Illiac IV History

- First massively parallel computer
  - SIMD (duplicate the PE, not the CU)
  - First large system with semiconductor-based primary memory
- Three earlier designs (vacuum tubes and transistors) culminating in the Illiac IV design, all at the University of Illinois
  - Logical organization similar to the Solomon (prototyped by Westinghouse)
  - Sponsored by DARPA, built by various companies, assembled by Burroughs
  - Plan was for 256 PEs, in 4 quadrants of 64 PEs, but only one quadrant was built
  - Used at NASA Ames Research Center in mid-1970s

Illiac IV

Illiac IV Architectural Overview

- One CU (control unit),
  64 64-bit PEs (processing elements),
  each PE has a PEM (PE memory)
- CU operates on scalars, PEs on vector-aligned arrays
  - All PEs execute the instruction broadcast by the CU, if they are in active mode
  - Each PE can perform various arithmetic and logical instructions
  - Each PE has a memory with 2048 64-bit words, accessed in less than 188 ns
  - PEs can operate on data in 64-bit, 32-bit, and 8-bit formats
- Data routed between PEs various ways
- I/O is handled by a separate Burroughs B6500 computer (stack architecture)

Programming Issues

- Consider the following FORTRAN code:

```
      DO 10 I = 1, 64
10    A(I) = B(I) + C(I)
```

  - Put A(1), B(1), C(1) on PU 1, etc.
    - Each PE loads RGA from base+1,
      adds base+2, stores into base,
      where "base" is base of data in PEM
    - Each PE does this simultaneously, giving a speedup of 64
  - For less than 64 array elements, some processors will sit idle
  - For more than 64 array elements, some processors might have to do more work
- For some algorithms, it may be desirable to turn off PEs
  - 64 PEs compute, then one half passes data to other half, then 32 PEs compute, etc.

The Illiac IV Array

- Illiac IV Array = CU + PE array
- CU (Control Unit)
  - Controls the 64 PEs (vector operations)
  - Can also execute instructions (scalar ops)
  - 64 64-bit scratchpad registers
  - 4 64-bit accumulators
- PE (Processing Element)
  - 64 PEs, numbered 0 through 63
  - RGA = accumulator
  - RGB = for second operand
  - RGR = routing register, for communication
  - RGS = temporary storage
  - RGX = index register for instruction addrs.
  - RGD = indicates active or inactive state

The Illiac IV Array (cont.)

- PEM (PE Memory)
  - Each PE has a 2048-word 64-bit local random-access memory
  - PE 0 can only access PEM 0, etc.
- PU (Processing Unit) = PE + PEM
- Data paths
  - CU bus — 8 words of instructions or data can be fetched from a PEM and sent to the CU (instructions distributed in PEMs)
  - CDB (Common Data Bus) — broadcasts information from CU to all PEs
  - Routing network — PE *i* is connected to PE *i* -1, PE *i* +1, PE *i* -8, and PE *i* +8
    - Wraps around, data may require multiple transfers to reach its destination
  - Mode bit line — single line from RGD of each PE to the CU

Illiac IV I/O System

- I/O system = I/O subsystem, DFS, and a Burroughs B6500 control computer
- I/O subsystem
  - CDC (Control Descriptor Controller) — interrupts the Burroughs B6500 upon request by the CU, loads programs and data from the DFS into the PEM array
  - BIOM (Buffer I/O Memory) — buffers (much faster) data from DFS to CPU
  - IOS (I/O Switch) — selects input from DFS vs. real-time data
- DFS (Disk File System)
  - 1 Gb, 128 heads (one per track)
  - 2 channels, each of which can transmit or receive data at 0.5 Gb/s over a 256-bit bus (1 Gb/s using both channels)

Illiac IV I/O System (cont.)

- Burroughs B6500 control computer
  - CPU, memory, peripherals (card reader, card punch, line printer, 4 magnetic tape units, 2 disk files, console printer, and keyboard)
  - Manages requests for system resources
  - OS, compilers, and assemblers
  - Laser memory
    - 1 Tb write-once read-only laser memory
    - Thin film of metal on a polyester sheet, on a rotating drum
    - 5 seconds to access random data
  - ARPA network link
    - High-speed network (50 Kbps)
    - Illiac IV system was a network resource available to other members of the ARPA network

Illiac IV Software

- Illiac IV delivered to NASA Ames Research Center in 1972, operational sometime (?) after mid -1975
  - Eventually operated M–F, 60-80 hours of uptime, 44 hours of maintenance / downtime
- No real OS, no shared use of Illiac IV, one user at a time
  - An OS and two languages (TRANQUIL & GLYPNIR) were written at Illinois
  - At NASA Ames, since PDP-10 and PDP-11 computers were used in place of the B6500, new software was needed, and a new language called CFD was written for solving differential equations

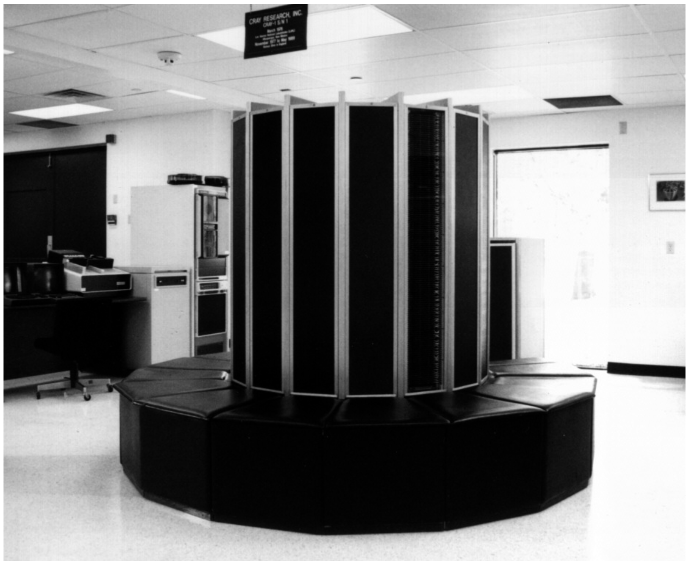
Cray-1 History

- In January 1978, a CACM article says there are only 12 non-Cray-1 vector processors worldwide:
  - Illiac IV is the most powerful processor
  - TI ASC (7 installations) is the most populous
  - CDC STAR 100 (4 installations) is the most publicized
- Recent report says the Cray-1 is more powerful than any of its competitors
  - 138 MFLOPS for sustained periods
  - 250 MFLOPS for short bursts
- Features: chaining (access intermediate results w/o memory references), small size (allows 12.5 ns clock = 80 MHz), memory with 1M 64-bit words

Cray-1 Physical Architecture

- Physical architecture
  - “World’s most expensive love-seat”
  - Cylindrical, 8.5’ in diameter (seat), 4.5’ in diameter (tower), 6.5’ tall (tower)
  - Composed of 12 wedge-like columns in 270° arc, so a “reasonably trim individual” can get inside to work
  - “Love seat” hides power supplies and plumbing for Freon cooling system
- Freon cooling system
  - In each chassis are vertical cooling bars lining each wall
  - Freon is pumped through a stainless steel tube inside an aluminum casing
  - Modules have a copper heat transfer plate that attaches to the cooling bars
  - 70F tube temp = 130F center of module

Cray-1



Cray-1 Architecture

- Computer architecture
  - 12 I/O channels, 16 memory banks, 12 functional units, 4KB of register storage
  - Only 4 chip types
  - Fast main memory, fast computation
- 4 chip types
  - 16x4 bit register chips (6 ns)
  - 1024x1 bit memory chips (50 ns)
  - Simple low- or high-speed gates with both a 5-wide and a 4-wide gate (5/4 NAND)
- Fabrication
  - 6"x8" printed circuit boards
  - ICs in 16-pin packages, up to 288 packages per board to build 113 different module types, up to 72 modules per 28-inch high chassis

Cray-1 Architecture (cont.)

- Memory (16 banks, 72 modules / bank)
  - 64 modules = 1 bit in 64 bit word
  - 8 modules = check byte for single-bit error correction, double bit error detection
- Functional units
  - 12 pipelined functional units in 4 groups: address, scalar, vector, and floating point
  - Scalar add = 3 cycles, vector add = 3 cycles, floating-point add = 6 cycles, floating-point multiply = 7 cycles, reciprocal approximation = 14 cycles
- Instruction formats
  - Either one or two 16-bit "parcels"
  - Arithmetic and logical instructions operate on 3 registers
  - Read & store instructions access memory

Cray-1 Registers

- Registers
  - 8 address registers (A), 64 address-save registers (B), 8 scalar registers (S), 64 scalar-save registers (T), & 8 64-word vector registers (V)
- 8 24-bit address registers (A)
  - Used as address registers for memory references and as index registers
  - Index the base register for scalar memory references, provide base address and index for vector memory references
  - 24-bit integer address functional units (add, multiply) operate on A data
- 64 24-bit address-save registers (B)
  - Used to store contents of A registers

Cray-1 Registers (cont.)

- 8 64-bit scalar registers (S)
  - Used in scalar operations
  - 64-bit integer scalar functional units (add, shift, logical, population/leading zero count) operate on S data
- 64 64-bit scalar-save registers (T)
  - Used to store contents of S registers, typically intermediate results of complex computations
- 8 64-element vector registers (V)
  - Each element is 64 bits wide
  - Each register can contain a vector of data (row of a matrix, etc.)
  - Vector Mask register (VM) controls elements to be accessed, Vector Length register (VL) specifies number of elements to be processed

Vector Arithmetic

- First, consider a vector on a SISD (non-parallel) machine
  - Vectors A, B, and C are each one-dimensional arrays of 10 integers
  - To add each corresponding value from A and B, storing the sum in C, would require at least 4 cycles, 40 cycles overall
  - If the CPU is a *vector processor*, loading, adding, and storing gets pipelined, so after a few cycles, a new value get stored into C each cycle, for 12 cycles overall, speedup of 40/12 = 3.33
  - The longer the vector, the more speedup
- Now consider a vector on a SIMD machine — each processor can do this vector processing in parallel
  - 64 processors => speedup of 213 over original computation!

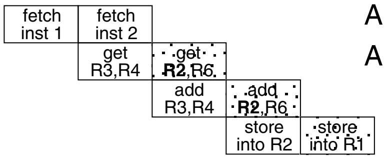
Chaining

- Vector operation operates on either two vector registers, or one vector register and one scalar register
- Parallel vector operations may be processed two ways:
  - Using different functional units and vector registers, or
  - By *chaining* — using the result stream from one vector register simultaneously as the operand set for another operation in a different functional unit
    - Intermediate results do not have to be stored in memory, and can even be used before a particular vector operation has finished
    - Similar to data forwarding in the IBM 360’s pipeline

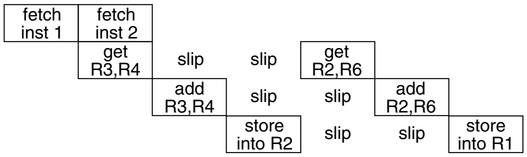
Handling Data Hazards

- *Write / read data hazard* example:

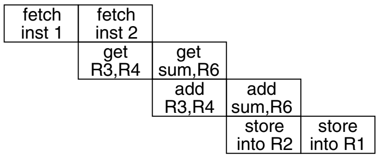
ADD R2, R3, R4
ADD R1, R2, R6



- Can be avoided with *register interlocks*



- Can also be avoided with *data forwarding*



Handling Data Hazards (cont.)

- Register interlocks
  - An instruction gets *blocked* until <u>all</u> its source registers are loaded with the appropriate values by earlier instructions
  - A “valid / invalid” bit is associated with each register
    - During decode stage, destination register is set to invalid (it will change)
    - Decode stage blocks until all its source (and destination) registers are valid
    - Store stage sets destination register to valid
- Data forwarding
  - Output of ALU is connected directly to ALU input buses
  - Result of an ALU operation is now available <u>immediately</u> to later instructions (i.e., even before it gets stored in its destination register)

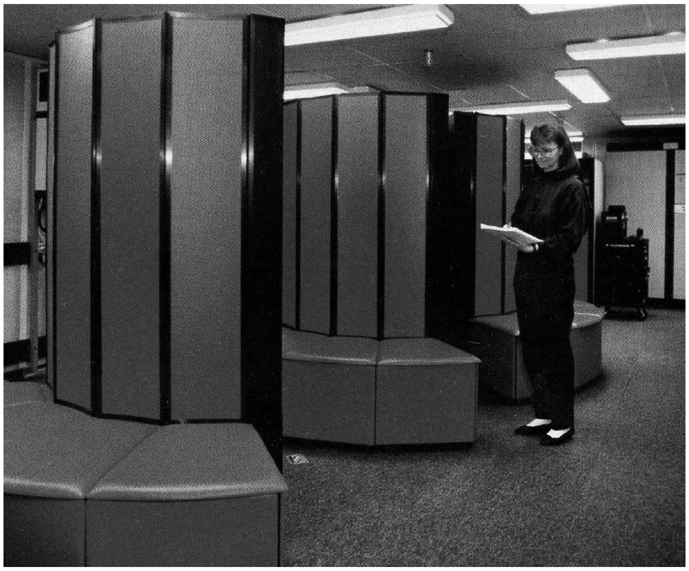
Miscellaneous

- Evolution
  - Seymour Cray was a founder of Control Data Corp. (CDC) and principal architect of CDC 1604 (non-vector machines)
  - 8600 at was to be made of tightly-coupled multiprocessors; it was cancelled so Cray left to form Cray Research
- Software
  - Cray Operating System (COS) — up to 63 jobs in a multiprog. environment
  - Cray Fortran Compiler (CFC) — optimizes Fortran IV (1966) for the Cray-1
    - Automatically vectorizes many loops that manipulate arrays
- Front-end computer
  - Any computer, such as a Data General Eclipse or IBM 370/168

Cray X-MP, Y-MP, and {CJT}90

- At Cray Research, Steve Chen continued to update the Cray-1, producing…
- X-MP
  - 8.5 ns clock (Cray-1 was 12.5 ns)
  - First multiprocessor supercomputer
    - 4 vector units with scatter / gather
- Y-MP
  - 32-bit addressing (X-MP is 24-bit)
  - 6 ns clock
  - 8 vector units
- C90, J90 (1994), T90
  - J90 built in CMOS, T90 from ECL (faster)
  - Up to 16 (C90) or 32 (J90/T90) processors, with one multiply and one add vector pipeline per CPU

Cray X-MP @ National Supercomputer Center in Sweden



Cray-2 & Cray-3

- At Cray Research, Steve Chen continued to update the Cray-1 with improved technologies: X-MP, Y-MP, etc.
- Seymour Cray developed Cray-2 in 1985
  - 4-processor multiprocessor with vectors
  - DRAM memory (instead of SRAM), highly interleaved since DRAM is slower
  - Whole machine immersed in Fluorinert (artificial blood substitute)
  - 4.1 ns cycle time (3x faster than Cray-1)
  - Spun off to Cray Computer in 1989
- Seymour Cray developed Cray-3 in 1993
  - Replace the “C” shape with a cube so all signals take same time to travel
  - Supposed to have 16 processors, had 1 with a 2 ns cycle time

Thinking Machines Corporation's Connection Machine CM-2

- Distributed-memory SIMD (bit-serial)
- Thinking Machines Corp. founded 1983
  - CM-1, 1986 (1000 MIPS, 4K mem / proc)
  - CM-2, 1987 (2500 MFLOPS, 64K…)
- Programs run on one of 4 front-end processors, which issues instructions to the Parallel Processing Unit (PE array)
  - Control flow and scalar operations run on front-end processors, parallel operations run on PPU
  - A 4x4 crossbar switch (Nexus) connects the 4 front-ends to 4 sections of the PPU
  - Each PPU section is controlled by a sequencer, which receives assembly language (Paris) instructions and broadcasts micro-instructions to each processor

CM-2 Processors

- Each node contains 32 processors (implemented by 2 custom processor chips), 2 floating-point accelerator chips, and memory chips
- Processor chip (contains 16 processors)
  - Contains ALU, flag registers, etc.
  - Contains NEWS interface, router interface, and I/O interface
  - 16 processors are connected in a 4x4 mesh to their N, E, W, and S neighbors
- RAM memory
  - 64Kbits, bit addressable
- FP acceleration (2 chips)
  - First chip is interface, second is FP execution unit
  - 2 chips serve 32 processors

CM-2 Processor Chip

- Instruction
  - All processor chips receive the same instruction from the sequencer
    - Individual processors may be masked using a flag bit for processor activation
  - Produces outputs based on memory / flag lookup table (256 possible functions)
  - Arithmetic is bit-serial
- 16 flag registers
  - 8 bits for general purpose use
  - 8 bits predefined
    - NEWS flag — accessible by neighbors
    - 2 flags for message router data movement and handshaking
    - Memory parity flag
    - Flag for daisy-chaining processors
    - Zero flag (hardcoded)
    - 2 diagnostic flags

CM-2 Interconnect

- Broadcast and reduction network
  - Broadcast
  - Reduction (e.g., bitwise OR, numerically largest, or sum)
  - Scan — collect cumulative results over sequence of processors (e.g., parallel prefix)
  - Spread (scatter)
  - Sort elements
- NEWS grid can be used for nearest-neighbor communication
  - Communication in multiple dimensions: 256x256, 1024x64, 8x8192, 64x32x32, 16x16x16x16, 8x8x4x8x8x4
  - Regular pattern avoids overhead of explicitly specifying destination address

CM-2 Interconnect (cont.)

- The 16-processor chips are linked by a 12-dimensional hypercube
  - Send or get values from arbitrary locations in data-parallel fashion
- Hypercube uses router for point-to-point communication between processor chips
  - Messages move across each of 12 dimensions in sequence
  - If no conflicts, a message will reach its destination within 1 cycle of the sequence
    - All processors can send a message (of any length), all messages are sent and delivered at same time
  - Actual throughput depends on message length and access patterns

CM-2 Nexus

- The nexus is a 4x4 crosspoint switch that connects up to 4 front-end computers to up to 4 sequencers
  - CM can be configured as up to 4 sections, each used separately
  - Any front-end can be connected to any section or combination of sections
  - Example: 64K processors, four 16K sections (1 to one FE, 1 to another FE, 2 to third FE, fourth PE for other tasks)
- Each section connects to one of 8 I/O channels (graphics display frame buffer, or I/O controller)
  - Transfers initiated by front-end computers
  - Data goes into buffers, when buffers are full it goes to a data vault (each has 39 disk drives, total capacity 10GB)

Software

- System software is based on OS used in front-end computers
  - Use familiar OS, languages
  - Front end handles all flow of control, including storing and executing program, and interaction with user and programmer
  - Languages: Paris, *LISP, CM-LISP, C*
- Paris (parallel instruction set)
  - Inter-processor communication, vector summation, matrix multiplication, sorting
  - Front-end processor sends Paris instructions to processor sequencers
    - Functions & subroutines (direct actions of processors, router, I/O, etc., including scan and spread operations), global variables (find out how many processor are available, etc.)
    - Sequencer produces low-level instructs.

DAP Overview

- Distributed-memory SIMD (bit-serial)
- International Computers Limited (ICL)
  - 1976 prototype, deliveries in 1980
  - ICL spun off Actime Memory Technology Ltd in 1986, became Cambridge Parallel Processing Inc in 1992
- Matrix of PEs
  - 32x32 for DAP 500, 64x64 for DAP 600
  - Connection to 4 nearest neighbors (w/ wrap-around), plus column & row buses
  - One-bit PEs with 32Kb–1Mb of memory
- DAP system = host + MCU + PE array
  - Host (Sun or VAX) interacts with user
  - Master control unit (MCU) runs main program, PE array runs parallel code

DAP MCU and HCU

- MCU (Master Control Unit)
  - 32-bit 10 MHz CPU w/ registers, instruction counter, arithmetic unit, etc.
  - Executes scalar instructions, broadcasts others to PE array
- HCU (Host Connection Unit)
  - Gateway between DAP and host
  - Motorola 68020, SCSI port, VME interface, two RS232 serial ports
  - Provides memory boundary protection, has EPROM for code storage, 1MB RAM for data and program storage
  - Data transfers are memory-memory transfers across VME bus
  - Provides medium-speed I/O plus fast data channels (e.g.,to high-resolution color display)

DAP Processing Element

- 3 1-bit registers
  - Q = accumulator, C = carry,
    A = activity control (can inhibit memory writes in certain instructions)
  - All bits of a register over all PEs is called a “register plane” (32x32 or 64x64 bits)
- Adder
  - Two inputs connect to Q and C registers
  - Third input connects to multiplexor, from PE memory, output of Q or A registers, carry output from neighboring PEs, or data broadcast from MCU
    - A register also get input from this mux
    - Mux output can also be inverted
  - PE outputs (adder and mux) can be stored in memory, under control of A reg
  - D reg for asynchronous I/O, S ref for instructs that both read & write to memory

PE Memory and MCU

- PE Memory
  - Each PE has between 32 Kb and 1 Mb
  - Vector (horizontal) mode: successive bits of a word are mapped onto successive bits of a single row of a store plane
  - Matrix (vertical) mode: successive bits… onto the same bit position in successive store planes
- MCU functionality
  - Instruction fetch, decoding, and address generation
  - Executes scalar instructions and broadcasts instruction streams to PEs
  - Transmits data between PE array memory and MCU registers
  - Transmits data between DAP and host file system or peripherals

Master Control Unit (MCU)

- Code store (memory)
  - 32 bit instructions,
    between 128 K words and 1 M words
- 32-bit general-purpose registers
  - M0 – M13: general purpose, operated on by arithmetic and logical operations, can be transferred to and from memory array
  - M1 — M7 can be used as “modifiers” for addresses and values
- Machine states
  - Non-privileged, interruptible (user mode)
  - Privileged, interruptible
  - Privileged, non-interruptible
- Datum / limit regs. for address checking

Master Control Unit (MCU) Instructions

- Addresses
  - A 32-bit word, within a row or column, within a store plane
- "DO" instruction
  - No hardware overhead for these loops
  - HW support allows instructions inside the loops to access, in successive iterations, successive bit planes, rows, columns,or words of memory
- Nearest neighbor
  - Specify direction in instruction for shifts
  - For vector adds, specify whether rows or columns are being added, which direction to send carry bit
  - Specify behavior at edge of operation

Gamma II^{Plus}

- Fourth-generation DAP, produced by Cambridge Parallel Processing in 1995
- Gamma II^{Plus} 1000 = 32x32
  Gamma II^{Plus} 4000 = 64x64
- PE memory: 128Kb–1Mb
- PE also contains an 8-bit processor
  - 32 bytes of internal memory
  - D register to transfer data to/from array memory (1-bit data path) and to/from internal memory (8-bit data path)
  - A register, similar to a 1-bit processor
  - Q register, like accumulator, 32 bits wide (any one of which can be selected as an operand), can also be shifted
  - ALU to provide addition, subtraction, and logical operations

Gamma II Plus 4000