Microarchitecture Specification for the Broadcast Hierarchy (TBH) Test Chip

Mike Rudnick, Sundaresh Cadambi, and Charles Hong

Oregon Graduate Center
Department of Computer Science
and Engineering
19600 N.W. von Neumann Drive
Beaverton, OR 97006-1999 USA

Technical Report No. CS/E 88-015

Abstract

This technical report contains the Microarchitecture Specification for the TBH (Broadcast Hierarchy) test chip, which was designed in the 1987 OGC Advanced VLSI class, and fabricated by MOSIS. In addition, the results of chip testing are included. The TBH chip simulates a single PBH (Physical Broadcast Hierarchy) with 8 transmitters and receivers. Data is sent serially and synchronously up a concentrator tree and then broadcast to the eight receivers.

Micro-Study for the TBH Test Chip

August 12, 1987

Sudarshan Cadambi, Charles Hong, Mike Rudnick
Class: CS&E 529
Professor: Dan Hammerstrom
Oregon Graduate Center

1. Introduction

The TBH (The Broadcast Hierarchy) chip implements a three level, 16 PN (Processing Node) communication structure. For each PN, only either the transmit or receive part of the PN is implemented. PNs are used to implement highly parallel VLCN (Very Large Connection Network) architectures.

The TBH test chip implements part of a broadcast domain communication structure. A broadcast domain is the group of PNs (and associated communication circuitry) interconnected via an instance of a broadcast communication structure. The higher the domain is in the hierarchy of broadcast domains, the larger the group of PNs connected by the broadcast domain.

A broadcast domain may contain a concentrate tree and/or a broadcast tree, or neither, depending upon the implementation. A concentrate tree is a tree with one-way communication paths running up the tree. The leaves of a concentrate tree are PNs. A broadcast tree is a tree with one-way communication paths running down the tree. Again, the leaves of the tree are PNs. A full concentrate-broadcast tree consists of one concentrate tree, one receive tree, and a communication structure connecting their roots. On the TBH test chip there is a concentrate tree, but no broadcast tree, only a global broadcast line.

There are 16 PNs on the TBH test chip. They are divided into 8 transmit PNs and 8 receive PNs. For the transmit PNs, only the transmit portion of each PN is implemented. Likewise a receive PN implements only the receive portion of a PN. The 8 transmit PNs are organized as a transmit buffer register file. Likewise the 8 receive PNs are organized as a receive buffer register file.

2. Functionality

2.1. PN Connections and Addresses

There are 8 unique PN addresses, 0 through 7. For any particular address, exactly one transmit PN and one receive PN will have that address. Each message sent consists of a 3-bit address followed by a 4-bit value. Whenever external data is loaded into a transmit PN, the low order (rightmost) three bits are the destination (receive) PN address and the high order (leftmost) four bits are the value. Each message is transmitted via the broadcast hierarchy communication structure (in this case the concentrate tree and global broadcast line) to the indicated receive PN.

The broadcast hierarchy communication structure transmits only one message at a time. Since multiple messages may be awaiting transfer, the communication structure determines which message has highest priority and completes transmission of that message first.

The TBH Test Chip uses a bandwidth slice prioritization scheme. Each switch, after having received the last bit of the last message, makes its priority choice according to the following scheme: n messages are taken from the higher address child followed by m messages from the lower address child. If a message is not available from the selected child, then message(s) from the other child are accepted.

For the TBH Test Chip, the n and m of the bandwidth slice prioritization scheme are both 1. This means when all the transmit PNs continuously have messages to transmit (ie, the communication structure is maximally loaded), each switch node will accept one higher priority (from the higher address child) message for every lower priority (from the lower address child) message.

Once a node accepts a message it will complete the transfer of that message before accepting a new message. In general, once a message has completed transmission, there will be a group of messages waiting to be transmitted (ie, pending). The next message to be transmitted will depend upon both the current state of the concentrate tree (ie, which part of its n high then m low cycle each switch node is in), the sending address of the messages waiting to be transmitted, and when each message first requested transmission relative to when the other pending messages first requested transmission.

2.2. Concentrate Tree Switch Addresses

For monitoring purposes the concentrate tree switches are assigned addresses. The switch addresses run from 0 through 6. Switch addresses 0 through 3 correspond to the first level of concentrate tree switches with 0 being the switch connected to the two lowest address transmit PNs and 3 being the switch connected to the two highest address transmit PNs. The second level of switches correspond to switch addresses 4 and 5, while the single switch in the last (third) level corresponds to switch address 6.

3. External Interface

There are three parts to the external interface - loading the transmit buffers, reading the receive buffers, and monitoring the transmit hierarchy tree nodes and global receive line. Unless explicitly stated otherwise, all external lines are to be valid at the rising edge of phase 1. Please refer to figure 1.

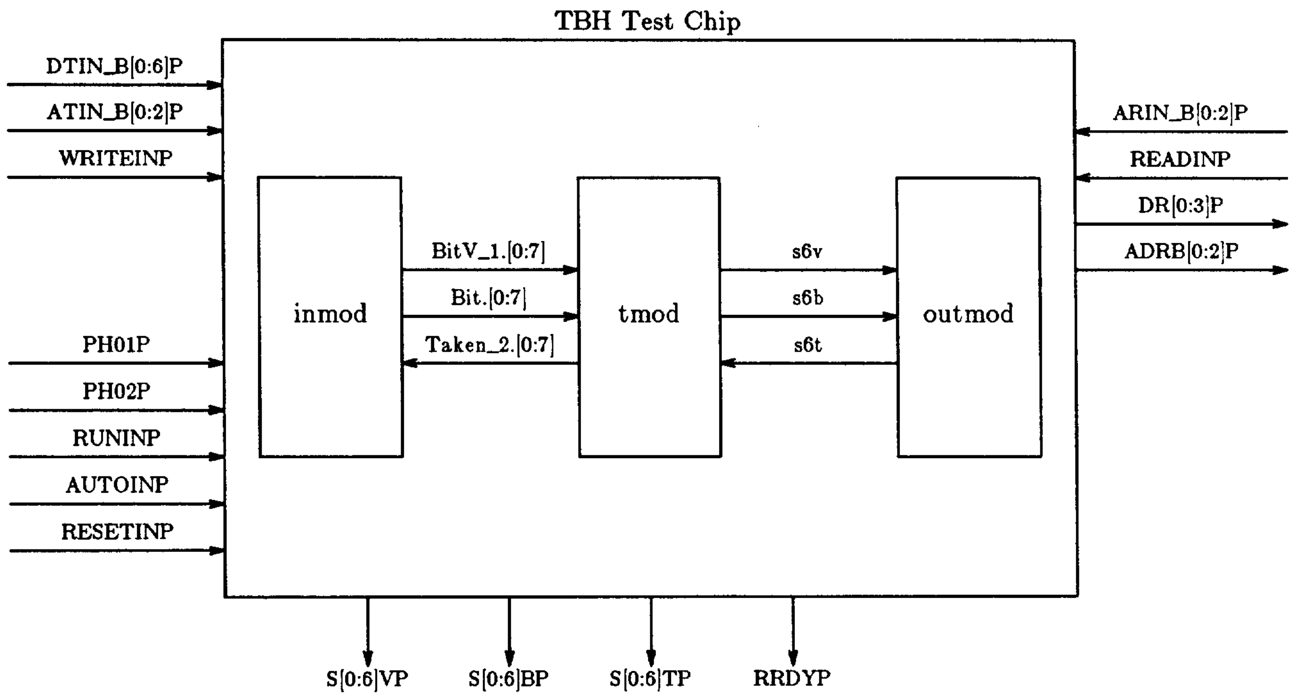


Figure 1: TBH Test Chip Overview ([0:6] is replace by a digit 0 through 6)

The operating sequence for the TBH Test Chip is:

1) reset the chip
2) load the transmit PNs
3) run the chip, monitoring concentrator tree traffic and received addresses
4) read out received values from the receive PNs

3.1. Transmit PN Buffer Input

DTIN_B <6:0> - PN data

DTIN_B <6:3> - "value"
DTIN_B <2:0> - "address"

ATIN_B <2:0> - transmit PN address

WriteIN - write-enable control signal for writing data to transmit buffer

DTIN_B <6:0> are the input pins for a message. The 3 low order lines (DTIN_B <2:0> are for inputing the receive PN address and the 4 high order lines (DTIN_B <6:3>) are for inputing the value.

ATIN_B <2:0> are the input pins for the addresses of the 8 transmit PN registers.

```
clock0
                ---             ---
PH01        _/     \________/     \_____________
                          ---             ---
PH02        ________/     \_________/     \____
            -
RunIN        \________________________________
                  ---------------
ATIN_B[i]______/                \___________
                  ---------------
WriteIN     ______/               \___________
                  ---------------
DTIN_B      ______/               \___________
```

3.2. Receive PN Buffer Output

DROUT_B <3:0> - receive PN's data

ARIN_B <2:0> - receive PN's address

RRdyOUT - receive buffer ready
ReadIN - read receive buffer data request

DROUT_B <3:0> are the output pins for reading message from a receiving PN's Register. The address of the receiving PN's Register is specified by the 3 address pins - ARIN_B <2:0>.

RRdyOUT signals to the outside that the receiving Register which is addressed by ARIN_B is ready for being read.

```
              ----------
ARIN_B   ____/          \__________

              ----------
ReadIN   ____/          \__________

                 ----------
DROUT_B  -------X__________X--------

                 ----------
RRdyOUT  -------X__________X--------
```

3.3. Debug, Monitor, and Control

ADRB2:0 - received message address
RUNINP - run TBH Test Chip (ie, send pending messages)
AUTOINP - run TBH Test Chip continuously on same transmit PN data
RESETINP - reset TBH Test Chip
PH01 - phase one clock
PH02 - phase two clock

for x = 0, 1, ..., 6:

SxVP - switch data valid
SxBP - switch data (bit)
SxTP - switch data taken

ADRB2:0 is the 3 bit address shift register of outMOD. The value on the pins is the current contents of the shift register and can be monitored realtime.

RUNINP high causes all valid messages in the transmit PNs to contend for transmission. RUNINP must be high by the leading edge of phase 1 for the TBH Test Chip to be active (run) during that clock cycle.

AUTOINP asserted causes the valid data in the transmit PNs to be sent repeatedly. That is to say that once a message has been sent, that same message is immediately requeued for transmission. This mode allows the concentrate tree prioritization scheme to be tested.

RESETINP high causes any messages currently being transmitted to abort and resets all logic to the initial state with the exception that the data contained in the transmit and receive PNs is set to zero. RESETINP takes precedence over all other signals. RESETINP should be asserted for one full clock cycle (or longer).

The highest level concentrate tree switch (Switch6) uses RUNINP to start and stop the flow of pending messages. This means that at any particular time when message flow is in a stopped state (ie, RUNINP low, and the current bit completed), several pending transmit PN messages may be in various states of working their way up the concentrate tree to become the current message.

PH01 and PH02 are phase one and phase two of a non-overlapped 12 MHz (84 ns) clock.

SxVP, SxBP, SxTP (x = 0, 1, ..., 6) are concentrate tree monitor lines. Each line shows the current (realtime) state of the valid (V), data (bit, or B), or taken (T) switch interconnect lines for one of the concentrator tree switches.

4. Microsimulation Modules

4.1. Overview

There are three modules in the TBH Test Chip microsimulator: a transmit PN register module, an interconnect structure module, and a receive PN register module.

The transmit PNs (inmod) serve to generate messages which contend to be transmitted via the concentrate tree and global receive line (tmod) to be received by the receive PNs (outmod).

The lines between the transmit PN register module and the tmod module, and between the tmod module and the receive PN module, plus the associated timing diagrams and interface protocol are described in the tmod section later in this document.

4.2. Transmit PN Buffer -- inMOD

4.2.1. inMOD interface

The inMOD module interfaces internal to the tMOD module for shifting bit-data to the lowest level switches and external to the outside pins for writing data into the registers.

The inMOD module interfaces externally with the outside for inputting message to the registers. There are 7 data lines and 3 address lines. A write-enable (WriteIN) line control/enable the address decoder. The input lines and timing diagram have been described in section 3.

The inMOD module interfaces internally with the 4 lowest level switches of the tmod module. Every 2 registers interfaces to a switch [i] through its bit-data line (Bit_1[i]) , its data-valid line (BitV_1[i]) and its bit-taken line (BTaken_2[i]). Since there are 8 registers and 3 interface lines per register, there are 18 signals connected to the tmod module.

The timing diagram is shown as below:

```
clock0
                ---       ---       ---       ---
PH01         _/   \_____/   \_____/   \_____/   \
                      ---       ---       ---
PH02         \_____/   \_____/   \_____/   \____
              ----------------------------------------
RunIN        /

Bit_1[i]      _!!!!!!!!____!!!!!!!!____!!!!!!!!____!
                ------------------------------------
BitV_1[i]    __/
                       ----      ----      ----
BTaken_2[i] ________/    \____/    \____/    \__
```

4.2.2. inMOD functionality

The inMOD module (or the Transmit PN Buffer) consists of: a Register File (8 registers); 8 Counters; 8 sets of control circuits; a 3-bit address decoder.

Each of the 8 Registers is 7 bits wide: the more significant 4 bits (bit6 --- bit3) are for storing the value, and the less significant 3 bits (bit2 --- bit0) are for storing the address of the receiving PN register. The bit0 is closest to the tmod module, and the bit6 is fartherest to the tmod module. The value and the address can be written, in parallel, through the 7 data input pins (DTIN_B) within one clock (CLK0) cycle, starting during PH2 and ending at PH1.

The 3-bit address decoder determines, by decoding the input address (ATIN_B), which register gets written. Only one register can be written in a clock cycle. And, the WriteIN needs to go low to change a new input address. The Transmit Register File is a "write-only" file from the external pins.

There is a 3-bit Counter corresponding to a register. The Counter counts from 0 to 6. When the ResetIN signal is asserted, all the Counters get reset to 0 and their control circuits generate no-valid-bit signals (BitV_1=low) to the tmod module. Whenever a new message gets loaded into a register, its Counter resets to 0 and outputs a bit-valid signal (i.e. BitV_1 goes high) to the tmod module. As long as there are valid bits waiting to be transmitted, the BitV_1 signal stays high. When a register receives a BTaken_2 high signal during PH2 (sending back from the tmod module), its Counter increments by one. At the same time, the bit-data in the register gets shifted 1 bit towards the lower order bit because its control circuit generates a "shift" signal during PH2. The bit0 data gets recirculated back to the bit6 so that when the AutoIN signal gets asserted (in addition to the RunIN) the message can be transmitted to the tmod module again and again.

Each Counter keeps track of how many bits are left in its register. If all 7 bits of a message have been transmitted to the tmod module, the BitV[i] signal goes low during the same PH2 when its Counter receives the last BTaken_2 signal from the tmod module. When the external ResetIN signal is asserted, all the control signals are set to zero.

Additional information on the inter-module interface & timing diagram can be found in the treeMOD section.

4.3. Receive PN Buffer -- outMOD

4.3.1. outMOD interface

The outMOD module interfaces internally to the tMOD module and externally to the outside pins.

The outMOD module interfaces internally with the highest-level switch (S6) through 3 lines. The S6 transmits bit-data through the Bit line and bit-valid signal through the SBitV line. When the outMOD module receives a valid bit-data, a data-taken signal is sent back to the S6 through the BTaken line. The timing diagram is shown below:

```
clock1
        ---       ---       ---       ---
ph01  _/   \_____/   \_____/   \_____/
             ---       ---       ---
PH02  \_____/   \_____/   \_____/   \_

Bit    _!!!!!!!____!!!!!!!!____!!!!!!!!____!
        ----      ----      ----      ----
SBitV  /    \____/    \____/    \____/
             ----      ----      ----
BTaken _____/    \____/    \____/    \____/
```

4.3.2. outMOD functionality

The Receive PN consists of: a serial shift-in Register File (8 registers); a 3-bit address buffer; a Counter; an address decoder.

Each of the 8 receiving registers is 4 bits wide for storing the 4-bit value. All the registers are "read-only" from the external address (ARIN_B) pins.

The outMOD module receives bit-data from the highest-level switch or the broadcast transmitter, when RunIN is asserted. When SBitV is high, the bit-data (Bit) is trapped. At the same time, a data-taken signal is sent back to the broadcast transmitter (i.e. BTaken is high).

The first 3 arriving bits of a new message are recognized as an address and are shifted into the 3-bit address buffer. The address is decoded to select one of the 8 registers. The next 4 arriving bits are interpreted as value-bits and get shifted into the proper receiving register. The first 3 bits (i.e. the address of one of the 8 receiving registers) can be monitored by the 3 external (ADRB2:0) pins.

The modulo-7 Counter monitors the serial message traffic and does the bit-keeping for the Receive PN Buffer. At the beginning of a new message, the Counter is reset to 0. The Counter increments by one when a new bit-data has been trapped and the BTaken signal has been sent back to S6. At any time of operation, the ReadIN input can be asserted and the contents of any register can be read out by selecting the address through the 3 external address (ARIN_B) pins. The address is decoded combinationally and the 4 bits data come out of the 4 data pins (DROUT_B) in parallel along with the RRdyOUT status bit indicating valid data.

Please refer to the tMOD section for additional information on the inter-module interface signals & timing diagram.

4.4. Interconnect Structure Module - tmod

The TBH Test Chip interconnect module is called tmod. It simulates the operation of the concentrate tree and the single global receive line. A three level concentrate tree is used, and so is large enough to handle 8 transmit PNs. The bottom level of the concentrate tree has four switch nodes, each connected to two transmit PNs, one to a high address child and one to a low address child.

4.4.1. tmod Interface

The tmod module interfaces (internal to the TBH Test Chip) with both the inmod module and the outmod module.

4.4.1.1. tmod Interface Input Lines

Each of four bottom level concentrate tree switch nodes are connected to two transmit PNs via a set of three signal lines - Valid_1.x, Bit.x, and Taken_2.x, x = 0, 1, ..., 7. These signals run between inmod and tmod. These module interface signals are listed below:

Valid_1.x - transmit PN[x] has next bit ready

Bit.x - next data bit of transmit PN[x]'s message

Taken_2.x - switch has taken current bit (Bit.x)

The Valid_1.x and Bit_1.x lines are inputs from the transmit PNs to the bottom level concentrate tree switches. The Taken_2.x lines are handshake lines from the bottom level concentrate tree switches to the transmit PNs.

Valid_1.x asserted means the transmit PN's next bit is ready. Taken_2.x asserted means the concentrate tree switch has accepted the current bit and is ready to accept the next bit of the message. The next bit is then placed on the Bit.x line and Valid_1.x is asserted.

Below is the timing diagram for three bits of a message from a transmit PN to a bottom level concentrate tree switch. The switch in this example does not take the first bit until the second clock cycle. This corresponds to the switch having been busy handling the end of some other message.

For all lines except Bit.x, high means high (Vdd!) and low means low (GND!). For Bit.x vertical "T"s means valid and low means not valid. Note that Valid_1.x must remain asserted until Taken_2.x is seen. This is because the switch node traps Bit.x with PH01.

```
              t0        t1        t2        t3        t4        t5
              ---       ---       ---       ---       ---       ---
PH01        _/   \_____/   \_____/   \_____/   \_____/   \_____/   \
                   ---       ---       ---       ---       ---
PH02        \_____/   \_____/   \_____/   \_____/   \_____/   \_____
                             ---       ---                 ---
Taken_2.x   ________________/   \_____/   \_______________/   \_____
                   ----------   -------   -----------------   ------
Valid_1.x   ______/          \_/       \_/                 \_/
             ----------------   -------   -----------------   ------
Bit.x       _||||||||||||||||___|||||||___|||||||||||||||||___||||||
```

If Valid_1.x, Taken_2.x, and Bit.x (where x = 0, ... , 7 indexes one of the inmod PNs) are changed to syv, syt, and syb, respectively (where y = 0, ... , 6 indexes one of the tmod switch nodes and v indicateds the valid line, t indicates the taken line, and b indicates the bit, ie, data, line), the above timing diagram shows the synchronous handshake between two concentrate tree levels.

4.4.1.2. tmod Interface Output Lines

s6t	handshake and data between
s6b	top level concentrate tree
s6v	switch and receive PNs

Below is the timing diagram for two bits of a message from Switch6 to the receive PNs. The timing diagram is from the point of view of Switch 6 (in terms of propagation delay).

```
              t0        t1        t2        t3        t4        t5
              ---       ---       ---       ---       ---       ---
PH01        _/   /e_____/   /e_____/   /e_____/   /e_____/   /e_____/   /e
                   ---       ---       ---       ---       ---
PH02        /e_____/   /e_____/   /e_____/   /e_____/   /e_____/   /e_____
                             ---       ---
s6t         ________________/   /e_____/   /e
              --------------      ----
s6b         _/||||||||||||||/e____/||||/e____
              ---       ---       ---
s6v         _/   /e_____/   /e_____/   /e_____
                          bit 0     bit 1
```

For all lines except s6b high means high (Vdd!) and low means low (GND!). For s6b, T's means valid and low means not valid.

4.4.2. tmod functionality

The tmod module is the interconnection communication structure between the transmit PNs and the receive PNs. It consists of a concentrate tree and a global receive line. Messages are generated by the transmit PNs, contend for transmission via the concentrate tree, and are ultimately received by the addressed receive PN.

As described earlier, the concentrate tree is a binary tree with a communication switch at each node. Each switch enforces a bandwidth slice priority scheme in deciding which of the potentially contending messages should be sent (given priority). Messages geting through to the top switch in the concentrate tree are sent along the global receive line to the receive PNs.

The highest level concentrate tree switch (Swith6) plays a pivotal function in the TBH Test Chip. Switch6 receives and implements the RUNINP line.

When the TBH Test Chip is running (ie, RUNINP asserted) and Switch6 sees RUNINP go low, Switch6 continues the current bit transfer (if any are in progress) to outmod (the receive PNs) and then blocks further bit transfers. In this way the TBH Test Chip is halted.

On RESETINP asserted, all concentrate tree switches will initialize to their powerup state.

5. Miscellaneous Helpful Information

Inmod signals valid as soon as data is written to a transmit register. The valid signal is not qualified by RUNINP. Tmod does not require this, but will accept it. When inmod sends the last bit of a message and AUTOINP is not asserted, inmod drops the valid line on PH02 (instead of at the beginning of the next PH01).

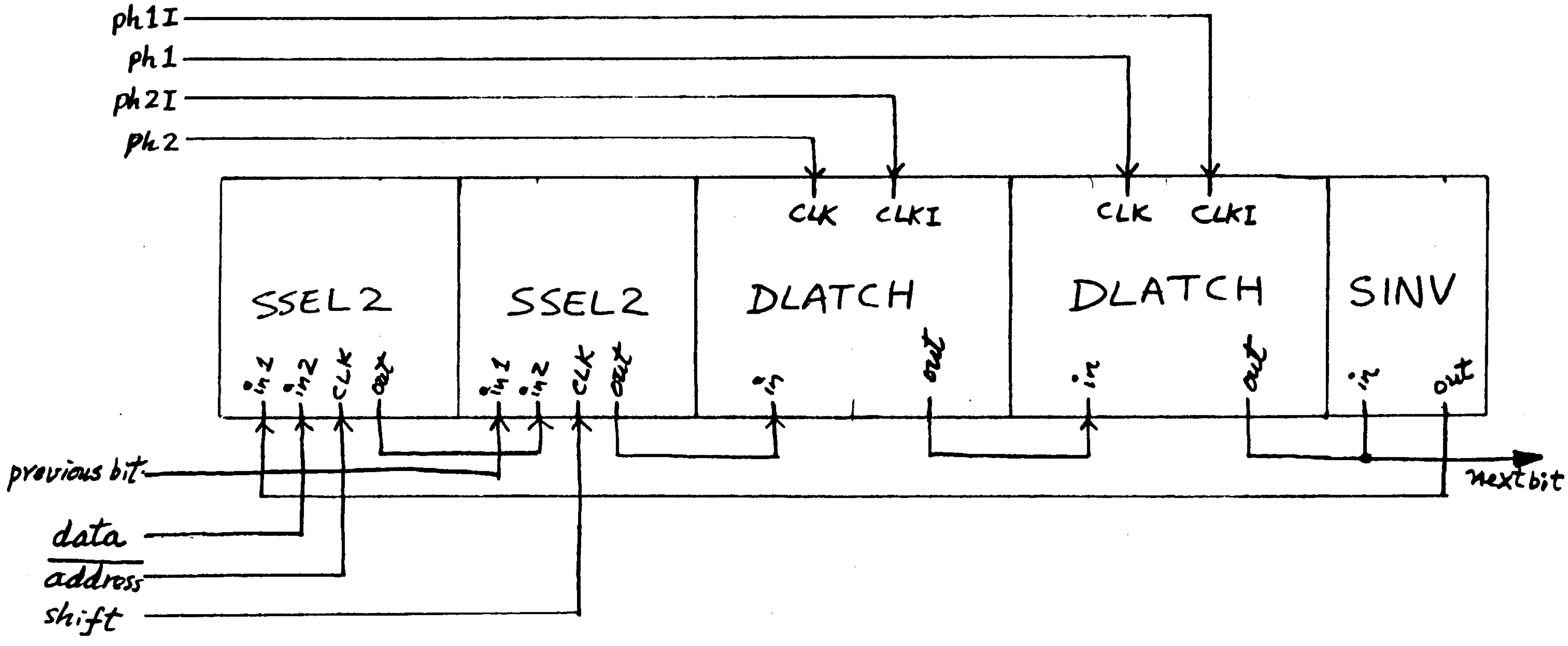
To load inmod PNs, the address and data lines must be valid before the beginning of PH02. In practice, asserting the address and data lines during PH01 works well. Also, WRITEINP and RUNINP must be asserted mutually exclusively. This means there must be at least one phase of null cycle after WRITEINP drops and before RUNINP is asserted. RUNINP must become true after PH02 drops and before PH01 starts. Finally, WRITEINP is a PH02 valid signal and must remain valid througout PH02.

In tmod, only the output from each switch is RUNINP qualified. This means when RUNINP drops each leaf node level switch will see the inmod register's valid line true (for those that are true) and load in a potentially extra bit as data. Because of this and in general, RUNINP must remain asserted during each individual run. RESETIN must be asserted for at least one complete cycle between individual runs.

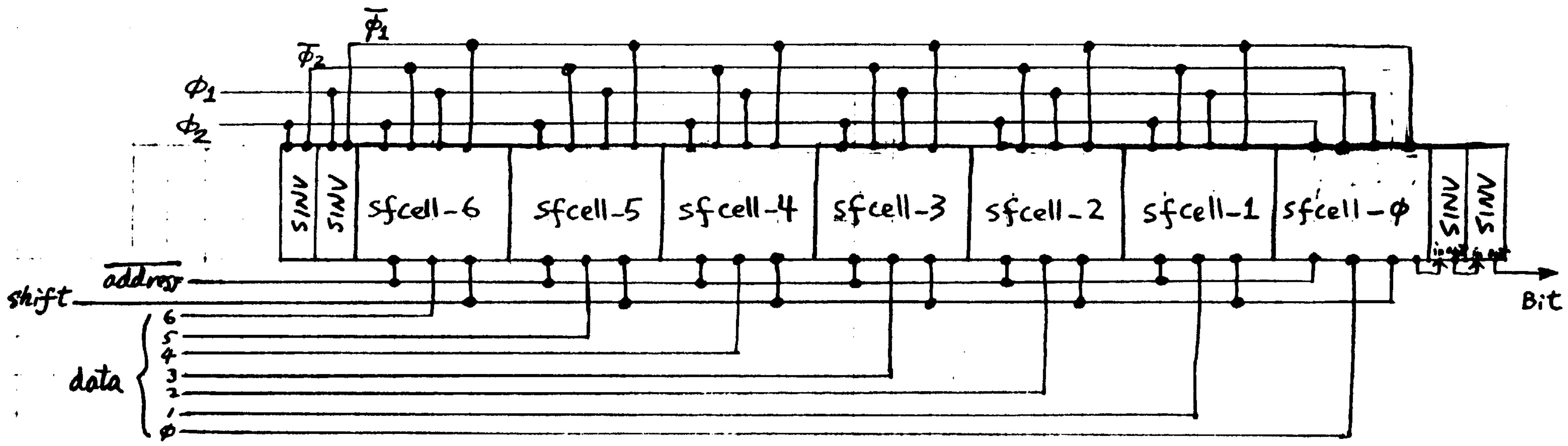
inMoD

G. HONG

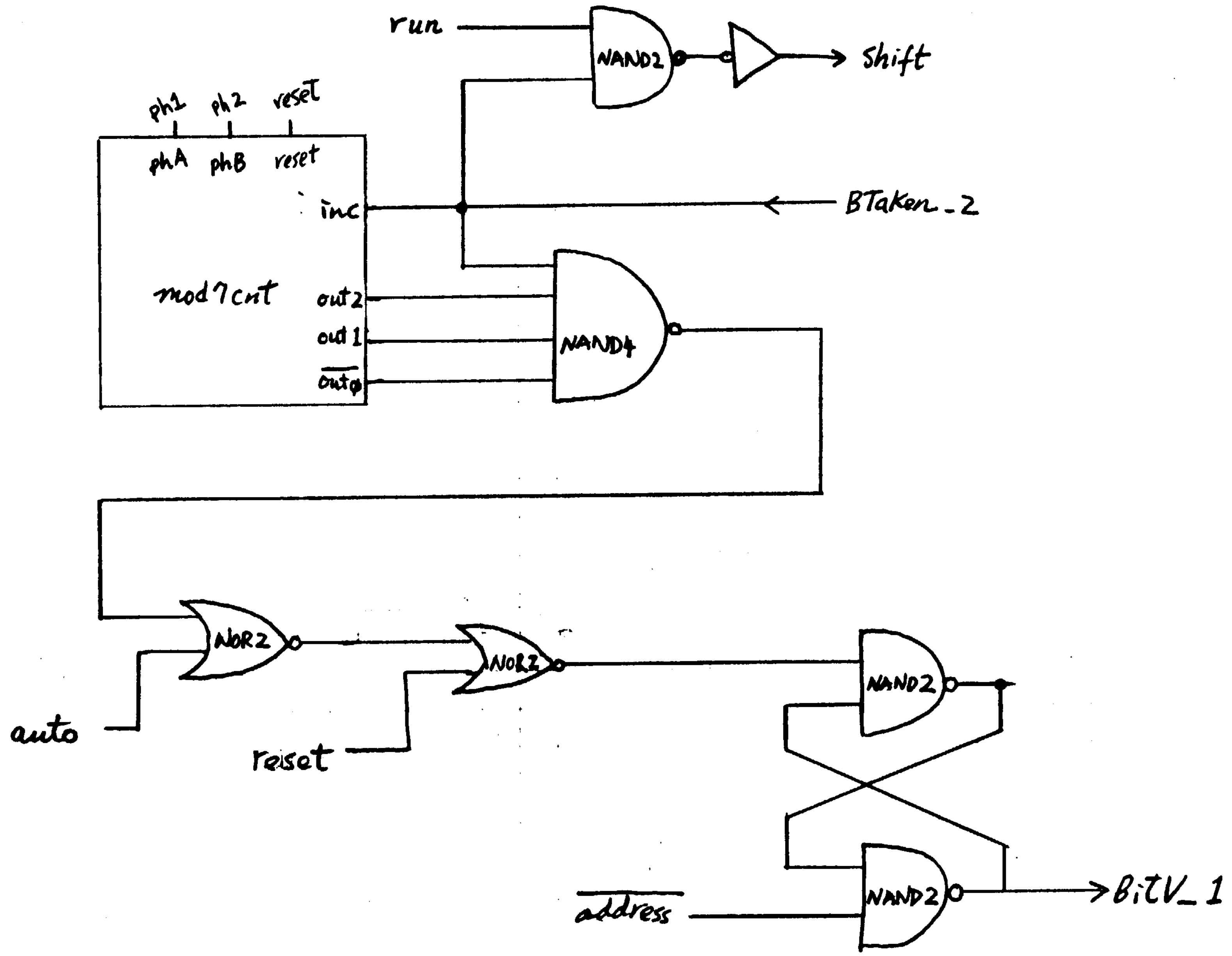
sfcell : 1-bit shift-register cell



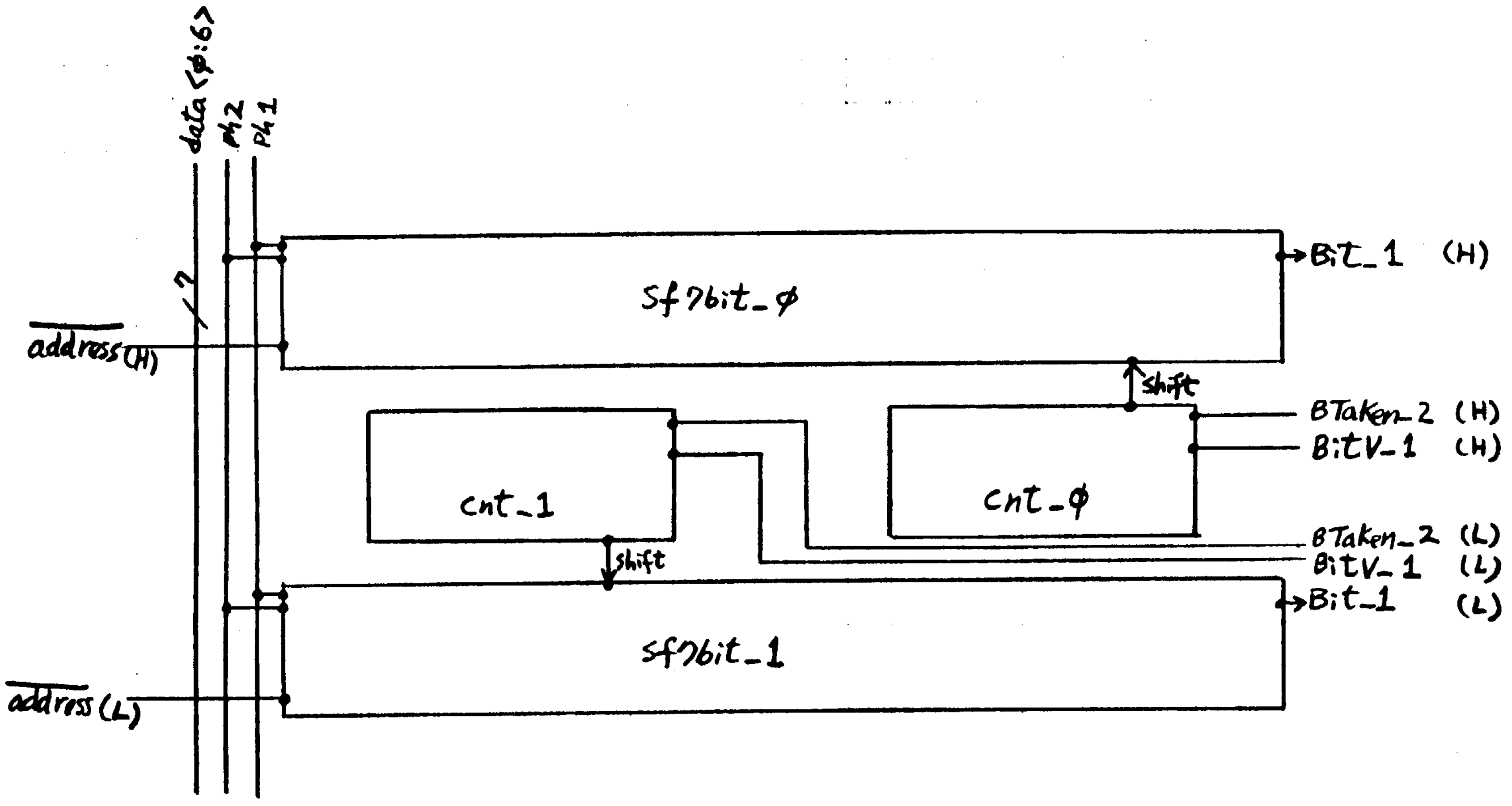
sf7bit : 7-bit shift register

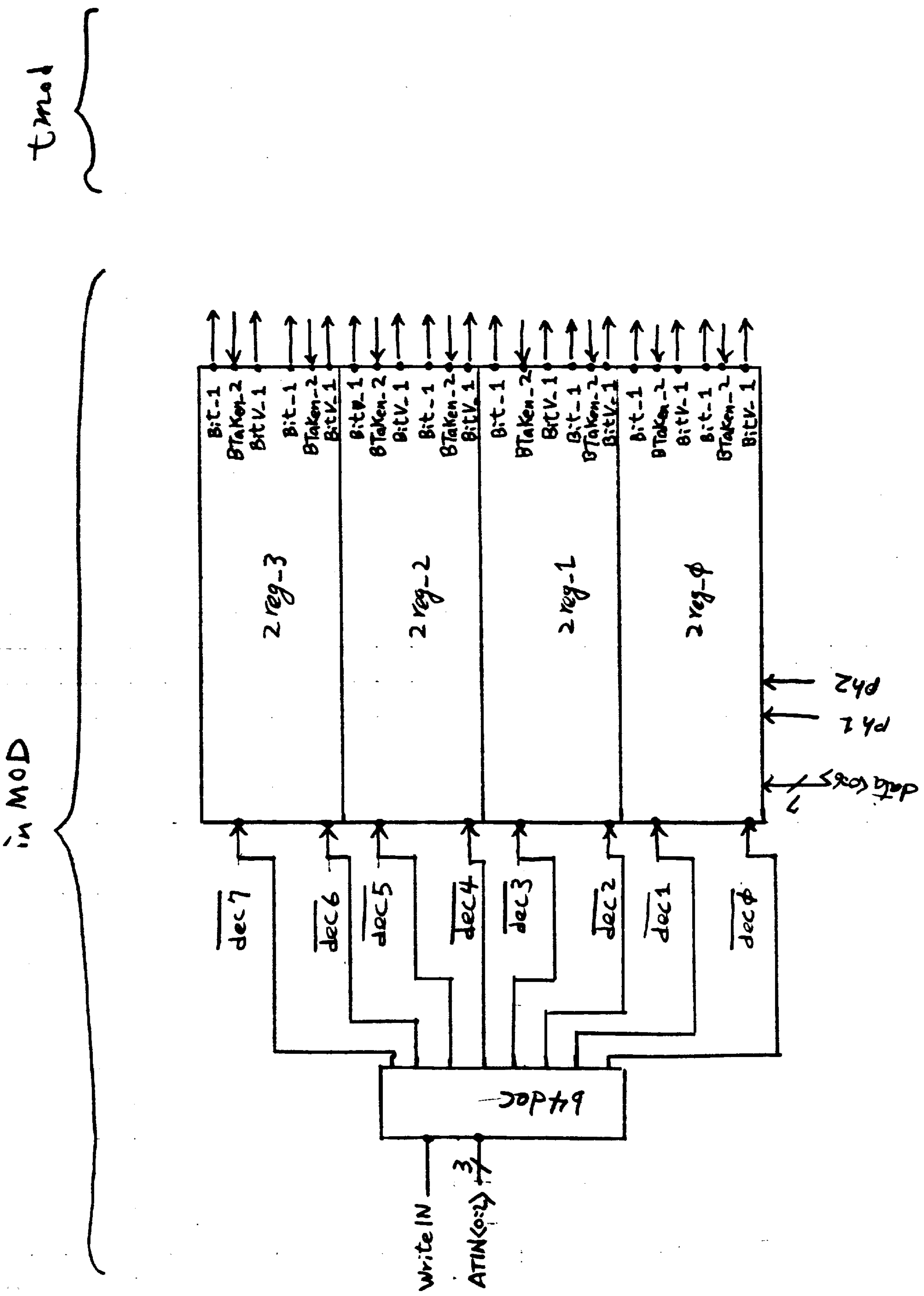


cnt: counter & control



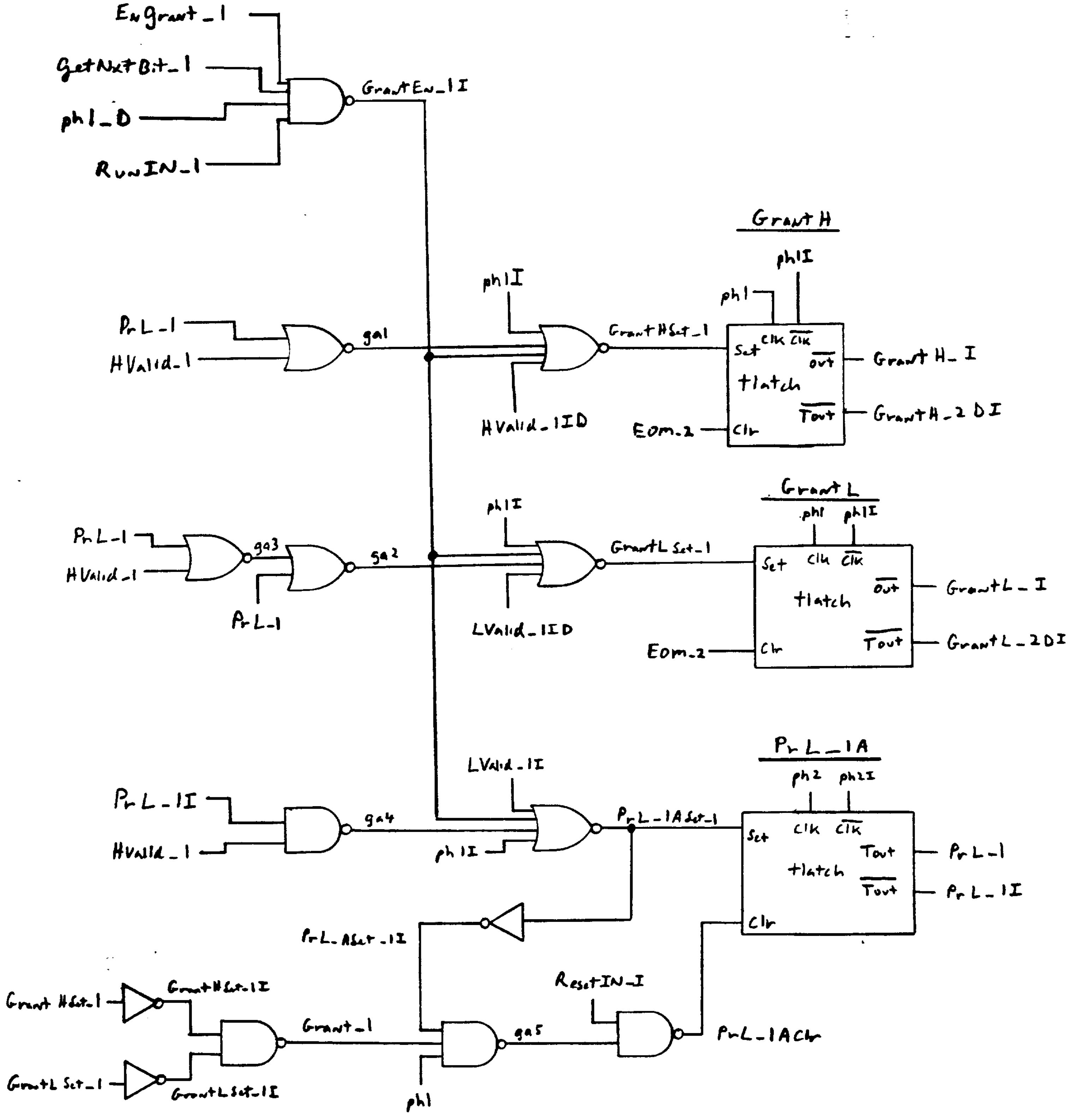
2 reg : 2 7-bit shift-registers & 2 counters



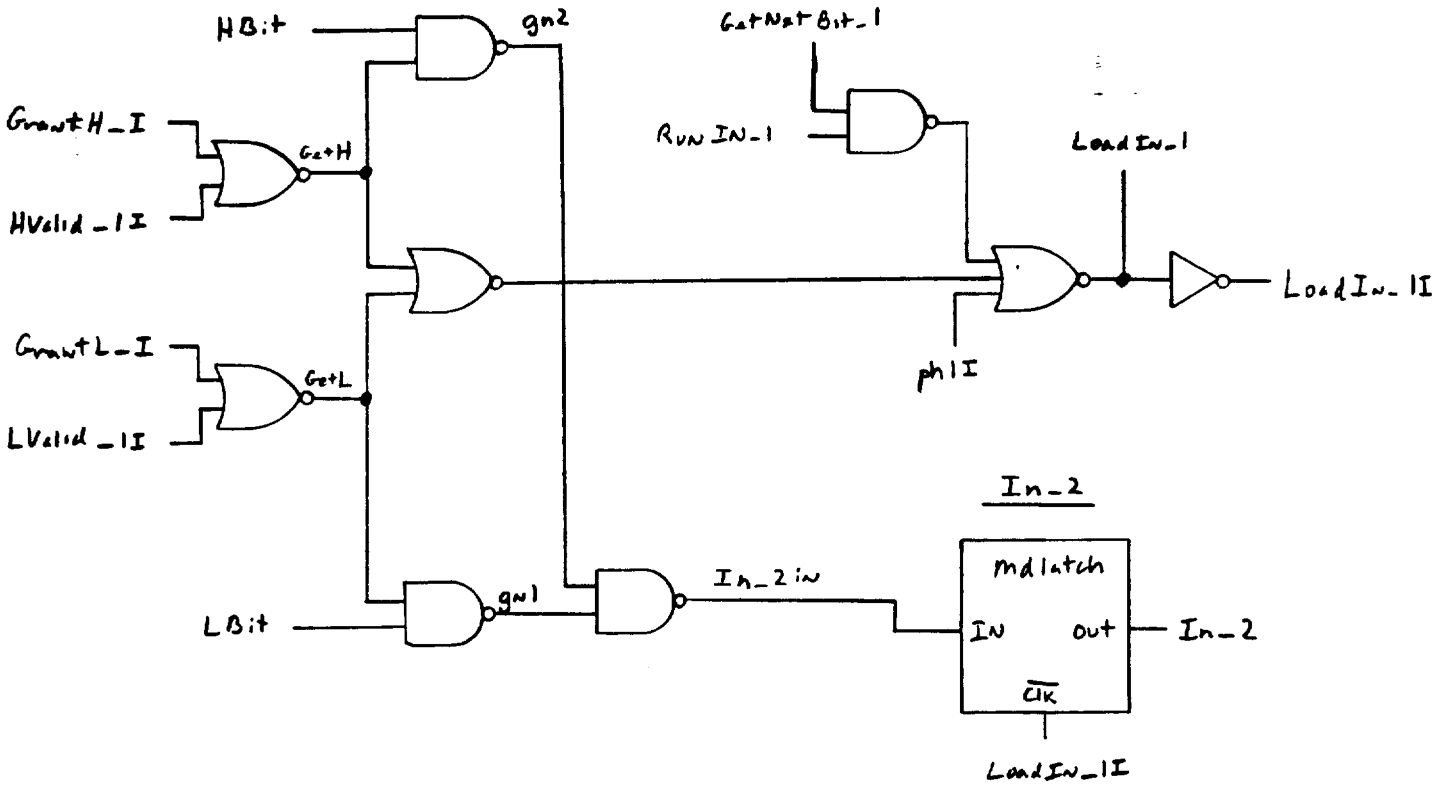


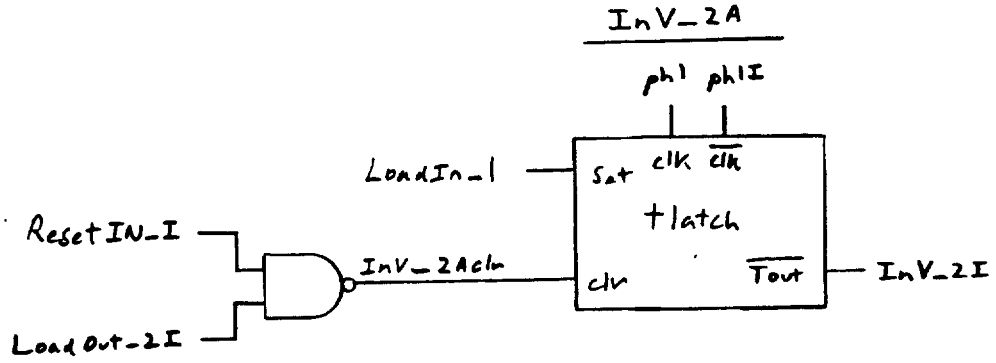
tmod
inMOD
2reg-3
2reg-2
2reg-1
2reg-φ
Bit-1
BTaken-2
BitV-1
ph2
ph1
data<0:6>
7
dec7
dec6
dec5
dec4
dec3
dec2
dec1
decφ
64dec
WriteIN
ATIN<0:2>
3

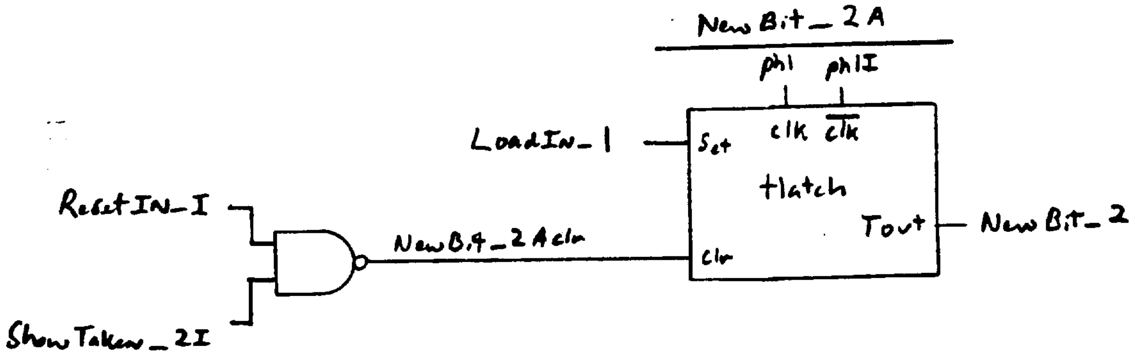
ga: Grant Arbitration Logic



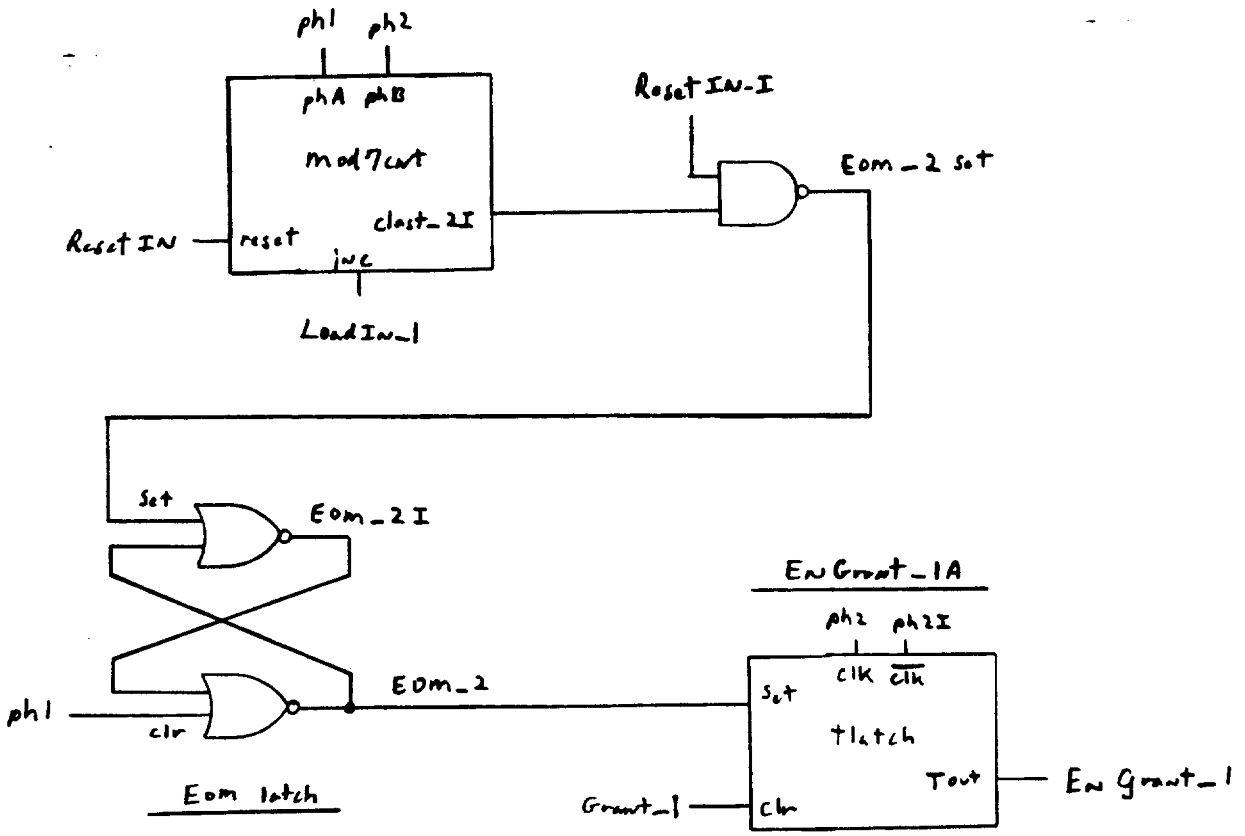
gn: Get Next Bit Logic



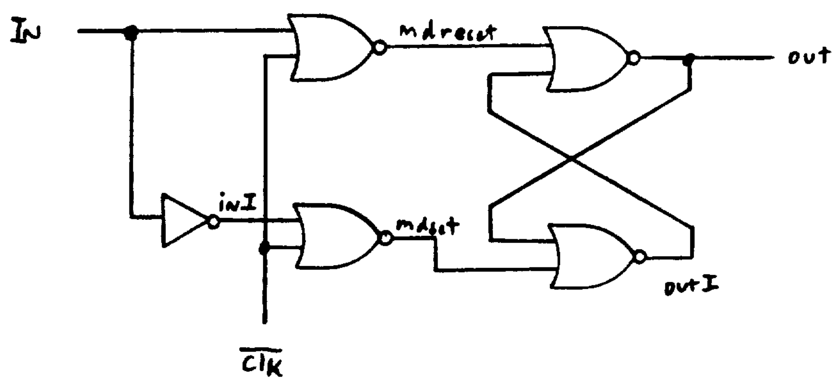




ce: Counter / Eom

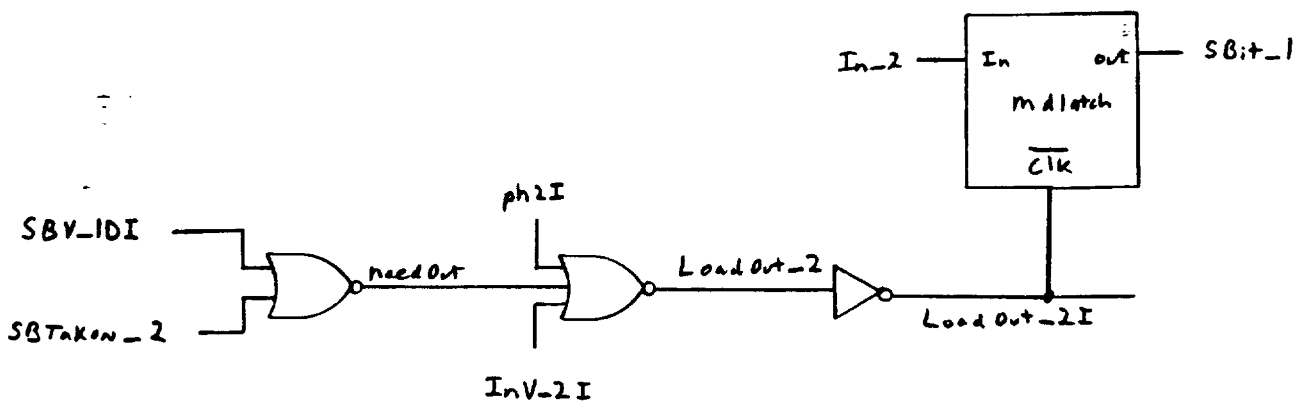


md latch

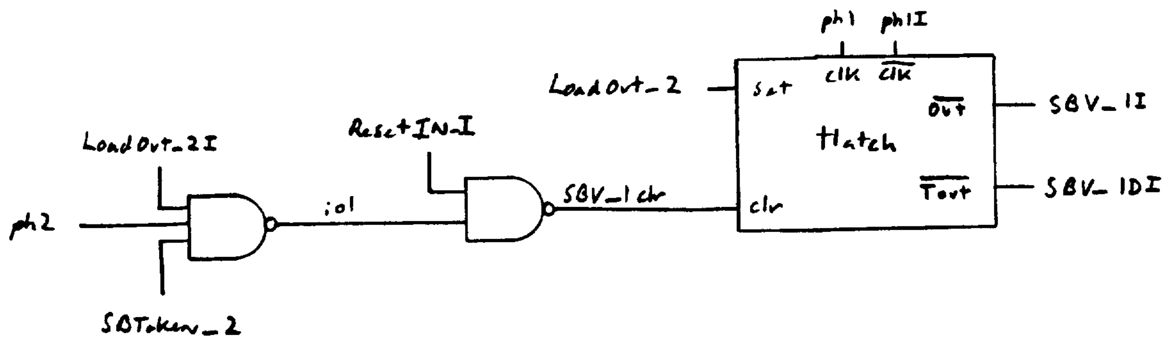


io: In - out

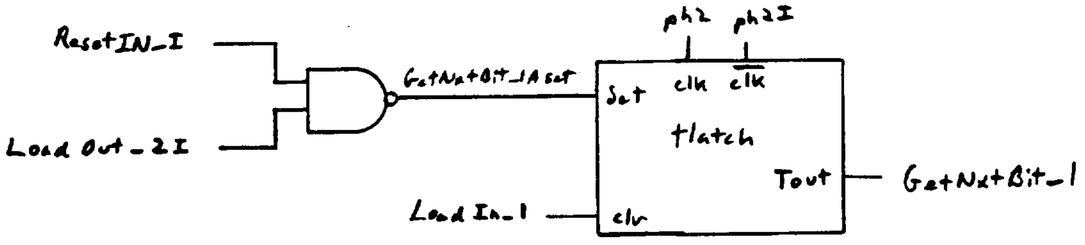
SBit_1



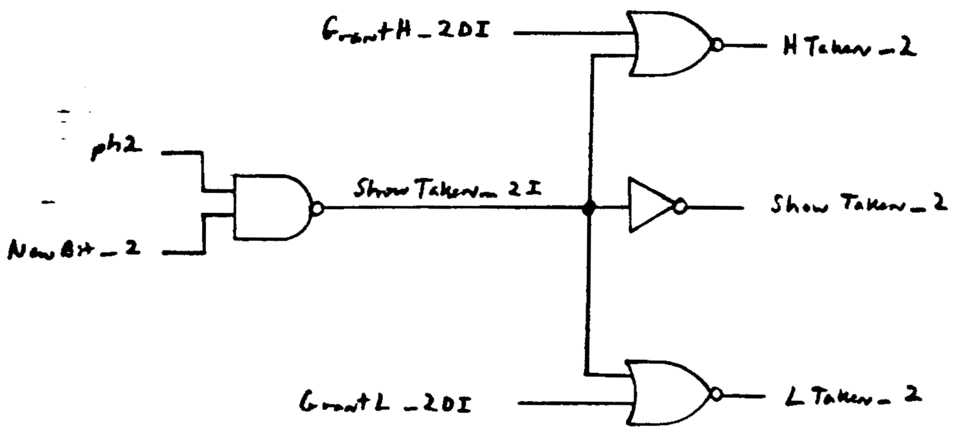
SBV_1

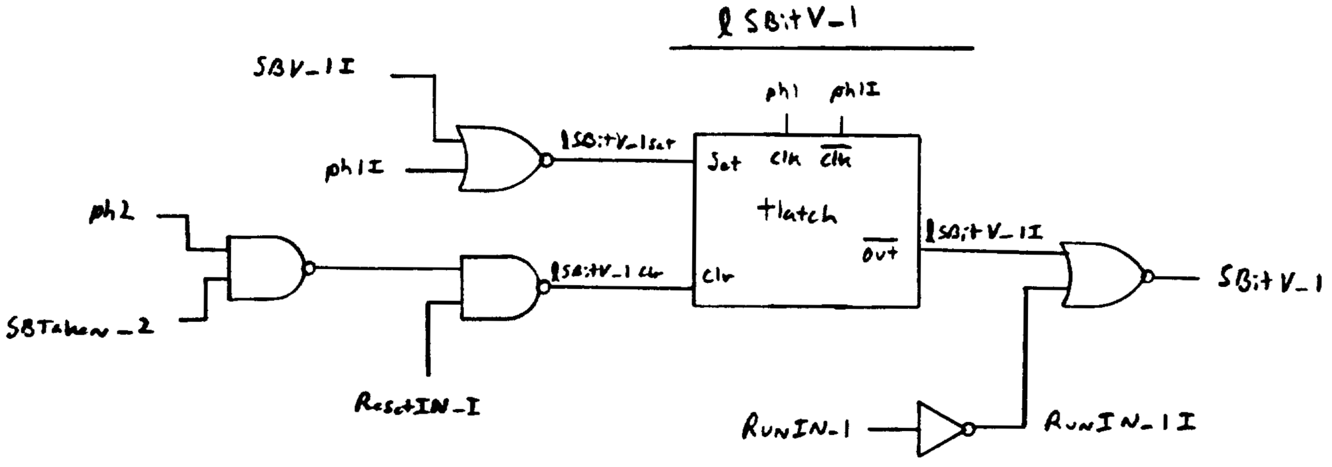


GetNxtBit_1A

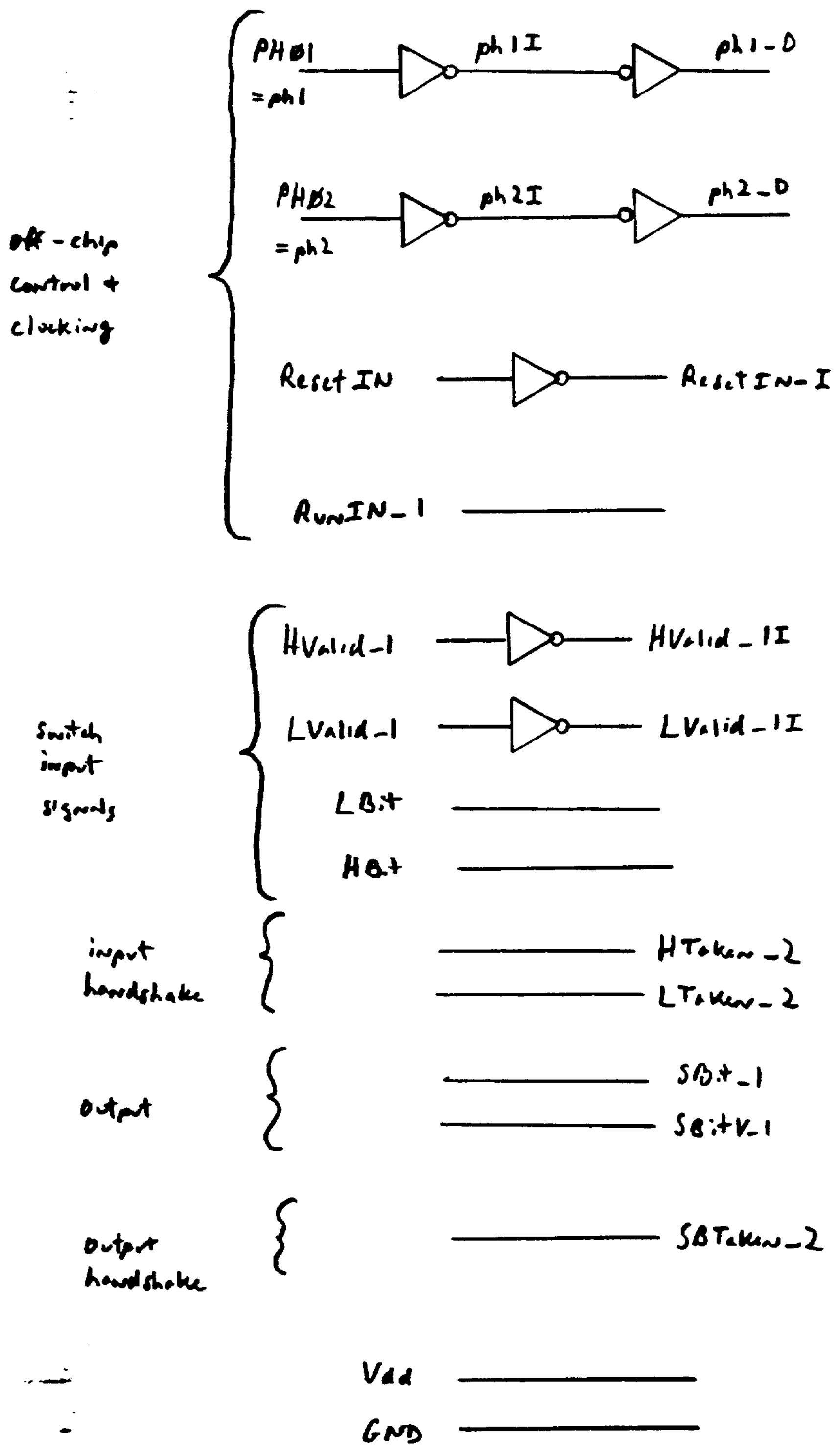


OS: Output Side

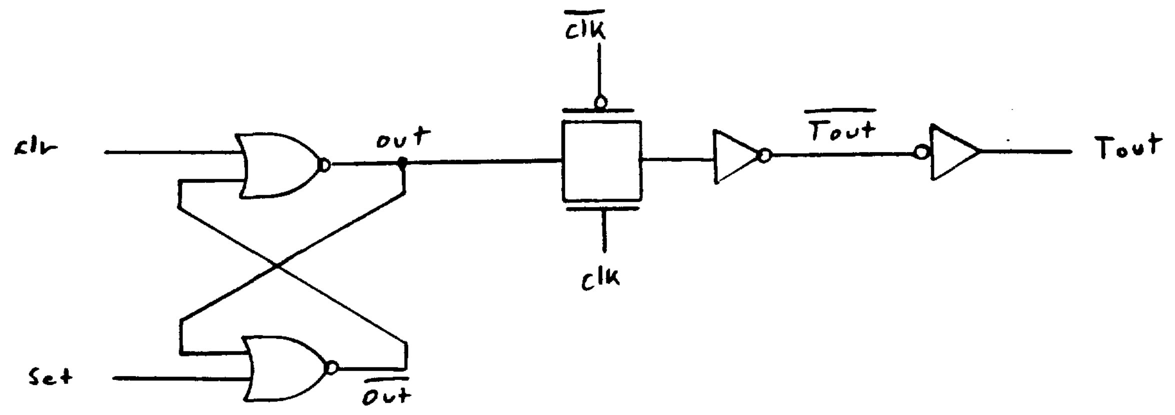




gl: switch global signals



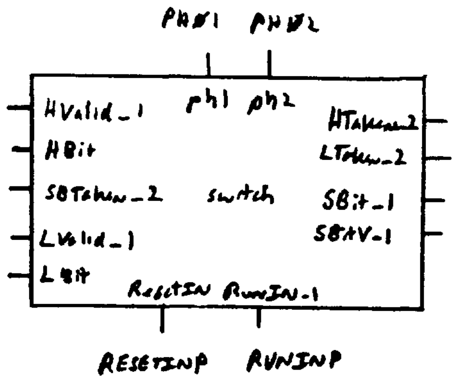
tlatch



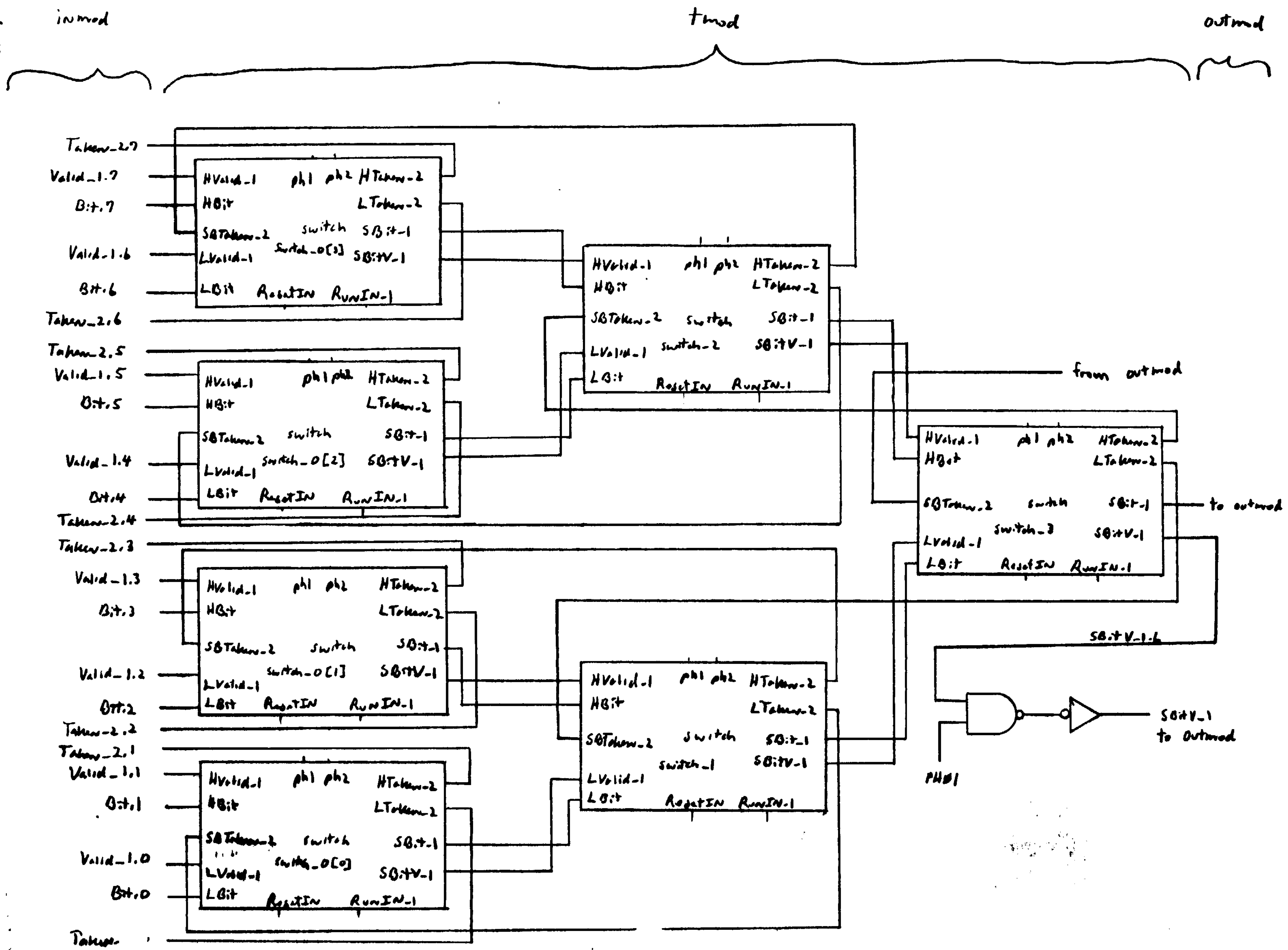
mod7cnt

see TBH outmod schematics

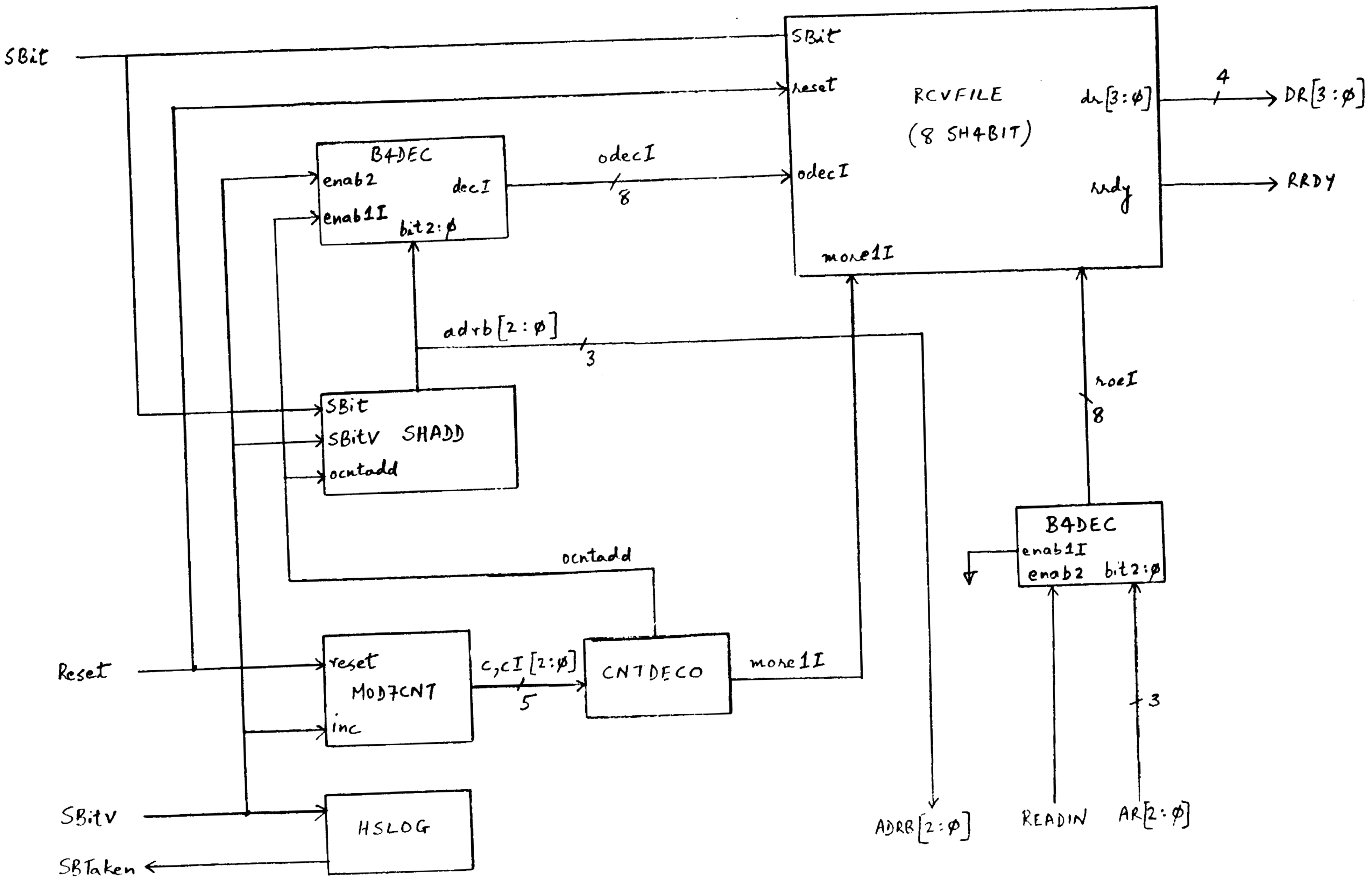
switch



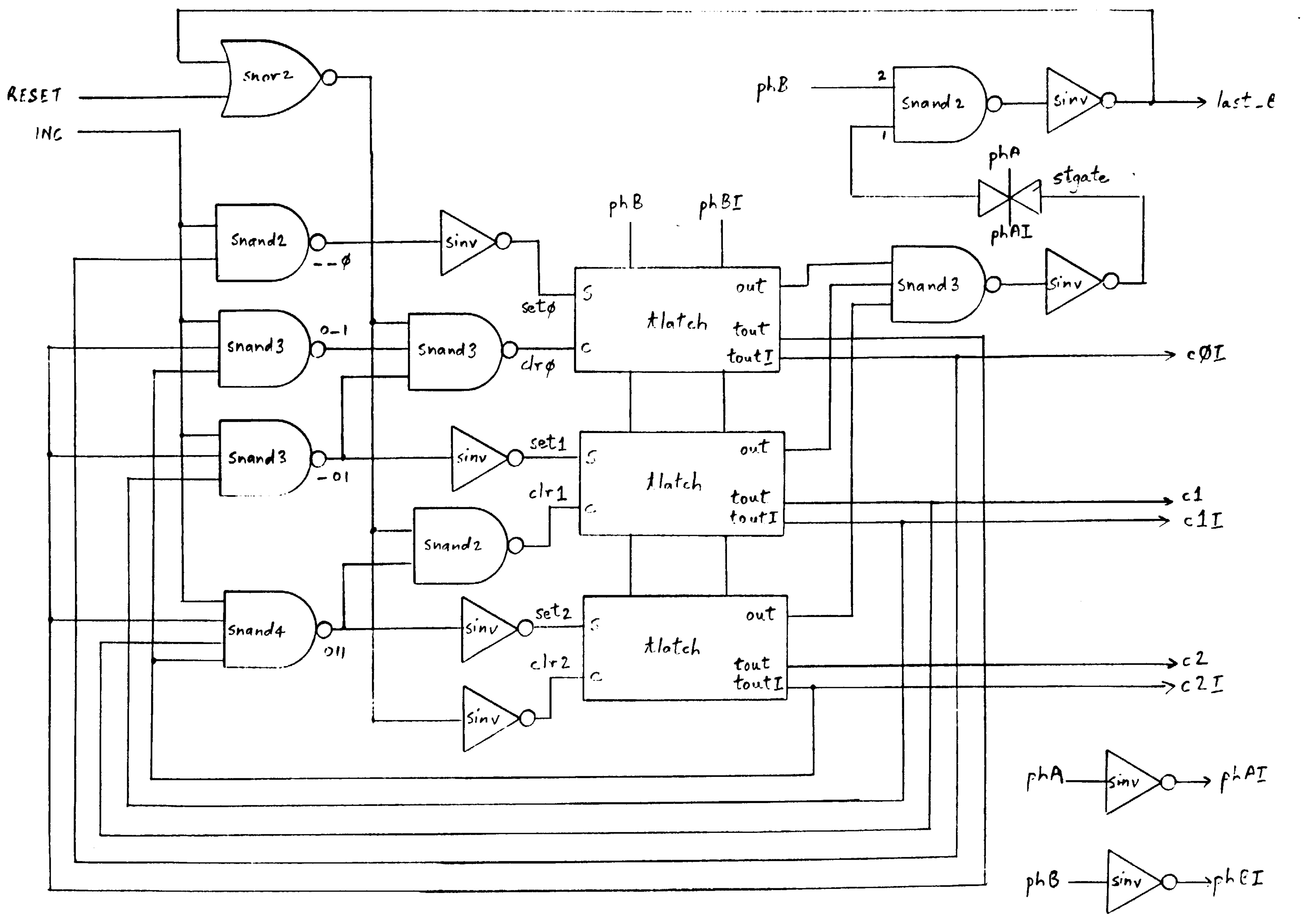
tmod-7


inmod
tmod
outmod
Token_2.7
Valid_1.7
Bit.7
Valid_1.6
Bit.6
Token_2.6
Token_2.5
Valid_1.5
Bit.5
Valid_1.4
Bit.4
Token_2.4
Token_2.3
Valid_1.3
Bit.3
Valid_1.2
Bit.2
Token_2.2
Token_2.1
Valid_1.1
Bit.1
Valid_1.0
Bit.0
Token
HValid_1
HBit
SBToken_2
LValid_1
LBit
ph1 ph2
HToken_2
LToken_2
switch
SBit_1
SBitV_1
ResetIN
RunIN_1
switch_O[3]
switch_O[2]
switch_O[1]
switch_O[0]
switch_2
switch_1
switch_3
from outmod
to outmod
SBitV_1.6
PHI1
SBitV_1
to outmod

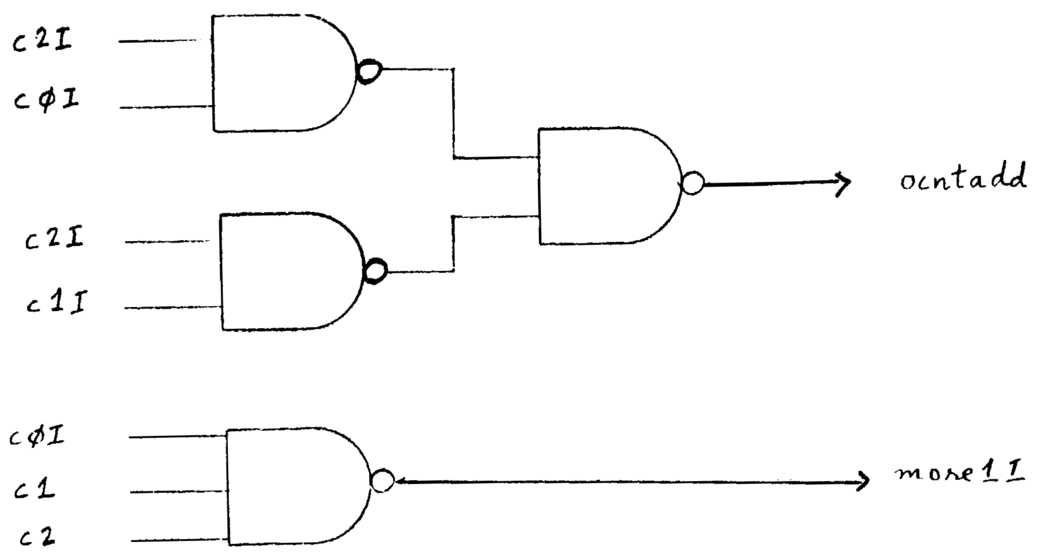
OUTMOD


SBit
Reset
SBitV
SBTaken
B4DEC
enab2
enab1I
decI
bit2:ϕ
odecI
8
SBit
SBitV
SHADD
ocntadd
adrb[2:ϕ]
3
ocntadd
reset
MOD7CNT
inc
c,cI[2:ϕ]
5
CNTDECO
more1I
HSLOG
RCVFILE
(8 SH4BIT)
SBit
reset
odecI
more1I
dr[3:ϕ]
rrdy
4
DR[3:ϕ]
RRDY
roeI
8
B4DEC
enab1I
enab2
bit2:ϕ
3
ADRB[2:ϕ]
READIN
AR[2:ϕ]

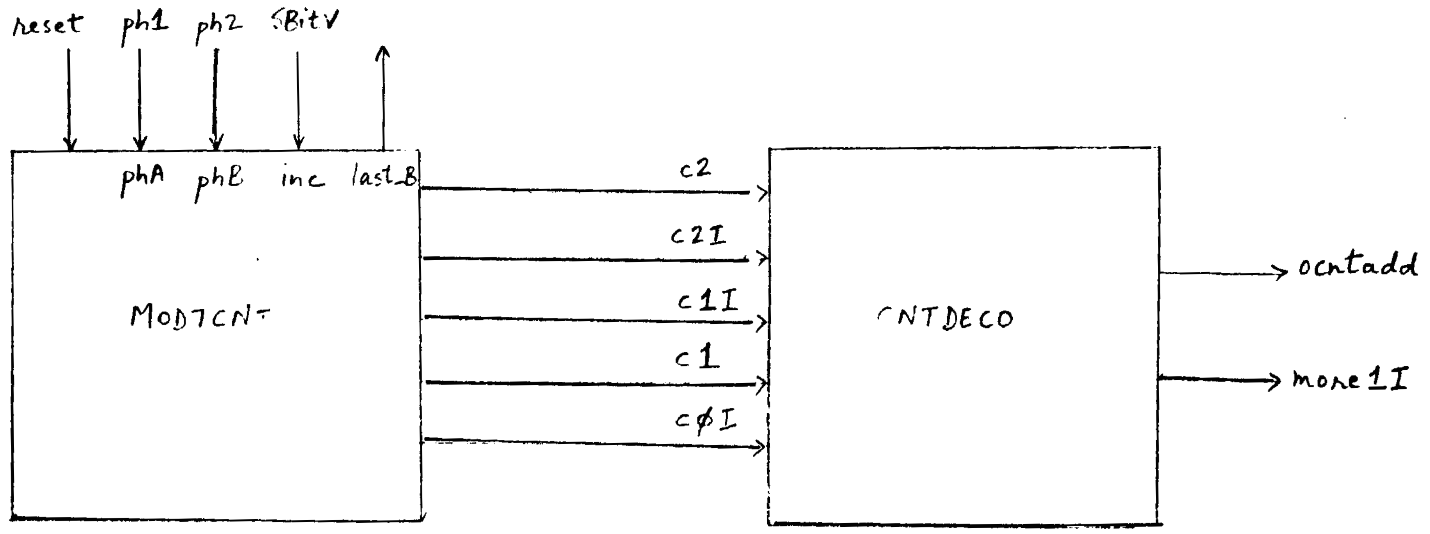
MOD7CNT



mod7cnt : modulo 7 counter for the TRH

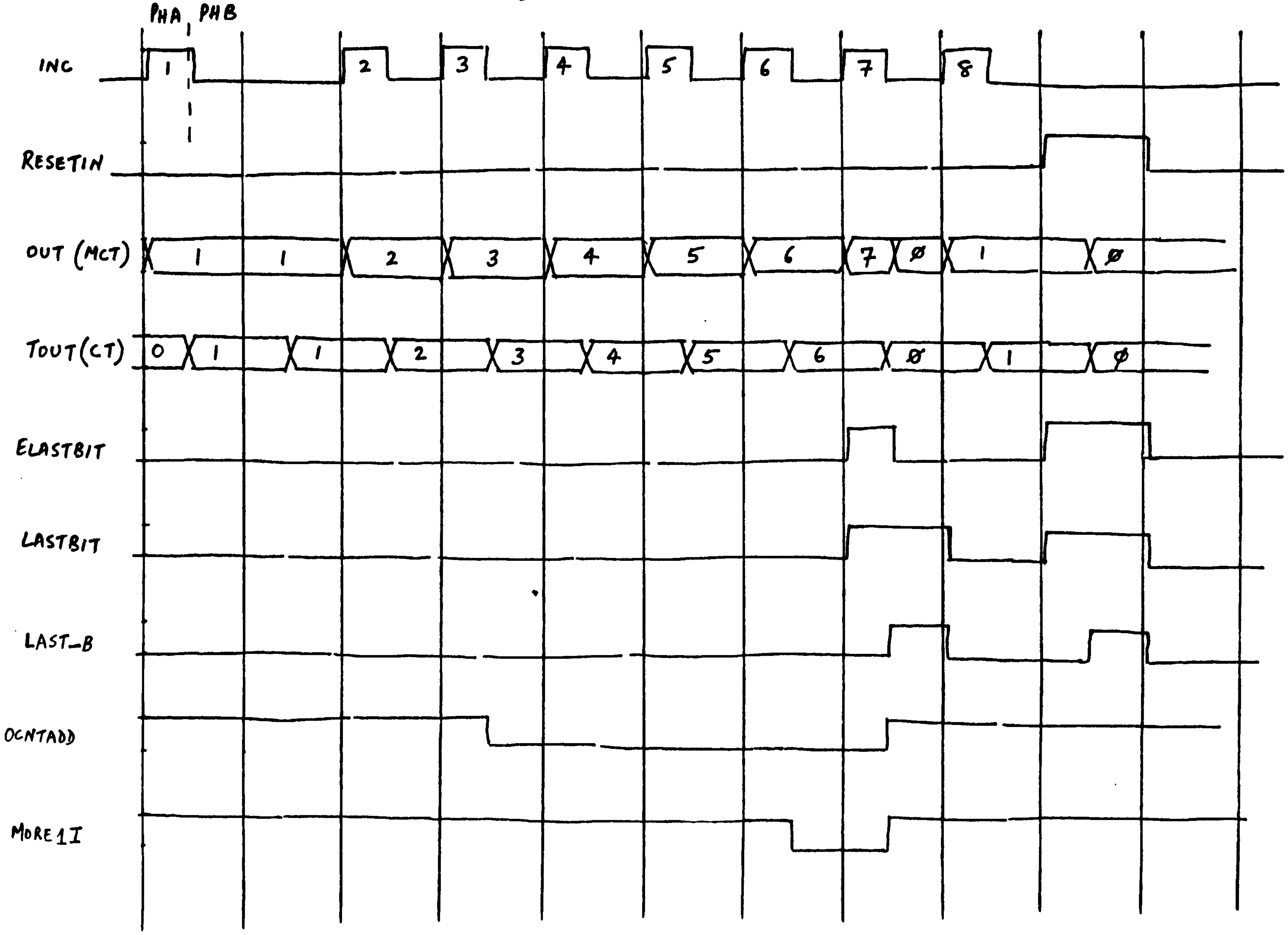


c2I
cφI
c2I
c1I
ocntadd
cφI
c1
c2
more1I

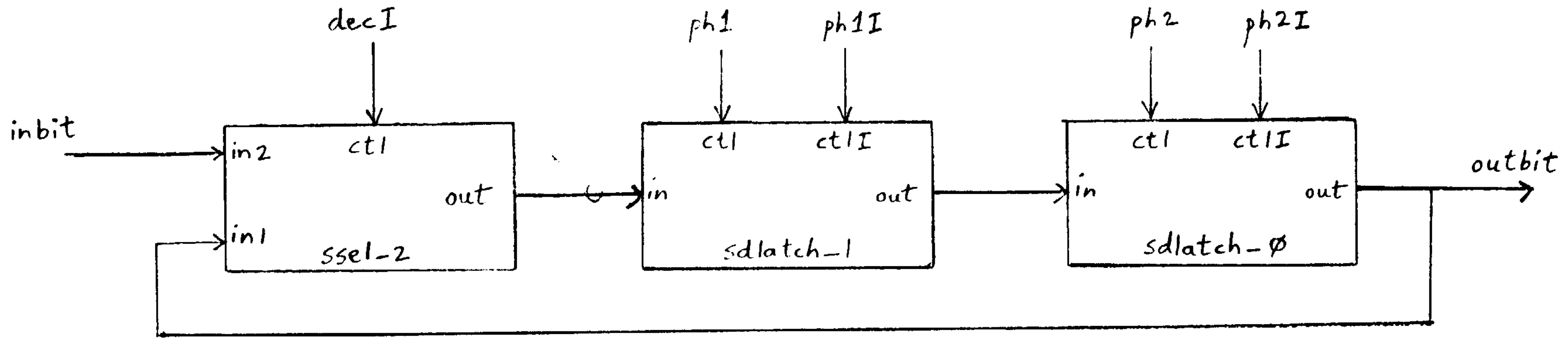


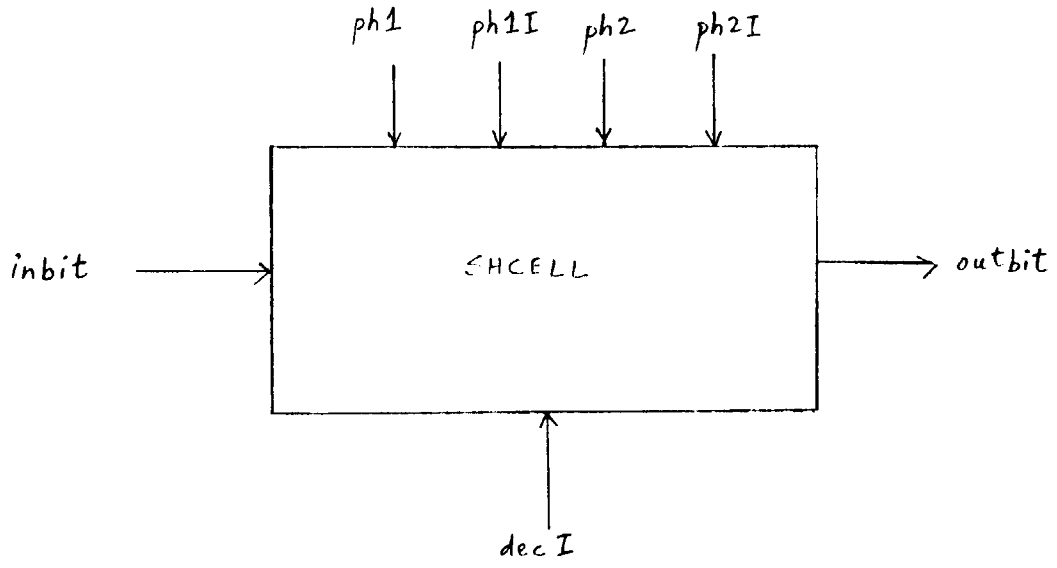
reset
ph1
ph2
SBitV
phA
phB
inc
last_B
MODTCNT
c2
c2I
c1I
c1
cφI
CNTDECO
ocntadd
more1I

GENERIC MOD7 COUNTER FOR THE TBH

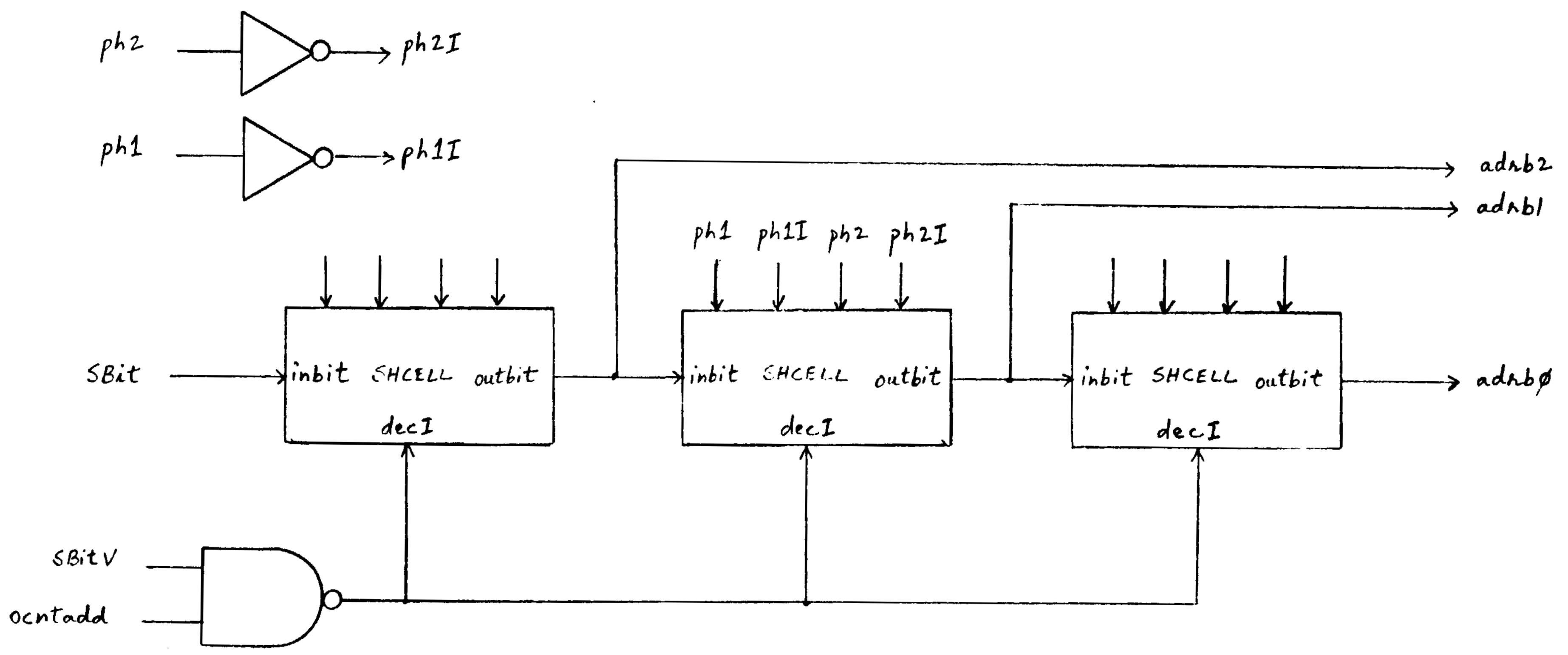


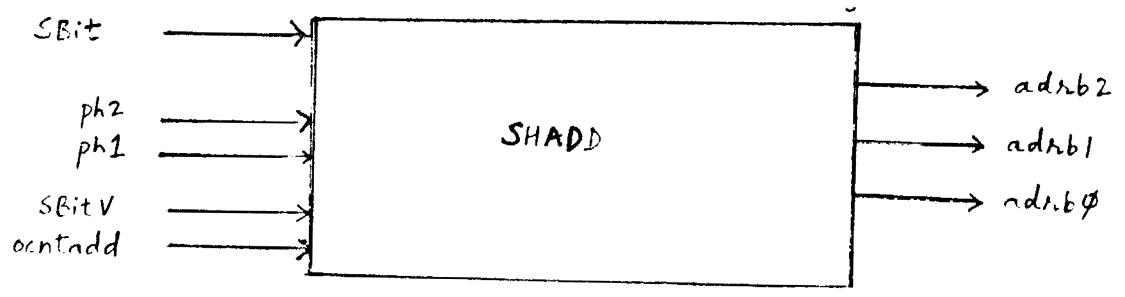
SHCELL





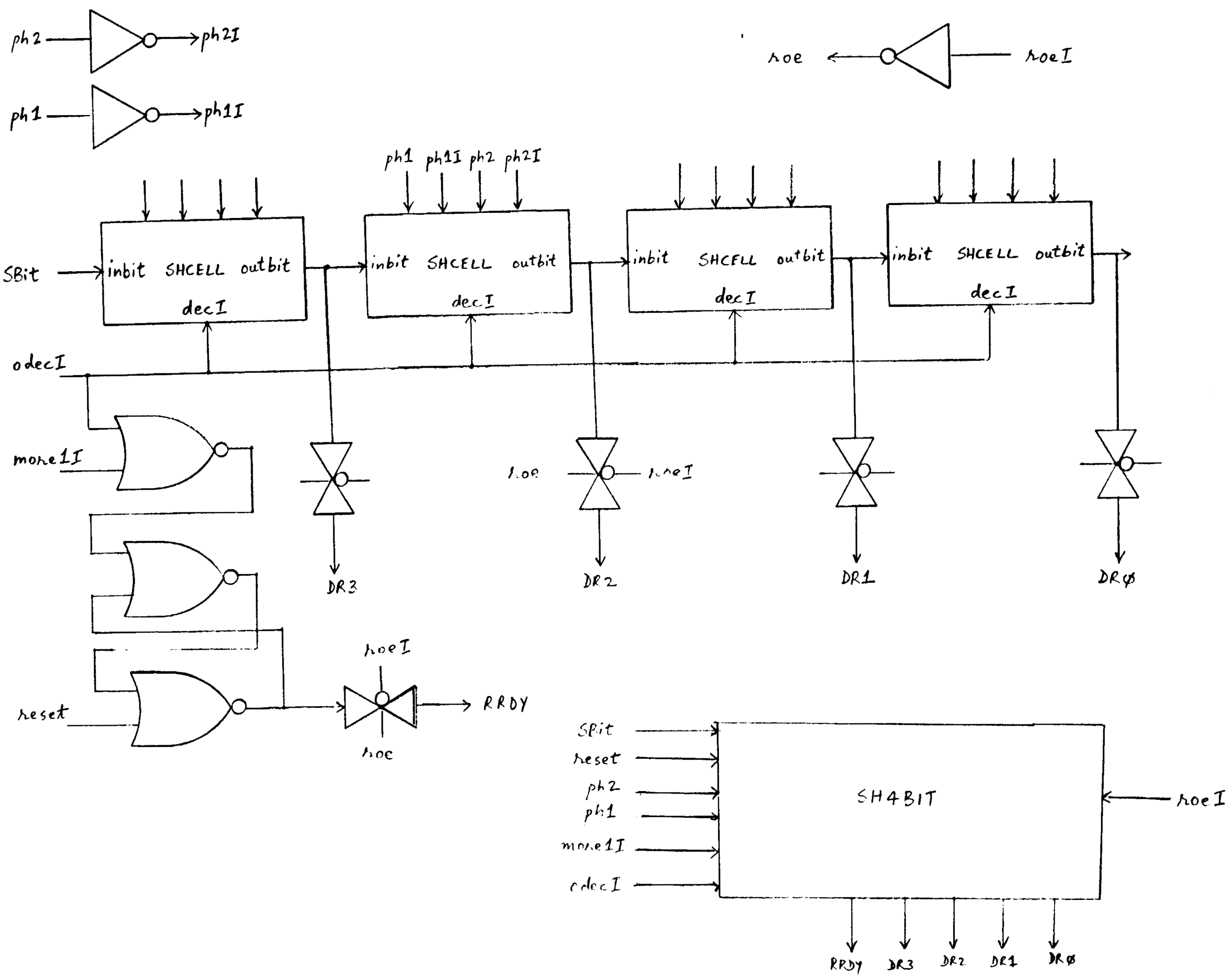
SHCELL


ph2
ph2I
ph1
ph1I
SBit
inbit SHCELL outbit
decI
ph1 ph1I ph2 ph2I
inbit SHCELL outbit
decI
inbit SHCELL outbit
decI
adrb2
adrb1
adrbø
SBitV
ocntadd

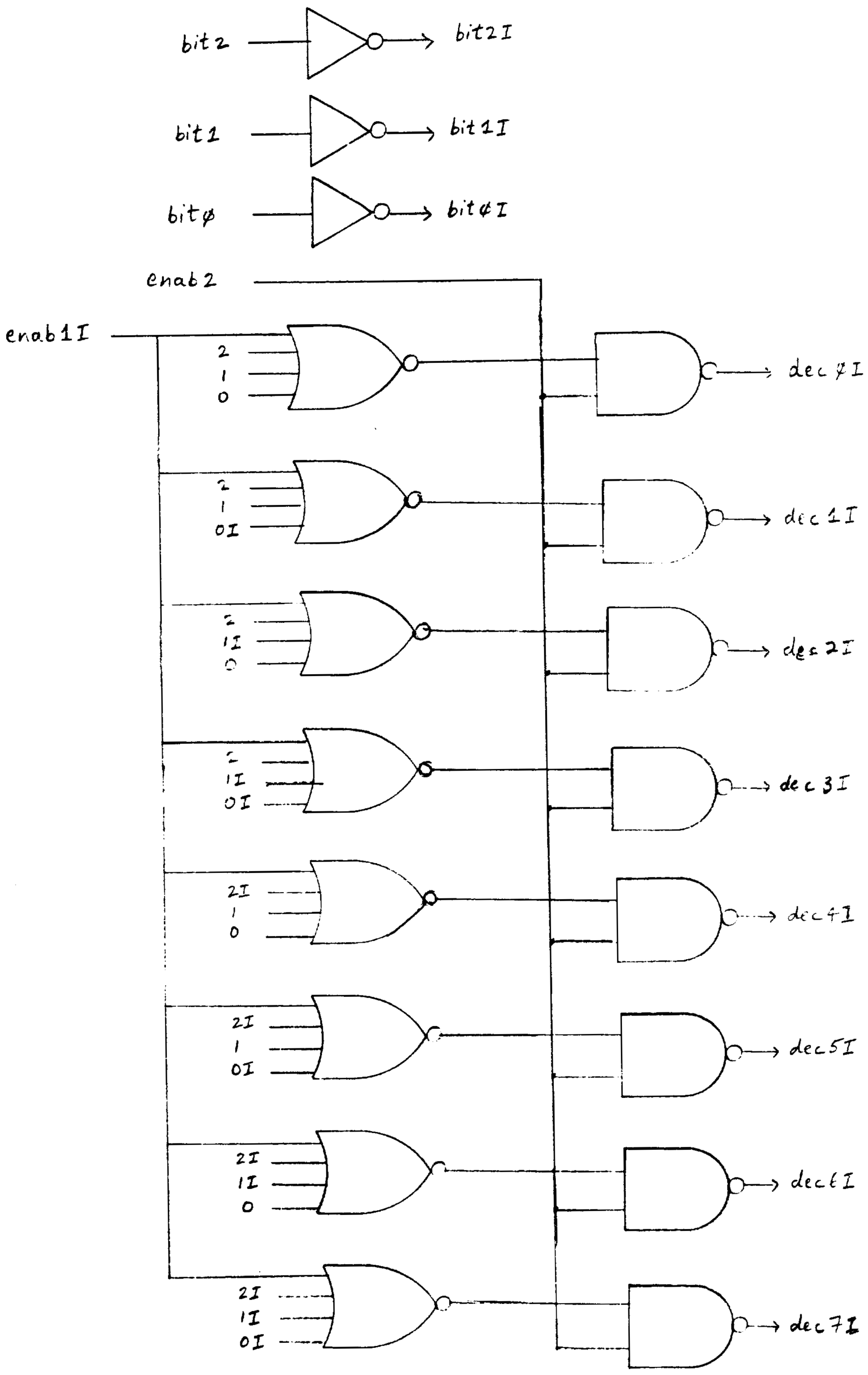

SBit
ph2
ph1
SBitV
ocntadd
SHADD
adrb2
adrb1
adrbø

SHADD

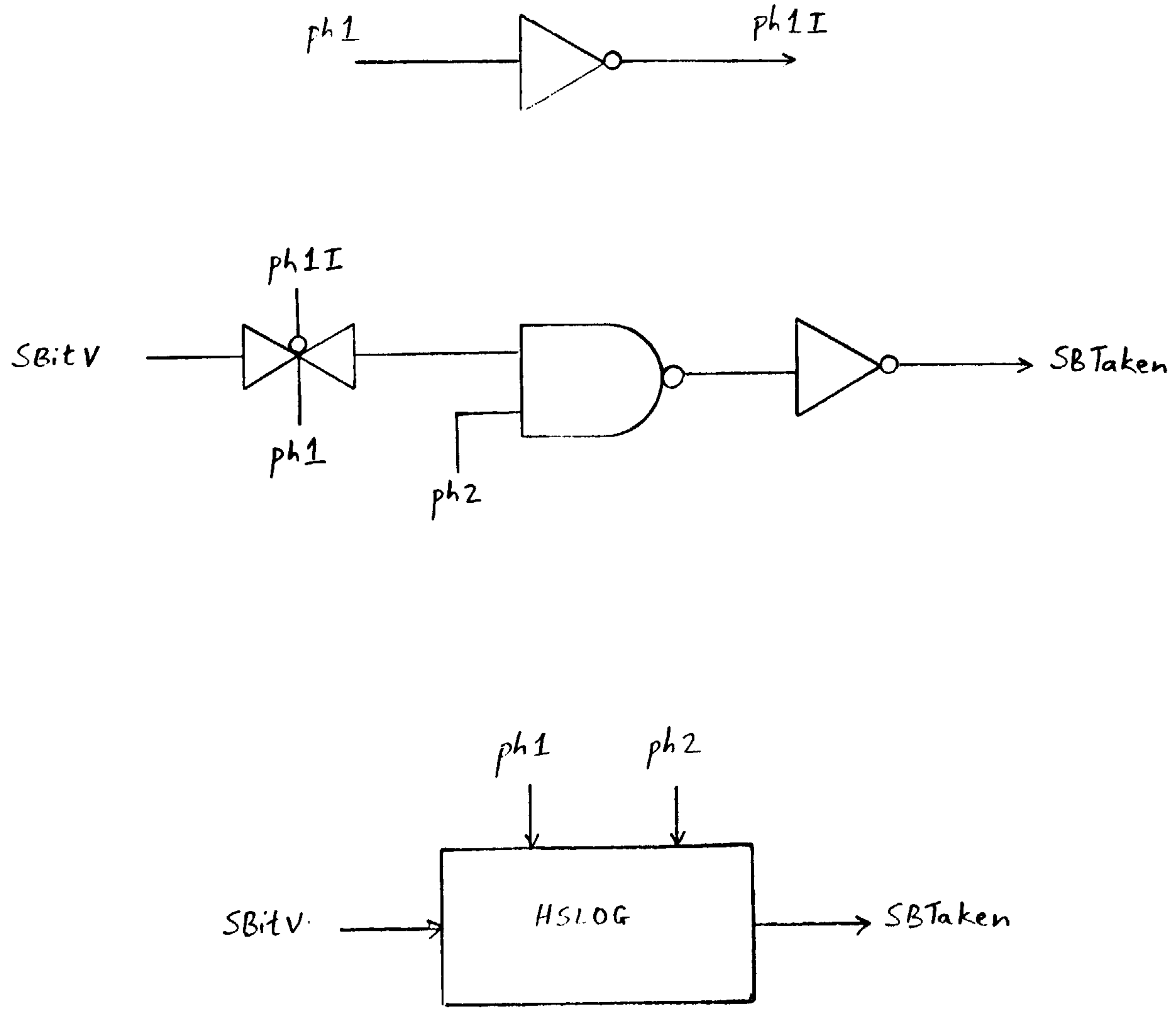
SH4BIT



SH4BIT



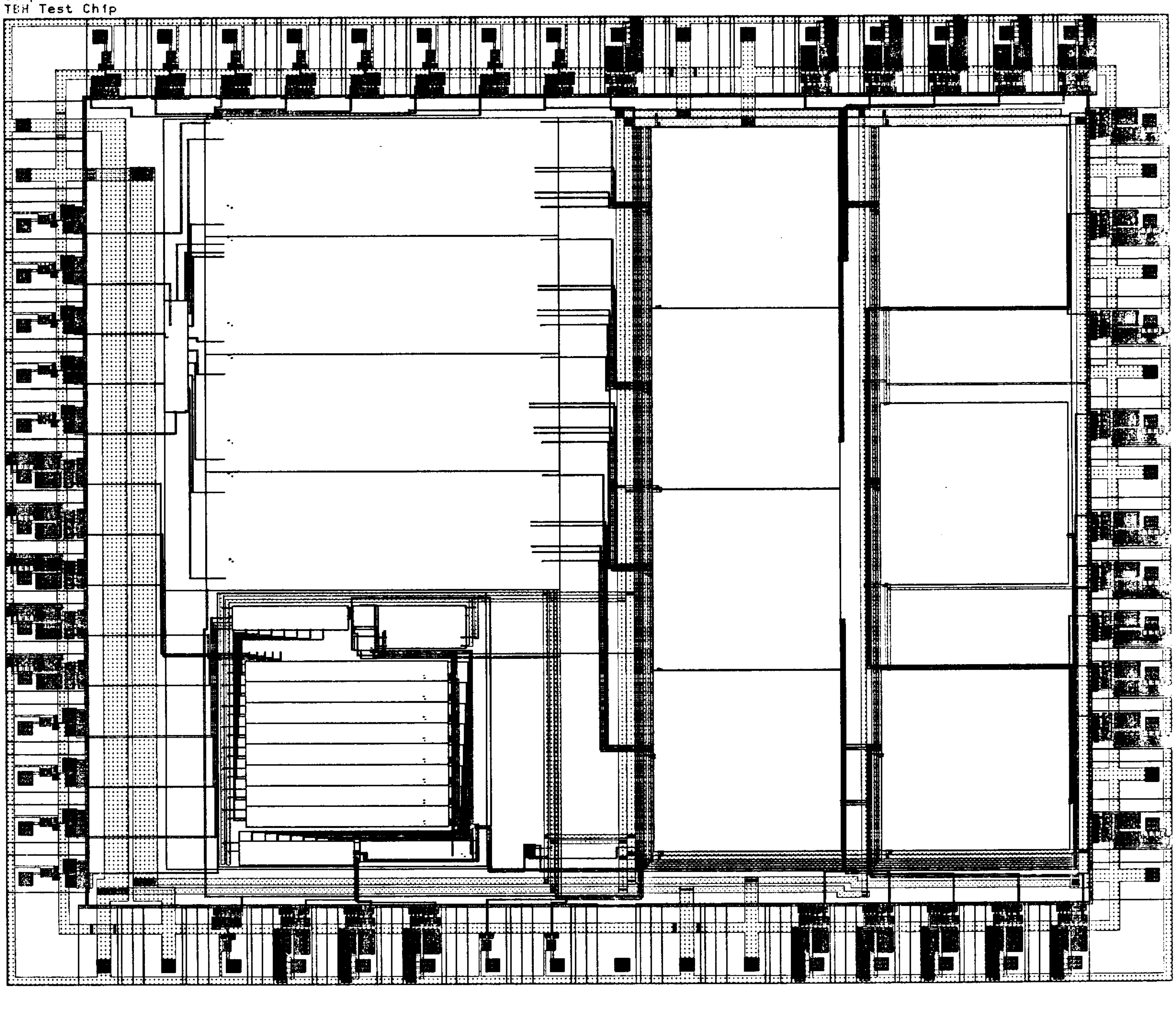
bit2
bit2I
bit1
bit1I
bitø
bitøI
enab2
enab1I
2
1
0
dec ø I
2
1
0I
dec 1I
2
1I
0
dec 2I
2
1I
0I
dec 3I
2I
1
0
dec 4I
2I
1
0I
dec 5I
2I
1I
0
dec 6I
2I
1I
0I
dec 7I



ph1
ph1I
SBitV
ph1I
ph1
ph2
SBTaken
ph1
ph2
SBitV
HSLOG
SBTaken

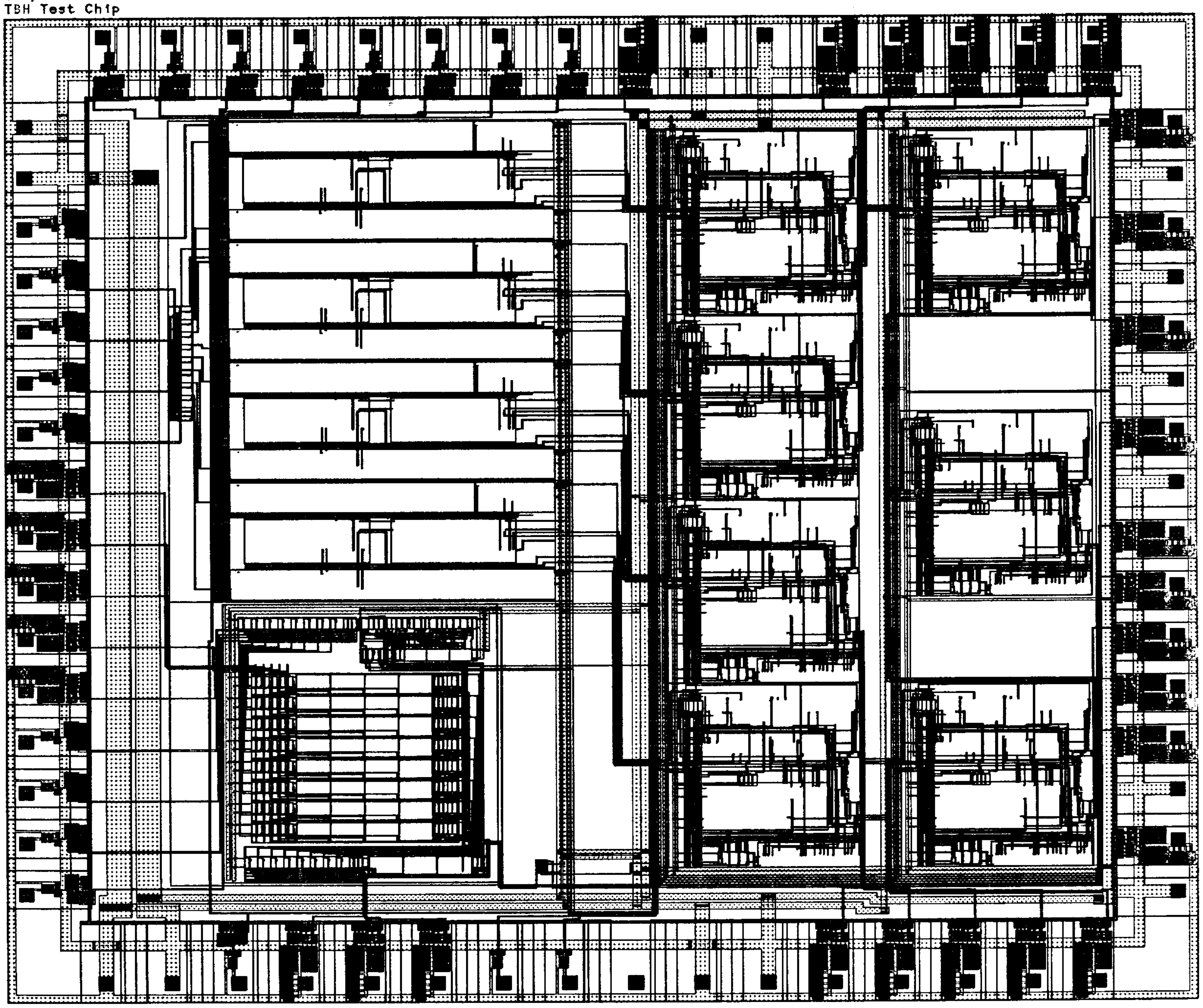
ogcvax:rudnick Job: cifplot Date: Mon Sep 28 11:47:44 1987

cifplot* Window: 9900 779850 9900 910050 --- Scale: 1 micron is 0.00117147 inches (30x)
TBH Test Chip



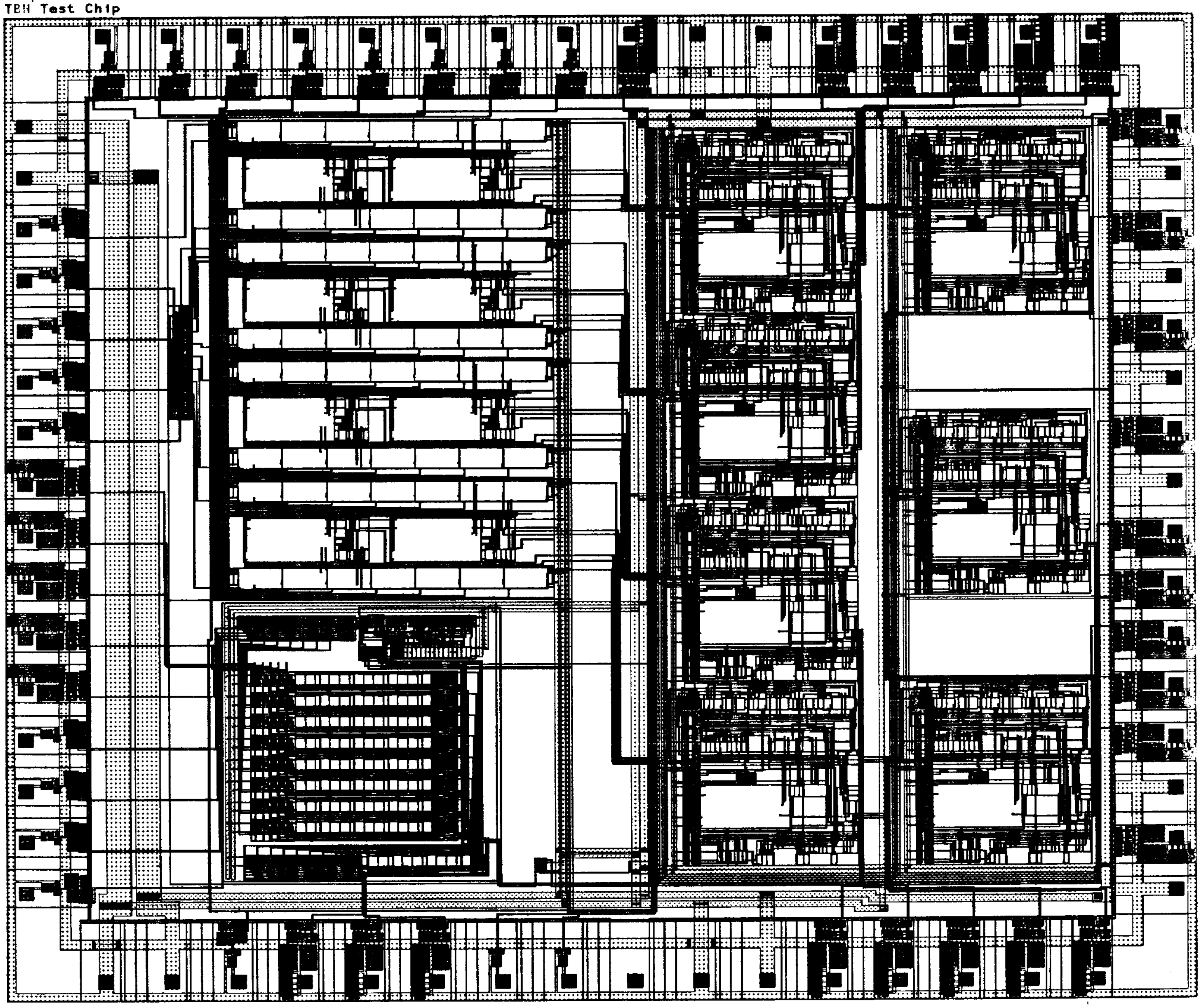
```
ogcvax:rudnick  Job: cifplot  Date: Mon Sep 28 12:06:16 1987

cifplot* Window: 9900 779850 9900 910050 --- Scale: 1 micron is 0.00117147 inches (30x)
TBH Test Chip
```



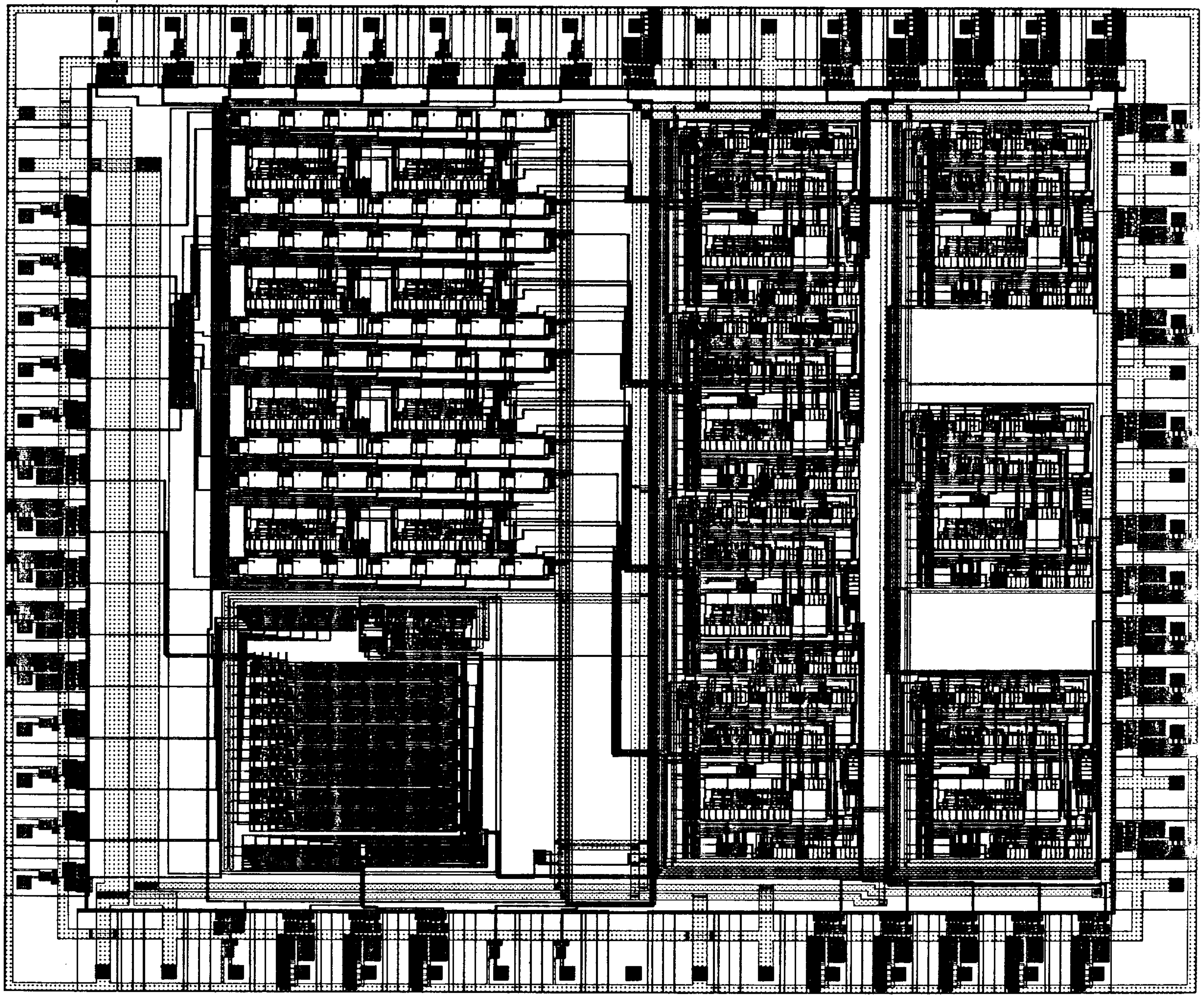
ogcvax:rudnick Job: cifplot Date: Mon Sep 28 12:43:34 1987

cifplot* Window: 9900 779850 9900 910050 --- Scale: 1 micron is 0.00117147 inches (30x)
TBN Test Chip



```
ogcvax:rudnick  Job: cifplot  Date: Mon Sep 28 13:25:10 1987


cifplot* Window: 9900 779850 9900 910050 --- Scale: 1 micron is 0.00117147 inches (30x)
TGH Test Chip
```



TBH Test Results

Four of the twenty four packaged TBH Test Chips appeared to have no flaws due to mask, packaging, or fabrication process. Most faults seemed to be fabrication "stuck high" and "stuck low" faults. The four chips that worked exhibited a design flaw which we have tracked down. In effect, the design methodology broke down (or, depending on how you look at it, we just made a mistake). This was due to our pushing to get the job done, setting aggressive design goals (one bit transfer per clock cycle through the pipeline), a partial miscommunication, and using a "shaky" design technique without proper testing (i.e., using a race-sensitive design without SPICE testing it).

Briefly, the data valid signal from the root of the concentrate tree to the "receive PNs" is supposed to be phase one trapped. Because of clock skew, this signal is going away before the trap closes. In retrospect, we should have done the following:

- added several gate delays to the signal to be trapped
- taken care to guarantee little or no clock skew to the trap logic

The result of this design flaw is that data could not flow past the root of the concentrate tree. We partly remedied this problem by laser fusing the "taken" line (running from the "receive PNs" to the root of the concentrate tree) to Vdd (i.e., locking it high). This allowed data to flow unrestricted through the tree. This also allowed us to verify that the concentrate tree was functioning correctly, which was the primary goal of the chip.

Probably the most striking thing we learned is that inter-module interfaces are the places where problems creep in most (no surprise here). We think this was especially true because different people worked on different modules. This means that not only were the module interfaces adding timing complexity, but also most miscommunication problems showed up at the interfaces. We had the experience of repeatedly "redefining" the details of the interface between the modules each time we discovered we had miscommunicated.

This suggests a tool should be developed to allow detailed specification of module interfaces and simulation of the design directly form the module interface specification. Perhaps the lead designer should take on the job of being responsible for this module interface specification and simulation as it changes throughout the design cycle. In our chip design process, the design flaw was introduced in the last days of layout and full chip simulation when we discovered we had conflicting notions of one of the details of the interface between two modules. Of course, this problem will only get worse as project size increases (and probably superlinearly).

We were surprised that the inter-SN (i.e., inter-switch-node) wiring layout was not as difficult as I had anticipated. We implemented an H_3 nonfat concentrate tree with a branching factor of $\alpha = 2$. It would have taken little more time to implement a branching factor of $\alpha = 4$, the standard PBH branching factor. Further, as long as the layout is regular and standard across, say, a wafer, a hierarchical layout can be performed using the Magic array command. This simplifies layout to the point where even for a complete wafer, the PBH layout can be done by hand.