ESE535:
Electronic Design Automation

Day 15: March 23, 2015
Dataflow
Previously

- Scheduling of concurrent operations
Want to See

• Abstract compute model
  – natural for parallelism and hardware

• Describe computation abstracted from implementation
  – Defines correctness
Today

• Dataflow
• SDF
  – Single rate
  – Multirate
• Dynamic Dataflow
• Expression

Behavioral
(C, MATLAB, …)
  Arch. Select
  Schedule
  RTL
  FSM assign
  Two-level,
  Multilevel opt.
  Covering
  Retiming
  Gate Netlist
  Placement
  Routing
  Layout
  Masks
Parallelism Motivation
Producer-Consumer Parallelism

- Can run concurrently
- Just let consumer know when producer sending data
Pipeline Parallelism

- Can potentially all run in parallel
- Like **physical** pipeline
- Useful to think about **stream** of data between operators
DAG Parallelism

- Doesn’t need to be linear pipeline
- Synchronize inputs
Graphs with Feedback

- In general may hold state
- Very natural for many tasks
Definitions
Operation/Operator

- **Operation** – logic computation to be performed
- **Operator** – physical block that performs an Operation
Dataflow / Control Flow

Dataflow
• Program is a graph of operations
• Operation consumes tokens and produces tokens
• All operations run concurrently

Control flow (e.g. C)
• Program is a sequence of operations
• Operation reads inputs and writes outputs into common store
• One operation runs at a time
  – defines successor
Token

• Data value with presence indication
  – May be conceptual
    • Only exist in high-level model
    • Not kept around at runtime
  – Or may be physically represented
    • One bit represents presence/absence of data
Token Examples?

• What are familiar cases where data may come with presence tokens?
  – Network packets
  – Memory references from processor
    • Variable latency depending on cache presence
  – Start bit on serial communication
Operation

- Takes in one or more inputs
- Computes on the inputs
- Produces results

- Logically **self-timed**
  - “Fires” only when input set present
  - Signals availability of output
Dataflow Graph

• Represents
  – computation sub-blocks
  – linkage

• Abstractly
  – controlled by data presence
Dataflow Graph Example
In-Class Dataflow Example
Stream

• Logical abstraction of a persistent point-to-point communication link
  – Has a (single) source and sink
  – Carries data presence / flow control
  – Provides in-order (FIFO) delivery of data from source to sink
Streams

• Captures communications structure
  – Explicit producer→consumer link up

• Abstract communications
  – Physical resources or implementation
  – Delay from source to sink
Register Transfer Level (RTL)

- Describe computation as logic and registers
- Equations (logic) define values to be clocked into next register
- Typically what you right in VHDL, Verilog
Dataflow Abstracts Timing

• Doesn’t say
  – on which cycle calculation occurs [contrast RTL]
• Does say
  – What order operations occur in
  – How data interacts
    • i.e. which inputs get mixed together
• Permits
  – Scheduling on different # of resources
  – Operators with variable delay [examples?]
  – Variable delay in interconnect [examples?]
Examples

• Operators with Variable Delay
  – Cached memory or computation
  – Shift-and-add multiply
  – Iterative divide or square-root

• Variable delay interconnect
  – Shared bus
  – Distance changes
    • Wireless, longer/shorter cables
  – Computation placed on different cores?
Difference: Dataflow Graph/Pipeline
Clock Independent Semantics

Interconnect
Takes n-clocks
Semantics

• Need to implement semantics
  – *i.e.* get same result as if computed as indicated

• But can implement any way we want
  – That preserves the semantics
  – Exploit freedom of implementation
Dataflow Variants
Synchronous Dataflow (SDF)

• Particular, restricted form of dataflow
• Each operation
  – Consumes a fixed number of input tokens
  – Produces a fixed number of output tokens
  – When full set of inputs are available
    • Can produce output
  – Can fire any (all) operations with inputs available at any point in time
Synchronous Dataflow

\[ \begin{align*}
X_1 & \rightarrow A \\
X_2 & \rightarrow B \\
Y_1 & \rightarrow C \\
Y_2 & \rightarrow D \\
\sqrt{X} & \rightarrow E \\
\end{align*} \]
SDF: Execution Semantics

while (true)
    Pick up any operator
    If operation has full set of inputs
        Compute operation
        Produce outputs
    Send outputs to consumers
Multirate Synchronous Dataflow

- Rates can be different
  - Allow lower frequency operations
  - Communicates rates to CAD
    - Something not clear in RTL
    - Use in scheduling, provisioning
  - Rates must be constant
    - Data independent
SDF

• Can validate flows to check legal
  – Like KCL $\rightarrow$ token flow must be conserved
  – No node should
    • be starved of tokens
    • Collect tokens

• Schedule operations onto processing elements
  – Provisioning of operators

• Provide real-time guarantees

• Simulink is SDF model
SDF: good/bad graphs
SDF: good/bad graphs
Dynamic Rates?

• When might static rates be limiting? (prevent useful optimizations?)
  – Compress/decompress
    • Lossless
    • Even Run-Length-Encoding
  – Filtering
    • Discard all packets from spamRus
  – Anything data dependent
Data Dependence

• Add Two Operators
  – Switch
  – Select
Switch
Filtering Example

\[
\text{dup} \rightarrow \text{spamRus?} \rightarrow \text{switch} \rightarrow \text{discard}
\]
Select
Constructing If-Then-Else
In-Order Merge

- **Task**: Merge to ordered streams in order onto a single output stream
  - Key step in merge sort

- Use to illustrate switch/select
Idiom to Selectively Consume Input

- Hold onto current head on loop
  - Shown left here
  - With T-side control

![Diagram showing a select-switch diagram]
In-Order Merge

- Use one for each of the two input streams
In-Order Merge

• Perform Comparison
In-Order Merge

• Act on result of comparison
Looping

• for (i=0;i<Limits;i++)
Universal

• Once we add switch and select, the dataflow model is as powerful as any other
  – E.g. can do anything we could do in C
  – “Turing Complete” in formal CS terms
Dynamic Challenges

• In general, cannot say
  – If a graph is well formed
    • Will not deadlock
  – How many tokens may have to buffer in stream
  – Right proportion of operators for computation
Expression
(Time Permitting)

How would we capture this in a Programming Language?
Expression

- Could express operations in C/Java
  - Each is own thread
- Link together with Streams
- *E.g.* SystemC
C Example

while (! (eos(stream_a) && ! (eos(stream_b)))
    A = stream_a.read();
    B = stream_b.read();
    Out = (a+b)*(a-b);
    stream_out.write(Out);
Connecting up Dataflow

stream stream1=new stream();
operation prod=new stock(stream1);
operation cons=new encrypt(stream1);
What have we gained?

• Ability to capture more freedom that exists
  – Freedom we can use to reduce costs
• A1: Model for expressing freedom that exists in the computation
  – Higher-level than an implementation
  – Perhaps as a useful intermediate
• A2: Model allows freedom for implementations (or instances) to take variable time?
Summary

• Dataflow Models
  – Simple pipelines
  – DAGs
  – SDF (single, multi)-rate
  – Dynamic Dataflow

• Allow
  – express parallelism
  – freedom of implementation
Big Ideas:

• Dataflow
  – Natural model for capturing computations
  – Communicates useful information for optimization
    • Linkage, operator usage rates

• Abstract representations
  – Leave freedom to implementation
Admin

- HW7
- Reading for Wednesday on Canvas
- HW8 Thursday