(14)

Appeared in the Proceedings of

Purdue 1971 Symposium on Applications of Computers to Electrical Engineering

April 26-29,1971 Sponsored by COSINE

# REGISTER TRANSFER MODULES (RTM)* FOR HIGHER LEVEL DIGITAL SYSTEM DESIGN

C. G. Bell
J. Grason
Carnegie-Mellon University
Pittsburgh, Pennsylvania

S. Mega
R. Van Naarden
P. Williams
Digital Equipment Corp.
Maynard,Massachusetts

## INTRODUCTION

Register transfer modules (called RTMs) are used as the basis for digital systems logical design. RTMs have the property of allowing a digital system to be specified in either a register transfer flow chart form, or a state diagram form which has complete construction (wiring) information, thus completely obviating the need for combinatorial and sequential switching circuit design.

In the design of digital systems the problem formulation and the design solution are most likely carried out at a register transfer concept level. Texts on logical and computer design discuss the register transfers as primitive components (e.g., Chu, 1970). Logical design simulators which use the register transfer language have been written and there have been attempts to carry out the detailed sequential and combinatorial logic designs from register transfer descriptions (e.g.,Friedman and Yang, 1969). Clark (1967) at Washington University, St. Louis, has been developing and evaluating a basic set of modules, called Macromodules, to fulfill the need of the novice user who must build very large systems. Despite the acknowledgement that there are primitives based on register transfer, there is yet to emerge a common set of modules that are taken as primitive in the same way we think of various flip-flop types and NAND and NOR gates.

Register Transfer Modules (RTM) is one basic set of modules for digital systems design at the register transfer level; these modules have been implemented by DEC.** The design of RTMs has been influenced by many of the above approaches and disciplines.

Several aspects of the RTM system are unique:

1. Digital system design is carried out entirely in terms of the modules; combinatorial and sequential switching circuit design are not used. (The process is akin to programming a sequential computer).

2. The most abstract, and usually the only representation of a given design has enough information for constructing the system. This representation is a standard flow chart to specify the control flow, coupled to a data part which holds the data and carries out data operations.

3. The register transfer modules make extensive use of MSI circuitry and can use LSI circuitry to provide even lower cost modules.

## DESIGN CONSIDERATIONS

RTMs were not designed solely for teaching register transfer level digital systems design. The conflicting durability and low cost objectives coupled with a relatively small market make an educational market based design too difficult. Instead, we tried to have a design which covers more user applications (hence a larger market) to reduce cost, and also to provide for more interesting design solution possibilities. When possible, the pedagogical aspects have been stressed, however.

The three problem classes for which the modules were designed are:

Special purpose, computer-related, and educational digital systems. The special purpose digital systems are larger than 20 MSI circuits, but smaller than a stored program computer (a typical RTM system would have 4 $\sim$ 100 control states, 1 $\sim$ 4 arithmetic units, and a small memory of 16 $\sim$ 1000 words). Computer related applications range from computer peripherals to the emulation of computers.

## THE RTM SYSTEM

The RTM system consists of about 20 different types of Register Transfer Modules (falling into four classes), and a method of interconnecting the modules via a common bus which carries data and timing interlock signals for the register transfers. Some of the modules connect to the bus in order to transfer data, and the remaining modules "control" when data is to be transferred.

The module types are based on the PMS primitive types of Bell and Newell (1971). PMS is the information flow level of digital systems in which information is taken as a commodity which is stored in memories (M), transmitted from place to place over links (L), changed in form (encoded) by transducers (T), routed to different places by switches (S), controlled by components called controls (K), and used to produce different information by data-operations (D), and processors (P). A collection of the PMS primitives usually forms a computer (C), which is at the very least a P-M pair.

* Trademark, Digital Equipment Corporation

** The modules were initially inspired by a grant from NSF(GY-5160) for undergraduate educational equipment to Carnegie-Mellon University.

The ISP (for Instruction Set Processor) language (Bell and Newell, 1971) is used to define the instructions of a machine in terms of the next lowest level, i.e., the register transfer level. ISP as we use it here is a language for describing the register transfer operations of the RTMs. We use only the parts of ISP that are commonly known by the digital systems engineer. Those readers unfamiliar with register transfer languages of this type are invited to examine either a programming language (e.g., Fortran) or the ISP language. The four basic module classes in RTM are:

1.-2. DM, for Data operation combined with Memory; and M, for memory. These modules are what we commonly think of as being a digital system (or at least the arithmetic unit). They are the register transfer gating paths and combinatorial circuits for the simple arithmetic and logical functions--hence the D part (for data operations). The D part carries out the evaluation of the righthand side of an arithmetic expression* as in a programming language in which an integer value is computed e.g., $\leftarrow A + B$, $\leftarrow A - B$, $\leftarrow A \oplus B$, $\leftarrow A + 1$.

The M (memory) part is just the registers (e.g., A, B) which hold data between statements; these essentially correspond to the variables which are declared in a program. The operations on memory are usually just reading ($\leftarrow$ M) and writing (M $\leftarrow$). For the current technology we always combine data operations (D) with memory (M). There are, however, modules with only an M part, e.g., core memory arrays. All DM and M modules connect to a common data bus (or busses). Types of DM and M modules are: the general purpose arithmetic element, a single transfer register, boolean flags (1 bit registers), read-write memories, and read only memories. The memories hold two's complement 8, 12, or 16-bit integers.

3. K, for control. The K modules are responsible for controlling the transfer of data among the various registers by appropriately evoking operations by DM and M types. The K modules are analogous to the control structure of a program. The K modules called K.simple control at which times the various statements (the DM and M's) are evoked (executed). The K.decision modules are used to make decisions about which operations are to be evoked next. The K.subroutine modules are used to connect a sequence of operations together as a subroutine. K.parallel-branch and K.parallel merge modules synchronize control when there is more than one operation taking place at a time. Other control modules include: clocks, delays, manual start keys, and serial merge of control flow.

* An expression "lefthand side $\leftarrow$ righthand side" is used to indicate the integer value of the righthand side being read (i.e., computed or taken as a source) and placed in the register on the lefthand side (i.e., a destination).

4. T, for Transducers. These modules provide an interface to the environment outside RTM. These include the Teletype interface, analog/digital converters, lights, switches, and interfaces to computers. These modules also connect to the common data bus.

There is not space here to describe all 20 RTM module types in detail, so the details of the modules will be introduced by giving the four modules which are necessary for non-trivial digital systems: K.simple, DM.gpa, K.decision, and K.bus.

## K(simple/s)

K.simple (Ks) is the basic module which evokes a function consisting of a data operation and a register transfer—in essence an arithmetic expression. When a Ks is evoked, it in turn evokes the function, consisting of the data operation followed by a register transfer, and when the function is complete, Ks evokes the next K in the control sequence. The diagram for Ks with its two inputs and two outputs is shown in Figure 1.

## K(decision/d)

K.decision (Kd) provides for the routing of control flow based on the condition of a boolean input. The diagram for Kd with its two inputs and two outputs is shown in Figure 2. Each time a decision control is evoked, it in turn evokes either of the controls following it, depending on whether the boolean input is true (a 1) or false (a 0).

## DM (general purpose arithmetic/gpa)

The DMgpa allows arithmetic function results (data operations) which have been performed on its two registers A and B, to be written into other registers (using the bus). Results can also be transferred (written) into A and B ($A \leftarrow$; $B \leftarrow$). The data operations are: $\leftarrow A$; $\leftarrow B$; $\leftarrow \neg A$; $\leftarrow \neg B$; $\leftarrow A + B$, $\leftarrow A - B$, $\leftarrow A - 1$, $\leftarrow A + 1$, $\leftarrow A \times 2$, $\leftarrow A \wedge B$, $\leftarrow A \vee B$; and $\leftarrow A \oplus B$. An input that evokes the function $\leftarrow$ (Result)/2 can be combined with the previous function outputs to give either $\leftarrow A/2$, $\leftarrow B/2$, $\leftarrow (A+B)/2$, etc. Two boolean inputs, shift in <16,-1>, allow data to be shifted into the left and righthand bits on /2 and x2 operations, respectively. Bits of registers A and B are available as boolean outputs.

## K(bus sense and control module/bus)

Each independent data bus in the system requires a centralized control module. It has a register, Bus, which always contains the result of the last register transfer that took place via the bus. K bus carries out several functions: monitoring register transfer operations; providing for single step manual control for algorithm flow checkout by the user; providing for sense lights (indicators); providing for a word source of zero, i.e., $\leftarrow 0$; forming boolean functions of the previous transfer which are available after each control step using the bus; power on initialization; manual startup; and bus termination.

ample: Sum of Integers to N

A small system to sum the integers to N can be built which uses the four aforementioned modules: a DMgpa, Kbus, Ksimple, and a Kdecision. In addition, a switch register to enter N, and a manual start control module to start the system are needed. Suppose we have an integer, N, and we would like to sum all the integers up to N, i.e., $S = 0 + 1 + 2 + \ldots + N$. Instead of counting to N, we start with N and count down to zero. The data and control parts together give us the RTM wiring diagram shown in Figure 3, with the data part shown on the right side and the control part shown on the left.

N is entered via a switch register. The control sequence is initiated by a K.manual-start (A human presses a key). This result, S and the variable N are held in a general purpose arithmetic module, DMgpa. The first control step reads T to register N,$(N \leftarrow T)$. The second step initializes the sum, S, $(S \leftarrow 0)$. The inner loop consists of the three functions: $S \leftarrow S+N$; $N \leftarrow N-1$; and a test for $N=0$.

Mechanical Structure

The modules are constructed using double sided printed circuit boards of either 5" x 8 1/2" (with 72 pins), or 2 1/2" x 4 1/4" (with 36 pins). A few RTMs, e.g., the general purpose arithmetic unit, are several of the above large boards; these necessitate module interconnections at the top and back of the printed circuit boards. Others, e.g., the simple control, are so small that several are placed on a single printed circuit board. The bus is pre-wired on a printed circuit board mounting panel containing connector pins. The boards plug into the back of this panel and the control wiring is done on the front using either "push on" or "wire wrap" connections. A medium sized RTM system (e.g., a small, stored program computer) can be constructed using a single 5 1/4" x 8 1/2" x 19" mounting panel.

## RESULTS

Numerous digital design problems have been "benchmarked" using the modules. These have ranged from multipliers to large minicomputers. The present modules were based on the experience gained in implementing a small, special purpose stored program computer. The process of specifying the computer took approximately two hours. The computer was wired, and aside from minor system circuits problems, uncovered in RTM, the computer operated essentially when power was applied, since there were no logic errors. The computer was designed from an actual application which had about 300 constants, 600 control steps and about 16 variables. If the 600 control steps had been hard-wired, the system would have operated a factor of ten faster, but would have been more expensive and less flexible. The computer had only 24 simple and 16 decision controls. (By comparison a DEC PDP-8 is roughly twice this size).

Because we have just acquired a module inventory, we as yet have no experience in their physical utilization in the laboratory. We have used them in undergraduate and graduate courses for one year, and students with basic logical design background formulate and solve digital systems problems after only a brief introduction. These students have used RTMs to express large stored program general and special purpose processors easily and clearly. Practicing logical design engineers have more difficulty in understanding how to solve digital system design problems using RTMs, apparently because they have not been taught any representation and design techniques higher than the logic level.

## CONCLUSIONS

There are methods (and modules) within the RTM concept for achieving more parallelism, connecting multiple bus structures and allowing more central (perhaps "microprogrammed") control structures. Simulators and other design aids have been written, and the underlying theory must be further clarified and expanded. We have avoided discussing these peripheral questions in order to give the reader an operational, closed view of RTMs, rather than convince him of their open-endedness.

The concept of using high level building blocks is not new, but we think this particular implementation of a set of simple blocks is quite useful to digital systems education and design. The many problem bench mark designs yield reasonably consistent results: the modules can be applied where there are between 4 and 100 control steps, a few arithmetic registers, a small read-write memory (100 words), and perhaps some read only memory. The user need only have a good fundamental understanding of the use of flow charts, and be familiar with the concept of registers and register operations on data.

## REFERENCES

Bell, C. G. and A. Newell, Computer Structures: Readings and Examples, McGraw-Hill Book Co., New York, N.Y., 1971.

Chu, Y., Introduction to Computer Organization, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1970.

Clark, W. A., "Macromodular Computer Systems", SJCC 1967, pp. 335-336 (introduction of a set of six papers, pp. 337-401 in same conference).

Friedman, T. D. and S. C. Yang, "Methods Used in an Automatic Logic Design Generator (ALERT)", IEEE Trans. on Computers, vol. C-18, no. 7, pp. 393-614, July 1969.

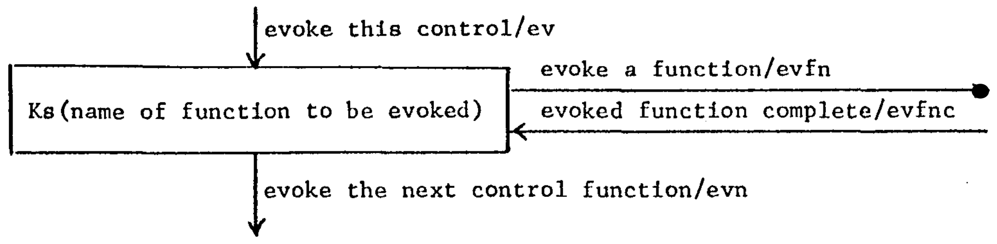

evoke this control/ev
Ks(name of function to be evoked)
evoke a function/evfn
evoked function complete/evfnc
evoke the next control function/evn

Figure 1. The Diagram for K.simple

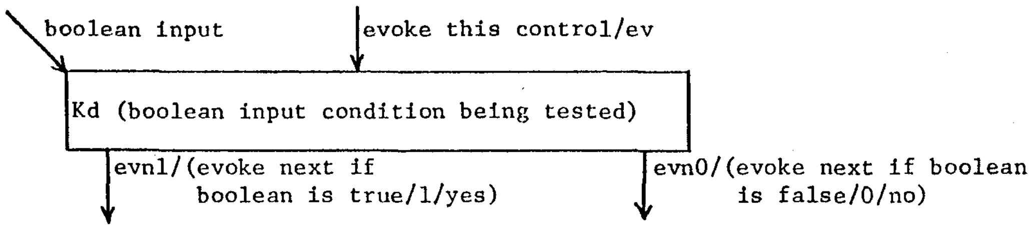
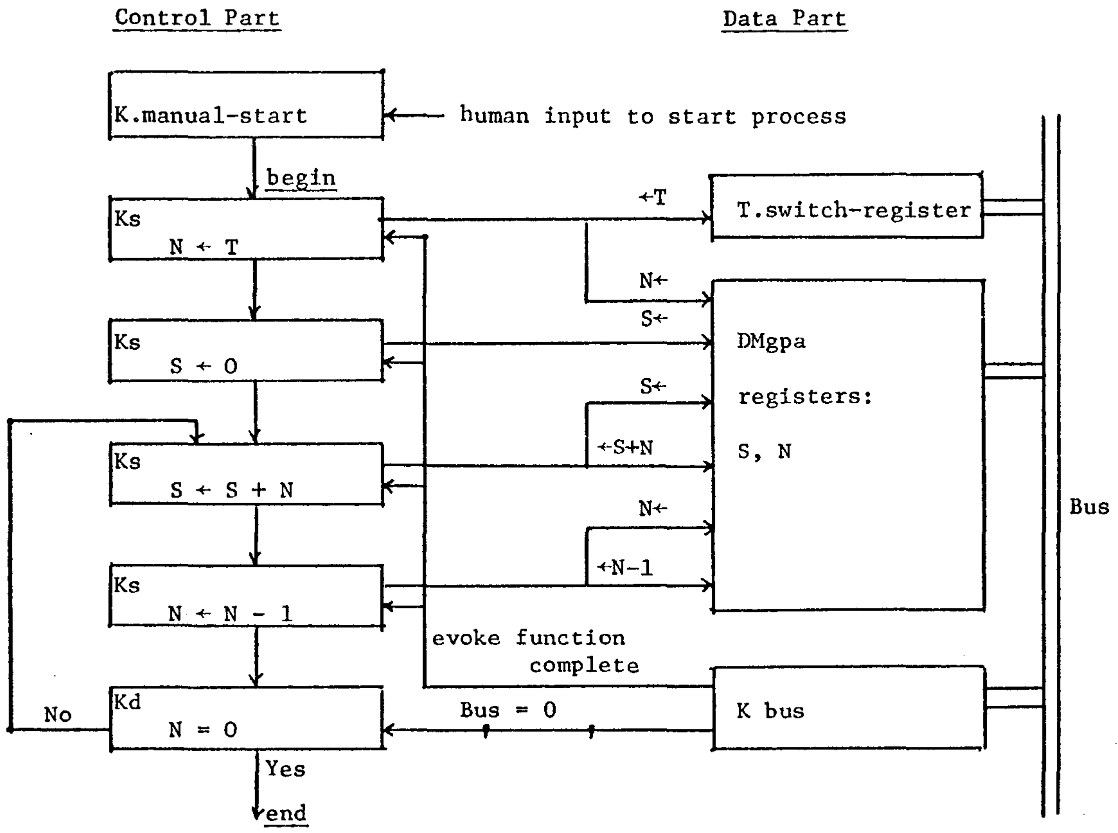

boolean input
evoke this control/ev
Kd (boolean input condition being tested)
evn1/(evoke next if boolean is true/1/yes)
evn0/(evoke next if boolean is false/0/no)

Figure 2. The Diagram for K.decision



Ks ≡ K.simple evoke module
Kd ≡ K.decision module

| | |
|---|---|
| ———— | Control flow and evoke wires |
| ════ | Bus for data wires |
| ——┼—— | Boolean variable wires |

Figure 3. RTM digital system to take a value from a switch register input, and to sum the integers to that value.