ECE 587 – Hardware/Software Co-Design
Lecture 24 Hardware Optimization II

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

April 15, 2015
Homework 3

- Due: 5:00pm 4/29 Chicago time
Reading Assignment

- This lecture: 6.6, 6.7, 6.8
- Next lecture: 6.9, 6.10, 6.11
Outline

Connection Sharing

Register Merging

Chaining and Multi-Cycling
Connection Sharing

- Interconnects do consume considerable amount of resource in modern chip designs.
  - Consist of metal wires, vias, and buffers.
- Merge connections into buses to reduce resource usage
  - Group connections not used at the same time
  - Use tri-state buffer to connect connection sources to bus
  - We may implement the selectors the same way as that’s exact how big mux’s are implemented.
Connection Sharing

- Interconnects do consume considerable amount of resource in modern chip designs.
  - Consist of metal wires, vias, and buffers.
- Merge connections into buses to reduce resource usage
  - Group connections not used at the same time
  - Use tri-state buffer to connect connection sources to bus
  - We may implement the selectors the same way as that's exact how big mux’s are implemented.
Connection Usage Table and Compatibility Graph

**FIGURE 6.16** SRA Datapath with labeled connections

**TABLE 6.6** Connection usage table

**FIGURE 6.17** (Gajski et al.)
Bus Assignment

(a) Compatibility graph for input buses  
(b) Compatibility graph for output buses  
(c) Bus assignment

Bus1 = [A, C, D, E, H]
Bus2 = [B, F, G]
Bus3 = [I, K, M]
Bus4 = [J, L, N]

FIGURE 6.17 Connection merging for SRA

(Gajski et al.)
Updated Datapath with Buses

FIGURE 6.16  SRA Datapath with labeled connections

FIGURE 6.18  SRA Datapath after connection merging

(GaJski et al.)

ECE 587 – Hardware/Software Co-Design  Spring 2015
Outline

Connection Sharing

Register Merging

Chaining and Multi-Cycling
Register Merging

- Organize registers with non-overlapping access times into register files
- Register input and output ports are shared to reduce the number of connections.
- Register-to-register delay may increase due to the overhead of decoding the addresses.
Register Merging

- Organize registers with non-overlapping access times into register files.
- Register input and output ports are shared to reduce the number of connections.
- Register-to-register delay may increase due to the overhead of decoding the addresses.
Register Merging

- Organize registers with non-overlapping access times into register files
- Register input and output ports are shared to reduce the number of connections.
- Register-to-register delay may increase due to the overhead of decoding the addresses.
Register Access Table and Compatibility Graph

(a) Register assignment  
(b) Register access table  
(c) Compatibility graph

\[ R1 = [a, t1, x, t7] \]
\[ R2 = [b, t2, y, t3, t5, t6] \]
\[ R3 = [t4] \]

\( \text{FIGURE 6.19} \) Register merging (Gajski et al., 2012)
FIGURE 6.20  Datapath schematic after register merging

(Gajski et al.)
Clock Period

- How to choose a clock period?
  - Smaller clock period
    - Most operations may take multiple cycles to complete.
    - The overall computation takes more cycle to finish.
    - Overhead associated with more states: execution time and power consumption
  - Larger clock period
    - Many operations may only need part of the clock cycle to complete, wasting the remaining slacks
    - The overall execution time could be longer.
- Issues addressed by chaining and multi-cycling
Clock Period

How to choose a clock period?

Smaller clock period

- Most operations may take multiple cycles to complete.
- The overall computation takes more cycle to finish.
- Overhead associated with more states: execution time and power consumption

Larger clock period

- Many operations may only need part of the clock cycle to complete, wasting the remaining slacks
- The overall execution time could be longer.

Issues addressed by chaining and multi-cycling
Clock Period

- How to choose a clock period?
- Smaller clock period
  - Most operations may take multiple cycles to complete.
  - The overall computation takes more cycle to finish.
  - Overhead associated with more states: execution time and power consumption
- Larger clock period
  - Many operations may only need part of the clock cycle to complete, wasting the remaining slacks
  - The overall execution time could be longer.

- Issues addressed by chaining and multi-cycling
Clock Period

- How to choose a clock period?
  - Smaller clock period
    - Most operations may take multiple cycles to complete.
    - The overall computation takes more cycle to finish.
    - Overhead associated with more states: execution time and power consumption
  - Larger clock period
    - Many operations may only need part of the clock cycle to complete, wasting the remaining slacks
    - The overall execution time could be longer.
- Issues addressed by chaining and multi-cycling
Chaining and Multi-Cycling

► Chaining
  ▶ Allow the result of an operation to be used immediately within the cycle by another operation
  ▶ Compose more complex combinational operations from simpler ones
  ▶ Fully utilize the whole clock period

► Multi-cycling
  ▶ Support operations requiring multiple cycles to finish
  ▶ Otherwise we have to set the clock period to the longest completion time among all operations and thus may waste a lot of slacks

► Recall our simplified HLS flow supports multi-cycling but not chaining.

► These two optimizations may change the schedule and thus may need expensive verifications.
Chaining and Multi-Cycling

- **Chaining**
  - Allow the result of an operation to be used immediately within the cycle by another operation
  - Compose more complex combinational operations from simpler ones
  - Fully utilize the whole clock period

- **Multi-cycling**
  - Support operations requiring multiple cycles to finish
  - Otherwise we have to set the clock period to the longest completion time among all operations and thus may waste a lot of slacks

- Recall our simplified HLS flow supports multi-cycling but not chaining.

- These two optimizations may change the schedule and thus may need expensive verifications.
Chaining and Multi-Cycling

- **Chaining**
  - Allow the result of an operation to be used immediately within the cycle by another operation
  - Compose more complex combinational operations from simpler ones
  - Fully utilize the whole clock period

- **Multi-cycling**
  - Support operations requiring multiple cycles to finish
  - Otherwise we have to set the clock period to the longest completion time among all operations and thus may waste a lot of slacks

- Recall our simplified HLS flow supports multi-cycling but not chaining.

- These two optimizations may change the schedule and thus may need expensive verifications.
Chaining and Multi-Cycling

- **Chaining**
  - Allow the result of an operation to be used immediately within the cycle by another operation
  - Compose more complex combinational operations from simpler ones
  - Fully utilize the whole clock period

- **Multi-cycling**
  - Support operations requiring multiple cycles to finish
  - Otherwise we have to set the clock period to the longest completion time among all operations and thus may waste a lot of slacks

- Recall our simplified HLS flow supports multi-cycling but not chaining.

- These two optimizations may change the schedule and thus may need expensive verifications.
FSMD Model w/o and w/ Chaining

(a) FSMD model for functional unit chaining

(Gajski et al.)
Updated Datapath with Chaining

FIGURE 6.22  Datapath with chained functional units

(Gajski et al.)
FSMD Model w/o and w/ Multi-Cycling

(a) FSMD model for functional unit chaining
(b) FSMD model for functional unit multi-cycling

FIGURE 6.21 Modified FSMD models for SRA algorithm

(Gajski et al.)
Updated Datapath with Multi-Cycling

FIGURE 6.23  SRA datapath with chained and multi-cycle functional units  
(Gajski et al.)
Connections can be merged into buses

Registers can be merged into register files to save ports and thus connections.

Chaining and multi-cycling help to make full use of the clock period.