Experience on RTN

V. Azcoiti, I. Campos, J.C. Ciria, A. Cruz, D. Iñiguez, F. Lesmes, C.E. Piedrafita, A. Rivero, A. Tarancón.

Departamento de Física Teórica, Universidad de Zaragoza, Spain

P. Téllez.

Servicio de Instrumentación Electrónica. Universidad de Zaragoza, Spain.

L.A. Fernández, A. Muñoz Sudupe, J.J. Ruiz.

Departamento de Física Teórica I.Universidad Complutense de Madrid, Spain.

A. González-Arroyo, P. Martínez.

Departamento de Física Teórica. Universidad Autónoma de Madrid, Spain.

J. Pech.

Institute of Physics, Prague, Czechoslovakia, INFN-Sezione di Roma , Dipartimento di Fisica II, Universitá di Roma, Italy and Departamento de Física Teórica, Zaragoza, Spain.

D. Badoni.

INFN-Sezione di Roma, Dipartimento di Fisica II, Universitá di Roma, Italy.

RTN is a MIMD parallel computer, with 64 INMOS T-805 processors, reconfigurable topology, 64 or 256 Mbytes DRAM and 100 Mflops-640 Mips peak performance.

Introduction

RTN is a MIMD machine consisting on 8 boards with 8 INMOS T-805 transputers at 20 MHz and one INMOS C004 Cross-Link each, plus a Controller board. Each T805 can access 1 or 4 Mbytes DRAM. The topology is reconfigurable from software in run-time and one can use RTN as a single machine or segment it into several independent ones. RTN is connected to the exterior via the Controller, through one link of its T805, therefore it can be connected to any Host through a Root Transputer. The Controller can also be connected to a SCSI gate through a C011. One or several boards can be connected directly by means of the INMOS Up-Down structure, although in that case it is impossible to segment the machine or work with closed topologies.

The Base Board

The Base Board has 8 T805 transputers at 20 MHz with access to 1 or 4 Mbytes DRAM each and is reconfigurable in run time by means of a C004. Each transputer accesses directly its own memory bank. Any transputer must establish asynchronous communication through links with any other to access the latter's bank. The 4 Mbytes bank is obtained using four

1 Mbyte chips. If four 256 K chips are used, one obtains the 1 Mbyte bank. Reconfigurability is obtained by means of one C004 Cross-Link in each board. The hardware configuration , shown in figs. 1 and 2, has been chosen in such a way that for most problems the path between two transputers will cross at most 2 Cross-Links, causing the signal a delay of at most 4 bits, which allows the transmission of the acknowledgement protocols simultaneously with the message. The communication speed through links can be chosen to be 5, 10 or 20 Mbits/sec by means of switches on the board.

Internal communications

Each board has one 32x3 pin connector. As fig. 1 shows, the links 1 and 3 of each transputer are connected to the neighbouring ones, forming a ring. The link 0 goes to the connector and the link 2 of the i'eth transputer (i = 0 to 7) goes to the i'eth link of the crosslink. The remaining 24 links of the C004, as well as the configuration link, go to the connector. The 0'th transputer has the EventAck and EventReq pins connected to the connector. This provides a link independent signal pass mechanism which is used to open a network with all links internally connected. In order to preserve compatibility with INMOS transputer boards, the Base Boards have UP and DOWN ports. The RESET and ANALYSE signals are transmitted from the UP port to the DOWN port directly.

Cross-Link and Back-Plane

The Cross-Link C004 configures the connection of 32 links. In the Base Board 8 links are connected to the on board transputers, and 24 are left for external connections. The Cross-Link configuration is managed by the Config-Link, which is connected to the connector, in order to be configured from an external transputer. The Cross-link is reset from the Board ten milliseconds after power connection, by means of a one millisecond pulse.

The 8 Base Boards and the Controller Board in RTN are connected by means of a Back-Plane in a standard VME crate. The eight links 16-23 from a board's Cross-Link are connected to the next board's eight transputers, the first and last boards closing a circle, as shown in fig. 2. The links 15 and 31 are connected via Back-Plane to the Controller Board. The remaining links (8-14 and 24-30) are used for the additional connections via Back-Plane between Cross-Links in different boards, minimising the longest distance, as shown in fig. 2.

The Controller Board

The Controller Board, fig. 3, has one T805 transputer, one C004 Crosslink, 4 Mbytes 80 ns DRAM and an ALTERA chip for logics and control. The board is connected to a B004 board through the UP gate and the zero link. The toroidal configuration leaves no link free, so that the communication with the exterior must be made some way other than by links. The chosen mechanism has been the Event.Request (ER) - Event.Acknowledgement (EA) facility. Memory address 8 works as an internal OCCAM channel, Event in what follows, which is enabled only when the ER transputer pin is Up. In turn, the EA pin, which normally is Down, goes Up when the input from channel Event is acted.

The hardware configuration in fig. 4, together with the normal state ER of B Up, everything else Down and the parallel programs:

```
       B                        C
CHAN OF ANY Event:      CHAN OF ANY Event:
INT any:                INT any:
PLACE Event AT 8:       PLACE Event AT 8:
SEQ                     SEQ
program                 program
Event ? any             Event ? any
...                     reconfigure and read
```

cause the following effects: The program on the Controller C awaits, at the instruction Event ? any , for the channel Event to be enabled. When the program running on the Board B executes the instruction Event ? any, EA at B is set Up and so is EA at C, which enables the channel Event on C, so that the instruction Event ? any is executed, so learning C that the program in B is over and going on with the execution of the program in C, which normally will consist on reconfiguring the network, opening it and reading the results. At the execution of Event ? any on C, EA on C goes Up, and ER on B, EA on B, ER on C and EA on C go all Down, recovering the initial state. In order to implement this mechanism on the eight boards, as well as to allow the resetting of part of the machine and so its segmentation into several independent ones, memory locations are used. The ER status of the 0th transputer of the eight boards is stored in the least significant byte of the memory location # 20 00 00 03, the 0'th card corresponding to the least significant bit. Analogously the EA, RESET, ANALYSE and ERROR status are mapped on the memory addresses:

```
ER    #20 00 00 03,RESET #20 00 00 00,ANALYSE #20 00 00 01 (wr)
ERROR #20 00 00 00 or    #20 00 00 01,EA      #20 00 00 02 (rd)
```

The management of those signals is done by means of the chip ALTERA. The following program resets the boards 1, 3 and 5.

```
VAL reset.addr IS #20000000:
VAL analyse.addr IS # 20000001:
PLACE reset AT reset.addr:
PLACE analyse AT analyse.addr:
INT reset, analyse, target:
TIMER clock:
INT time:
SEQ
  target := #0000002A
  analyse := 0
  reset := 0
  reset := target
  clock ? time
  clock ? AFTER time PLUS 78:
  reset := #00000000
:
```

RTN operating system

The peculiar management of RESET, ANALYSE and ERROR just described prevents the use of the INMOS operating system. The latter provides two mechanisms: SUBSYSTEM allows

the independent management of the system connected to that port. DOWN transmits signals by the Up Down INMOS standard to all the reachable transputers. RTN can be connected through the controller board to the B004 in the host either to the Subsystem or the Down ports. The rest of RTN has no Up Down connection to Controller, the connection being as eight independent subsystems. In order to run a program on RTN the existence of a physical path through links between the Root and any transputer is necessary, as well as the reset status of all transputers which should receive code. All the tasks necessary to facilitate the running of programs on RTN have been programmed, forming a library of utility programs in OCCAM or C. The structure of a program running on RTN is as follows:

1. A procedure for the Root transputer, another one for the controller and the procedures to be run on the rest of the transputers.
2. In the main.pgm in OCCAM (or main.cfs in C) a booting path is specified.
3. Procedures have no arguments. Particularly links are defined through *.inc's.
4. According to the structure of the *.pgm, a configuring program must be run which allows booting.

Experience

RTN has started functionning in Zaragoza in November 1991 and in Rome in January 1992. Up to now the computer has been dedicated 90% to physical problems (U(1)-Higgs [1], SU(2), Spin Glasses [2]). Although no complete efficiency tests have been carried out yet, in a program running on all 64 transputers with heavy communication to study a system on a 16^4 lattice, the efficiency loss is less than 5 per cent. Comparisons with other computers can be made on the basis of the production programs. The OCCAM version of a program initially developed to run on a vector computer, with heavy floating point operation, yields a time ratio 1.37:1 with respect to a CRAY-XMP (250 Mflops peak). Another program with heavier use of fixed point operations, yields a time ratio 1/4 with respect to a VAX 9000. The RTN at Zaragoza has been working to this date an estimated 4800 hours. As its memory has no parity check, weekly reliability checks have been performed, a very low failure rate having been detected.

Acknowledgements

We thank for finantial support Diputación General de Aragón, Comisión Interministerial de Ciencia y Tecnología, Caja de Ahorros de la Inmaculada and the Univeristies of Madrid and Zaragoza.

References

1. *The Confining-Higgs phase transition in U(1)-Higgs LGT.* Phys. Lett., in press.
2. *Numerical evidence of a critical line in the 4d Ising spin glass.* Europhys. Lett., in press.

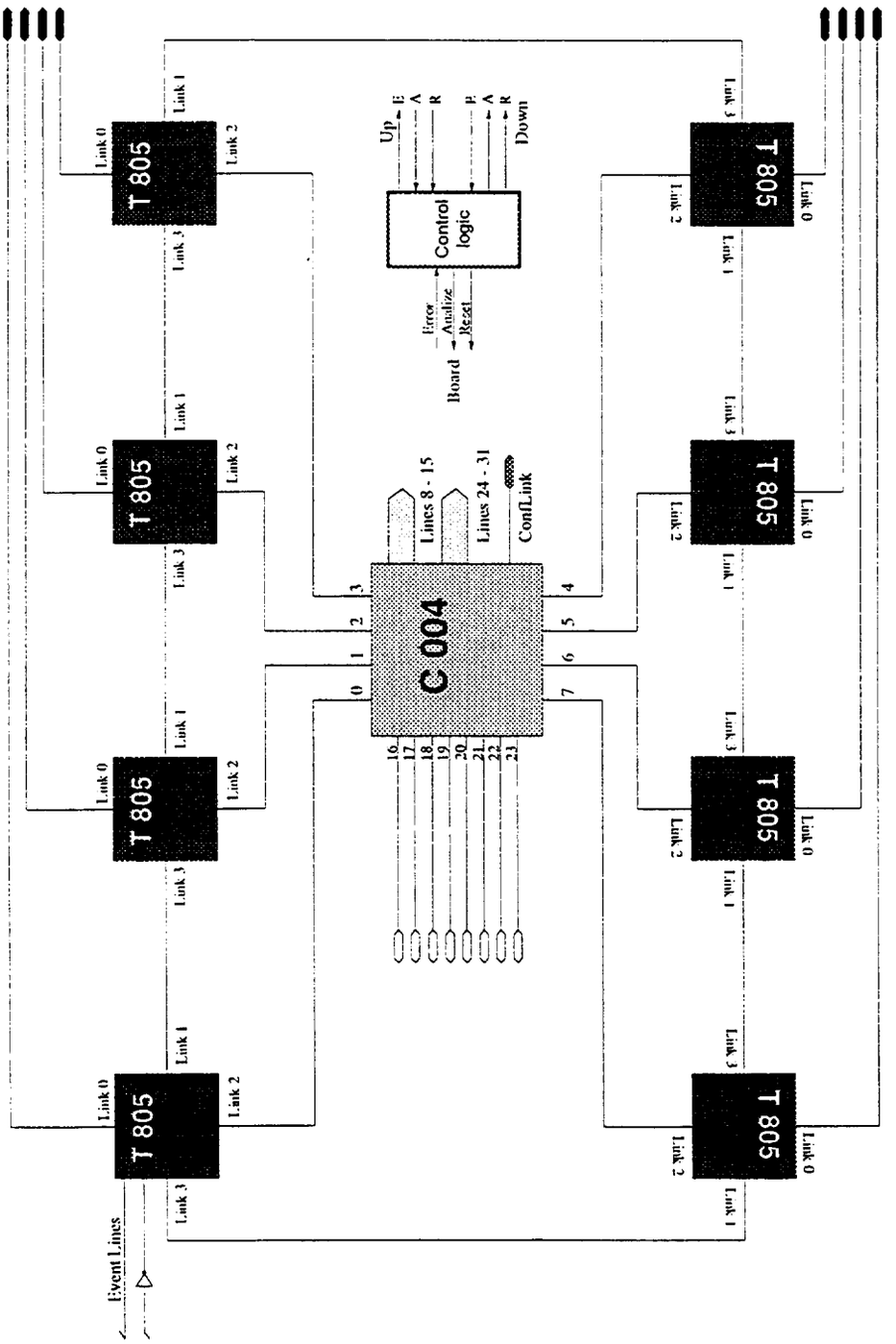
T 805
C 004
Control logic
Up
Down
Board
Error
Analize
Reset
Lines 8 - 15
Lines 24 - 31
ConfLink
Event Lines

Figure 1

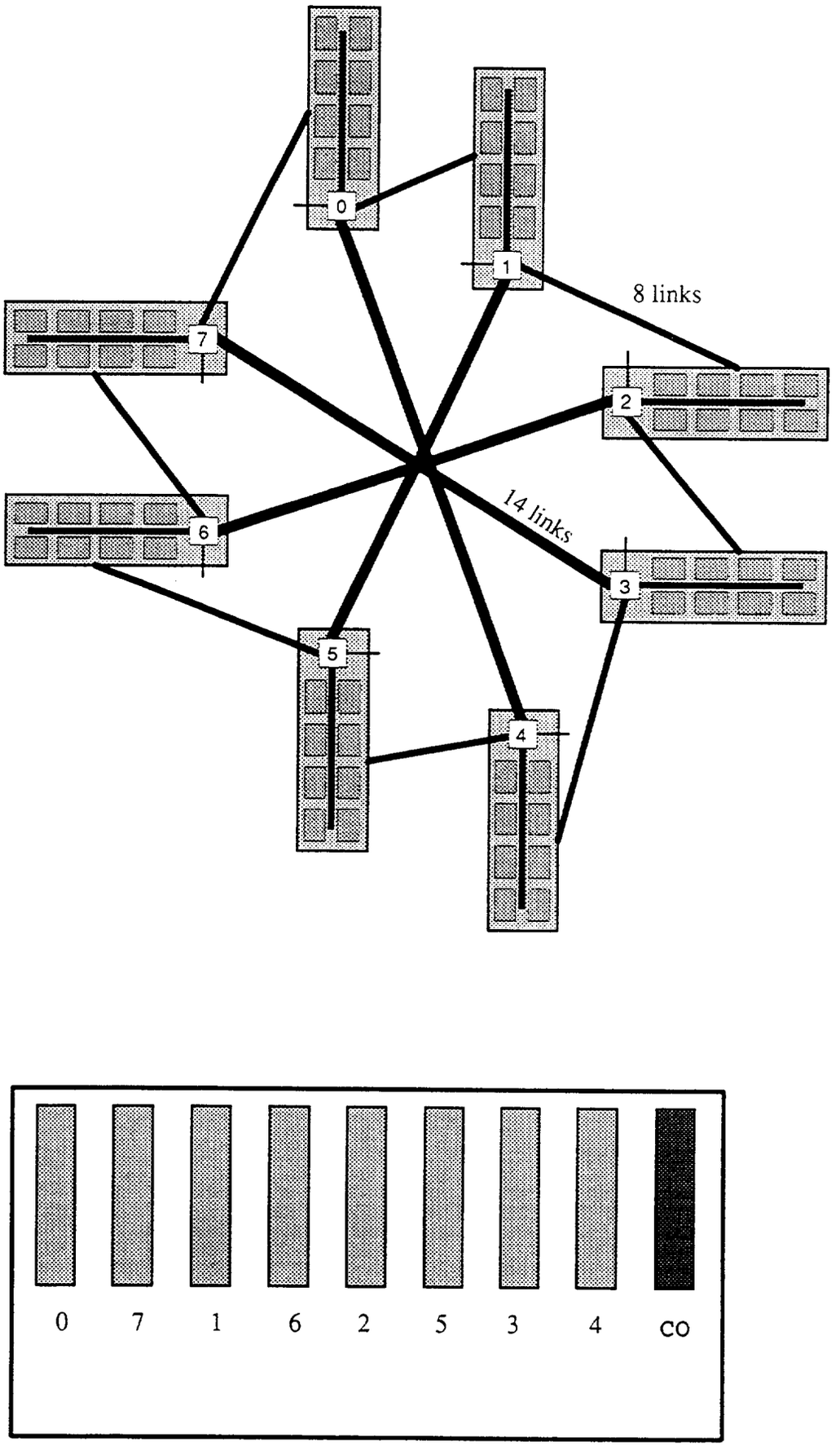

Figure 2

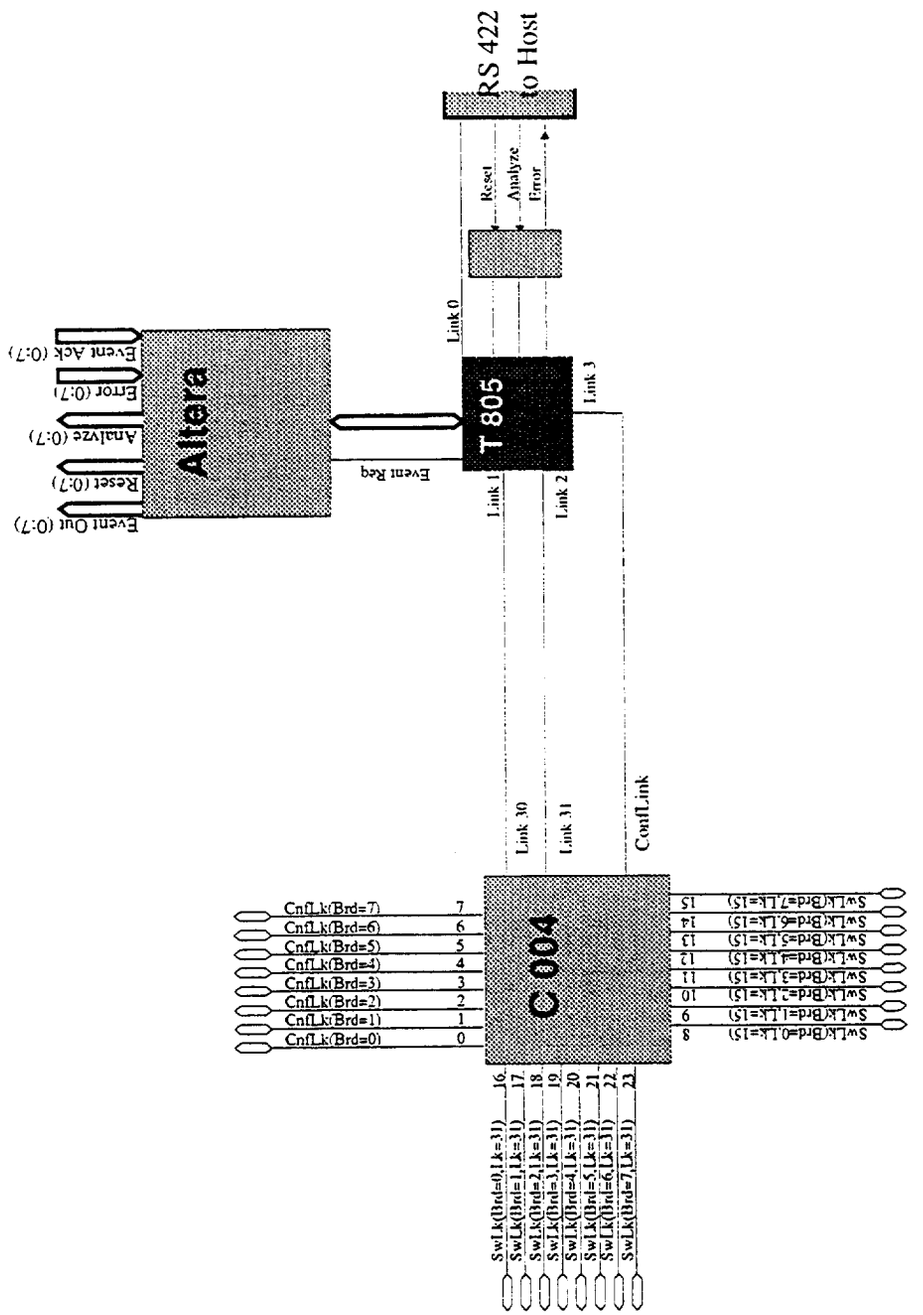
RS 422
to Host
Reset
Analyze
Error
Link 0
T 805
Link 3
Link 1
Link 2
Event Req
Altera
Event Ack (0:7)
Error (0:7)
Analyze (0:7)
Reset (0:7)
Event Out (0:7)
Link 30
Link 31
ConfLink
C 004
CnfLk(Brd=7) 7
CnfLk(Brd=6) 6
CnfLk(Brd=5) 5
CnfLk(Brd=4) 4
CnfLk(Brd=3) 3
CnfLk(Brd=2) 2
CnfLk(Brd=1) 1
CnfLk(Brd=0) 0
SwLk(Brd=0,Lk=15) 8
SwLk(Brd=1,Lk=15) 9
SwLk(Brd=2,Lk=15) 10
SwLk(Brd=3,Lk=15) 11
SwLk(Brd=4,Lk=15) 12
SwLk(Brd=5,Lk=15) 13
SwLk(Brd=6,Lk=15) 14
SwLk(Brd=7,Lk=15) 15
SwLk(Brd=0,Lk=31) 16
SwLk(Brd=1,Lk=31) 17
SwLk(Brd=2,Lk=31) 18
SwLk(Brd=3,Lk=31) 19
SwLk(Brd=4,Lk=31) 20
SwLk(Brd=5,Lk=31) 21
SwLk(Brd=6,Lk=31) 22
SwLk(Brd=7,Lk=31) 23

Figure 3

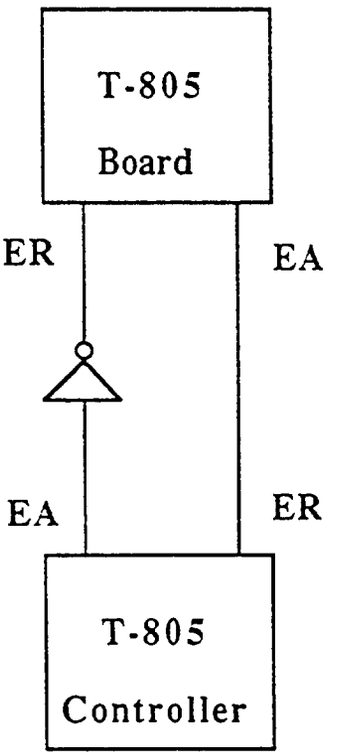

Figure 4.