

CS184c: Computer Architecture [Parallel and Multithreaded]

Day 5: April 17, 2001
Network Interface
Dataflow Intro



CALTECH cs184c Spring2001 -- DeHon

Admin

CALTECH cs184c Spring2001 -- DeHon

Projects

- Get idea this week
- Plan on meet/formulate details next

CALTECH cs184c Spring2001 -- DeHon

Reading

- Net Interface
 - Read AM
 - Skip/skim Henry/Joerg
- Dataflow General
 - Read DF Architectures (skim sect 4)
 - Read Two Fundamental Issues
- DF Architectures
 - Read ETS, TAM
 - Skim *T

CALTECH cs184c Spring2001 -- DeHon

Talks

- Ron Weiss
 - Cellular Computation and Communication using Engineered Genetic Regulatory Networks
 - [(Bio) Cellular Message Passing ☺]
 - Thursday 4pm
 - Beckman Institute Auditorium

CALTECH cs184c Spring2001 -- DeHon

Today

- Active Messages
- Processor/Network Interface Integration
- Dataflow Model

CALTECH cs184c Spring2001 -- DeHon

What does message handler have to do?

- Send
 - allocate buffer to compose outgoing message
 - figure out destination
 - address
 - routing?
 - Format header info
 - compose data for send
 - put out on network
 - copy to privileged domain
 - check permissions
 - copy to network device
 - checksums
- Not all messages require
- Hardware support
- Avoid (don't do it)

CALTECH cs184c Spring2001 -- DeHon

Not all messages require
Hardware support
Avoid (don't do it)

What does message handler have to do?

- Receive
 - queue result
 - copy buffer from queue to privilege memory
 - check message intact (checksum)
 - message arrived right place?
 - Reorder messages?
 - Filter out duplicate messages?
- Figure out which process/task gets message
- check privileges
- allocate space for incoming data
- copy data to buffer in task
- hand off to task
- decode message type
- dispatch on message type
- handle message

CALTECH cs184c Spring2001 -- DeHon

Active Messages

- Message contains PC of code to run
 - destination
 - message handler PC
 - data
- Receiver pickups PC and runs
- [similar to J-Machine, conv. CPU]

CALTECH cs184c Spring2001 -- DeHon

Active Message Dogma

- Integrate the data directly into the computation
- Short Runtime
 - get back to next message, allows to run directly
- Non-blocking
- No allocation
- Runs to completion
- **...Make fast case common**

CALTECH cs184c Spring2001 -- DeHon

User Level NI Access

- Avoids context switch
- Viable if hardware manage process filtering

CALTECH cs184c Spring2001 -- DeHon

Hardware Support I

- Checksums
- Routing
- ID and route mapping
- Process ID stamping/checking
- Low-level formatting

CALTECH cs184c Spring2001 -- DeHon

What does AM handler do?

- Send
 - compose message
 - destination
 - receiving PC
 - data
 - copy/queue to NI
- Receive
 - pickup PC
 - dispatch to PC
 - handler dequeues data into place in computation
 - [maybe more depending on application]
 - idempotence
 - ordering
 - synchronization

CALTECH cs184c Spring2001 -- DeHon

Example: PUT Handler

- Message:
 - remote node id
 - put handler (PC)
 - remote adder
 - data length
 - (flag adder)
 - data
- No allocation
- Idempotent
- Receiver:
 - poll
 - r1 = packet_pres
 - beq r1 0 poll
 - r2=packet(0)
 - branch r2
 - put_handler
 - r3=packet(1)
 - r4=packet(2)
 - r5=packet+r4
 - r6=packet+3
 - mdata
 - *r3=packet(r6)
 - r6++
 - blt r6,r5 mdata
 - consume packet
 - goto poll

CALTECH cs184c Spring2001 -- DeHon

Example: GET Handler

- Message Request
 - remote node
 - get handler
 - local addr
 - data length
 - (flag addr)
 - local node
 - remote addr
- Message Reply can just be a PUT message
 - put into specified local address

CALTECH cs184c Spring2001 -- DeHon

Example: GET Handler

```
get_handler
- out_packet(0)=packet(6)
- out_packet(1)=put_handler
- out_packet(2)=packet(3)
- out_packet(3)=packet(4)
- r6=4
- r7=packet(7)
- r5=packet(4)
- consume packet
- r5=r5+4

mdata
- out_packet(r6)=*r7
- r6++
- r7++
- blt r6,r5 mdata
- send out_packet
- goto poll
```

CALTECH cs184c Spring2001 -- DeHon

Example: DF Inlet synch

- Consider 3 input node (e.g. add3)
 - “inlet handler” for each incoming data
 - set presence bit on arrival
 - compute node when all present

CALTECH cs184c Spring2001 -- DeHon

Example: DF Inlet Synch

- inlet message
 - node
 - inlet_handler
 - frame base
 - data_addr
 - flag_addr
 - data_pos
 - data
- Inlet
 - move data to addr
 - set appropriate flag
 - if all flags set
 - enable DF node computation
- ? Care not enable multiple times?

CALTECH cs184c Spring2001 -- DeHon

Interrupts vs. Polling

- What happens on message reception?
- Interrupts
 - cost context switch
 - interrupt to kernel
 - save state
 - force attention to the network
 - guarantee get messages out of input queue in a timely fashion

CALTECH cs184c Spring2001 -- DeHon

Interrupts vs. Polling

- Polling
 - if getting many messages to same process
 - message handlers short / bounded time
 - may be fine to just poll between handlers
 - requires:
 - user-level/fine-grained scheduling
 - guarantee will get back to
 - avoid context switch cost

CALTECH cs184c Spring2001 -- DeHon

Interrupts vs. Polling

- Can be used together to minimize cost
 - poll network interface during batch handling of messages
 - interrupt to draw attention back to network if messages sit around too long
 - polling works for same process
 - interrupt if different process
 - common case is work on same process for a while

CALTECH cs184c Spring2001 -- DeHon

AM vs. JM

- J-Machine handlers can fault/stall
 - touch futures...
- J-Machine fast context with small state
 - not get to exploit rich context/state
- AM exploits register locality by scheduling together larger block of data
 - processing related handlers together (same context)
 - more next week (look at TAM)

CALTECH cs184c Spring2001 -- DeHon

1990 Message Handling

- nCube/2 160 μ s (360ns/byte)
- CM-5 86 μ s (120ns/byte)

CALTECH cs184c Spring2001 -- DeHon

Active Message Results

- CM5 (user-level messaging)
 - send 1.6 μ s [50 instructions]
 - receive/dispatch 1.7 μ s
- nCube/2 (OS must intervene)
 - send 11 μ s [21 instructions]
 - receive 15 μ s [34 instructions]
- Myrinet (GM)
 - 6.5 μ s end-to-end GMs
 - 1-2 μ s host processing time

CALTECH cs184c Spring2001 -- DeHon

Hardware Support II

- Roll presence tests into dispatch
- compose message data from registers
- common case
 - reply support
 - message types
- Integrate network interface as functional unit

CALTECH cs184c Spring2001 -- DeHon

Presence Dispatch

- Handler PC in common location
- Have hardware supply null handler PC when no messages current
- Poll:
 - read MsgPC into R1
 - branch R1
- Also use to handle cases and priorities
 - by modifying a few bits of dispatch address
 - e.g. queues full/empty

CALTECH cs184c Spring2001 -- DeHon

Compose from Registers

- Put together message in registers
 - reuse data from message to message
 - compute results directly into target
 - user register renaming and scoreboarding to continue immediately while data being queued

CALTECH cs184c Spring2001 -- DeHon

Common Case Msg/Replies

- Instructions to
 - fill in common data on replies
 - node address, handler?
 - Indicate message type
 - not have to copy

CALTECH cs184c Spring2001 -- DeHon

Example: GET handler

- Get_handler
 - R1=i0 // address from message register
 - R2=*R1
 - o2=R2 // value into output data register
 - SEND -reply type=reply_mesg_id
 - NEXT

CALTECH cs184c Spring2001 -- DeHon

AM as primitive Model

- Value of Active Messages
 - articulates a model of what primitive messaging needs to be
 - identify key components
 - then can optimize against
 - how much hardware to support?
 - What should be in hardware/software?
 - What are common cases?
 - Should get special treatment?

CALTECH cs184c Spring2001 -- DeHon

Dataflow

CALTECH cs184c Spring2001 -- DeHon

Dataflow

- Model of computation
- Contrast with Control flow

CALTECH cs184c Spring2001 -- DeHon

Dataflow / Control Flow

- Program is a graph of operators
- Operator consumes tokens and produces tokens
- All operators run concurrently
- Program is a sequence of operations
- Operator reads inputs and writes outputs into common store
- One operator runs at a time
 - Defines successor

CALTECH cs184c Spring2001 -- DeHon

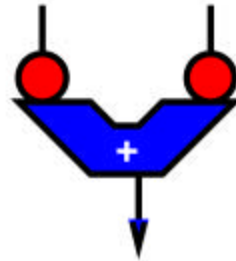
Token

- Data value with presence indication

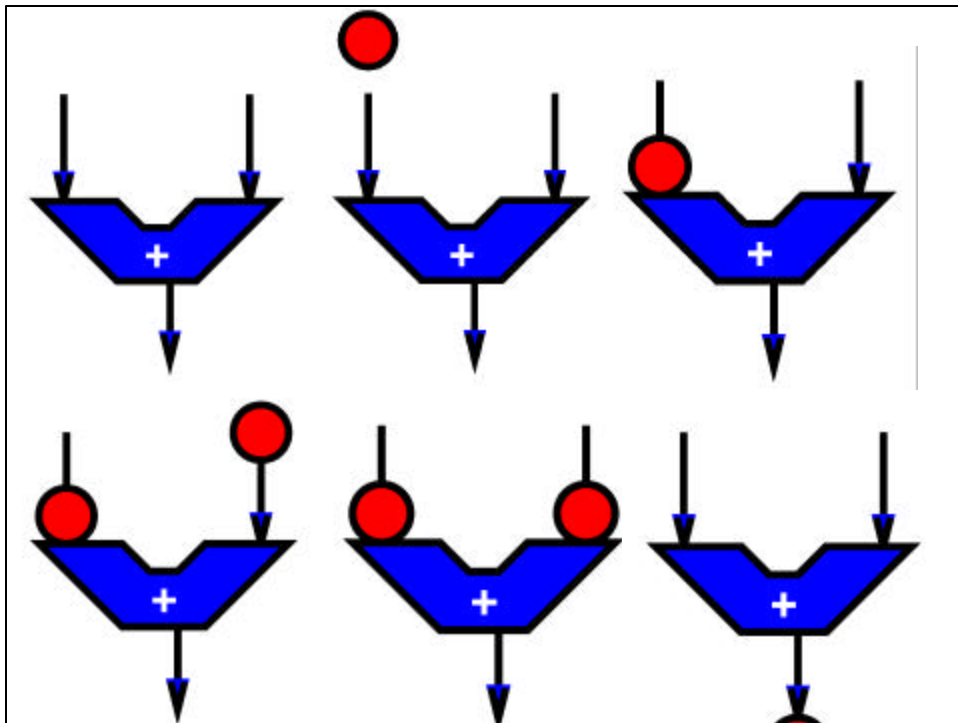
CALTECH cs184c Spring2001 -- DeHon

Operator

- Takes in one or more inputs
- Computes on the inputs
- Produces a result
- Logically self-timed
 - “Fires” only when input set present
 - Signals availability of output

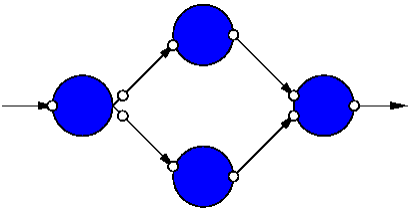


CALTECH cs184c Spring2001 -- DeHon



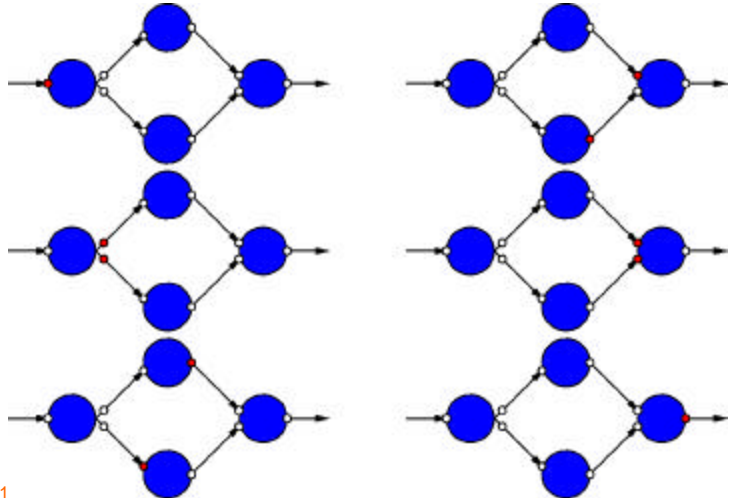
Dataflow Graph

- Represents
 - computation sub-blocks
 - linkage
- Abstractly
 - controlled by data presence



CALTECH cs184c Spring2001 -- DeHon

Dataflow Graph Example



CALTECH cs184c Spring2001 -- DeHon

Straight-line Code

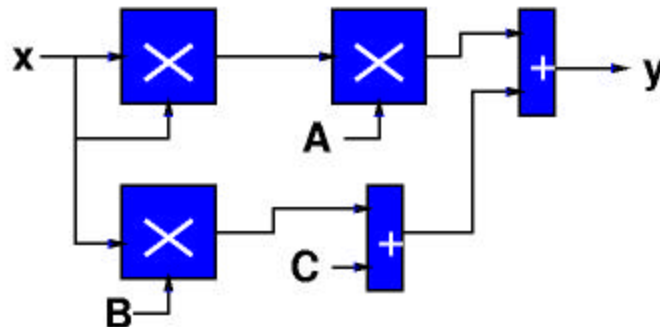
- Easily constructed into DAG
 - Same DAG saw before
 - No need to linearize

CALTECH cs184c Spring2001 -- DeHon

CS184b

Dataflow Graph

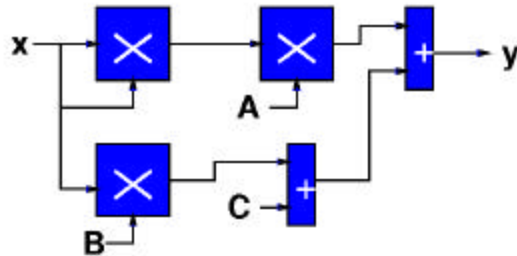
- Real problem is a graph



CALTECH cs184c Spring2001 -- DeHon

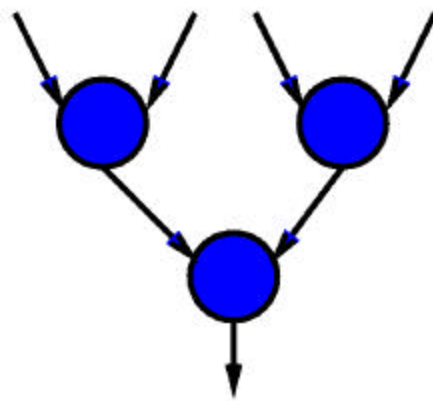
Task Has Parallelism

MPY R3,R2,R2 MPY R4,R2,R5
 MPY R3,R6,R3 ADD R4,R4,R7
 ADD R4,R3,R4



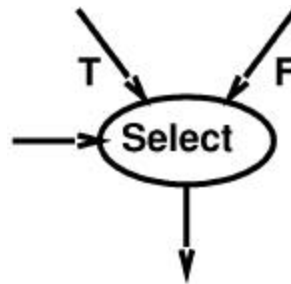
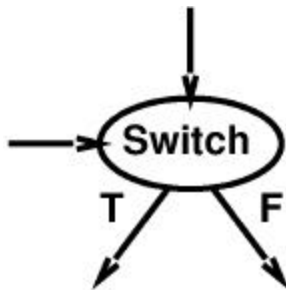
DF Exposes Freedom

- Exploit dynamic ordering of data arrival
- Saw aggressive control flow implementations had to exploit
 - Scoreboarding
 - OO issue



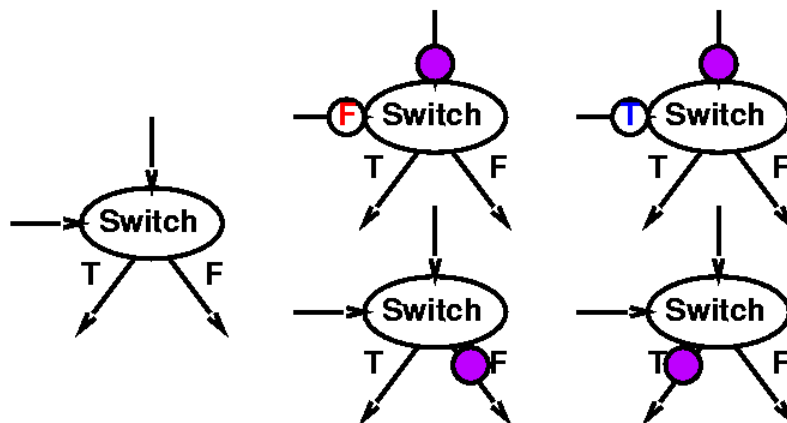
Data Dependence

- Add Two Operators
 - Switch
 - Select



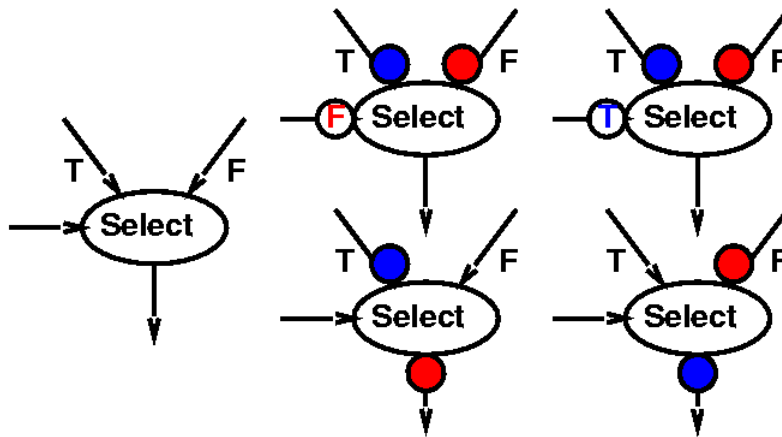
CALTECH cs184c Spring2001 -- DeHon

Switch



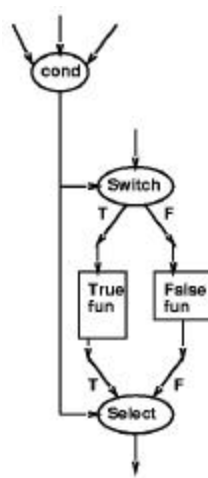
CALTECH cs184c Spring2001 -- DeHon

Select



CALTECH cs184c Spring2001 -- DeHon

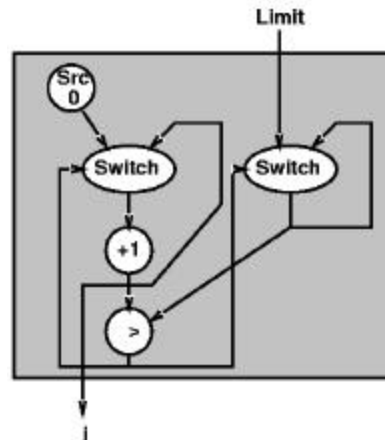
Constructing If-Then-Else



CALTECH cs184c Spring2001 -- DeHon

Looping

- For ($i=1; i < \text{Limit}; i++$)



CALTECH cs184c Spring2001 -- DeHon

Procedure Calls

- Semantics: instantiate new subgraph
- Work fibonacci example

CALTECH cs184c Spring2001 -- DeHon

Functional Languages

- Natural connection to functional languages
- ...but can be target of well-behaved sequential languages...

CALTECH cs184c Spring2001 -- DeHon

Class stopped here

4/17/01

CALTECH cs184c Spring2001 -- DeHon

Key Element of Control

- Synchronization on Data Presence
- Construct:
 - Futures
 - Full-empty bits
 - I-structures

CALTECH cs184c Spring2001 -- DeHon

Future

- Future is a promise
- An indication that a value will be computed
 - And a handle for getting a handle on it
- Sometimes used as program construct

CALTECH cs184c Spring2001 -- DeHon

Future

- Future computation immediately returns a future
- Future is a handle/pointer to result
- (define (dot a b)
 - (cons (future (* (first a) (first b)))
 (dot (rest a) (rest b))))
- (define (sum-reduce a)
 - (+ (first a) (sum-reduce (rest a))))

CALTECH cs184c Spring2001 -- DeHon

Strict/non-strict

- Strict operation requires the **value** of some variable
 - E.g. add, multiply
- Non-strict operation only needs the handle for a value
 - E.g. cons, procedure call
 - (anything that just passes the handle off)

CALTECH cs184c Spring2001 -- DeHon

Futures

- Safe with functional routines
 - Create dataflow
- Can introduce non-determinacy with side-effecting routines
 - Not clear when operation completes

CALTECH cs184c Spring2001 -- DeHon

Future/Side-Effect hazard

- (define (decrement! a b)
 - (set! a (- a b)))
- (print (* (future (decrement! c d))
(future (decrement! d 2))))

CALTECH cs184c Spring2001 -- DeHon

Full/Empty bit

- Tag on data indicates data presence
 - E.g. tag in memory, RF
 - Like Scoreboard present bit
- When computation allocated, set to empty
 - E.g. operation issued into pipe, future call made
- When computation completes
 - Data written into slot (register, memory)
 - Bit set to full
- On data access
 - Bit full, strict operation can get value
 - Bit empty, strict operation block on completion

CALTECH cs184c Spring2001 -- DeHon

I-Structure

- Array/object will full-empty bits on each field
- Allocated empty
- Fill in value as compute
- Strict access on empty
 - Queue requester in structure
 - Send value to requester when written and becomes full

CALTECH cs184c Spring2001 -- DeHon

I-Structure

- Allows efficient “functional” updates to aggregate structures
- Can pass around pointers to objects
- Preserve ordering/determinacy
- E.g. arrays

CALTECH cs184c Spring2001 -- DeHon

Big Ideas

- Primitives/Mechanisms
 - End-to-end/common case
 - don't burden everything with features needed by only some tasks
- Abstract Model
- More than one compute model
- Expose freedom exists in computation
- Separate essential/useful work from overhead

CALTECH cs184c Spring2001 -- DeHon

[MSB-1] Ideas

- Minimize Overhead
 - moving data around is not value added operation
 - minimize copying
- Overlap Compute and Communication
 - queue
 - don't force send/receive rendezvous
- Get the OS out of the way of **common operations**

CALTECH cs184c Spring2001 -- DeHon