QCHIP: A VLSI INTERFACE FOR A LOCAL NETWORK

Bill MacGregor
Steve Toner
Bolt Beranek and Newman Inc.
29 April 1981

1. Introduction

The QChip is a one-package local network interface for low to moderate bandwidth data transmission, operating on the store-and-forward principle. QChips can be connected in a ring topology with no additional components, and rings can be interconnected by active switching elements to form larger networks. Figure 1-1 shows how terminals, computers, and instrumentation can communicate via a QChip local network. If QChips are in close proximity to one another (e.g., within the same cabinet) no additional circuitry is needed between them; if the links between nodes are longer, twisted pair or optical fiber media may be used to connect QChips with the addition of driver and receiver circuits.

The store-and-forward nature of the design distinguishes QChip from all other local network designs of which we are aware. The properties of this technique have not been fully explored for the combination of interface speed, size, and cost represented by the QChip design. Two important aspects of the approach are the confinement of some types of transmission errors to the active sender-receiver pair and the ability to support expansion of the

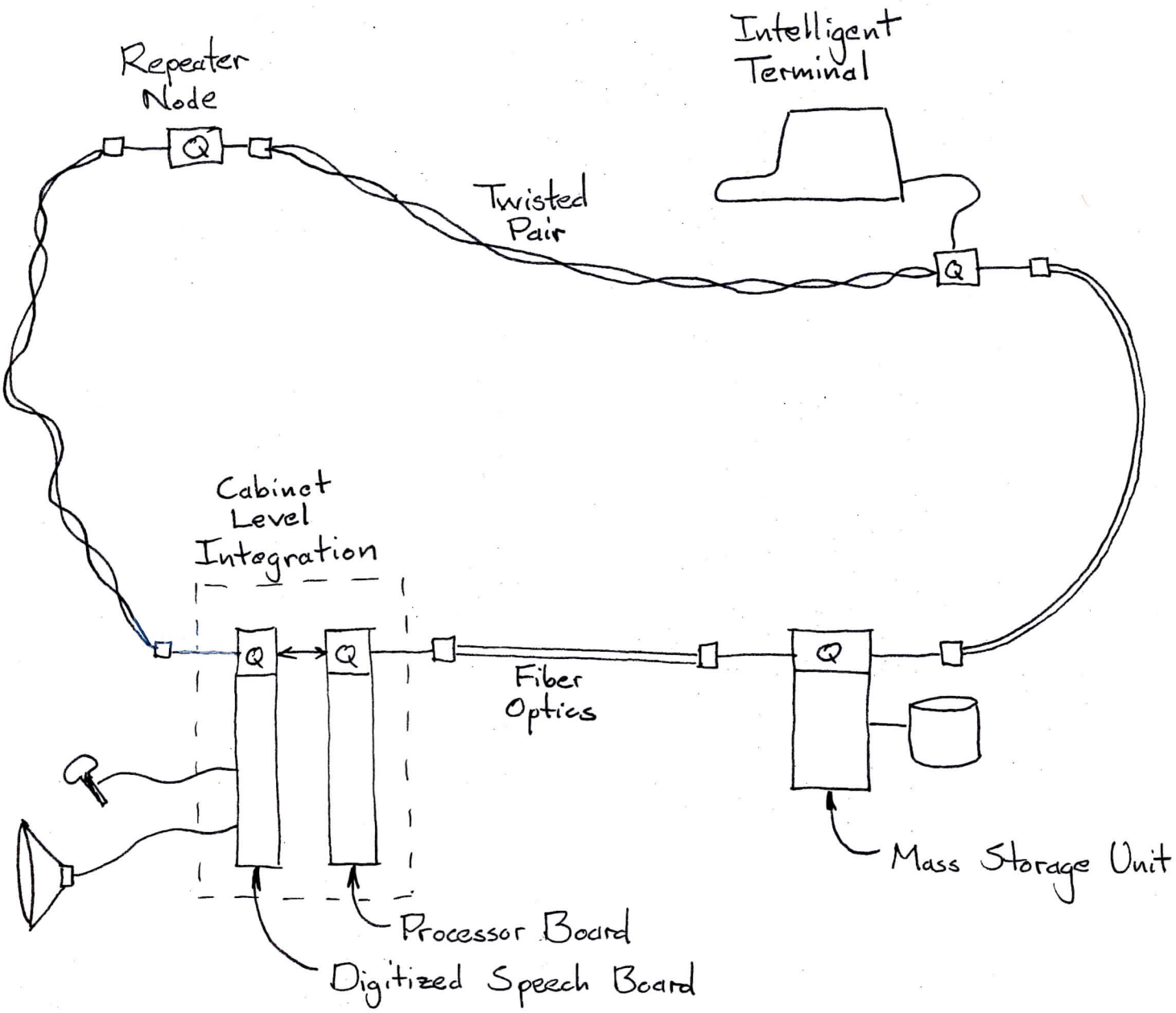


Figure 1-1: A Typical QChip Network

network through packet switching. Packet switching nodes, which might be developed in a later phase of this work, would permit the use of topologies other than the single ring.

The QChip has been designed in a pragmatic manner, to occupy a particular niche in local network technology. The unit of information transfer between QChips is a 20 bit packet containing 8 data bits. Initially QChip networks are restricted to a maximum of 15 nodes, although expansion is possible as mentioned above. The following points summarize our viewpoint and motivation for the design.

1. Even though VLSI technology is easily sufficient to construct a complete network interface on one chip, no general purpose single-package interface is commercially available today.

2. If a single chip interface were available, we could consider board-to-board interconnection via a network, reducing the costs associated with expensive PC board interface circuitry and edge connectors.

3. A major goal has been to control the complexity of the design so that it might reasonably be implemented as part of a VLSI multiproject chip.

4. No essential network functionality has been sacrificed as a result of (3).

The major effect of requirement (3) has been to constrain the amount of on-chip data storage and the number of states in the finite state control. For example, we discounted the possibility of implementing a contention network interface for these reasons; nor has flow control been included within the QChip. The logic design of QChip is tailored for VLSI implementation, based on four Finite State Machines (FSM's) with eight or fewer states per FSM.

The important features of the QChip concept are:

- Asynchronous operation of ring nodes.
- Specifically-addressed and broadcast messages.
- Passive listener mode.
- Parity and 'grand tour' bits for detecting and discarding packets in error.
- Buffering for one packet on input and output at each node.
- Asynchronous host-QChip data transfer protocol.

Asynchronous packet transmission means that loss of framing for

one packet does not affect other packets in the ring. It also simplifies the serial clocking scheme and encourages the design of symmetric nodes. One node is distinguished (by asserting an input line to the QChip) as the Master, and is responsible for the detection and deletion of undeliverable packets circulating in the ring.

If the chip performs at the target clock rate of 4 MHz, which we believe to be a conservative estimate, the time for a single packet (one 8 bit byte) to traverse five QChip-QChip links and be delivered at its destination is approximately 30 microseconds. Due to the inherent pipelining effect in store-and-forward data transmission, the maximum throughput over the same path is 1.4 megabits/second. Although the minimum packet propagation time increases linearly with the path length (i.e., number of intervening nodes), the limiting throughput is roughly constant. QChip may be especially suited to applications such as digitized speech that present data to the network at a constant rate, but are tolerant of propagation delay.

We recognize the disadvantages of the store-and-forward approach. These include the large overhead of address and control bits per packet (12 bits out of 20), the long propagation delay of messages through the ring, and the need for separately adjusted clocks at each station. We feel these factors are outweighed in importance by the simplicity of the host-QChip interface, the simplicity of error detection and correction, the inherent capacity for pipelining multi-packet messages in the ring, the modest complexity of the QChip itself, and the ease of system extension through packet switching techniques.

2. Project Status

The features of a QChip prototype (the Mk I) were specified in August 1980. Since that time the layout of the Mk I part has been completed, and over two dozen functional tests successfully performed using a software simulation package. The simulation tests have given us confidence that the QChip design is logically correct, nonetheless the Mk I design includes extra logic and pinouts to make testing of the circuit possible even if chips from the first fabrication run are not correct in every detail. We believe that the design is ready for fabrication and electrical testing at this time.

3. Packet Format

The QChip packet format is shown in Figure 3-1.

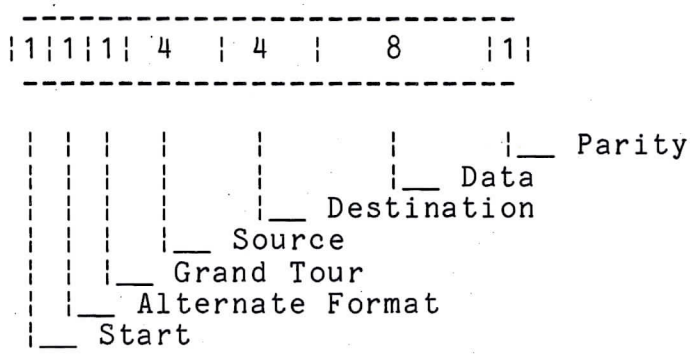


Figure 3-1: Packet Format

The fields of the QChip packet have the meanings:

Start — (1 bit) Always 1; signals beginning of packet during serial transmission.

Alternate Format — (1 bit) When a packet is injected the alternate format bit is obtained from the host. If a QChip receives a packet with the alternate format bit set, it will forward the packet but take no other action.

Grand Tour — (1 bit) The GT bit is used to detect packets with invalid destination addresses that are circulating in the ring. If the Master input signal to this QChip is not asserted, the QChip sets the GT bit of an injected packet to 0, and does not inspect or change the GT bit of a forwarded packet. If the Master input is asserted, the GT bit of injected or forwarded packets is set to 1; a packet arriving at this node with the GT bit set to 1 is discarded. (Note: if an arriving packet was injected by this node, it is delivered before it is discarded.)

Source — (4 bits) Values 1 through 15 decimal represent specific source node addresses; the value 0 is unused.

Destination — (4 bits) Values 1 through 15 decimal represent specific destination addresses, and 0 implies a broadcast to all nodes.

Data — (8 bits) An arbitrary data byte not inspected or changed by the QChip.

Parity — (1 bit) A parity bit is generated whenever a packet is transmitted and checked when it arrives. The parity bit is generated to produce

an even number of 1's in the packet. An arriving packet with incorrect parity is discarded before any further processing.

4. Principles of Operation

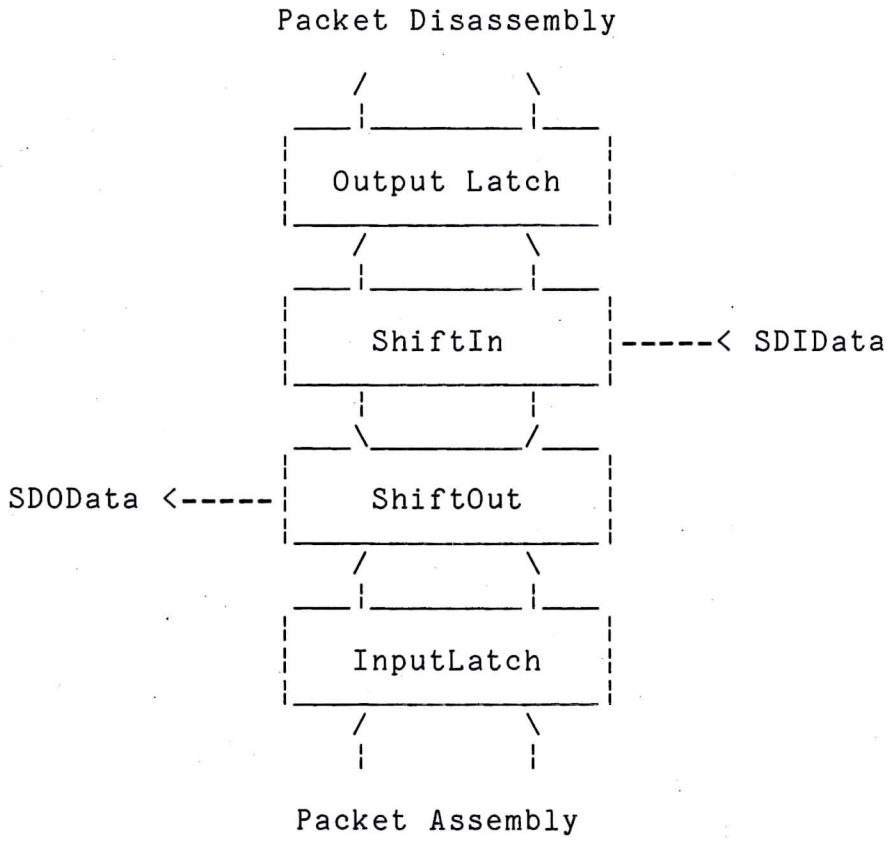


Figure 4-1: Internal Block Diagram

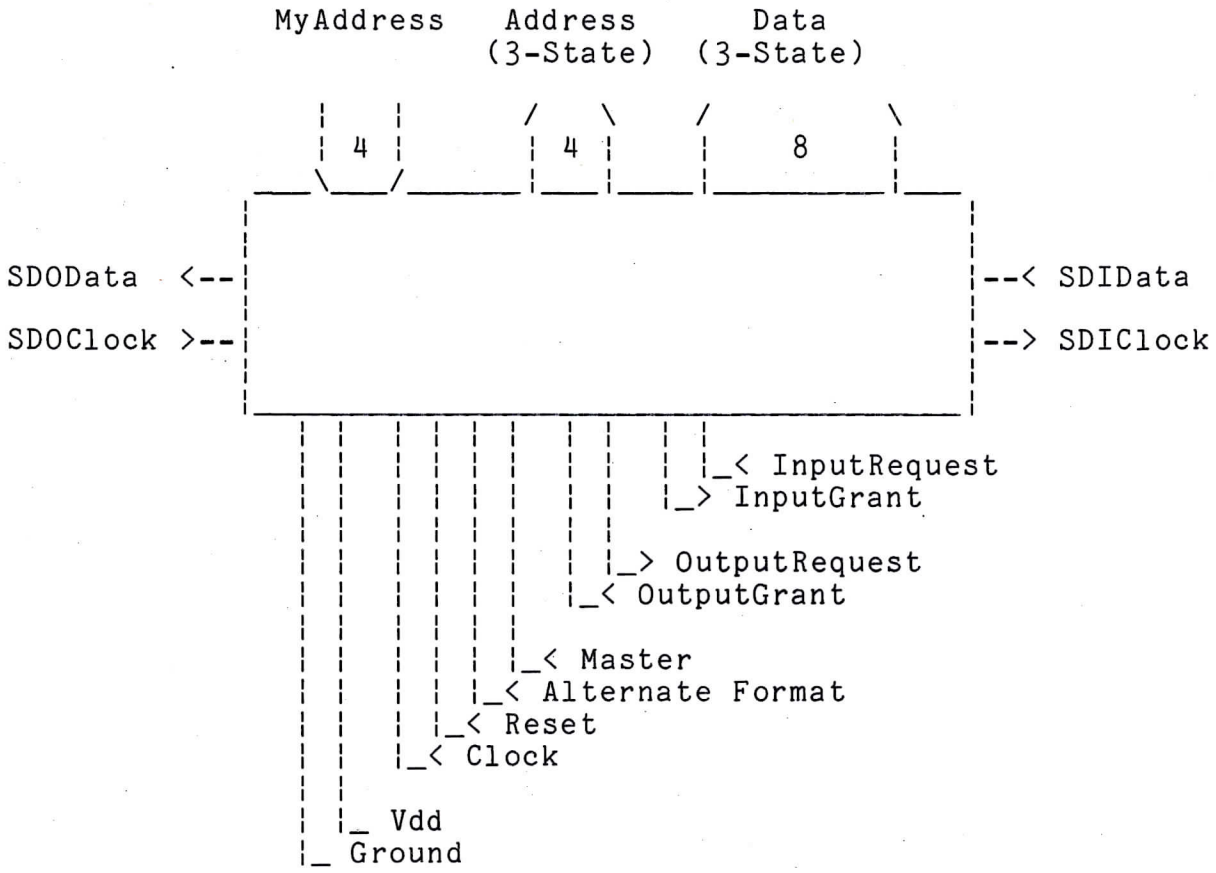


Figure 4-2: QChip Pinouts

4.1. Store-and-forward Packet Transmission

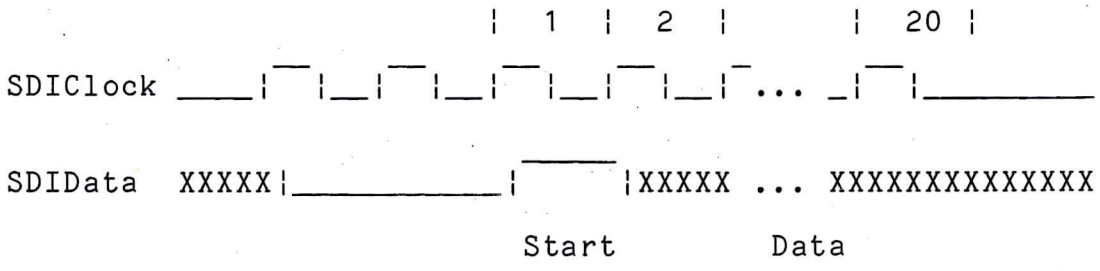


Figure 4-3: Store-and-forward Timing

As long as this QChip is not prepared to receive a packet from its predecessor, output SDIClock is held low. When it is ready to receive a packet the clock signal (from pin Clock) is

gated to SDIClock. During the second half of the clock cycle (SDIClock low) the QChip latches SDIData. The detection of a 1 signals the start bit of a packet; the packet is clocked into ShiftIn, beginning on this clock with the start bit, and continuing through 19 additional clock periods. After twenty bits have been received the clock is removed from SDIClock.

Parity is computed within the QChip as the packet arrives. After the twenty bits of a packet have been received the parity is checked; if it is odd the packet is immediately discarded. A packet with even parity is checked for delivery and/or forwarding. It will be forwarded if it is not a master, or if it is a master and the GT bit is not set, and any of the following hold:

1. the alternate format bit is set;
2. it is specifically addressed to a different node;
3. it is a broadcast not originating at this node;
4. this node is a passive listener.

If the packet is to be forwarded it is transferred in parallel from ShiftIn to ShiftOut as soon as ShiftOut is empty. At this point the clock is again gated onto SDIClock (unless there is an injection pending, see the next section).

The packet remains in ShiftOut until a clock is detected on SDOClock which causes it to be transmitted to the next QChip node.

4.2. Packet Injection

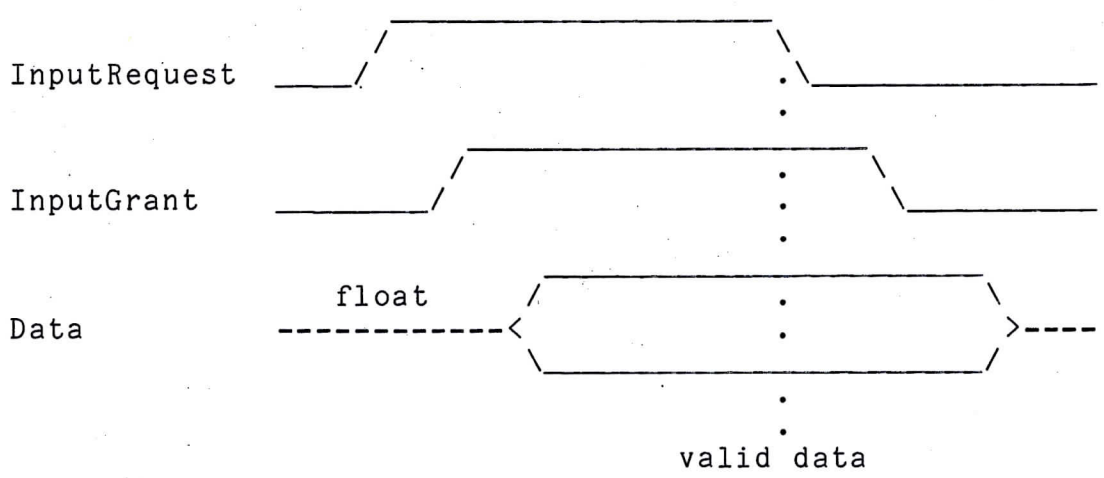


Figure 4-4: Packet Injection Timing

The injection of packets into the network is done in two steps: first, the packet is transferred from the host into the InputLatch by means of a protocol conducted on the lines InputRequest and InputGrant; second, at an appropriate time the packet is transferred from the InputLatch to the ShiftOut register, from which it proceeds to the next node in the normal manner.

In a quiescent state both InputRequest and InputGrant are low. The host asserts InputRequest when it wishes to inject a packet; when the InputLatch is empty (the preceding packet has been transferred out) and no packet is waiting for delivery to this node, the QChip asserts InputGrant in response. The host then places the input data on the 3-State bus and after the data has settled deasserts InputRequest. The falling edge of InputRequest signals the QChip that valid data is present; the data is stored in InputLatch and InputGrant is deasserted. The falling edge of InputGrant permits the host to remove the input data from the 3-State bus.

The transfer of the InputLatch contents to ShiftOut is such as to prevent the possibility of deadlock due to overpopulation of the ring. InputLatch will only be transferred to ShiftOut if both ShiftIn and ShiftOut are empty. In an attempt to reach this state, the QChip may block further input into ShiftIn (by withholding the clock from SDIClock) <u>if</u> ShiftIn is empty and ShiftOut has begun the transfer of its contents to the next node. In this case, the injection will complete as soon as the packet in ShiftOut has fully departed. Injection cannot complete until both ShiftIn and ShiftOut are empty, and packets cannot be blocked from ShiftIn except in the case above.

These tests are sufficient to prevent deadlock in a ring network; they do not prevent permanent blocking. The hosts can cooperate to limit their individual transmission rates and thereby prevent permanent blocking.

4.3. Packet Delivery

An attempt will be made to deliver an arriving packet to the host if:

1. this QChip is a passive listener (MyAddress=0);
2. the packet is a broadcast message (Destination=0);
3. the message is addressed to this QChip (Destination=MyAddress).

On the clock following the arrival of the packet, the state of OutputLatch is tested. If OutputLatch is empty and the packet should be delivered, it is copied from ShiftIn to OutputLatch and the chip will assert OutputRequest to the host as soon as possible (after any host-to-QChip transfer in progress is completed). If the OutputLatch is full, and the packet is not to be forwarded, it is immediately discarded.

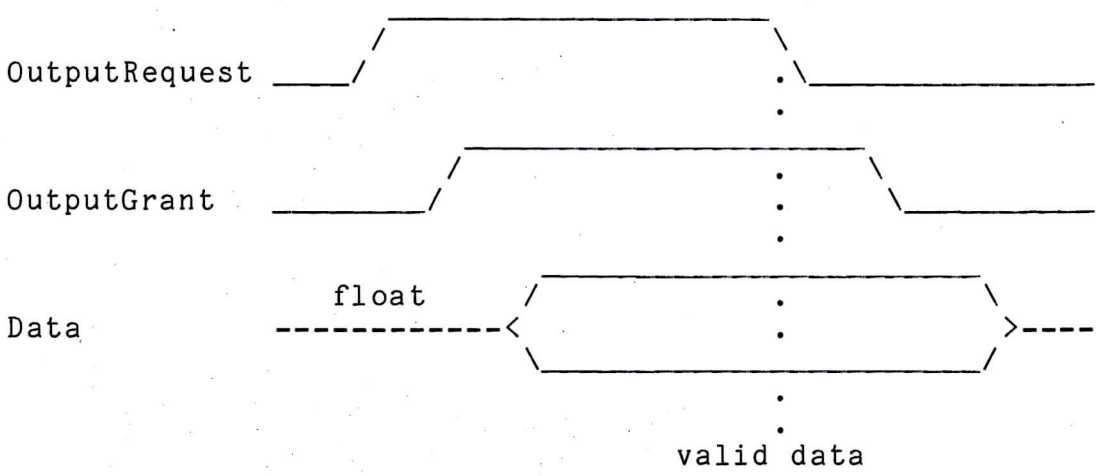


Figure 4-5: Packet Delivery Timing

When the QChip asserts OutputRequest it will not assert InputGrant until the host completes the output transfer. The host raises OutputGrant when it is prepared to accept the arriving packet. In response the QChip places the output data on the 3-State bus and then deasserts OutputRequest. The falling edge of OutputRequest signals the host that valid data is present on the bus; the host latches the data and lowers OutputGrant. After the falling edge of OutputGrant the QChip returns the bus to its high-impedance state.

Note that delivery has precedence over injection, but since deliveries cannot occur more frequently than one packet shift time (>20 clocks) the host should be able to interleave injections and deliveries without difficulty.

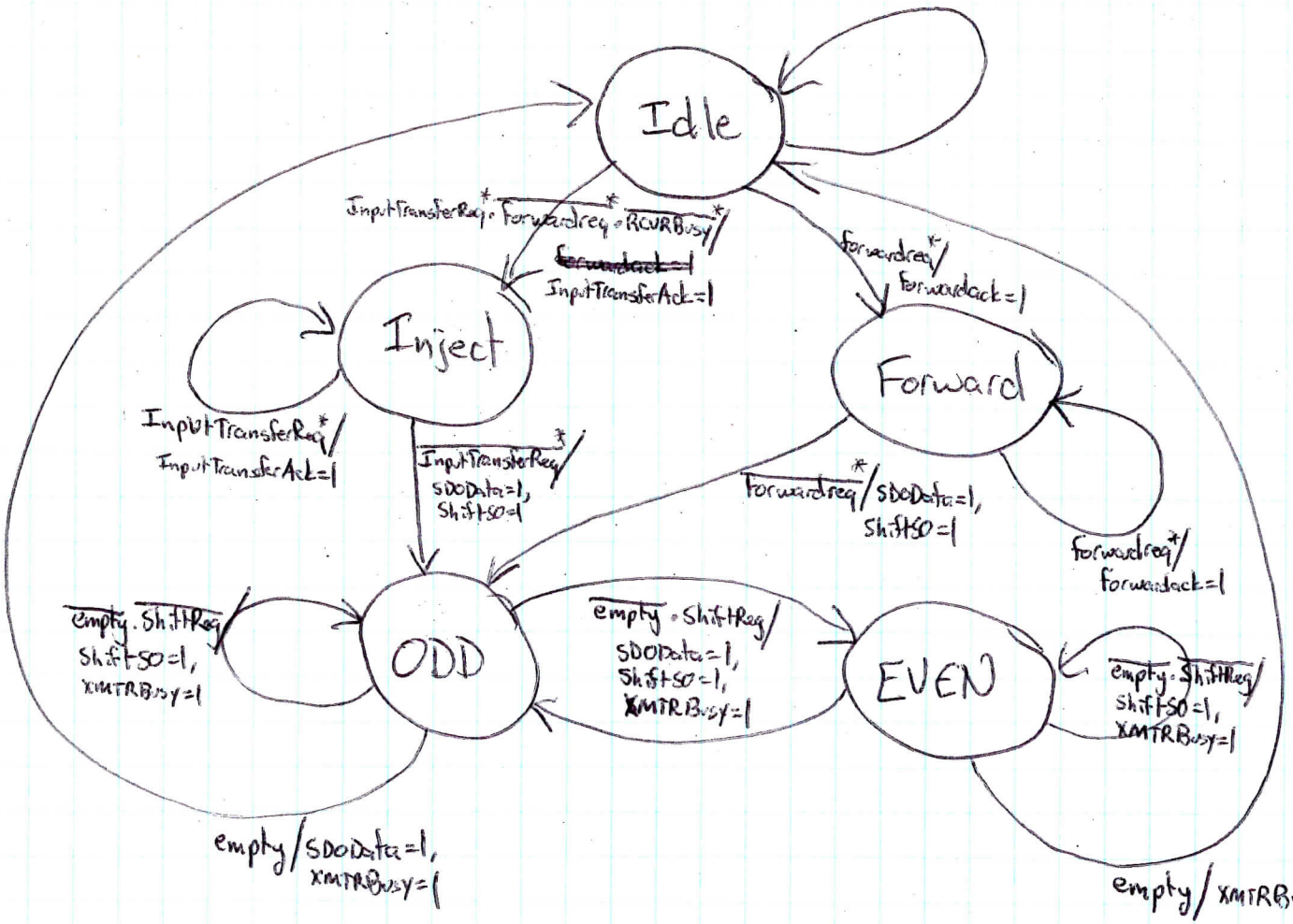
4.4. Notes

1. The reset line clears all internal registers and restores the QChip to a quiescent state.

2. Passive listeners are not permitted to transmit messages. A node may, however, change its status from passive to active or vice versa by altering MyAddress.

3. If a packet with the Grand Tour bit set is received by a QChip with the Master signal asserted, and the packet is addressed to this QChip either specifically or as a broadcast, the packet will be delivered but not forwarded.

TRANSMITTER STATE DIAGRAM



* Asynchronous input

INPUTS:

- empty- From ShiftOut
- ShiftReg- From ShiftOut
- InputTransferReq*- From input machine
- forwardreq*- From RCVR
- RCVRBusy*- From RCVR

OUTPUTS:

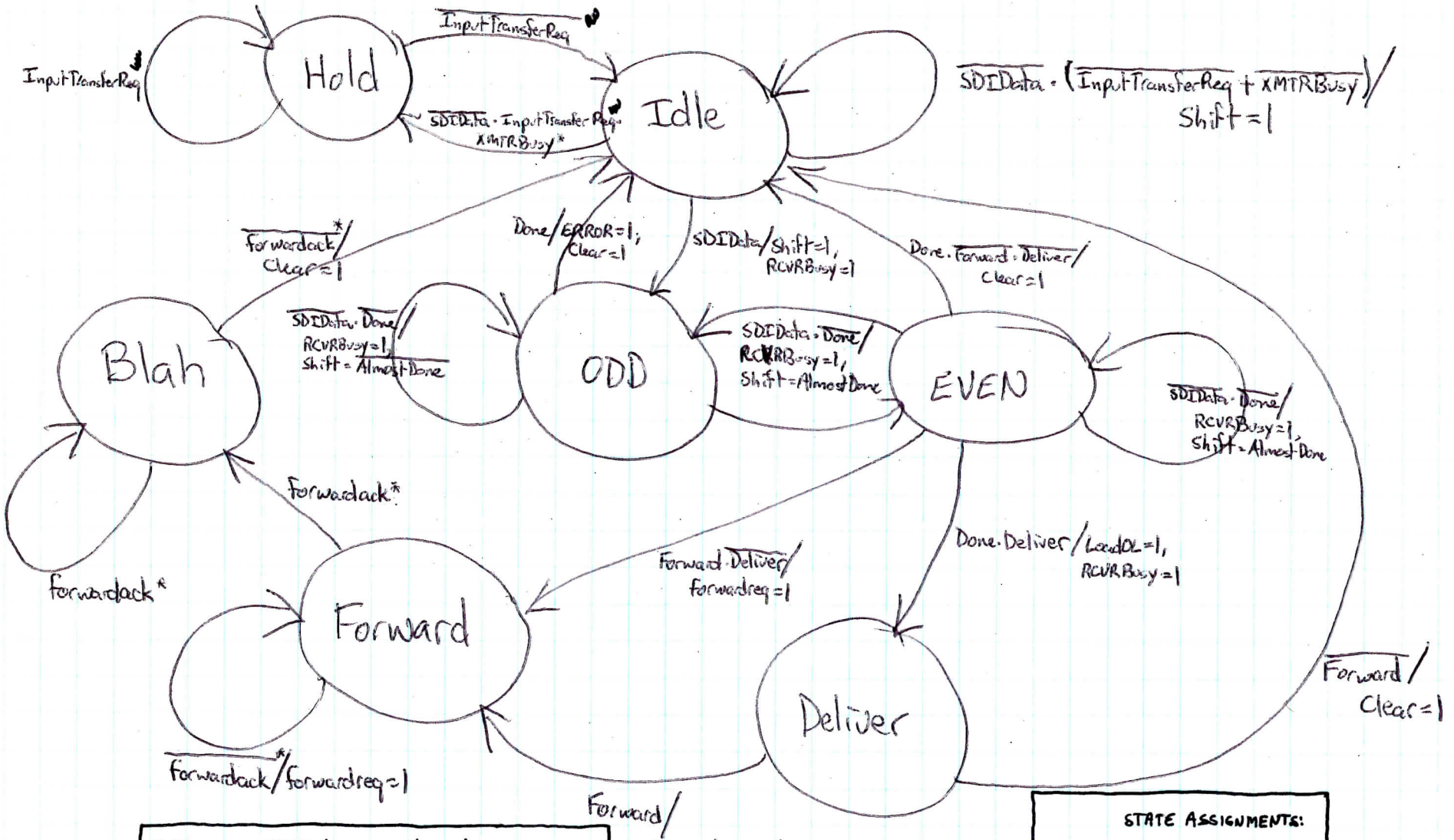
- SDOData- External
- forwardack- To RCVR
- InputTransferAck- To input machine
- ShiftSO- To ShiftOut
- XMTRBusy- To RCVR

STATE ASSIGNMENTS:

- 000- Idle
- 001- Forward
- 100- Inject
- 101- ODD
- 111- EVEN

24-APR-81

RECEIVER STATE DIAGRAM



INPUTS:
- SDIData – external
- InputTransferReq – From input machine
- XMTRBusy* – From XMTR
- AlmostDone – From ShiftIn
- Done – From ShiftIn
- forwardack* – From XMTR
- Forward – } RandomLogic outputs
- Deliver – }

OUTPUTS:
- Shift – To ShiftIn
- Clear – To ShiftIn
- RCVRBusy – To XMTR
- LoadOL – To OutputLatch, Output machine
- forwardreq – To XMTR
- ERROR – External

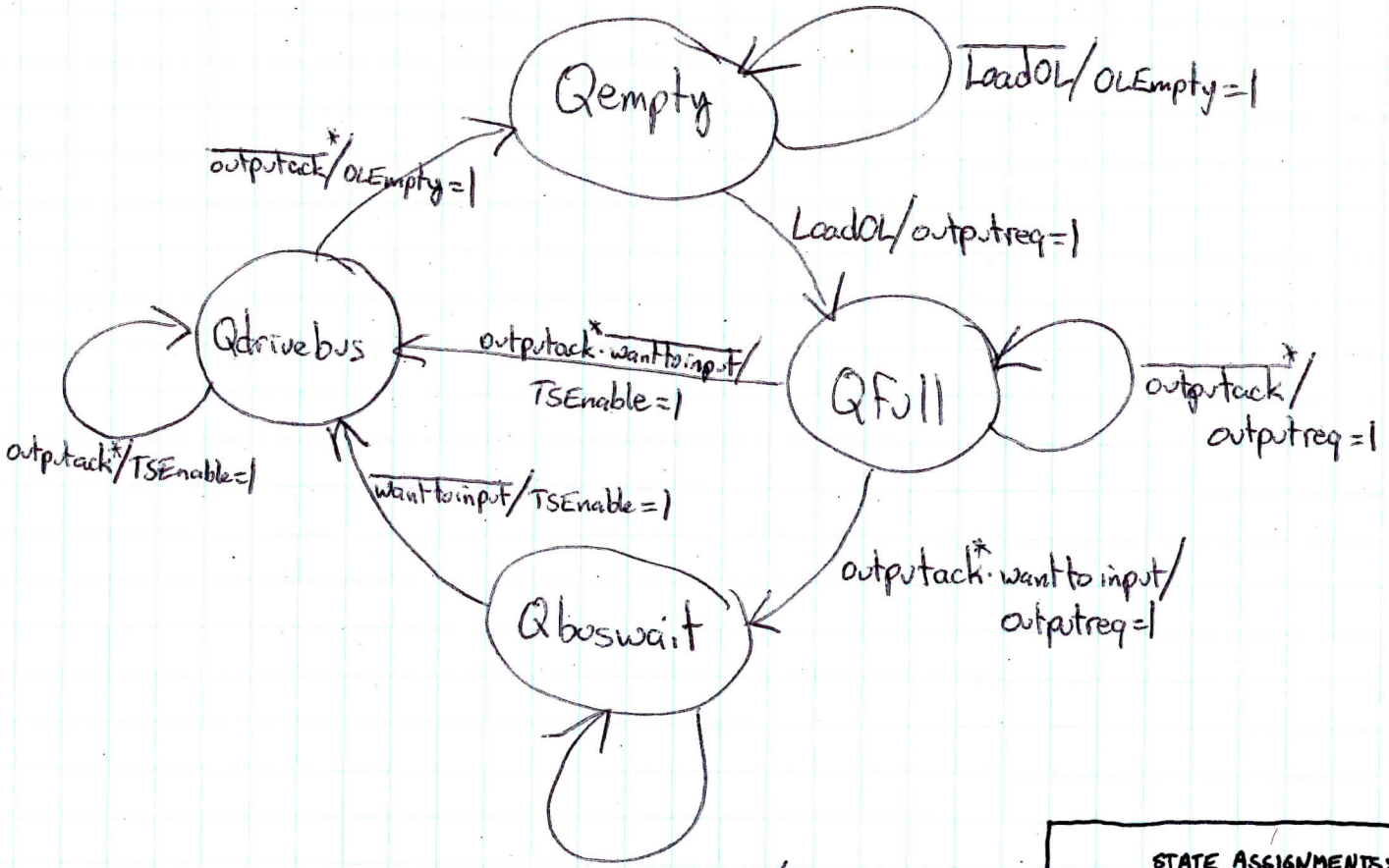
STATE ASSIGNMENTS:
- 000 – Idle
- 001 – Hold
- 010 – ODD
- 011 – EVEN
- 100 – Blah
- 101 – Forward
- 111 – Deliver

* Asynchronous input

24-APR-81

OUTPUT MACHINE STATE DIAGRAM

Synchronous with Receiver, Input machine



INPUTS:

- outputack* - External
- wanttoinput - From input machine
- LoadOL - From RCVR

OUTPUTS:

- outputreq - External
- TSEnable - To input machines tri-state pads
- OLEmpty - To Random Logic

STATE ASSIGNMENTS:

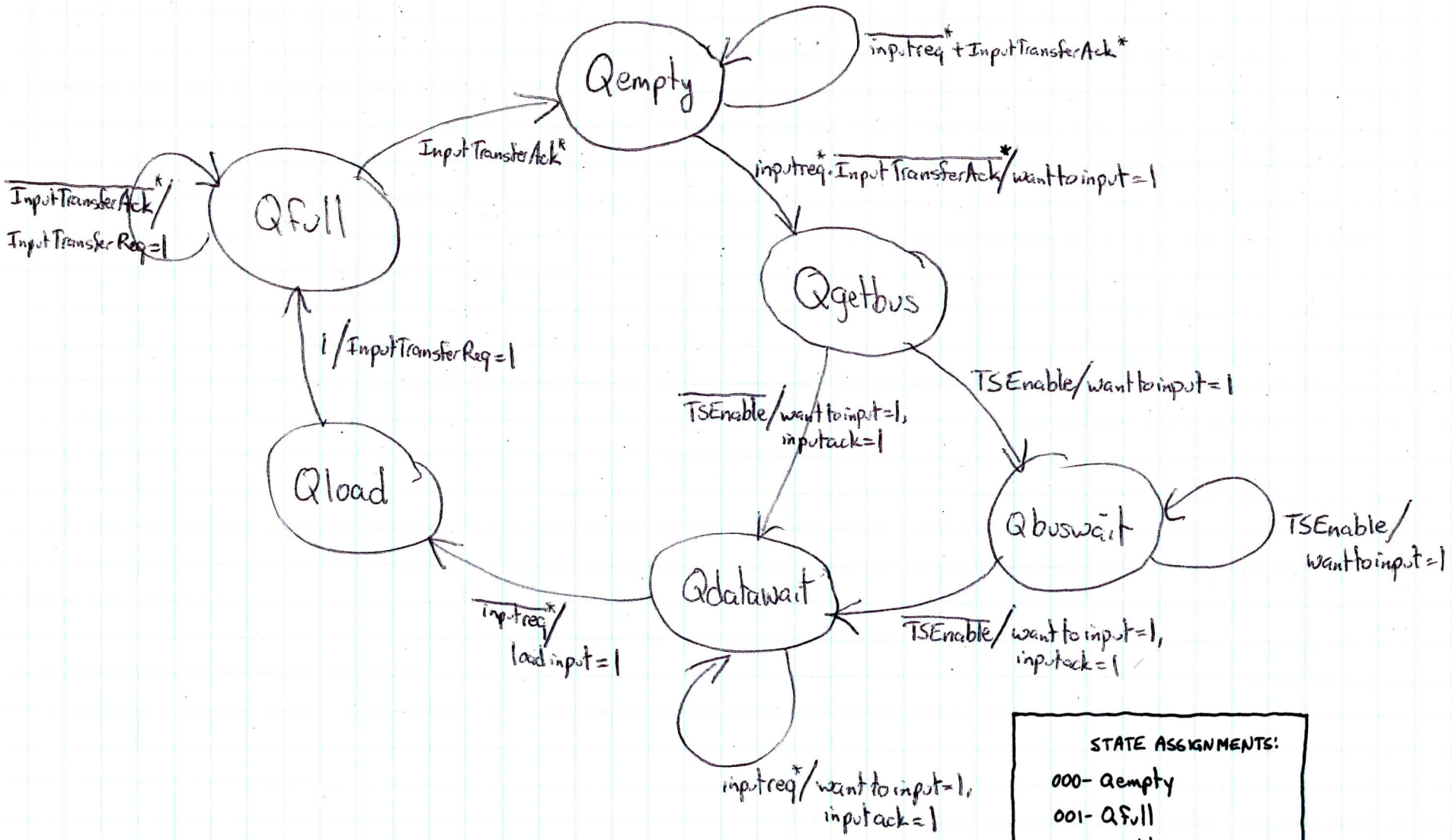
00 - Qempty
01 - Qdrivebus
10 - Qbuswait
11 - QFull

*Asynchronous input

24-APR-81

INPUT MACHINE STATE DIAGRAM

Synchronous with Receiver, Output Machine



STATE ASSIGNMENTS:

- 000 - Qempty
- 001 - Qfull
- 010 - Qgetbus
- 011 - Qload
- 110 - Qbuswait
- 111 - Qdatawait

INPUTS:

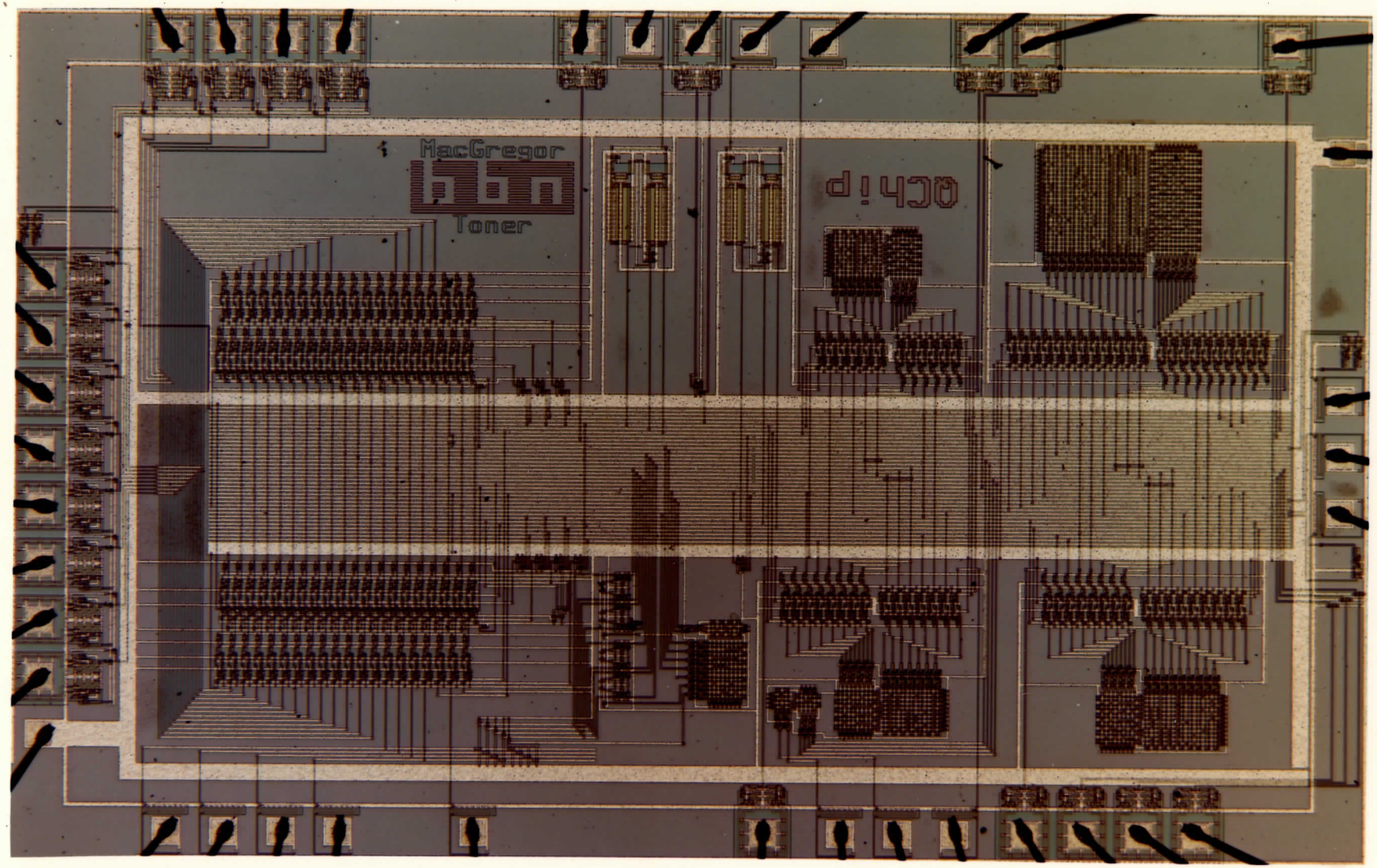
- inputreq* - External
- InputTransferAck* - From XMTR
- TSEnable - From output machine

OUTPUTS:

- wanttoinput - To output machine
- inputack - External
- loadinput - To ~~ShiftOut~~ InputLatch
- InputTransferReq - To XMTR

* Asynchronous input

24-APR-81


MacGregor
Toner

1/7/82 - chips received (yay!) Pinouts:

1	DATA2	40	DATA3
2	DATA1	39	DATA4
3	DATA0 (MSB)	38	DATA5
4	GND	37	DATA6
5	MyAddr3 (LSB)	36	DATA7 (LSB)
6	MyAddr2	35	Addr3 (LSB)
7	MyAddr1	34	Addr2
8	MyAddr0 (MSB)	33	Addr1
9	Alt Fmt	32	Addr0 (MSB)
10	—	31	Done
11	~~OutputRequest~~ InputGrant	30	Clock
12	FSMSelect0 (MSB)	29	SDIClock
13	FSMSelect1 (LSB)	28	SDOClock
14	~~OutputGrant~~ InputRequest	27	~~InputRequest~~ OutputGrant
15	SDOData	26	~~InputGrant~~ OutputRequest
16	FSMState2 (LSB)	25	SOEmpty
17	FSMState1	24	ERROR
18	FSMState0 (MSB)	23	VDD
19	—	22	Master
20	$\overline{\text{Reset}}$	21	SDIData

FSMselect0	FSMselect1	State Machine
0	0	~~Xmit~~ Transmitter
0	1	Input Machine
1	0	Receiver
1	1	Output Machine

?

XX3465

Received 7 chips. Of these,

0 had working receivers

1 drew 145 ma & had outputs stuck @ 1 (suspect Fab (#5) or bonding error) All others drew ≈ 70 ma

1 had receiver state stuck at 0 ~~drew ≈ 70 ma~~ (transmitter (#2) almost worked, though - didn't seem to load p.b.

2 had working input machines, but (#1, #3) couldn't do handshake with transmitter

2 Sent only partial packets (#6, #6)

1 Sent Full packets @ 1MHz (#4)

→ Receiver problems

→ Transmitter problems

→ FSMs - what worked (all seemed to work as far as they could be tested ... ODD → Idle state worked in RCVR)

→ SRs - seemed to propogate bits random distance down reg

- Working guy @ 2MHz had 1-gobbling problem

→ On working chip, $V_{dd_{min}} \approx 3.10\,V$ max untested! 1 output ≈ 3V ($V_{dd} = 5V$)

$f_{max} \approx 1MHz$ $f_{min} = ?$

→ Suspected problem (with diagrams) Why simulator didn't catch it

1895
2
1790

~~6~~ ~~10~~

12

~~6~~ chips · 3 FSMs = ~~18~~ 12 total

2 can't

~~18~~ 6