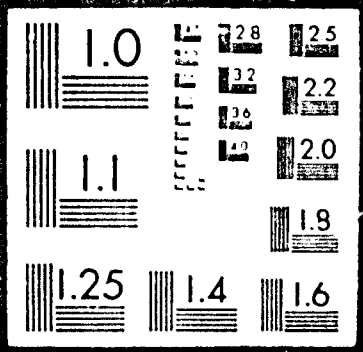


Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Project MAC
Massachusetts Institute of Technology

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
N/A

3. REPORT TITLE
AUTONOMOUS, SYNCHRONOUS COUNTERS CONSTRUCTED ONLY OF J-K FLIP FLOPS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*
FRANK MANNING

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
May 1972	64	6

8a. CONTRACT OR GRANT NO.
N00014-70-A-0362-0001

b. PROJECT NO. N/A

c. N/A

d. N/A

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
NONE

DDC
RECEIVED
JUN 27 1972
A

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited

11. SUPPLEMENTARY NOTES
NONE

12. SPONSORING MILITARY ACTIVITY
Office of Naval Research

13. ABSTRACT

This report describes research into some properties of autonomous, synchronous counters constructed with only the simplest form of J-K Flip-Flop. The research revolved around a system with a special-purpose digital machine and a general-purpose computer. The special-purpose machine searched through all the possible counters constructed of five or fewer J-K Flip-Flops for all counters with a period equal to that specified by the input to the systems. The descriptions of the counters found were transmitted by the special-purpose machine to the computer. Software analyzed this output data for various attributes, such as counters that cycle the same way independent of their starting-state.

Useful information for designers of digital machines, several proofs, and some insight into counters constructed only of J-K Flip-Flops resulted from this analysis. An example of the useful information is a table which gives synchronous counters using the minimum number of J-K Flip Flops with no other logic to realize periodic behavior with $P \leq 31$. One proof shows that any synchronous counter that uses only n J-K Flip-Flops and cycles with a period $P = 2^n - 1$ must have a trap state. Many other topics and proofs appear in the report.

The work is significant in three main ways. First, the results may be useful to designers of digital machines. Second, some interesting proofs are presented and conjectures that may lead to proofs areoffered. Third, a good solution of a problem is offered in which the interaction of a special-purpose digital machine and a general-purpose computer is far superior to the exclusive use of either.

I

DD FORM 1 NOV 68 1473 (PAGE 1)

S/N 0102-014-6600

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
J-K Flip Flops						
Computer-Aided Design						
Synchronous Counters						
Exhaustive Search						
Logic Design						

TI

AUTONOMOUS, SYNCHRONOUS COUNTERS

CONSTRUCTED ONLY OF J-K FLIP-FLOPS

Frank Manning

MAY 1972

This research was supported by the Advanced Research Projects Agency of the Department of Defense under ARPA Order No. 433, and was monitored by ONR under Contract No.N00014-70-A-0362-0001.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

CAMBRIDGE MASSACHUSETTS 02139

AUTONOMOUS, SYNCHRONOUS COUNTERS CONSTRUCTED ONLY OF J-K FLIP-FLOPS

ABSTRACT

This report describes research into some properties of autonomous, synchronous counters constructed with only the simplest form of J-K Flip-Flop.

The research revolved around a system with a special-purpose digital machine and a general-purpose computer. The special-purpose machine searched through all the possible counters constructed of five or fewer J-K Flip-Flops for all counters with a period equal to that specified by the input to the system. The descriptions of the counters found were transmitted by the special-purpose machine to the computer. Software analyzed this output data for various attributes, such as counters that cycle the same way independent of their starting state. Useful information for designers fo digital machines, several proofs, and some insight into these counters resulted from this analysis.

This report reproduces a thesis of the same title submitted to the Department of Electrical Engineering, Massachusetts Institute of Technology, in partial fulfillment of the requirements for the degree of Master of Science, March 1972.

ACKNOWLEDGEMENTS

I wish to thank Professor Edward Fredkin for his valuable guidance throughout the preparation of this report. I also thank Robert Fenichel and Professor Francis Lee for their useful suggestions about the direction of this research. Many people helped me with specific aspects of my work. Chief among these were John Roe in logic design and Jeff Golden in programming. Other members of the AI Lab were also helpful.

Two people deserve special thanks for their assistance in the preparation of the final version of this report. Madeleine Amyot helped in many ways, including editing. Anne Rubin did an excellent job as typist.

Thanks for personal assistance and encouragement go first to my parents, Thomas and Mary Anne Boschert Manning, and also to my friend, Lynn Talbert.

This research was supported in part by Project MAC, an M.I.T. research project sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number N00014-70-A-0362-0001.

TABLE OF CONTENTS

I. INTRODUCTION

A. Overview

The two classes of logic elements found in computers and other digital machines are combinational logic and memory. In an ideal combinational logic element, the output is an instantaneous function of the inputs. In a memory element the output may depend on the value of some inputs at previous times; the element may "remember" something about earlier inputs. In physical combinational logic devices, information travels from the inputs to the output with a slight delay. In this sense these devices have a very short memory.

Several memory devices can store the smallest unit of information, the bit, whose state corresponds to one of two possible conditions. Ferrite cores and semiconductors are commonly used for memory. Ferrite cores have many advantages, but they do not accept or provide information as quickly as some semiconductor memories. One fast, flexible, inexpensive memory element is the J-K Flip Flop. A study of an idealized version of this device appears in this report.

Counters are used in digital machines for various reasons, such as the synchronization of processes. Our chief concern is with automonous synchronous counters with only one type of logic element, the J-K Flip-Flop. The specific nature of these counters is described in the next section. The work focused on finding some of the capabilities and limits of this restricted class of machine. All of these counters of up to five J-K Flip-Flops were examined for various attributes, such as periodicity. Results from this examination appear in the report. For instance, counters with periods from one to thirty-one appear here. Each one is an interconnection of no more than five J-K Flip-Flops. Proofs and other insights into J-K Flip-Flop counters of arbitrary length also appear in the report.

B. Definition of some terms

The counters studied here are constructed of only one type of logic element, the J-K Flip-Flop (JKFF)*. Several slightly different JKFFs are commercially available. In this report the JKFF is viewed as a two-state device with three inputs and two complementary outputs.

THE JKFF

The outputs Q and $\bar{Q}$ are constant until the input to C changes from a logical 0 to a 1 (CT). When CT occurs, the new value of Q, called Q(t + 1), is calculated from the values of Q, J, and K when CT occurred according to the equation

$$Q(t + 1) = Q(t).\bar{K}(t) + \bar{Q}(t).J(t).$$

$\bar{Q}$ is always the complement of Q.

$$\bar{Q}(t + 1) = \overline{(Q(t + 1))}$$

Thus we can partition the output history of each JKFF into discrete portions of time -- 1,2, ... t, t + 1 -- separated by the occurrence of CT. The equations for the JKFF imply that Q changes from a 0 to a 1 only if J is 1 when CT occurs. Q changes from a 1 to a 0 only if K is 1 when CT occurs. When Q is 0, $\bar{Q}$ is 1. When Q is 1, $\bar{Q}$ is 0.

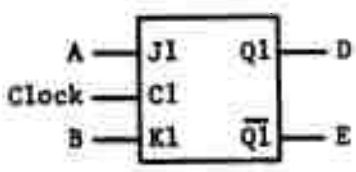
J(t)	K(t)	Q(t+1)	$\bar{Q}$(t+1)
0	0	unchanged	
0	1	0	1
1	0	1	0
1	1	changed	

TRANSITION RULES FOR THE JKFF

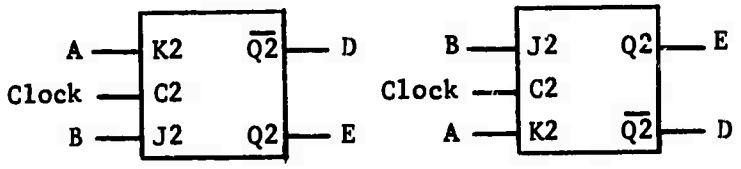
*The underline implies that the term underlined appears in the Terminology Section, which points to the definition of the term. The parentheses in the proper context imply that the parenthesized term is an abbreviation for the preceding expression; as such, it appears in the Abbreviation Table.

Other inputs are available in some commercial JKFFs. The most common of these are the "set" and "reset" inputs. Enabling the set input at any time forces a JKFF's Q to become a 1 almost instantaneously. Enabling the reset input similarly forces the JKFF's Q to become a 0. These inputs will not be considered further in this report. However, it is useful to note that proper use of the set and reset inputs of a collection of JKFFs can force them to assume any of their possible states. This method is often used to start a counter or other machine in a particular state.

We will use the symmetry property of the JKFF.



JKFF1



TWO REPRESENTATIONS OF JKFF2

JKFF1 is functionally identical to JKFF2 for $t \geq 1$ if:

1) J1 and K2 connect to the same source,
2) K1 and J2 connect to the same source,
3) C1 and C2 connect to the same source of timing pulses,
4) Q1 is compared to $\overline{Q2}$,
5) $\overline{Q1}$ is compared to Q2, and
6) $Q1(1) = \overline{Q2}(1)$.

For JKFF1:

$$Q1(t + 1) = Q1(t).\bar{B} + \overline{Q1}(t).A.$$

For JKFF2:

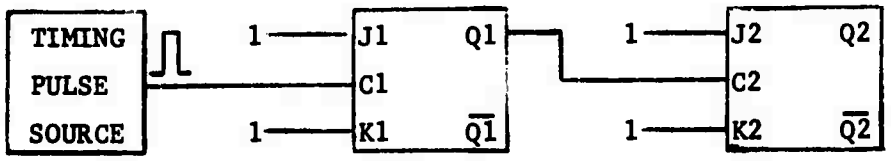
$$Q2(t + 1) = Q2(t).\bar{A} + \overline{Q2}(t).B.$$

Complementing both sides of this equation:

$$\overline{Q2}(t + 1) = \overline{Q2}(t).\bar{B} + \overline{\overline{Q2}}(t).A$$

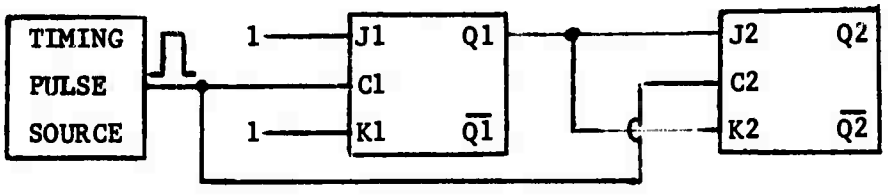
Therefore if $Q1(1) = \overline{Q2}(1)$, $Q1(t) = \overline{Q2}(t)$ and $\overline{Q1}(t) = Q2(t)$ for all $t \geq 1$.

A counter is a device which accepts timing pulses and outputs information concerning the number of these pulses. Here we study autonomous, synchronous counters, constructed of only one type of logic element, the JKFF (ASJKCs). Each counter is autonomous because after it is initialized its component JKFFs receive no inputs other than timing pulses to their Cs from any element outside the counter. In these ASJKCs the C input of each JKFF connects to the source of timing pulses. This results in simultaneous calculation of the new states of all the JKFFs in the ASJKC. This synchronous operation contrasts with that of asynchronous counters, in which changes of state for the component JKFFs do not occur simultaneously.



AN ASYNCHRONOUS COUNTER

For the counter above, the delay through JKFF1 that would occur in a physical counter results in asynchronous operation.



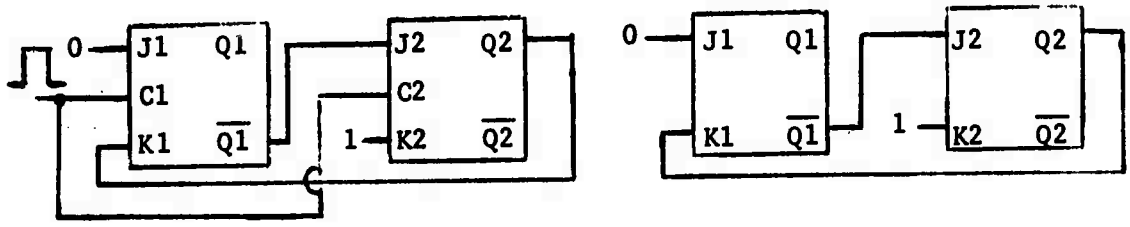
AN ASJKC

Because this report deals with synchronous counters, the connection of each C to the same source of timing pulses will be assumed; it will not be mentioned or indicated.

Because ASJKCs contain only one logic element, the only possible input source for each J and K is the constant 1, the constant 0, a Q of

the ASJKC, or a $\bar{Q}$ of the ASJKC. The input source for each input of an ASJKC is indicated by a connection list (CL) in the form:

$$CL = (J1\ K1\ J2\ K2 \ldots JN\ KN).$$



ALTERNATE REPRESENTATIONS OF CL (0 Q2 $\overline{Q1}$ 1)

The ASJKC's state at time t is indicated by an ordered list of the states of the component JKFFs.

$$S(t) = (Q1(t)\ Q2(t) \ldots QN(t))$$

Given the start-state S(1) and CL, we can calculate S(t) for all $t \geq 1$; for we can find the input values J(t), K(t) and Q(t) for all JKFFs for $t \geq 1$. For a ASJKC of n JKFFs (n-ASJKC) there are 2^n possible states. For example, two JKFFs imply the $2^2 = 4$ possible states (0 0) (0 1) (1 0) (1 1). In $2^n + 1$ successive states -- $S(1), S(2), \ldots S(2^n + 1)$ -- at least one state must occur twice. Assume that the first occurrence of this state was at $t = i$, the next at $t = j$. If $j - i$ CTs took the ASJKC from state S through some path back to state S, the next $j - i$ CTs must do the identical thing because S(t) and CL are the same at $t = i$ and $t = j$. The same argument applies for $S(j)$ and $S(j + (j - i))$, etc. Therefore, if $P = j - i$

$$S(t) = S(t - P) \qquad t \geq j$$

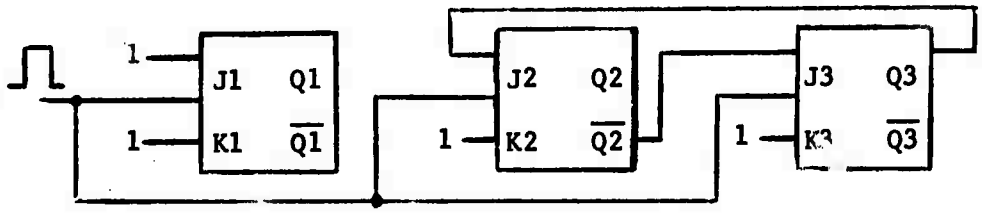
P is called the period of the ASJKC. Because $j - i \leq 2^n$, $P \leq 2^n$. As we pointed out, an ASJKC begins its periodic behavior before $t = 2^n + 1$. The period associated with various ASJKCs is of central importance in this report.

An ASJKC, like any other counter, may be minimal, safe, Gray, or binary. An ASJKC of period P composed of n JKFFs (n, P ASJKC) is <u>minimal</u> if $n = GI(\log_2 P)$*. The counter could not be realized with fewer logic elements. A counter is <u>safe</u> if it eventually cycles through the same states after being started in any of the 2^n possible start states. A counter is <u>Gray</u> if only one bit changes for each CT that occurs during its count. A counter is <u>binary</u> if the following conditions are true.

1) After it begins its count in S(1) = S, the lowest-order bit -- the one that changes state most often -- changes each time CT occurs. Each higher-order bit changes only when all lower-order bits are 1.

2) Rule 1 applies at all times except at t = xP. When the counter leaves S(xP), the bits of the counter may change in any way that results in S(xP + 1) = S = S(1).

A <u>composite</u> ASJKC is composed of two or more ASJKCs each with P ≥ 2. These connect to the same source of timing pulses but do not connect to each other in any other way. The composite counter with a period of six shown below is composed of two counters. JKFF1 is a counter with a period of two. The combination of JKFF2 and JKFF3 forms a counter with a period of three that cycles through the states (0 0), (0 1), (1 0), and then returns to (0 0). The (1 1) state goes to the (0 0) state, so the counter is safe.



A SAFE COMPOSITE ASJKC WITH P = 6

*GI stands for the "greatest-integer" function. This function rounds all numbers with a fractional part up to the nearest integer. GI(2.0) = 2, GI(2.1) = 3, GI(2.99) = 3.

The period of a composite ASJKC is found from the period of its component ASJKCs in the following way:

1) Factor the period of each of the component ASJKCs into prime factors.
   $P_1 = F_1 \times F_2 \times \ldots$, $P_2 = F_2 \times F_4 \times \ldots$, etc.
2) Associate with each factor F a number t equal to the maximum number of times F occurs in the factorization of any of the Ps. $F_1 \rightarrow t_1$, $F_2 \rightarrow t_2$, $F_3 \rightarrow t_3$.
3) The period of the composite ASJKC is the product
   $P = \prod (F_n)^{t_n}$.

For example, the period of a composite ASJKC with component periods of 12, 18, and 27 is found in the following way:

1) $12 = 2 \times 2 \times 3$, $18 = 2 \times 3 \times 3$, $27 = 3 \times 3 \times 3$.
2) 2 occurs twice in the factorization of 12, and 3 occurs three times in the factorization of 27.
3) Therefore the period of the composite ASJKC is P
   $P = 2^2 \times 3^3 = 4 \times 27 = 108$.

No composite ASJKC can be Gray or usefully binary. Because each component has $P \geq 2$, at least one bit of each component must change each time. Because there are at least two components, at least two bits must change each time. Because a Gray counter changes only one bit each time, a composite ASJKC can't be Gray. In a binary counter the lowest-order bit changes each time the counter is not returning to the start-state. If $P > 2$, there must be a time when the lowest-order bit is zero and the counter is not returning to the start-state. When this is true, none of the higher-order bits of the counter may change. Since a composite ASJKC changes at least two bits each time, the maximum period of a composite binary ASJKC is two. Since one JKFF can provide identical behavior, a composite binary ASJKC is useless.

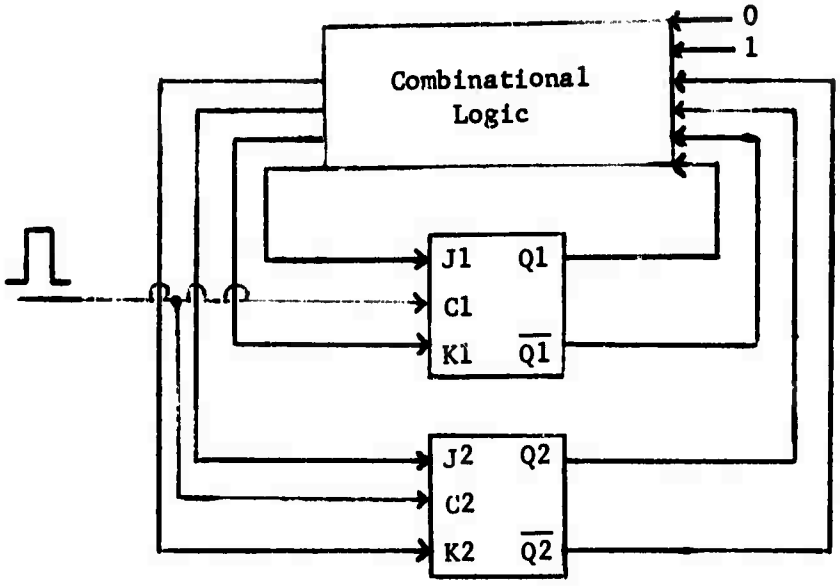
A composite ASJKC is safe if and only if its component ASJKCs are safe.

C. Scope of Report

This report describes research into some properties of ASJKCs. The research revolved around a system in which hardware searched through all n-ASJKCs with $n \leq 5$ for all ASJKCs with a period equal to that specified by the input to the machine. Descriptions of the ASJKC found were passed to the PDP-10 general-purpose computer. Software analyzed the output data for various things, such as safe counters. This interaction of hardware and software led to interesting experimental data. For instance, if five or fewer JKFFs are needed to synthesize a safe or at least unsafe ASJKC with a particular period, an ASJKC with the fewest number of JKFFs needed to provide that behavior is given in Section III. A method for using these ASJKCs to build composite ASJKCs is also presented in that section. The proof in Section IV that shows limits on ASJKCs arose from the experiments. A general method of synthesizing an autonomous, synchronous machine made only J-K Flip Flops that runs through a particular sequence of states is presented in Section VII. Insights into ASJKCs and the problems associated with studying them appear throughout the report.

The work is significant in three main ways. First, the results may be useful to the designer of digital machines. Second, some interesting proofs are presented and conjectures that may lead to proofs are offered. Third, a good solution of a problem is offered in which the interaction of a special-purpose digital machine and a general-purpose computer is far superior to the exclusive use of either.

Synchronous counters are used in most digital machines. Any counter of period P needs at least GI($\log_2$ P) bits of memory, and the JKFF is a commonly used memory element. A synchronous counter that uses this memory element and no other logic element is attractive for several reasons. Given the choice of a particular commercial JKFF, the ASJKC allows a very high rate of timing pulses.



AN AUTONOMOUS SYNCHRONOUS COUNTER

This is true because there are no logic gates to introduce delays as information about the state of the counter travels to the inputs. When an ASJKC is minimal, no functionally identical JKFF counter using fewer logic elements is possible. Even when an ASJKC is not minimal there is a possible economy in using only one logic element. Thus Section III, in which we present a table of ASJKCs with instructions telling how to use it to build composite ASJKCs, may be a useful guide to the designer of digital machines. The implications of some of the proofs mentioned below may also interest him.

Proofs of the following statements appear in Section IV.

1) There is no minimal binary n-ASJKC with $n > 3$.

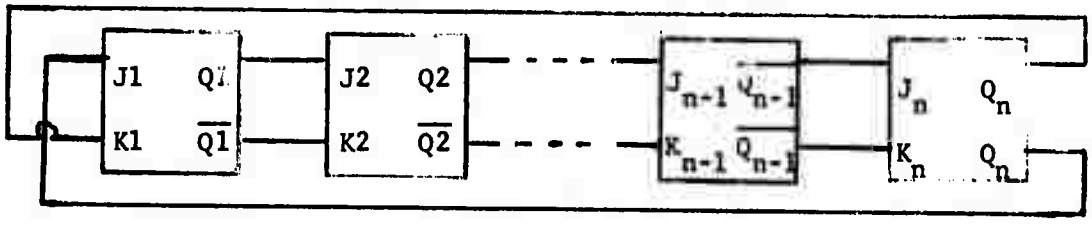
2) Any Gray ASJKC which uses all its JKFFs is equivalent to a switch-tailed shift-register and therefore has a period $P = 2n$.*

*Our definition of a Gray ASJKC implies that all JKFFs of the ASJKC participate in the count. Our proof does not apply to n-ASJKCs in which fewer than n JKFFs may count in Gray Code.

3) Any n, 2^n - 1 ASJKC with n > 2 is unsafe.

4) The constant 1 is useless as an input for n, P ASJKCs with odd period $P > 1.5 \times 2^{n-1}$.

Other less interesting proofs and mathematical analyses occur throughout the report. The report also notes trends in experimental data which lead to conjectures.



GENERAL SWITCH-TAILED SHIFT REGISTER

The report presents a good solution using the interaction of a special-purpose machine (SPM) and a general-purpose computer in a problem domain where the exclusive use of either is unfeasible. As Section II-A shows, the amount of computation necessary is impractically time-consuming for a general-purpose computer. This implies the need for a SPM. However, the large amount of output information requires the memory and computational facilities of a general-purpose computer. The interaction of a SPM and the PDP-10 general-purpose computer facilitated by an Execuport 300 terminal resolves this conflict.

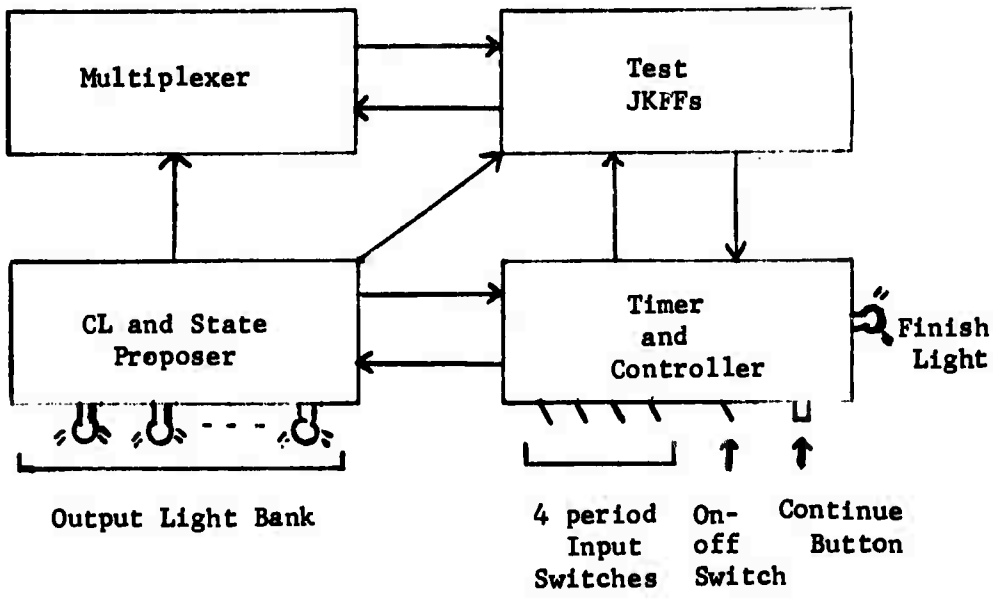
D. Previous Work

It appears that past study of ASJKCs has taken one of two directions. Francis Lee and his associates wrote a computer program which searched for one minimal n, P ASJKC, with $n \leq 4$ for each $P \leq 16$. Robert Fenichel and his associates took a different approach. They built a digital machine which searched through all n-ASJKCs with $n \leq 3$ for all ASJKCs with a period equal to the input period. The advantage of Lee's approach was that it took little human effort. The advantage of Fenichel's approach was that it provided complete information about the limited problem area.

Because an exhaustive search by a computer program through all possible 3-ASJKCs and 4-ASJKCs would take too long, Lee tried to find only one token ASJKC for each period. His program tested ASJKCs proposed in an orderly way. When it found an ASJKC with a period that hadn't been found earlier, it output the description of the ASJKC and its period. The search continued until ASJKCs for all periods of interest had been found or too much time had elapsed. This method did not find an ASJKC for P = 13 or P = 16. However, because the computer's search was not an exhaustive one, the conclusion could not be made from this search that no 4,13 ASJKC or 4,16 ASJKC exists.

Fenichel's approach answered all the questions mentioned in Section I-C for n-ASJKCs with $n \leq 3$. He designed a digital machine to search through all 3-ASJKCs for ASJKCs with the same period as the number input to the machine, $P \leq 8$. Each time such an ASJKC was found, the machine stopped and displayed the CL and start-state of the winning ASJKC on a bank of lights. The operator of the machine recorded this information on paper and signalled to the machine to continue its search. Slight modification of this machine allowed a similar search through 2-ASJKCs. ASJKCs found in these searches were examined for the properties of being Gray, binary, or safe. Analysis of this data and the data from the experiment described later in this report led to many conjectures about ASJKCs. Three of the proofs in Section IV resulted from these conjectures.

Because Fenichel's machine is similar to the one described later in this report, it's interesting to note its structure. Once the On-off switch



FENICHEL'S MACHINE

is turned on, the Timer and Controller synchronizes the proposal and subsequent test of ASJKCs. The CL and State Proposer is a counter. Each count corresponds to a particular CL and start-state. When the test of a CL and start-state begins, the Test JKFFs are forced into one of their eight possible states according to the command of the State Proposer. The CL Proposer's input to the multiplexer determines whether each input is connected to a logical 0, a logical 1, or one of the outputs of the Test JKFFs. After the Test JKFFs are connected appropriately and forced into a start-state they receive timing pulses from the Timer and Controller. The Timer and Controller monitors the response of the Test JKFFs to determine whether they cycle with period P. If the latter does not occur, the CL and State Proposer receives a pulse which signals

it to propose a new combination of CL and start-state. If the Test JKFFs do cycle with period P, the digital machine stops. Information telling the CL and start-state which caused success is output on the Output Light Bank. After the operator copies this information onto paper, he signals the machine to continue. When the CL and State Proposer reaches its highest count, there are no more ASJKCs to try. The Timer and Controller turns on the Finish Light.

II THE EXPERIMENT

A. Overview

A system was designed to test all possible n-ASJKCs with n = 4 and 5 for a periodicity specified by the input to the system. A particular n,P ASJKC test was run if either a safe or an unsafe x,P ASJKC was not found for $x < n$. For example, there was a search for 4, 9 ASJKCs because this search could yield minimal period—9 ASJKCs. There was no search for 5, 9 ASJKCs because minimal safe and unsafe 4, 9 ASJKCs were found.

The major steps taken by the system in looking for all n, P ASJKCs are below.

1) Propose all possible Connection Lists (CLs) for n-ASJKCs.

2) Eliminate each proposed CL that is easily shown to be equivalent to a CL that is definitely tested.

3) Determine whether the ASJKC implied by the combination of a CL that isn't eliminated because of redundancy and the all-zero start-state (00...0) returns to the all-zero state from that state in P units of time. As we'll see, this results in an exhaustive search through all possible n-ASJKCs.

4) Output each winning CL into a PDP-10 file for n, P ASJKCs. If the file contains 2047 winning CLs, stop output to the PDP-10 and simply count the number of winning CLs. This saves time and computer resources.

Some of the steps occur simultaneously for different CLs.

The n, P ASJKC PDP-10 file is later analyzed by software. All files are checked for safe ASJKCs, and some files are examined more extensively.

Exhaustive search was used in searching for ASJKCs with the properties of interest for two reasons. First, if any ASJKC with five or fewer JKFFs had one of the properties of interest, our system found that ASJKC. This thorough information assures that each entry in the table of ASJKCs for $2 \leq P \leq 32$ represents the minimum number of JKFFs needed

to realize that ASJKC. This information was also a valuable guide in formulating the proofs of Section IV. For instance, the experimental fact that there is no n, $2^n - 1$ ASJKC with $3 \leq n \leq 5$ led to the proof that this was the case for all n, $2^n - 1$ ASJKCs with $n \geq 3$. The second reason for using the exhaustive search was that it necessitated the interesting job of finding techniques to speed up Fenichel's implementation of this approach.

The technique of exhaustive search that Fenichel used was adequate in searching 3-ASJKCs but would have been inadequate in searching 5-ASJKCs. Three general observations led to drastic reductions in the amount of computation and output necessary for search. First, testing all CLs with all 2^n starting-states is equivalent to testing all CLs with the all-zero start-state (S(1) = (00 ... 0)). Second, only 2n - 1 of the 2n + 2 possible sources of input to a JKFF in a ASJKC are useful.* This means that only $(2n - 1)^{2n}$ instead of $(2n + 2)^{2n}$ CLs need be proposed. For n = 5, $\left(\frac{2n-1}{2n+2}\right)^{2n} \approx \frac{1}{18}$. Third, because many CLs result in equivalent ASJKCs some redundant ones need not be tested. This observation led to a further reduction in tests to about 1/45 of those that would have been necessary in searching 5-ASJKCs.

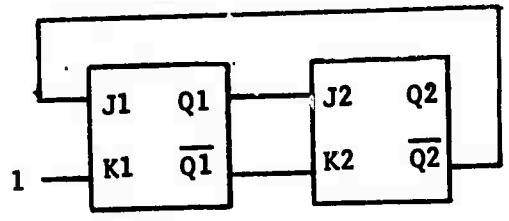
Testing each CL with the all-zero start-state is equivalent to testing each CL with all 2^n possible start-states. Consider a counter started in some state other than the all-zero start-state. Because of the symmetry property discussed in Section I-B we can relabel all JKFFs whose start-state is a 1.

$$Q \rightarrow \bar{Q}, \quad \bar{Q} \rightarrow Q, \quad J \rightarrow K, \quad K \rightarrow J.$$

If we start this new <u>symmetrically equivalent counter</u> in the all-zero start-state it cycles with the same period as the first counter. Because each counter specified by a CL and a start-state cycles in the

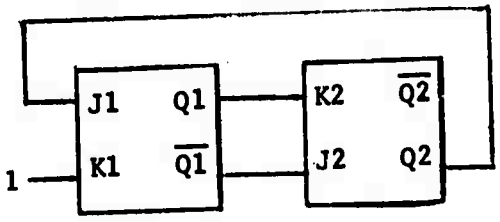
*The last proof in Section IV shows that for odd periods with $P > 1.5 \times 2^{n-1}$ only 2n-2 possible sources of input are useful.

same way as some counter specified by a CL with an all-zero start-state, testing all CLs with the all-zero start-state results in an exhaustive search.



S(1) = (0 1)

CL = ($\overline{Q2}$ 1 Q1 $\overline{Q1}$)



S(1) = (0 0)

CL = (Q2 1 $\overline{Q1}$ Q1)

SYMMETRICALLY EQUIVALENT COUNTERS

The connection of one of the inputs of a JKFF to either of its outputs or 0 is useless.

When J connects to the constant 0,

$$Q(t+1) = Q(t).\bar{K}(t)+\bar{Q}(t).0 = Q(t).\bar{K}(t).$$

This implies that if Q ever becomes a 0 it remains a 0. Consequently the Q is either a constant 1 or a constant 0 after the ASJKC begins its periodic behavior. Because it duplicates the function of constants already available, a Q with its J connected to 0 is useless. Because of the symmetry property, connecting K to 0 is similarly useless -- as soon as $\bar{Q}$ is 0 it stays 0.

When J and Q of the same JKFF are connected,

$$Q(t+1) = Q(t).\bar{K}(t) + \bar{Q}(t).Q(t) = Q(t)\bar{K}(t).$$

This is equivalent to connecting J to 0, which is useless. Because of symmetry, connecting K to $\bar{Q}$ is similarly useless.

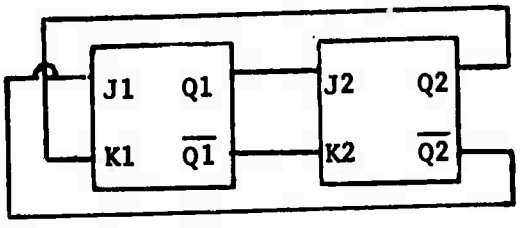
When J and $\bar{Q}$ of the same JKFF are connected,

$$Q(t+1) = Q(t).\bar{K}(t) + \bar{Q}(t).\bar{Q}(t) = \bar{K}(t) + \bar{Q}(t).$$

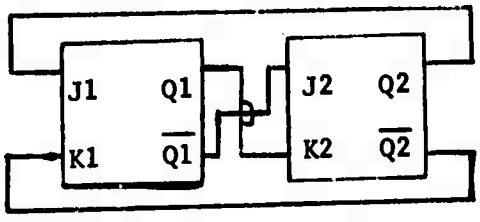
When J connects to the constant 1,

$$Q(t+1) = Q(t).\bar{K}(t) + \bar{Q}(t).1 = \bar{K}(t) + \bar{Q}(t).$$

Because connecting a JKFF's J to its $\bar{Q}$ is equivalent to connecting its J to 1, the connection of J to $\bar{Q}$ is redundant and therefore useless. By symmetry, connection of a JKFF's K to its Q is similarly useless.



S(1) = (0 0)
CL = ($\overline{Q2}$ Q2 Q1 $\overline{Q1}$)



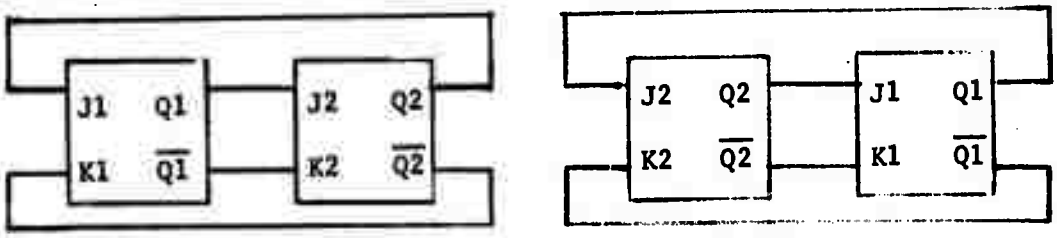
S(1) = (0 0)
CL = (Q2 $\overline{Q2}$ $\overline{Q1}$ Q1)

PERMUTATIONALLY EQUIVALENT COUNTERS

Because all the ideal JKFFs we consider are functionally identical, different Connection Lists (CLs) with the all-zero start-state may specify equivalent counters, as in the case above. Two ASJKCs are <u>permutationally equivalent</u> if some renumbering of the JKFFs in one counter implies a CL and S(1) for that counter identical to the CL and S(1) of the other counter. CLs are permutationally equivalent if their counters are permutationally equivalent. For instance, the x-counters and CLs above are permutationally equivalent. Relabelling the ASJKC on the left, JKFF1 → JKFF2, JKFF2 → JKFF1 results in CL = (Q2 $\overline{Q2}$ $\overline{Q1}$ Q1) and S(1) = (00). This is identical to the combination for the ASJKC on the right. Permutationally equivalent ASJKCs obviously cycle with the same period. In fact, if the proper outputs are compared they behave identically.

For an n-ASJKC there are at most n! different but permutationally equivalent CLs corresponding to all possible relabellings of n JKFFs.

There are fewer permutationally equivalent CLs when some different labellings result in identical CLs, as in the case below.



S(1) = (0 0)
CL = (Q2 $\overline{Q2}$ Q1 $\overline{Q1}$)

S(1) = (0 0)
CL = (Q2 $\overline{Q2}$ Q1 $\overline{Q1}$)

EFFECT OF RELABELLING ONE ASJKC

A way was needed for using the fact of permutationally equivalent CLs to reduce the number of CLs tested. Because the technique was applied to $(2n-1)^{2n}$ CLs ($\approx 3.5 \times 10^9$ CLs for 5-ASJKCs), it had to take a short time. It also had to be inexpensive and easy to implement.

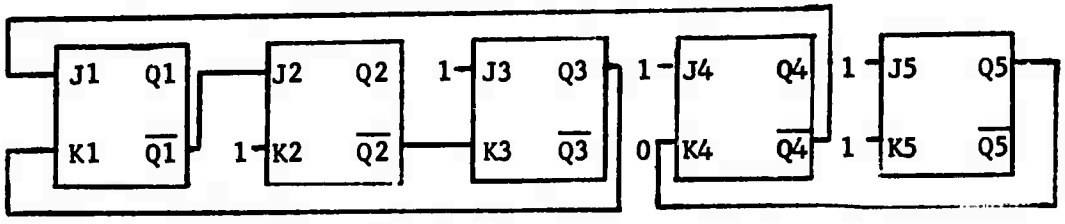
Consider the following partition of all possible inputs to a JKFF in a 5-ASJKC.

J	K	Fraction of Times Proposed	Arbitrarily Assigned Priority
1	1	1/81	1
1	Q	4/81	2
1	$\bar{Q}$	4/81	3
Q	1	4/81	4
Q	Q(S)*	4/81	5
Q	$\bar{Q}$(S)	4/81	6
Q	Q($\bar{S}$)	12/81	7
Q	$\bar{Q}$($\bar{S}$)	12/81	8
$\bar{Q}$	1	4/81	9
$\bar{Q}$	Q(S)	4/81	10
$\bar{Q}$	$\bar{Q}$(S)	4/81	11
$\bar{Q}$	Q($\bar{S}$)	12/81	12
$\bar{Q}$	$\bar{Q}$($\bar{S}$)	12/81	13

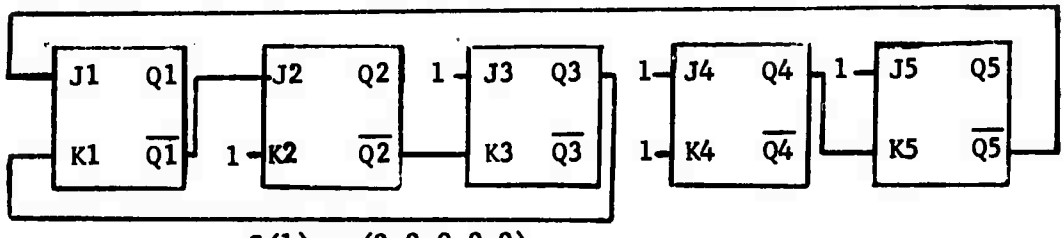
*"S" means "of the same JKFF that J connects to". $\bar{S}$ means "of a different JKFF from the one J connects to". For instance, the entry for J = Q, K = $\bar{Q}$(S) indicates that four out of eighty-one times J1 connects to a Q and K1 connects to the $\bar{Q}$ of the same JKFF.

The assignment of each JKFF of a 5-ASJKC to one of the thirteen categories does not depend on the renumbering of any of the JKFFs in that ASJKC. Each 5-ASJKC implies a five-digit, base-thirteen partition number (PN) corresponding to the input category of each of the five JKFFs. For instance, CL = ($\overline{Q5}$ Q3 $\overline{Q1}$ 1 1 $\overline{Q2}$ 1 1 1 Q4) implies PN = 12 9 3 1 2. The PN is trivially found from the CL. Permutationally equivalent CLs result in PNs with identical digits in correspondingly permuted positions.

This partitioning is used by the system to eliminate most redundant CLs from testing. The PN is treated by the system as a base-thirteen number with its most significant digit (corresponding to the input category of JKFF1) on the left. Only when the PN's digits cannot be permuted to yield a higher number is the corresponding CL tested. This is equivalent to saying that each digit of the PN must be greater than or equal to the digit to its right for a CL to be tested. For instance, only the upper of the two permutationally equivalent CLs below would be tested by our system.



S(1) = (0 0 0 0 0)
CL = ($\overline{Q4}$ Q3 $\overline{Q1}$ 1 1 $\overline{Q2}$ 1 Q5 1 1)
PN = 12 9 3 2 1 → TESTED



S(1) = (0 0 0 0 0)
CL = ($\overline{Q5}$ Q3 $\overline{Q1}$ 1 1 $\overline{Q2}$ 1 1 1 Q4)
PN = 12 9 3 1 2 → NOT TESTED

TWO PERMUTATIONALLY EQUIVALENT CLs

The fact that the digits of the PN of the lower counter can be permuted to yield the higher PN reflects the fact that the two CLs are permutationally equivalent. The system would test only the upper, higher CL.

This partitioning technique is remarkably good at eliminating permutationally redundant CLs. As Appendix B shows, for 5-ASJKCs only 1/45 of the proposed CLs result in testing. For 6-ASJKCs, only 1/141 of the proposed CLs would be tested. Compare this to a technique that eliminates all redundancies for n-ASJKC CLs by forming an n-digit, base $(2n - 1)^2$ number from the CL. Because permutationally equivalent CLs do not result in PNs with a simple relationship, it's hard to use a PN formed in this way to look for redundancies.

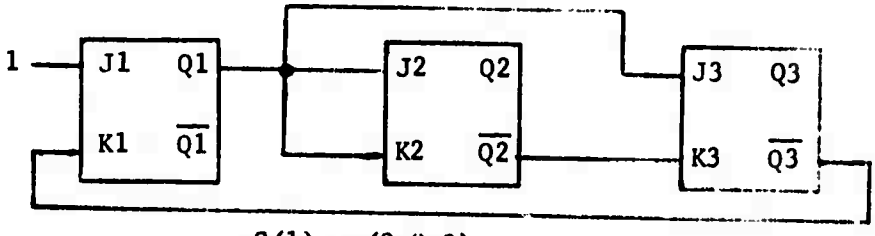
The problem with this method is best illustrated by an example. Consider the partition of inputs to a JKFF in a 3-ASJKC shown on the next page.

J	K	Fraction of Times Proposed	Arbitrarily Assigned Priority
1	1	1/25	1
1	Q(L)*	1/25	2
1	$\bar{Q}$(L)	1/25	3
1	Q(R)	1/25	4
1	$\bar{Q}$(R)	1/25	5
Q(L)	1	1/25	6
Q(L)	Q(L)	1/25	7
Q(L)	$\bar{Q}$(L)	1/25	8
Q(L)	Q(R)	1/25	9
Q(L)	$\bar{Q}$(R)	1/25	10
$\bar{Q}$(L)	1	1/25	11
$\bar{Q}$(L)	Q(L)	1/25	12
$\bar{Q}$(L)	$\bar{Q}$(L)	1/25	13
$\bar{Q}$(L)	Q(R)	1/25	14
$\bar{Q}$(L)	$\bar{Q}$(R)	1/25	15
Q(R)	1	1/25	16
Q(R)	Q(L)	1/25	17
Q(R)	$\bar{Q}$(L)	1/25	18
Q(R)	Q(R)	1/25	19
Q(R)	$\bar{Q}$(R)	1/25	20
$\bar{Q}$(R)	1	1/25	21
$\bar{Q}$(R)	Q(L)	1/25	22
$\bar{Q}$(R)	$\bar{Q}$(L)	1/25	23
$\bar{Q}$(R)	Q(R)	1/25	24
$\bar{Q}$(R)	$\bar{Q}$(R)	1/25	25

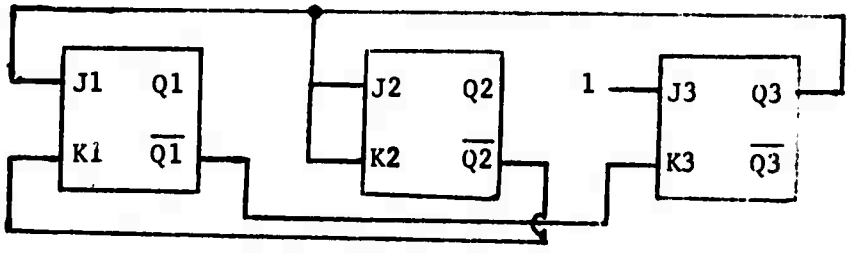
A PARTITION OF INPUTS IN A 3-ASJKC

*"L" means the leftmost JKFF that the input could connect to. "R" means "the rightmost ...". For instance, Q(L) for J1 is Q2; Q(L) for J2 is Q1; Q(L) for J3 is Q1.

This partitioning scheme results in the following PNs for two permutationally equivalent CLs.



S(1) = (0 0 0)
CL = (1 $\overline{Q3}$ Q1 Q1 Q1 $\overline{Q2}$)
PN = 5 7 10



S(1) = (0 0 0)
CL = (Q3 $\overline{Q2}$ Q3 Q3 1 $\overline{Q1}$)
PN = 18 19 3

TWO COUNTERS WITH PERMUTATIONALLY EQUIVALENT CLs

Although these counters result from permutationally equivalent CLs, their PNs are not related in a simple way. Consequently with this partitioning scheme the elimination of redundancies is complex and expensive. This scheme implies testing of more than $\frac{1}{120} = \frac{3}{8} \times \frac{1}{45}$ of the CLs proposed. This improvement is not significant enough to warrant choice of this scheme over the simpler, less costly one.

We now have enough information to estimate the time it would take a special-purpose machine using the base—13 PN to search through all 5-ASJKCs or 6-ASJKCs for ASJKCs of a particular period. Make the following reasonable assumptions:

1) The timing portion of the machine cycles once every 165 nsec. This is true for our conservatively designed system.

2) Until the machine finds a CL to test, it proposes and tests for redundancy one new CL each timing cycle. When it finds a CL to test, the proposer portion stops proposing until completion of that test.

3) The average test of a nonredundant CL takes 2^{n-1} timing cycles. The average test is actually higher for higher periods. However, the average for all periods is about 2^{n-1}.

4) The time taken to output winning CLs is negligible. Only CL proposal and CL testing take time.

If these assumptions are true the runtime of the machine equals the sum of the CL-proposal time and the CL-test time. CL-proposal time is $(2n-1)^{2n} \times 165$ nsec. CL-test time depends on the fraction of proposed CLs that are tested. For $n = 5$ this fraction $F = 1/45$, and for $n = 6$ $F = 1/141$. CL-test time is $F \times (2n-1)^{2n} \times 2^{n-1} \times 165$ nsec. Therefore, the total time for a test is about 780 seconds for $n = 5$ and 177 hours for $n = 6$. The time to test 6-ASJKCs is long but not impossibly long. We chose not to spend the extra time and effort to test 6-ASJKCs. The time to test 5, 25 ASJKCs actually took 815 seconds and took a negligible amount of time to output 91 winning CLs. This deviation from the prediction is small and results from the fact that the assumptions above are not precise, as we will see.

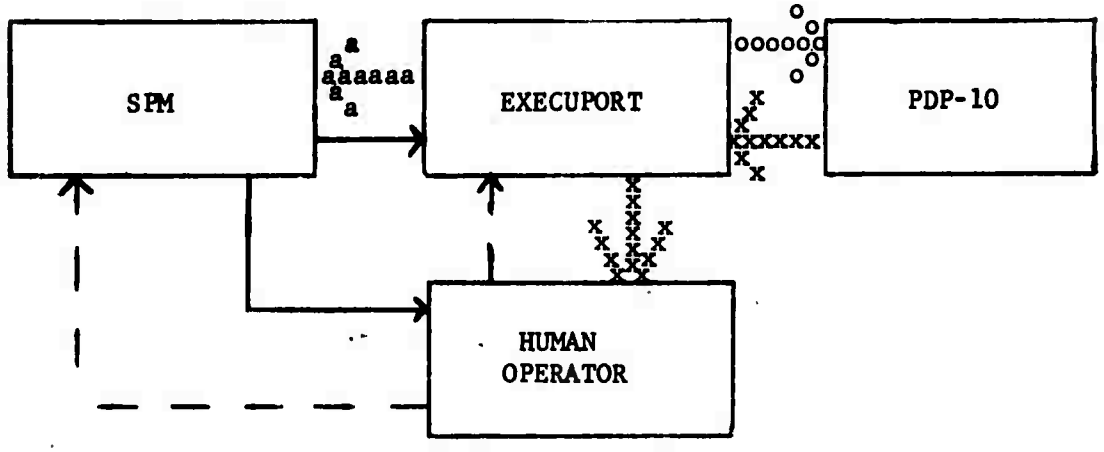
All software schemes which were considered take too much computer time and consequently cost too much. This is due to the slower cycle time and lower parallel processing capability of the machines we could use. Consequently we chose to search for winning CLs with a special-purpose machine incorporating ideas presented in this section.

The special-purpose machine would have been inadequate as an output device. For any test we wanted to output all winning CLs less than

some large number which we later chose to be 2048. A scheme such as Fenichel's, in which paper marked by a human serves as a memory device for winning CLs, would have been too slow to handle the large amount of output expected. The choice of the PDP-10 as an available general-purpose computer to handle output was a natural one. It served as a memory for the output information and enabled simple, quick software analysis of that output information. An Execuport 300 terminal facilitated the linking of the special-purpose machine and the PDP-10. The Execuport 300 accepted data from its operator-controlled keyboard and the special-purpose machine. It passed this data over a phone line to the PDP-10.

B. The System

In the system designed a special-purpose machine (SPM) searches for n-ASJKCs of a specified period P with n = 4 or 5, $1 \leq P \leq 32$. Winning CLs are output to the PDP-10 as 5-character words. As was mentioned, the SPM was chosen because of its speed and economy in computation. The PDP-10 was chosen because as an available general-purpose computer it provided large storage and software facilities for output data. The SPM and PDP-10 interface through an Execuport 300, a terminal with a keyboard and facilities for connection to the SPM. The Execuport communicates with the PDP-10 through a standard telephone. A human operator opens a PDP-10 file, initiates the exhaustive search, and closes and names the file when the search is over.



\- - - Input Commands

——— Output Information

aaaaaa Timing Pulses

oooooo Information From SPM and Operator

xxxxxx Echo of oooooo

SYSTEM FOR FINDING AND FILING ASJKCs

In the search for a particular n,P ASJKC the following steps occur.

1) The operator turns the SPM's On-off Switch to Off. He sets the SPM's input switches to the period P.

2) The operator uses the Execuport to type commands to the PDP-10 which prepare it to file information.

3) The operator turns the SPM's On-off Switch to On. This initiates the exhaustive search. When successful CLs are found, they are transmitted as 5-character words through the Execuport to the PDP-10 file. If there are more than 2047 winning CLs, only the first 2047 are entered into the file. The SPM displays a count of the number of winning CLs found.

4) When the SPM signals that its search is over by turning on a Finish Light, the operator types commands which close and name the file. This file is later examined by software.

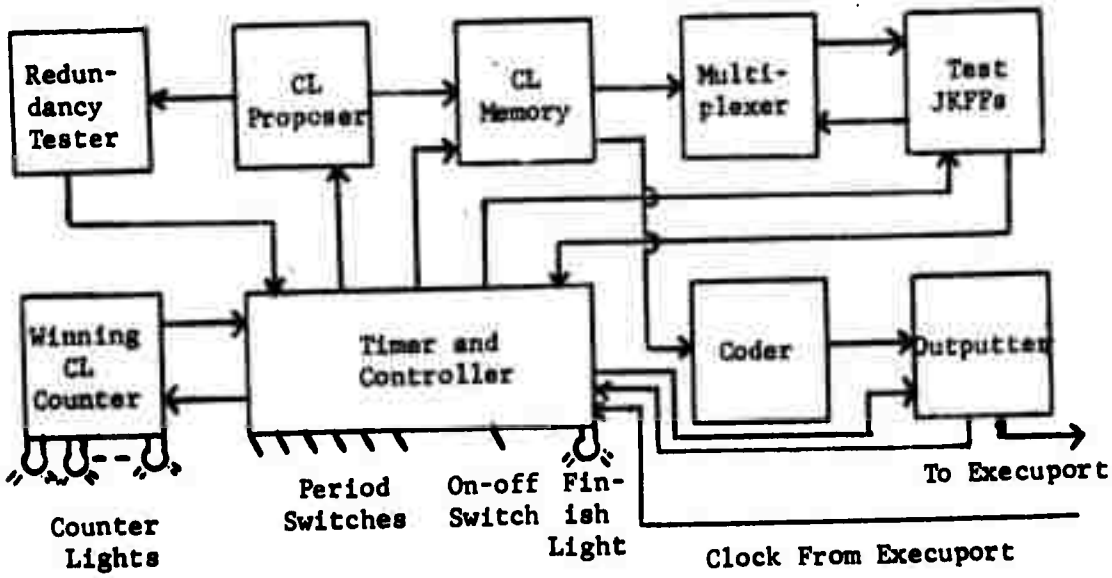
Below is a description of each component of the system other than the operator.

Execuport: This transmits information from the operator and the SPM to the PDP-10. A 300 bit/second clock in the Execuport determines the rate of transmission of this information. The ease of communication between the components of the system that the Execuport allows offsets the disadvantage of its low information rate. All information sent to the PDP-10 from the Execuport is echoed by the PDP-10. The Execuport prints this echoed information, which the operator uses to monitor the run.

SPM: The SPM's input is a 6-bit code specifying in binary the input period P with $1 \leq P \leq 32$. When the On-off Switch is turned On, the SPM is initialized and the search begins. When a successful CL is found, it is transmitted to the PDP-10 as a 5-character word. The SPM counts

all winning CLs and transmits the first 2047 found. When the SPM completes its exhaustive search, it turns on its Finish Light.

The SPM was designed to handle 5,P ASJKC tests. A trivial modification -- moving a few wires and stopping the machine after it proposed 1/81 of the CLs proposed for the 5,P ASJKC tests -- allowed 4,P ASJKC tests.



THE SPM*

As the SPM diagram indicates, there is a great deal of communication between components of the SPM. A description of each component and its relationship to the other component for 5,P ASJKC tests follows.

CL_Proposer: This proposes each of the $(9)^{10}$ CLs to the Redundancy Tester. The CL Proposer is realized as a synchronous counter which counts in base-9 from 0000000000 to 8888888888. Each digit determines the input source for one of the ten inputs. The leftmost digit applies to J1, the next to K1, the next to J2, etc. The connection implied by an input digit for one of the input digits of JKFFN is found in the following way.

*The SPM was designed, built, and debugged by the author.

1) Eliminate QN and $\overline{QN}$ from the list
($Q1\ \overline{Q1}\ Q2\ \overline{Q2}\ Q3\ \overline{Q3}\ Q4\ \overline{Q4}\ Q5\ \overline{Q5}\ 1$).

2) An input digit value of X implies the connection of the input to the (X + 1)th member of the new list.

For instance, for the inputs to JKFF1: $0 \rightarrow Q2$, $2 \rightarrow Q3$, $8 \rightarrow 1$. For the inputs to JKFF2, $0 \rightarrow Q1$, $2 \rightarrow Q3$, $8 \rightarrow 1$. The Timer and Controller provides the timing pulses for the CL Proposer.

Redundancy_Tester: This examines the proposed CL for permutational redundancy by partitioning the possible inputs to each JKFF into thirteen classes as previously described. The Redundancy Tester is realized with combinational logic. This logic tells the Timer and Controller whether a CL is redundant about 100 nsec. after the CL is proposed. The system conservatively allows 165 nsec. to pass between proposals of CLs.

CL_Memory: This stores a CL if it passes the redundancy test. The stored CL controls the interconnection of the Test JKFFs through the Multiplexer. The CL Memory allows the testing of one CL while the CL Proposer searches for the next CL to be tested. This inessential time-saver should not have been included in the SPM because it does not save much time and it obscures interesting runtime statistics. Without the CL Memory the total time for proposal and test for period P of CLs is the sum of the proposal time and the test time. For 5-ASJKCs this equals $(9)^{10} \times 165$ nsec.$\times\left(1 + \frac{N_{test}}{45}\right)$, where N_{test} is the average number of 165 nsec. machine cycles needed to test a nonredundant CL for period P. If the CL Memory is used, the most time is saved if each nonredundant CL is followed by 44 redundant ones. The total time for proposal and test then just equals the time for proposal, which equals $(9)^{10} \times 165$ nsec.

The percentage of time saved by using the CL Memory is therefore at most $\frac{(N_{test} \times 100)}{(45 + N_{test})}$. Since the average N_{test} for the 5,P ASJKCs tested is approximately 16, the CL Memory reduces the time for proposal and testing by at most about 26%. We guessed earlier that the average time for proposal and testing of CLs is about 780 seconds. There were 20 searches through 5-ASJKCs, as the input period ranged from 13 to 32. The time to test 4-ASJKCs is negligible. Therefore the CL Memory saved at most about .26 x 20 x 780 seconds ≈ 4060 seconds. Considering the time needed to wire in the CL Memory, this saving of runtime was not worthwhile. Another disadvantage of the CL Memory is that in allowing the times for proposal and testing of CLs to overlap it makes it difficult to determine experimentally the precise value of N_{test} for different input periods.

Multiplexer: This logically connects each input of the Test JKFFs to its proper source as determined by the CL Memory.

Test JKFFs: For each test of a nonredundant CL, these are connected in a way determined by the CL Memory and started in the all-zero start-state. The Timer and Controller then provides timing pulses and monitors the response.

Winning CL Counter: This starts with a count of 0 and advances the count by 1 each time a CL is found that results in the return of the test JKFFs to the all-zero state from the all-zero state in P units of time. After 2047 winning CLs are found the Counter signals the Timer and Controller to discontinue output of winning CLs to the PDP-10.

Coder: Information must be sent through the Execuport to the PDP-10 packed in the form

start bit - 7 bits of information - parity bit - stop bit.

The Coder converts the 10-digit base-9 number in the CL Memory into five 7-bit numbers. Each 7-bit number is passed to the Outputter, surrounded by the proper bits, and transmitted as a print-character that corresponds to one of the eighty-one possible connections of the inputs of a JKFF.

Outputter: At a signal from the Timer and Controller, this transmits coded winning CLs through the Execuport to the PDP-10. A 300 bit/second clock from the Execuport insures synchronization of transmission and reception of this data. The Outputter surrounds each 7-bit package of information with start, parity, and stop bits. The Outputter separates each 10-bit pack of information by at least five "pause bits". This pause insures that the PDP-10 input facilities are ready to handle each transmitted character.

Timer and Controller: This coordinates the processes of the SPM. When the On-off Switch is turned On, it initializes the CL Proposer and Winning CL Counter. It then regulates searching and outputting. When the search is over, it turns the Finish Light on.

During the search, the Timer and Controller regulates several parallel processes. The CL Proposer proposes CLs until it finds a nonredundant one that the CL Memory cannot accept because of a CL test in progress. When this test is finished the new CL is transferred to the CL Memory and the CL Proposer continues. Test of a CL begins with the transfer of a new CL to the CL Memory and the setting of the Test JKFFs to the all-zero start-state. The Test JKFFs receive timing pulses until one of two conditions occurs.

1) The number of timing pulses sent to the Test JKFFs equals the input period.

2) The Test JKFFs return to the all-zero state.

If both (1) and (2) are true, a winning CL has been found. The Winning CL Counter is advanced. If its count is less than 2048 and the Outputter is ready, the coded CL is sent to the Outputter for transmission to the PDP-10. If the Outputter is busy transmitting a winning CL the new winning CL remains in the CL Memory until the Outputter is ready for it. The maximum slowing of SPM runtime by output would occur if the first 2047 CLs proposed were winners. Runtime with the resultant output would exceed the runtime of the same search with no output by $2047 \times \frac{1}{4}$ second $\approx$ 512 seconds. If only one of conditions (1) and (2) above is true, the tested CL is a loser. The test is discontinued and the CL Memory waits for the next nonredundant CL.

When the CL Proposer reaches the count 8800000000, the Timer and Controller discontinues the searching and turns the Finish Light on. Counting from 8800000000 to 8888888888 is unnecessary because the only nonredundant CL in this range is 8888888888, which corresponds to a connection of all inputs to a constant 1.

PDP-10: This stores the winning CLs. When a run is finished the winning CLs are filed on a disk and given a mnemonic name. Software later examines this file, and outputs for each CL data in the form

(Period Safe? CL) .

The "period" entry allows a check of all CLs in the file. This was always the same as the one predicted by the input to the SPM. The "safe" entry tells whether the CL results in a safe ASJKC. Following this entry is the CL in its uncoded form. Software also makes other tests that we will describe of some files.

C. Runtime Statistics

Variations in runtime for different 5,P ASJKC tests are due to variations in the time to test nonredundant CLs and the time to output winning CLs. The statistics below demonstrate this fact.

Period	Runtime	Number of Winning CLs Output
18	1140 seconds	2047
25	815 "	91
32	880 "	0

Runtime is slowed when a winning CL must wait in the CL Memory for another winning CL to be output. This occurs when winning CLs are found by the SPM within $\frac{1}{4}$ second of each other. This high density of winning CLs occurred for P = 18 and resulted in its long runtime compared to P = 25 or 32. For the P = 18 case, the last of the first 2047 winning CLs was output after .44 of the total CLs proposed had been proposed. This first portion of the SPM run took twelve minutes, and the rest of the run took seven minutes.

When output does not cause a bottleneck, variations in runtime are caused by variations in the time to test nonredundant CLs. Because one of the two conditions for ending one of these tests is that the Test JKFFs receive P timing pulses, tests take longer for higher P. This is demonstrated by the fact that the runtime for 5,25 ASJKCs was less than that for 5,32 ASJKCs. It's difficult to predict the precise effect of varying P because the proposal and testing times overlap. If the maximum overlap had occurred in which each nonredundant CL was followed by 44 redundant ones, no variation in runtime would have occurred for P = 25 and 32. This maximum overlap obviously did not occur.

III EXPERIMENTAL RESULTS

This section discusses data resulting from the experiment described in the previous section and earlier work by Robert Fenichel. There is too much data to make presentation of all of it feasible or worthwhile. Instead the most important aspects of the data are presented.

The following table shows the number of winning CLs found for each 5,P ASJKC test for $13 \leq P \leq 32$. Because the proportion of equivalent counters varies for each period, this table only gives a rough idea of the relative number of non-equivalent counters. As we've seen, an n-ASJKC corresponding to a particular CL may be started in any one of 2^n start-states. Each could result in different behavior. Therefore each one or its equivalent should be tested. However in cycling with period P, an ASJKC enters P different states with the same CL. Each of these P combinations of CL and state is equivalent to the combination of some CL with the all-zero start-state. Two combinations of CL and state may be equivalent to the same CL with the all-zero state. This is true for CL = (Q2 $\overline{Q2}$ Q1 $\overline{Q1}$), S(1) = (0 0) or S(1) = (1 1). This equivalence usually does not occur. If all the all-zero start-state CLs are different, the 5,P ASJKC search will find P winning CLs equivalent to the same CL in P different start-states. Each of these CLs may have permutationally redundant CLs which are not eliminated by the redundancy test and which therefore appear as winning CLs. Therefore there is not an exact correspondence between the number of winning CLs found and the number of different CLs with at least P start-states leading to periodicity P. In fact, one would expect a smaller proportion of different CLs in tests for higher P. This biasing of the count could have been eliminated by software for files with fewer than 2048 winning CLs. We only used this approach for four files because we were not interested enough in the precise data to make extensive use of software analysis of files worth the effort.

Period	Number of Winning CLs
13	18,222
14	41,938
15	22,999
16	12,020
17	1,775
18	4,763
19	479
20	3,569
21	1,395
22	432
23	468
24	508
25	91
26	176
27	159
28	594
29	90
30	1,388
31	968
32	0
Total	112,034

NUMBER OF WINNING CLs FOR 5,P ASJKC SEARCHES

This data suggests certain general statements. There are more winning CLs for low periods than for high periods. When a period has some winning CLs which correspond to composite ASJKCs with components of three or fewer JKFFs, that period has more winning CLs than the period near it. This is true for P = 14, 15, 18, 20, 21, 24, 28, and 30.

There is a large number of winning CLs for P = 31 and no winning CLs for P = 32.

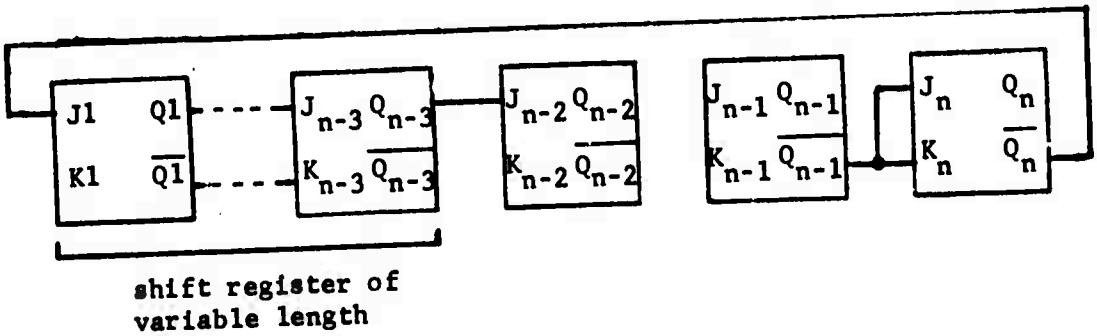
Software analysis indicates the effect of the fact that P different start-states of a winning CL result in periodicity P. Software analyzed the files for P = 25, 26, 27, and 29 -- the four smallest files with some CLs. The table below shows the different counters for these periods by showing a CL which results in one of the counters when associated with the all-zero start-state.

Period	Number of Winning CLs Output	Number of Different Counters	Different Counters
25	91	1	$(\overline{Q5}\ \overline{Q4}\ Q1\ \overline{Q1}\ Q2\ Q2\ Q3\ \overline{Q2}\ \overline{Q4}\ \overline{Q4})$
26	176	2	$(\overline{Q5}\ \overline{Q4}\ Q1\ \overline{Q1}\ Q2\ \overline{Q2}\ Q3\ \overline{Q3}\ \overline{Q4}\ \overline{Q4})$
			$(\overline{Q5}\ Q4\ Q1\ \overline{Q1}\ Q2\ Q2\ Q3\ \overline{Q2}\ \overline{Q4}\ \overline{Q4})$
27	159	2	$(\overline{Q5}\ Q4\ Q1\ \overline{Q1}\ Q2\ Q2\ \overline{Q3}\ Q2\ \overline{Q4}\ \overline{Q4})$
			$(\overline{Q5}\ Q3\ Q1\ \overline{Q1}\ Q2\ \overline{Q2}\ Q3\ \overline{Q3}\ \overline{Q4}\ \overline{Q4})$
29	90	1	$(\overline{Q5}\ Q4\ Q1\ \overline{Q1}\ Q2\ \overline{Q2}\ Q3\ \overline{Q3}\ \overline{Q4}\ \overline{Q4})$
	516	6	

ASJKCs FOR P = 25, 26, 27, 29

The CLs in the table above are remarkably similar. Six inputs (J1, J2, K2, J3, J5, K5) have the same connection in all six CLs. Two inputs (K3, J4) each have one of two connections. Two inputs (K1, K4) each have one of three connections. Software examined all ($2 \times 2 \times 3 \times 3 = 36$) possible connections of K3 to Q2 or $\overline{Q2}$, J4 to Q3 or $\overline{Q3}$, K1 to Q3 or Q4 or $\overline{Q4}$, K4 to Q2 or $\overline{Q2}$ or $\overline{Q3}$, and the other inputs to the same outputs as in the table above. Twelve CLs resulted in ASJKCs

that did not return to the all-zero state from the all-zero state. We did not examine the periodicity of these ASJKCs. The other twenty-four did return to the all-zero state with periods of 4, 7, 8, 9, 10, 13, 16, 18, 19, 20, 21, 24, 25, 26, 27, and 29. It would be interesting to study generalizations of this ASJKC. For instance, what is the effect of insertion of shift-registers of varying length between the first and second JKFFs of this ASJKC. This implies a CL of the form $(\overline{Q_n}\ W\ Q1\ \overline{Q1}\ Q2\ \overline{Q2}\ \ldots\ Q_{n-4}\ \overline{Q_{n-4}}\ Q_{n-3}\ X\ Y\ Z\ \overline{Q_{n-1}}\ \overline{Q_{n-1}})$, where W is Q_{n-2} or Q_{n-1} or $\overline{Q_{n-1}}$, X is Q_{n-3} or $\overline{Q_{n-3}}$, Y is Q_{n-2} or $\overline{Q_{n-2}}$, and Z is Q_{n-3} or $\overline{Q_{n-3}}$ or $\overline{Q_{n-2}}$.



FIXED CONNECTIONS FOR A GENERAL ASJKC

The files for P = 25, 26, 27 and 29 represent 160 (= 25 + (2 x 26) + (2 x 27) + 29) different combinations of CL and state. If we make the reasonable assumption that no two of these combinations are equivalent to the same CL with the all-zero start-state, then the average number of permutationally equivalent winning CLs found for these files was 516/160 = 3.2. This is close to the estimated average of all permutationally equivalent CLs which are tested, which an interpretation of Appendix B shows is about 2.7 if the approximation is made that every proposed CL is permutationally equivalent to 119 different CLs.

The table on the following page can be a useful guide to the logic designer. The entry in the "Minimum ASJKC" column gives the CL of what our exhaustive search proves is the n,P ASJKC with the smallest n. If there is a safe minimum ASJKC it appears in the "Minimum ASJKC" column, and a ← in the "Minimum Safe ASJKC" column indicates this. If the minimum ASJKC is not safe, an unsafe one is entered in the "Minimum ASJKC" column and a safe x,P ASJKC with the smallest x is entered in the "Minimum Safe ASJKC" column. CLs are in the form (J1 K1 .... JN KN) and all CLs give appropriate behavior when started in the all-zero start-state. To the right of each CL is an entry which indicates whether that CL corresponds to a minimal ASJKC. For P = 7 the minimum ASJKC is minimal and unsafe. Because it is unsafe the minimum safe ASJKC appears in the appropriate column. This minimum safe ASJKC is not minimal. When a (a x b x ...z) appears before a CL, that CL represents a composite ASJKC with components of period a, b, ...z. For instance, for P = 6 the minimum ASJKC shown is a composite ASJKC with components of period 2 and 3. A "?" indicates that a particular entry is unknown. No example of that entry exists for $n \leq 5$.

Period	A Minimum ASJKC	Minimal ?	A Minimum Safe ASJKC	Minimal ?
2	(1 1)	Yes	←	Yes
3	(Q2 1 $\overline{Q1}$ 1)	Yes	←	Yes
4	($\overline{1}$ 1 Q1 Q1)	Yes	←	Yes
5	(Q2 1 Q3 1 $\overline{Q1}$ 1)	Yes	←	Yes
6	(2 x 3) (1 1 Q3 1 $\overline{Q2}$ 1)	Yes	←	Yes
7	(Q3 Q3 $\overline{Q1}$ Q1 Q2 $\overline{Q2}$)	Yes	(1 Q4 1 Q3 1 Q1 Q2 Q1)	No
8	(Q3 Q3 $\overline{Q3}$ $\overline{Q1}$ Q2 $\overline{Q2}$)	Yes	←	Yes
9	($\overline{Q3}$ Q4 Q1 Q1 $\overline{Q2}$ $\overline{Q1}$ Q3 Q1)	Yes	←	Yes
10	(2 x 5) (1 1 $\overline{Q3}$ 1 Q4 1 $\overline{Q2}$ 1)	Yes	←	Yes
11	(1 Q2 Q1 Q4 Q2 Q2 Q3 1)	Yes	←	Yes
12	(3 x 4) (Q2 1 $\overline{Q1}$ 1 1 $\overline{1}$ Q3 $\overline{Q3}$)	Yes	←	Yes
13	(Q3 $\overline{Q3}$ $\overline{Q1}$ Q1 Q2 $\overline{Q4}$ Q2 $\overline{Q5}$ Q4 $\overline{Q2}$)	No	←	Yes
14	(2 x 7) (1 1 Q4 $\overline{Q4}$ $\overline{Q2}$ Q2 Q3 $\overline{Q3}$)	Yes	(2 x 7) (1 1 1 Q5 1 Q4 1 Q2 Q3 Q2)	No
15	(Q4 Q4 Q1 $\overline{Q1}$ Q2 $\overline{Q2}$ $\overline{Q3}$ Q3)	Yes	(3 x 5) (Q2 1 $\overline{Q1}$ 1 $\overline{Q4}$ 1 Q5 1 $\overline{Q3}$ 1)	No
16	(1 1 1 Q3 1 Q5 Q3 Q3 Q2 Q1)	No	←	No
17	(1 Q2 Q3 $\overline{Q5}$ Q1 $\overline{Q4}$ $\overline{Q3}$ Q1 $\overline{Q2}$ Q1)	Yes	←	Yes
18	(2 x 9) (1 1 $\overline{Q4}$ $\overline{Q5}$ Q2 $\overline{Q2}$ $\overline{Q3}$ Q2 $\overline{Q4}$ Q2)	Yes	←	Yes
19	(Q3 Q4 Q5 Q1 $\overline{Q4}$ $\overline{Q4}$ Q5 Q2 $\overline{Q2}$ Q1)	Yes	←	Yes
20	(4 x 5) (1 1 Q1 Q1 $\overline{Q4}$ 1 Q5 1 $\overline{Q3}$ 1)	Yes	←	Yes
21	(3 x 7) (Q2 1 $\overline{Q1}$ 1 Q5 Q5 $\overline{Q3}$ Q3 Q4 $\overline{Q4}$)	Yes	(3 x 7) (Q2 1 $\overline{Q1}$ 1 1 Q6 1 Q5 1 Q3 Q4 Q3)	No
22	(2 x 11) (1 1 1 Q3 Q2 Q5 Q3 Q3 $\overline{Q4}$ 1)	Yes	←	Yes
23	(Q5 1 Q1 Q1 Q2 Q2 Q3 Q3 $\overline{Q4}$ 1)	Yes	?	No
24	(3 x 8) (Q2 1 $\overline{Q1}$ 1 Q5 Q5 $\overline{Q5}$ $\overline{Q3}$ Q4 $\overline{Q4}$)	Yes	←	Yes
25	($\overline{Q5}$ $\overline{Q4}$ Q1 $\overline{Q1}$ Q2 $\overline{Q2}$ Q3 $\overline{Q2}$ $\overline{Q4}$ $\overline{Q4}$)	Yes	?	No
26	($\overline{Q5}$ $\overline{Q4}$ Q1 $\overline{Q1}$ Q2 $\overline{Q2}$ Q3 Q3 $\overline{Q4}$ $\overline{Q4}$)	Yes	(2 x 13) (1 1 Q4 $\overline{Q4}$ $\overline{Q2}$ Q2 Q3 $\overline{Q5}$ Q3 $\overline{Q6}$ Q5 $\overline{Q3}$)	No
27	($\overline{Q5}$ Q3 Q1 $\overline{Q1}$ Q2 $\overline{Q2}$ $\overline{Q3}$ Q3 $\overline{Q4}$ $\overline{Q4}$)	Yes	?	No
28	(4 x 7) (1 1 Q1 Q1 Q5 Q5 $\overline{Q3}$ Q3 Q4 $\overline{Q4}$)	Yes	(4 x 7) (1 1 Q1 Q1 1 Q6 1 Q5 1 Q3 Q4 Q3)	No
29	($\overline{Q5}$ Q4 Q1 $\overline{Q1}$ Q2 $\overline{Q2}$ Q3 $\overline{Q3}$ $\overline{Q4}$ $\overline{Q4}$)	Yes	?	No
30	(2 x 15) (1 1 Q5 Q5 $\overline{Q2}$ $\overline{Q2}$ Q3 $\overline{Q3}$ $\overline{Q4}$ Q4)	Yes	(2 x 3 x 5) (1 1 Q3 1 $\overline{Q2}$ 1 $\overline{Q5}$ 1 Q6 1 $\overline{Q4}$ 1)	No
31	($\overline{Q5}$ Q5 Q1 Q1 $\overline{Q2}$ Q2 Q3 Q3 Q4 Q4)	Yes	?	No
32	?	No	?	No

MINIMUM SAFE AND UNSAFE ASJKCs FOR $2 \leq P \leq 32$

The table indicates that there is no safe or unsafe minimal ASJKC for P = 13, 16, or 32. The absence of an $n,2^n$ ASJKC for $n = 4$ and 5 suggests the conjecture that there is no $n,2^n$ ASJKC for $n \geq 4$. We have not been able to prove that this is true. The table also shows that there is no minimal, safe ASJKC for P = 7, 13, 14, 15, 16, 21, 23, 25, 26, 27, 28, 29, 30, 31, or 32. As one proof in Section IV shows, there is no safe n, 2^n-1 ASJKC for $n \geq 3$.

A composite ASJKC of period P may be formed from component ASJKCs in the table in the following way.

1) Factor P into prime factors and associate with each prime factor F a number t equal to the number of times F occurs.

$$P = (F_1)^{t_1} \times (F_2)^{t_2} \times \ldots (F_2)^{t_2}.$$

2) The composite ASJKC may be realized by any set of component ASJKCs such that each $(F)^t$ is a factor of at least one component's period and each component's period P_c is of the form

$$P_c = (F_1)^{x_1} \times (F_2)^{x_2} \times \ldots \times (F_2)^{x_2} \text{ with } 0 \leq x_n \leq t_n \text{ and } 1 < P_c < P.$$

There may be no set of component ASJKCs that satisfies these conditions, there may be one set, or there may be more than one set. When there is more than one set, choice of a particular set depends on other considerations. For instance, a set with the fewest JKFFs or one that results in a safe ASJKC may be chosen. For example, consider forming a composite ASJKC with P = 60.

1) $60 = (2)^2 \times 3 \times 5$

2) Possible sets of composite ASJKCs are (4 x 15), (5 x 12), (3 x 20). The unsafe (4 x 15) ASJKC uses fewer JKFFs than the (5 x 12) or the (3 x 20), so it might be the most desirable choice for a particular application. The table on the next page shows some composite ASJKCs for $33 \leq P \leq 64$.

Period	Composite ASJKC	Period	Composite ASJKC
33	(3 x 11)	49	--
34	(2 x 17)	50	(2 x 25)
35	(5 x 7)	51	(3 x 17)
36	(4 x 9)	52	(4 x 13)
37	--	53	--
38	(2 x 19)	54	(2 x 27)
39	(3 x 13)	55	(5 x 11)
40	(5 x 8)	56	(7 x 8)
41	--	57	(3 x 19)
42	(2 x 21)	58	(2 x 29)
43	--	59	--
44	(4 x 11)	60	(4 x 15)
45	(5 x 9)	61	--
46	(2 x 23)	62	(2 x 31)
47	--	63	(7 x 9)
48	(3 x 16)	64	--

SOME COMPOSITE ASJKCs FOR $33 \leq P \leq 64$

IV PROOFS

Inquiry into questions rising from this study led to the proofs presented in this section. These proofs use a mathematical approach very different from the experimental apprcach of the last two sections. All of the proofs use the specification of the particular ASJKC discussed to make conclusions about the necessary inputs to the JKFFs. The fact that each input to a JKFF may only be provided by the constant 1 or one of the outputs of the other JKFFs in the ASJKC helps lead to the conclusion reached.

Two of the proofs use the fact that if the output of a JKFF in a ASJKC is constant x times, and then changes, there must be at least x JKFFs in the ASJKC. If QA = 0 from t = 1 to t = x and then QA = 1 at t = x + 1, JA = 0 from t = 1 to t = x - 1 and then JA = 1 at t = x. If QA = 1 from t = 1 to t = x and then QA = 0 at t = x + 1, KA = 0 from t = 1 to t = x - 1 and then KA = 1 at t = x. For $x \geq 2$, one of the outputs of another JKFFB is necessary to provide this input. But this needs one of its inputs 0 from t = 1 to t = x - 2, then 1 at t = x - 1. This line of reasoning continues demanding more JKFFs until we get to a JKFF that has an output that changes from t = 1 to t = 2. This implies one of its inputs is 1 at t = 1, and this requirement can be fulfilled by any of the JKFFs in the ASJKC or the constant 1. This argument implies that if a JKFF is constant x times in a row and then changes, there must be at least x JKFFs in the ASJKC to provide necessary inputs.

Some proofs also use the fact that symmetrically equivalent ASJKCs cycle with the same period. The proofs are much clearer when we assume a relabelling of the JKFFs of an ASJKC that results in behavior that includes or excludes a particular state.

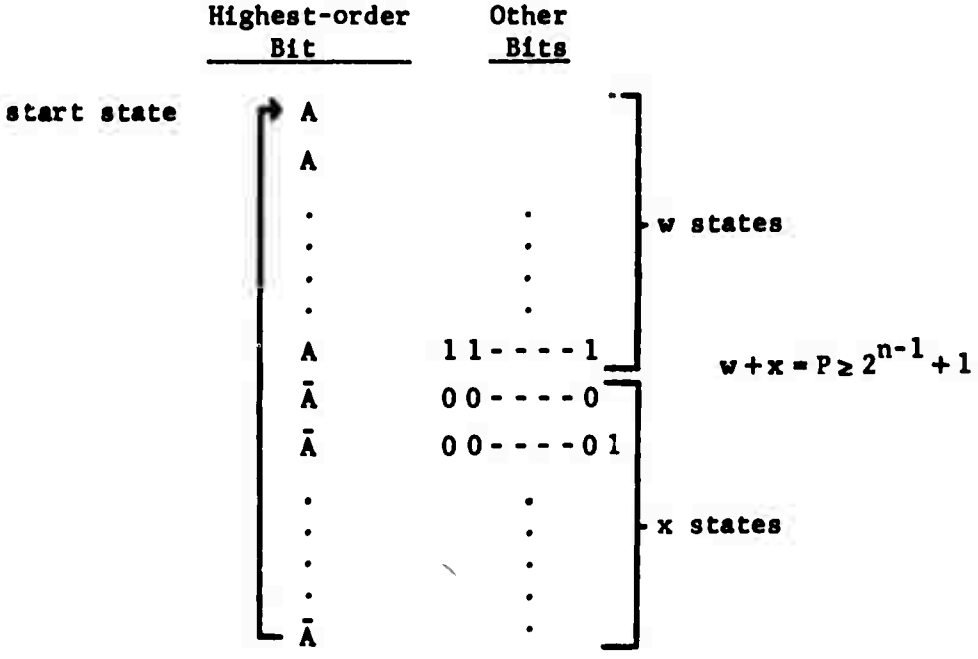
Assertion. There is no minimal binary n-ASJKC with $n > 3$.

Proof: Assume there is such a counter. Because it is minimal, $2^{n-1} + 1 \leq P \leq 2^n$. Because it is binary, the highest-order bit changes only if one of two conditions occurs.

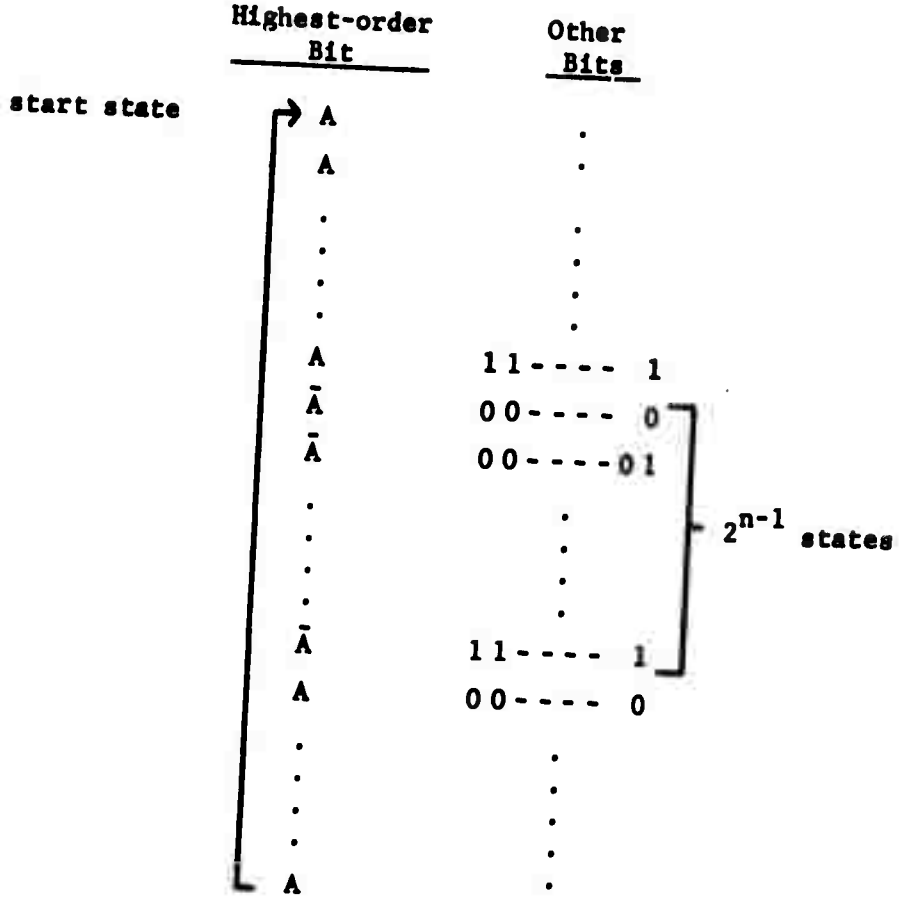
1) All the lower-order bits are 1. This occurs at most two times during one cycle = P successive states.

2) The final state changes to the start state. This occurs once each cycle and may or may not imply the change of the highest-order bit.

Two mutually exclusive, collectively exhaustive possibilities for the binary counter of period P are shown below.



CASE A: Highest-order bit different in start state, final state.



CASE B: Highest-order bit same in start state, final state.
TWO POSSIBILITIES FOR A MINIMAL BINARY ASJKC

Let g equal the greatest number of times in a row that the highest-order bit is constant during one cycle. For Case A, $g \geq 2^{n-2} + 1$. For Case B, $g = 2^{n-1}$. For $n > 3$, $2^{n-1} > 2^{n-2} + 1$. Therefore for $n > 3$, $g \geq 2^{n-2} + 1$. Because the highest-order bit is constant g times in a row and then changes, the ASJKC must have at least g JKFFs. Therefore $n \geq g \geq 2^{n-2} + 1$, $n > 3$. Because this statement is never true, a minimal binary n-ASJKC with $n > 3$ is logically contradictory. It can't exist.

Appendix A shows how w JKFFs can cycle with a period P through any sequence of their 2^w possible states if they are part of a (P x w), P ASJKC. In particular, some of the JKFFs of a large enough ASJKC can cycle through any binary or Gray sequence of states we can name. Fewer than (P x w) JKFFs are often adequate to realize a certain behavior. The necessary JKFFs can be found if one can find the necessary inputs to each of the w JKFFs. Appendix A explains this further.

The proof about Gray ASJKCs presented below is broken into two parts. The first part of the proof is for Gray n-ASJKCs with $n \geq 2$. It is shown that any such ASJKC that uses all its JKFFs is symmetrically equivalent to a switch-tailed shift-register or one of its permutational equivalents. The statement that the ASJKC must use all its JKFFs disallows ASJKCs in which some of the JKFFs never change state. The second part of the proof shows that any Gray counter with just 1 JKFF is equivalent to a switch-tailed shift-register.

Assertion: Any Gray ASJKC which uses all its JKFFs is equivalent to a switch-tailed shift-register and therefore has a period $P = 2n$.

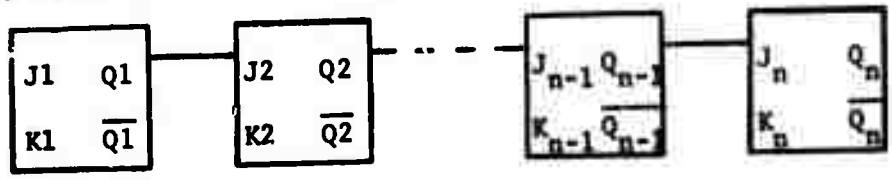
Proof: The statement above is true if it is true for n-ASJKCs with $n \geq 2$ and $n = 1$.

Part_1_Assertion: Any Gray n-ASJKC with $n \geq 2$ is symmetrically equivalent to a switch-tailed shift-register or one of its permutational equivalents and has a period $P = 2n$.

Part_1_Proof: Assume a Gray n-ASJKC with $n \geq 2$. Because of the symmetry property its JKFFs can be labelled so that the Gray ASJKC starts in the all-zero state. This does not affect the Grayness or period of the counter. Since by definition only one JKFF can change at a time, the last JKFF to change -- Q_n -- is 0 at least from $t = 1$ to $t = n$. Q_n must change at some time to be useful. In fact, it must change at $t = n + 1$.

If Q_n is 0 x times in a row and then 1, there must be at least x JKFFs in the ASJKC. Since there are only n JKFFs, Q_n must change to 1 at t = n + 1. If we label the ASJKC so that Q1 changes first, Q2 changes second, etc., then this necessarily implies the sequence of states and partial ASJKC shown below.

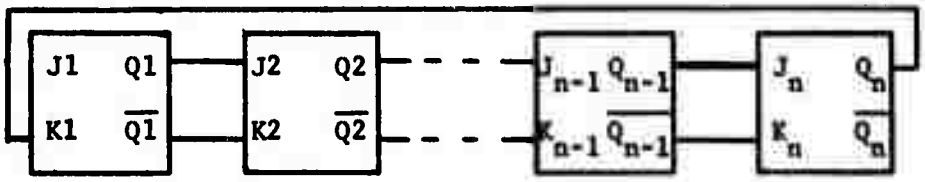
Time	Q_n	Q_{n-1}	Q_{n-2}	. . . . .	Q3	Q2	Q1
t = 1	0	0	0		0	0	0
t = 2	0	0	0		0	0	1
	0	0	0		0	1	1
	.	.	.		1	.	.
	.	.	.			.	.
	.	.	.		.	.	.
	.	.	.		.	.	.
	0	0	1		1	1	1
t = n	0	1	1		1	1	1
t = n+1	1	1	1		1	1	1



Only this counter or a permutationally equivalent one could result in the proven behavior of $Q_n(t)$ for $1 \leq t \leq n + 1$.

Because Q1(t) = 1 for $2 \leq t \leq n + 1$, K1(t) = 0 for $2 \leq t \leq n$. For $n \geq 2$, this implies that K1 connects to Q_n. Similarly because K2 must be low from t = 3 to t = n + 1, it must connect to $\overline{Q1}$. This argument applies for all $x \geq 2$, so that K_x connects to $\overline{Q_{n-1}}$. We've now specified the sequence of states and partial counter shown on the next page.

Time	Q_n	Q_{n-1}	Q_{n-2}	. . . . .	Q2	Q1
t = 1	0	0	0	. . . . .	0	0
t = 2	0	0	0	. . . . .	0	1
	0	0	0	. . . . .	1	1
	.				.	.
	.				.	.
	.				.	.
t = n+1	1	1	1	. . . . .	1	1
t = n+2	1	1	1	. . . . .	1	0
t = n+3	1	1	1	. . . . .	0	0
	.	.	.		.	.
	.	.	.		.	.
	.	.	.		.	.
	1	0	0	. . . . .	0	0



Because Q1(t) = 0 for $n + 2 \leq t \leq 2n + 1$ and because $n \geq 2$, J1(t) = 0 for $n + 2 \leq t \leq 2n$. Therefore J1 connects to $\overline{Q_n}$. Therefore the ASJKC is a switch-tailed shift-register. Any Gray n-ASJKC with $n \geq 2$ that uses all its bits is therefore symmetrically equivalent to a switch-tailed shift-register or one of its permutational equivalents. We've also shown that this implies P = 2n.

Part_2_Assertion: Any Gray 1-ASJKC is equivalent to a switch-tailed shift-register and has a period P = 2.

Part_2_Proof: For the one JKFF to count in Gray Code, it must change state each time. Therefore J connects to 1 or $\bar{Q}$, and K connects to 1 or Q. Section II-A shows that connection of J to 1 is equivalent to connection of J to $\bar{Q}$ and connection

of K to 1 is equivalent to connection of K to Q. Therefore the Gray 1-ASJKC is equivalent to a switch-tailed shift-register. It has a period of $2 \times 1 = 2$.

Because the major assertion is true for $n = 1$ and true for $n \geq 2$, it is true for all n. A corollary of this proof is that there is no minimal Gray n-ASJKC for $n \geq 4$.

Assertion: Any n, $2^n - 1$ ASJKC with $n > 2$ is unsafe.

Proof: Assume a n, 2^n-1 ASJKC. During its cycle through $2^n - 1$ states there is one state it doesn't reach. For clarity use the symmetry property to label the JKFFs of the ASJKC to make this the all-zero state. This does not affect the period or safety of the counter. Then for all x, $Q_x = 0$ $2^{n-1} - 1$ times, $Q_x = 1$ 2^{n-1} times.

If $Q_x = 1$, K_x determines its next state. When $Q_x = 1$, there are two mutually exclusive, collectively exhaustive connections of K_x that we'll consider.

1) K_x connects to a Q or $\bar{Q}$. Because the all-zero state is the only one not entered during a cycle, in one cycle

$K_x(t) = 0, Q_x(t) = 1 \rightarrow Q_x(t+1) = 1$ $\quad 2^{n-2}$ times,

$K_x(t) = 1, Q_x(t) = 1 \rightarrow Q_x(t+1) = 0$ $\quad 2^{n-2}$ times.

2) K_x connects to 1. During one cycle

$K_x(t) = 1, Q_x(t) = 1 \rightarrow Q_x(t+1) = 0$ $\quad 2^{n-1}$ times.

If $Q_x = 0$, K_x determines its next state. When $Q_x = 0$ there are three mutually exclusive, collectively exhaustive connections of J_x that we'll consider.

1) J_x connects to a Q. During one cycle

$J_x(t) = 0, Q_x(t) = 0 \rightarrow Q_x(t+1) = 0$ $\quad 2^{n-2} - 1$ times,

$J_x(t) = 1, Q_x(t) = 0 \rightarrow Q_x(t+1) = 1$ $\quad 2^{n-2}$ times.

2) J_x connects to a $\bar{Q}$. During one cycle

$J_x(t) = 0,\ Q_x(t) = 0 \rightarrow Q_x(t+1) = 0$ $\quad 2^{n-2}$ times,

$J_x(t) = 1,\ Q_x(t) = 0 \rightarrow Q_x(t+1) = 1$ $\quad 2^{n-2} - 1$ times.

3) J_x connects to 1. During one cycle

$J_x(t) = 1,\ Q_x(t) = 0 \rightarrow Q_x(t+1) = 1$ $\quad 2^{n-1} - 1$ times.

There are six possible combinations of the categories above.

1) $K_x = 1,\ J_x = 1 \rightarrow Q_x(t+1) = 0$ $\quad 2^{n-1}$ times;

2) $K_x = 1,\ J_x = Q \rightarrow Q_x(t+1) = 0$ $\quad 2^{n-1} + 2^{n-2} - 1$ times;

3) $K_x = 1,\ J_x = \bar{Q} \rightarrow Q_x(t+1) = 0$ $\quad 2^{n-1} + 2^{n-2}$ times;

4) $K_x \neq 1,\ J_x = 1 \rightarrow Q_x(t+1) = 0$ $\quad 2^{n-2}$ times;

5) $K_x \neq 1,\ J_x = Q \rightarrow Q_x(t+1) = 0$ $\quad 2^{n-2} + 2^{n-2} - 1$ times;

6) $K_x \neq 1,\ J_x = \bar{Q} \rightarrow Q_x(t+1) = 0$ $\quad 2^{n-2} + 2^{n-2}$ times.

Because the all-zero state is avoided during each cycle, $Q_x(t + 1) = 0$ $2^{n-1} - 1$ times each cycle. Therefore the six categories above imply the six equations below.

Category	Equation	Occurs When?
1)	$2^{n-1} - 1 = 2^{n-1}$	Never
2)	$2^{n-1} - 1 = 2^{n-1} + 2^{n-2} - 1$	Never
3)	$2^{n-1} - 1 = 2^{n-1} + 2^{n-2}$	Never
4)	$2^{n-1} - 1 = 2^{n-2}$	$n = 2$
5)	$2^{n-1} - 1 = 2^{n-2} + 2^{n-2} - 1$	All n
6)	$2^{n-1} - 1 = 2^{n-2} + 2^{n-2}$	Never

These equations show that in a n, 2^n - 1 ASJKC with n > 2 each J must connect to a Q. Therefore the ASJKC remains in the all-zero state after starting there. This proves that a n, 2^n - 1 ASJKC with n > 2 is unsafe.

The proof below reduces by one the number of useful inputs to any JKFF in a n, P ASJKC with odd period $P > 1.5 \times 2^{n-1}$. This fact could be used to reduce the size of searches for these ASJKCs by a machine like ours.

Assertion: The constant 1 is useless as an input for n, P ASJKCs with odd period $P > 1.5 \times 2^{n-1}$.

Proof: Consider a ASJKC in which the J of one ASJKC connects to the constant 1. There are two mutually exclusive, collectively exhaustive connections of KA that we'll consider.

1) KA connects to the constant 1. Then JKFFA changes state each time. Therefore the ASJKC must have an even period.

2) KA connects to one of the outputs of one of the other JKFFs. Because JA = 1, each time QA = 0 it changes to QA = 1 at the next instant of time. Therefore QA only becomes 0 after QA = 1, KA = 1. Because K connects to one of the outputs, this can occur at most 2^{n-2} times during one cycle. Therefore during one cycle QA = 0 at most 2^{n-2} times. P equals the number of times QA = 0 during a cycle plus the number of times QA = 1 during a cycle. Therefore $P \leq (2^{n-2} + 2^{n-1} = 1.5 \times 2^{n-1})$.

Because the period of the ASJKC is either even with JA = KA = 1 or less than $1.5 \times 2^{n-1}$ for JA = 1, KA ≠ 1, JA can't connect to the constant P if the period is odd and greater than $1.5 \times 2^{n-1}$. Because of the symmetry property, this is also true for KA. Therefore our proof is complete.

V SUGGESTIONS FOR FURTHER RESEARCH

The three basic approaches described in this report -- hardware-oriented collection of data, software-aided analysis of data, and mathematical analysis -- can be extended for further study of ASJKCs.

Exhaustive study of 6, P ASJKCs for thirty-six of the periods in the range $21 \leq P \leq 64$ could lead to discovery of "minimum safe" or "minimum unsafe" entries for these periods. These entries are already known for the other periods in this range. If this study occurred in the near future it would have to use a SPM. Section II-A contains an approximation of the runtime of a machine with a basic cycle time of 165 nsec and no CL Memory. The estimated time to search all 6-ASJKCs for a particular period is 177 hours. Faster logic and less conservative design could cut this time to below 100 hours. The SPM could also use the fact that the constant 1 is useless for odd periods $P > 48$ to shorten tests for these periods. The SPM would have to run less than a total of 150 days to test all 36 periods. This is a long time but it isn't prohibitively long. The SPM is inexpensive and it can run unattended after it is started.

Lee's software approach might find a few unknown entries for n-ASJKCs with $n > 5$. The huge space to be searched and the sparseness of output for several periods for $n < 5$ suggests that this approach would miss many existing counters of interest.

Software could also help study generalizations of ASJKCs found during this research. For instance, the configuration mentioned in Section IV could be studied. Some useful ASJKCs or proofs about these general configurations might result from this approach.

Mathematical analysis of ASJKCs is an interesting approach that can be extended. This has taken the form of proofs concerning the limits of ASJKCs. This work could be continued. For instance, a proof proving or disproving the validity of the conjecture that there is no $n, 2^n$ ASJKC for $n \geq 4$ would be interesting and also helpful to the experimenter tempted to search for such a counter. Proofs showing how to synthesize certain types of ASJKCs would also be worthwhile.

VI SUMMARY

The report described research into some properties of autonomous, synchronous counters constructed with only the simplest form of J-K Flip-Flop. The study used a system in which a special-purpose digital machine and a PDP-10 general-purpose computer interacted. This system searched through all the counters of up to five J-K Flip-Flops for properties of interest. Description of some of the relevant design issues appears in the report.

Other information for the designer appears here. Useful counters of up to five J-K Flip-Flops are presented with techniques of using these for synthesis of other counters. A general method is given for synthesizing an autonomous, synchronous machine, made only of J-K Flip Flops, that moves through a specific sequence of states. Proofs show some constraints on a designer of these counters.

The four proofs resulted from analysis guided by the experimental results.

1) There is no minimal binary n-ASJKC with $n > 3$.

2) Any Gray ASJKC which uses all its JKFFs is equivalent to a switch-tailed shift-register and therefore has a period $P = 2n$.

3) Any n, $2^n - 1$ ASJKC with $n > 2$ is unsafe.

4) The constant 1 is useless as an input for n, P ASJKCs with $P > 1.5 \times 2^{n-1}$.

VII APPENDICES

A. Specific Sequences of States

A specific sequence of states for each JKFF of a counter implies knowledge of what input values are required for those JKFFs. Sometimes these inputs can be provided by the constant 1 or the outputs of the JKFFs of the counter. This is true for a minimal ASJKC. On the other hand, sometimes new JKFFs must be added to the ASJKC to provide inputs. These in turn may require new JKFFs to provide inputs for them. The number of new JKFFs needed depends on the particular sequence of states, but the maximum number needed can be calculated for certain general categories of operation.

Knowledge of the state of w JKFFs from t = 1 to t = y determines the input value at each time from t = 1 to t = y - 1 for one of the two inputs — J or K — for each JKFF. The input whose value is determined may change for each JKFF during this time. The rules for determining the inputs to a JKFF are

$$Q(t) = 0,\ Q(t+1) = 0 \rightarrow J(t) = 0;$$

$$Q(t) = 0,\ Q(t+1) = 1 \rightarrow J(t) = 1;$$

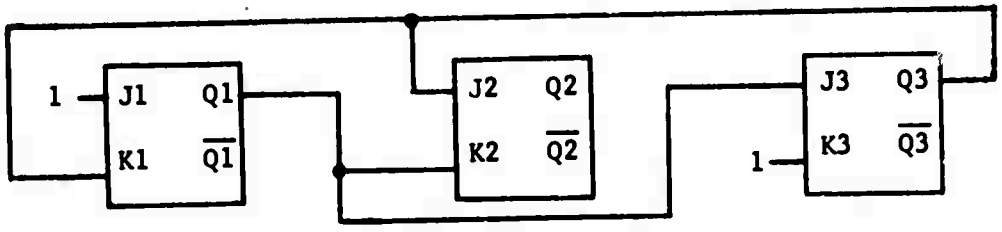
$$Q(t) = 1,\ Q(t+1) = 0 \rightarrow K(t) = 1;$$

$$Q(t) = 1,\ Q(t+1) = 1 \rightarrow K(t) = 0.$$

Sometimes the proper inputs to a JKFF can be supplied by one of the w JKFFs or the constant 1. When this isn't the case, use of only JKFFs demands the introduction of new JKFFs to supply appropriate inputs. For instance, consider the following sequence of five states for two JKFFs and the associated necessary inputs.

t	Q1	Q2	J1	K1	J2	K2
1	0	0	1	0 or 1	0	0 or 1
2	1	0	0 or 1	0	0	0 or 1
3	1	0	0 or 1	1	1	0 or 1
4	0	1	1	0 or 1	0 or 1	0
5	1	1				

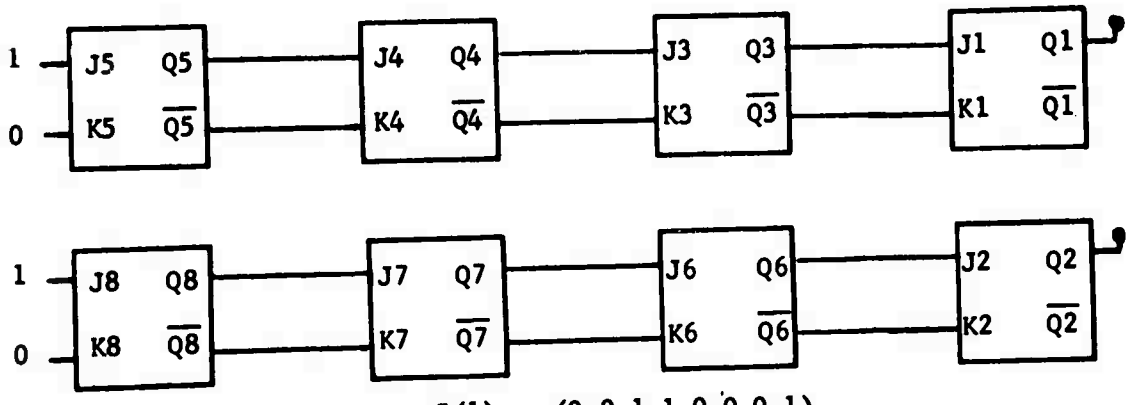
J1 can be supplied by the constant 1 and K2 can be supplied by Q1. There is no available input for K1 or J2 with the proper behavior. Therefore a new JKFF must be introduced if we insist on using only JKFFs in the counter. The new JKFF3 can provide the correct inputs for K1 and J2 if Q3(1) = 0, Q3(2) = 0, Q3(3) = 1, and K1 and J2 connect to Q3. If J3 connects to Q1 and K3 connects to anything, Q3 will behave properly. The counter below would therefore realize the appropriate behavior of Q1 and Q2 for t = 1 to t = 5.



S(1) = (0 0 0)

EFFICIENT WAY TO DRIVE JKFF1 AND JKFF2

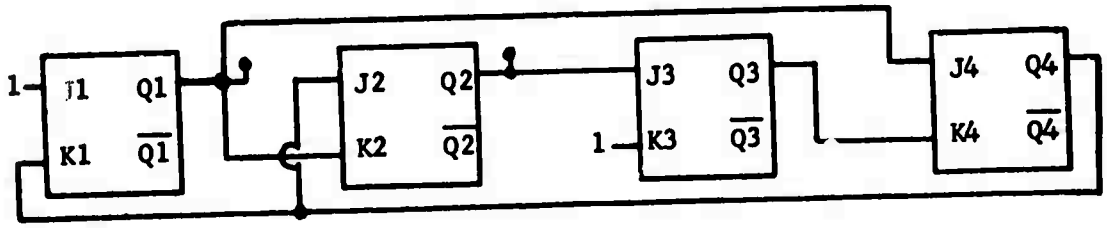
Whenever w JKFFs are required to go through some specific sequence of y states, a w(y - 1)-ASJKC in which w JKFFs are observed is always sufficient to realize the proper behavior. At worst a properly initialized (y - 2)-bit shift-register could provide the proper inputs for each of the w JKFFs from t = 1 to t = y - 1. The diagram below illustrates this general method for the sequence of five states of Q1 and Q2 already considered.



S(1) = (0 0 1 1 0 0 0 1)

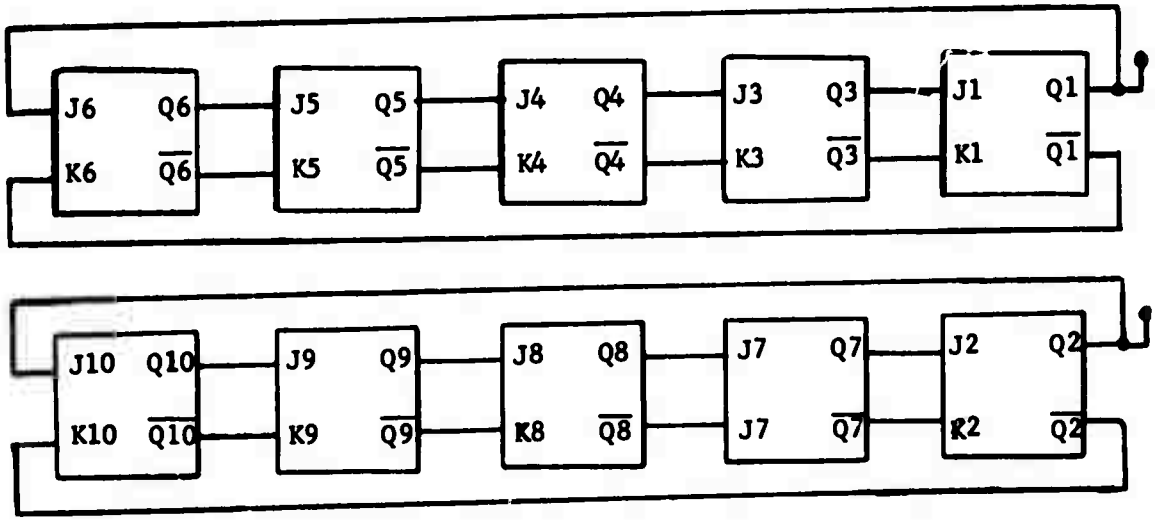
GENERAL METHOD FOR PROVIDING INPUTS

When w JKFFs cycle with a period P through P specific states either of the two methods already described can be used to provide inputs. One can carefully introduce new JKFFs only when they are determined to be necessary for providing inputs. On the other hand, at worst each of the w JKFFs can have its inputs provided by a properly initialized P-bit shift-register with its last bit inputting to its first bit. Therefore at worst a properly initialized $P \times w$ - ASJKC is sufficient to realize any periodic behavior with period P of w JKFFs. The two diagrams below indicate two realizations of two JKFFs cycling with period 5 through the five states of the preceding example.



S(1) = (0 0 0 0)

EFFICIENT WAY TO DRIVE JKFF1 AND JKFF2



S(1) = (0 0 1 1 0 1 0 0 1 1)

GENERAL METHOD FOR PROVIDING INPUTS

B. Effectiveness of Permutational Redundancy Tester

Combinatorial mathematics can be used to determine how many proposed CLs for a n-ASJKC are tested when the 13-category redundancy test described in Section II-A is used. During proposal of the $(2n - 1)^{2n}$ possible CLs the base-13 PN sometimes has digits that are the same. The number of digits that are the same determine what fraction of CLs that result in PNs in a certain "sameness category" are tested. For instance, for 5-ASJKCs only 1/120 of the CLs with PNs in which all digits are different are tested, because only one of the 120 possible permutations of these digits results in a PN with each digit except the lowest-order one greater than or equal to the closest lower-order digit. On the other hand, all CLs with PNs in which all digits are the same are tested. The sum of the number of CLs in each "sameness category" times the corresponding fraction of times the members of that category are tested gives the total number of proposed CLs that are tested.

A slight modification of the procedure above simplifies the calculation. We calculate the fraction of proposed CLs tested if the 1,1 category is dropped. This fraction times the total number of proposed CLs—$(2n - 1)^{2n}$—gives an estimate of the number of tested CLs. The 1,1 category only occurs $1/(2n - 1)^2$ of the time for each JKFF. Calculation of the fraction of proposed CLs tested when there are 12 categories covering $(2n - 1)^2 - 1$ possible inputs for each JKFF is therefore almost equivalent to considering 13 categories with $(2n - 1)^2$ possible inputs. Because this simplification eliminates the least probable category and increases the probability of digits being the same, slightly fewer CLs than the number we'll calculate are actually tested.

The table below shows all possible "sameness categories" for $n = 5$ and 6. The number of times each category occurs, the fraction of this number that are tested, and the resultant number that are tested are shown. The number of times a "sameness category" occurs is determined by applying combinatorial arguments to the eight categories of size

(n - 1) and four categories of size (n - 1) x (n - 2) for each JKFF. For instance, the number of times out of $(80)^5$ that a 5-digit PN with four digits the same and one digit different occurs is equal to the number of times the four identical digits are in the category of size 4 plus the number of times they are in the category of size 12. The number equals $32 \times 4 \times 4 \times 4 \times 76 \times 5 + 48 \times 12 \times 12 \times 12 \times 68 \times 5 = 28,979,200$.

n = 5: Categories	Number of Occurrences	Fraction Tested	Total Tested
all different (d)	932,904,960	1/120	7,774,208
2 same(s), 3d	1,598,668,800	2/120	26,644,480
2s, 2s, 1d	366,059,520	4/120	12,201,984
3s, 2d	306,954,240	6/120	15,347,712
3s, 2s	42,229,760	12/120	4,222,976
4s, 1d	28,979,200	24/120	5,795,840
5s	1,003,520	120/120	1,003,520
Total	$3,276,800,000 = (80)^5$		72,990,720

Fraction tested = 72,990,720/3,276,800,000 $\approx$ 1/45

Number tested out of $(81)^5$

$\approx (((81)^5 \times 72,990,720)/3,276,800,000) \approx 7.8 \times 10^7$

n = 6:

Categories	Number of Occurrences	Fraction Tested	Total Tested
6d	327,915,000,000	1/720	455,437,500
2s, 4d	1,144,935,000,000	2/720	3,180,375,000
2s, 2s, 2d	703,181,250,000	4/720	3,906,562,500
2s, 2s, 2s	42,918,750,000	8/720	476,875,000
3s, 3d	422,685,000,000	6/720	3,522,375,000
3s, 2s, 1d	231,075,000,000	12/720	3,851,250,000
3s, 3s	8,328,750,000	36/720	416,437,500
4s, 2d	83,418,750,000	24/720	2,780,625,000
4s, 2s	13,573,125,000	48/720	904,875,000
5s, 1d	7,697,250,000	120/720	1,282,875,000
6s	256,125,000	720/720	256,125,000
Total	2,985,984,000,000 = $(120)^6$		21,033,812,500

Fraction tested = 21,033,812,500/2,985,984,000,000 ≈ 1/141

Number tested out of $(121)^6$

≈$((121)^6 \times 21,033,812,500)/2,985,984,000,000 \approx 2.2 \times 10^{10}$

EFFECT OF REDUNDANCY TEST FOR n = 5, 6

VIII ABBREVIATION TABLE

ASJKC: Autonomous, synchronous counter constructed only of J-K Flip-Flops. An x-ASJKC has x JKFFs. An x,y ASJKC has x JKFFs and cycles with a period of y.

CL: Connection list.

CT: Clock transition. This denotes the transition 0 → 1 of the source of timing pulses to one or more JKFFs. This signals the recalculation of state for those JKFFs.

GI: Greatest-integer function. This rounds all numbers with a fractional part up to the nearest integer -- GI(2.0) = 2, GI(2.01) = 3, GI(2.99) = 3.

JKFF: J-K Flip-Flop

P: Period

PDP-10: The general-purpose computer used in our experiment.

PN: Partition number. Each of the input categories of a JKFF is assigned a different PN.

SPM: Special-purpose digital machine.

IX TERMINOLOGY

X REFERENCES

Hennie, Frederick C., Finite-state Models for Logical Machines, John Wiley and Sons, Inc., New York, London, and Sydney, 1968.

Scientific American, Information, W. H. Freeman and Company, San Francisco, 1966.

Caldwell, Samuel H., Switching Circuits and Logical Design, John Wiley and Sons, Inc., New York, 1958.

Duryee, P. S., "Counter Designs Swing Without Gates," Electronic Design, December 6, 1967.

Langdon, Glen G., "A Survey of Counter Design Techniques," Computer Design, October, 1970.

Richards, R. K., Digital Design, John Wiley and Sons, 1971, pp 120-154.