DIGITAL ELECTRONICS AND MICROPROCESSOR

TH3

5th SEM ELECTRICAL ENGG.

PREPARED BY:

ER. BIKRAM KESHARI PARIDA

[Lecturer Dept. of EE, KALINGA NAGAR POLYTECHNIC,TARAPUR, JAJPUR ROAD]

1 BasicDigitalConcepts

By converting continuous analog signals into a finite number of discrete states, a process called*digitization*, then to the extent that the states are sufficiently well separated so that noise does create errors, the resulting digital signals allow the following (slightly idealized):

- storageoverarbitraryperiodsoftime
- flawlessretrievalandreproductionofthestoredinformation
- flawlesstransmissionoftheinformation

Someinformationisintrinsicallydigital,soitisnaturaltoprocessandmanipulateit usingpurelydigitaltechniques.Examplesarenumbersandwords.

The drawback to digitization is that a single analog signal (*e.g.*a voltage which is a functionoftime,likeastereosignal)needsmanydiscretestates,or*bits*,inordertogive a satisfactory reproduction.For example, it requires a minimum of 10 bits to determine a voltageat any given time to an accuracy of $\approx$0.1%.For transmission, one now requires 10 lines instead of the one original analog line.

The explosion in digital techniques and technology has been made possible by the incredible increase in the density of digital circuitry, its robust performance, its relatively low cost,anditsspeed.Therequirementofusingmanybitsinreproductionisnolongeranissue: The more the better.

Thiscircuitryisbaseduponthetransistor,whichcanbeoperatedasaswitchwith two states.Hence, the digital information is intrinsically*binary*.So in practice, the terms digital and binary are used interchangeably.In the following sections we summarize some conventions for defining the binary states and for doing binary arithmetic.

BinaryLogicStates

The following table attempts to make correspondences between conventions for defining binary logic states.In the case of the TTLlogic gates we will be using in the lab, the Lowvoltagestateisroughly0–1VoltandtheHighstateisroughly2.5–5Volts.Seepage475of thetextfortheexactconventionsforTTLaswellasotherhardwaregatetechnologies.

Boolean Logic	BooleanAlgebra	VoltageState (positivetrue)	VoltageState (negativetrue)
True(T)	1	High(H)	Low(L)
False(F)	0	L	H

The convention for naming these states is illustrated in Fig.1.The “positive true” case isillustrated.Therelationshipbetweenthelogicstateandlabel(inthiscase“switchopen”) at some point in the circuit can be summarized with the following:

*Thelabelledvoltageis*High(Low)*whenthelabel’sstatedfunctionis*True(False).

Inthefigure,thestatedfunctioniscertainlytrue(switchopen),andthisdoescorrespondto ahighvoltageatthelabelledpoint.(Recallthatwiththeswitchopen,Ohm’sLawimplies thatwithzerocurrent,thevoltagedifferenceacrossthe“pullup”resistoriszero,sothat

thelabelledpointisat+5Volts.Withaclosedswitch,thelabelledpointisconnectedto ground, with a 5 Volt drop across the resistor and a current of $I=V/R$=5 mA through it.)

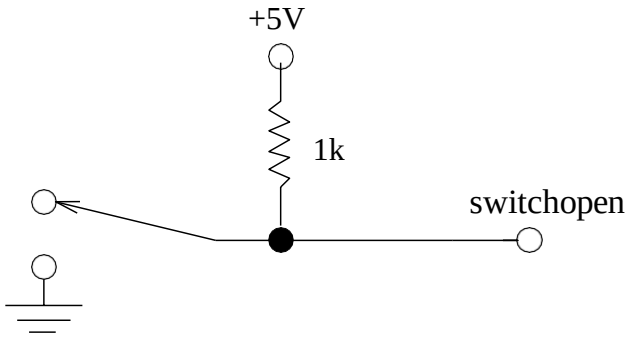


Figure1:Illustrationforlabellinglogicstates("positivetrue").

Withtheconventionknownas"negativetrue",thelabelwouldbechangedto"switch closed"withabaroverit:switch$\overline{\text{closed}}$.Ourstatementbecomes:

*Thelabelledvoltageis*Low(High)*whenthelabel'sstatedfunctionis*True(False).

Sointhefigure,thestatedfunction(switchclosed)istruewhenthevoltageislow.Thebar ismeanttoenvoketheBooleaninversionoperation:$\bar{T}$=F,$\bar{F}$=T,$\bar{\bar{T}}$=T,andsoforth.

BinaryArithmetic

Eachdigitinbinaryisa0ora1andiscalleda*bit*,whichisanabbreviationof*binarydigit*. Thereareseveralcommonconventionsforrepresentationofnumbersinbinary.

Themostfamiliaris*unsignedbinary*.Anexampleofa8-bitnumberinthiscaseis

$$01001111_2=0\times2^7+1\times2^6+\cdots+1\times2^0=64+8+4+2+1=79_{10}$$

(Generallythesubscriptswillbeomitted,sinceitwillbeclearfromthecontext.)Toconvert from base 10 to binary, one can use a decomposition like above, or use the following algorithm illustrated by 79:79/2=39, remainder 1, then 39/2=19 r 1, and so forth.Then assemble all the remainders in reverse order.

The largest number which can be represented by$n$bits is 2^n-1.For example, with 4 bits the largest number is 1111_2= 15.

The most significant bit (MSB) is the bit representing the highest power of 2, and the LSB represents the lowest power of 2.

Arithmeticwithunsignedbinaryisanalogoustodecimal.Forexample1-bitaddition andmultiplicationareas follows:0+0=0, 0+1=1, 1+1=0, $0\times0=0$, $0\times1=0$, and $1\times1=1$.Notethatthisis*differentfromBooleanalgebra*,asweshallseeshortly,where $1+1=1$.

Another convention is called*BCD*("binary coded decmal").In this case each decimal digitisseparatelyconvertedtobinary.Therefore,since7=0111_2and9=1001_2,then 79 =01111001(BCD).Notethatthisis*different*thanourpreviousresult.Wewilluse BCD quiteoftenthiscourse.It isquiteconvenient, forexample, when decimalnumerical displays are used.

Yetanotherconventionis*Graycode*.Youhaveahomeworkproblemtopracticethis. Thisislesscommonlyused.

RepresentationofNegativeNumbers

Therearetwocommonlyusedconventionsforrepresentingnegativenumbers.

With*sign magnitude*, the MSB is used to flag a negative number.So for example with4-bit numbers we would have 0011=3 and 1011=−3.This is simple to see, but is not good for doing arithmetic.

With*2's complement*, negative numbers are designed so that the sum of a number and its 2's complement is zero.Using the 4-bit example again, we have 0101=5 and its 2's complement −5=1011.Adding (remember to carry) gives 10000=0.(The 5th bit doesn't count!)Both addition and multiplication work as you would expect using 2's complement. There are two methods for forming the 2's complement:

1. Makethetransformation0→1and1→0,thenadd1.
2. Add some number to -2^{MSB} to get the number you want.For 4-bit numbers an example of finding the 2's complementof 5 is −5= −8+3 =1000+0011 =1011.

HexadecimalRepresentation

Itisveryoftenquiteusefultorepresentblocksof4bitsbyasingledigit.Thusinbase16 there is a convention for using one digit for the numbers 0,1,2,...,15 which is called *hexadecimal*.Itfollowsdecimalfor0–9,thenuseslettersA–F.

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

2 LogicGatesandCombinationalLogic

GateTypesandTruthTables

The basic logic gates are AND,OR,NAND,NOR,XOR,INV, and BUF. The last two are not standard terms; they stand for “inverter” and “buffer”, respectively.The symbols for these gatesandtheircorrespondingBooleanexpressionsaregiveninTable8.2ofthetextwhich, forconvenience,isreproduced(inpart)inFig.2.

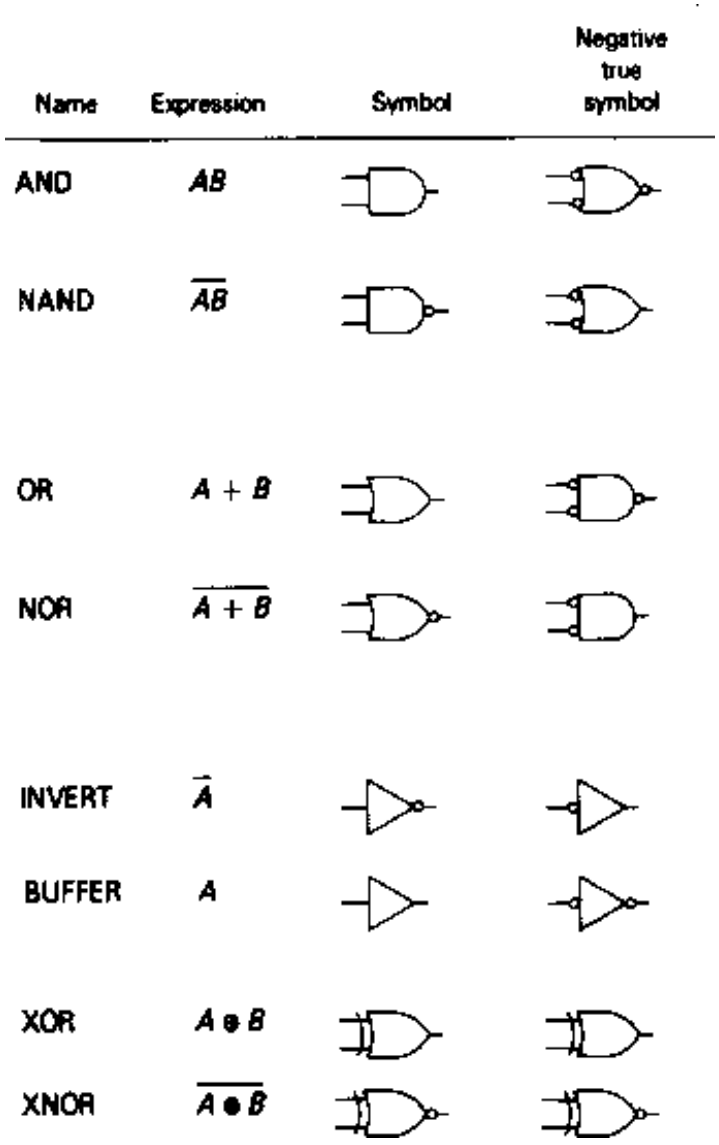
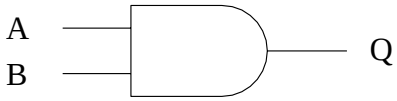


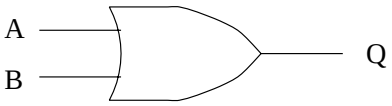
Figure2:Table8.2fromthetext.

All of the logical gate functions, as well as the Boolean relations discussed in the next section,follow from the truth tables for the AND and OR gates.We reproduce these below.We also show the XOR truth table, becauseit comes up quiteoften, although, as weshall see,it is not elemental.



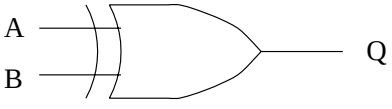
A	B	Q
0	0	0
1	0	0
0	1	0
1	1	1

Figure3:ANDgate.



A	B	Q
0	0	0
1	0	1
0	1	1
1	1	1

Figure4:ORgate.



A	B	Q
0	0	0
1	0	1
0	1	1
1	1	0

Figure5:XOR(exclusiveOR)gate.

BooleanAlgebraandDeMorgan'sTheorems

Boolean algebra can be used to formalize the combinations of binary logic states.The fundamental relations are given in Table 8.3 of the text.In these relations,*A*and*B*are binary quantities, that is, they can be either logical true (T or 1) or logical false (F or 0). Most of these relations are obvious.Here are a few of them:

$$AA=A; \qquad A+A=A; \qquad A+\overline{A}=1; \qquad A\overline{A}=0; \qquad \overline{\overline{A}}=A$$

Recall that the text sometimes uses an apostrophe for inversion (A').We use the standard overbar notation ($\overline{A}$).

Wecanusealgebraicexpressionstocompleteourdefinitionsofthebasiclogicgates we began above.Note that the Boolean operations of "multiplication" and "addition" are definedbythetruthtablesfortheANDandORgatesgivenaboveinFigs.3and4.Using thesedefinitions,wecandefineallofthelogicgatesalgebraically.Thetruthtablescanalso beconstructedfromtheserelations,ifnecessary.SeeFig.2forthegatesymbols.

- AND:$Q=AB$ (seeFig.3)
- OR:$Q=A+B$ (seeFig.4)
- NAND:$Q=\overline{AB}$
- NOR:$Q=\overline{A+B}$
- XOR:$Q=A\oplus B$ (definedbytruthtableFig.5)
- INV:$Q=\overline{A}$
- BUF:$Q=A$

Example:CombiningGates

Let's re-express the XOR operation in terms of standard Boolean operations.The following truth table evaluates the expression$Q=\overline{A}B+A\overline{B}$.

A	B	$\overline{A}B$	$A\overline{B}$	Q
0	0	0	0	0
1	0	0	1	1
0	1	1	0	1
1	1	0	0	0

Weseethatthis truthtableis identicalto theone for the XORoperation.Therefore,we can write

$$A\oplus B=A\overline{B}+\overline{A}B \qquad (1)$$

AschematicofthisexpressionintermsofgatesisgiveninFig.6(aswellasFig.8.25of the text).Recall that the open circles at the output or input of a gate represent inversion.

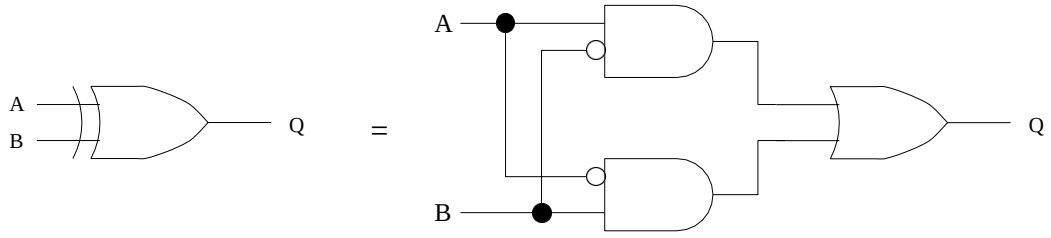


Figure6:RealizationoftheXORgateintermsofANDandORgates.

GateInterchangeablilty

In an example from the homework, we can make an INV gate from a 2-input NOR gate. Simply connect the two inputs of the NOR gate together.Algebraically, if the two original NORgateinputsarelabelled$B$and$C$,andtheyarecombinedtoform$A$,thenwehave $Q= B + C=A + A= \overline{A}$,whichistheINVoperation.

Note that an INV gate can not be made from OR or AND gates.For this reason the OR and ANDgatesarenotuniversal.Soforexample,nocombinationofANDgatescanbecombined to substitute for a NOR gate.However, the NAND and NOR gates *are universal*.

DeMorgan

Perhaps the most interesting of the Boolean identities are the two known as DeMorgan's Theorems:

$$\overline{A+B}=\bar{A}\bar{B} \qquad (\text{or},\ A+B=\overline{\bar{A}\,\bar{B}}) \tag{2}$$

$$\overline{AB}=\overline{A}+\overline{B} \qquad (\text{or},\ AB=\overline{\overline{A}+\overline{B}}) \tag{3}$$

Theseexpressionsturnouttobequiteuseful,andweshallusethemoften.

An example of algebraic logic manipulation follows. Itis the one mentioned at the end of Lab 1.One is to show that an XOR gate can be composed of 4 NAND gates.From the section above we know$A\oplus B=\overline{A}B+A\overline{B}$.Since$A\overline{A}=0$and $B\overline{B}=0$, we can add these, rearrange, and apply the two DeMorgan relations to give

$$A\oplus B=A(\overline{A+B})+B(\overline{A+B})=A(\overline{AB})+B(\overline{AB})=\overline{A(\overline{AB})}\,\overline{B(\overline{AB})}$$

SymbolicLogic

The two DeMorgan expressions above can be envoked using gate symbols by following this prescription:*Changegateshape(*AND↔OR*)andinvertallinputsandoutputs.*

ByexaminingthetworightmostcolumnsofFig.2,oneseesthatthetransformation between 3rd and 4th columns for the gates involving AND/OR gates works exactly in this way.Forexample,theDeMorganexpression $\overline{AB}=\overline{A}+\overline{B}$isrepresented symbolicallybythe equivalence between the 3rd and 4th columns of the 2nd row ("NAND") of Fig.2.We will go over how this works, and some more examples, in class.

LogicMinimizationandKarnaughMaps

As we found above, given a truth table, it is always possible to write down a correct logic expression simply by forming an OR of the ANDs of all input variables for which the output is true (Q = 1).However, for an arbitrary truth table such a procedure could produce a very lengthy and cumbersome expression which might be needlessly inefficient to implement with gates.

ThereareseveralmethodsforsimplificationofBooleanlogicexpressions.Theprocessis usually called "logic minimization", and the goal is to form a result which is efficient.Two methods we will discuss are algebraic minimization and Karnaugh maps.For very complicated problems the former method can be done using special software analysis programs. Karnaugh maps are also limited to problems with up to 4 binary inputs.

Let'sstartwithasimpleexample.Thetablebelowgivesanarbitrarytruthtableinvolving 2 logic inputs.

Table1:Exampleofsimplearbitrarytruthtable.

A	B	Q
0	0	1
0	1	1
1	0	0
1	1	1

Therearetwooverallstategies:

1. Write down an expression directly from the truth table.Use Boolean algebra, if desired, to simplify.

2. UseKarnaughmapping("K-map").Thisisonlyapplicableifthereare $\leq$4inputs.

In our example above, we can use two different ways of writin down a result directly from thetruthtable.WecanwritedownallTRUEtermsandORtheresult.Thisgives

$$Q=\bar{A}\bar{B}+\bar{A}B+AB$$

Whilecorrect,withoutfurthersimplificationthisexpressionwouldinvolve32-inputAND gates,2inverters,and13-inputORgate.

Alternatively,onecanwritedownanexpressionforalloftheFALSEstatesofthetruth table.Thisissimplerinthiscase:

$$\overline{Q}=A\bar{B} \quad \rightarrow Q=\overline{A\bar{B}}=\bar{A}+B$$

wherethelaststepresultsfromEqn.3.Presumably,thetwoexpressionscanbefoundto be equivalent with some algebra.Certainly, the 2nd is simpler, and involves only an inverterand one 2-input OR gate.

Finally,onecantryaK-mapsolution.Thefirststepistowriteoutthetruthtablein the form below, with the input states the headings of rows and columns of a table, and the corresponding outputs within, as shown below.

Table2:K-mapoftruthtable.

$A\backslash B$	0	1
0	1	1
1	0	1

Thesteps/rulesareasfollows:

1. Formthe2-dimensionaltableasabove.Combine2inputsina“graycode”way–see 2nd example below.
2. Formgroupsof1’sandcirclethem;thegroupsarerectangularandmusthavesidesof length $2^n \times 2^m$, where$n$ and m are integers$0, 1, 2, \ldots$.
3. Thegroupscanoverlap.
4. Writedownanexpressionoftheinputsforeachgroup.
5. ORtogetherthese expressions.That’sit.
6. Groupscanwrapacrosstableedges.
7. As before,onecan alternativelyform groupsof 0’sto givea solutionfor$\bar{Q}$.
8. Thebiggerthegroupsonecanform,thebetter(simpler)theresult.
9. There are usuallymany alternative solutions,all equivalent,some better than others depending upon what one is trying to optimize.

Hereisonewayofdoingit:

$A\backslash B$	0	1
0	1	1
1	0	1

Thetwogroupswehavedrawn are $\bar{A}$and$B$.So thesolution(asbefore)is:

$$Q=\bar{A}+B$$

2.4.1 K-mapExample2

Let’susethistodeterminewhich3-bitnumbersareprime.(Thisisahomeworkproblem.) Weassumethat0,1,2are*not*prime.Wewillletourinputnumberhavedigits$a_2a_1a_0$.Here is the truth table:

HereisthecorrespondingK-mapandasolution.

Notethatwheretwoinputsarecombinedinaroworcolumnthattheirprogression followsgraycode,thatisonlyonebitchangesatatime.Thesolutionshownaboveis:

$$Q=a_1a_0+a_2a_0=a_0(a_1+a_2)$$

Table3:3-digitprimefinder.

Decimal	a_2	a_1	a_0	Q
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Table4:K-mapoftruthtable.

$a_2 \backslash a_1a_0$	00	01	11	10
0	0	0	1	0
1	0	1	1	0

K-mapExample3:FullAdder

Inthisexamplewewilloutlinehowtobuildadigital*fulladder*.Itiscalled"full"because itwillincludea"carry-in"bitanda"carry-out"bit.Thecarrybitswillallowasuccession of 1-bit full adders to be used to add binary numbers of arbitrary length.(A*half adder* includes only one carry bit.)

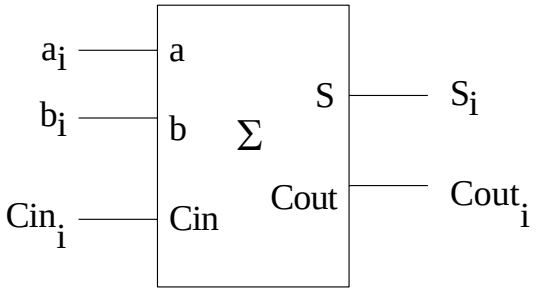


Figure7:Blockschematicoffulladder.(Wenameouradderthe"Σchip").

The scheme forthe fulladder is outlined in Fig.7.Imagine that we are adding two *n*-bit binary numbers.Let the inputs $a_i$and $b_i$be the *i*-th bits of the two numbers.The carry in bit$C\text{in}_i$represents anycarryfromthesumoftheneighboringlesssignificantbitsatposition $i-1$.That is,$C\text{in}_i=1$ if$a_{i-1}= b_{i-1}= 1$, and is 0 otherwise.The sum$S_i$at position*i*is therefore the sum ofa_i,b_i, and$C\text{in}_i$.(Note that this is an arithmetic sum,*not*a BooleanOR.) A carry for this sum sets the carry out bit, $C\text{out}_i=1$, which then can be applied to the sum of the $i+1$ bits.The truth table is given below.

$C\text{in}_i$	a_i	b_i	S_i	$C\text{out}_i$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

With$C\text{in}_i=0$,weseethattheoutputsum$S_i$isjustgivenbytheXORoperation,$a_i \oplus b_i$. And with$C\text{in}_i=1$,then$S_i=a_i \oplus b_i$.Perhapsthesimplestwaytoexpressthisrelationship is the following:

$$S_i=C\text{in}_i \oplus (a_i \oplus b_i)$$

Todeterminearelativelysimpleexpressionfor$C\text{out}_i$,wewilluseaK-map:

$C\text{in}_i \backslash a_i b_i$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Thisyields

$$Cout_i = a_i b_i + Cin_i a_i + Cin_i b_i = a_i b_i + Cin_i(a_i + b_i)$$

whichinhardwarewouldbe22-inputORgatesand22-inputANDgates.

Asstatedabove,thecarrybitsallowouraddertobeexpandedtoaddanynumberof bits.Asanexample,a4-bitaddercircuitisdepictedinFig.8.Thesumcanbe5bits,where theMSBisformedbythefinalcarryout.(Sometimesthisisreferredtoasan“overflow” bit.)

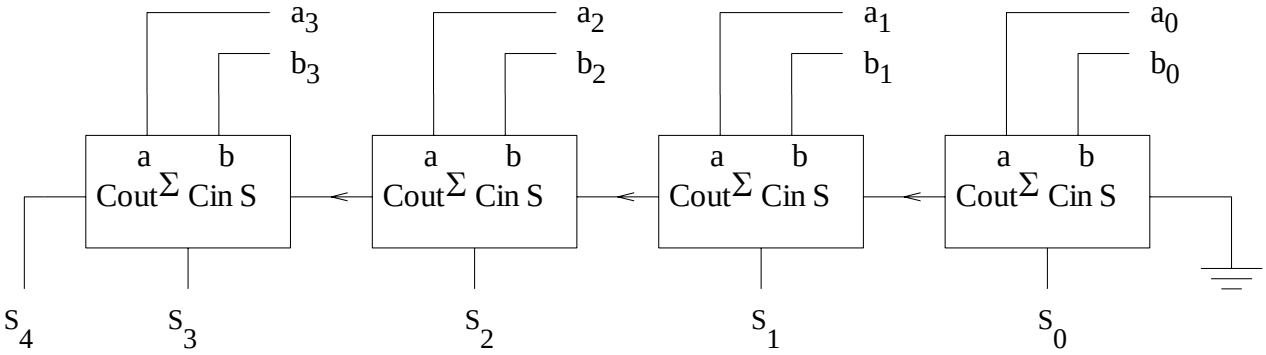


Figure8:Expansionof1-bitfulladdertomakea4-bitadder.

MakingaMultiplierfromanAdder

Inclasswewilldiscusshowtouseourfulladder(the“Σchip”)tomakeamultiplier.

Multiplexing

A multiplexer (MUX) is a device which selects one of many inputs to a single output.The selectionis done by using an input*address*.Hence,a MUX cantake many data bits andput them, one at a time, on a single output data line in a particular sequence.This is an example of transforming*parallel*data to*serial*data.A demultiplexer (DEMUX) performs the inverse operation, taking one input and sending it to one of many possible outputs. Again the output line is selected using an address.

A MUX-DEMUX pair can be used to convert data to serial form for transmission, thus reducingthenumberofrequiredtransmissionlines.Theaddressbitsaresharedbythe MUX and DEMUX at each end.If *n* data bits are to be transmitted, then after multiplexing, the number of separate lines required is $\log_2 n + 1$, compared to*n*without the conversion to serial.Hence for large*n*the saving can be substantial.In Lab 2, you will build such a system.

Multiplexers consistoftwofunctionallyseparatecomponents, a*decoder*andsomeswitches or gates.The decoder interprets the input address to select a single data bit.We use the example of a 4-bit MUX in the following section to illustrate how this works.

A4-bitMUXDesign

We wish to design a 4-bit multiplexer.The block diagram is given in Fig.9.There are 4 inputdatabitsD_0–D_3,2inputaddressbitsA_0andA_1,oneserialoutputdatabit$Q$,and

an (optional)enable bit*E*which is used forexpansion (discussed later).First we will design the decoder.

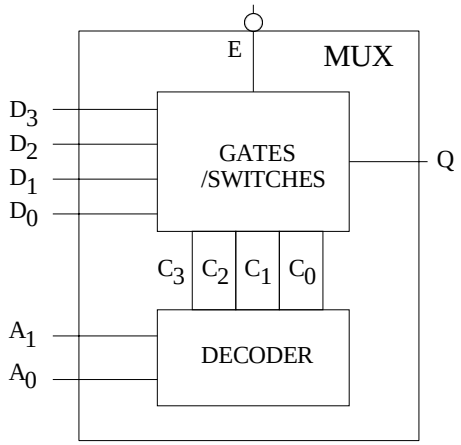


Figure9:Blockdiagramof4-bitMUX.

We need*m*address bits to specify2^{m}data bits.So in our example,we have 2 address bits.The truth table for our decoder is straightforward:

A_1	A_0	C_0	C_1	C_2	C_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Theimplementationofthetruthtablewithstandardgatesisalsostraightforward,as given in Fig.10.

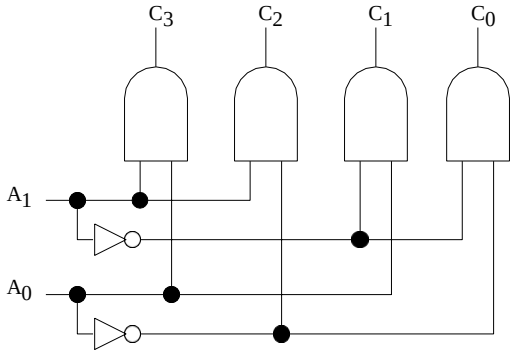


Figure10:Decoderforthe4-bitMUX.

For the “gates/switches” part of the MUX, the design depends upon whether the input data lines carry digital or analog signals.We will discuss the analog possibility later.The digitalcaseistheusualandsimplestcase.Here,thedataroutingcanbeaccomplished

simplybyforming2-inputANDsofthedecoderoutputswiththecorrespondingdatainput, andthenforminganORoftheseterms.Explicitly,

$$Q=C_0D_0+C_1D_1+C_2D_2+C_3D_3$$

Finally, ifanENABLE line*E*isincluded, it issimply ANDed withtherighthandside ofthis expression.ThiscanbeusedtoswitchtheentireMUXICoff/on,andisusefulforexpansion to more bits.as we shall see.

3 Flip-FlopsandIntroductorySequentialLogic

We now turn to digital circuits which have states which change in time, usually accordingto an external clock.The*flip-flop* is an important element of such circuits.It has the interesting property of*memory*:It can be set to a state which is retained until explicitly reset.

SimpleLatches

The following 3 figures are equivalent representations of a simple circuit.In general these are called flip-flops. Specifically, these examples are called SR ("set-reset") flip-flops, or SR latches.

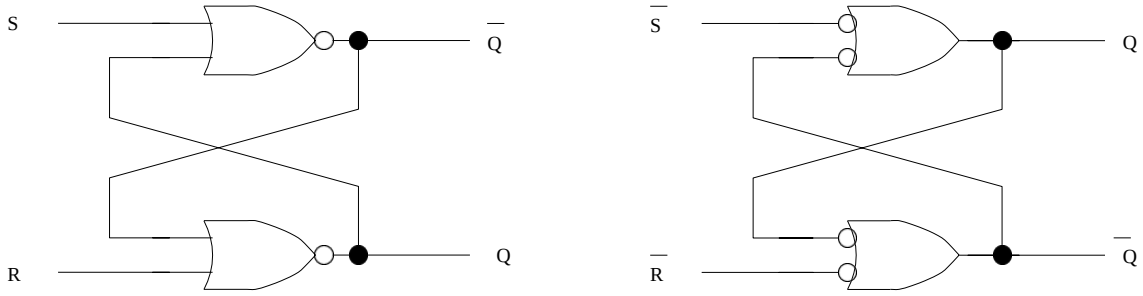


Figure11:TwoequivalentversionsofanSRflip-flop(or"SRlatch").

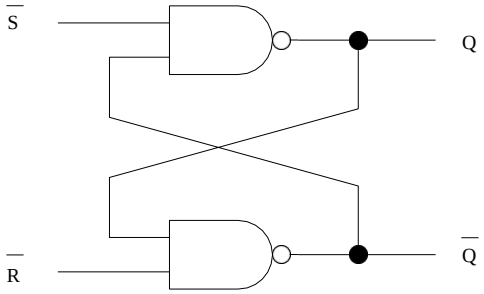


Figure12:YetanotherequivalentSRflip-flop,asusedinLab3.

ThetruthtablefortheSRlatchisgivenbelow.

S	$\overline{S}$	R	$\overline{R}$	Q	$\overline{Q}$
1	0	0	1	1	0
0	1	1	0	0	1
0	1	0	1	retainsprevious	
1	0	1	0	0	0

The state described by the last row is clearly problematic, since$Q$ and Q $\overline{\text{s}}$hould not be the same value.Thus,the $S = R = 1$ inputs should be avoided.

Fromthetruthtable,wecandevelopasequencesuchasthefollowing:

1. $R=0, S=1 \Rightarrow Q=1$ (set)
2. $R=0, S=0 \Rightarrow Q=1$ ($Q=1$ state retained: "memory")
3. $R=1, S=0 \Rightarrow Q=0$ (reset)
4. $R=0, S=0 \Rightarrow Q=0$ ($Q=0$ state retained)

In alternative language, the first operation "writes" a true state into one bit of memory. It can subsequently be "read" until it is erased by the reset operation of the third line.

Latch Example: Debounced Switch

A useful example of the simple SR flip-flop is the debounced switch, like the ones on the lab prototyping boards. The point is that any simple mechanical switch will bounce as it makes contact. Hence, an attempt to provide a simple transition from digital HIGH to LOW with a mechanical switch may result in an unintended series of transitions between the two states as the switch damps to its final position. So, for example, a digital counter connected to Q would count every bounce, rather than the single push of the button which was intended.

The debounced configuration and corresponding truth table are given below. When the switch is moved from A to B, for example, the output Q goes LOW. A bounce would result in $A = B = 1$, which is the "retain previous" state of the flip-flop. Hence, the bounces do not appear at the output Q.

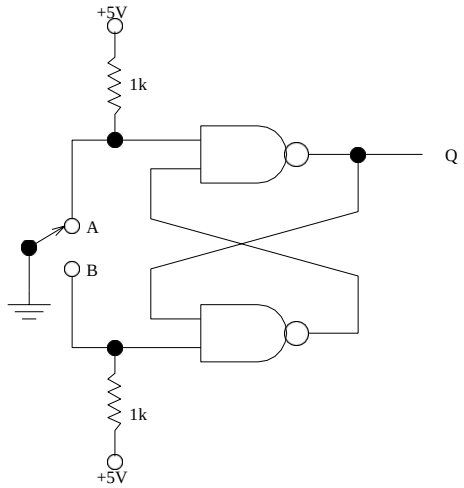


Figure 13: A debounced switch.

A	B	Q
1	0	0
0	1	1
1	1	retains previous
0	0	not allowed

ClockedFlip-flops

We will soon get used to the idea of a clock as an essential element of digital circuitry. Whenwespeakofaclocksignal,wemeanasequenceofevenlyspaceddigitalhighandlow signalsproceedingatafixedfrequency.Thatis,theclockisacontinuoussequenceofsquare wave pulses.There are a number of reasons for the importance of the clock.Clearly it is essentialfordoinganykindofcountingortimingoperation.But,itsmostimportantrole is in providing*synchronization*to the digital circuit.Each clock pulse may represent the transition to a new digital state of a so-called "state machine" (simple processor) we will soonencounter.Oraclockpulsemaycorrespondtothemovementofabitofdatafromone location in memory to another.A digital circuit coordinates these various functions by the synchronization provided by a single clock signal which is shared throughout the circuit. A moresophisticatedexampleofthisconceptistheclockofacomputer,whichwehavecome toassociatewithprocessingspeed(*e.g.*330MHzfortypicalcurrentgenerationcommercial processors.)

WecanincludeaclocksignaltooursimpleSRflip-flop,asshowninFig.14.Thetruth table, given below, follows directly from our previous SR flip-flop, except now we include alabel for the n^{th} clock pulse for the inputs and the output.This is because the inputs have no effect unless they coincide with a clock pulse.(Note that a specified clock pulse conventionally referstoaHIGHlevel.)Asindicatedinthetruthtable,theinputsS_n=R_n=0representthe flip-flopmemorystate. Significantly,onenotesthattheinterval*between*clockpulsesalso correspondstothe"retainpreviousstate"oftheflip-flop.Hencetheinformationencoded bytheonebitofflip-flopmemorycanonlybemodifiedinsynchronizationwiththeclock.

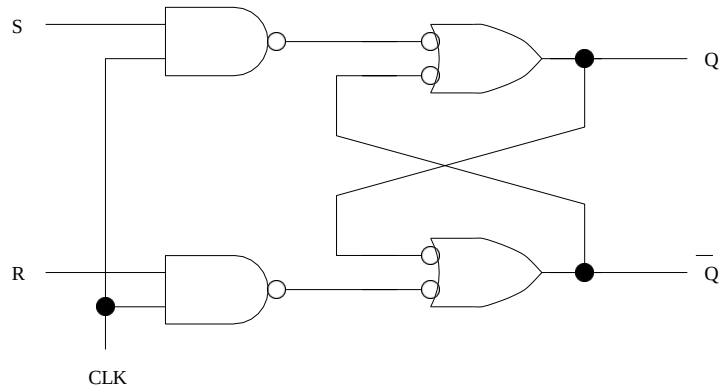


Figure14:AclockedSRflip-flop.

S_n	R_n	Q_n
1	0	1
0	1	0
0	0	Q_{n-1}
1	1	avoid

We are now set to make a subtle transition for our next version of the clocked flip-flop. The flip-flop memory is being used to retain the state between clock pulses.In fact, the statesetupbythe*S*and*R*inputscanberepresentedbyasingleinputwecall"data",or

D.This is shown in Fig.15.Note that we have explicitly eliminated the bad $S=R=1$ state with this configuration.

We can override this data input and clock sychronization scheme by including the "jam set"($\overline{S}$)and"jamreset"($\overline{R}$)inputsshowninFig.15.Thesefunctionjustasbeforewith theunclockedSRflip-flop.Notethatthese"jam"inputsgobyvariousnames.Sosometimes the set is called "preset" and reset is called "clear", for example.

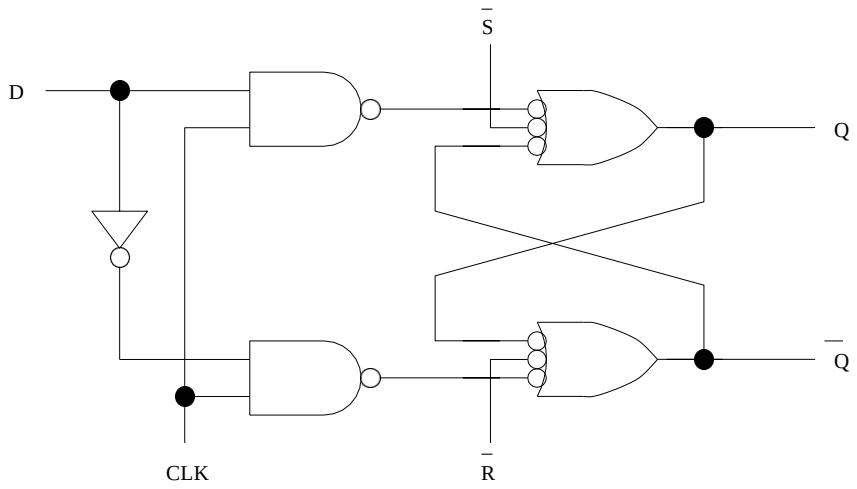


Figure15:A"D-typetransparent"flip-flopwithjamsetandreset.

Atypicaltimingdiagramforthisflip-flopisgiveninFig.16.Notethatthejamreset signal $\overline{R}$ overrides any action of the data or clock inputs.

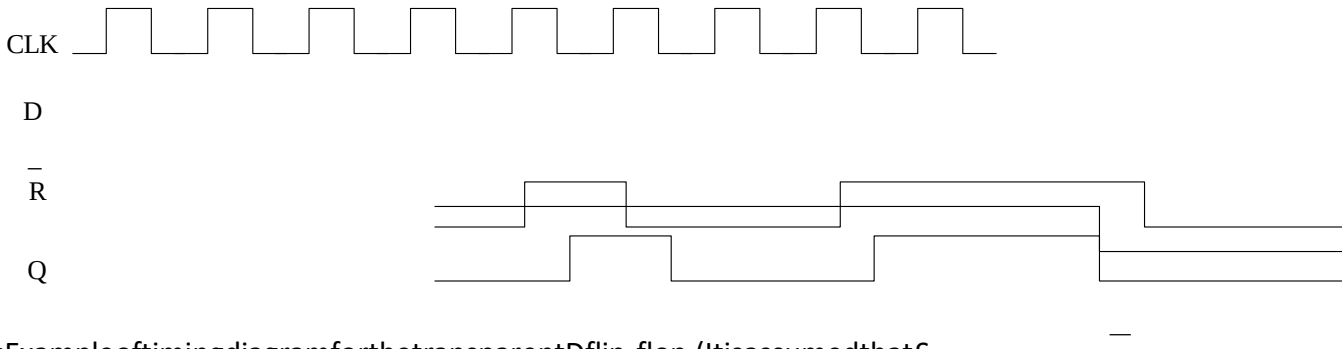


Figure16:Exampleoftimingdiagramforthetransparent Dflip-flop.(Itisassumedthat $\overline{S}$ isheldHIGHthroughout.)

EdgeTriggeredFlip-Flops

We need to make one final modification to our clocked flip-flop.Note that in the timing diagram of Fig.16 that there is quite a bit of apparent ambiguity regarding exactly when the*D*inputgetslatchedinto*Q*.Ifatransitionin*D*occurssometimeduringaclockHIGH,for example,whatwilloccur?Theanswerwilldependuponthecharacteristicsoftheparticular electronics being used.This lack of clarity is often unacceptable.As a point of terminology,

the clocked flip-flop of Fig.15 is called a *transparent D-type* flip-flop or latch.(An example in TTL isthe7475IC.)

Thesolutiontothisisthe*edge-triggered*flip-flop.Wewilldiscusshowthisworksforone example in class. It is also discussed some in the text.Triggering on a clock rising or falling edgeissimilarinallrespectstowhatwehavediscussed,exceptthatitrequires2–3coupled SR-type flip-flops, rather than just one clocked SR flip-flop.The most common type is the *positive-edge triggered D-type*flip-flop.This latches the*D*input upon the clock transition from LOW to HIGH. An example of this in TTLis the 7474 IC. It is also common to employ a *negative-edgetriggeredD-type*flip-flop,whichlatchesthe*D*inputupontheclocktransition from HIGH to LOW.

The symbols used for these three D-type flip-flops are depicted in Fig.17.Note thatthe small triangle at the clock input depicts positive-edge triggering, and with an inversion symbol represents negative-edge triggered.TheJKtype of flip-flop is a slightlier fancier version of the D-type which we will discuss briefly later.Not shown in the figure are the jam set and reset inputs, which are typically included in the flip-flop ICpackages.In timing diagrams,theclocksforedge-triggereddevicesareindicatedbyarrows,asshowninFig.18.

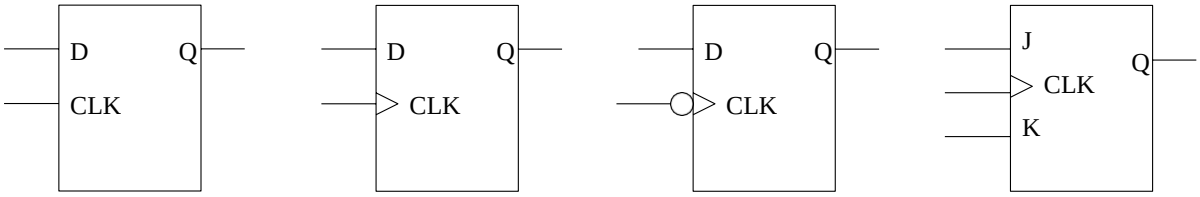


Figure 17:Symbols for D-type and JKflip-flops.Left to right:transparent D-type, positive-edgetriggeredD-type,negative-edgetriggeredD-type,andpositive-edgetriggeredJK-type.

Figure18:Clocksintimingdiagramsforpositive-edgetriggered(left)andnegative-edge triggered (right) devices.

Foredge-triggereddevices,theambiguityregardinglatchtimingisreducedsignificantly. But at high clock frequency it will become an issue again.Typically, the requirements areas follows:

- Thedatainputmustbeheldforatimet_{setup}beforetheclockedge.Typically,$t_{\text{setup}} \approx$ 20nsorless.
- ForsomeICs,thedatamustbeheldforashorttimet_{hold}aftertheclockedge.Typically $t_{\text{hold}} \approx$3ns,butiszeroformostnewerICs.
- The output*Q*appears after a short propagation delay$t_{\text{prop}}$of the signal through the gates of the IC. Typically,$t_{\text{prop}} \approx$ 10 ns.

From these considerations we see that for clocks of frequency much less than $\sim 1/(10\text{ns}) = 100$ MHz, these issues will be unimportant, and we can effectively consider the transitions to occur instantaneously in our timing diagrams.

4 Counters,Registers,andStateMachines

We can now apply what we know about basic flip-flops circuit elements to develop new functions:countersand registers.Indoingso,wewillintroducethe“statemachine”,a clockedsequential“processor”.Wewillexaminethislattertopicinmoredetailinafew weeks.

DividebyTwoCounter

The edge-triggered D-type flip-flops which we introduced in the previous Section are quite useful and versatile building blocks of sequential logic.A simple application is the divide-by-2 countershowninFig.19,alongwiththecorrespondingtimingdiagram.

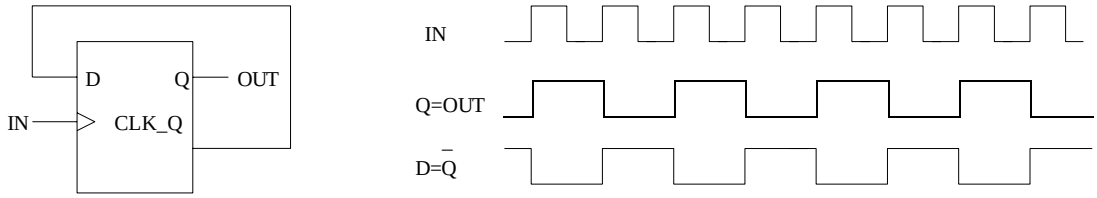
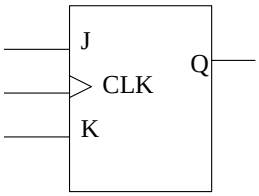


Figure19:Positiveedge-triggeredD-typeflip-flopconnectedasdivide-by-2counter.

UsingtheJKFlip-flop

In Lab 4 you will build an asynchronous (ripple) counter using a sequence of cascadedJK flip-flops, rather than the D-type which is used in our discussion below.For reference, the JKtruth table isgiven in Fig.20.Note that thereis no fundamentaladvantage to using the JKinstead of the D-type, only that the JK,with the additional J=K=1 state, makes the divide-by-2 function slightly simpler to implement.



J_n	K_n	Q_n
1	0	1
0	1	0
0	0	Q_{n-1}
1	1	$\overline{Q_{n-1}}$

Figure20:TheJKFlip-flop.

AsynchronousCounter

Flip-flops can be connected in series,as shown in Fig.21.The resulting outputs are givenin Fig.22.(Note that labels in these two figures correspond when $A \equiv 2^0$, $B \equiv 2^1$, $C \equiv 2^2$, and$D \equiv 2^3$.Hence, this is a 4-bit counter, with maximum count $2^4 - 1 = 15$. Itis clearly possible to expand such a counter to an indefinite number of bits.

While asynchronous counters are easy to assemble, they have serious drawbacks for some applications.In particular, the input must propogate through the entire chain of flip-flops before the correct result is achieved.Eventually, at high input rate, the two ends of the chain, representingtheLSBandMSB,canbeprocessingdifferentinputpulsesentirely.(Perhaps in lab you can see this effect on the oscilloscope with a very high input frequency.)Thesolution to this is the *synchronous counter*, which we will discuss below as an example of astate machine.

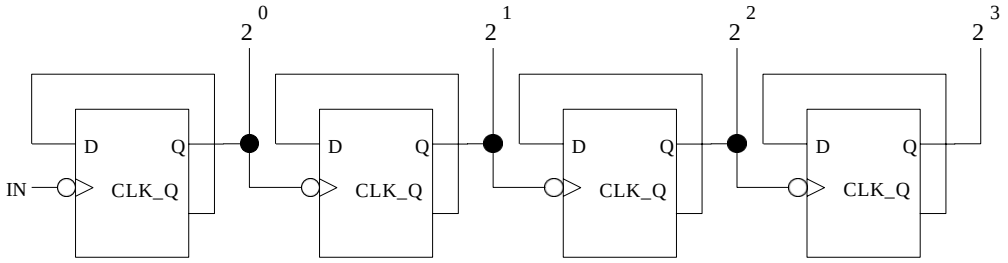


Figure 21:Asynchronous(“ripple”)countermadefrom cascadedD-typeflip-flops.

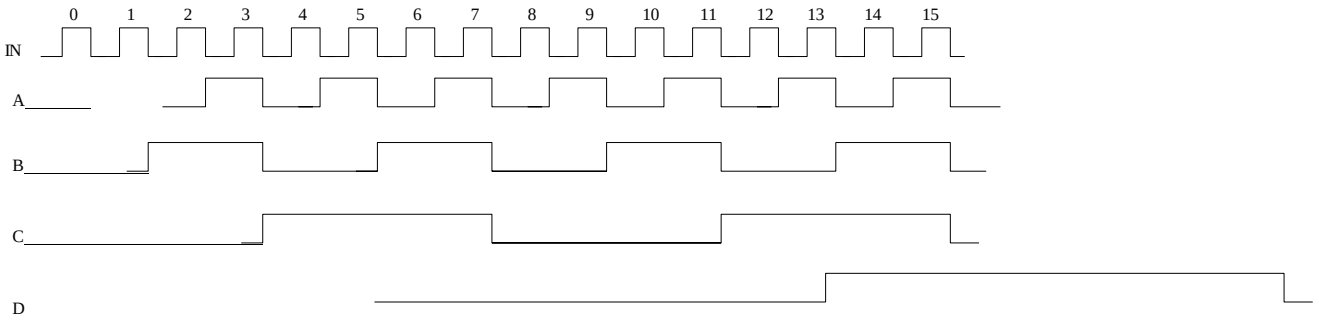


Figure22:Waveformsgeneratedbytheripplecounter.

Registers

BasicRegister

Thefigurebelowrepresentsa4-bitmemory.Wecanthinkofitas4individualD-type flip-flops.The important point about a data register of this type is that all of the inputs are latched into memory synchronously by a single clock cycle.

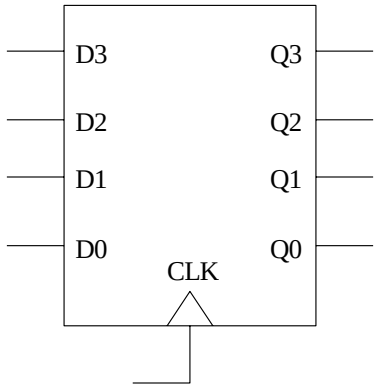


Figure23:4-bitdataregister.

ShiftRegisters

Thefigurebelowisanexampleofa4-bit*shiftregister*.Theseconfigurationsarequite useful,particularlyfortransformingserialdatatoparallel,andparalleltoserial.Inthe circuitbelow,apulseappearingat"serialin"wouldbeshiftedfromtheoutputofoneflip-floptothenextoneachclockcycle.Henceaserialbitpatternattheinput(4bitslongin this example) would appear as 4 parallel bits in the outputs Q_0–Q_3 after 4 clock cycles.This represents the serial-to-parallel case.

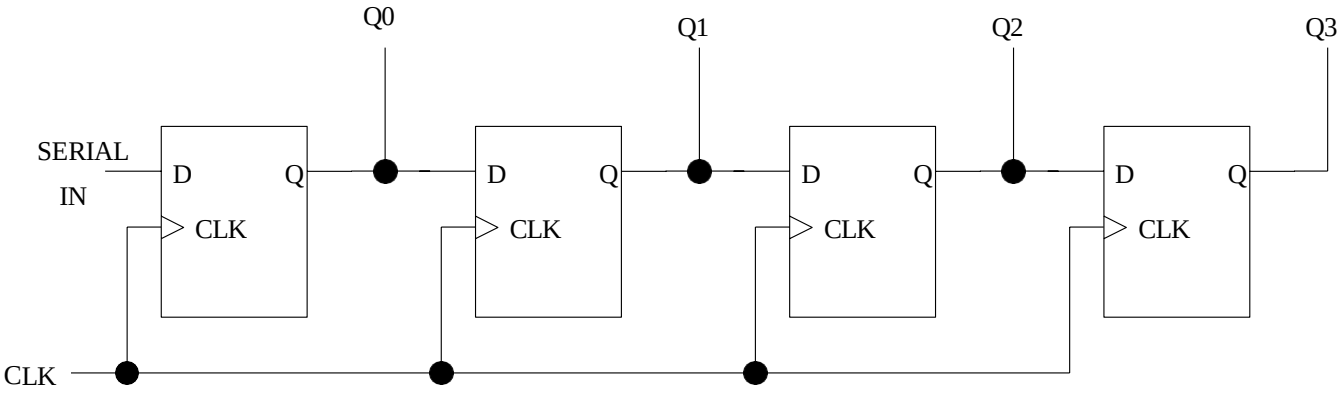


Figure24:4-bitshiftregister.

Wewilldiscussseveralexamplesofshiftregistersafewlectureshence.

5 Analog/DigitalConversion

In this section we discuss the important topic of analog to digital conversion (often written A/D), and digital to analog conversion (D/A). On one hand, most electrical measurements areintrinsically analog.Totake advantageofthe greatcapabilities available fordigitaldata storage, processing,and computation,on the other hand,requires the conversionof analog todigital.Hence,analogtodigital(A/D)conversiontechniqueshavebecomeextremely important.A great deal of technical effort has gone into producing A/D converters (ADCs) whichare fast,accurate,andcheap.D/A converters(DACs)arealsoveryimportant.For example, video monitors convert digital information generated by computers to analog signals whichareusedtodirecttheelectronbeamataspecifiedportionofthemonitorscreen.DACs areconceptually simpler thanADCs, althoughitisdiffcultinpracticetobuildapreciseDAC. We will discuss D/A conversion before A/D. But first we go over some underlying ideas.

A/DResolution

First of all we should keep in mind that there are several different schemes for encoding analog information as bits, depending upon what is required by a particular application. One extreme is that of encoding the complete analog signal in as much detail as possible. Forexample,amusicalinstrumentproducesananalogsignalwhichisreadilyconvertedtoan analog electrical signal using a microphone. If this is to be recorded digitally, one naturally wouldchoosetodigitizeenoughinformationsothatwhentherecordingisplayedback,the resulting audio is not perceived to be significantly different from the original.In this case the analog signal is a voltage which varies with time, $V(t)$.

Atanytimet_0, $V(t_0)$can besampled andconverted todigital.Theanalogsignal must be sampled for a finite time, called the *sampling time*, Δt.One may guess that it is necessary to sample the analog signal continuously, with no gaps between consecutive samples.This turns out to be overkill.The*Nyquist Theorem*states that if the maximum frequency of inerestintheanaloginputisf_{max},thenperfectreproductiononlyrequiresthatthesampling frequencyf_{samp}beslightlygreaterthantwicef_{max}.Thatis,

$$f_{\text{samp}} > 2f_{\text{max}}$$

For example, for audio signals the maximum frequency of interest is usually 20 kHz.In this case the input analog must be sampled at a little over 40 kHz.In fact, 44 kHz is typically used.

Alternatively, it might not be of interest to represent the entire analog input digitally. Perhaps only one feature of the analog signal is useful.One example is “peak sensing,” whereonesamplesanddigitizestheinputonlyattheinstantwhereaninstrument’s output achieves a maximum analog output. Or one may average (“integrate”) an input signal over some predefined time, retaining only the average value to be digitized.

For any of these sampling schemes, there remains the issue of how many bits are to be usedtodescribethesampledsignal$V(t_0)$.ThisisthequestionofA/D*resolution*.Weneeda standarddefinitionofresolution.Let’ssay,forexample,thatwechoosetodigitizetheinput using 12 bits.This means that we will try to match our analog input to 1 of 2^{12}=4096 possible levels.This is generally done by ascribing a number from 0 to 4095. So, assuming our ADC works correctly, the digital estimate of the analog input can, at worst, be wrongby the range of the LSB. On average, the error is half of this.This defines the resolution. Therefore,forour12-bitexample,theresolutionis$1/(2\cdot 4096)$,oralittleworsethan0.01%.

D/AConversion

The basic element of a DAC is the simplest analog divider:the resistor.First, we need to review the two important properties of an operational amplifier ("op-amp") connected in the inverting configuration.This is shown in Fig.25.The two important properties are

1. The"−"inputiseffectivelyatground.("virtualground")
2. Thevoltagegainis$G \equiv V_{out}/V_{in} = -R_2/R_1$.Anequivalentstatementisthatfora current at the−input of$I_{in} = V_{in}/R_1$, the output voltage is$V_{out} = GV_{in} = -R_2I = -V_{in}R_2/R_1$.Sometimes this is written in the form $V_{out} = gI_{in}$, where g is the *transconductance*, and $g = -R_2$in this case.

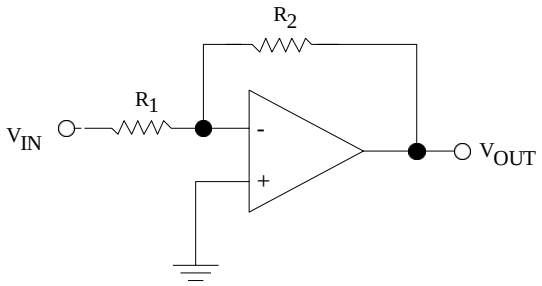


Figure25:Invertingop-ampconfiguration.

ThebasicideaofmostDACsisthenmadeclearbythe4-bitexampleillustratedinFig. 26.The input 4-bit digital signal defines the position of the switches labelleda_0–a_3.AHIGHinputbitwouldcorrespondtoaswitchconnectedto1.0V,whereasaLOWconnectsto ground.Theconfigurationinthefigurerepresents abinaryinputof1010,or10_{10}.Sincethe virtual ground keeps the op-amp input at ground, then for a switch connected to ground, there can be no current flow.However, for switches connected to1.0 V, the current presented totheop-ampwillbe1.0Vdividedbytheresistanceofthatleg.AllIegswithHIGHswitches thencontributesomecurrent.Withthebinaryprogressionofresistancevaluesshowninthe figure, the desired result is obtained.So for the example shown, the total current to the op-ampis$I = 1.0/R + 1.0/(4R) = 5/(4R)$.Theoutputvoltageis

$$V_{out} = -RI = 5/4 = 1.25\text{V}$$

When all input bitsare HIGH ($1111 = 15_{10}$), we find$V_{out} = 15/8$V. A simple check of our schemeshowsthat

$$(5/4)/(15/8) = 2/3 = 10/15 = 1010/1111$$

asexpected.

TheR-2RLadder

This represents aratherminor point, althoughitisaninteresting idea.The "R-2Rladder"is ofpracticalinterestbecauseitusesonlytworesistorvalues.Sinceitisdifficulttoaccurately fabricate resistors of arbitrary resistance, this is beneficial.The two resistances of the R-2R

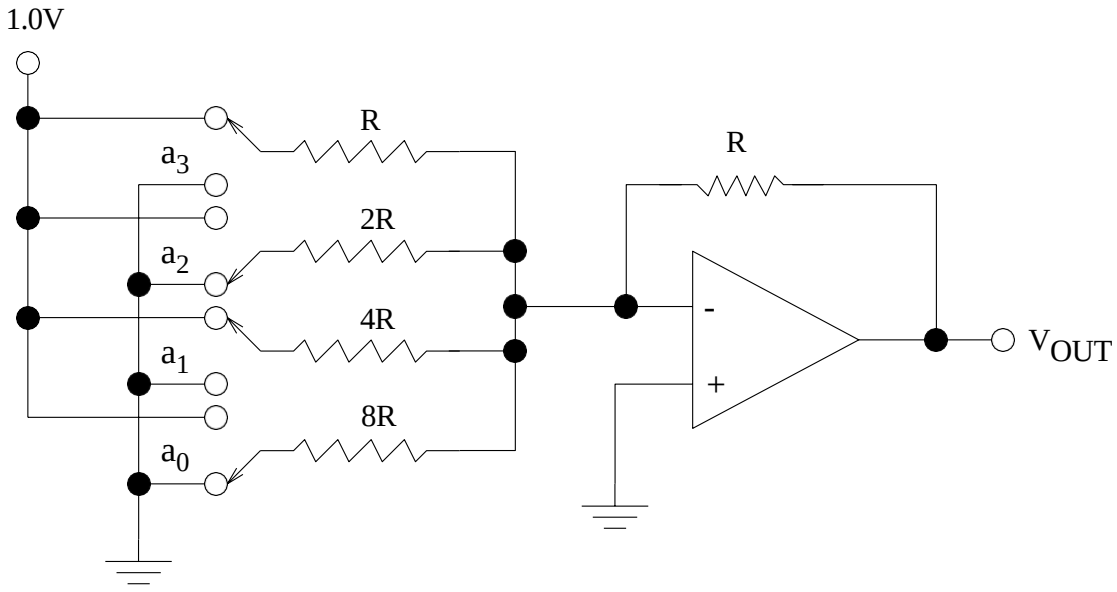


Figure26:Example4-bitDACscheme.

aretobecontrastedwiththeschemerepresentedbythecircuitofFig.26,whichemploys asmanyresistancevaluesastherearebits.TheideabehindtheR-2Rladderhingeson noticingthepatternofequivalencesrepresentedbyFig.27,whichcanbeusedtoreplicate anarbitrarilylongladder,andhencehandleinarbitrarynumberofbits.

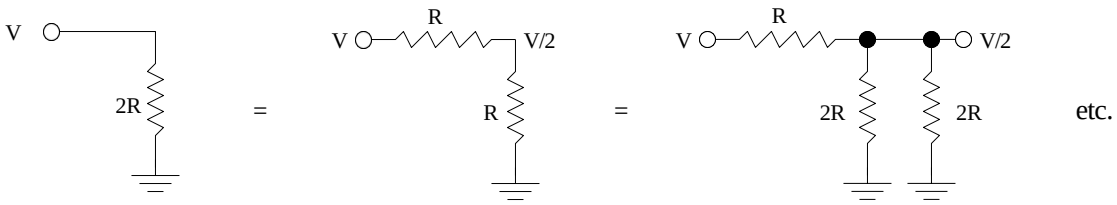


Figure27:PrincipleoftheR-2Rladder. Therightmost2Rresistorcanbeindefinitely relicated with this equivalent circuit.

A/DConversion

ADCsfallinto3generaltypesoftechnique:

(1) parallelencoding(flash): fast;limitedaccuracy

(2) successiveapprox.(feedback):med.fast;goodaccuracy

(3) singleordoubleslope: slow;bestpotentialaccuracy

Allofthesetechniquesuseadeviceknownasa*comparator*.Thiswasdiscussedin431/531 andinthetextChapters4and9.Here,wewillnotdiscusshowcomparatorswork,butwedo need toknow whatthey do. There aremany makes ofcomparators. Wewill usethe model LM311inlab.Figure28showsacomparatorschematically.Internally,thecomparatorcan bethoughtofasafast,veryhigh-gaindifferentialamplifier("A")withinputs"+"and"−." We can put a "threshold voltage" at the "−" input.Call itV_{th}.The circuit inputV_{in} is connectedtothe"+"input.When$V_{in}>V_{th}$,thecomparatoramplifiesthisdifferenceuntil theoutputreachesitslargestpossiblevalue,whichisdeterminedbytheconnectionthrough thepull-upresistor.Intheconfigurationshown here, aswellasinLab5,the$\sim$1kΩ pull-up resistor is connectedto +5 V. (Note that while +5 V is convenientfor many digital circuits,it is possible to use other values, such as +12 V.) When$V_{in}<V_{th}$, the output swings the other way.This level is usually determined by a connection to one of the comparator pins. Here, it is ground.

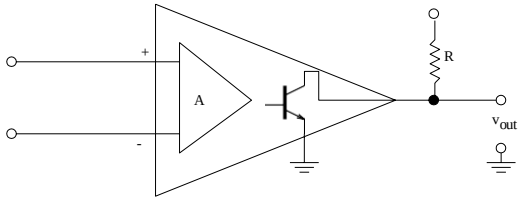


Figure28:Comparator.

Hence,thecomparatorrepresentsaone-bitADC.Whentheanaloginputexceedsthe pre-defined threshold, the output goes to digital HIGH, and when the input is less that the threshold, the output goes to digital LOW.

FlashADCs

Inthisscheme,theinputisfannedoutinparalleltoseveralcomparatorswithmonotonically increasing thresholds.The pattern of comparator outputs is then analyzed by some combi-nationallogic(*i.e.*gates)todeterminetheoutput.Thistechniqueiscalled *flash*(or*parallel*) *encoding*.WeexemplifytheflashADCschemewiththe2-bitADCshowninFig.29.With n=2bits,weneedtodefine2^n=4possiblestates.These statesrepresent4separateinter-vals.Theanaloginputwillfallintooneoftheseintervals,andwewillencodethisassignment with the 2 bits.Defining the boundaries of 2^nintervals requires 2^n-1 comparators, with the threshold of each comparator set to the appropriate boundary voltage.

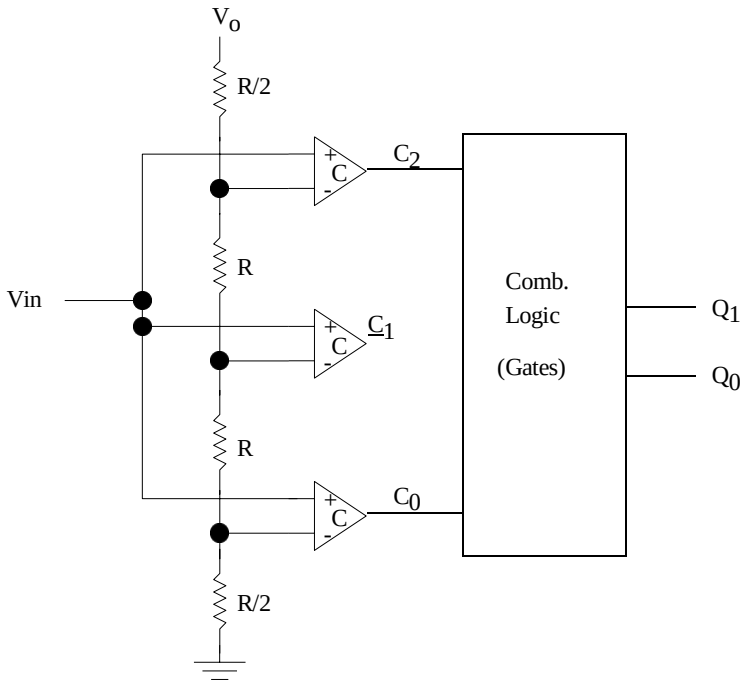


Figure29:Schematicofa2-bitflashADC.

Let's go through a concrete example.Assume that our FADC circuit is designed to handle analogvoltageinputsignalsintherange-0.5to3.5V.Thus,wehavea4-volttotalinput range,witheachintervalspanning1.0V.Therefore,eachstatewillhaveamaximumerror, orresolution,ofhalftheinterval,or0.5V.(Thisis$4.0/(2\cdot 2^n)$,aswesaidpreviouslyinour definition of resolution.)So an input which is in the range 2.5–3.5 V will give a HIGH output onlytocomparatoroutputC_2,andourdigitalestimationwillcorrespondto3.0V.Hence, thethresholdfortheuppercomparator(its"$-$"input)shouldbesetat2.5V.Similarlyfor the remaining comparators we work out the values which are given in the table below, where V_{est} isthedigitalestimate which corresponds toeachstate.

V_{in}range	Comparator	Threshold	V_{est}	$C_2C_1C_0$	Q_1Q_0
2.5–3.5V	C_2	2.5V	3.0V	111	11
1.5–2.5V	C_1	1.5V	2.0V	110	10
0.5–1.5V	C_0	0.5V	1.0V	100	01
$-$0.5–0.5V	–	–	0.0V	000	00

UsingOhm's andKirchoff'sLaws,wearriveat theresistanceratios showninFig.29in order to achieve the desired comparator thresholds.All that remains is to determine the gate logictoconvertthepatternofcomparatoroutputstoa2-bitdigitaloutput.Generalizing from the above, we see that we have agreement with our previous statements:For an n-bit ADC,werequire2^n-1comparators,andtheresolutionis$\Delta V/2^{n+1}$,whereΔVisthefull range of analog input.

SuccessiveApproximationADCs

This technique is illustrated by Fig.30, which is also the one given for Lab 5.It uses adigital feedback loop which iterates once on successive clock cycles.The function of the *successive approximation register*, or SAR, is to make a digital estimate of the analog input based on the 1-bit output of the comparator.The current SAR estimate is then converted back to analog by the DAC and compared with theinput.The cycle repeats until the "best" estimateisachieved.Whenthatoccurs,thispresentbestestimateislatchedintotheoutput register (written into memory).By far the most common algorithm employed by SARs is the *binary search algorithm*.This is the one used by the SAR in Lab 5, and is illustrated in the example in the next secion.

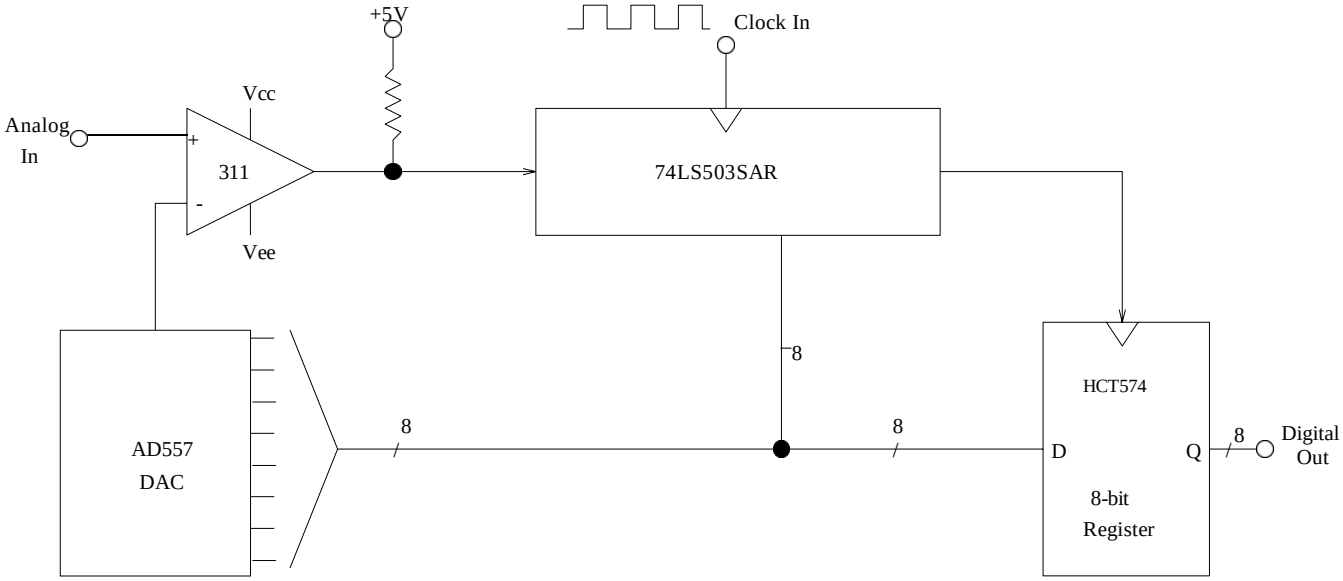


Figure30:Schemefor8-bitsuccessive approximation,orfeedback,ADC.

BinarySearchExample

In this example we will see the binary search algorithm in action.The binary search algorithm canbesummarizedwiththefollowingwords:*Gotothemidpointoftheremainingnon-excludedrange.*Inourexample,weassumean8-bitADCwithanexpectedinputvoltage rangeof0to10V.So,naturallywechoosethedigitaloutputtobe00000000_2= 0when theinputis0V,and11111111_2=255whentheinputis10V.Hence,theLSBrepresentsa voltage step ΔV= 10/255 =39.22 mV.

Let the input voltage be some arbitrary value, 7.09 V. Now let's see how the algorithm works.Translating the words for the algorithm, written above, to what the SAR actually doesisstraightforward.TheSARalwaysoutputsoneoftworesults,dependinguponwhether the output from the comparator was TRUE or FALSE. More precisely,the comparator will issueaHIGHifthecurrentestimateistoosmallcomparedtotheactualinput,oraLOWifitistoo big.The SAR then does the following:

1. Ifestimatetoosmall,add1toMSB−(n+1);or

2. If estimate too big, subtract 1 from MSB$-$(n+1) .

where n is the current clock cycle (see table).

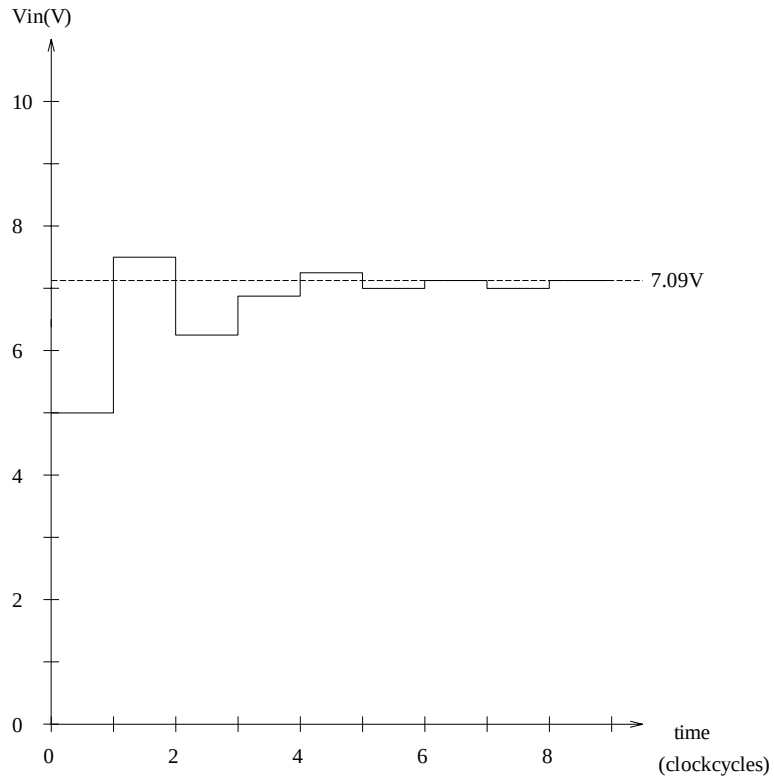


Figure 31:Binary search algorithm in action.The analog input is 7.09 V. The digital estimateforeachclockcycleisrepresentedbythesolidline,andcorrespondstothevalueof V_{est} in the table below.

ClockCycle,n	SARBits	SARBitSum	V_{est}(V)	comp.decision
0	01111111	127	4.98	toosmall
1	10111111	127+64=191	7.49	toobig
2	10011111	191$-$32=159	6.24	toosmall
3	10101111	159+16=175	6.86	toosmall
4	10110111	175+8=183	7.18	toobig
5	10110011	183$-$4=179	7.02	toosmall
6	10110101	179+2=181	7.10	toobig
7	10110100	181$-$1=180	7.06	toosmall

Thebinarysearchalgorithmisguaranteedtofindthebestpossibleestimateinanumber of clock cycles equal to the number of bits.In the example above, the best estimate was actually determined on the seventh clock cycle (n=6).But since the input value was between the digital estimates 180 and 181, there was no way for the ADC to determine which estimate was closer to the actual input value (without adding one more bit).Since theinputcanfallanywherewithin180and181withequallikelihood,thereshouldbeno

bias introduced with this method due to systematically choosing a digitalestimate which is too small or too big.This is the desired outcome.

The binary-search algorithm is fast and efficient, and also has the advantage that it completes its estimation in a well determined number of clock cycles.Hence, the final digital resultcanalwaysbelatchedafter*n*clockcycles,where*n*isthenumberofbits.(Many ADCsactuallywaitoneadditionalclockcycleinordertoguaranteethatbitshavesettled, arelatchedproperly,andareresetforthenextinput.

Single/DualSlopeADCs

Thesetechniques areslowerthanflashorsuccessive approximation,butinprinciplecanbe quite accurate.The improved accuracy is for two reasons, because time, which is robustly measured using digital techniques, is used as the measured quantity, and because there is some immunity to noise pickup, especially for the dual slope case.

The single slope technique is illustrated in Fig.32, which is taken from Figure 9.54 ofthe text.The device near the input and the capacitor is an FETtransistor which is used asa switch.When the input to the FET gate, which comes from the *Q* output of the D-type flip-flop, is LOW, then theFETis switched off, and it draws no current.However, when*Q* goes HIGH, the FETpulls the +input of the comparator to ground, and holds it there.The boxmarked"osc"representsatypicaldigitalclock.Thearrowwithinthecircleconnectedto +V_{cc}isthesymbol fora"current source", which meansthatitsoutputisaconstant current, regardless of the impedance at its output (within reasonable bounds).

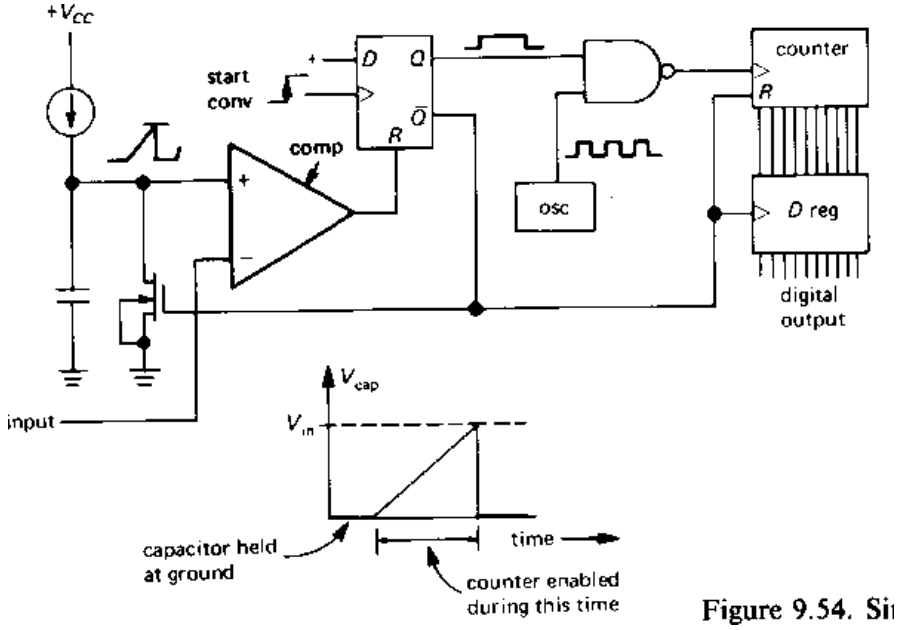


Figure 9.54. Si

Figure32:Schemeforsingle-slopeADC,fromtext.

Theprocessbeginswhenarising-edgesignalissenttotheflip-flop,forexamplefrom a debounced switch.Since the*D*input is HIGH, then*Q*goes HIGH. Hence the counter, no longerbeingheldatresetbytheflip-flop,beginscounting.AtthesametimetheFETis

switchedoffandasignalissenttothe− inputofthecomparator.Nowwemustanalyze the nature of this signal.

ThevoltageacrossacapacitorV_{cap},isrelatedtoitsstoredchargeby$V_{cap}=Q/C$,where $C$is the capacitance.Differentiating gives$dV_C/dt=I/C$.Now, because of the current source, the right-hand side of this equation is a constant.Finally, since one side of the capacitorisatground,thenthecomparator+inputisjustV_{cap}.Hence,wecanintegrate ourexpression overatimeintervalΔttogive:

$$V_+=V_{cap}=(I/C)\Delta t$$

SinceI/Cisaknownconstant,thisequationallowsonetoconverttheV_+inputtoatime $\Delta t$to be measured by the counter.This linear relation between$V_+(=V_{cap})$ and $\Delta t$is illustrated in the figure.The counter stops (is reset) and its final count stored in the register whenV_+becomesequaltoV_{in},thuschangingthestateofthecomparator.Thisalsoresets theflip-flop,thusreturning thecircuit toitsinitialstate.

The dual-slope ADCs work similarly, but with a two-step process.First, a capacitoris charged for a fixed time$\tau$with a current source whose current is proportional toV_{in},$I=\alpha V_{in}$, where$\alpha$is the constant of proportionality.Hence,V_{cap} is proportional toτ: $V_{cap}=\alpha V_{in}\tau/C$.ThecapacitoristhendischargedatconstantcurrentI'andthetime$\Delta t$to dosoismeasured.Therefore,

$$\Delta t=[C/I'][\alpha\tau/C]V_{in}=\beta V_{in}$$

where$\beta=\alpha\tau/I'$isaknownconstant.

This technique has two advantages compared with single-slope.First, we see from the equationabovethattheresultisindependentof$C$.Thisisgood,asprecisecapacitance values are difficult to fabricate.Second, the integration of the input voltage in the charge-up stepallows60Hzpickupnoise(orotherperiodicnoise)tobeaveragedtozero.

6 Counters,Registers,andStateMachinesII

The general scheme for a*state machine*is given in Fig.33.It has*n*bits of memory,*k* inputs, and*m*outputs.It consists of a synchronous data register (lower box) which stores themachine's *present state*. Aset ofseparate flip-flopscanbeused forthis,aslongasthey are clocked synchronously.The logic in the upper box acts upon the current state, plus any inputs, toproducethemachine's next state,aswellasanyoutputs.Uponeach pulse ofthe clock input CLK, the machine is moved from the present state to the next state.We will introduce this topic using counters asexamples, then moving to more general applications. Wewillsee,infact,thatthestatemachineprepresentsasimple*processor*:Theinputscanbe generalizedtobetheprocessorprogramandthelogicmightbereplacedbyarandom-access memory (RAM).

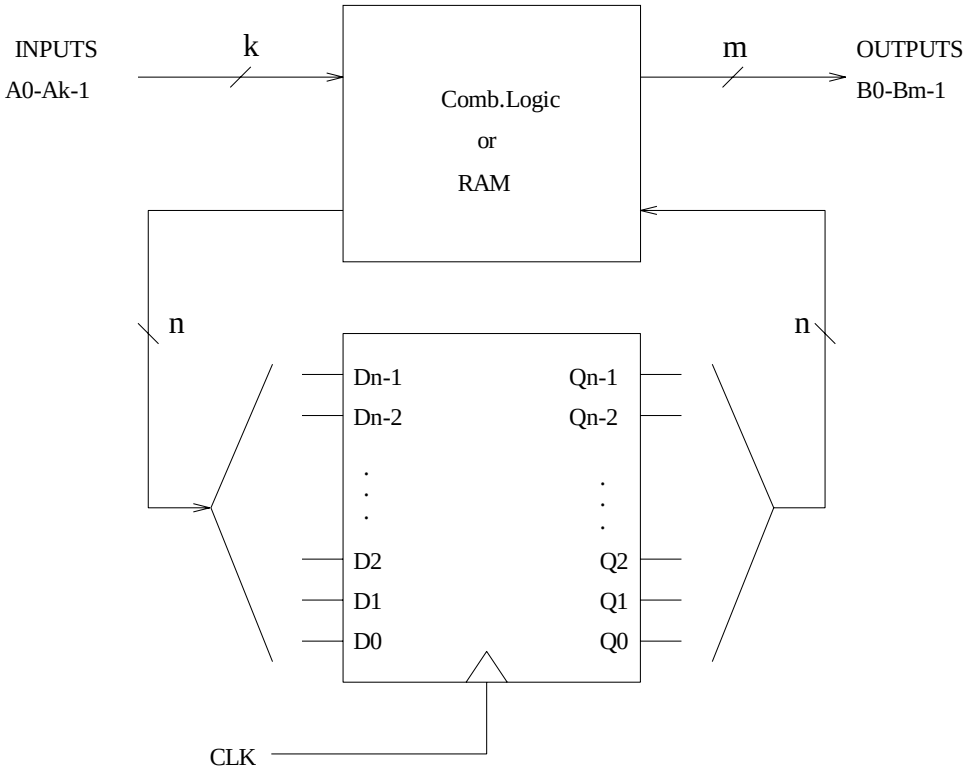


Figure33:Generalschemeforstatemachine.

Thestrategyforapplyingthisschemetoagivenproblemconsistsofthefollowing:

1. Identifythenumberofrequiredstates,*I*.Thenumberofbitsofmemory(*e.g.*number of flip-flops) required to specify the *m* states is at minimum $n = \log_2(m)$.
2. Makea*state diagram*whichshowsallstates,inputs,andoutputs.
3. Makeatruthtableforthelogicsection.Thetablewillhave*n*+*k*inputsand*n*+*m* outputs.
4. Implementthe truthtableusing our combinational logictechniques.

StateMachineIntroduction:SynchronousCounters

Countersimplementedasstatemachinesarealways*synchronous*,thatistheentirecircuitis in phase with the clock.Recall that our previous "ripple" counters were asynchronous — logic was initiated at different times throughout the circuit.Synchronous systems are essential wheneverasequentialsystemrequiresmorethanaverymodestspeedorcomplexity.

Example:Up/down2-bitSynchronousCounter

A2-bitcounterrequires4states,witheachstatecorrespondingtooneof the4possible 2-bit numbers.Hence, 2 bits of memory are required.We will use 2 flip-flops (D-type) to implement this.The state diagram is given in Fig.34.Each circle represents one of the states, and thearrows represent a clock pulse which offers atransition to another state(or possibly toremainatthepresent state).The4statesarespecified bythe2bitsofmemory: A =00,B= 01,C=10,D=11.Notethatwearefreetolabelthestatesaswechoose, as long as they are uniquely specified.However, in this case it is easiest to choose labels whichcorrespondtoourdesiredoutputs,thatisthe2-bitbinarysequence00,01,10,and 11.Hence, these labels are equivalent to our desired outputs, call them Q_1Q_0, which are available at each state.(Note that the lettered labels *A*–*D* are superfluous; they could be omitted.)

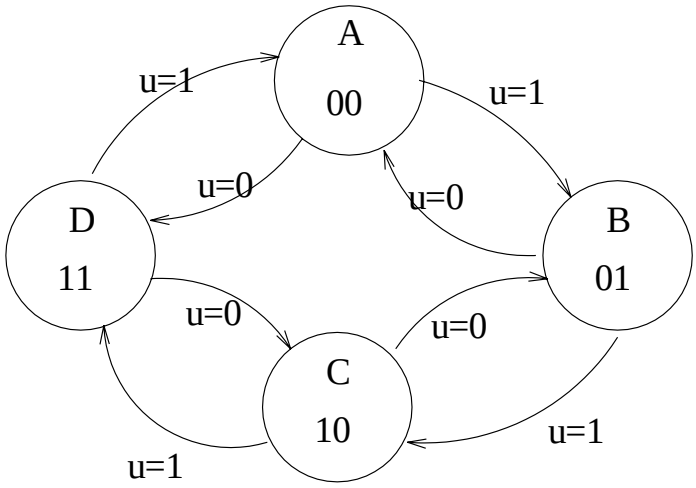


Figure34:Statediagramfor2-bitup/downsynchronouscounter.

Our processor has one input bit*u*, which programs the up-counting (*u*=1) or down-counting (*u* =0) functions.In this case, the state machine outputs are the two bits of the present state,Q_1Q_0, so we do not reproduce them in our truth table.The truth table for the logic is below.

	PresentState		NextState	
u		Q_1Q_0		D_1D_0
1	A	00	B	01
1	B	01	C	10
1	C	10	D	11
1	D	11	A	00
0	A	00	D	11
0	D	11	C	10
0	C	10	B	01
0	B	01	A	00

We can now envokethe logic as usual.We have 2 “outputs”,D_0andD_1, which are tobe evaluated separately.From the truth table, or using a K-map, we see that

$$D_1=u\oplus(Q_0\oplus Q_1); \qquad D_0=\overline{Q_0}$$

Example:Divide-by-Three SynchronousCounter

Ourstatemachineissupposedtocountinputpulses(inputattheCLK)andsetanoutputbit HIGH on every 3rd input pulse.Note that this could represent either a 2-bit (total) counter, or more generally the 2 least-significant bits of a many-bit counter.

We require 3 states, therefore we need 2 bits of memory (2 D-type flip-flops, for example). These2flip-flopscandescribe4states,sowewillhaveone“unused”state.Astatediagram is shown in Fig.35, with one way of labelling the states and output bit (called$p$) given.

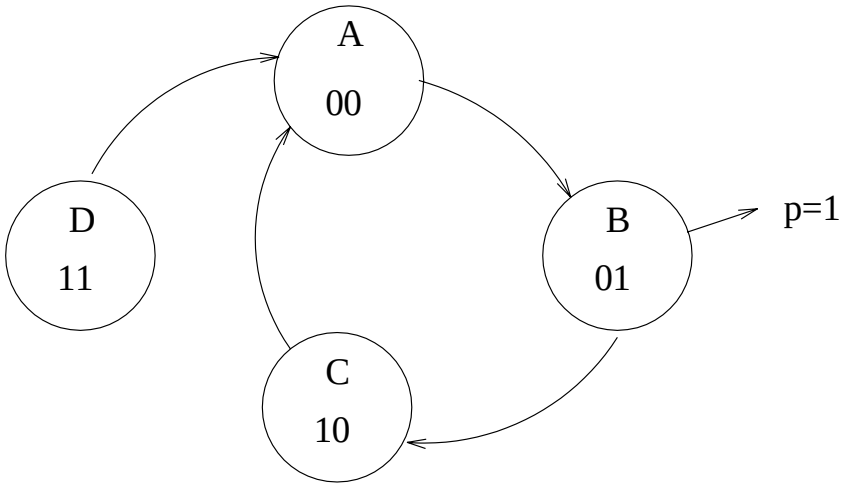


Figure35:Statediagramforadivide-by-3synchronouscounter.

The truth table for the combinational logic is below.It is important that the “extra state” D=11 be given an exit path, otherwise your processor may end up there upon power-up and remain stuck.(This effect has probably come to your attention with the “frozen” computer, which may require a reboot.)Also, note that we could have taken the output$p$fromanyofthestates$A$,B,or$C$.

PresentState		NextState		Output
	Q_1Q_0		D_1D_0	p
A	00	B	01	0
B	01	C	10	1
C	10	A	00	0
D	11	A	00	0

What are the logic expressionsfor our 3 “outputs” of this truth table (D_1, D_2, and p)?How would this be implementedwith D-type flip-flops and logic gates?WithJKflip-flops replacing the D-type?

7 MemoriesandProcessors

MemoryTerminology

We will not discuss the topic of data storage technologies*per se*.We are mostly interested here in the question of how data storage can be effectively *organized*.The important common element of the memories we will study is that they are*random access memories*, or RAM. This means that each bit of information can be*individually*stored or retrieved — with a valid input address.This is to be contrasted with*sequential*memories in which bits must be stored or retrieved in a particular sequence, for example with data storage on magnetic tape.Unfortunately the term RAM has come to have a more specific meaning:A memory for which bits can both be easily stored or retrieved ("written to" or "read from").Here isa rundown on some terms:

- RAM. In general, refers to random access memory.All of the devices we are considering tobe"memories"(RAM,ROM,etc.)arerandomaccess.ThetermRAMhasalso cometo meanmemorywhichcanbeboth easilywrittento and readfrom.Thereare two main technologies used for RAM:

  1.) Static RAM. These essentially are arrays of flip-flops.They can be fabricatedin ICs as large arrays of tint flip-flops.)"SRAM" is intrisically somewhat faster than dynamic RAM.

  2.) Dynamic RAM. Uses capacitor arrays.Charge put on a capacitor will producea HIGH bit if its voltage $V=Q/C$exceeds the threshold for the logic standard in use. Since the charge will "leak" off through the resistance of the connections in times of order $\sim$1 msec, the stored information must be continuously refreshed (hence the term "dynamic").Dynamic RAM can be fabricated with more bits per unit area in anIC than static RAM. Hence, it is usually the technology of choice for most large-scale IC memories.

- ROM.*Read-only memory*.Information cannot be easily stored.The idea is that bits are initially defined and are never changed thereafter.As an example, it is generally prudent for the instructions used to initialize a computer upon initial power-up to be storedinROM.ThefollowingtermsrefertoversionsofROMforwhichthestoredbits *can* be over-written, but not easily.

- PROM.*Programmable*ROM. Bits can be set on a programming bench by burning "fusible links," or equivalent.This technology is also used for programmable array logic (PALs), which we will briefly discuss in class.

- EPROM.ROMwhichcanbeerasedusingultravioletlight.

- EEPROM. ROM which can be erased electronically.A

few other points of terminology:

- Asyou know,a*bit*isabinary digit.Itrepresentsthesmallestelementofinformation.

- A*byte*is8bits.

- A"*K*"ofmemory is2^{10}=1024bits(sometimes written KB).Andamegabit(MB)is $1K \times 1K$bits.

- RAM is organized into many data “words” of some prescribed length.For example, a RAMwhichhas8*K*=8192memorylocations,witheachlocationstoringadataword of“width”16bits,wouldbereferredtoasaRAMofsize8*K*×16.Thetotalstorage capacity of this memory would therefore be 128KB, or simply a “128K” memory. (Withmodern*verylargescaleintegration*(VLSI)technology,atypicalRAMICmight be ∼ 16 MB.
- Besides the memory “size,” the other important specification for memory is the *access time*.Thisisthetimedelaybetweenwhenavalidrequestforstoreddataissentto a memory and when the corresponding bit of data appears at the output.A typicalaccesstime,dependinguponthetechnologyofthememory,mightbe∼10ns.

MemoryConfiguration

Asstatedabove,theterm“memory” referstoaparticularwayoforganizinginformation— by random access — which is distinct from the less specific term “data storage.” Figure 36 showshow an 8-bit RAM (8×1) is organized.(This is a very small memory, but illustrates the concepts.)Our RAM consists of three main components: an 8-bit multiplexer, an 8-bit demultiplexer, and 8 bits of storage.The storage shown consists of edge-triggered D-type flip-flops.Hence, this is evidently a “static RAM.” (There is no fundamental reason forusing edge-triggered flip-flops.They could just as easily be level-triggered, like the simple “clocked” S-R flip-flop of Fig.14.)

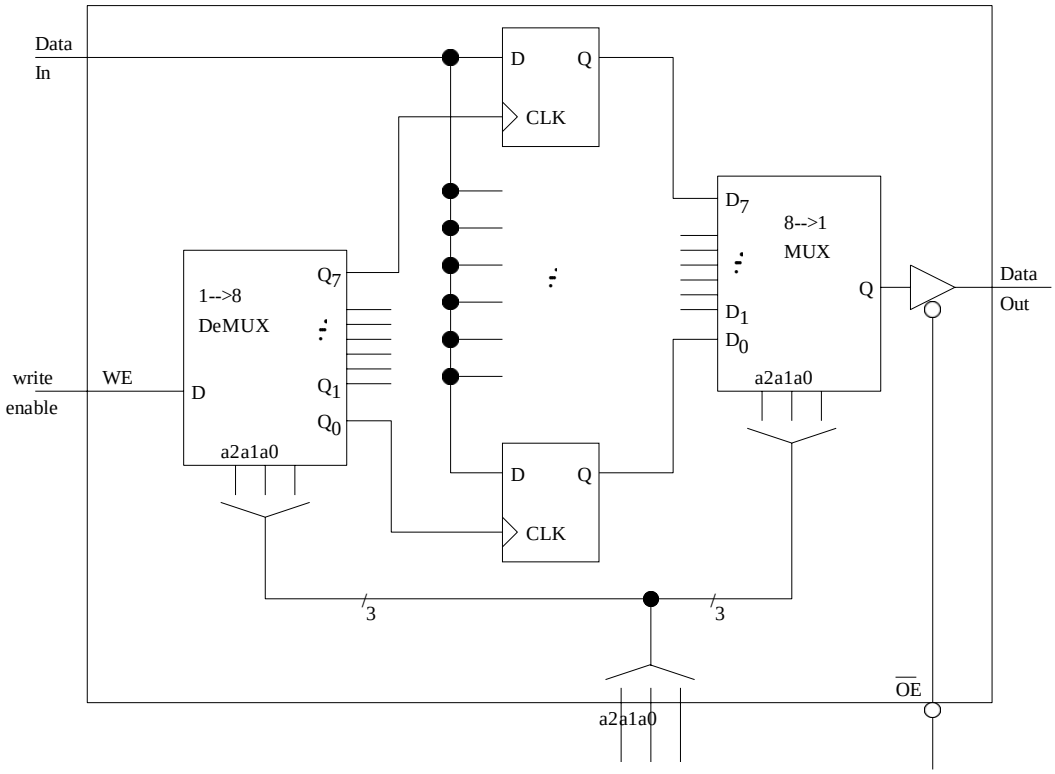


Figure36:An8×1bitRAM.

Our example RAM has 6 external connections which are inputs (data in, write enable (WE), 3-state enable $\overline{(}\overline{\text{O}}$E),and3 address bits ($A = a_2a_1a_0$), and has one output connection (dataout),giving7externalconnectionstotal,plus2forpower/ground.Towriteinformation to the RAM, one would supply a valid address, for example$A$=101.The data bit to be written to location 101 is to appear at the data input as either a logic HIGH orLOW signal. And to enable the writing into this bit, theWE signal must be asserted.This then appearsat theQ_5 output of the demultiplexer, and is passed on to the appropriate flip-flop, which stores the input data bit and passes it on to the Q_5multiplexer input.

ToreaddatafromourRAM,oneassertsanaddress,sothattheselectedbitissentto theMUX output andthen the3-state buffer.Thepurposeof the3-state bufferis to ensure that no digital outputs are directly connected together, for example if our RAM output were connectedtoadata"bus,"whichinturnwasconnectedtoseveralotherdevices.Recall that the 3-state devices have outputs which are effectively disconnected if there is no enable signal.So if the output data connection of our RAM is connected to a data bus, then the OE signalmustbecoordinatedwithanyotheroutputsalsoconnectedtothedatabus.Whenit isOKtoreaddatafromtheRAM(allotheroutputdevicesaredisconnectedfromthebus), the$\overline{\text{OE}}$signalisassertedandtheMUXoutputwillappearattheRAMoutput.

One could of course also store the 8 bits of data directly to an 8-bit data register, ratherthan using the RAM configuration outlined above.In this case, the number of external connectionsis17(8datain,8dataout,and1clock),comparedwiththe7ofourRAM.For a more realistic case where the number of bits of memory n is much larger than our example, we generalize the above to arrive at $4+\log_2(n)$ external connections for the RAM, compared with $1 + 2n$ for the standalone register.Obviously for large n, the register is impractical, whereastheRAMremainsreasonable.Actually,itisevensomewhatbetterthanthisforthe RAM case, since the number of external connections does not grow with the width of the stored data words.Hence, a RAM of size$1K\times16$=16KBrequires only 14 connections. This is to be compared with 32,001 connections for the register.Note that the RAM can only supply one bit at a time to the output.This may seem like a handicap, but is actually well matched to standard microprocessors.

AState Machine withMemory

For reference, our usual state machine configuration is shown again in Fig.37.Now we considertheuseofamemorywithastatemachine,asdepictedinFig.38. Arandomaccess memory is used in place of the usual combinational logic.(A ROM has been specified, to emphasizethatwearenotchangingthememory—onceitisdefinedinitially,itisonlyread from.Thememoryisusedtoconvenientlyencodetheconnectionbetweenpresentandnext states.

To start with,let's assume a state machine with no external inputs or outputs.Thenthe state machine's present state (PS) becomes an *address* which is input to the ROM. The *data word* stored in the ROM at that address then corresponds to the next state (NS). This correspondence had been initially programmed into the ROM, just asthe specific combinational logic in our old state machine had to be pre-determined.So if the PS as defined by the Q bits at the data register are, for example, 1001, then the ROM data word at address 1001will bethe NS which is then passed back tothe register.When there arealso external inputs, as there will be for most anything of interest, these are combined with the PS bitstoformalongeraddressfortheROM.Similarly,anyexternaloutputsarecombinedwith

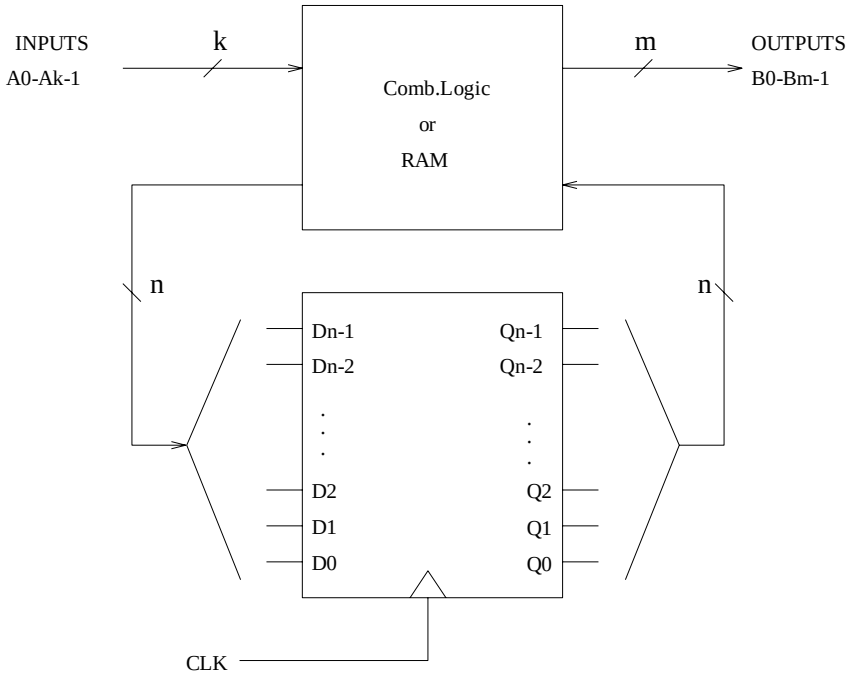


Figure37:Thestandardstatemachineconfiguration.

theNSbitsinthedataword.

Thisshouldbecomeclearwithanexample.

Example:Divideby2or3Counter

WewilluseastatemachinewithROM,asinFig.38,todesignacounterwhicheitherdivides by 2 or by 3, depending upon the value of an external input bit*p*.This state machine will require 3 states, therefore we will need to describe 4 states, using 2 bits.We can label the states*A* = 00,*B*= 01,*C*= 10,and*D* = 11.Let*p*= 0bethedivideby2case,and*p*=1 thedivideby3.Theoutputbit*r* = 1whenthepresentstateis*B*.Otherwise*r* = 0.State *D*isnormallyunused.Thetruthtableisbelow.The student shoulddrawthecorresponding state diagram.

	PresentState		NextState		
p	Q_1Q_0			D_1D_0	*r*
0	00	A	B	01	0
0	01	B	A	00	1
0	10	C	A	00	0
0	11	D	A	00	0
1	00	A	B	01	0
1	01	B	C	10	1
1	10	C	D	11	0
1	11	D	A	00	0

ThisROM requires3 addressbits (2 for PS and 1 for input bit *p*), whichcorresponds to8 locationsinmemory.Eachlocation hasadatawordwhichhaslength3bits(2forNSand

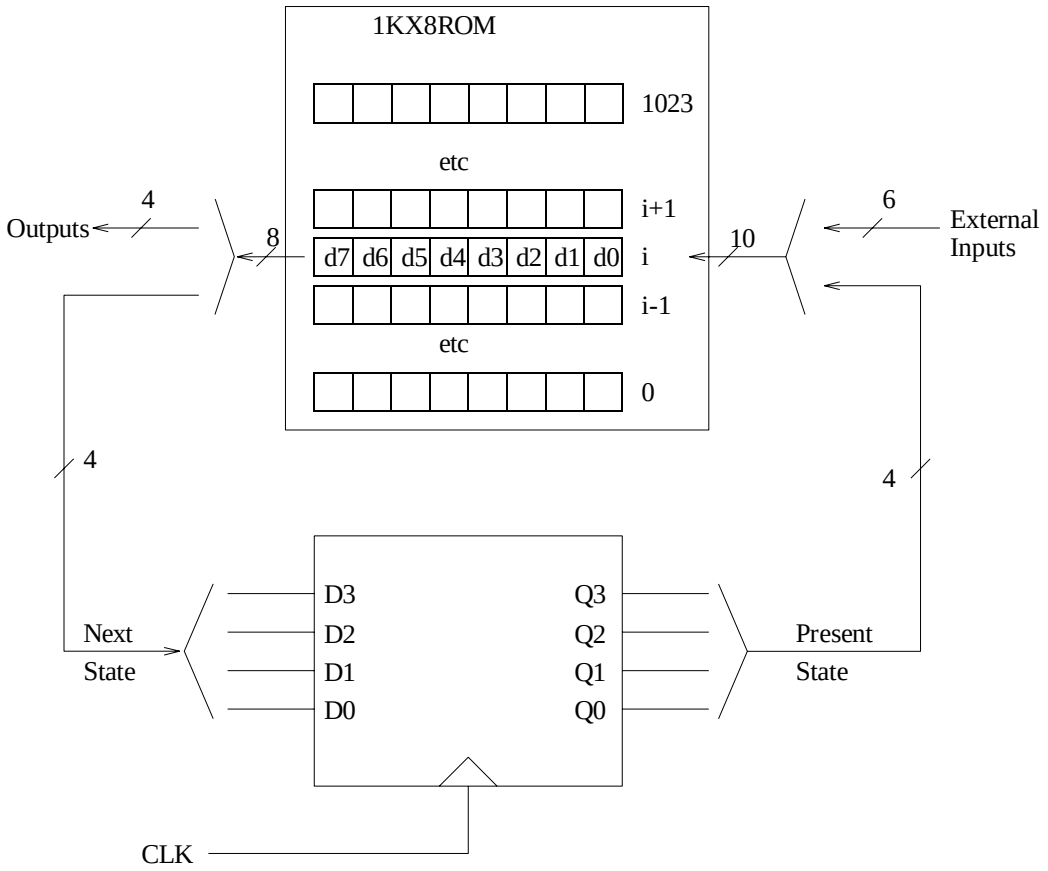


Figure38:Towardamicroprocessor:Replacingthecombinationallogicwithamemory.

1for the output bit *r*).Therefore, the size of this memory is 8×3, or 24 total bits.A very small ROM indeed.The programming of the ROM is very straightforward and can be read directly from the truth table above.We just need to set an encoding convention.Let the addressesbe encodedas$pQ_1Q_0$and the data wordsasD_1D_0r.For example,let's look at the 5throwofthetruthtable.Theaddresswouldbe 100 andthe data wordatthisaddress
would be 010. The remaining bits of the ROM would be programmed in the same way.So one would initially "burn in" these bit patterns into the ROM and put it into the circuit. That's all there is to it. Of course if one were careful not to overwrite the memory, or if an evolving logical pattern were required, then a RAM could be used instead of the ROM.

GeneralizationtoMicroprocessors

Astatemachinewithzeroinputbitscanperformacounter-likefunction,butnotmore: itsnextstateislimitedtobeafunctiononlyofthepresentstate.Asingleinputbitcan be used to "program" the state machine to behave in one of two possible ways for each present state, as we discussed, for example, with the up/down counter of Section 4.4.1, or the example in the preceeding section.On the other hand, with*n* inputs, the machine can perform 2^ndifferent operations.So, for example, with*n* =8 the machine can perform one of 256 different operations on each clock cycle.This tremendous potential and flexibility. Theinputbitscanthemselvesbesequenced —storedexternallyinaspecificsequence which

is then applied step by step to the state machine inputs on successive clock cycles.Such a storedsequence ofoperationsisa*program*andthe256operationsrepresent theprogramming operations. In Fig. 38 we have essentially configured a simple*microprocessor*. The inputs and outputs would need to be connected to buses (via 3-state buffers where appropriate), which in turn are also connected to memories which store the program and any output or inputdata.Thebuseswouldalsobeconnectedtovariousinput/outputdevices,massstorage devices, etc.