EECS 151/251 A
Final Discussion!!!

April 26, 2024

Berkeley
UNIVERSITY OF CALIFORNIA

Content

- Have practice problems from every HW covering key topics
- We can choose what topics we want to focus on

Early Stuff (Not Comprehensive)

- Circuit Building Blocks
  - Transistor Layout
  - CMOS circuits (nMOS vs pMOS, Fabrication)
  - Logic Gates
  - FPGA LUTs and Interconnect
  - K-Maps, Boolean Algebra
  - Transmission Gates/ Tri-state gates
- Verilog and Blocks
  - Flip Flops
  - FSMs, Creating FSMs from a word problem, Encoding FSMs
  - Shift Registers

After Midterm 2 Stuff (Not Comprehensive)

- Single Cycle RISC-V Datapath
- Pipelined RISC-V (Hazards, mechanisms)
- Power and Energy in ICs (low-power design techniques)
- Circuit Timing/Delay
- Memory Architecture
- Parallelism (Spatial vs Temporal)
- List Processor Design and Optimizations
- Adders, Multipliers, and Shifters
- Clock Constraints

CMOS Circuit

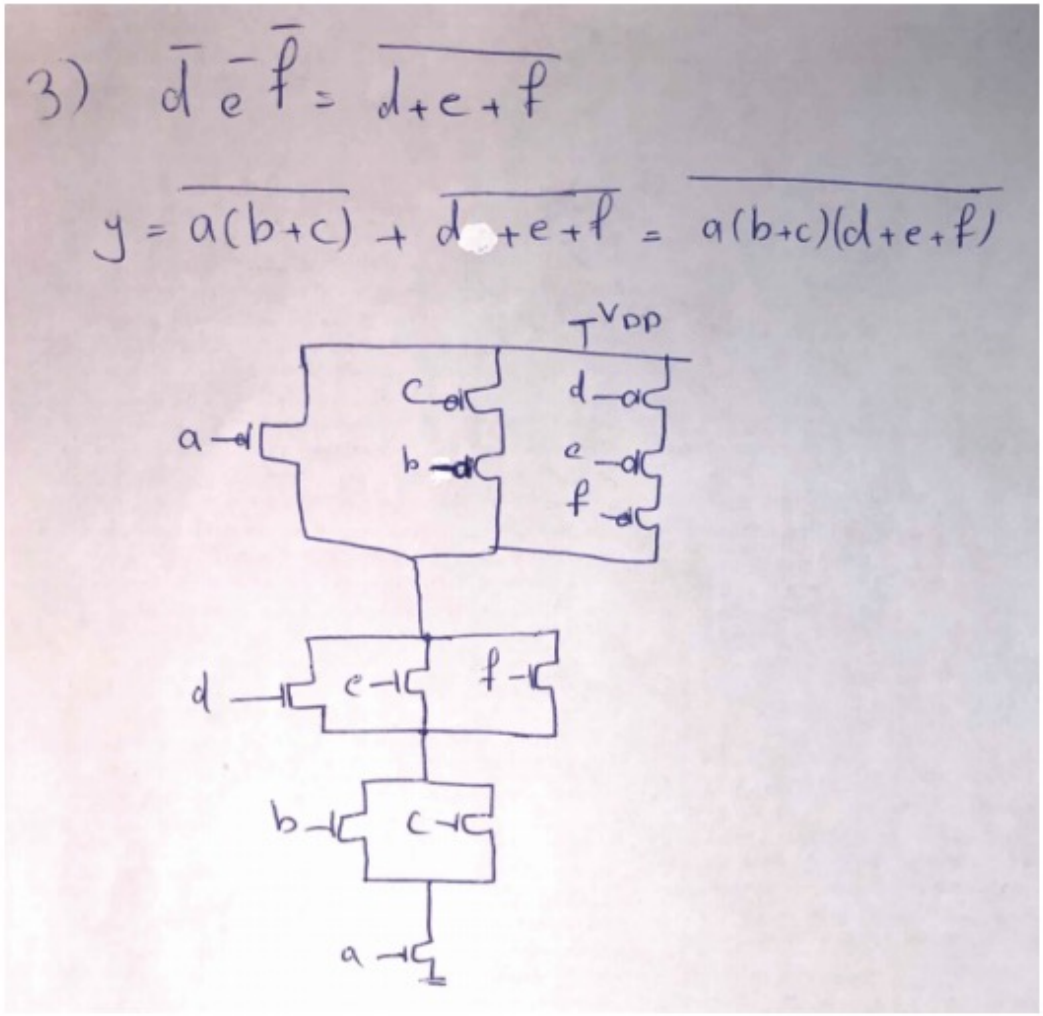
Spring 2020 Final

3. **CMOS gate [4 pts]:**

Without assuming that inverted inputs are available, derive a single logic gate with the following function:

$$y = (a(b + c))' + d'e'f'$$

CMOS Circuit

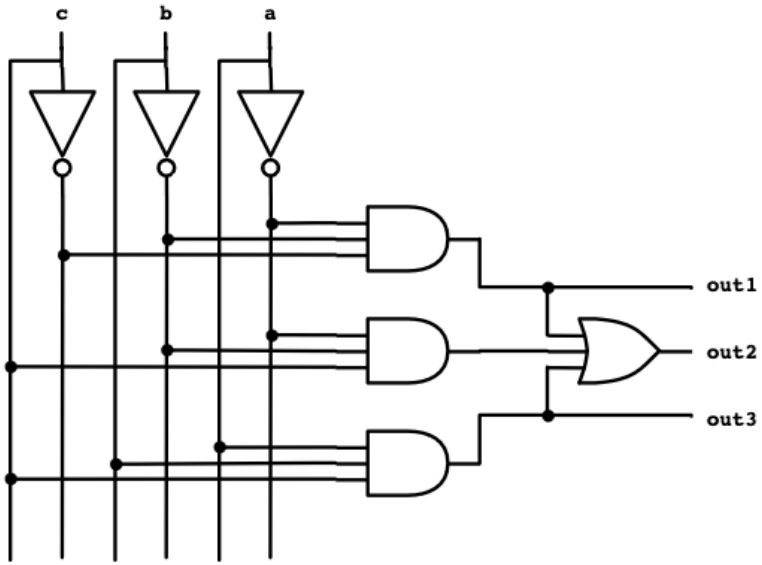


LUT Mapping

Spring 2020 Final

1. **LUT mapping [5 pts]:**

   a) Without trying to simplify the circuit, and by drawing on the circuit diagram below, demonstrate how to map these functions to the minimum number of 3-LUTs.

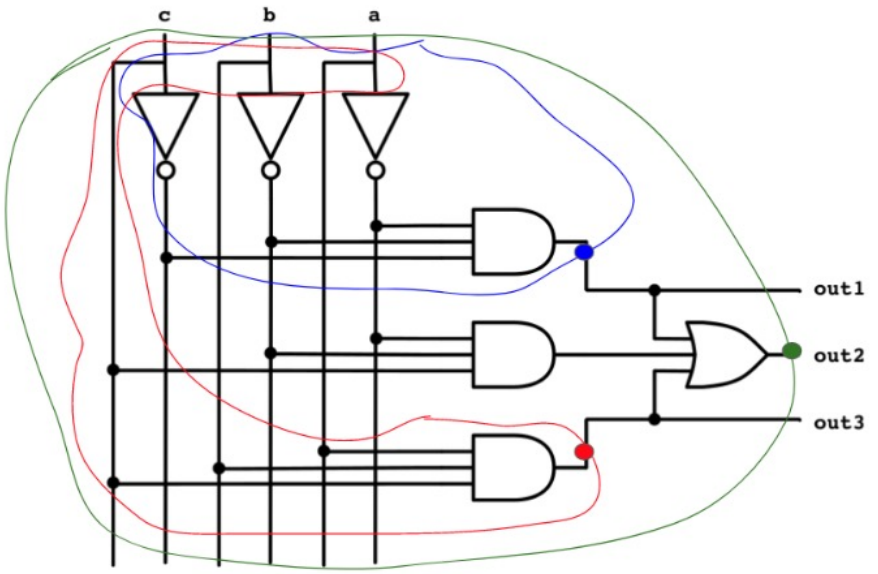


   b) Redraw the circuit and show how to map the function to the minimum number of 2-LUTs.

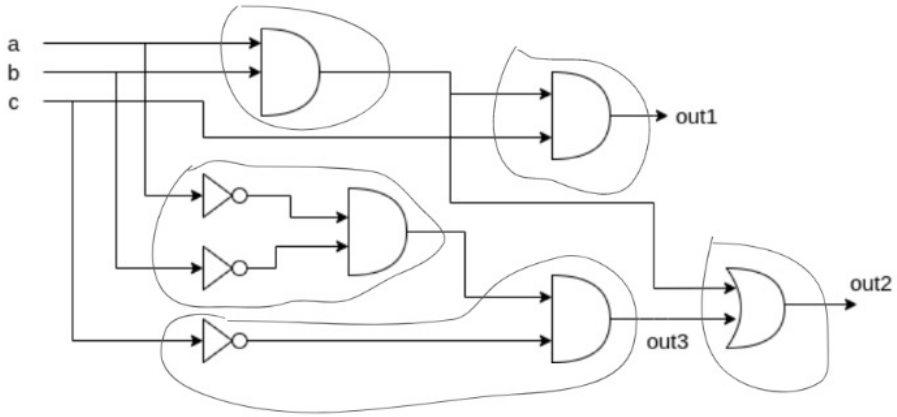
LUT Mapping

Solution:

(a) 3-LUT mapping



(b) 2-LUT mapping



Shift Register/Counter

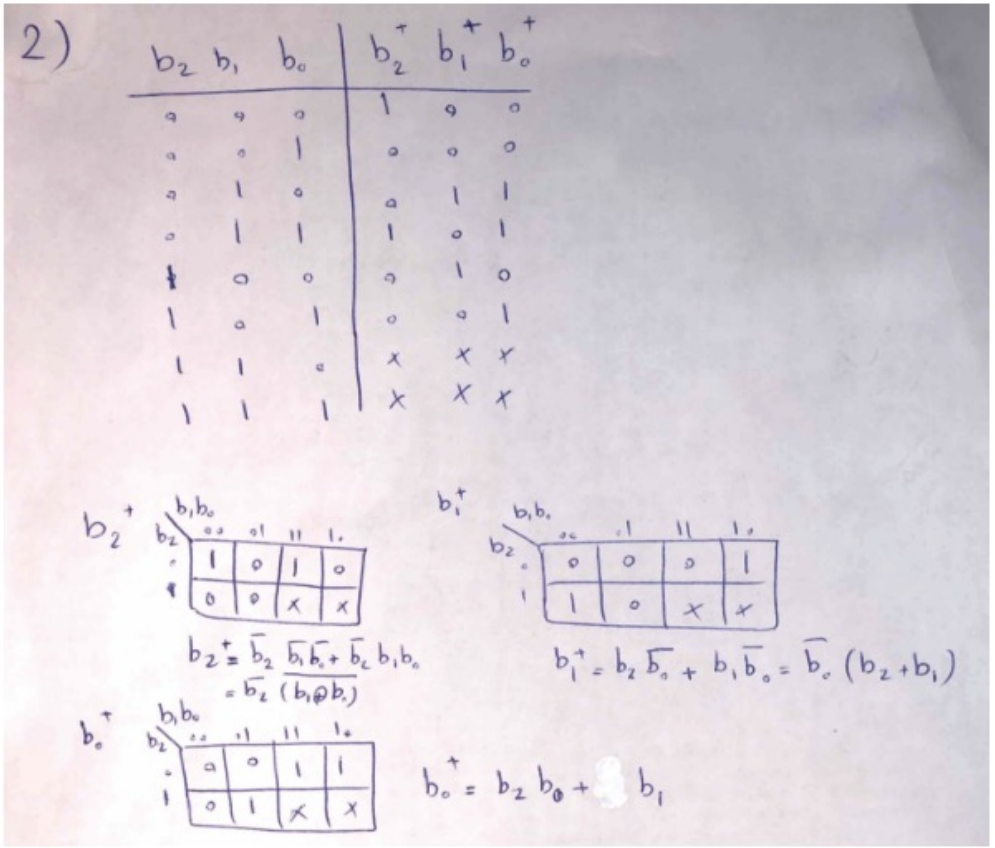
Spring 2020 Final

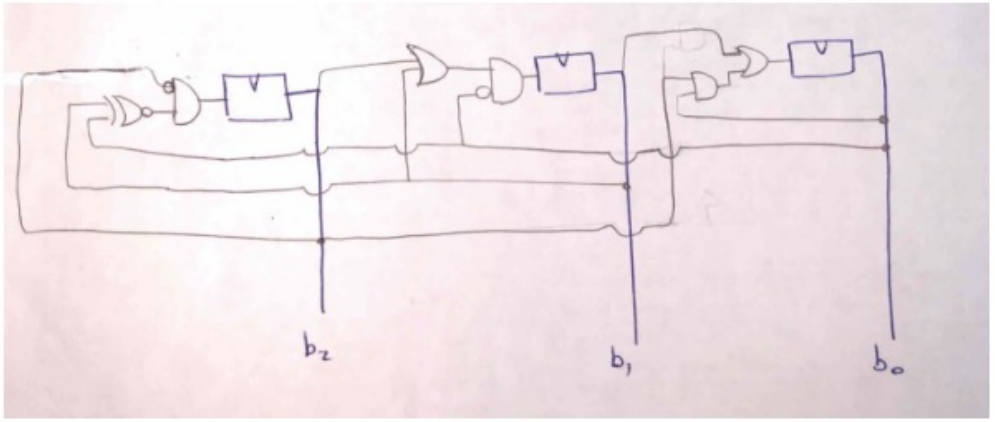
Counter design [6 pts]:

A synchronous counter has the following sequence of output values: 000, 100, 010, 011, 101, 001, 000, ...

Minimize the logic and draw the diagram of a circuit that implements this counter, using flip-flops and logic gates.

Shift Register/ Counter





Berkeley
UNIVERSITY OF CALIFORNIA

FSMs

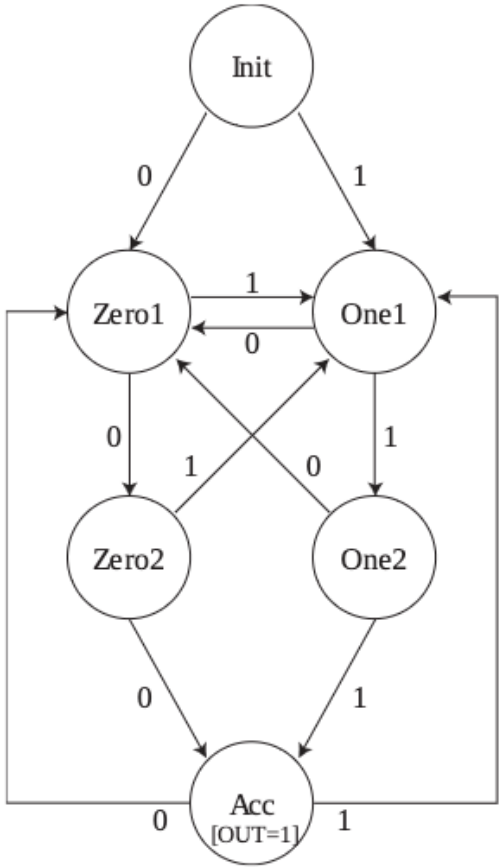
Spring 2021 Midterm 1

8 State Transition Diagrams [5 pts]

In the space below, neatly draw the state transition diagram for a Moore finite state machine with the following specification. The FSM has a single input, IN, and a single output, OUT. After reset it outputs a 0. When it sees a consecutive sequence of three 1's or three 0's at its input, it outputs a 1, then starts looking again. (For example, for the input sequence 0111_1111_0001, it outputs a sequence 0000_1001_0001, assuming that it resets at the beginning and outputs a 0 in the first cycle.)

FSMs

Solution:



Labels on the edges represent the value of IN, and OUT is 0 unless specified.

New Content

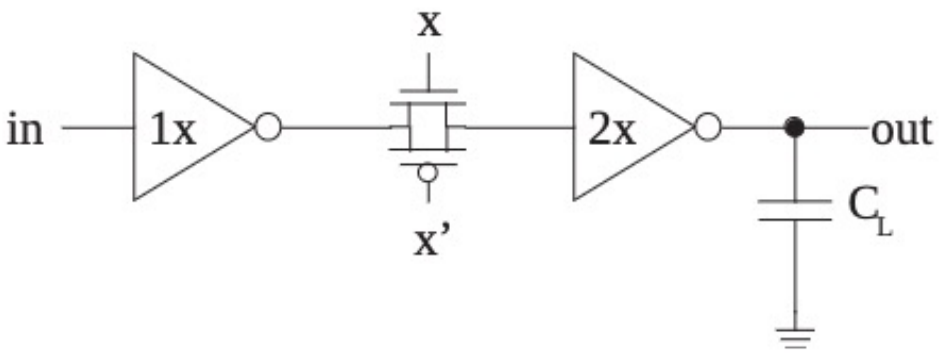
Circuit Delay

Spring 2021 Midterm 1

16 Circuit Delay [5 pts]

Consider the circuit shown below with a transmission gate in series with a couple inverters. The right most inverter is sized 2X and drives a capacitive load represented by C_L. We represent the drive resistance of a unit sized inverter as R_N and its input capacitance as $3C_N$ (we assume the inverter is sized to have balanced pullup and pulldown strength). The transistors in the transmission gate have the same sizes as the ones in the unit sized inverter. Furthermore, assume that $\gamma = 1$. Ignore wire resistance and capacitance.

Write an expression for the total delay from input to output of this circuit, in terms of R_N, C_N, and C_L.



Circuit Delay

Solution:

The parasitic capacitance of 1x inverter is $3C_N$ as $\gamma = 1$. There is $3C_N$ capacitance on each side of the transimission gate. The transmission gate has resistance R_N ($R_N/2$ is also okay). The input capacitance of 2x inverter is $6C_N$. So, the delay of the first inverter is

$$\ln 2 \cdot (R_N(3C_N + 3C_N + 3C_N + 6C_N) + R_N(3C_N + 6C_N)) = 24 \ln 2 \cdot R_N C_N \tag{1}$$

The paracitic capacitance of 2x inverter is $6C_N$. The delay of the second inverter is

$$\ln 2 \cdot \frac{R_N}{2}(6C_N + C_L) = 3 \ln 2 \cdot R_N C_N + \ln 2 \cdot \frac{R_N C_L}{2} \tag{2}$$

Therefore, the total delay is

$$27 \ln 2 \cdot R_N C_N + \ln 2 \cdot \frac{R_N C_L}{2} \tag{3}$$

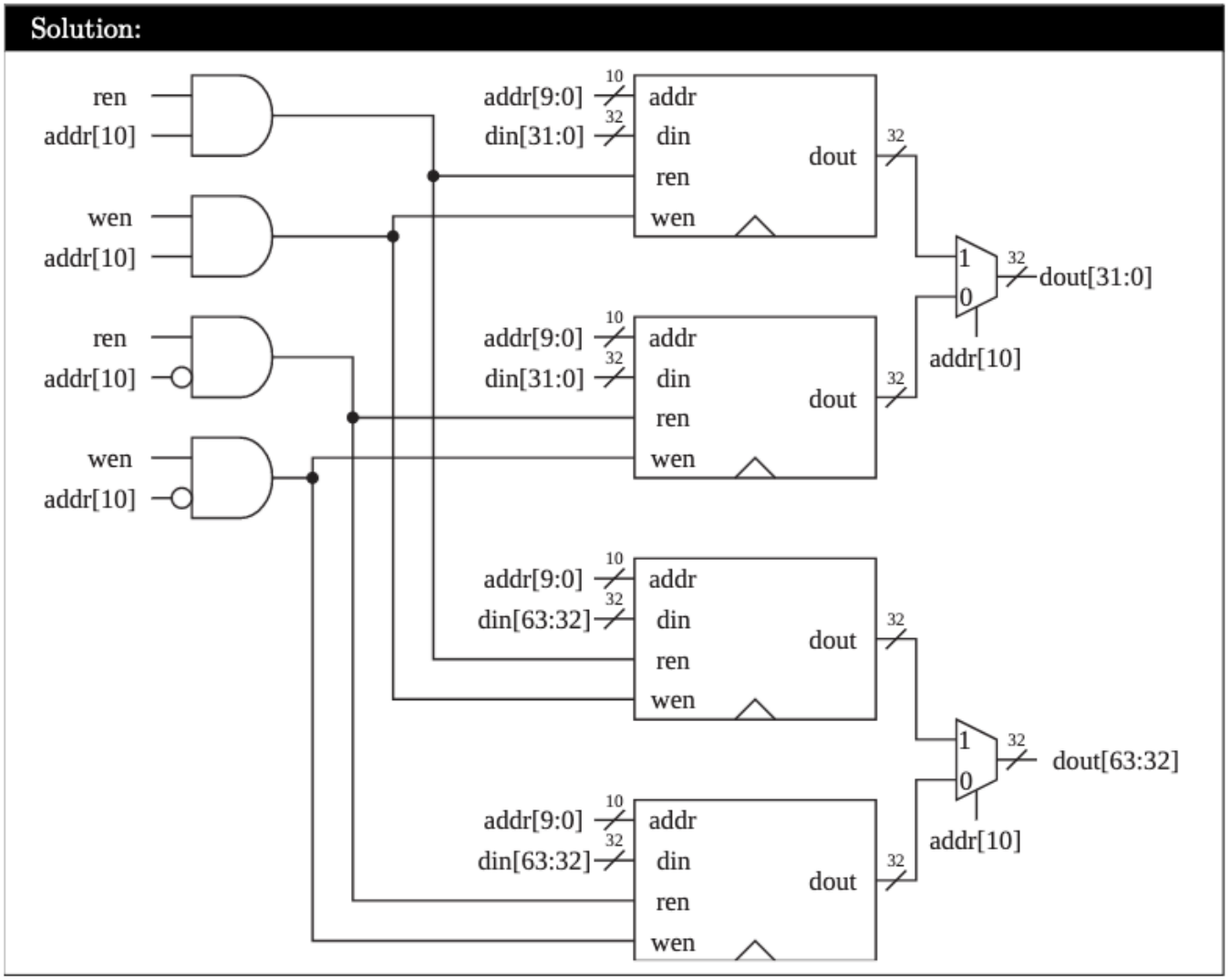
Memory Composition

Spring 2023 HW 8

Problem 1: Memory Composition

Neatly draw a block diagram for a 2048×64 single-port RAM using 1024×32 single-port RAMs. You are also allowed to use logic gates and multiplexers. The single-port RAMs have 4 input signals (`addr`, `din`, `ren`, `wen`) and one output signal (`dout`).

Memory Composition



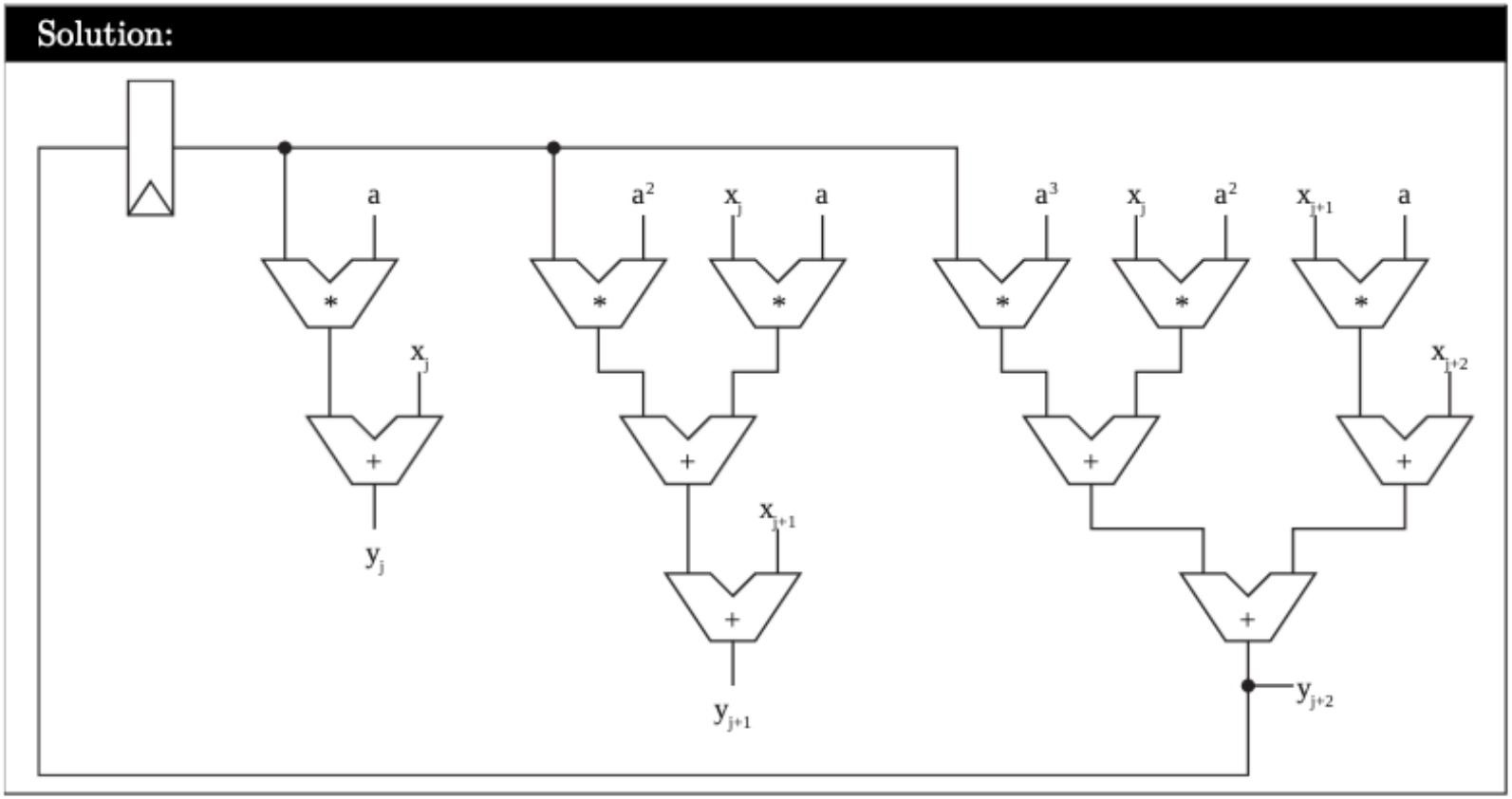
Parallelism (Loop Unrolling)

Spring 2023 HW 8

Problem 4: Loop Unrolling

Draw a block diagram for (direct hardware implementation of) $y_i = a * y_{i-1} + x_i$ with loop unrolling of interval 3, where a is constant. $\{x_i\}$ is the input sequence and $\{y_i\}$ is the output sequence. The unrolled circuit takes $\{x_j, x_{j+1}, x_{j+2}\}$ as input and generates $\{y_j, y_{j+1}, y_{j+2}\}$ as output where j is a multiple of 3. Minimize the logic (arithmetic operation) depth.

Parallelism (Loop Unrolling)



Power and Energy

Spring 2020 Final

4. **Multi-core Power and Energy [15 pts]:**

Your new job is at ManyCores Inc. and you are asked to help design a embedded product that uses your 8 core chip.

You measure the target workload and find that, with all 8 cores running, the chip consumes 10W of power while running at 1 GHz and Vdd = 1V. You would like to get the average power consumption down below 7.75W.

You notice that the cores are only 80% busy during the measurements (they spend 20% of the time running, but just waiting for input).

Looking at the datasheet you see that the chip is capable of running with Vdd in the range of 0.8-1.2V.

You also see that the chip has a controller that can turn off all power (both switching and leakage power) to any or all of the cores.

After some testing you determine that 50% of the total power is in leakage.

tudent ID number:

You ask you co-workers what to do to get to the target average power while maintaining the desired workload performance.

Co-worker #1 says: Just lower the clock frequency.

Co-worker #2 says: Lower Vdd.

Co-worker #3 says: Raise Vdd and "race to halt".

Who is right? (Evaluate all three options as a technique to save average power, while maintaining the desired performance. Show your work for each option.)

Power and Energy

$P_{tot} = 10W$ @ $f = 1\,GHz$, $V_{dd} = 1V$

$P_{tot} = P_{sw} + P_{leak}$

$= 5W + 5W$

$P_{avg} = (5W + 5W) \cdot 0.8 + 5W \cdot 0.2 = 9W$

#1 lower clock

$P_{sw} = \alpha C (0.8 V_{dd})^2 (0.8 f)$

$= 0.512\, \alpha C V_{dd}^2 f$

$= 0.512 \cdot 5W$

$= 2.56\,W$

$P_{leak} = 5W$

$P_{tot} = 7.56\,W$

$P_{avg} = 7.56\,W$

(if V_{dd} not changed)

$P_{sw} = 4W$

$P_{avg} = 9W$

#2 lower V_{dd}

$P_{sw} = \alpha C (0.8 V_{dd})^2 (0.8 f)$

$= 0.512 \cdot 5W$

$= 2.56$

$P_{tot} = 7.56\,W$

$P_{avg} = 7.56\,W$

#3 race to halt

$P_{sw} = \alpha C (1.2 V_{dd})^2 (1.2 f)$

$= 1.728 \cdot 5W$

$= 8.64\,W$

$P_{tot} = 13.64\,W$

$P_{avg} = 13.64 \cdot 0.667$

$= 9.09\,W$

Lower V_{dd} best option

Datapath Design

Spring 2020 Final

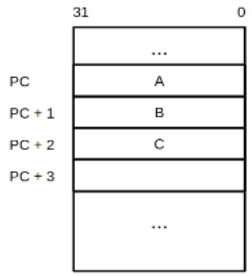
7. **Problem 7. Single Instruction Computer Design [20 pts]:**

Several proposals in the past have been made for Turing complete computers that operate with only a single type of instruction. One such proposal is for a computer based on the "SubLeq" (subtract and branch if less than or equal) instruction.

A SubLeq instruction has 3 32-bit operands: `A B C`.

Execution of one instruction `A B C` subtracts the value in the memory location at the address stored in A from the content of a memory location at the address stored in B and then writes the result back into the location with the address in B. If the value after subtraction in B is less or equal to zero, the execution jumps to the address specified in C; otherwise execution continues to the next instruction, i.e. the address of the memory location following C.

For this problem we assume that all instructions and all data is stored in a single memory module. It is a simple dual-port (one read port, one write port) asynchronous read and synchronous write memory. Unlike the RISC-V, memory is word-addressed (32-bit words), not byte addressed. Instructions are stored in the memory as illustrated below:



To complete this problem, design a datapath circuit and specify the controller for this computer. Your goal is to maximize the performance while keeping the amount of hardware to a minimum (priority on performance). (Hint: first, generate a design without worrying about cost and performance, then later think about optimizations to improve cost and performance. Turn in both versions.)

a) Show your datapath design in the form of a block diagram. You may use registers, arithmetic units, multiplexors, simple logic gates, and the memory module described above. Label all control points.

b) For the controller portion, write the RT Language description.

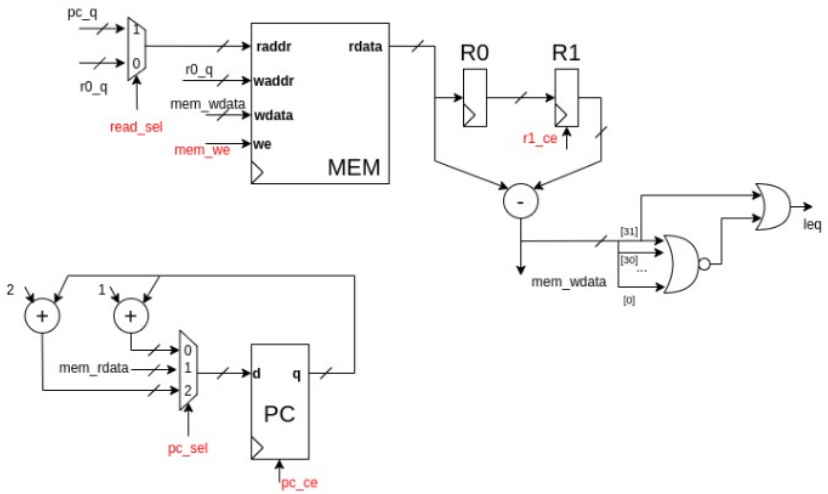
Berkeley
UNIVERSITY OF CALIFORNIA

Datapath Design

Solution:

Optimization for clock cycles

Datapath.



The control signals are highlighted in red. The critical path is the path from asychronous read port from the memory block to the subtractor to the write data port of the memory block.

(b) Controller (RT Language).

Here are the steps to execute an instruction.

- Read operand `A` from the memory and store it to `R0`;
- Read `mem[A]` from the memory and store it to `R0`,
  Increment `PC` to read the next operand;
- Read operand `B` from the memory and store it to `R0`,
  Store `mem[A]` to `R1`;
- Setup a write to memory,
  If `leq` == 0, increase PC by 2 to read the next instruction (branch is not-taken),
  otherwise increase PC by 1 to read the next operand;
- Read operand `C` from the memory and store `C` to `PC` (branch is taken);

RTL code (only need to write assignments to synchronous elements, the control signals are implied from the code):

```
repeat {
    R0 <- Mem[PC];
    R0 <- Mem[R0], PC <- PC + 1;
    R0 <- Mem[PC], R1 <- R0;

    R0 <- Mem[R0], Mem[R0] <- Mem[R0] - R1,
    if (leq == 0) {
        PC <- PC + 2,
        continue;
    else {
        PC <- PC + 1;
    }
    PC <- Mem[PC];
} until (1);
```

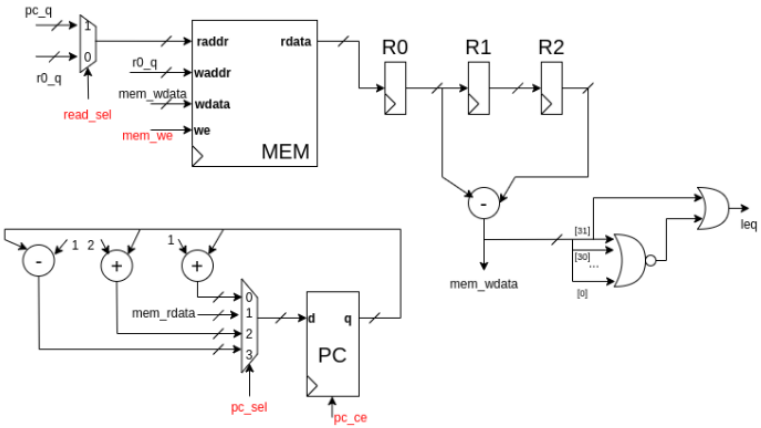
The number of cycles to execute an instruction: best case is 4 cycles (branch is taken) , worst case is 5 cycles (branch is not taken).

Datapath Design

Solution:

Optimization for clock frequency

Datapath. We predict branches not taken. If the prediction is wrong, we roll back to the previous memory location to read operand C.



The control signals are highlighted in red.

(b) Controller (RT Language).
RTL code (non-CE `R1`, `R2` register assigments are omitted for brevity)

```
repeat {
    R0 <- Mem[PC],

STATE1:
    R0 <- Mem[R0],
    PC <- PC + 1;


     R0 <- Mem[PC];

     R0 <- Mem[R0],
     PC <- PC + 2;

     Mem[R1] <- R2 - R0,
     if (leq == 0) {
         R0 <- Mem[PC],
         j STATE1;
     }
     else {
         PC <- PC - 1,
     }

     PC <- Mem[PC];
}
```

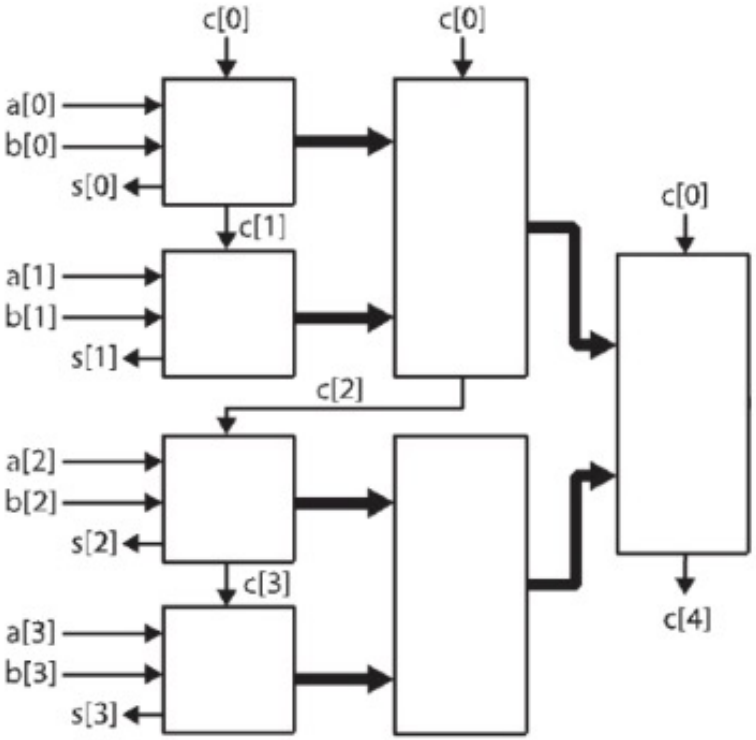
The number of cycles to execute an instruction: best case is 4 cycles, worst case is 6 cycles. This is optimal in performance if we assume that branches are usually not taken. Alternatively, if no branch prediction is performed, it always takes 5 cycles to execute an instruction which is also a valid solution.

Adders

Spring 2023 HW 10

Problem 5: Carry Look-ahead Adder

The figure below shows the structure of 4-bit carry look-ahead adder. Assuming each box takes the same delay uniformly from each input to dependent output, what is the maximum delay? Can you generalize it for N-bit carry look-ahead adder where N is power of 2 larger than 4?



Adders

Solution:

After simple analysis, $s[3]$ turns out to have the largest delay of 4 units coming through $c[2]$ from either top two boxes on the left hand side. Note that the propagate and generate from the second box does not depend on $c[1]$.

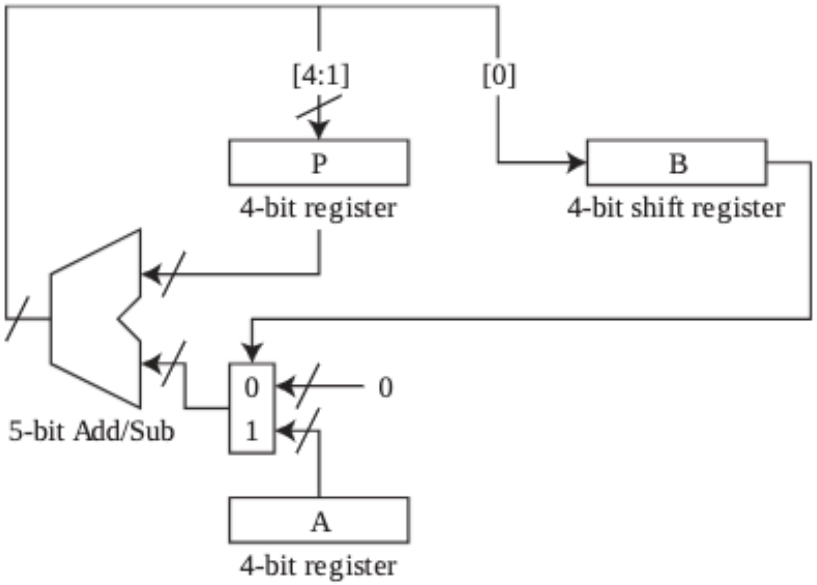
For 8-bit adder, we copy the same structure for the upper half bits and assign $c[4]$ to the place of $c[0]$, while one more box is added to the right end for $c[8]$. Since the path through $c[4]$ has larger delay than the paths coming from $a[4]$ or $a[5]$ to $s[7]$, the maximum delay is $3+4-1=6$. By induction, the path coming through $c[N/2], c[N/2+N/4], \ldots$ to the box for the most significant bit is the critical path, which has 2 unit larger delay than the previoius power of 2. Therefore, the maximum delay for N-bit carry look-ahead adder is $2\log_2 N$.

Multipliers

Spring 2023 HW 11

Problem 1: Signed Shift-and-Add Multiplication

Simulate -3×-5 ($A = -3, B = -5$) on the shift-and-add multiplier shown below. Write down the values of B and P for each cycle. Note that the inputs to the 5-bit Add/Sub are sign-extended, and it performs subtraction only in the last cycle.



Multipliers

Solution:

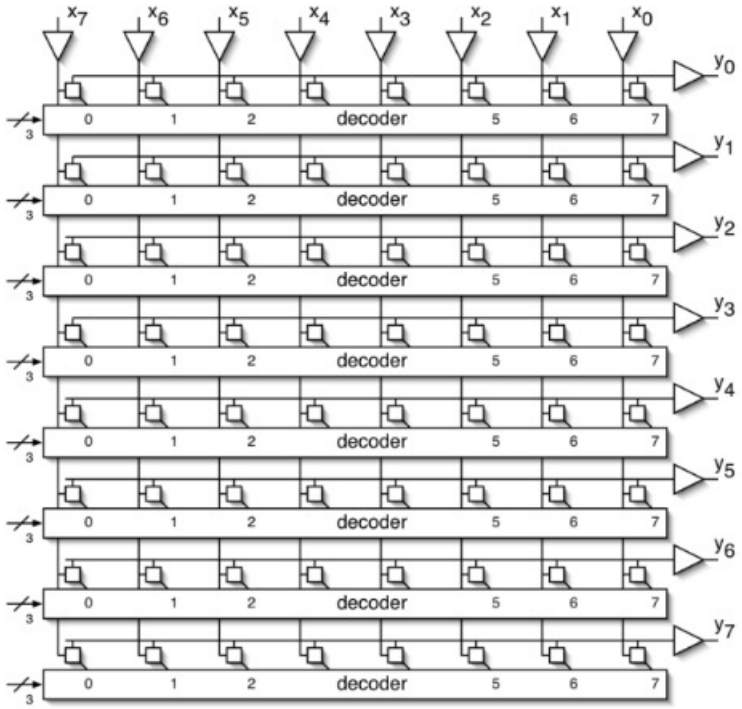
1. $B = 1011, P = 0000$
2. $B = 1101, P = 1110$
3. $B = 1110, P = 1101$
4. $B = 1111, P = 1110$
5. $B = 1111, P = 0000$

Verify $\{P, B\} = 00001111 = 15$.

Shifters

Spring 2023 HW 11

How would you configure the decoders in the cross-bar switch below to reverse the order of bits i.e. $y_i = x_{7-i}$ for $i = 0, 1, \ldots, 7$?



Shifters

Solution:

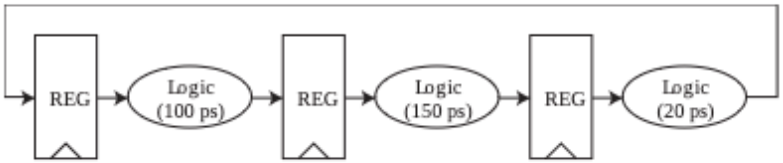
From the top, assign $0, 1, \ldots, 7$.

Clock Uncertainty

Spring 2023 HW 11

Problem 6: Clock Uncertainity

Given setup time 20 ps, clock-to-q delay 10 ps, and cycle-to-cycle jitter 10 ps, what is the maximum frequency of the following circuit by adjusting the clock skew?



Clock Uncertainty

Solution:

We can equally distribute the delay between registers by setting the clock skew of the middle register to +10 ps and that of the right register to +70 ps relative to the first register. Then, the delay between registers is $90 + 20 + 10 = 120$ ps. We need additional $2 \times 10 = 20$ ps for jitter, so the maximum frequency is $1/140 = 7.14$ GHz.