OUTLINE OF PART 2

• Static data-flow structures
• Performance analysis
• Implementation techniques

DATA FLOW

• Classical data flow graph:
  – Vertices (nodes) for computations
  – Edges for dependencies (edges are implicit FIFO buffers)
• Here:
  – Computation model, closer to actual asynchronous implementation (e.g. no implicit FIFOs)

PIPELINES AND RINGS

• Assume a 4-phase protocol (dual rail or bundled data).
• Values are either “valid” or “empty”.
• When a value has been copied to the next latch, two subsequent latches hold the same value.
• The newest value is the token.
• The oldest one is the bubble.
• Bubbles are essential to keep data flowing.
• The shortest ring consists of three latches holding one empty token, one valid token and one bubble.
• See Figure 3.2 of [Sparsoe 2001]

STATIC DATA FLOW BUILDING BLOCKS

• Latches, providing storage
• Function blocks, the asynchronous equivalents of combinational circuits; transparent to handshaking
• Unconditional data flow:
  – Fork, join and merge
• Conditional data flow:
  – Multiplexer and demultiplexer
• See Figure 3.3 of [Sparsoe 2001]
STEP-BY-STEP EXAMPLE WITH CYCLIC DATA FLOW

- See Figures 3.4 and 3.5 of [Sparsoe 2001]
- Note the occurrence of *forks* and *joins*
- Note that cycle has three latches

DATA-FLOW MODELS

- State machine, see Figure 3.6 of [Sparsoe 2001]
- IF statement, see Figure 3.8 of [Sparsoe 2001]
  - Version with demux/mux
  - Version with demux/merge
- WHILE statement, see Figure 3.10 of [Sparsoe 2001]
- GCD example, see Figure 3.11/3.12 of [Sparsoe 2001]

PERFORMANCE ANALYSIS

- Easy for synchronous circuits:
  - Longest path in combinational logic determines clock period
  - Number of clock cycles needed times clock period gives duration of computation
- What about asynchronous circuits?
  - Performance is not easy to compute
  - Depends on initialization and behavior of environment

BUBBLE DENSITY AND FIFO PERFORMANCE

- 2 latches per valid token:
  - Valid, empty, valid, empty, …
  - FIFO shift time is proportional to FIFO length because bubble needs to travel all along FIFO
- 3 latches per valid token:
  - Valid, empty, bubble, valid, empty, bubble, …
  - FIFO shift time is constant
- See Figure 4.1 of [Sparsoe 2001]
PARALLEL-LOAD SHIFT REGISTER EXAMPLE

• See Figures 4.2/4.3 of [Sparsoe 2001]
• Throughput is improved by adding latches to control path

IMPLEMENTATIONS OF BASIC DATA-FLOW BLOCKS: FORK, JOIN, MERGE

• Versions for 4-phase bundled data and 4-phase dual rail
• **Fork** needs to synchronize the acknowledge
• **Join** needs to synchronize the request (automatic in dual rail)
• **Merge** assumes mutual exclusion of the two input channels: the first request is passed and then the merge block waits for corresponding acknowledge
• See Figure 5.1 of [Sparsoe 2001]

ASYMMETRIC MULLER C-ELEMENT

• It may occur that there is a known (temporal) relation between the inputs of a Muller C-element.
• In such a case, the transistor network may be simplified.
• See Figure 5.2 of [Sparsoe 2001]

INTERMEZZO: RIPPLE-CARRY ADDITION

• A full adder is a 3-input, 2-output circuit:
  – Inputs: a, b and carry-in
  – Outputs: sum and carry-out
• Can alternatively be formulated by first computing intermediate signals:
  – Carry propagate: p
  – Carry generate: g
  – Or, alternatively, carry kill: k
• See pages 64/65 of [Sparsoe 2001]
FUNCTION BLOCKS: STRONG AND WEAK INDICATION

- A function block is strongly indicating if:
  - It starts producing valid outputs after all inputs have become valid.
  - It starts producing empty outputs after all inputs have become empty.
- A function block is weakly indicating if:
  - Valid or empty outputs are produced as soon as possible, before all inputs have become valid or empty.

MATCHED DELAY AND SPECULATIVE COMPLETION

- Matched delays are used in bundled data protocols to delay passing of a request signal.
- How implemented? Dummy layout, standard cells.
- Delay elements can consume some 10% of power.
- Speculative completion:
  - Predict data-dependent computation delay and select appropriate delay element.
  - See Figure 5.9 of [Sparsoe 2001].

DELAY-INSENSITIVE MINTERM SYNTHESIS

- Dual-rail implementation technique
- Rather than building a combinational function from elementary dual-rail elements, the function is directly implemented from its truth table.
- Remember: a minterm is a product of all inputs (true or negated).
- See Figure 5.10 of [Sparsoe 2001] for a full-adder example.