6.1 Tradeoffs

A particular desired behavior, like adding two 4-bit numbers, may have alternative circuit implementations. A ***tradeoff*** is a design decision that improves one implementation metric while worsening another. An implementation ***metric*** is a measurement of an implementation's goodness. A common circuit metric is a circuit's size, with smaller size being better. Another is circuit delay, with less delay being better. Unfortunately, decreasing size usually increases delay, representing a tradeoff. Other common metrics include power and cost.

In contrast to a tradeoff, a design decision that improves some metric(s) without worsening any others is called an ***optimization***.

PARTICIPATION ACTIVITY 6.1.1: A behavior may have alternative implementations that trade off size and delay.

Animation captions:

1. A circuit implementation has a particular size and delay. This circuit has a size of 60 and delay of 2 (this generic example omits units).
2. An alternative circuit for the same behavior may improve size (60 reduced to 40), but worsen delay (2 increased to 4), representing a tradeoff.
3. A behavior may have many alternative implementations that trade off size and delay.

PARTICIPATION ACTIVITY 6.1.2: Tradeoffs.

Consider implementing a particular behavior. Indicate whether implementation B is a tradeoff compared to implementation A. (Units are intentionally omitted for this generic example).

1) A's size is 100, delay is 2.
   B's size is 60, delay is 4.
   - ○ Tradeoff
   - ○ Not a tradeoff

2) A's size is 100, delay is 2.
   B's size is 80, delay is 2.
   - ○ Tradeoff
   - ○ Not a tradeoff

Multiple approaches exist for determining circuit size. One approach estimates transistors, assuming every gate input requires 2 transistors, and ignoring inverters for simplicity. A 2-input gate requires 2 inputs · 2 trans/input = 4 transistors. A 3-input gate requires 3 · 2 = 6 transistors. A 4-input gate: 8 transistors.

Wires also contribute to size, but ignoring wires as above is a common approximation.

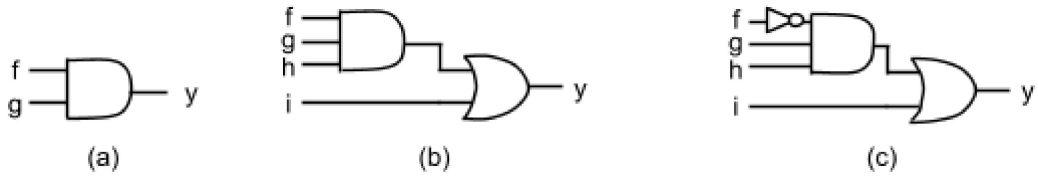
Although each gate may have a unique delay and wires also have delay, a quick approach for approximating circuit delay counts the number of gates from a circuit's inputs to output, known as ***gate delays***. Inverters are ignored for simplicity.

If multiple paths exist from inputs to output, the circuit's delay is the longest path, called the circuit's ***critical path***.

Real gates have differing delays. A 3-input gate has slightly longer delay than a 2-input gate. Wires also have delay. But counting gate-delays as above is a common approximation.

PARTICIPATION ACTIVITY 6.1.3: Circuit size and delay.

Determine size in units of transistors, and delay in units of gate-delays.



1) Size of (a)

transistors

Check Show answer

2) Size of (b)

transistors

Check Show answer

3) Size of (c)

transistors

Check Show answer

4) Delay of (b)

gate-delays

Check Show answer

5) Delay of (c)

gate-delays

Check Show answer

Size and delay can be estimated directly from an equation. Ex: y = abc + def has a 3-input AND for abc, another 3-input AND for def, and a 2-input OR, so size is 3·2 + 3·2 + 2·2 = 16 transistors. The circuit has a column of AND gates followed by an OR, so delay is 2 gate-delays.

PARTICIPATION ACTIVITY 6.1.4: Estimating size and delay from an equation.

1) y = a. Size = ___ transistors?
   - 0
   - 2

2) y = ab. Size = ___ transistors?
   - 4
   - 8

3) y = ab + bc. Size = ___ transistors?
   - 8
   - 12

4) y = ab + bc. Delay = ___ gate-delays?
   - 1
   - 2

5) y = a'b'. Size = ___ transistors?
   - 4
   - 6

6) y = ab + a'b'. Delay = ___ gate-delays?
   - 2
   - 3

7) y = a + bcd. Size = ___ transistors?

- 10
- 12

8) y = a + b + cd. Size = ___ transistors?

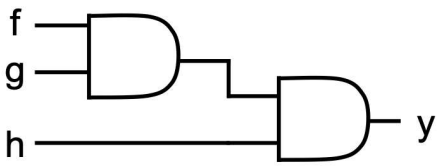
- 10
- 12

CHALLENGE ACTIVITY 6.1.1: Tradeoffs.

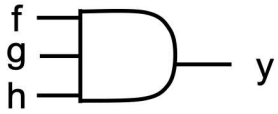
347136.1658324.qx3zqy7

Start

Compare circuit (b) to (a).



(a)



(b)

What is the difference in total number of transistors between (a) and (b)?

Ex: 5

What is the difference in gate-delays between (a) and (b)?

Is (b) an optimization or a tradeoff over (a)?

1 2 3

Check Next

6.2 Carry-lookahead adders

A carry-ripple adder has a drawback of long delay, due to each digit having to wait for carries to ripple through earlier digits. A ***carry-lookahead adder*** uses logic to quickly pre-compute the carry for each digit, and thus has less delay than a carry-ripple adder, but larger size, representing a tradeoff.

PARTICIPATION ACTIVITY 6.2.1: Carry-lookahead adder has less delay, but larger size, due to logic that pre-computes carries to avoid rippling.

Animation captions:

1. Carry-ripple adder is slow due to carries having to ripple from digit to digit.
2. Carry-lookahead adder pre-computes carries using lookahead logic, so is faster, but uses more gates.

PARTICIPATION ACTIVITY 6.2.2: Adder tradeoffs.

1) Which adder type has shorter delay?
   - ○ Carry-ripple
   - ○ Carry-lookahead

2) Which adder type requires fewer gates?
   - ○ Carry-ripple
   - ○ Carry-lookahead

The basic lookahead idea is to create a circuit that quickly computes whether a digit's output carry will be 1 or 0. Each digit's carry-out circuit makes use of two values, g and p.

- When a AND b are both 1's, the digit ***generates*** a carry-out of 1 regardless of the carry-in. Let g = ab.
- When a OR b is 1, the digit sets carry-out to 1 if the carry-in is 1, akin to the digit ***propagating*** the carry-in to carry-out. Let p = a + b.

With those two values, the expression for each digit's carry-out is co = ab + (a + b)ci = g + p·ci. Each digit's carry-in can then be substituted with the digit-to-the-right's carry-out (being connected), yielding two-level logic for each carry-in bit. Thus, all carry bits can be computed without any rippling, at the expense of the large number of gates in the carry-lookahead logic.

PARTICIPATION ACTIVITY 6.2.3: Carry-lookahead adders.

Animation captions:

1. A digit's carry-out is for sure 1 if a AND b are 1's. Thus, ab "generates" a carry. Let g = ab.
2. Also, if a OR b is 1, carry-in of 1 means carry-out of 1; the carry-in "propagates" to the carry-out. Let p = (a + b).
3. Thus, the 0'th digit's carry-out is c0 = g0 + p0cin.
4. For c1, substitute c0 with c0's expression.
5. For c2, substitute with c1's expression. Likewise for c3.
6. Thus, carry-lookahead logic is just two-level AND/OR circuit, so no rippling. sum/p/g blocks also have just two-level AND/OR circuits too.

PARTICIPATION ACTIVITY 6.2.4: Carry-lookahead adder.

Consider a 4-bit carry-lookahead adder.

1) Given a0 = 0 and b0 = 1, g0 = ?

Check Show answer

2) Given a0 = 0 and b0 = 1, p0 = ?

Check Show answer

3) Given g0 = 0, p0 = 1, and cin = 0, c0 = ?

Check Show answer

4) Given g1 = 0, p1 = 1, g0 = 0, p0 = 1, and cin = 0, what is c1?
Note: c1 = g1 + p1c0 = g1 + p1g0 + p1p0cin.

Check Show answer

PARTICIPATION ACTIVITY 6.2.5: Carry lookahead size and delay.

Consider a 4-bit carry-lookahead adder. (Note: Assume every gate input requires 2 transistors and ignore inverters.)

1) c0's circuit has ___ gate-delays from the lookahead block's p/g inputs to the c0 output.
   - ○ 2
   - ○ 4

2) c3's circuits has ___ gate-delays from the lookahead block's p/g inputs to the c3 output.
   - ○ 2
   - ○ 8

3) c0's circuit has ___ transistors.
   - ○ 2
   - ○ 8

4) c3's circuit has ___ transistors.
   - ○ 8
   - ○ 38

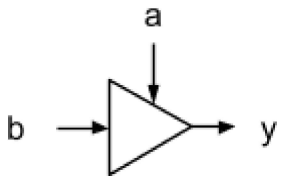
5) Consider 8-bit lookahead logic. c7's circuit would have ___ transistors. Hint: ANDs: 4 + 6 + 8 + ... + 18, OR: 18.
   - ○ 38
   - ○ 106

The lookahead logic gets dramatically larger for wider adders. (And slower too in reality, because for example 18-input gates are much slower than 2-input gates.) Thus, a designer might construct a 32-bit adder by connecting 8 4-bit carry-lookahead adder blocks in carry-ripple fashion, for example.

6.3 Register files

A ***three-state buffer*** is a component that outputs y = b if input a is 1, and outputs Z if a is 0. Z represents an electrical situation known as ***high impedance***. A wire with Z is akin to no wire. As such, three-state buffers support efficient combining of multiple wires into one wire, as will be seen when introducing register files below. Of course, only one connected wire can have a non-Z (0 or 1) value, else the values collide. Further details on three-state buffer and high impedance are beyond this material's scope.

Figure 6.3.1: Three-state buffer.



a	y
0	'Z' (high impedance)
1	b

PARTICIPATION ACTIVITY 6.3.1: Three-state buffer.

Given a three-state buffer with control input a, data input b, and data output y.

1) If a = 1 and b = 1, then y = ?
   - 1
   - 'Z'

2) If a = 1 and b = 0, then y = ?
   - 1
   - 0

3) If a = 0 and b = 0, then y = ?
   - 0
   - 'Z'

A three-state buffer is more commonly called a ***tri-state buffer***, but that name is trademarked. The term *three-state driver* (or *tri-state driver*) is also used.

An NxM ***register file*** efficiently implements access to N M-bit registers. A 16x32 register file has 16 registers, each 32 bits wide. To avoid having 512 (16 times 32) wires connecting with those registers for loads, a register file consolidates loads through one 32-bit data input. Of course, the tradeoff is that only one register can be loaded at a time, as indicated by a 4-bit address input. Loading a register in a register file is known as a ***write*** operation. The data input, address input, and load enable input for

writing are together called a ***write port***. Similarly, a register file consolidates read wires into a single 32-bit data output, 4-bit address input, and enable input, forming a ***read port***. The read port may use three-state buffers to efficiently connect the data wires.

PARTICIPATION ACTIVITY 6.3.2: 4x8 register file.

Animation captions:

1. A 4x8 register file has 4 registers, each 8-bits wide.
2. The write port consists of a write address, a write enable control signal, and write data.
3. W_addr determines which register is enabled for load on the rising clock edge. W_en must be set to 1 to load the register.
4. If W_en is set to 0, no register is loaded on the rising clock edge.
5. The read port consists of a read address, a read enable control signal, and read data. RA_addr determines which register is read. RA_en must be set to 1 to read the register.
6. A register file commonly has multiple read ports.
7. Writes are synchronous, but reads asynchronous. Reads starts right away. But a write occurs on a rising clock. Shortly after, the newly-written value appears at the read's output port.
8. A register file is commonly represented as a block symbol.

PARTICIPATION ACTIVITY 6.3.3: Register file.

1) A 32x8 register file consists of ____ registers.

   - 8
   - 32
   - 40

2) The write address, write enable control, and write data are collectively known as what?

   - Write port
   - Read port
   - Write decoder

3) W_en must be set to 1 to write to, or load, a register.

   - True
   - False

4) Given a 32x8 register file, how many bits is W_addr?

- 2
- 3
- 5

5) Assuming a register file with one write port and two read ports, write and read operations can occur simultaneously.

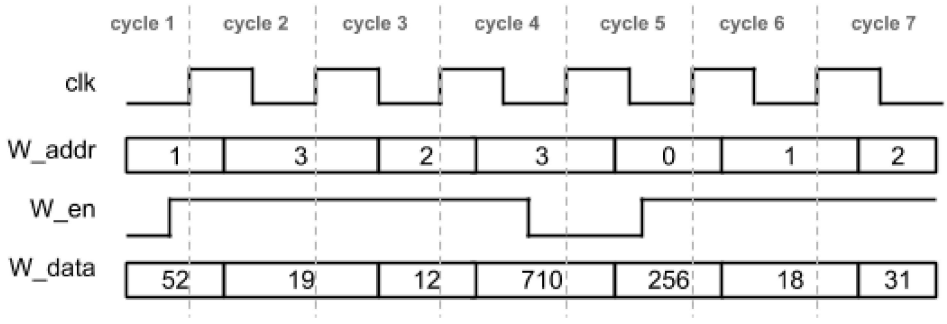
- True
- False

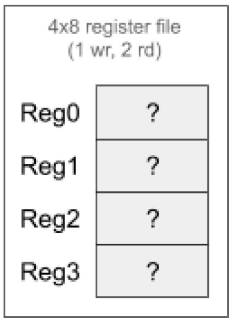
6) Assuming a register file with one write port and two read ports, two read operations can occur simultaneously.

- True
- False

PARTICIPATION ACTIVITY 6.3.4: Register files: Writing.

For the given values of W_addr, W_en, and W_data, indicate the register file's contents in a given clock cycle.





1) Reg1, cycle 2

Check Show answer

2) Reg1, cycle 6

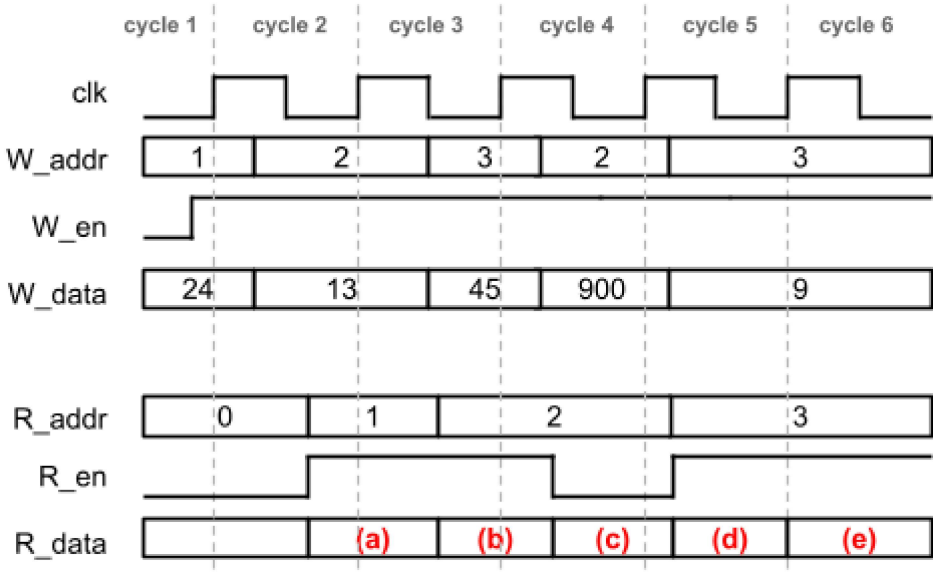
Check Show answer

3) Reg3, cycle 5

Check Show answer

PARTICIPATION ACTIVITY 6.3.5: Register files: Writing and reading.

Assume a 4x32 register file with one read port and one write port. Determine R_data's value at each indicated time. If appropriate, type: Z.



1) (a)

Check Show answer

2) (b)

Check Show answer

3) (c)

Check Show answer

4) (d)

Check Show answer

5) (e)

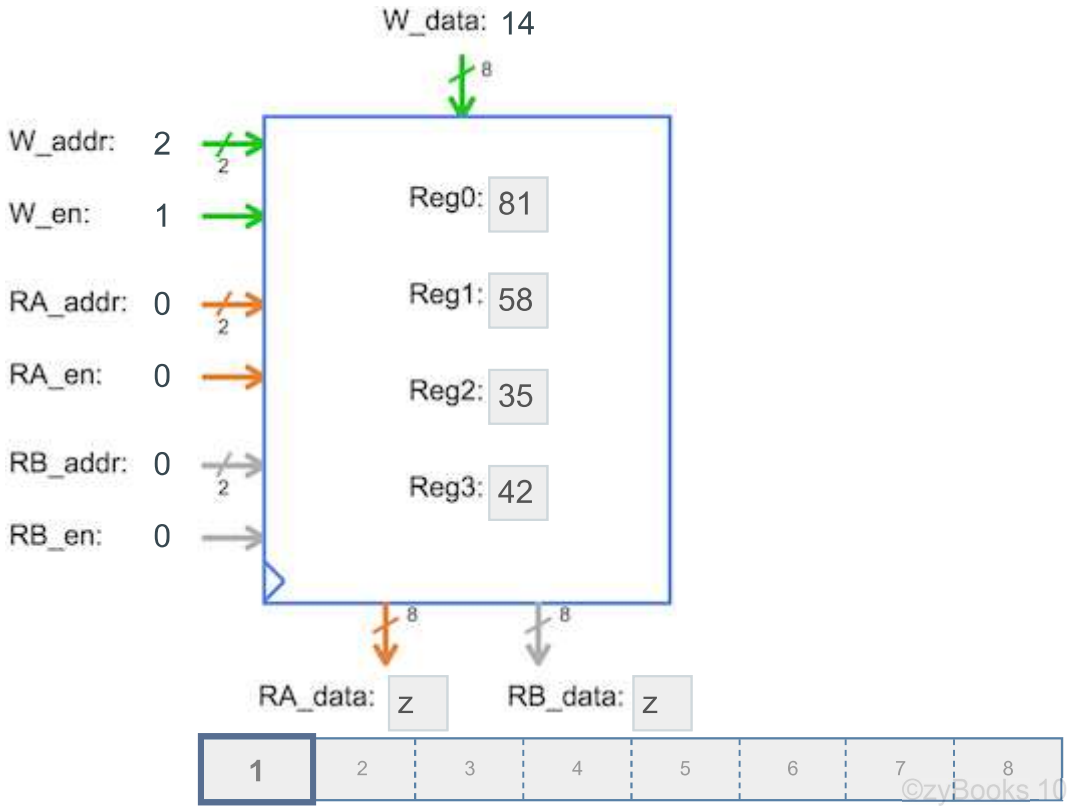
Check Show answer

CHALLENGE ACTIVITY 6.3.1: Register file writing and reading.

347136.1658324.qx3zqy7

Start

Update the values after rising clk.



Check Next

6.4 Example: Above-mirror display using a register file

Routing congestion and fanout

A previous above-mirror display used four 8-bit registers to display four different values. The system can be extended to support displaying 16 32-bit values by using 16 32-bit registers, a 4x16 decoder, and a 32-bit 16x1 mux. However, that design leads to routing congestion and high fanout problems. ***Routing congestion*** occurs when too many wires are present in a small area. And, ***Fanout*** is the number of gates (or transistors) controlled by a single output of a circuit. A high fanout can result in insufficient current to control transistors.

PARTICIPATION ACTIVITY 6.4.1: Above-mirror display using registers to display 16 32-bit values.

Animation content:

undefined

Animation captions:

1. Sixteen 32-bit parallel load registers are connected to a 32-bit 16x1 mux using 16 x 32 = 512 wires. Too many wires in a small area introduces routing congestion.
2. The data input C branches into 16 subwires to connect to the registers, resulting in a fanout of 16. Each branch reduces the current, and a high fanout can yield a current too small to control the transistors.

PARTICIPATION ACTIVITY 6.4.2: Routing congestion and fanout.

1) If the above-mirror display circuit was modified to display 12 values, each with 16-bit data, how many wires will be connected to the mux?

- 12
- 16
- 192

2) Fanout can lead to ____.

- long delays over wires

- fewer wires in a circuit
- stronger currents in the branched wires

Above-mirror display using register file

To avoid routing congestion and fanout, the above mirror display can be implemented using a 16x32 register file.

PARTICIPATION ACTIVITY 6.4.3: Above-mirror display with register file.

Animation content:

undefined

Animation captions:

1. The above mirror display can be designed with a 16x32 register file having one write port and one read port.
2. The car computer's C output connects to the register file's W_data input. WA is the address for the sensor value, which connects to the register file's W_addr input. And load connects to W_en.
3. Since the system always outputs one of the register values to the display, the R_en input is set to 1. The RA input specifies the register file address of the sensor data displayed.

PARTICIPATION ACTIVITY 6.4.4: Above-mirror display using register file.

1) The above-mirror display with 16 32-bit registers uses 529 wires. How many fewer wires are used in the above-mirror display with register file?
   - 74
   - 455
   - 529

2) What value is the R_en input set to?
   - R_en is set to 1 only when the display changes the value to display, and is set to 0 otherwise.
   -

R_en is always 0

R_en input is always 1

6.5 Multi-function registers

A designer commonly wants to ***clear*** a register, meaning to load 0's. Registers commonly come with a control input named "clear" or "clr". A register with clear and load control inputs can be implemented using 4x1 muxes. A register with multiple functions, like clear and load, is called a ***multi-function register***.

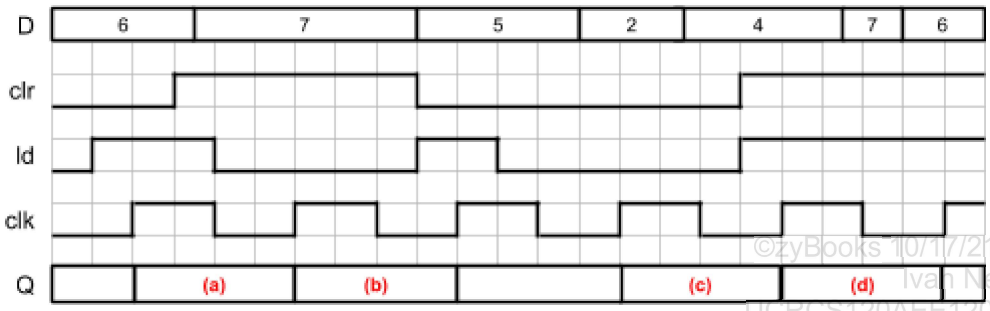
PARTICIPATION ACTIVITY 6.5.1: Multi-function register with load and clear.

Animation captions:

1. 4x1 muxes can be added to create a register with a clear and load control input.
2. When clr = 0 and ld = 0, register maintains present value.
3. When clr = 0 and ld = 1, register loads new input values.
4. When clr = 1 and ld = 0, register loads 0's.
5. If clr = 1 and ld = 1, register designer may decide to just maintain present value.
6. A multi-function register is commonly represented as a block symbol.

PARTICIPATION ACTIVITY 6.5.2: Trace a register's behavior: Load and clear.

For the given values of D, clr, ld, and clk, indicate the register's Q value.



1) (a)

Q =

Check Show answer

2) (b)

Q =

Check Show answer

3) (c)

Q =

Check Show answer

4) (d)

Q =

Check Show answer

A ***shift register*** shifts contents by one position, shifting in a new bit too. Ex: A 4-bit right-shift register holding 1100 becomes 0110 after the shift, assuming the shifted-in bit is 0; the rightmost bit (0) is discarded.

PARTICIPATION ACTIVITY 6.5.3: A register with shift-right and load.

Animation captions:

1. On each rising clock edge, a shift-right register shifts the present value one position to the right. q2 is replaced by shr_in. The rightmost bit is discarded.
2. 4x1 muxes are added to the D flip-flops to create a 3-bit register with shift-right, load, and maintain operations.
3. When shr = 0 and ld = 0, register maintains present value.
4. When shr = 0 and ld = 1, register loads new input values.
5. When shr = 1 and ld = 0, register shifts present value to the right one position.
6. A shift/load register is commonly represented as a block symbol.

PARTICIPATION ACTIVITY 6.5.4: Register with shift right and load.

For the given values of D, shr, shr_in, ld, and clk, indicate the register's Q value.

1) D is 111, shr_in is 0, shr is 0, ld is 0, and Q is 101.
clk rises. What does Q become?

Check Show answer

2) D is 111, shr_in is 0, shr is 0, ld is 1, and Q is 101.
clk rises. What does Q become?

Check Show answer

3) D is 111, shr_in is 0, shr is 1, ld is 0, and Q is 101.
clk rises. What does Q become?

Check Show answer

4) D is 111, shr_in is 1, shr is 1, ld is 0, and Q is 101.
clk rises. What does Q become?

Check Show answer

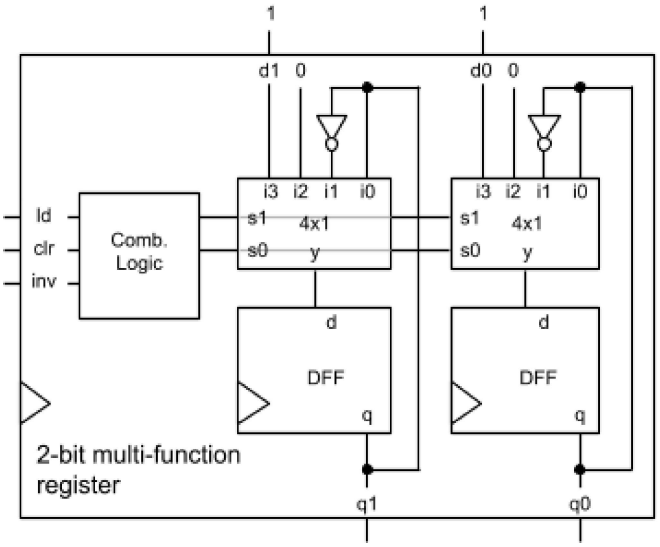
A register can have various combinations of functions, such as load, clear, and increment. Some simple combinational logic can convert the control signals to mux select lines.

PARTICIPATION ACTIVITY 6.5.5: Register with load, clear, and increment.

Animation captions:

1. Register with load, clear, and increment.
2. s1s0 = 11 causes incremented value to be loaded.
3. Some combinational logic is needed to convert the control inputs to the mux select inputs.
4. An equation is derived for each mux select line.

PARTICIPATION ACTIVITY 6.5.6: Multi-function registers: Load, clear, invert.



ld	clr	inv	s1	s0	Register function
0	0	0	0	0	Maintain
0	0	1	(F)		Invert bits
0	1	0	(G)		Clear
0	1	1	0	0	(Maintain)
1	0	0	(H)		Load
1	0	1	0	0	(Maintain)
1	1	0	0	0	(Maintain)
1	1	1	0	0	(Maintain)

1) (F)

- s1s0 = 00
- s1s0 = 01
- s1s0 = 10

2) (G)

- s1s0 = 00
- s1s0 = 10
- s1s0 = 11

3) (H)

- s1s0 = 11
- s1s0 = 10
- s1s0 = 01

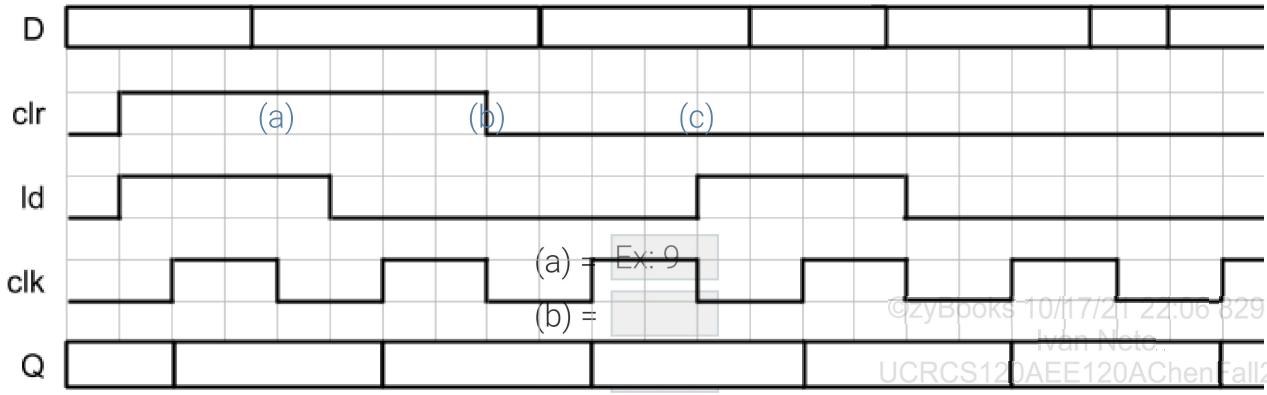
CHALLENGE ACTIVITY 6.5.1: Multi-function registers.

347136.1658324.qx3zqy7

Start

Indicate the 3-bit register's Q value. Assume the register's initial value is 7.

6 5 1 5 5 2 6



1 2 3

Check Next

6.6 ALUs

An ***ALU*** is a datapath component capable of performing various arithmetic and logic functions, like add, subtract, and AND. ALU is short for ***arithmetic-logic unit***. Using an ALU rather than multiple components trades off reduced size for increased delay.

An ALU can be designed using muxes in front of an adder's A, B, and cin inputs to pass certain values to the adder. For example, A - B can be computed by passing A, B', and 1, respectively.

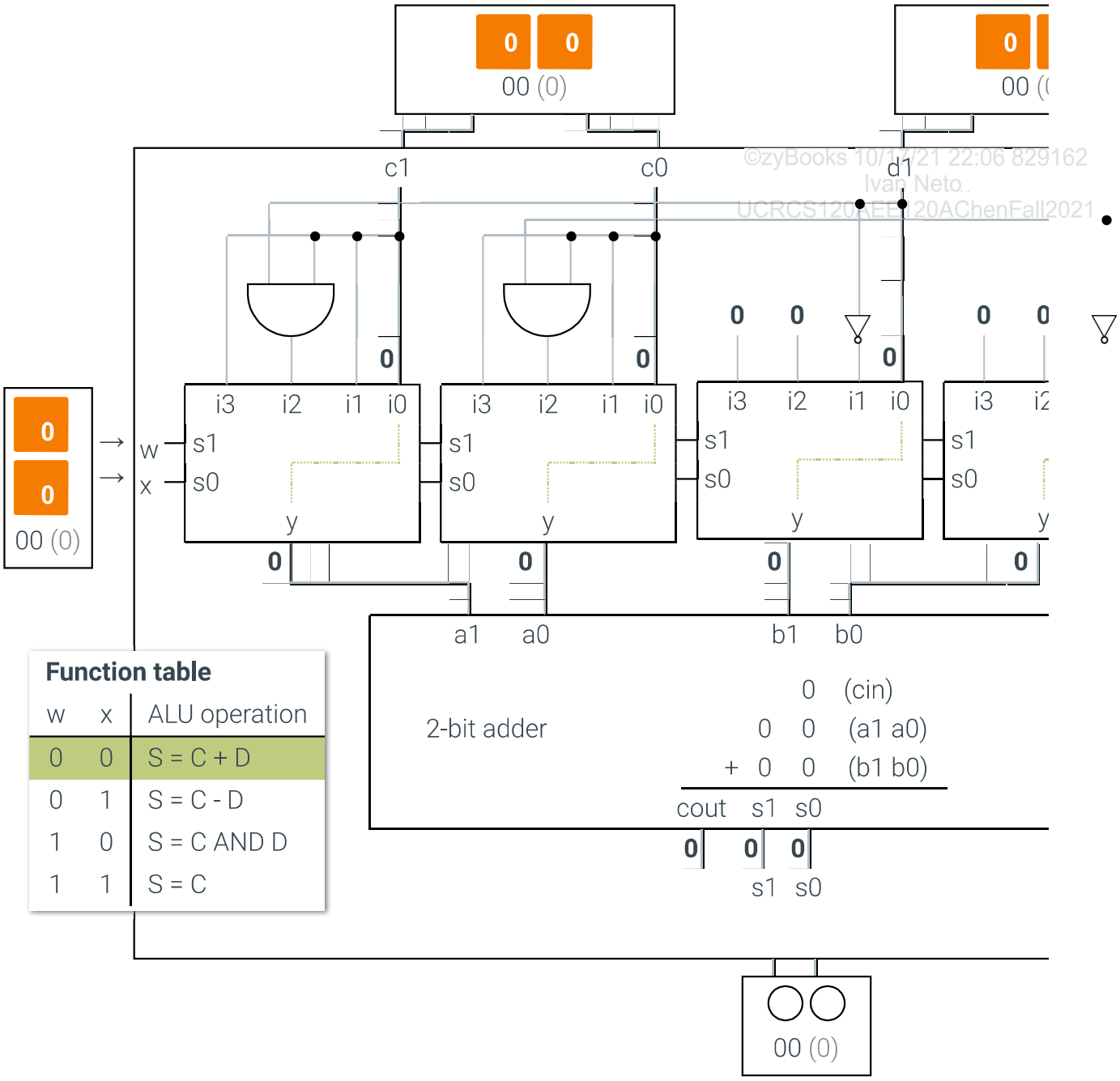
PARTICIPATION ACTIVITY 6.6.1: 2-bit ALU.

Animation captions:

1. An ALU starts with an adder. 4x1 muxes are added to allow four operations.
2. When wx = 00, the ALU adds. A + B + 0 = A + B.
3. When wx = 01, the ALU subtracts. A + B' + 1 = A - B.
4. When wx = 10, the ALU performs a bitwise AND of inputs A and B. (A AND B) + 0 + 0 = A AND B.
5. When wx = 11, the ALU passes A to the output. A + 0 + 0 = A.
6. An ALU is commonly represented as a block symbol.

PARTICIPATION ACTIVITY 6.6.2: ALU Simulator.

Click on the buttons to change the data inputs or control inputs.



Function table

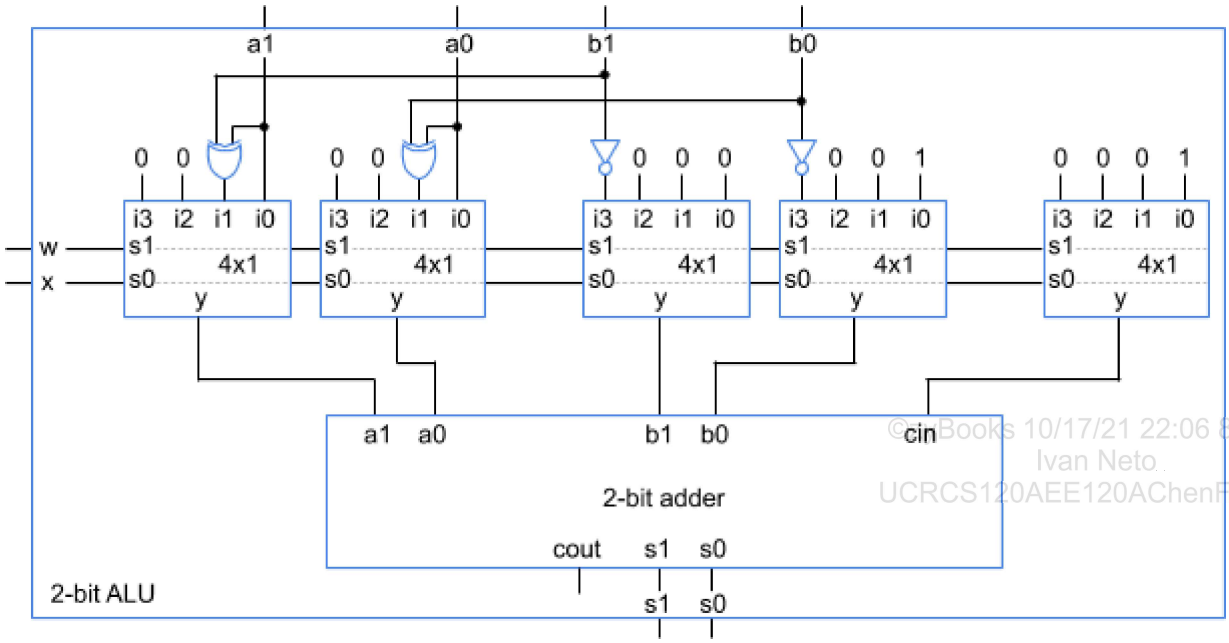
w	x	ALU operation
0	0	S = C + D
0	1	S = C - D
1	0	S = C AND D
1	1	S = C

A more-capable ALU is described below. For each desired ALU operation, the items to be passed through the muxes to the adder's A, B, and cin inputs are also listed.

Table 6.6.1: A more-capable ALU's operation table with the corresponding mux configurations.

Control inputs			ALU operation	Mux configuration		
v	w	x		A	B	cin
0	0	0	S = A + B	A	B	0
0	0	1	S = A - B	A	B'	1
0	1	0	S = A + 1	A	0	1
0	1	1	S = 0	0	0	0
1	0	0	S = A AND B	AB	0	0
1	0	1	S = A OR B	A OR B	0	0
1	1	0	S = A XOR B	A XOR B	0	0
1	1	1	S = NOT A	A'	0	0

PARTICIPATION ACTIVITY 6.6.3: ALU operations.



Refer to the ALU implementation above.
For each of the wx values provided, determine the corresponding ALU operation.

wx = 10 | wx = 00 | wx = 11 | wx = 01

	S = 0
	S = B'
	S = A + 2
	S = A XOR B

Reset

PARTICIPATION ACTIVITY 6.6.4: ALU mux connections.

Assume an 8-bit ALU. Determine the mux configuration needed to implement the given ALU operation.

1) S = E + F
   - A = E, B = F, cin = 0
   - A = E, B = F, cin = 1

2) S = E - F
   - A = E, B = F, cin = 1
   - A = E', B = F, cin = 1
   - A = E, B = F', cin = 1

3) S = E AND F
   - A = E AND F, B = 1, cin = 1
   - A = E AND F, B = 0, cin = 1
   - A = E AND F, B = 0, cin = 0

4) S = 1
   - A = 1, B = 0, cin = 0
   - A = 1, B = 1, cin = 1

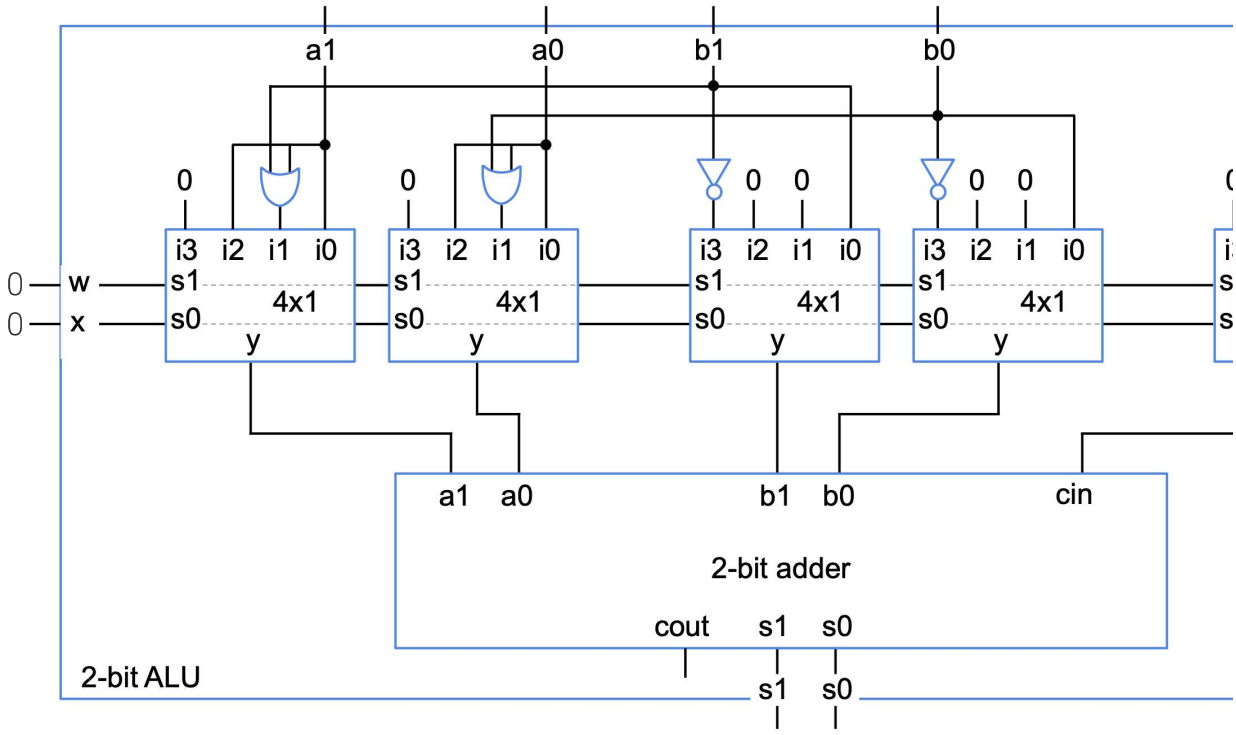
5) S = 2E (multiply 2 · E)
   - A = E, B = E, cin = 0

A = 2, B = E, cin = 0

CHALLENGE ACTIVITY 6.6.1: ALUs.

347136.1658324.qx3zqy7

Start



S = Ex: A + B

1 2

Check Next

6.7 SRAM and DRAM

Memory basics

An NxM ***memory*** is a digital component that retains bit values, consisting of N words of M bits each. Each word has a unique address. Ex: A 4096x8 memory has 4096 8-bit words (for a total of 32,768 bits), with word addresses from 0 to 4095. Nearly every computing device requires memory.

PARTICIPATION ACTIVITY 6.7.1: Memory basics.

Animation captions:

1. An NxM memory has N words, each M bits wide (4096x8 here). Word addresses are 0, 1, 2, ..., 4094, and 4095. Input addr requires 12 bits to represent 0-4095.
2. To write, wen (write enable) is set with 1, and the target word's address is provided (2). On the next rising clock, the data (15) on wdata (write data) is stored into that word.
3. Other words can be written similarly. In this case, word 4094 is written with 33. Note that 15 remains stored in word 2.
4. Setting ren (read enable) with 1 causes the addressed word to appear on rdata (read data), typically asynchronously.

Such memory is often called ***random access memory*** (or ***RAM***) because any "random" word can be quickly accessed, in contrast to older sequentially-accessed memory technologies like tape that had to first be spun or moved to access a particular word.

Most RAM is ***volatile memory***, meaning bit values are lost if electrical power is removed.

Note: For a processor, "word" may refer to 4 bytes. But for a memory, word means one address location, however wide.

PARTICIPATION ACTIVITY 6.7.2: Memory basics.

Consider the memory above. Question actions follow the previous question's actions. All words initially contain 0. Answer in decimal, not binary.

1) On rising clock 1, addr is 9, wen is 1, and wdata is 44.
On rising clock 2, addr is 5, wen is 1, and wdata is 77.
What value is in word 9?

Check Show answer

2) On rising clock 3, addr is 15, wen is 0, and wdata is 305.
What value is now in word 15?

Check Show answer

3) addr is 5, ren is 1.
What soon appears on rdata?

Check Show answer

4) How many bits does word 3 contain?

Check Show answer

5) How many total bits does the memory store?

Check Show answer

6) Based on the general memory design described above, is simultaneously setting both ren and wen with 1 reasonable? Type yes or no.

Check Show answer

SRAM and DRAM

Memory comes in two common forms:

- A ***static RAM*** (or ***SRAM***) typically uses 6 transistors to store each bit value, by passing the bit into a loop within those transistors.
- A ***dynamic RAM*** (or ***DRAM***) typically uses 1 transistor and 1 capacitor to store each bit value, by charging the capacitor.

SRAM is faster than DRAM, but DRAM is denser and cheaper. SRAM is thus typically used by processors for small fast on-chip cache, while DRAM is used for the larger main memory off-chip.

Also, processors and SRAM are made using different chip design processes than DRAM, so putting DRAM on-chip with a processor is rare.

PARTICIPATION ACTIVITY 6.7.3: SRAM vs. DRAM.

Animation captions:

1. SRAM read/writes are about 10x faster than for DRAM.
2. But DRAM's are about 5x denser (using only 1 transistor/cell). Because of the density and mass production, DRAM's are about 100x cheaper too.
3. Thus, SRAM is used primarily for a processor's small but fast cache or other small on-chip memory, while DRAM is used for a processor's large main memory off-chip.

PARTICIPATION ACTIVITY 6.7.4: SRAM vs. DRAM.

1) Which is faster?
   - SRAM
   - DRAM

2) For a fixed size, which can store more bits?
   - SRAM
   - DRAM

3) Consider a processor that accesses a memory once per instruction. Suppose each instruction's time is 0.5 ns to access SRAM and 0.5 ns to run the instruction, so a 1 billion instruction program takes 1 second to execute. How many seconds would the program take using DRAM instead?
   - 1.2 sec
   - 5.5 sec

4) Consider a processor that requires 1 GB of DRAM, where the DRAM costs $50. About how much would the memory cost if all the DRAM was instead SRAM?

- $500
- $5000

Memory size

A memory's size may be specified in various ways.

- *4096x32*: Indicates the number of words, and the bits per word.
- *131,072 bits*: Indicates the total number of bits.
- *16,384 bytes*: Indicates the total number of bytes (a byte is 8 bits).
- *16 KBytes (or 16 KB)*: Approximate number of bytes. The K is the metric kilo, for 1,000. Note: This method is common *but inaccurate*, because 16 Kbytes means 16,000 bytes rather than the actual 16,384 bytes.
- *128 Kbits (or 128 Kb)*: Indicates the total number of bits. Again, this method is common *but inaccurate*.

Memory sizes are commonly measured in MB (megabytes, or 1 million bytes)), GB (gigabytes, or 1 billion bytes), or TB (terabytes, or 1 trillion bytes). Ex: A 16 GB memory. The uppercase B means bytes (like GB), while lowercase b means bits (like Gb).

Memory sizes are powers of 2, so metric prefixes like kilo, mega, and giga, which are powers of 10, are inaccurate. Alternative prefixes, known as ***IEC prefixes***, exist like kibi (2^{10} or 1024), mebi (2^{20} or 1,048,576), gibi (2^{30} or 1,073,741,824), and tebi (2^{40} or 1,099,511,627,776). In kibi, the ki refers to the metric prefix kilo, and the bi to "binary". A kibi is abbreviated Ki, as in 1 KiB for 1 kibibyte. Likewise for other IEC prefixes.

When metric prefixes are used, those prefixes are known to actually refer to the nearest power of 2, so a kilobyte is known to mean 1024 bytes (a kibibyte) and not 1000 bytes.

PARTICIPATION ACTIVITY 6.7.5: Memory sizes.

1) A 512x8 memory has how many *words*?

Check Show answer

2) A 64x4 memory has how many *bits*?

Check Show answer

3) In 1 KB, does the B mean bytes or bits?

Check Show answer

4) For memory, 1K actually refers to what value? Just type a number.

Check Show answer

5) What IEC prefix refers to the power of 2 nearest to 1 million?

Check Show answer

6) How many bits are in a 1 KB memory? Just type a number.

Check Show answer

7) How many bits are in a 2 Kb memory? Just type a number.

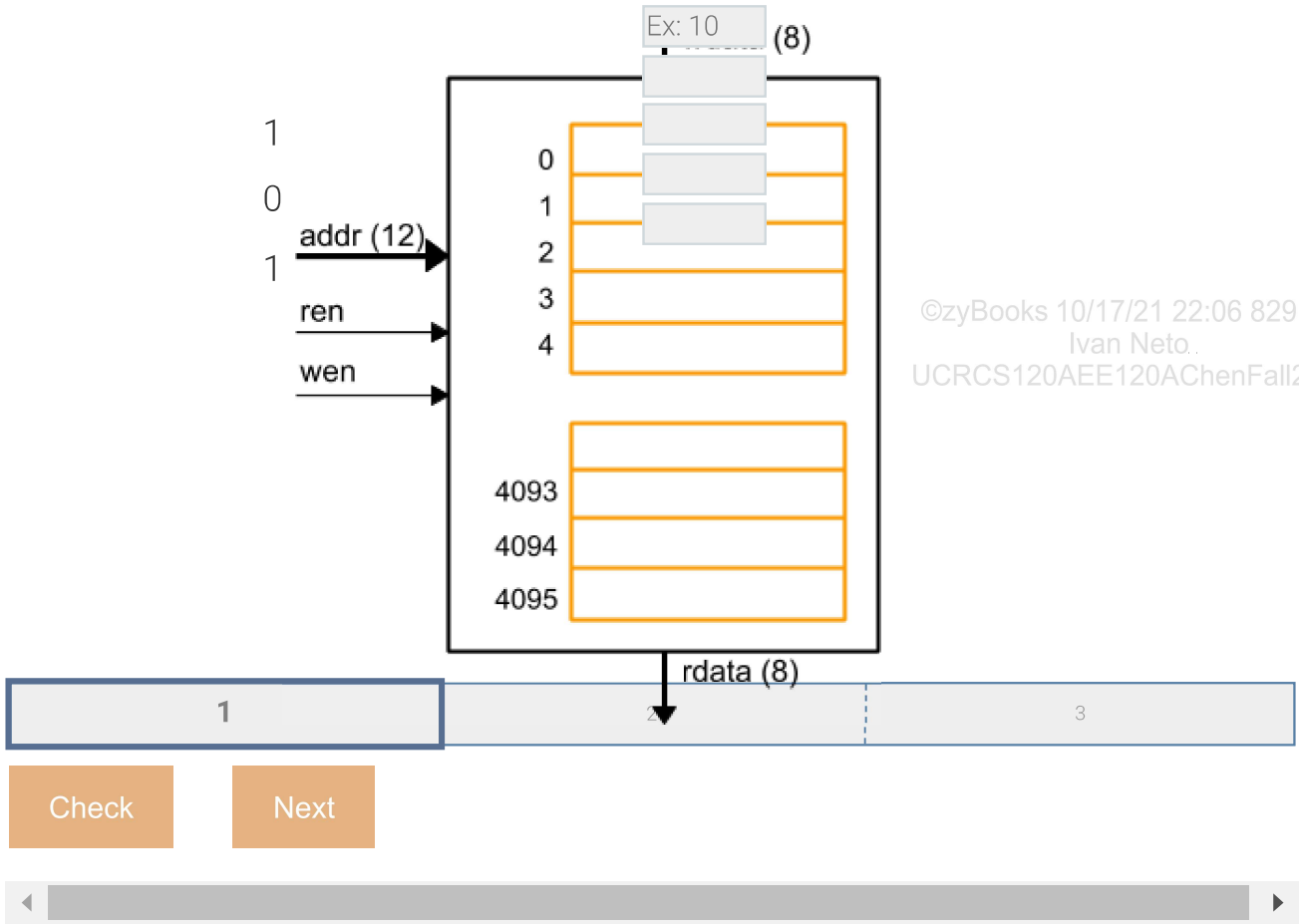
Check Show answer

CHALLENGE ACTIVITY 6.7.1: SRAM and DRAM.

347136.1658324.qx3zqy7

Start

Assume all words initially contain 0. Update the values after a rising clock for addresses 0-4.



Exploring further:

- SRAM (Wikipedia)
- DRAM (Wikipedia)
- SRAM vs. DRAM (diffen.com)
- IEC prefixes

6.8 RAM design

SRAM

An NxM SRAM consists of N locations each M bits wide. An SRAM could be built from N M-bit registers, with a decoder connecting to each register's load input for writes, and the decoder controlling a large mux for reads, but such a design has excessive wiring and mux logic for large RAMs. Instead, each SRAM location is typically built from an array of cells, with each cell passing a word enable line and data line through.

PARTICIPATION ACTIVITY 6.8.1: SRAM design: Array of cells, and decoder.

Animation captions:

1. Conceptually, SRAM is an array of cells that each stores one bit. A decoder sets one word's enable (we) with 1 when write enable (wen) is 1. (Read enable not shown)
2. This address causes decoder output 4095 to be 1, causing the write data 10110001 to be stored in word 4095.

A common ***SRAM cell*** design uses two inverters to store a bit, and two transistors to let a 1 or 0 into that loop. Because each inverter is two transistors, each cell has six transistors.

PARTICIPATION ACTIVITY 6.8.2: SRAM 6-transistor cell basics.

Animation captions:

1. The basic idea of an SRAM cell is to store a bit in a two-inverter loop. The 1 becomes a 0, which becomes a 1 again, and the process repeats.
2. The question is how to get a bit into that loop. A transistor can allow a new bit into the loop. When word enable is 1, wdata's value enters the loop.
3. A detail: The previous value on the right may interfere with the new value. Another transistor is added, with wdata' (wdata's complement).
4. The word enable is fed through for the cell to the right. The data lines are fed through for the cell below.

Reads use an electrical technique beyond this section's scope. In short, when read-enable (ren) is 1, both data lines are set with 1. we is also set with 1, so the transistors on both sides conduct, causing either the left or right data line's voltage to be pulled down slightly. A special "sense amplifier" circuit detects which side was slightly pulled down, to determine whether a 0 or 1 was stored.

PARTICIPATION ACTIVITY 6.8.3: SRAM design.

Consider the above SRAM design.

1) How many cells exist in one word?

Check Show answer

2) How many total cells exist?

Check Show answer

3) For one cell, how many data lines enter from the top?

Check Show answer

4) In the animation, for one cell, a word enable (we) line enters from the left, and exits from the _____.

Check Show answer

5) For a cell, we is 1 and wdata is 1, causing 1 to pass through the left transistor and enter the left side of the inverter loop. What value will appear on the right side of the inverter loop?

Check Show answer

6) we is 1, wdata is 1, and 1 is on the left side of the inverter loop, and 0 on the right. Then we and wdata are set with 0. What value is on the left side of the inverter loop?

Check Show answer

7) A cell has 0 on the left and 1 on the right of the inverter loop, with we = 0. What value is presently stored in the cell?

Check Show answer

Check Show answer

DRAM

A ***DRAM cell*** stores a bit as a charge on a capacitor, with one transistor that can pass current to the capacitor. With only 1 transistor and 1 capacitor, a DRAM cell is much smaller than an SRAM cell. But the capacitor in a DRAM cell leaks, requiring repeated reads and writes to ***refresh*** the cell, which is one reason DRAM access is slower than SRAM.

PARTICIPATION ACTIVITY 6.8.4: DRAM 1-transistor 1-capacitor cell basics.

Animation captions:

1. The basic idea of a DRAM is to store a 1 or 0 as a charge on a capacitor. Writing a 1 charges the capacitor's top plate, causing positive voltage there.
2. But capacitors leak. So the charge slowly leaks to ground. Thus, each DRAM cell must be "refreshed" (read, and written right back), typically within 60 ms.

PARTICIPATION ACTIVITY 6.8.5: DRAM cells.

1) How many transistors are in a DRAM cell?
   - ○ 1
   - ○ 2

2) Where is a 1 stored in a DRAM cell?
   - ○ Transistor
   - ○ Capacitor

3) To prevent a stored bit from leaking completely away, a DRAM repeatedly reads and then writes each cell, called a _____.
   - ○ replenish
   - ○ refresh

4) A DRAM is non-volatile.
   - ○ True
   - ○ False

5) A DRAM requires a controller to handle refreshes.

- True
- False

Variations of SRAM and DRAM cells exist, such as SRAM cells with 4 transistors and two resistors, but the above are the most common forms.

Exploring further:

- SRAM (Wikipedia)
- DRAM (Wikipedia)

6.9 ROM design

ROM using floating-gate transistors

Flip-flops, SRAMs, and DRAMs lose stored bits when electrical power is removed (*volatile* memory). In contrast, ***ROM*** (read-only memory) retains stored bits when power is removed (*non-volatile* memory), by using technology that usually has slow writes, so writes are less frequent than reads; hence the term "read-only" (a misnomer when writing is in fact possible). Writing to a ROM is called ***programming*** the ROM.

Bit storage in a ROM commonly uses a ***floating-gate transistor*** having a special region where electrons can be trapped, staying there even without power. Applying a large positive voltage traps the electrons (programming). A large negative voltage frees the electrons (erasing). Applying such large voltages long enough to trap or free electrons is slow, which is why writing a ROM is slow.

PARTICIPATION ACTIVITY 6.9.1: ROM using floating-gate transistors.

Animation captions:

1. A normal transistor does not conduct when the gate input is 0, and conducts when the gate is 1.
2. A floating-gate transistor has a special region called a floating-gate. The transistor can work like a normal transistor.

3. A large positive voltage causes electrons to tunnel into the special region. When that voltage is removed, the electrons are trapped. The transistor has been "programmed".
4. Now, the transistor won't conduct even with a 1 at the gate. The negative electrons "block" the positive 1 from switching the transistor to conduct.
5. In a ROM, for an address, one word line will be 1. The data lines become 0 for unprogrammed transistors, and 1 for programmed transistors (uses special circuitry).

Note: The above is a "logical" view of a ROM, oversimplifying electrical features beyond this material's scope.

PARTICIPATION ACTIVITY 6.9.2: ROM using floating-gate transistors.

1) An unprogrammed floating-gate transistor (no electrons trapped in the floating gate) _____.
   - acts as a normal transistor
   - acts opposite a normal transistor.
   - won't conduct

2) A programmed floating-gate transistor (electrons trapped in the floating gate) _____.
   - acts as a normal transistor
   - acts opposite a normal transistor.
   - won't conduct

3) Programming a floating-gate transistor is done via a large _____.
   - positive voltage
   - negative voltage
   - hammer

4) Erasing a floating-gate transistor is done via a large _____.
   - positive voltage
   - negative voltage

5) A 256x16 ROM would have how many word lines?

- 8
- 16
- 256

6) A 256x16 ROM would have how many floating-gate transistors?

- 16
- 256
- 4096

7) A ROM has 8 bits per location. Location 99 has only the first two transistors programmed (electrons trapped). What value is stored at location 99?

- 11000000
- 00111111
- 11111111

8) The ROM shown above would use _____ to achieve the 1's and 0's at the data output.

- normal logic
- special circuitry

ROM types

Numerous ROM types exist.

- ***Mask-programmed ROM***: The word line to bit line connections are hardwired during chip manufacturing, and can never be changed.
- ***OTP ROM*** (one-time programmable ROM): The word line to bit line connections have a fuse that can be "blown" to break the connection. A user can program the device only once.
- ***EPROM*** (erasable programmable ROM): Programming uses large positive voltage to trap electrons in a floating-gate transistor, but erasing is done by placing the chip under ultraviolet light to provide the energy to free the trapped electrons. This ROM type preceded the more convenient EEPROM.
- ***EEPROM*** (electrically-erasable programmable ROM): Programming uses large positive voltage to trap electrons in a floating-gate transistor, and erasing uses a large negative voltage.
- ***Flash***: Programming uses large positive voltage to trap electrons in a floating-gate transistor, and erasing uses a large negative voltage to quickly erase entire blocks of locations at one time, like a "flash".

PARTICIPATION ACTIVITY 6.9.3: ROM types.

Animation captions:

1. A mask-programmed ROM has connections for 1's and 0's made during manufacturing, which involves passing light through "masks" to create wires on a chip.
2. A one-time programmable (OTP) ROM has fuses. Blowing some fuses but leaving others intact programs the 1's and 0's. Once blown, fuses can't be reconnected (hence "one-time").
3. EPROM uses a large positive voltage to trap electrons in a floating-gate transistor (FGT above). Erasing is done by placing chip package (having a window) under UV light.
4. EEPROM is like EPROM, but conveniently erased using a large negative voltage (no need to place chip under UV light).
5. Flash is like EEPROM, except entire blocks (thousands of words or more) can be quickly erased at once (in a "flash"), rather than one word at a time like EEPROM.

PARTICIPATION ACTIVITY 6.9.4: ROM types.

Match the ROM type with the most likely usage scenario.

EEPROM **Mask-programmed ROM** **OTP ROM** **EPROM** **Flash**

	Storing program instructions in a calculator that will be produced in the hundreds of millions.
	Storing a unique ID number for a secure card key, which should never be changed.
	Storing a program in a microprocessor chip being used for prototyping in an engineering lab, with the chip being reprogrammed a few times a day, in the 1980s.
	Storing 256 phone numbers in a portable keychain device, each number reprogrammable by some button clicks.

A digital photo frame to which 8 photos can be uploaded, with the frame showing a different photo every minute.

Reset

Exploring further:

- ROM (Wikipedia)
- Flash (Wikipedia)

6.10 Queues (FIFOs)

Queue basics

A designer's data storage needs may require reading items in the same order that the data was written in. A ***queue*** is a list that is written to at the end of the list, and read from the front of the list, with a read also removing the item from the list. A queue is known as ***first-in first-out*** or ***FIFO*** as the first item being written into the list will be the first item read from the list.

Queues are commonplace in digital systems. Examples include:

- A computer keyboard writes the pressed keys to a queue and requests the computer to read the queued keys.
- A digital video camera writes recently captured video frames into a queue, and concurrently reads the frames from the queue before compressing and storing the frames on a storage device.
- A computer printer may store print jobs in a queue while the jobs are waiting to be printed.
- A computer network router receives data packets from an input port and writes the packets into a queue which the router reads.

PARTICIPATION ACTIVITY 6.10.1: Writing and reading a queue implemented using a register file.

Animation content:

undefined

Animation captions:

1. A queue can be implemented using a register file or RAM.
2. An 8-word queue implemented using a register file has a front and end indices that are both initially 0 (empty queue). Item A is written to the end (address 0), and end is incremented to address 1.
3. Next, item B is written to the end (address 1), and end is incremented to address 2. The next item (C) is written to address 2, and end is incremented to address 3.
4. When a read is performed, the item at the address of front is read, which is A. Once item A is read, front is incremented.

PARTICIPATION ACTIVITY 6.10.2: Queues (FIFO).

1) Given an initially empty 8-word queue, what is the address of the end index after the following operations?

Write 15
Write 18
Write 10

- 0
- 3
- 4

2) Given an initially empty 8-word queue, what is the address of the front index after the following operations?

Write 25
Write 28
Write 20
Read
Read

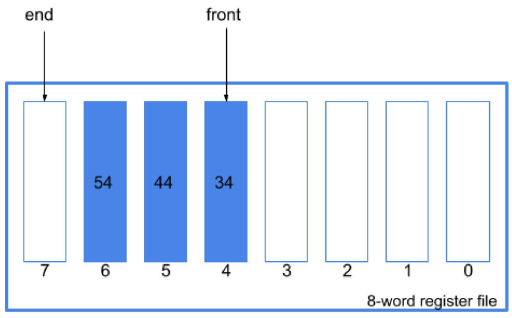
- 0
- 1
- 2

3) Given an initially empty 8-word queue, what is the address of the front index after the following operations?

Write 35
Read
Write 30
Write 38

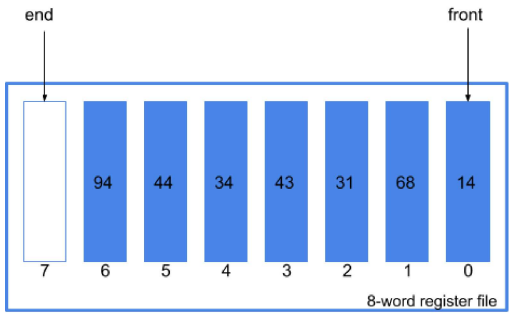
- 0
- 1
- 2

4) For the queue below, what is the address of the end index after the operation: Write 94?



- 0
- 7
- The operation leads to an error. The queue cannot go past the highest address.

5) For the queue below, what is the address of the end index after the operation: Write 55?



- 0
- 7
- The operation leads to an error because the front index is 0.

6) If front = end, the queue must be _____.

- empty
- full
- either full or empty.

Queue datapath with a register file

A queue can be implemented with a register file and additional logic. The queue's controller can be built with an FSM. In many uses of a queue, the circuit writing the queue operates independently from the circuit reading the queue. The controller must be designed to detect when a circuit that uses a queue tries to read an empty queue or write to a full queue. If undetected, the queue may be put into a misleading internal state like the front and end indices getting crossed over. A queue often has outputs that indicate the state of the queue. Ex: A 1-bit full output that is 1 if the queue is full, and 0 otherwise.

PARTICIPATION ACTIVITY 6.10.3: Architecture of an 8-word 16-bit queue.

Animation content:

undefined

Animation captions:

1. An 8-word queue can be implemented using an 8-word two-port register file and additional combinational logic.
2. Two 3-bit up-counters maintain the front and end indices. An up-counter ensures that the counters wrap around from 7 to 0. An equality comparator detects if front = end.
3. On a write operation, the controller sets the register file's wr input to. The end up-counter output connects to waddr. Then, end up-counter's inc input is set to 1 to increment the end index.
4. On a read operation, the controller sets the register file's rd input to 1. The front up-counter output connects to raddr. Then, the front up-counter's inc input is set to 1 to increment the front index.
5. When front = end, the preceding action determines if the queue is full or empty. If the action was a write, the controller sets full =1, and if the action was a read the controller sets empty = 1.

PARTICIPATION ACTIVITY 6.10.4: Queue architecture.

1) What is the minimum size up-counters

needed for a 16-word queue?

- 4-bit
- 16-bit

2) In the above example, if front = end and the preceding action was a write, the queue is _____.

- full
- empty

3) In the above example, if front = end and the preceding action was a read, the queue is _____.

- full
- empty

CHALLENGE ACTIVITY 6.10.1: Queues.

347136.1658324.qx3zqy7

Start

Given an initially empty 8-word queue, what is the address of the front index after the following operations?

Write 62
Write 78
Write 89
Read
Read
Read

Ex: 9

1 2 3 4

Check Next

6.11 Chip economics

High NRE cost encourages mass-production

A ***chip*** (aka ***integrated circuit*** or ***IC***) is a digital circuit manufactured on a fingernail-sized piece of silicon, typically placed inside a black or silver insulating package.

Non-recurring engineering (or ***NRE***) cost is the cost to design and set up a computer chip for manufacturing. Due to the complexity of modern chips having billions of transistors, NRE costs may be tens or hundreds of millions of dollars. That cost adds to a chip's cost, depending on the number of chips made. Ex: NRE cost for a chip may be $10,000,000. If 10,000 chips are made, $10,000,000/10,000 = $1000 needs to be added per chip to cover NRE cost. But if 1,000,000 chips are made, only $10,000,000/1,000,000 = $10 need be added. Thus, mass-producing a chip allows for lower chip costs, since NRE cost can be spread.

PARTICIPATION ACTIVITY 6.11.1: High NRE cost favors mass-producing chips.

Animation captions:

1. The non-recurring engineering (NRE) costs for a chip's circuits may be 10 million dollars. Like the engineering costs for a mass-produced car, those costs only occur once.
2. That design is used to create identical chips. If only 10,000 chips are made and sold, then $1000 must be added to each chip to cover the $10M NRE costs.
3. But if 1,000,000 chips are sold, only $10 need be added to cover the $10M NRE. Hence, chip makers strive to design fewer chips and then mass produce those chips.

PARTICIPATION ACTIVITY 6.11.2: NRE and mass-produced chips.

Assume NRE cost is evenly distributed among chips sold.

1) NRE cost for a given chip is $10 million. If 1 million chips are sold, how much is added per chip to cover NRE cost?

Check Show answer

2) NRE cost for a given chip is $10 million. If 10 million chips are sold, how much is added per chip to cover NRE cost?

Check Show answer

3) A chip maker currently sells a 128 MB chip for $10. A customer wants 10,000 256 MB chips for $50 per chip. The NRE cost for designing a 256 MB chip will be $10 million. Should the chip maker design a new 256 MB chip for the customer? Type yes or no.

Check Show answer

Low yield of large chips encourages smaller chips

The chip manufacturing process simultaneously creates multiple identical chips on a silicon wafer (a round silicon slice), and then each chip is cut out. The manufacturing process may take hours or days and is costly. Unfortunately, a wafer may have defects, like a broken wire, that render some chips unusable. ***Yield*** is the percentage of chips that are usable and free from significant defects. Ex: If 40 of 50 chips on a wafer are usable, yield is 40/50 = 80%. Larger chips are more likely to enclose a defect and thus have lower yield.

PARTICIPATION ACTIVITY 6.11.3: Smaller chips are less likely to enclose a defect, giving higher yields, and ultimately lower cost.

Animation captions:

1. Chips are manufactured as one large silicon wafer and then cut out, like baking one large cookie that is then cut into smaller cookies.
2. Customers may want large chips, like a 4 GB memory.
3. Unfortunately, defects (shown as X) occur on wafers, so only some chips are usable. The wafer manufacturing cost is distributed across the usable chips ($30 for this 4 GB chip)
4. With smaller chips, a defect impacts fewer chips, giving a higher "yield" of usable chips, which may lead to lower overall cost ($5 for these 1 GB chips, so only $20 for 4 GB).

For learning purposes, this section shows only a few chips per wafer. However, a real wafer may fit hundreds or even thousands of chips.

PARTICIPATION ACTIVITY 6.11.4: Yield.

Assume wafer manufacturing cost is evenly distributed among usable chips. Ex: If the wafer manufacturing cost is $50 and a wafer has 10 usable chips, the cost is $50 / 10 = $5 per chip.

1) A wafer holds 50 chips and costs $100. What is the cost per chip, assuming all chips are usable?

Check Show answer

2) A wafer holds 50 chips. 10 defects appear on the wafer, making 5 chips unusable. How many usable chips result?

Check Show answer

3) A wafer holds 50 chips. 20 defects cause 15 to be unusable. What is the yield? Answer as: 50%

Check Show answer

4) An $80 wafer holds 50 small chips. 10 defects cause only 40 chips to

be usable (80% yield). What is the wafer cost per usable chip?

Check Show answer

5) An $80 wafer holds 20 large chips. 10 defects cause only 10 chips to be usable (50% yield). What is the wafer cost per usable chip?

Check Show answer

6) If a wafer just holds one massive chip, and any defect renders a chip unusable, how many defects can be tolerated?

Check Show answer

Example 6.11.1: Memory card composed of multiple memory chips.

If one looks inside a computer, one is likely to find memory chips arranged in an array to form a larger memory (described in another section). Below is a memory card from a personal computer, with 16 4 Gb chips (8 on each side of the card). Those 16 chips form a 16*4 = 64 Gb memory, meaning an 8 GB memory. Building a larger memory by composing smaller memory chips is more economical, as described above.

CHALLENGE ACTIVITY 6.11.1: Chip economics.

347136.1658324.qx3zqy7

Start

NRE cost for a given chip is $800 million.

If 200 million chips are sold, how much is added per chip to cover NRE cost?

$ Ex: 1 Assume NRE cost is evenly distributed among chips sold.

1 2 3

Check Next

6.12 Composing memory

Composing memory chips horizontally

Because small memory chips may be cheap and widely available due to chip economics, building a memory by composing smaller memory chips is often cheaper than buying a single larger chip.

A designer may wish to create a wider memory, in which case memory chips can be composed horizontally, as below.

PARTICIPATION ACTIVITY 6.12.1: Composing two memory chips horizontally into a memory with twice as many bits per location.

Animation captions:

1. A designer may want a memory with 1 G locations each 16 bits wide, but only have 1G x 8 (1 GB) memory chips available.
2. The designer can use two 1 GB chips, splitting the 16-bit data input and output into the left 8-bits and the right 8-bits.
3. The address and control lines are simply connected to both memory chips. The two 8-bit-wide memories act like one 16-bit memory.

PARTICIPATION ACTIVITY 6.12.2: Composing memory chips horizontally.

1) A designer wishes to build a 32-bit wide memory using 1G x 8 memory chips. How many memory chips are needed?
   - ○ 4
   - ○ 8

2) A memory with 1 G addresses has 30 address inputs. If two 1G x 8 memory chips are composed horizontally to build a 16-bit-wide memory, how many address lines connect to the left chip?
   - ○ 29
   - ○ 30

3) A designer wishes to build a 10-bit wide memory using any number of 8-bit wide memory chips. How many chips are required?
   - ○ 2

Not possible

Composing memory chips vertically

A designer may wish to create a memory with more locations, in which case memory chips can be composed vertically, as below.

For this purpose, when read is not enabled, a memory chip outputs neither 0's nor 1's, but electrically outputs nothing, allowing different rows' read data output lines to simply be connected when creating a larger memory.

PARTICIPATION ACTIVITY 6.12.3: Composing two memory chips vertically into a memory with twice as many locations.

Animation captions:

1. A designer can compose two 1G x 8 memory chips vertically to form a 2G x 8 memory. The 8 wdata inputs are connected to both chips. The 8 rdata outputs also connect.
2. A 2 GB memory has 31 addresses, but each 1 GB memory has just 30. Bit 30 (the 31st bit) controls a 1x2 decoder that activates either the upper or lower chip.
3. If an address is less than 1G (starts with 0), the top memory should be activated.
4. If an address is greater than 1G (starts with 1), the bottom memory should be activated.
5. Each ren and wen enable input is thus ANDed with one of the decoder outputs, so that the one appropriate memory chip will be active based on the address. A 2 GB memory is achieved.
6. An even larger memory uses more of the leftmost address bits and a larger decoder. Ex: A 4 GB memory has 32 address bits. Bits 31 and 30 control a 2x4 decoder that activates one of four 1 GB chips.

PARTICIPATION ACTIVITY 6.12.4: Composing memory chips vertically.

1) Two 1 GB memory chips are composed into a 2 GB memory. What size decoder is needed? Type 1x2, 2x4, or 3x8.

Check Show answer

2) Two 1 GB memory chips are composed into a 2 GB memory. Which address bit controls the decoder? Type 33, 32, 31, or 30.

Check Show answer

3) 256-byte memory chips (8 address inputs) are composed into a 1 KByte memory (10 address inputs). What size decoder is used? Type 1x2, 2x4, 3x8, or 4x16.

Check Show answer

4) 256-byte memory chips (8 address inputs) are composed into a 1 KByte memory (10 address inputs). What address bits control the decoder? Type either:
10, 9
9, 8
9, 8, 7
9

Check Show answer

5) Eight 1 KB memory chips (10 address inputs) named C0, C1, ..., C7 (top to bottom) are composed into an 8 KB memory (13 address inputs). Which chip is active for address 1110110000000?

Check Show answer

6) A designer has 1 KB (1K x 8) memory chips available. The designer wants to create a 4K x 24 memory. How many 1 KB chips will be needed?

Check Show answer

CHALLENGE ACTIVITY 6.12.1: Composing memory.

347136.1658324.qx3zqy7

Start

A designer wishes to build a 128-bit wide memory using 8G x 16 memory chips.

How many memory chips are needed?

Ex: 5

1 2 3

Check Next

6.13 Product Profile: Cell phone

Cellular networks

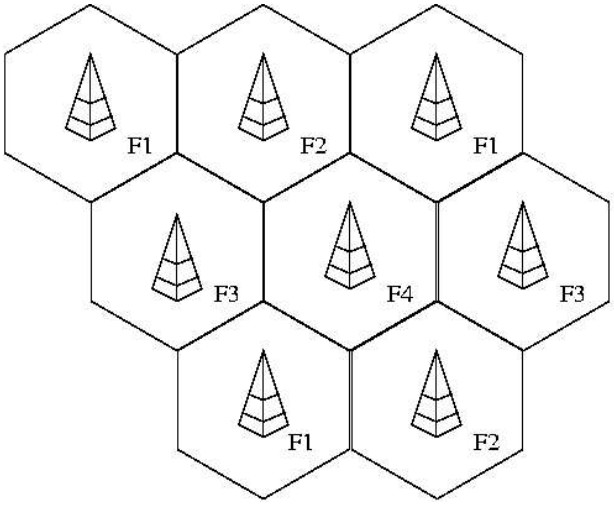
A ***cellular network,*** or ***mobile network***, is a network primarily used for communication whose endpoint is a wireless cellular device. Many cellular devices, such as cell phones, smartphones, and tablets, use cellular networks for voice, text, video communication, and Internet access. A cellular network is dispersed over a land area, which is broken up into smaller areas called a ***cell***. Each cell contains one or more ***cell towers***, or ***base stations***, which provide an interface between wireless devices and wired devices. A control channel determines which tower a device within the cell connects to, but often the chosen tower is closest to the device. Radio waves deliver the signal between the tower and the device.

Cellular service is provided by cellular network providers. Different providers use different cell towers and therefore have different coverage areas. Cellular network providers in the U.S. include Verizon, AT&T, Sprint, and T-Mobile.

Figure 6.13.1: A cell network is divided into smaller regions called cells

(left), each cell contains one or more cell towers (right)

(left), each cell contains one or more cell towers (right).



Source: Cells (Mozzerati/CC SA 1.0) and cell tower (Nikhilb239/CC BY-SA 4.0) via Wikimedia Commons

PARTICIPATION ACTIVITY 6.13.1: Cellular networks.

Cellular network providers **Cellular devices** **Cell** **Radio waves**

	A smaller land area within a cellular network that contains a cell tower or base station.
	Verizon, AT&T, Sprint
	Mechanism used by a cell tower and cellular device to communicate.
	Cell phones, smartphones, tablets

Reset

Cell tower switching

As a cell phone moves throughout an area, the phone may receive a signal from different cell towers. Each cell has a set of frequencies available. When a cell phone makes or receives a call, the phone receives a signal at the frequency of a nearby tower. Frequencies can be reused in other cells, but to reduce signal interference, the same frequencies are often not used in adjacent cells. Thus, upon entering a new cell, the phone likely receives a signal from a different tower at a different frequency; this process is called ***switching*** or ***handover***.

PARTICIPATION ACTIVITY 6.13.2: Cell phone switching between cell towers.

Animation captions:

1. Each cell tower transmits a signal at a certain frequency. Adjacent cells transmit at different frequencies to reduce interference.
2. A call begins inside of a particular cell. The phone communicates using the frequency of a nearby cell tower.
3. New cells may be entered as the phone moves within a cellular network. Upon entering a new cell, the phone switches frequencies to communicate with the new cell tower.

Dropped phone calls result from an unsuccessful handoff, which can be caused by a number of factors:

- All towers in a cell have no available frequencies (Ex: No service at football stadium during a game because of the large number of people and devices)
- Signal interference from man made or natural structures (Ex: Tall buildings, thick walls, mountains)
- Few or no cellular network provided cells in the given area (Ex: Blank areas on a Verizon coverage map)

PARTICIPATION ACTIVITY 6.13.3: Cell tower switching and dropping calls.

1) When moving to a new cell during a call, a cell phone continues to receive a signal from the tower in the original cell.
   - ○ True
   - ○ False

2) When moving to a new cell during a call, a cell phone receives a signal from a new tower likely with the same original frequency.
   - ○

True

False

3) A phone call can drop because a signal cannot be switched to a nearby tower.

True

False

Cellular phone calls

All the cell towers of a service provider connect to a central mobile telephone switching office. The switching office assigns phone calls to specific radio frequencies within a cell, handles switching among cells of a phone moving between cells, and links the cellular system to the regular landline system.

PARTICIPATION ACTIVITY 6.13.4: Cellular phone calls.

Animation content:

undefined

Animation captions:

1. When Levi turns on his phone, Levi's phone listens for a special radio frequency that communicates commands from cell tower A. If no signal is received, Levi's phone reports a "no service" error.
2. When Levi's phone receives a signal from cell tower A, the phone transmits a registration request with the phone's ID. Each cell phone is associated with a unique ID.
3. Cell tower A communicates Levi's phone's ID to the switching office, which records that the phone is in cell A. Levi's phone intermittently sends a control signal to update the phone's location.
4. Jo calls Levi. Jo's phone is in cell B. So Jo's phone makes the call request with cell tower B. Cell tower B communicates with the switching office to locate Levi's phone, which is in cell A.
5. The switching office assigns a pair of frequencies for the call. One frequency is for talking, and one is for listening. The frequency pair is called a channel. Levi and Jo's phones each have a channel assigned.
6. If a call comes in over the regular phone system, the call first goes to the switching office. The switching office locates the callee's cell phone to connect the call.

PARTICIPATION ACTIVITY 6.13.5: Cellular phone calls.

Consider two callers, Eli and Ann. Eli is in cell A, and Ann is in cell B.

1) Eli calls Ann. What happens if cell A has no available frequencies?
   - Eli hears a message indicating that Ann's phone is unavailable.
   - Cell tower A borrows a pair of frequencies from a nearby cell tower to facilitate the call.
   - Cell A will always have available frequencies.

2) Consider a third caller, Ray, using a landline phone. Ray places a call to Eli. What happens next?
   - Ray gets an error message because a call cannot be placed from a landline to a cell phone.
   - The call is received at the switching office, and the records are checked for an ID match for Eli's phone.
   - Ray's landline phone directly connects to cell tower A.

3) If Eli was driving while talking to Ray, which scenario would cause the call to drop?
   - Eli's phone disconnects from the car's bluetooth.
   - Eli's phone's battery dropped to 20%.
   - While driving, Eli switches to a new cell, and the new cell tower does not have any available frequencies to assign to the call.

4) The frequency pair, called a channel, enables ____.
   - only talking
   - only listening
   -

talking and listening simultaneously

Basic cell phone components

A cell phone requires sophisticated digital circuitry to carry out calls.

PARTICIPATION ACTIVITY 6.13.6: Basic cell phone components.

Animation content:

undefined

Animation captions:

1. A cell phone has printed circuit boards with several chips implementing analog and digital circuits. Analog components include the antenna and radio transceivers.
2. An ADC (analog-to-digital converter) converts the received analog radio signals to a digital stream of 1s and 0s. Conversely, a DAC (digital-to-analog converter) converts a digital stream to an analog signal.
3. The DSP (digital signal processor) performs numerous signal-processing tasks, including filtering, speech compression, and voice-coding, on the radio signal.
4. A processor executes operations implemented in software, include running the phone's operating system and exciting installed apps. Non-volatile memory is used to store user information, contacts, apps, etc.
5. The encoder/decoder that connects to the microphone and speaker. Other essential components include the SIM card, battery, touchscreen display, and cameras.

PARTICIPATION ACTIVITY 6.13.7: Basic components of a cell phone.

1) If a cell phone battery dies or is removed from the phone, all data stored on the phone is lost.
   - ○ True
   - ○ False

2) A cell phone antenna can only receive signals.
   - ○ True
   - ○ False

Carrier frequency

Many filtering operations are used in cell phone communication. For example, filtering is used to remove the carrier frequency from a signal to retrieve the data transmitted. A ***carrier frequency*** is a signal added to a voice signal for the purpose of transmission using a process called modulation. The carrier signal may be a sine wave of a particular frequency. Ex: A base station might broadcast a wave at 1700 MHz, where 1700 MHz is the carrier frequency. A receiving device, like a cell phone, locks on to the carrier frequency, and then filters out the carrier signal, to recover the voice data signal.

PARTICIPATION ACTIVITY 6.13.8: Carrier frequency.

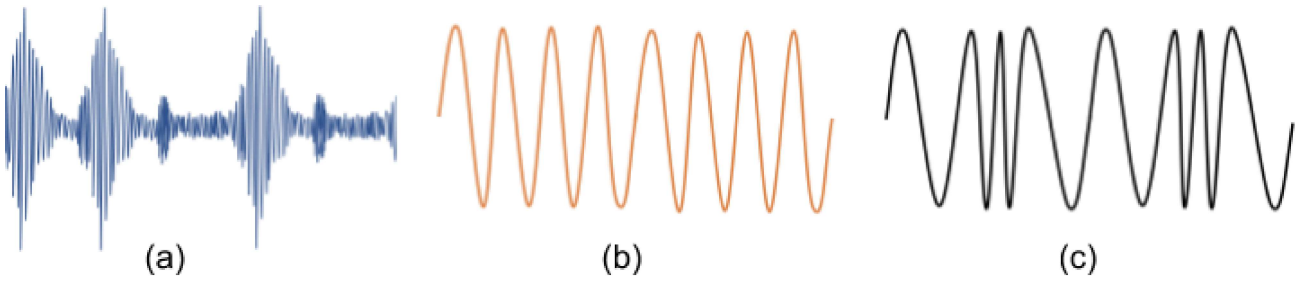
Animation content:

undefined

Animation captions:

1. To transmit a voice signal, a cell phone multiplexes the voice data signal with the carrier frequency of the assigned channel before transmitting the signal to the cell tower.
2. When a voice signal is received, a cell uses a demodulation filter to filter out the carrier signal and recover the original voice data.

PARTICIPATION ACTIVITY 6.13.9: Carrier frequency.



Voice input | Modulated signal | Carrier frequency

(a)

(b)

(c)

Reset

Digital filtering basics

While filtering radio signals can be implemented using analog components, digital signal processing is commonly used in cell phones. Digital signal processing operates on a stream of digital data that comes from digitizing an input signal, such as an audio, video, or radio signal. A ***digital filter*** takes a stream of digital inputs and generates a stream of digital outputs with some feature of the input stream removed or modified. A ***stream*** is a sequence of values separated by a fixed amount of time. Filtering a data stream is the task of removing particular aspects of the input signal, and outputting a new signal without those aspects. One common filter used by many systems is to remove noise from a signal.

PARTICIPATION ACTIVITY 6.13.10: Filtering basics.

Animation content:

undefined

Animation captions:

1. A stream of digital temperature values coming from a car engine's oil temperature sensor is sampled every second. A digital filter is used to remove inaccurate measurements. The filter has an input X and an output Y.
2. Any sudden spike in the temperature is not accurate. The digital filter averages the values to reduce or remove such noise and generates an output stream Y.
3. An FIR filter multiplies the current and previous values by constants and adds the results to compute a weighted average. x(t) is the current value, x(t-1) is the previous value, and x(t-2) is the value before x(t-1).

PARTICIPATION ACTIVITY 6.13.11: Filter basics.

1) For the above example, given x(t) = 189, x(t-1) = 181, x(t-2) = 182, what is the filter output, Y?

Check Show answer

2) To create an FIR filter that computes the average of the

current and three previous values, the FIR filter's equation becomes: y(t) = c0 * x(t) + c1 * x(t-1) + c2 * x(t-2) + c3 * x(t-3)
What constant should be used for c0? Type as #.##

Check Show answer

FIR filter datapath

An ***FIR filter*** or a ***finite response filter*** multiplies the present input value by a constant, and adds that result to the previous input value times a constant, adds that result to the next-earlier input value times a constant, and so on. A designer using an FIR filter achieves a particular filtering goal by choosing the FIR filter's constants, c0, c1, c2, etc. Mathematically, an FIR filter can be described as follows:

$$y(t) = c0 * x(t) + c1 * x(t-1) + c2 * x(t-2)$$

t is the present time step. x is the input signal, and y is the output signal. Each term $c0 * x(t)$ is called a tap. So the equation represents a 3-tap FIR filter.

PARTICIPATION ACTIVITY 6.13.12: Datapath for a 3-tap FIR filter.

Animation content:

undefined

Animation captions:

1. A 3-tap FIR filter's equation has three terms. A particular filtering goal is achieved by choosing the filter's constants.
2. The FIR filter can be implemented as a digital datapath. X is the datapath input, and Y is the datapath output.
3. Three registers are instantiated to hold the X values. The X_ld input connects to all X registers' ld input. If X_ld is 1, xt0 is loaded with X, xt1 is loaded with xt0, and xt2 is loaded with xt1, simultaneously.
4. The c0, c1, and c2 registers and instantiated to hold the FIR filter's constant values.
5. 3 multipliers and 2 adders are instantiated to implemented the FIR computation. The output of the second adder connects to output Y.
6. For the moving average filter example, the constants are 0.33, 0.33, and 0.34. If x(t-2) = 184, x(t-1) = 179, and x(t) = 180, Y will be 181.03.

PARTICIPATION ACTIVITY 6.13.13: FIR filter datapath.

1) How many registers will a 7-tap FIR filter require?
   - 7
   - 14
   - 16

2) How many multipliers will a 7-tap FIR filter require?
   - 6
   - 7
   - 13

3) Consider the datapath of a 100-tap filter designed similar to the above example. How many adders will a 100-tap filter need?
   - 50
   - 99
   - Unknown

4) How many taps can an FIR filter have?
   - A minimum of 25 taps
   - A maximum of 100
   - The number of taps of an FIR filter depends on the application.

©zyBooks 10/17/21 22:06 829162
Ivan Neto
UCRCS120AEE120AChenFall2021