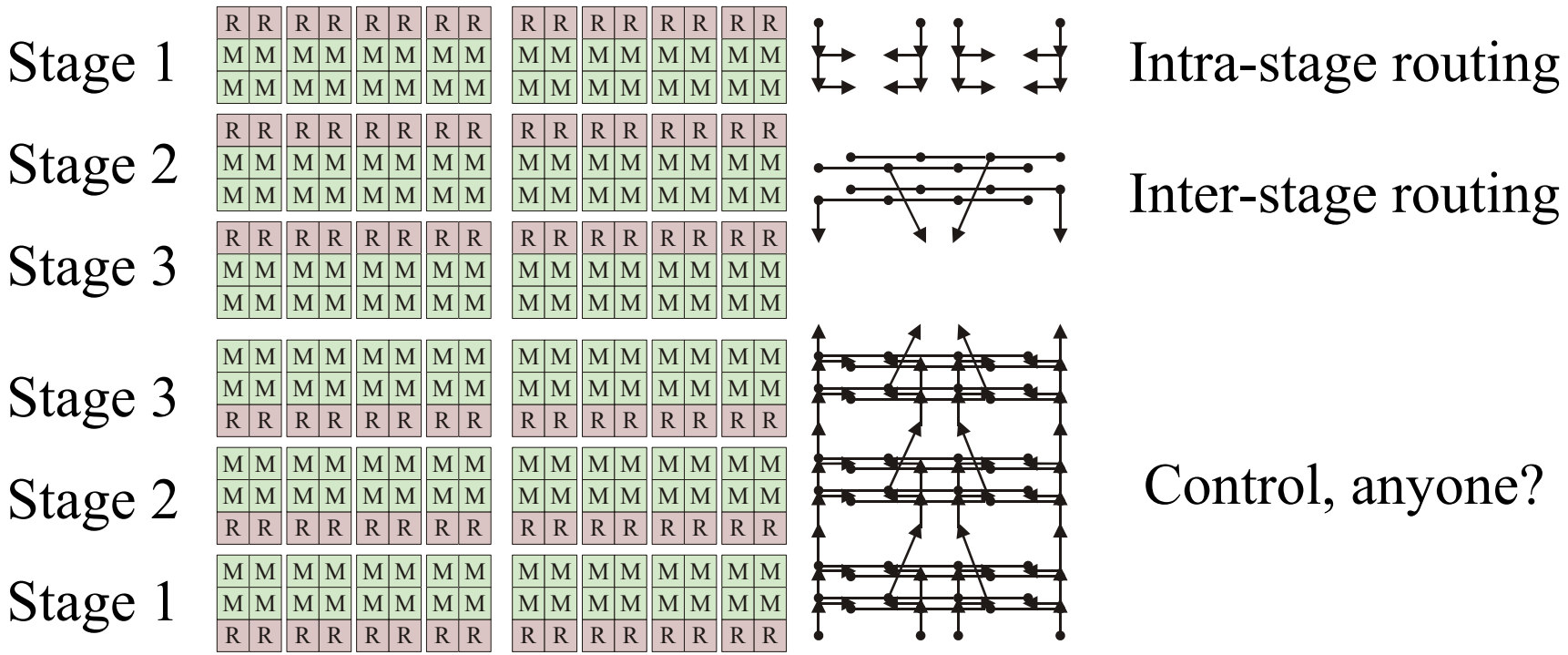


- **Inter-dragonfly (inter-stage) routing**



64-point FFT

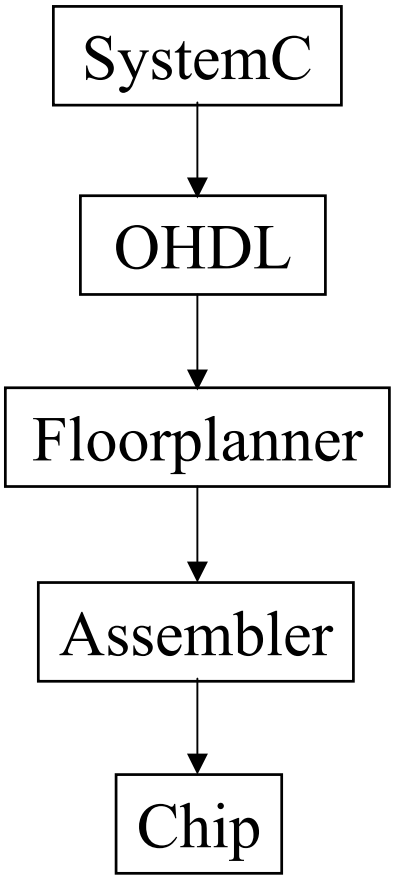
- **Fully pipelined ☪ 16 ns throughput**
  - **16 cycles per dragonfly, 48 pipelined dragonflies**
  - **Out-of-order input and output**



1024-point FFT

- 1024-point FFT in 160ns
  - 64 butterflies (128 MAC, 128 ALU, 64 RF)
  - Several options for data movement between butterfly stages
    - Many DSP solutions use memory for data routing
    - FPOA has a variety of options
      - Use party lines to route: two options per hop, add as many levels of indirection as needed
      - Use ALUs to route: four NN and four PL options per ALU, add as many levels of indirection as needed
      - Use ALUs to track stride of each butterfly stage, generate address into RF or IRAM
      - Store address sequence in an RF or IRAM

Tools



- **Object HDL (OHDL) is the assembly language for the chip configuration**
  - **Verilog structural modules and wires**
  - **Object-specific assembly**
- **Design in SystemC (translates to OHDL) or code directly in OHDL**
  - **Cycle-accurate simulation either way**
- **Assign chip resources via Floorplanner GUI**
- **Compile to bit stream via Assembler**

Applications

- General-purpose mix
  - Processors = ALU-TF, RF
  - Periphery = IRAM, XRAM, GPIO
- DSP FFT and FIR
  - Processors = ALU-TF, MAC, RF
  - Periphery = Narrow IRAM, Narrow XRAM, GPIO and/or LVDS
  - Future processor: FEC
- Networking
  - Processors = ALU-TF, CAM, RF-CRC
  - Periphery = Wide IRAM, Wide XRAM, LVDS, SerDes

Roadmap

- First chip is a mixed mix
  - Demonstrate both DSP and networking applications
    - MACs for high-performance DSP FFT, FIR
    - ALU-TF and RF-CRC for both DSP and networking
    - 12 banks of IRAM (total 85.5KB)
    - One bi-directional 16-bit LVDS interface (one Rx, one Tx)
    - 192 CMOS GPIO pins (four GPIO objects)
- Next two chips are specialized
  - DSP FFT, FIR
    - More MACs, more fine-grained memory
  - Networking
    - SerDes I/O (4Gb/s), more bulk memory

Conclusions

- The "object" approach (FPOA) enables
  - High-speed programmable COTS silicon
    - 20x20 processors = 10x10mm die = 400G ops/s at 20W
  - Field upgrades via programming (PROM or JTAG)
    - Program is loaded into embedded SRAM
    - PROM can be AES-encrypted; FPOA can be copy-protected
    - Field debug via AES-authorized JTAG
  - High-performance alternative to FPGA
    - FPOA is more coarse-grained
      - Fewer "electron decisions" ☪ higher performance
  - Low-risk alternative to ASIC
    - Proven objects, just tile a new mix: Tape-out < 1 month!