PURDUE UNIVERSITY DEPARTMENT OF PHYSICS

Physics 53600

Electronics Techniques for Research

Now in PowerPoint!

Spring 2020 Semester

Prof. Matthew Jones

ANNOUNCEMENT

- Obvious changes to the course:
  - No in-person lectures: you'll have to read the lecture notes yourself
  - No more labs: don't worry about it – your grade will be based on work done so far
  - Remaining assignments will try to cover topics that would have been explored in the lab
  - Second mid-term: simplest to cancel it
  - Final exam: I'm not sure what to do about this yet, but I'll figure something out.
- Changes to grading scheme:
  - Old scheme: Assignments (30%) exams (40%) lab (30%)
  - New scheme: Assignments (50%) exams (25%) lab (25%)

ANNOUNCEMENT

- Because there won't be any in-person lectures, you will have to read the lecture notes yourself.
- To demonstrate that you have read them, you will be required to answer *one or two simple questions* before the next lecture is posted.
- The question will be somewhere (like maybe at the end?) and you just have to e-mail me the answer

  mjones@physics.purdue.edu
- To make this easy, please make your subject look like this:

  "PHYS53600 Lecture xx questions Your Name"
- These will be part of your assignment grade, maybe contributing 10% of your total grade.

ANNOUNCEMENT

- Remember that you have Assignment #4 due today, March 26th!
- Please scan, or somehow generate a PDF file of your solutions and e-mail them to the grader

  lim185@purdue.edu
- It might be useful to use a standard subject:

  "PHYS53600 Assignment #4 Your Name"

LECTURE 19 QUESTION #1

Why is a ripple counter a bad design for a binary counter?

LECTURE 19 QUESTION #2

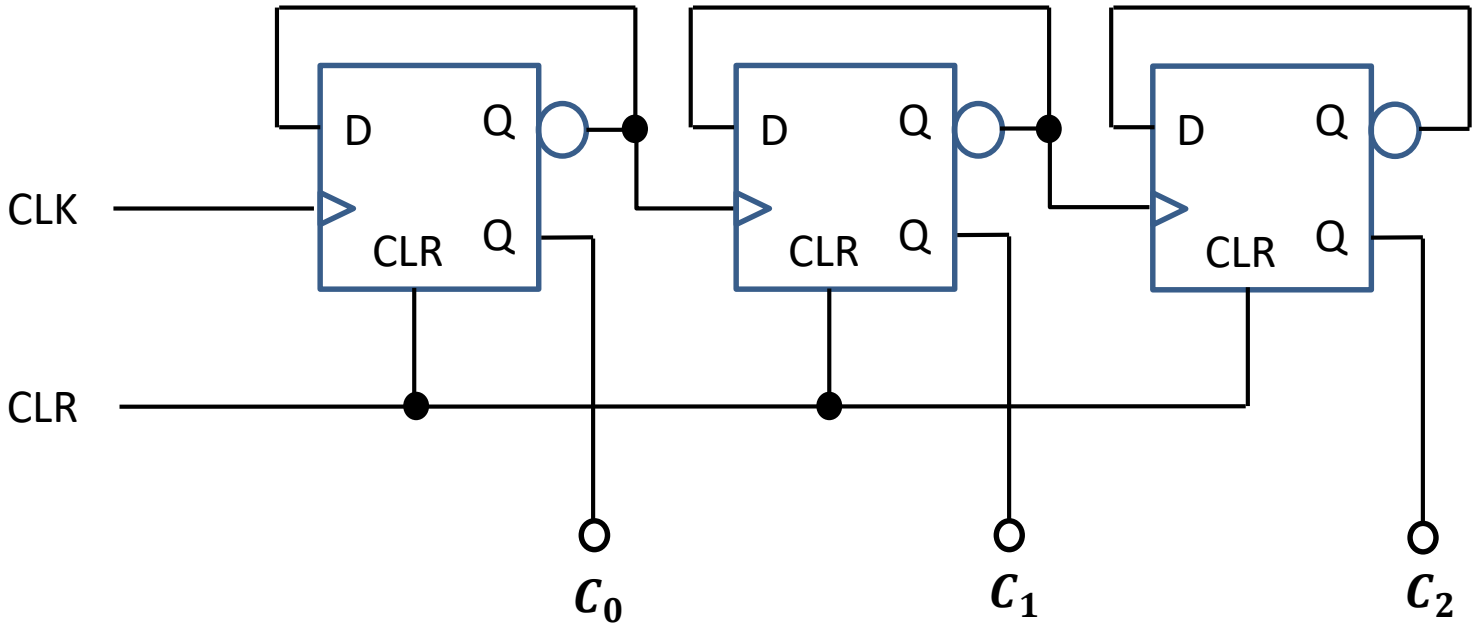
Why is it good design practice to distinguish between clock signals and normal logic signals?

Other Sequential Logic Elements

- More sophisticated functional components can be constructed out of flip-flops
- Usually, you don’t need to understand exactly how they are constructed – you mostly need to understand their interfaces
- You can normally treat them as “black boxes” where the details are hidden, but the interface is well-defined.

Counters

- The first example is a “ripple counter”:



Counters

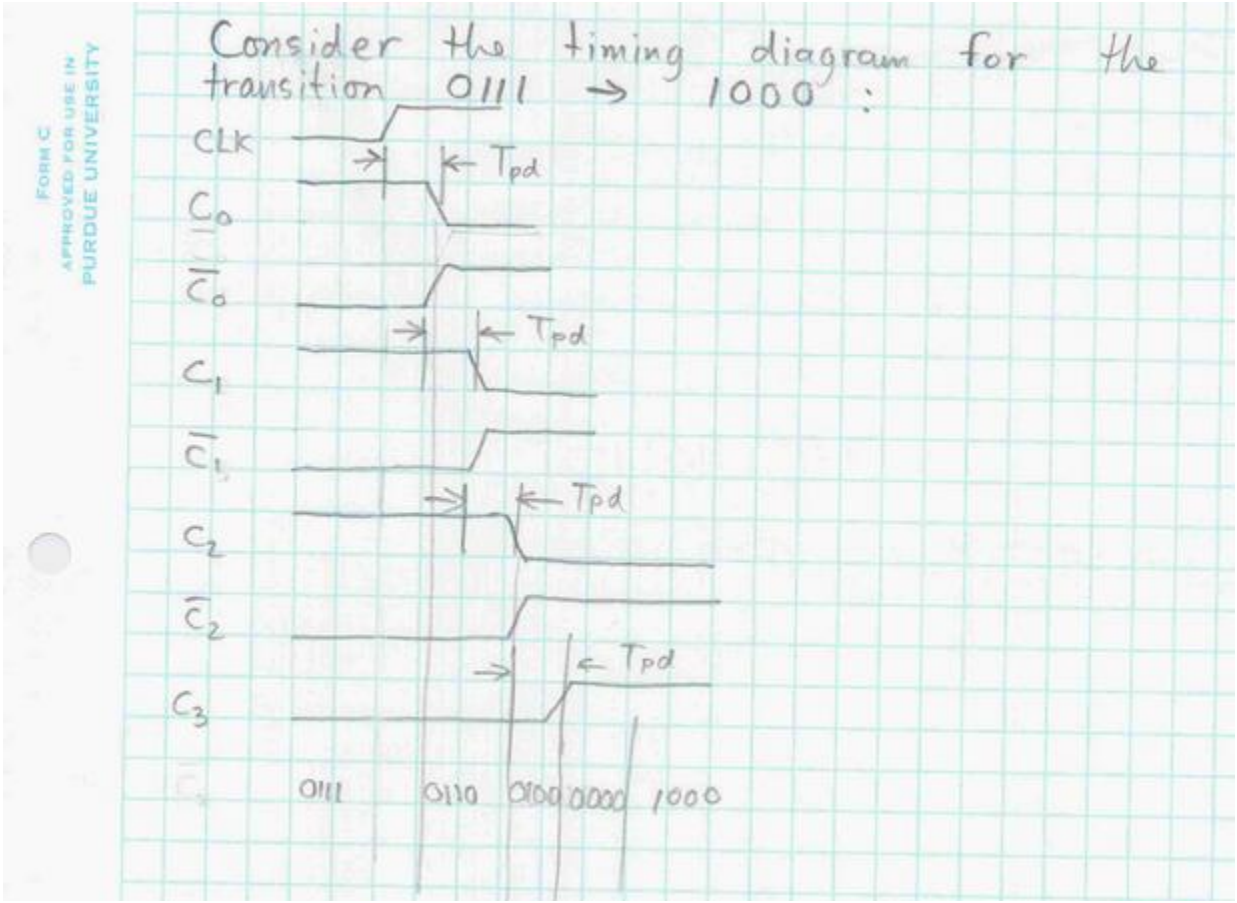
- The outputs simply count in binary:

Clock cycle	C_2	C_1	C_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Ripple Counters

- Ripple counters are easy to construct
  - You just wire up a bunch of flip-flops with no additional logic
- The problem arises due to the finite propagation delay that is inherent in all logic circuits
- They have the undesirable property of having lots of intermediate states
  - The output doesn't simply switch from one binary value to the next
  - There are lots of intermediate values that are rapidly cycled through before the final number stabilizes

Ripple Counters



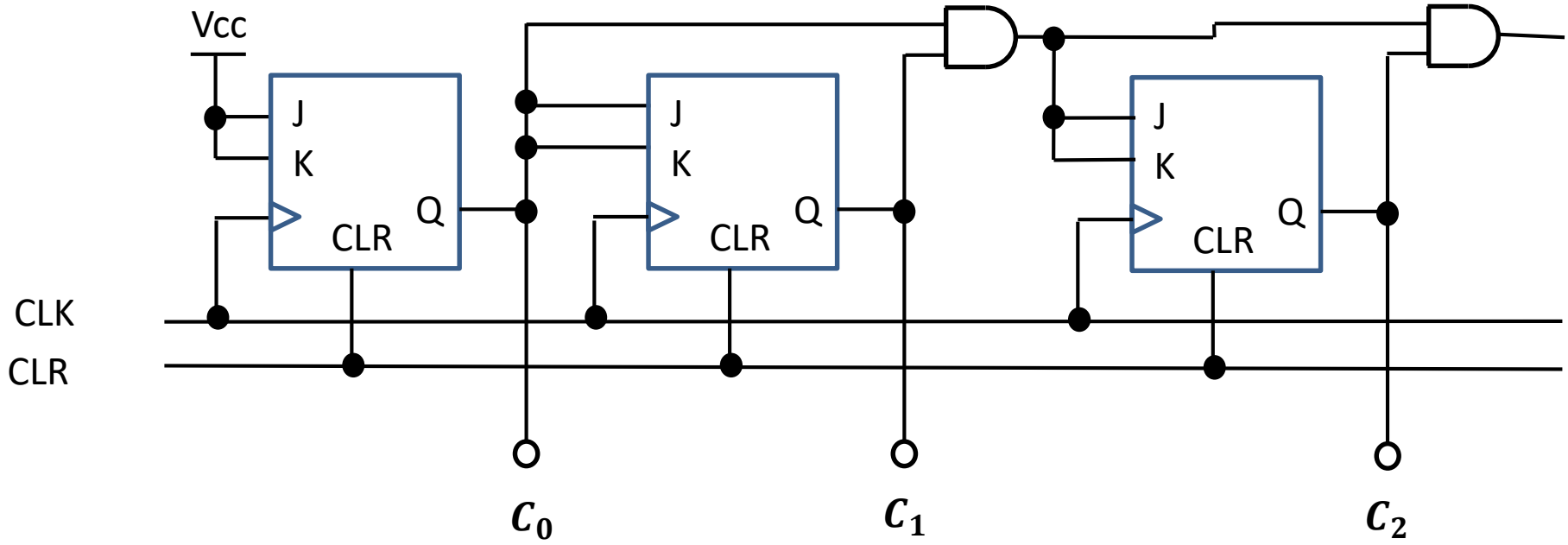
The output will briefly give the values 0111, 0110, 0100, 0000 before stabilizing at 1000.

Ripple Counters

- When would this *not* matter?
  - Consider a human interface that displays a count on (for example) a 7-segment LED display
  - The intermediate transitions probably happen too fast for the human eye to perceive
- When *would* this matter?
  - If the output was latched on the leading edge of the input clock, then the maximum possible clock frequency would be $f_{max} \approx 1/(n \times T_{pd})$
  - This rapidly becomes very impractical

Synchronous Counters

- Consider the following circuit constructed of JK flip-flops
  - They toggle when J=K=1
  - No transition when J=K=0



Synchronous Counters

- The big difference between the ripple counter and this design is that now the inputs are valid long before the next clock edge.
- The output switches synchronously with the clock edge with no intermediate states.
- The maximum clock frequency is $f_{max} \approx 1/T_{pd}$ and is independent of the number of stages

Dedicated Clock Networks

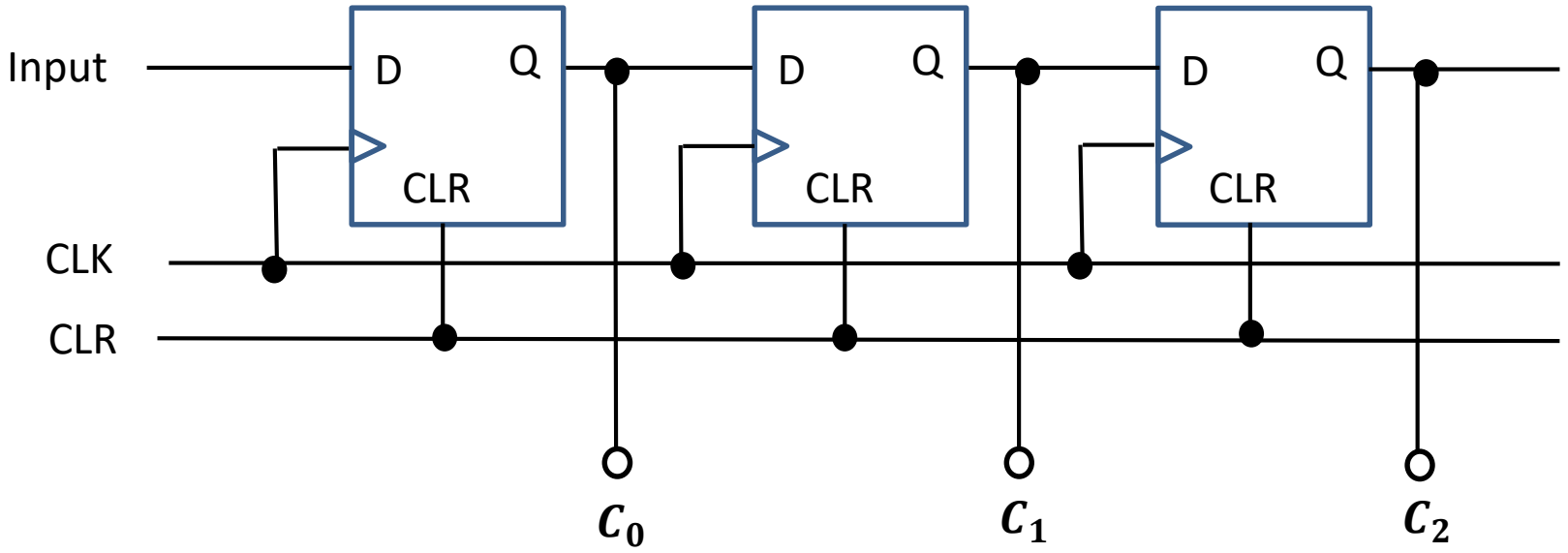
- It can be important to distinguish between clock signals and normal logic signals
- Clock signals:
  - Often drive a large number of inputs (large C)
  - This requires drivers with low output impedance R so that RC remains small
  - Dedicated routing (ie, wiring) allows the signals to arrive at all inputs at (nearly) the same time (this is called clock skew)
  - This helps to make the behavior of logic circuits more predictable

Dedicated Clock Networks

- It can be important to distinguish between clock signals and normal logic signals
- Logic signals:
  - Usually only drive a relatively small number of other inputs
  - Only timing constraint is their signal propagation is faster that the clock period (plus setup time)
  - Don’t need low output impedance drivers
  - Lower power consumption
  - Fewer constraints on signal routing (ie, wiring)

Other Logic Elements

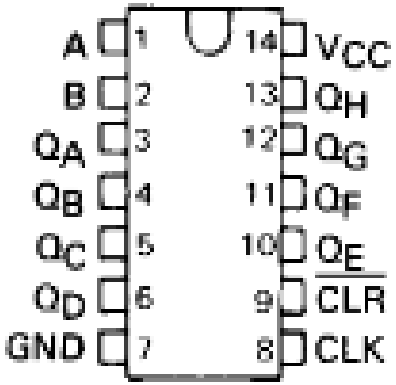
- Lots of other logic elements can be constructed using flip-flops.
- Shift register:



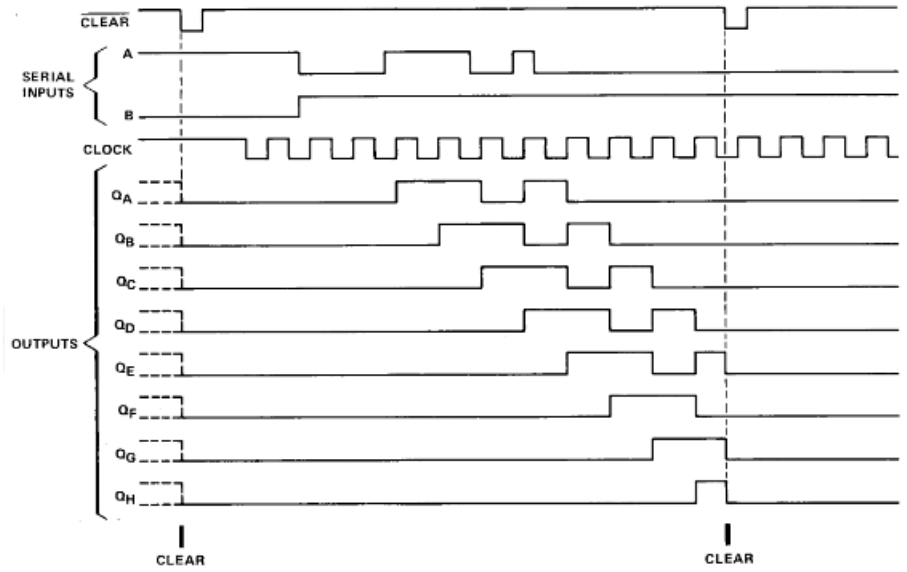
Shift Register

- Serial to parallel shift register: 74LS164

SN54164, SN54LS164 . . . J OR W PACKAGE
SN74164 . . . N PACKAGE
SN74LS164 . . . D OR N PACKAGE
(TOP VIEW)



typical clear, shift, and clear sequences



FUNCTION TABLE

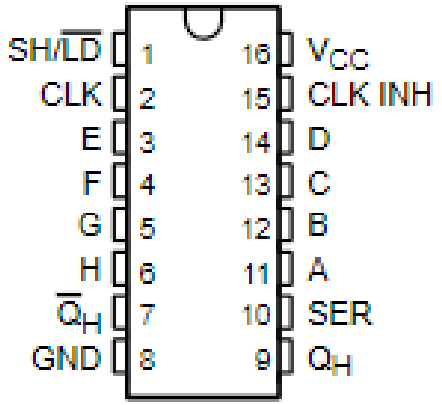
INPUTS				OUTPUTS			
$\overline{\text{CLEAR}}$	CLOCK	A	B	Q_A	Q_B	...	Q_H
L	X	X	X	L	L		L
H	L	X	X	Q_{A0}	Q_{B0}		Q_{H0}
H	↑	H	H	H	Q_{An}		Q_{Gn}
H	↑	L	X	L	Q_{An}		Q_{Gn}
H	↑	X	L	L	Q_{An}		Q_{Gn}

H = high level (steady state), L = low level (steady state)
X = irrelevant (any input, including transitions)
↑ = transition from low to high level.
Q_{A0}, Q_{B0}, Q_{H0} = the level of Q_A, Q_B, or Q_H, respectively, before the indicated steady-state input conditions were established.
Q_{An}, Q_{Gn} = the level of Q_A or Q_G before the most-recent ↑ transition of the clock; indicates a one-bit shift.

Shift Register

- The 74LS195 is a parallel-to-serial shift register:

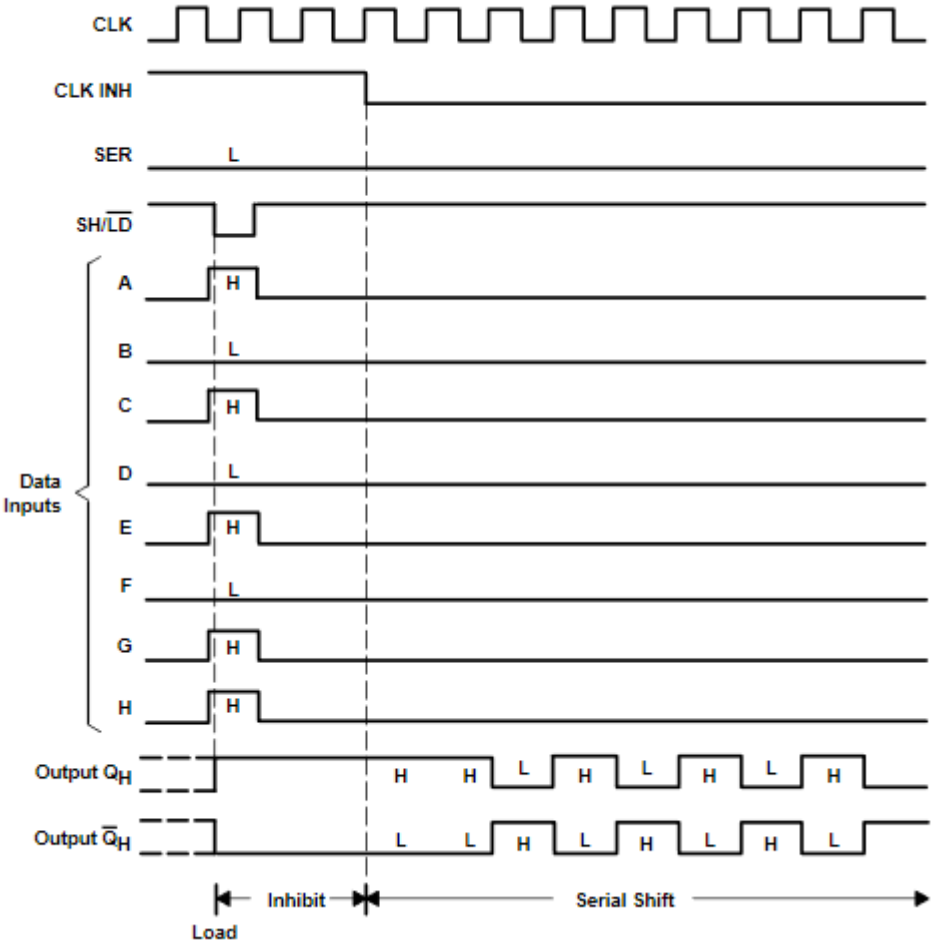
SN54165, SN54LS165A . . . J OR W PACKAGE
SN74165 . . . N PACKAGE
SN74LS165A . . . D, N, OR NS PACKAGE
(TOP VIEW)



FUNCTION TABLE

INPUTS					INTERNAL OUTPUTS		OUTPUT
SH/$\overline{LD}$	CLK INH	CLK	SER	PARALLEL A . . . H	$\overline{Q}_A$	$\overline{Q}_B$	Q_H
L	X	X	X	a . . . h	a	b	h
H	L	L	X	X	Q_{A0}	Q_{B0}	Q_{H0}
H	L	↑	H	X	H	Q_{An}	Q_{Gn}
H	L	↑	L	X	L	Q_{An}	Q_{Gn}
H	H	X	X	X	Q_{A0}	Q_{B0}	Q_{H0}

typical shift, load, and inhibit sequences



Memory

- The basic function of a memory element is to store and subsequently retrieve data (ie, bits)
- Required inputs:
  - Data (one bit)
  - Address (n bits)
- The number of bits that can be stored determines the width of the address bus
  - If the address bus has n bits, then you can store 2^n bits in the memory element.
- Required outputs:
  - Data (one bit)
- Other signals are needed to control the operation
  - Read vs write, for example.

Conceptual Design

- Conceptually, a memory element could be constructed using flip-flops (to store each bit) and a decoder (combinatorial logic) to enable an individual memory element selected by the input address.
- To store multiple bits simultaneously, just use multiple one-bit memory elements in parallel.
- In practice, there are lots of tricks to make memory more efficient (cost, size, speed)

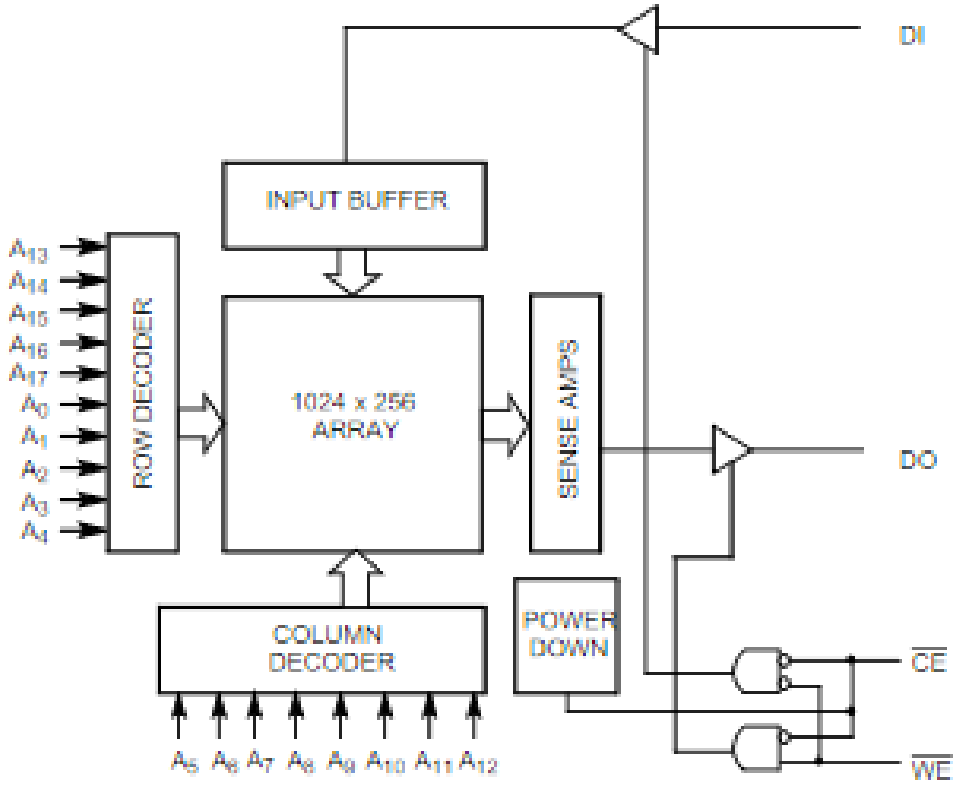
Example Memory Element

CYPRESS PERFORM

CY7C197N

256 K × 1 Static RAM

Logic Block Diagram

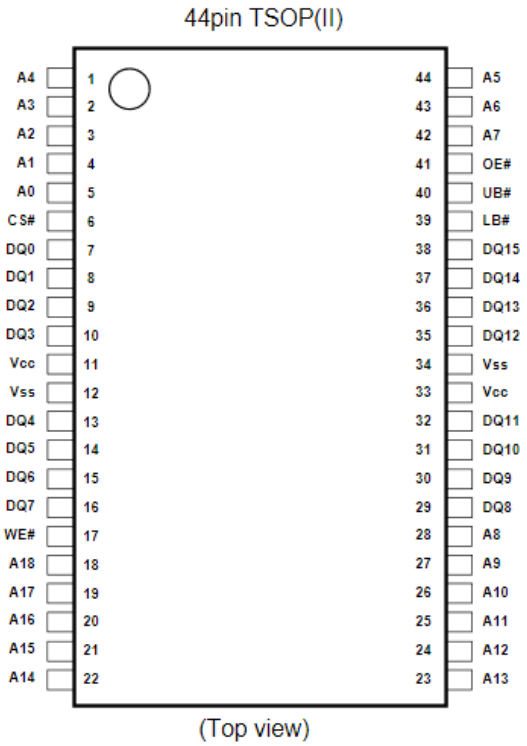


A Modern Example

RENESAS

RMLV0816BGSB - 4S2

8Mb Advanced LPSRAM (512k word × 16bit)



Pin Description

Pin name	Function
V_{CC}	Power supply
V_{SS}	Ground
A0 to A18	Address input
DQ0 to DQ15	Data input/output
CS#	Chip select
OE#	Output enable
WE#	Write enable
LB#	Lower byte select
UB#	Upper byte select

Functionally, this is very similar to the previous example.

Although each memory element stores 16 bits, you can selectively write to either the upper or the lower (or both) byte by means of the LB# and UB# inputs.

Dual Port RAM

- Sometimes reading and writing must be asynchronous
  - In a data acquisition system, data samples can be written at a constant rate
  - The process analyzing the data retrieves it as needed in an unspecified
- Dual Port RAM has two independent address and data busses.
  - Potential conflicts if both busses try to write to the same address simultaneously

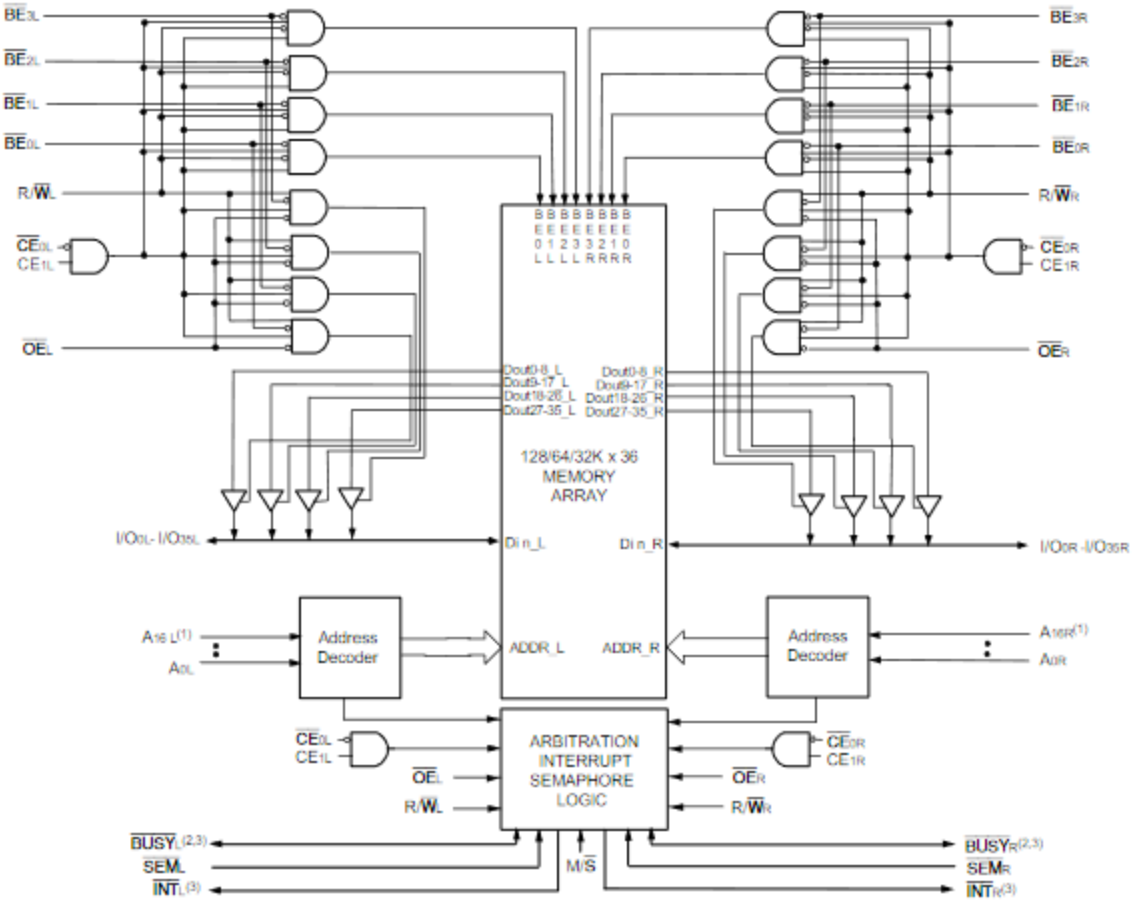
Dual Port RAM Example

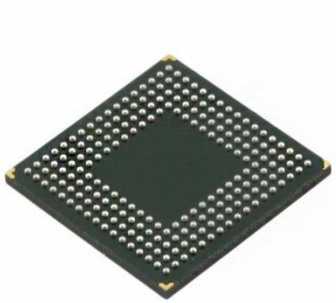
RENESAS

HIGH-SPEED 3.3V
128/64/32K x 36
ASYNCHRONOUS DUAL-PORT
STATIC RAM

IDT70V659/58/57S

Functional Block Diagram





FIFO's

- Lots of applications require reading and writing data at different rates.
- A FIFO (First In First Out) is a block of memory where the read and write addresses are automatically incremented with each read/write.
- The AVERAGE write rate must not exceed the AVERAGE read rate, or else the memory will fill up.
- The INSTANTANEOUS rates can be very different.

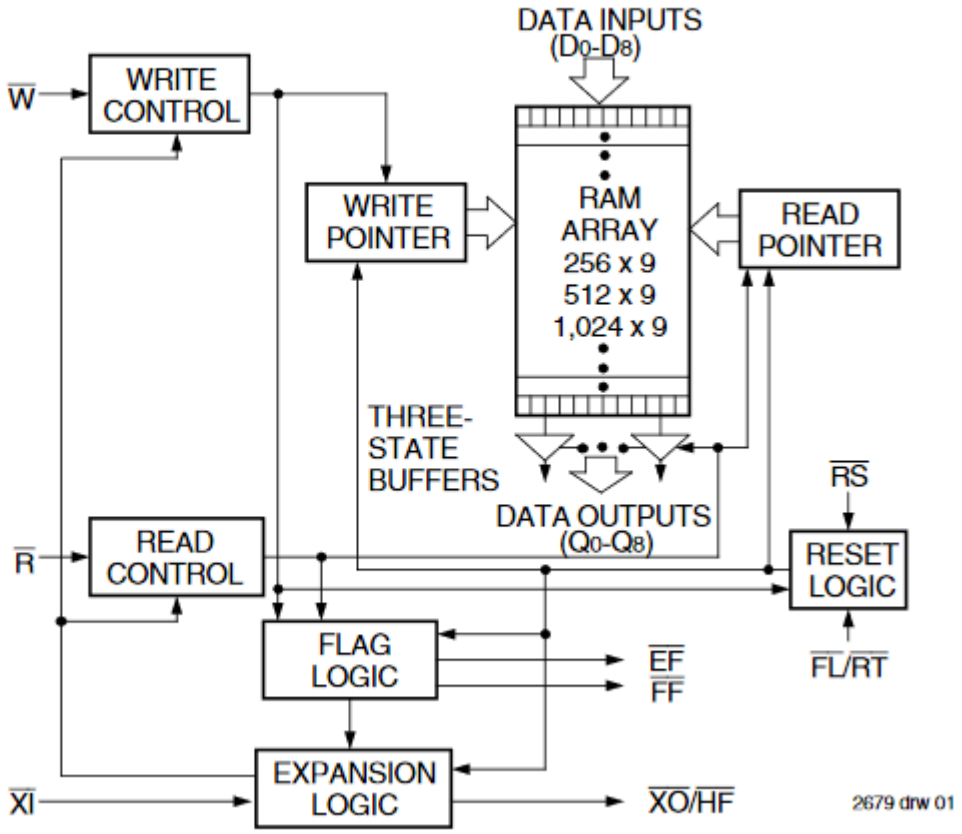
FIFO's

RENESAS

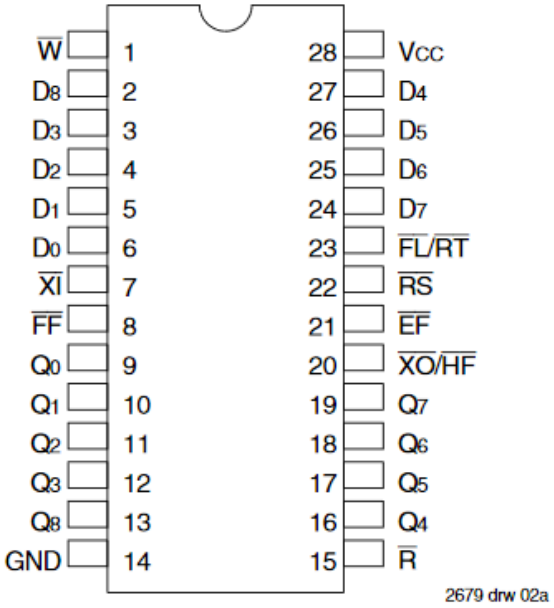
CMOS ASYNCHRONOUS FIFO
256 x 9, 512 x 9 and 1,024 x 9

IDT7200L
IDT7201LA
IDT7202LA

FUNCTIONAL BLOCK DIAGRAM



PIN CONFIGURATIONS



Dynamic RAM

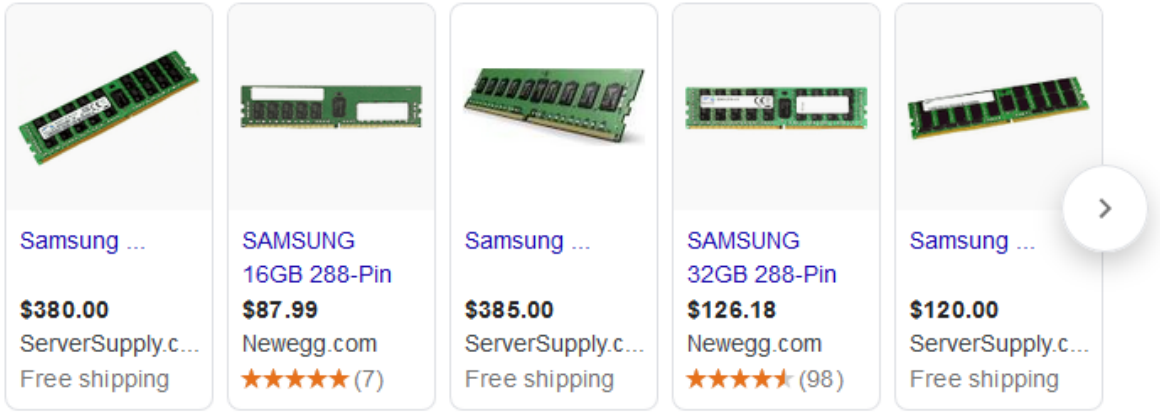
- The previous examples are static RAM (SRAM) which work as described.
- They retain the information stored in them as long as power is applied.
- An alternative is dynamic RAM (DRAM):
  - Lower power
  - Higher speed
  - Higher density
  - Lower cost
  - But it is forgetful… it needs to be periodically reminded even when power is applied.

Dynamic RAM

- A dynamic RAM storage element consists of one transistor and one capacitor per bit.
- This is much simpler than using a flip-flop for each bit.
- Charge stored in each memory element can gradually dissipate, so the DRAM contents needs to be periodically refreshed.
- While it is being refreshed, it is not available for read/write operations.
- Requires external memory controller components, but on average, the cost is lower and the speed is higher.

Current State of the Art

Usually used as packaged modules that combine all the necessary support logic, or which interface to standard interface signals.



Samsung Begins Mass Production of Industry's First 16GB LPDDR5 DRAM for Next-Generation Premium Smartphones

Korea on February 25, 2020

Audio Share

Other Types of RAM

- Read only memory (older technology)
  - Retains data even when power is removed
  - Can only be programmed once, using special circuitry.
  - Erasable ROM could be erased by exposing to UV light.
- Electrically Erasable Read Only Memory
  - Can be programmed and erased electronically
  - Slower than SRAM or DRAM
  - Limited number of erase/write cycles
  - Example: SIM cards, SD cards, thumb drives, etc…