ECE 587 – Hardware/Software Co-Design
Lecture 25 Hardware Synthesis III

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

April 13, 2026

Reading Assignment

- This lecture: 6
- We will study state-of-the-art hardware accelerators and interconnection networks for the rest of the semester.

Outline

Register Merging

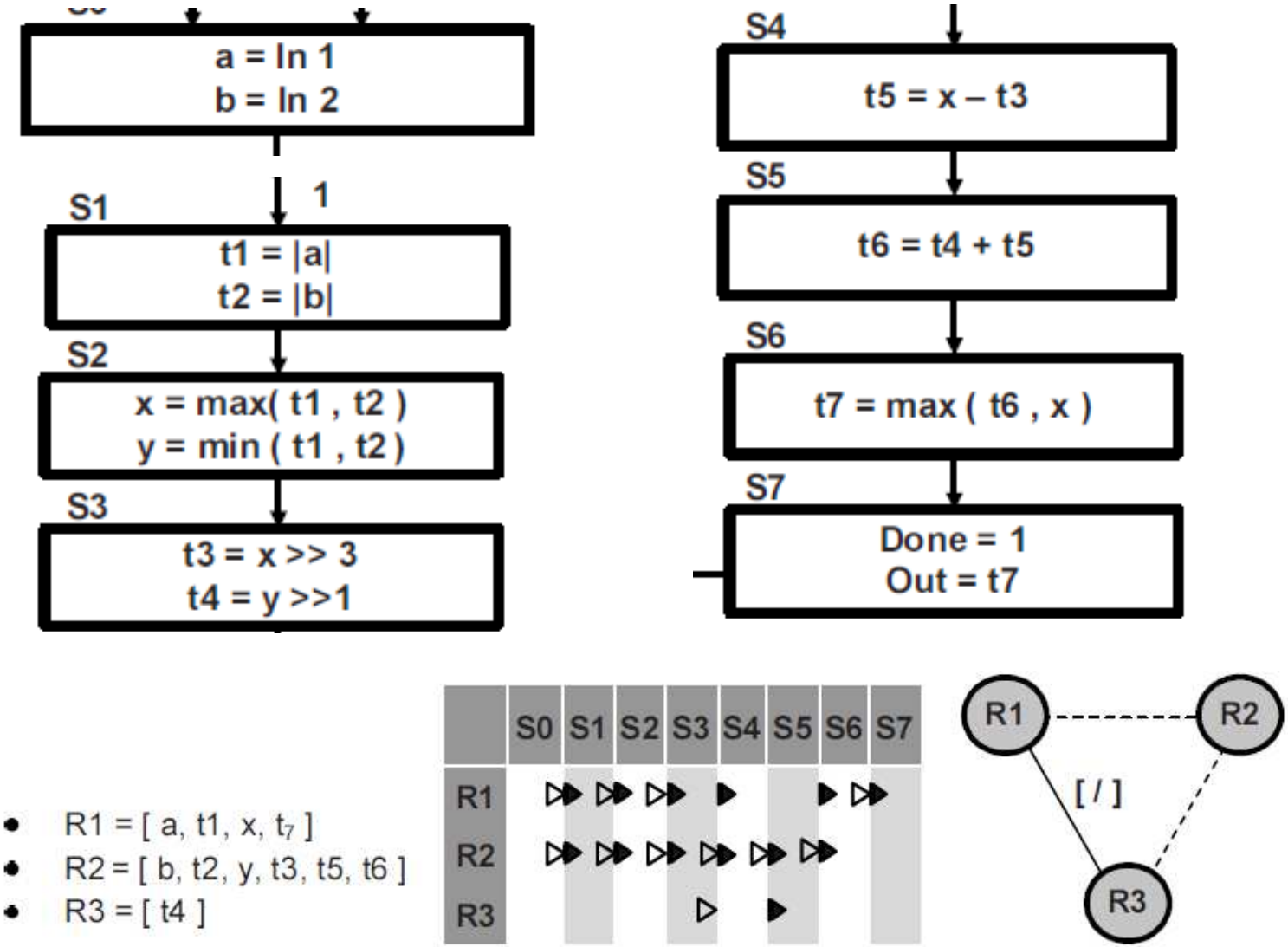
Chaining and Multi-Cycling

Pipelining

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

FIGURE 6.19 Register merging

(Gajski et al.)

Updated Datapath with Register File

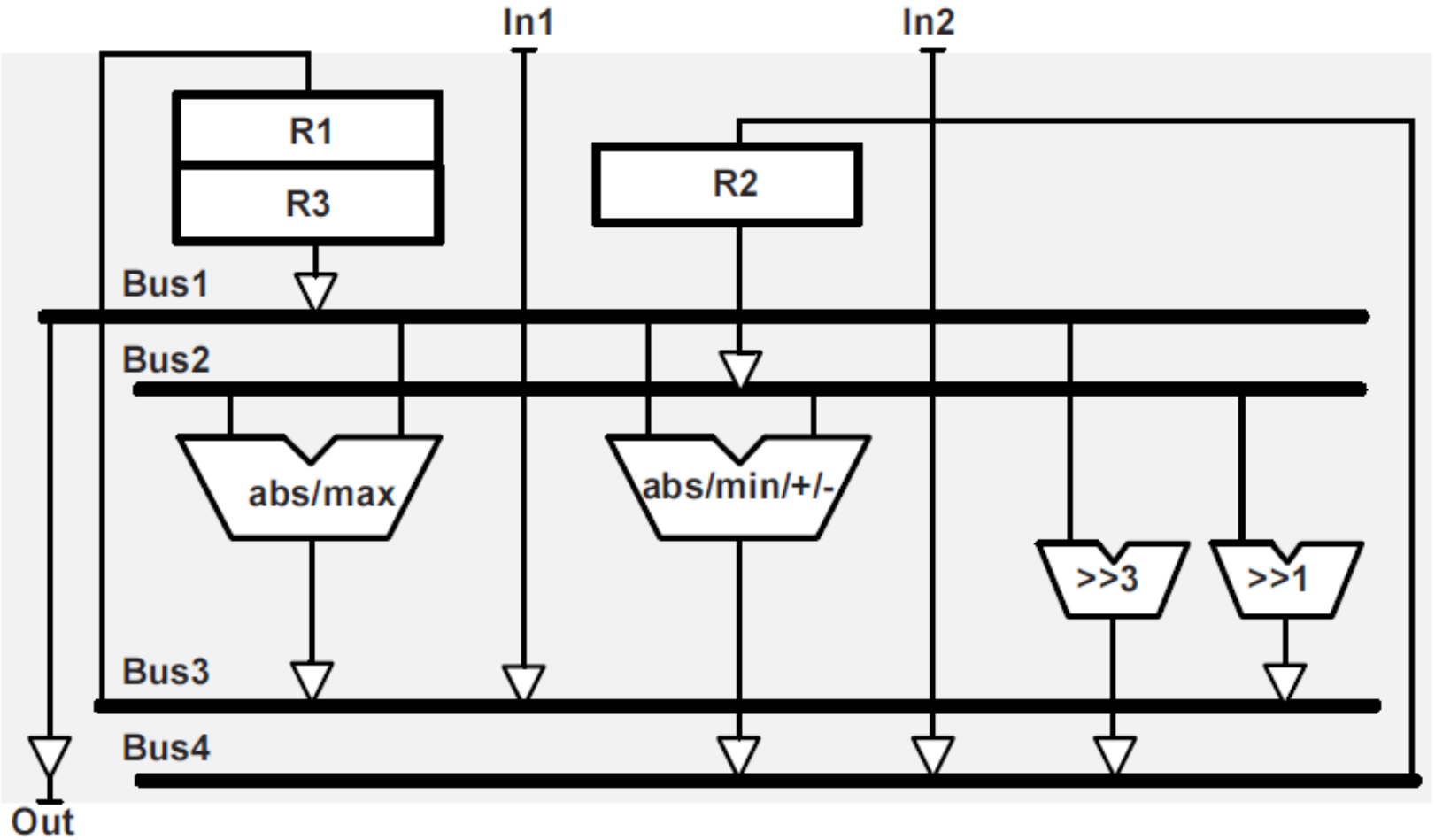


FIGURE 6.20 Datapath schematic after register merging

(Gajski et al.)

Outline

Register Merging

Chaining and Multi-Cycling

Pipelining

Clock Period

- How to choose a clock period?
- Smaller clock period
  - Most operations may take multiple cycles to complete.
  - The overall computation takes more cycles 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.

FSMD Model w/o and w/ Chaining

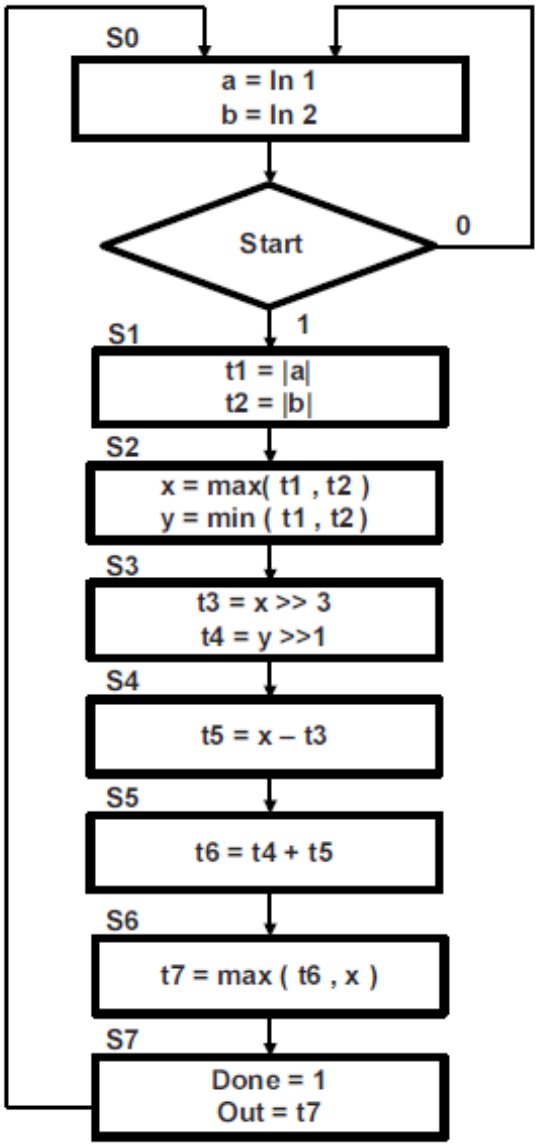
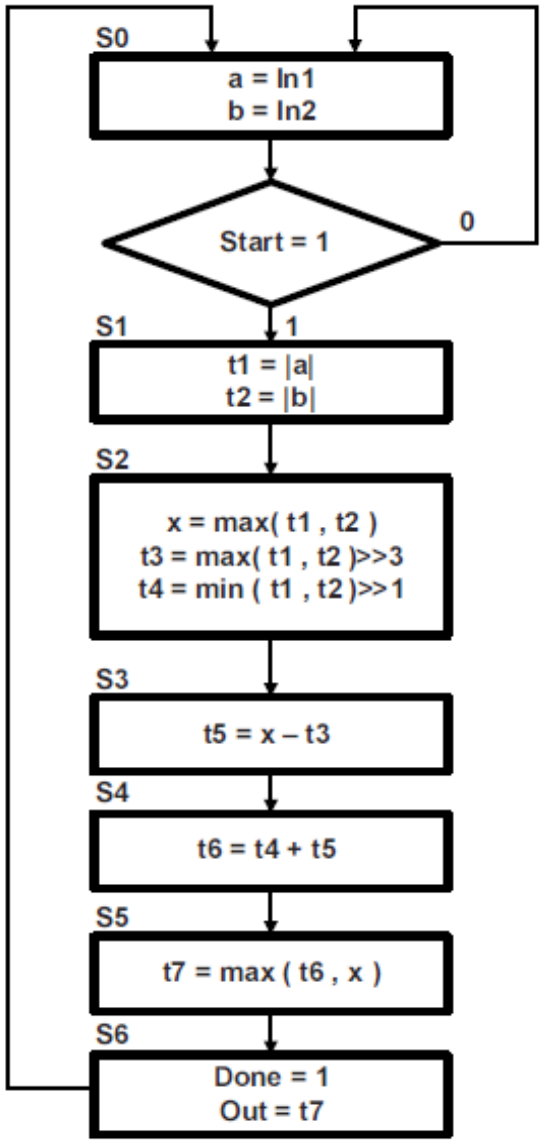


FIGURE 6.8



(a) FSMD model for functional unit chaining

FIGURE 6.21

(Gajski et al.)

Updated Datapath with Chaining

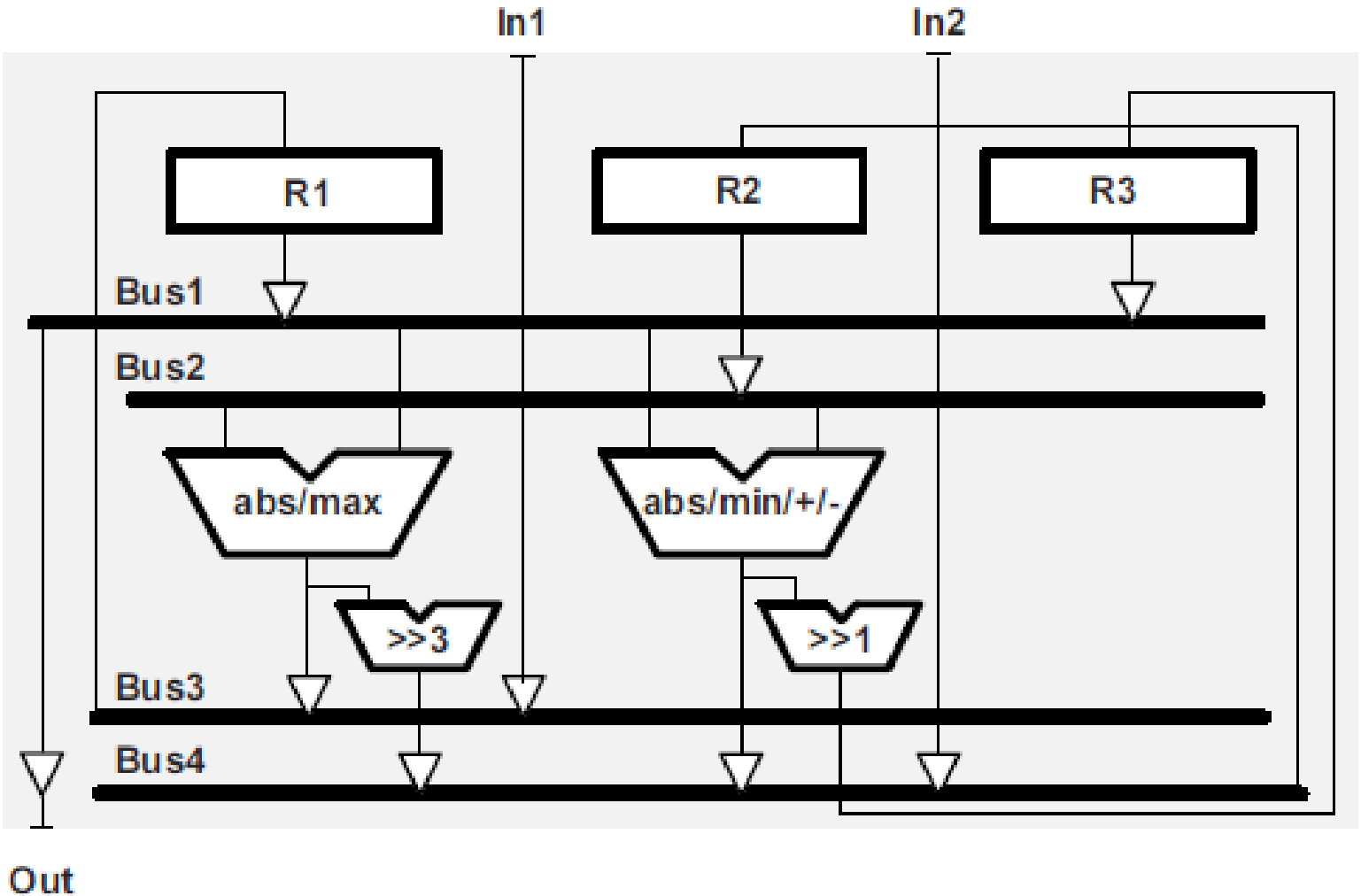
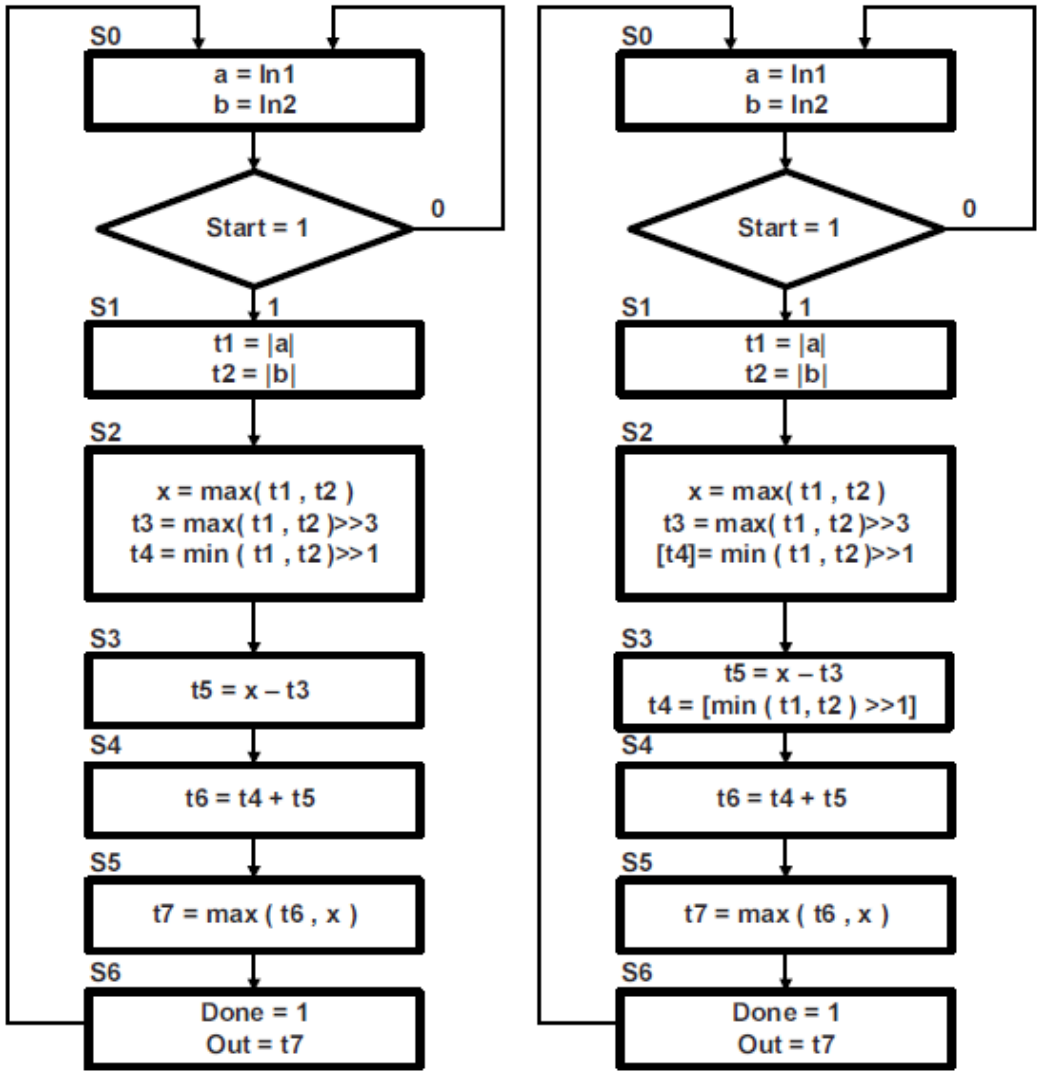


FIGURE 6.22 Datapath with chained functional units

(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

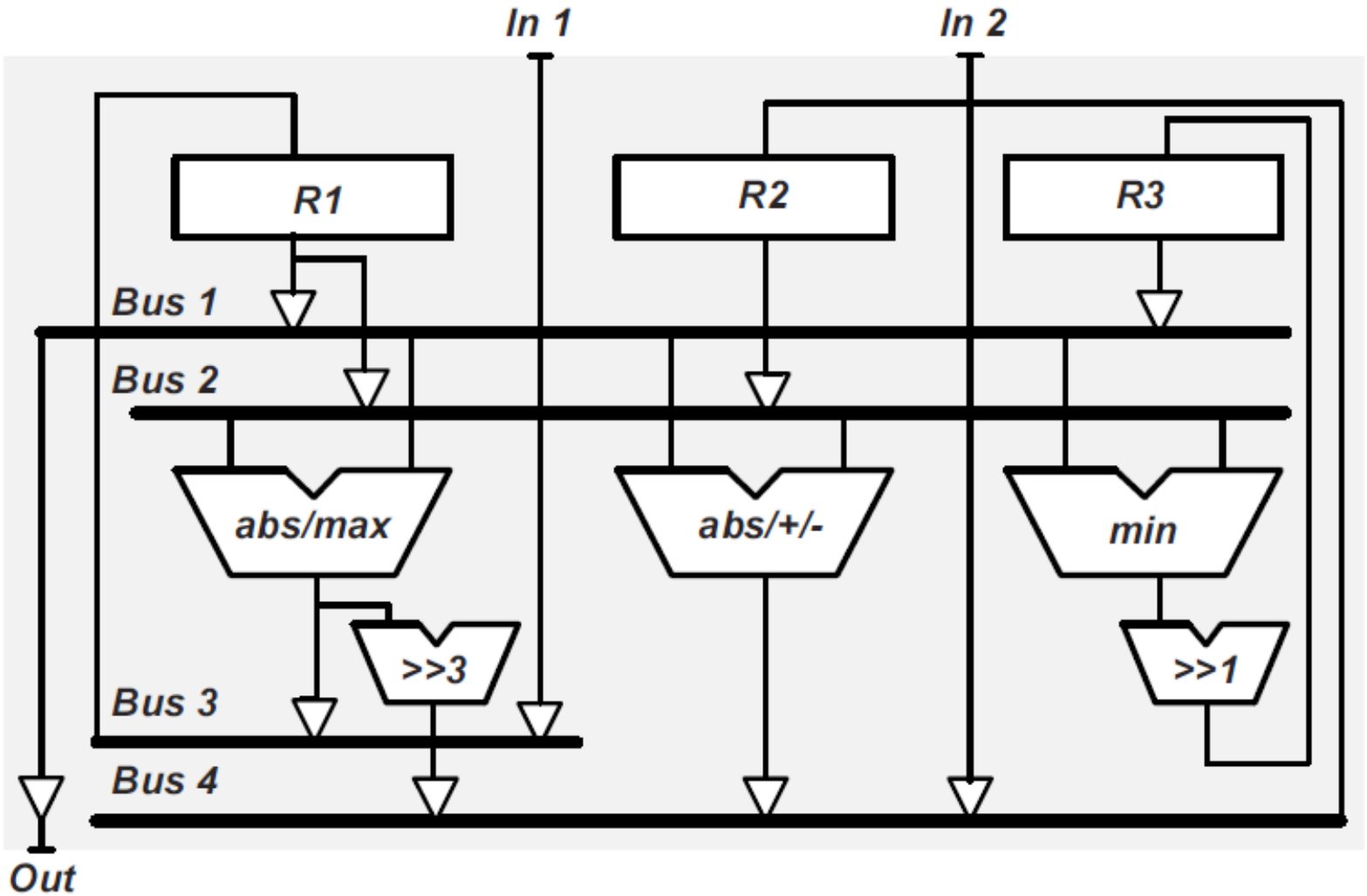


FIGURE 6.23 SRA datapath with chained and multi-cycle functional units (Gajski et al.)

Outline

Register Merging

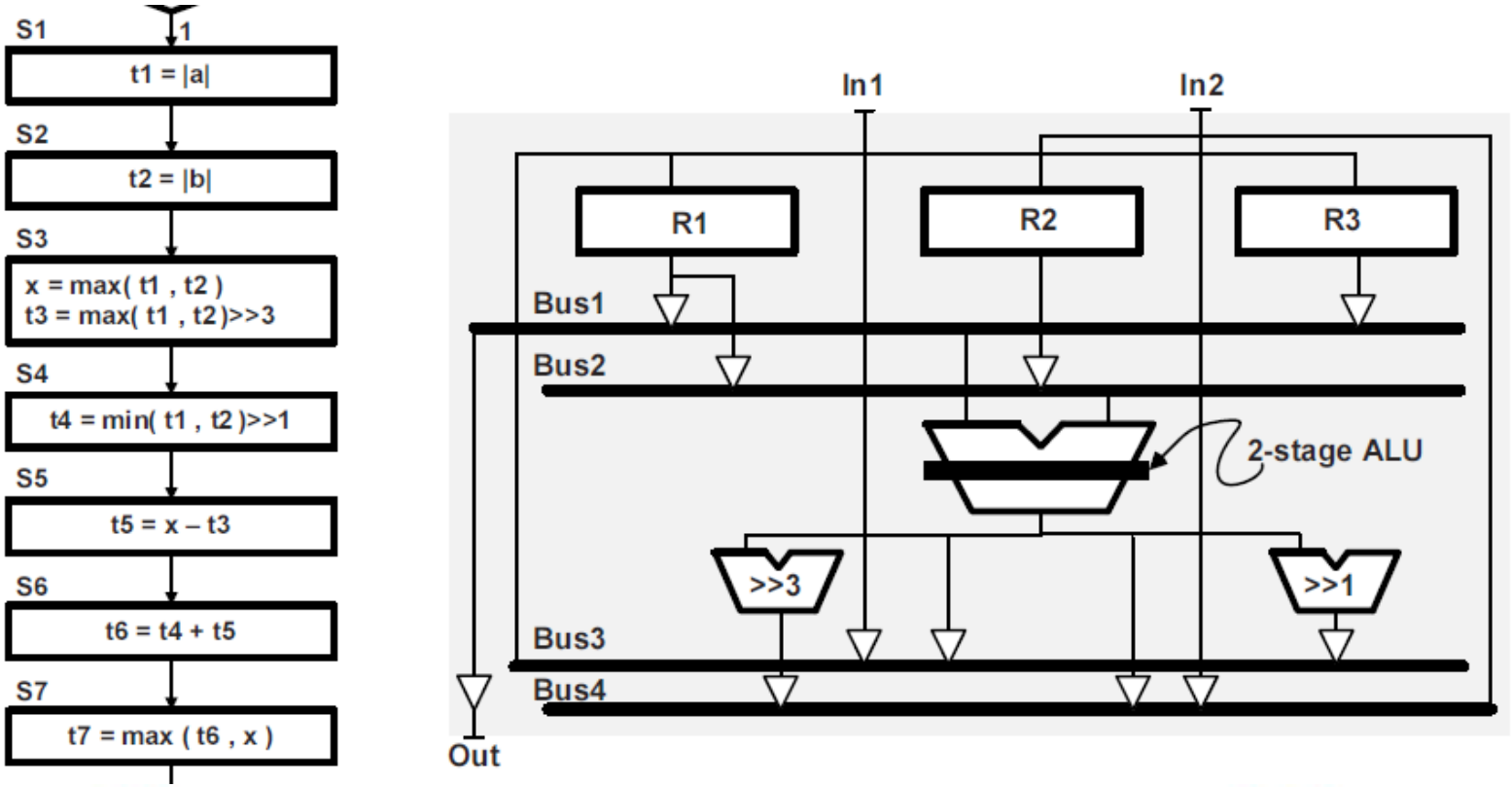
Chaining and Multi-Cycling

Pipelining

Pipelining

- If an operation takes more than 1 cycle to finish, part of the functional unit could be idling for at least one clock cycle.
  - Waste of resource!
- Pipelining
  - Introduce additional flip-flops/latches into the functional units so the idling parts can be reused for other operations.
  - As if there are additional functional units
  - You may also pipeline a unit for a higher frequency
- Cost of pipelining
  - Additional internal sequential elements
  - A small overhead to clock period
- Complications
  - Pipelining changes cycle-to-cycle behavior of the design.
  - It is generally not possible to pipeline a RTL design without breaking equivalence checking so we prefer to apply it in HLS.

Functional Unit Pipelining



(a) FSMD model (b) Datapath with 2-stage pipelined ALU

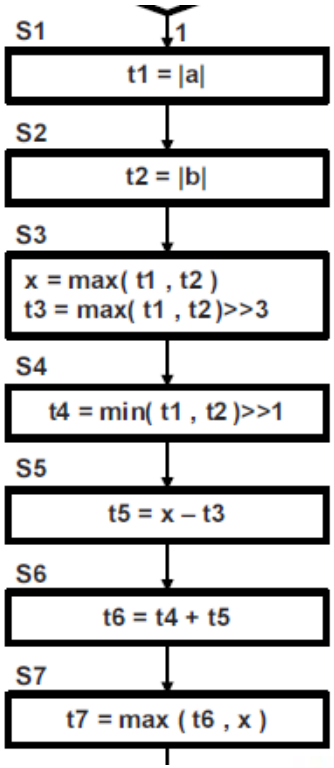
FIGURE 6.24 Functional unit pipelining

(Gajski et al.)

Scheduling with Pipelined Units

- A unit can be pipelined into multiple stages
  - Assume the number of stages is the number of clock cycles
- One set of new operands can be introduced per each clock cycle.
  - As usual, they should persist throughout that clock cycle.
  - Unlike multi-cycling, it is not necessary to maintain the inputs throughout the whole computation.
- The output will be ready after the designated number of cycles.
  - Following the same order you send in the operands
  - The same as multi-cycling

Scheduling Details



	S0	S1	S2	NO	S3	S4	S5	NO	S6	NO	S7	NO	S8
Read R1		a			t1	t1	X				X		t7
Read R2			b		t2	t2	t3		t5		t6		
Read R3									t4				
ALU stage 1		|a|	|b|		max	min	-		+		max		
ALU stage 2						max	min	-		+		max	
Shifters						>>3	>>1						
Write R1	a		t1			X						t7	
Write R2	b			t2		t3		t5		T6			
Write R3							t4						
Write Out													t7

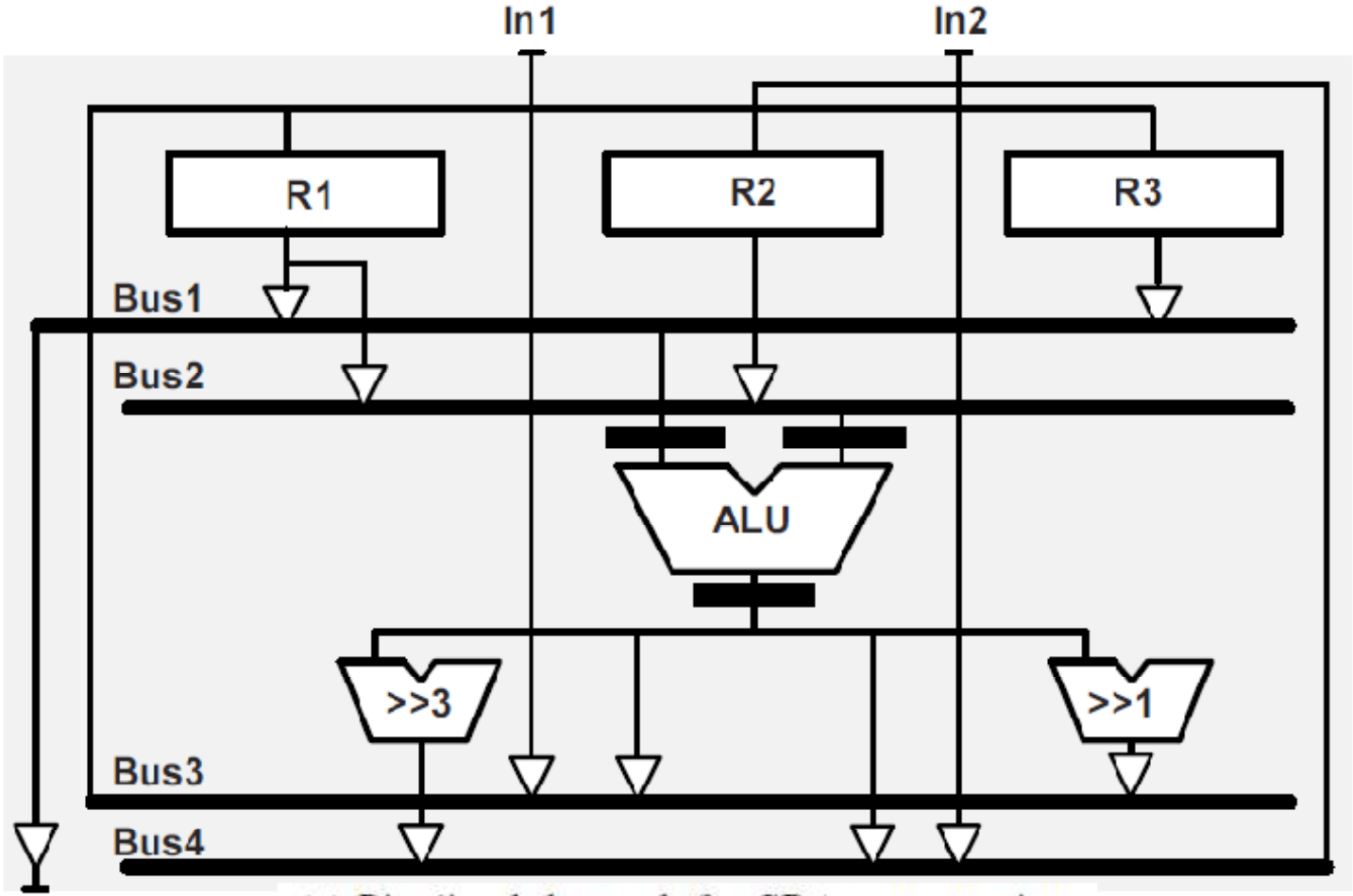
(c) Timing diagram

FIGURE 6.24 Functional unit pipelining

(Gajski et al.)

- Performance can be improved by bypassing certain pipeline stages.
  - e.g. the second stage of ALU for absolute value computation

Datapath Pipelining



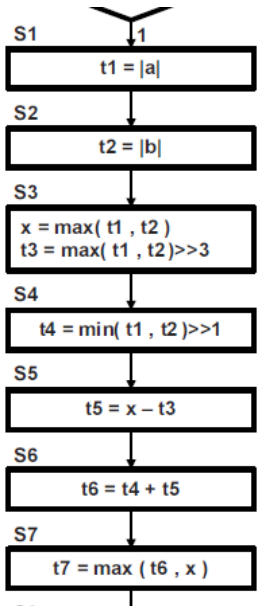
(a) Pipelined datapath for SRA computation

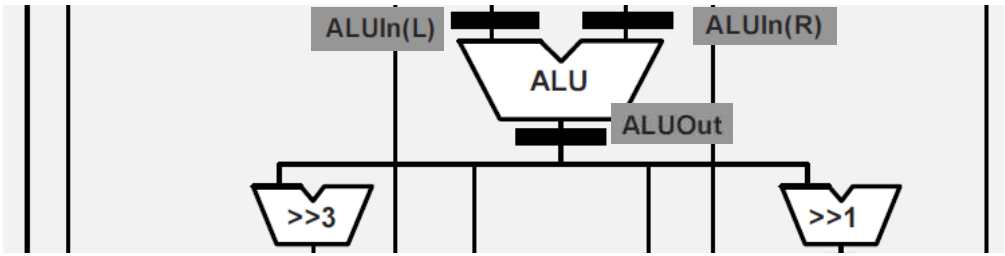
FIGURE 6.25 Datapath pipelining

(Gajski et al.)

- If interconnects/register files will consume significant amount of time, we may pipeline the whole datapath.

Scheduling Details





	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Read R1		a				t1	t1		x						x			t7
Read R2			b			t2	t2		t3			t5			t6			
Read R3												t4						
ALUIn(L)		a				t1	t1		x			t4			x			
ALUIn(R)			b			t2	t2		t3			t5			t6			
ALUOut			|a|	|b|			max	min		-			+			max		
Shifters									>>3	>>1								
Write R1	a			t1					x								t7	
Write R2	b				t2				t3			t5			t6			
Write R3										t4								
Write Out																		t7

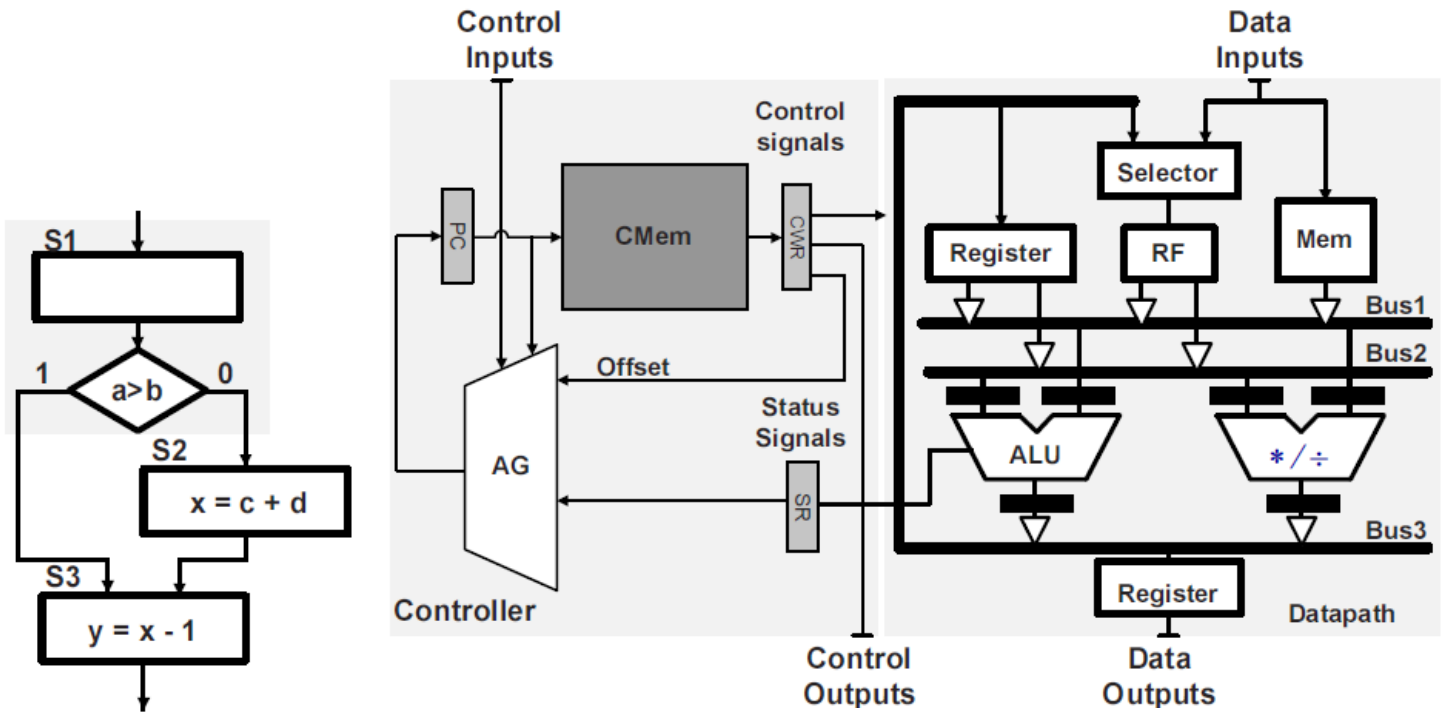
(b) Timing diagram

FIGURE 6.25 Datapath pipelining

(Gajski et al.)

- Performance can be improved by forwarding outputs not yet stored to registers.
  - e.g. forward ALUout to ALUIn(L) and ALUIn(R)

Control and Datapath Pipelining



(a) FSMD model excerpt (b) RTL design

FIGURE 6.26 Control and datapath pipelining

(Gajski et al.)

- Registers at input/output (SR/CWR) to simplify timing, especially when interconnect delay matters
- Internal register (PC) to mitigate memory access delays for programmable hardware.

Scheduling Details

Clock cycle #	0	1	2	3	4	5	6	7	8	9	10
Read PC	10	11	12	13	14	15	16	17	18	19	20
Read CWR		S1	NO	NO	NO	S2	NO	NO	S3		
Read RF(L)		a				c			x		
Read RF(R)		b				d			1		
Write ALUIn(L)		a				c			x		
Write ALUIn(R)		b				d			1		
Write ALUOut							c+d			x-1	
Write RF								x			y
Write SR			a>b								
Write PC	11	12	13	14/16	15	16	17	18	19	20	

(c) Timing diagram

FIGURE 6.26 Control and datapath pipelining

(Gajski et al.)

Summary

- 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.
- Pipelining helps to reuse idling hardware components.