Report Number: WUCS-88-23

1988-10-01

Design of an 8-BIT VLSI Packet Switch Element

Einir Valdimarsson

This paper describes the design of the Packet Switch Element Chip, one of the components of a high speed broadcast packet switching network. The chip is a 2x2 switch element, from which a switch fabric of arbitrary size can be constructed. It is currently being implemented in 2um CMOS technology.

DESIGN OF AN 8-BIT VLSI PACKET SWITCH ELEMENT

Einir Valdimarsson

WUCS-88-23

October 1988

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

Abstract

This paper describes the design of the Packet Switch Element Chip, one of the components of a high speed broadcast packet switching network. The chip is a 2×2 switch element, from which a switch fabric of arbitrary size can be constructed. This is a second version of the chip and it implements 8-bit data lines. It is currently being implemented in 2 μm CMOS technology.

This work supported by Bell Communications Research, Bell Northern Research, Italtel SIT, NEC and National Science Foundation grant DCI-8600947.

Contents

1 Introduction 3

1.1 Packet Switch 3

2 Chip Specification 6

2.1 Operation 6

2.2 PSE Interface 7

3 Design of Major Functional Blocks 11

3.1 Input Shift Register 11

3.2 Input Control Circuit 12

3.3 Buffer Shift Register 14

3.4 Header Modification Circuit 15

3.5 Timing Control Circuit 17

3.6 Output Control Circuit 18

4 Implementation 21

4.1 Integrated Circuit Design 21

4.2 Circuit Design Verification 21

4.3 Packaging and Pin Asignment 21

4.4 Testability features 21

List of Figures

1 Switch Module 3
2 Packet Format 4
3 Packet Switch Element Interface 7
4 Packet Switch Element Block Diagram 9
5 Input Circuit 10
6 Input Shift Register 11
7 Input Control Circuit 12
8 Header Register and Decoder 13
9 Buffer Shift Register 14
10 Header Modification circuit 15
11 Timing Control Circuit 17
12 Timing Circuit Timing 18
13 Output Control Circuit 19
14 Pin Assignment of the PSE Chip 22
15 Test circuitry in Input Control Circuit 23
16 Test Circuitry in Header Modification Circuit 25
17 Chip block diagram 39

DESIGN OF AN 8-BIT VLSI PACKET SWITCH ELEMENT

Einir Valdimarsson
ev@wuccrc.wustl.edu

1. Introduction

This paper describes the design of the Packet Switch Element Chip (PSE) for use within a broadcast packet switching network. The broadcast packet switching network is described in [Tu85] and [Tu86]. This is a second version of the chip implementing 8-bit data lines. The main difference from the first version [St88a] is in grant propagation and number of buffers.

1.1. Packet Switch

The overall structure of the prototype packet switch is shown in Figure 1. The switch module terminates 7 fiber optic links, and consists of five major components: the connection processor (CP), packet processors (PPs), copy network (CN), broadcast translation circuits (BTCs), and routing network (RN). The CN, BTCs, and RN are collectively known as the Switch Fabric (SF), and are constructed from banyan-interconnected PSEs. The PPs serve as buffers between the transmission

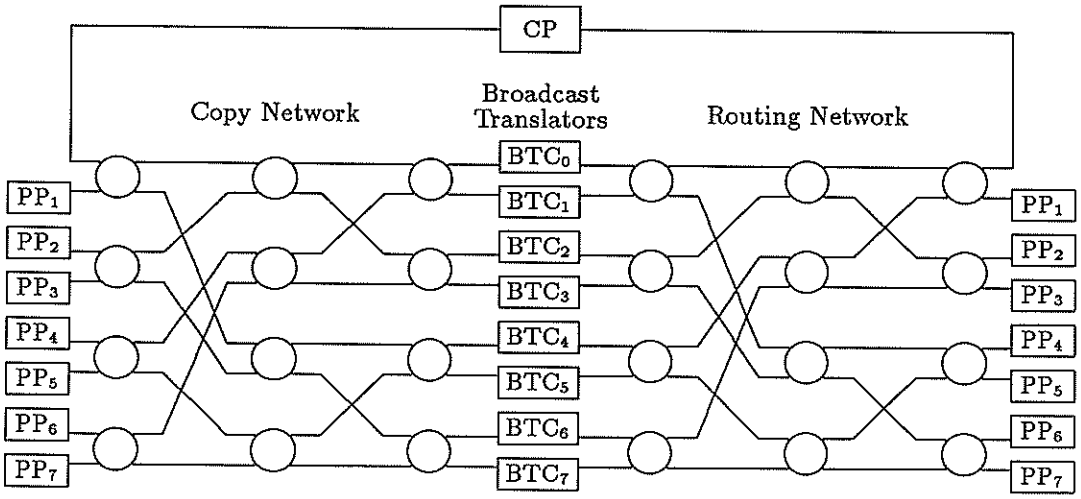

Figure 1: Switch Module

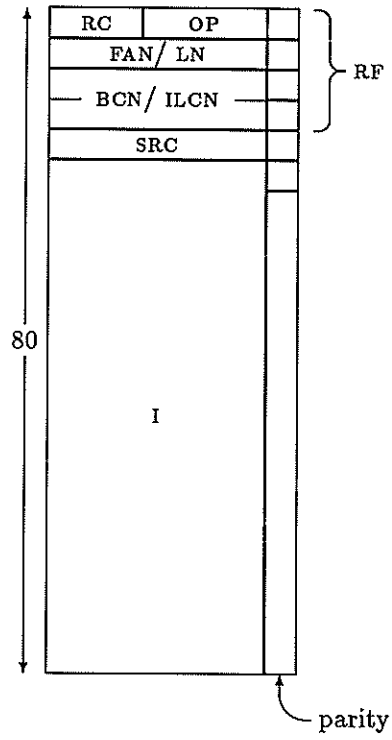

Figure 2: Packet Format

links and also perform the address translation required by routing. The RN routes the packets to the appropriate output link based on information encoded in the routing field (RF) of the packet. Since a banyan interconnection is self-routing, the PSEs need only examine a single address bit to select the proper output port. The CN replicates broadcast packets to multiple output links based on information in the RF, indicating the fanout of the packet. Point-to-point packets are arbitrarily routed through the CN. The distribution network (DN), not present in the prototype, will randomly distribute packets before the RN to avoid congestion occurring from persistent poor traffic patterns (resulting, for example, from communities of interest).

The format of the packets processed by the PSE is shown in Figure 2. The packet is organized as a sequence of 8-bit words and an odd parity bit. Each packet contains exactly 80 words, the first five of which constitute the packet *header*. Only the first four words, the routing field (RF), are used by the PSE. The meaning of the fields are:

- *Routing Control* (RC). These three bits determine how the packet is processed by the switch elements.
  - 000 signifies an empty packet slot,
  - 001 signifies a normal point-to-point data packet,
  - 010 signifies a normal broadcast packet,
  - 100 signifies a test packet.

- *Operation* (OP). This field specifies which of several control operations is to be performed for this packet. Not used by the PSE.
- *Fanout* (FAN). If RC = 010, the second word of the packet is taken to be the fanout, that is the number of switch fabric output ports that require copies of the packet.
- *Link Number* (LN). If RC = 001, the second word of the packet is taken to be the number of outgoing link to which the packet should be delivered.
- *Broadcast Channel Number* (BCN). If RC = 010, the third and fourth words of the packet are taken to be the broadcast channel number. All packets within a particular multi-point channel have the same broadcast channel number. Only 256 distinct BCN are recognized.
- *Internal Logical Channel Number* (ILCN). If RC = 001, the third and fourth words of the packet are taken to be the internal logical channel number. This will become the external logical channel number when the packet exits the switch module.
- *Specific Path Specification.* If RC = 100, the next three words specify output ports for each of the three networks. The packet will routed through each of these.
- *Source* (SRC). This word identifies which switch fabric input port the packet came from, and is not used by the PSE.
- *Information* (I). This field normally contains user information. In the case of control packets, additional control information may be placed here.

2. Chip Specification

2.1. Operation

A single PSE circuit is used within the routing, copy, and distribution networks that comprise the switch fabric. PSE routing decisions are based on the operation mode (OM) and packet type (RC in the packet header). The operation modes are:

- *Route* — Route all packets based on the appropriate bit of the LN field of the packet header.
- *Copy* — Replicate packets based on the FAN field of the packet header.
- *Distribute* — Distribute packets arbitrarily and uniformly between outputs.

The packet types are:

- *Point-to-point* — Use LN to route packets only for *route* operation mode.
- *Broadcast* — Use FAN to replicate packets only for *copy* operation mode.
- *Specific Path* — Use LN to route packets in all operation modes.

When arbitrary routing decisions can be made, the following policies are followed:

- Ties among input ports for the same output port are broken such that the last input port favored loses, to avoid individual starvation.
- Packets that can be routed to either output are uniformly distributed between output ports based on an output toggle. This includes all packets in the distribution network, and point-to-point and unreplicated broadcast packets in the copy network.
- Packets requiring both output ports in the copy network are favored over packets needing only one output.
- Packets requiring a specific output port are favored over packets requiring either output.

All packets are routed in the RN based on the LN. The PSEs are interconnected to form a banyan network, which is self-routing. The routing algorithm, is simply to route to output port 0 or 1, depending on whether the LN bit corresponding to the switch element stage number (SN) is a 0 or 1.

Broadcast packets are replicated in the CN such that the number of packets leaving the CN is equal to the original FAN of the packet. Each PSE makes a decision to copy if the FAN is greater than half the number of reachable outputs, specifically if FAN $> 2^{SN}$. The FAN of each replicated packet is adjusted to reflect the number of copies that must be subsequently made in the CN. This copy algorithm delays packet replication as long as possible to reduce congestion in the CN. Point-to-point packets, and broadcast packets not requiring replication in a particular PSE, are routed based on the distribution algorithm described above.

Packets in the distribution network are randomly distributed among the outputs of each PSE. This is accomplished by toggling a favored output bit for each input port, and thus distributing packets evenly across the outputs of the switch element.

Each PSE operates on the basis of a *packet cycle*, indicating the timing of packet arrival. A packet cycle is initiated by *packet-time* going high, indicating that data is present on the *upstream data*

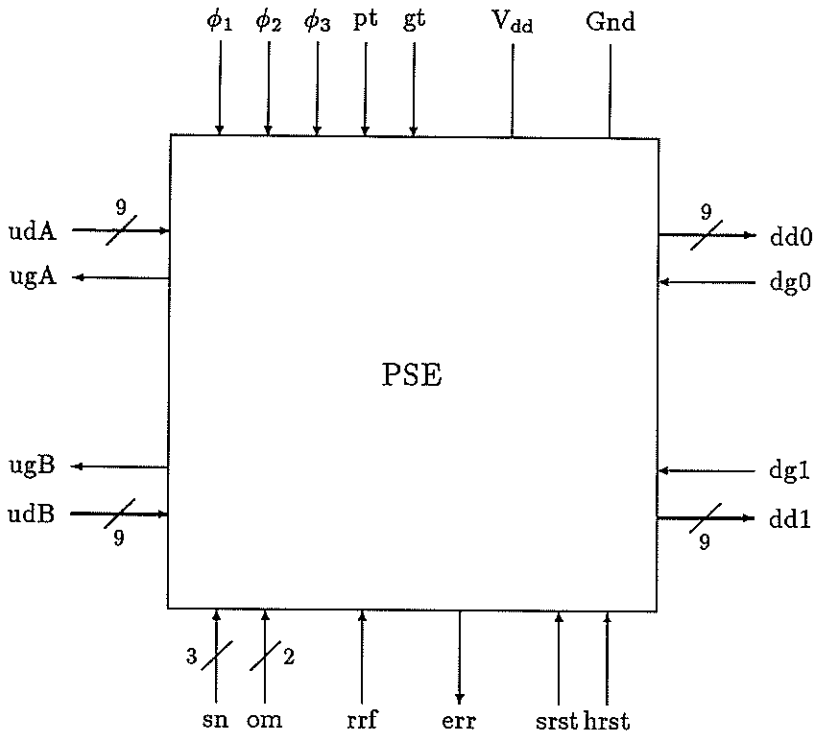

Figure 3: Packet Switch Element Interface

lines at the input to the PSE. The Packets are either buffered, or if possible cut-through directly to the appropriate output port. Each input is capable of fully buffering two 80 word (8 bits + parity) packets. If a packet is cut through, it appears on the *downstream data* lines at the appropriate output of the PSE twenty four clock cycles later. Successive words of the packet are received/transmitted every clock cycle.

Grants are propagated backward from a node to its upstream neighbour, indicating that switch element stage will be able to accept packets (due to an empty buffer slot).

2.2. PSE Interface

The external interface signals of the PSE are shown in Figure 3 and are described briefly below. Signals incorporating multiple lines are represented collectively by square brackets, e.g. `x[210]` refers to `x2,x1,x0`, `x[ab][10]` refers to `xa1,xa0,xb0,xb1`, and `x[10][10]'` refers to `x1,x0,`$\overline{\texttt{x1}}$`,`$\overline{\texttt{x0}}$.

- *Upstream data leads* (ud[876543210][AB]). Incoming data from upstream switch elements; 8-bits data plus odd parity for each input side of switch (a,b). Parity is upstream data lead ud0[AB.
- *Downstream data leads* (dd[876543210][10]). Outgoing data to downstream switch elements; 8-bits data plus odd parity for each output port of switch (0,1).
- *Upstream grants* (ug[AB]). Grant signals to upstream neighbors, granting permission to transmit a packet during the next packet cycle.
- *Downstream grants* (dg[10]). Grant signals from downstream neighbors, indicating permission has been given to transmit a packet on the corresponding output port during the next packet cycle.
- *Packet-time* (pt). Goes high when first word of a packet is present on the ud leads.
- *Operation mode* (om[10]). Switch element operation mode: 01=RN, 10=DN, 11=CN.
- *Stage number* (sn[210]). Indicates the switch network stage of which the PSE is a part. The furthest downstream column of switch elements is stage 0.
- *Rotate routing field* (rrf). Causes the PSE to rotate the second through fourth words of test packets.
- *Error flag* (err). Report errors, including parity errors and packet header validity errors.
- *Hard reset* (hrst). Initialize internal control registers and reset err.
- *Soft reset* (srst). Reset error flag err.
- *Clock* (ϕ_1, ϕ_2, ϕ_3). Two-phase, symmetric, non-overlapping clock (ϕ_1, ϕ_2) and third phase (ϕ_3) overlapping ϕ_1 used in timing control circuit.
- *Power and Ground* (Vdd,Gnd).

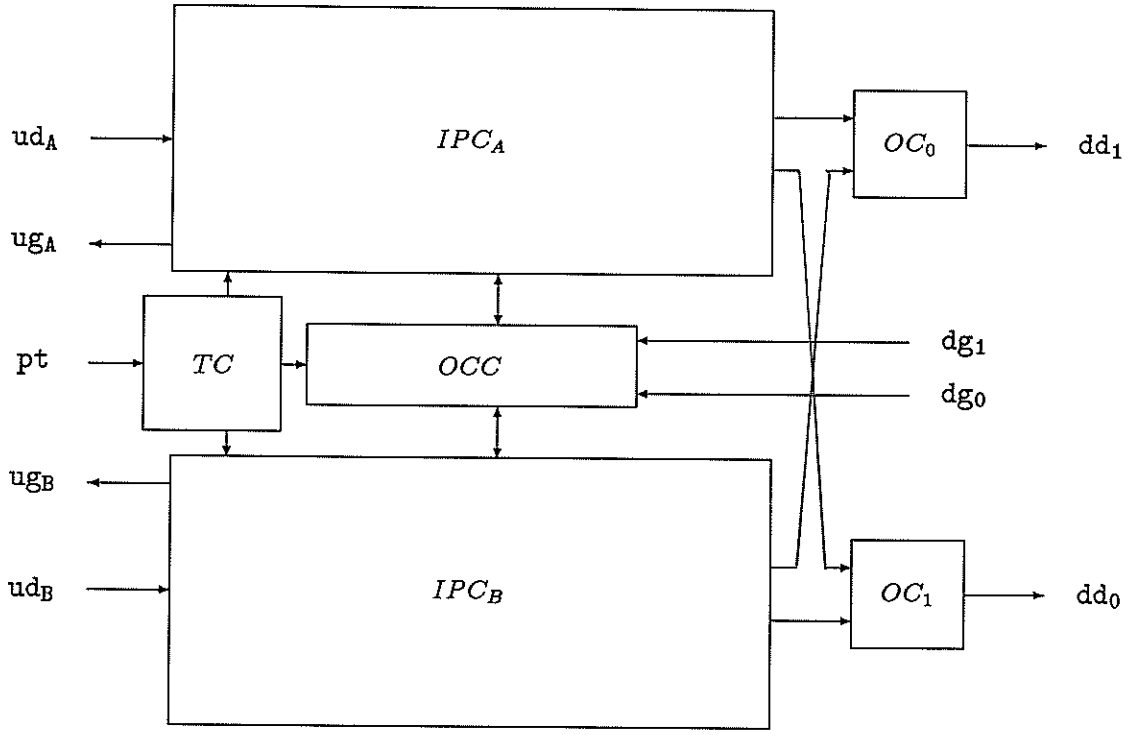

Figure 4: Packet Switch Element Block Diagram

The PSE is a buffered 2-input, 2-output switch with broadcast capability. The main data path is symmetric with respect to each side of the switch element (a,b) and each output (0,1). Control and arbitration logic is shared across both halves of the switch. A block diagram is shown in Figure 4.

The major components of the switch element are:

- *Output Control Circuit* (OCC) — The OCC arbitrates access to the two output ports, based on the downstream grant signals and port requests received from the input port circuits. The port requests are given in the form of three bit request vectors, $\mathtt{r[N01]_A}$ and $\mathtt{r[N01]_B}$. The response is given in the form of two bit enable vectors $\mathtt{en[01]_A}$ and $\mathtt{en[01]_B}$. It also keeps track of last favored input and output.
- *Timing Circuit* (TC) — This circuit generates signals of the form t_i and $t_{i:j}$, for various values of i,j. Signal t_i is high during the ith clock period. Signal $t_{i:j}$ is high during t_i and stays high through t_j. These signals are used as basic control signals for the rest of the PSE.
- *Input Circuit* (ICP_A, ICP_B) — There is one input circuit for each input port. Each IPC includes two buffers large enough to hold a single packet, plus control circuitry to extract information from the packet header, generate the request vector for the OCC and use the resulting enable vector to make decisions on the disposition of the packets. It also modifies

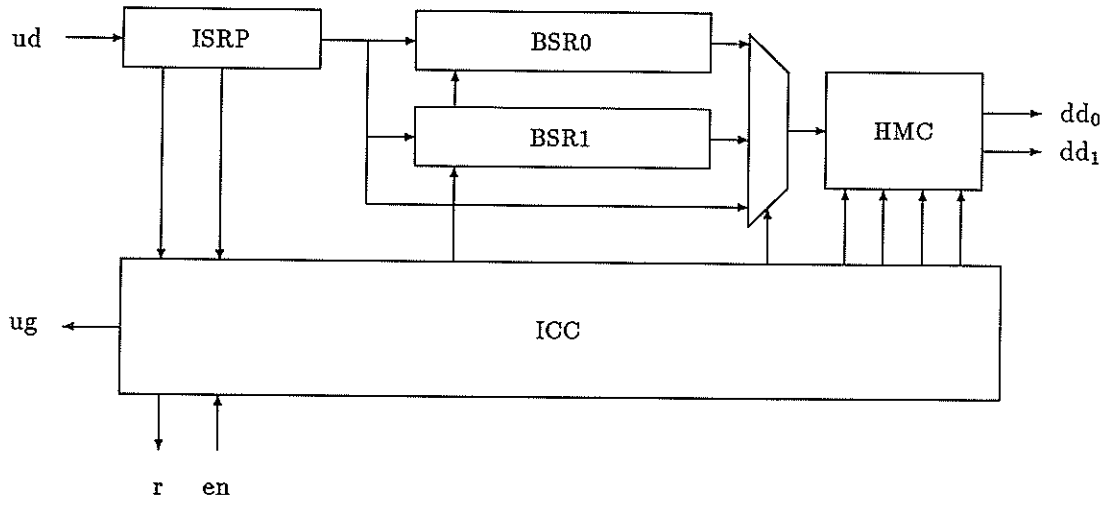

Figure 5: Input Circuit

the packet header when necessary. The structure of the input circuit is shown in Figure 5 and is explained below.

- *Input Shift Register* — The two input shift registers (ISRP, one per switch input) are 21 stage 9-bit static shift registers with an output tap after the first two stages. Packets are shifted into each input shift register from the corresponding upstream data lines. The first two stages of shift delay are for synchronization with pt. Parity is checked after the first two shift stages. The remaining 19 stages allow sufficient time for control and routing decisions to be made by packet header and output control circuit.
- *Input Control Circuit* — The ICC controls the flow of packets through the input circuit. It extracts and decodes header information from incoming packets and stores the decoded information for packets stored in the packet buffers. Using this information, it requests output ports from the OCC and based on the results, controls the flow of packets hrough the IPC. It also generates the upstream grant signals. Invalid header information sets an error flag, and forces the packet type to *point-to-point* (so that bad packets are not broadcast through the fabric).
- *Packet Buffer and Cut-Through Path* — The packet buffer shift registers (BSR, two per switch input) are 80 stage 9-bit static shift registers. Based on the availability of requested outputs, a packet is either shifted into a packet buffer, or cut through directly to an output port. A 9-bit 3-to-1 multiplexor selects one of the buffer or cut-through paths to the output routing logic.
- *Header Modification Circuit* — This component (HMC makes minor changes to the header as specified by the ICC. If the test bit is asserted, words 1-3 of the routing field are rotated, with word 1 becoming word 3, word 2 becomming 1 and word 3 becoming word 2. If the copy bit is asserted the packet is sent to both output ports and the fanout fields of the copies are modified. The low order bit of the bcn determines which copy gets the "extra" copy when the fanout value is odd. The en[01] signals control the passage of data onto the output links, with en0 enabling output 0 and en1 enabling output 1.

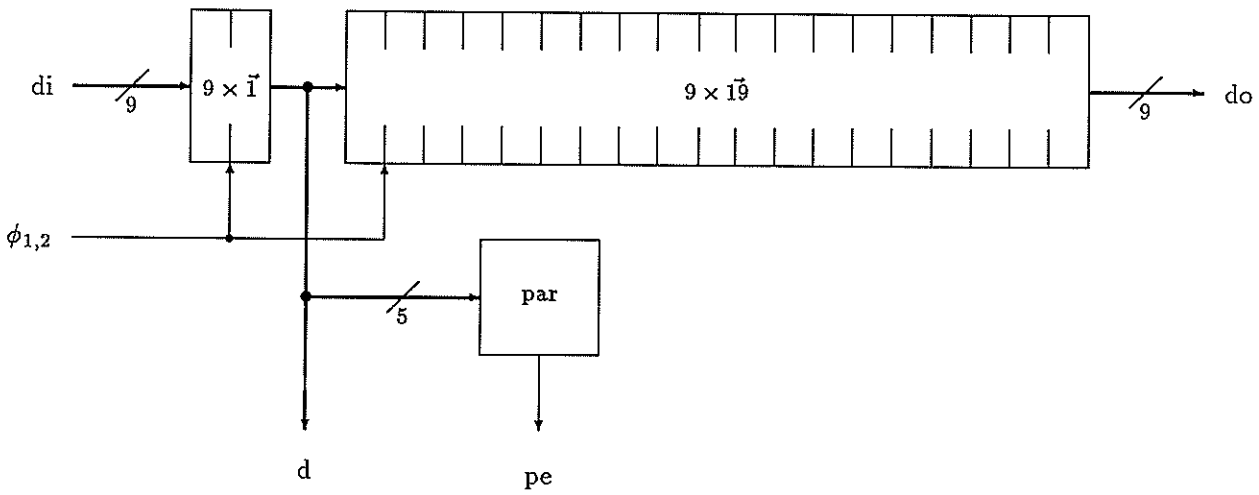

Figure 6: Input Shift Register

3. Design of Major Functional Blocks

This section provides a more detailed description of each of the functional blocks described previously.

3.1. Input Shift Register

The input shift register logic is shown in Figure 6.

The interface signals to the input shift register are:

- Inputs
  - *Upstream Data* (`di[876543210]`) – The 9-bit word input from the `di` inputs to the PSE.
  - *Clock* (`Phi1, Phi2`) – Shift register clocking.
- Outputs
  - *Data* (`do[876543210]`) – The 9-bit output from the shift register; input data delayed by 21 clock cycles.
  - *Header Data* (`d[76543210]`) – The 8-bit output from the first two stage of the shift register.
  - *Parity Check* (`pe`) – Outputs indicating odd parity error of data on the `d` lines.

The input shift register `ISRP` consists of three major components: a two stage 9-bit shift register, a parity check circuit,a nineteen stage 9-bit shift register and a clock driver.

The two stage shift register shifts 9-bit packet data in from the upstream data (`di`) inputs of the PSE, allowing for synchronization with the beginning of a packet cycle. The output of this stage is passed to the input control circuit block and to the later shift register stages.

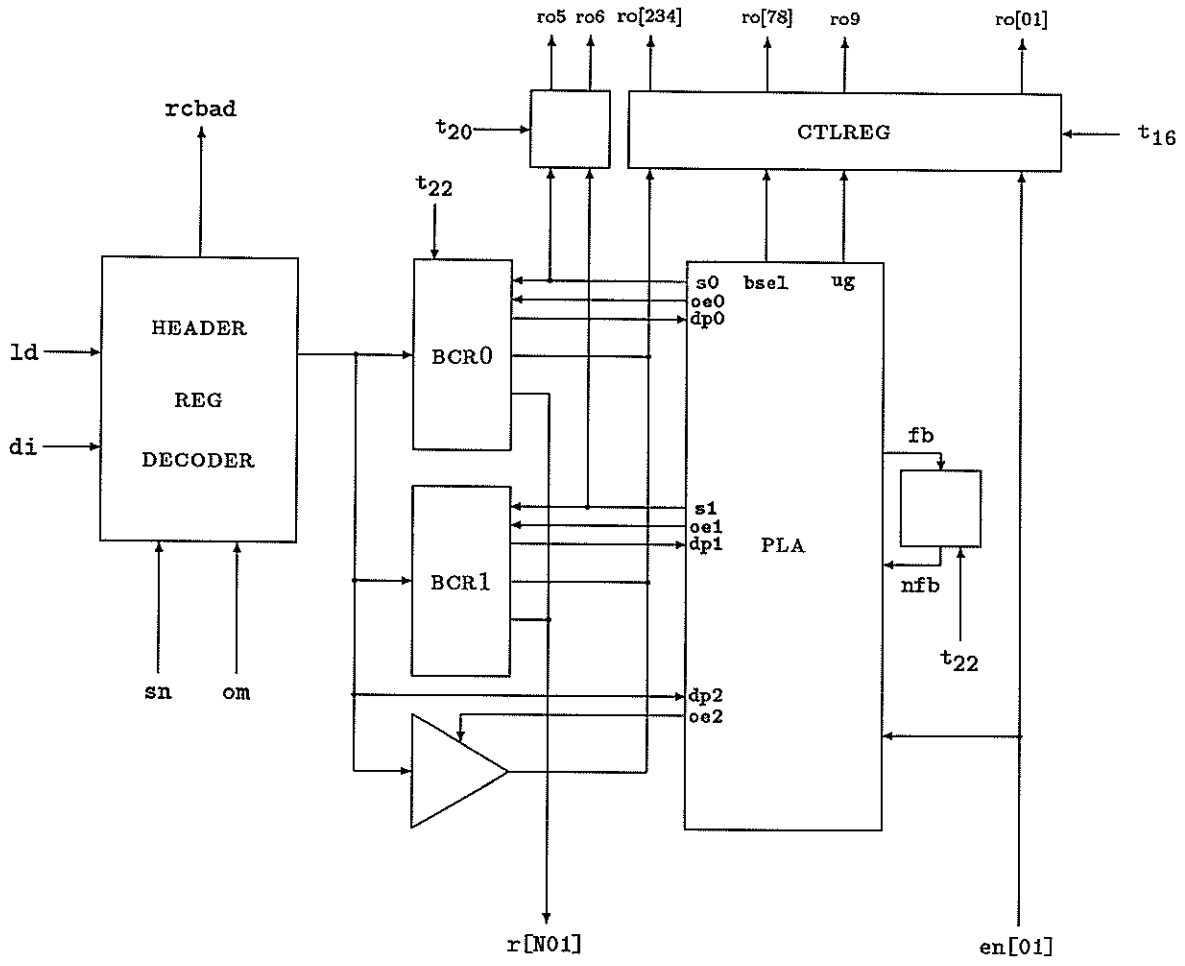

Figure 7: Input Control Circuit

The parity check circuit PAR is logically an EXCLUSIVE-OR tree to determine the parity of the incoming data. Output (pe) are passed to the ICC and ERROR to take appropriate action in case of a parity error. Parity is not restored if in error, but an error will cause the error output (err) of the PSE to go high.

The remaining 19 stages of shift allow the input control circuit circuit and output control circuit sufficient time to make control decisions that affect routing of the packet, before the packet proceeds too far into the PSE data path. These decisions include whether the packet is to be buffered or cut through, if any header modification is to occur, and to which output port the packet is destined.

3.2. Input Control Circuit

The input control circuit is shown in Figure 7.

The interface signals to the ICC are:

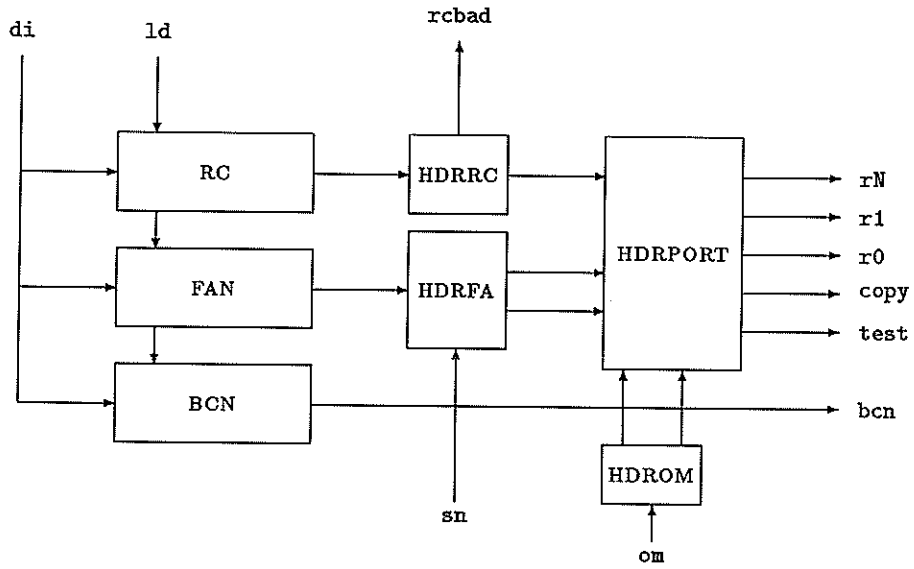

Figure 8: Header Register and Decoder

- Inputs
  - *Header Data* (`di[76543210]`) – The 8-bit word from the output of the second stage of the input shift register, which is selectively loaded into the appropriate header registers.
  - *Header Register Load* (`ld0, ld1, ld2, ld3`) – The load control lines to load the header FA, RC and BCN into the corresponding header registers.
  - *Operation Mode* (`om[10]`) – The switch network operation mode within the switch fabric.
  - *Stage Number* (`sn[210]`) – The stage number of which this PSE is a part.
  - *Routing Control Parity Error* (`pe`) – Parity of RC was invalid.
- Outputs
  - *Output Port Request* (`r[N01]`) – Switch element output port(s) requested: 000=(no port), 100=(either port), 101=(port 1), 110=(port 0), 111=(both ports).
  - *Routing Control Bad* (`rcbad`) – Invalid RC, unknown packet type.
  - *Upstream Grant* (`ro9`) – Upstream Grant signal.
  - *Select Buffer* (`ro[56]`) – Select buffer BSR0, BSR1, shift signal for buffers.
  - *HMC control signals* (`ro[0123478]`) – Control signals for HMC: `en[01],copy, bcn, test, bsel[01]`.

The header logic block HEADER consists of three major components: the header registers, the header request logic, and the header register multiplexor.

The header registers HDRREGS are four 8-bit registers (parity is not needed), which store the contents of the two header words RC, FAand also one bit of BCN. The four registers share a common input data bus, which comes from the input shift register ISRP. The load inputs are successively clocked by the timing control circuit TCC, as the header is shifted through ISRP.

The header decoder logic HDRDEC examines the relevant contents of the header registers and determines what output ports the packet requires. It includes the circuit HDRRC that examines the

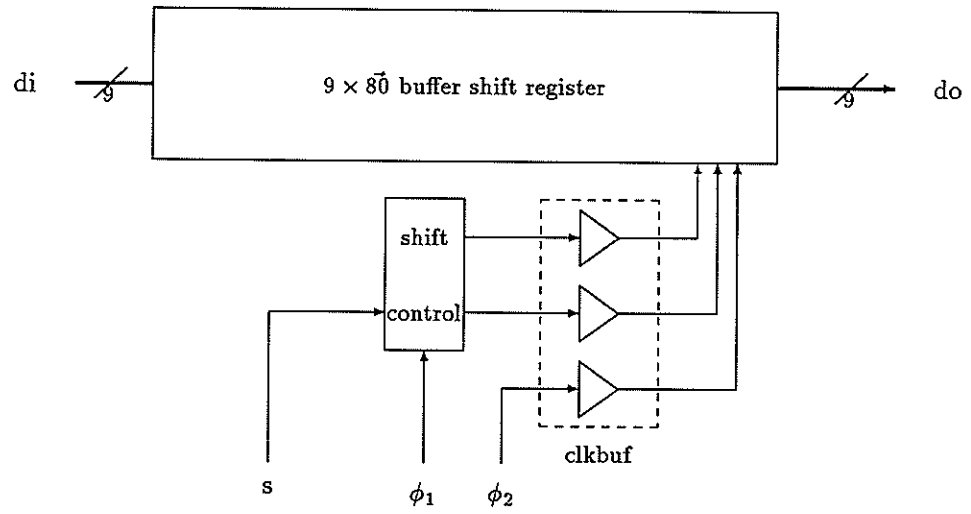

Figure 9: Buffer Shift Register

RC and determines the packet type (`nil, point, bdcst, test`), as well as indicating if the RC is invalid (`rcbad`), and determining the high order request bit (`rN`), *i.e.* if any request is to be made. The circuit `HDRFA` examines the FA. The specific output port requested (`req0'1`) is determined by examining the SN^{th} bit of the ADR in the case of a point-to-point packet. The relation of the FAN to the SN (`eq2sn, lt2sn`) is determined for broadcast packets ($FAN = 2^{SN}, FAN < 2^{SN}$). The `HDROM` circuit decodes the operation mode of the switch element. The `HDRPORT` circuit uses the outputs of `HDRRC`, `HDRFA`, and `HDROM` to determine the specific port request bits `r[10]`. The `HDRDEC` also computes the `r[N], copy` and `test` signals for incoming packets.

A PLA provides overall control of the ICC. If requesting ports are not available the decoded header information will be loaded into buffer control register `BCR0, BCR1`, corresponding to the buffer shift registers. Each `BCR` has six data inputs and six tristate data outputs. In addition the incoming header information signals have a tri-state output. The `BCR`s have two control inputs, controlled by PLA. Only one of the `BCR`s or the incoming vector can be enabled at a time, to provide the request vector to the OCC. The PLA also generates the upstream grant signals. Specification for the PLA is presented in appendix B.

3.3. Buffer Shift Register

The buffer shift register is shown in Figure 9.

The interface signals to the buffer shift register are:

- Inputs
  - *Data In* (`di[876543210]`) – The 9-bit input from ISRP.
  - *Shift Data* (`s`) – Shift data in the buffer/store data in the buffer.
  - *Clock* (`Phi1, Phi2`) – Shift register clocking.
- Outputs

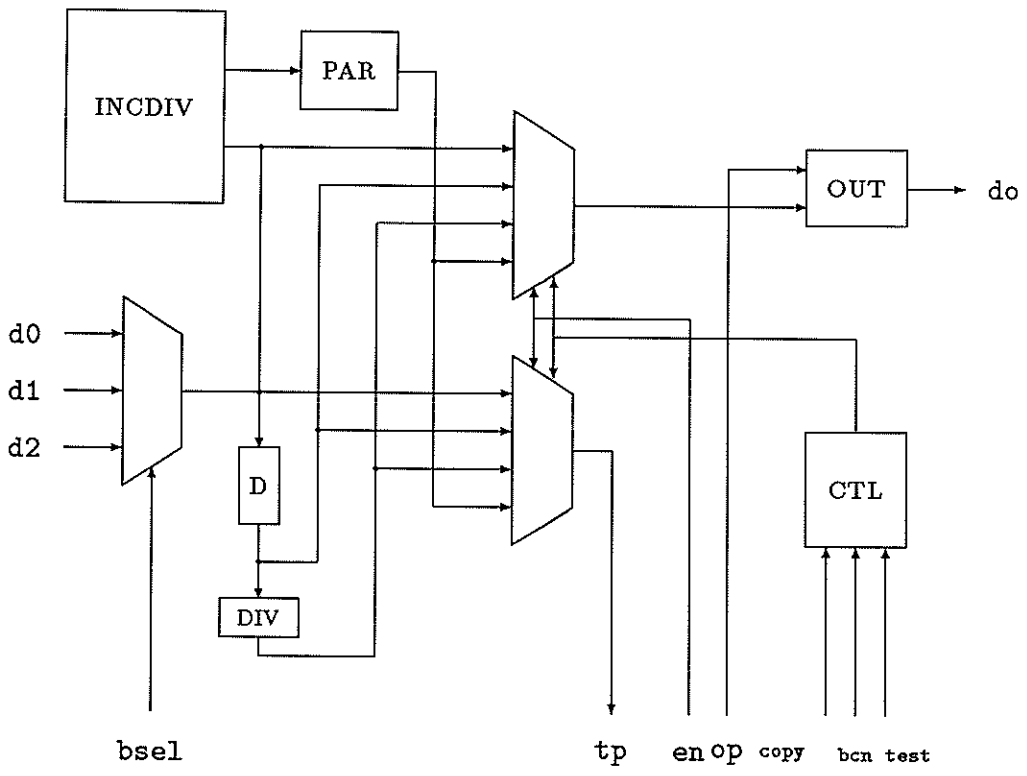

Figure 10: Header Modification circuit

– *Data Out* (do[876543210]) – The 9-bit word output from the buffer.

The packet buffer BSR is based on an 80 stage 9-bit shift register. The shift register additionally contains very large line drivers (CLKBUF) to drive the shift register clock lines, and control logic (CLKSHD) to qualify shift and hold signals with phi1. Those signals shift or hold the packet in the buffer along with phi2.

3.4. Header Modification Circuit

The output header modification circuit is shown in Figure 10.

The interface signals to the header modification circuit are:

- Inputs

- *Data Input* (d0[876543210]) – Packet data proceeding to one or both output ports, which has passed through the BSR0.
- *Data Input* (d1[876543210]) – Packet data proceeding to one or both output ports, which has gone through cut-through path.
- *Data Input* (d2[876543210]) – Packet data proceeding to one or both output ports, which has passed through the BSR1.
- *Buffer Select* (bsel[10]) – Selection of path through BSR0, BSR1 or the cut-through path.
- *Output Enable* (en[10]) – Enables the packet data to the appropriate output port.
- *Output Data from Other Port* (op[876543210]) – Packet data from the output routing logic of the other half of the PSE.
- *Multiplexor Control* (copy bcn test t1 t2) – Signals used to control the multiplexors used for changing the header fields.

- Outputs
  - *Downstream Data* (do[876543210]) – The 9-bit output to the dd outputs of the PSE.
  - *Output Data to Other Half* (tp[876543210]) – Packet data to the output routing logic of the other half of the PSE.

The header modification block HMC consists of several components: the data path multiplexor,the delay shift register D, the increment divide by two INCDIV the output multiplexor, and the output data latch.

The header modification circuit HMC performs modification of FAN for replicated broadcast packets. It consists of a PLA which increments and (integer) divides FAN by two and par calculating parity, and an EXCLUSIVE-OR gate (DIV) to maintain parity for a simple divide by two. The usual data path goes through the delaying shift register (d) and through the mainpath multiplexor which selects the BSRs or the cut-through path.

The output multiplexor OUTMUX selects the appropriate data path to proceed to switch port outputs, based on the output select from the control logic. The paths to be selected are:

- delayed data from BSR, the normal path for packet data
- data divided by two for FAN of replicated packets
- data incremented and divided by two for FAN of replicated packets
- undelayed data from MAINPATH, used only during test packet header rotation, controlled by the rrf PSE input

Two 9-bit 4-to-1 multiplexors are used, enabled such that during FAN modification of replicated packets, one output will receive the packet with FAN divided by two, and the other output the packet with FAN incremented and divided by two. The BCN of the header controls this. In particular, if BCN is odd, ouput port 0 will receive the packet with FAN simply divided by two, if BCN is even it will be port 1. This allows the determination of which copy of a packet will reach a particular output port, which is required by the BTC to perform the correct channel translation on each copy.

Input from one multiplexor in OUTMUX is passed, if enabled, through to the downstream data register OUT. Input from the other multiplexor is output (tp) to the other half of the switch, destined for the corresponding output. Packet data from this switch half is combined (ORed) with packet

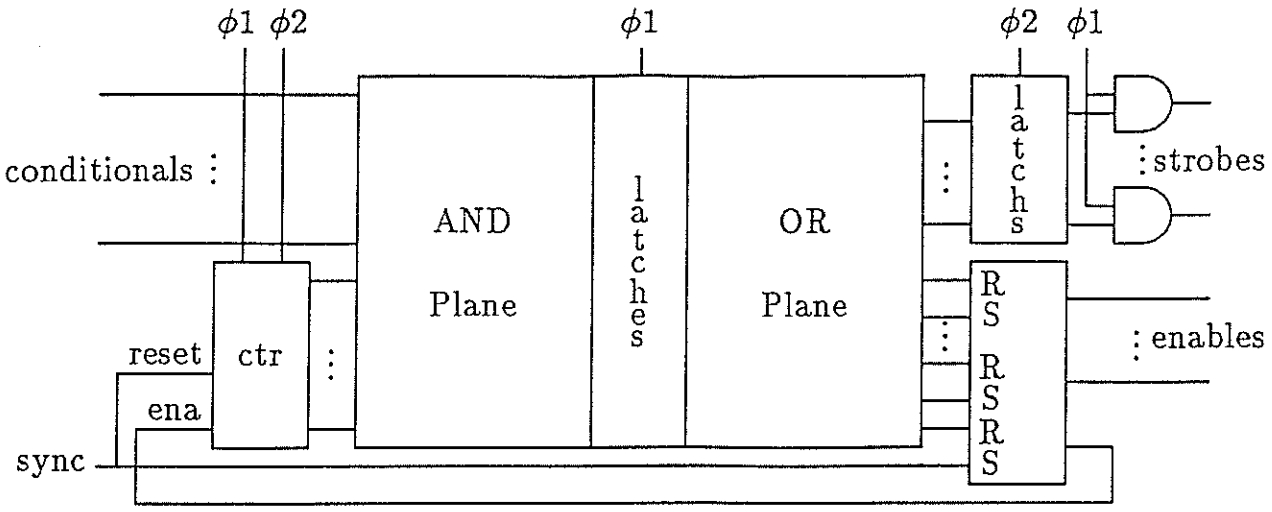

Figure 11: Timing Control Circuit

data from the other switch half (op) destined for this output. Only one of the outputs can have a path enabled to a given output port, so the enabling and combination of the data path behaves as a 2 × 2 crosspoint switch. The output is latched in a final two register stages (OUT), to be shifted out to the downstream data leads (dd) of the PSE.

3.5. Timing Control Circuit

The switch element timing control circuit is shown in Figure 11.

The interface signals to the timing control circuit are:

- Inputs
  - *Clock* (phi1, phi2, phi3) – Clock signal used by TCC.
  - *Packet time* (pt) – Start of a packet time.
  - *Reset* (rst) – Reset signal.
- Outputs
  - *Control Outputs* (t1, t2, t3, t4, t16) – Control outputs qualified with phi1.

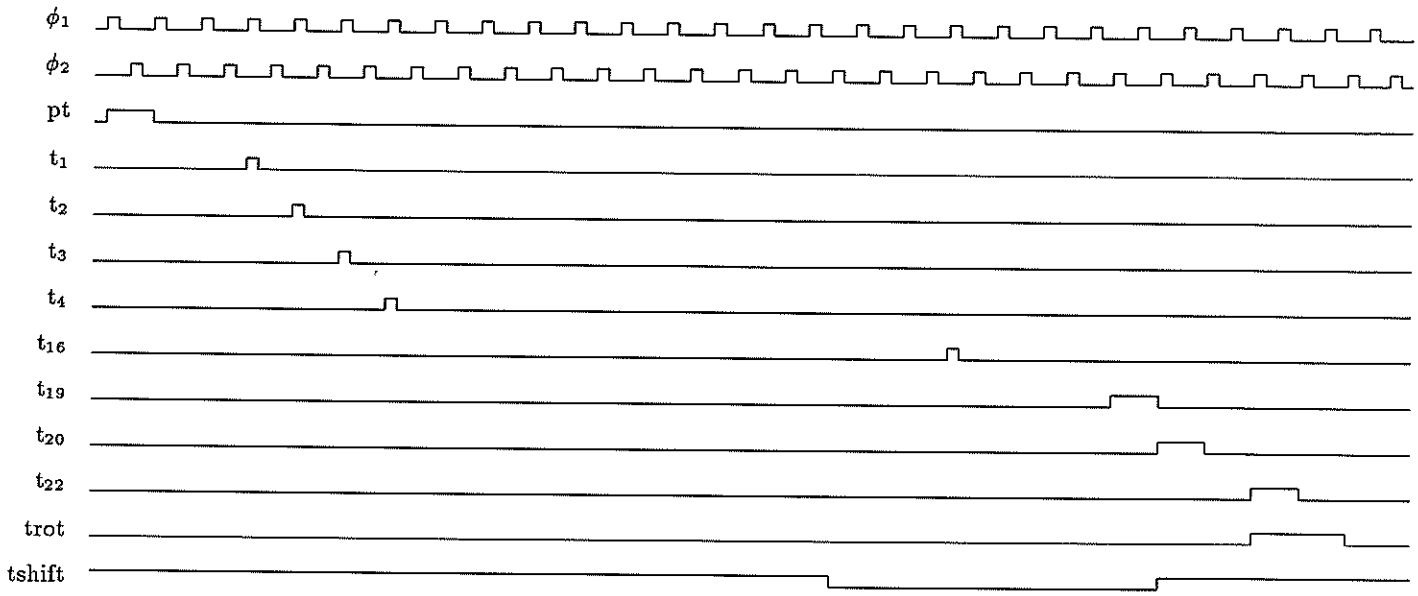

Figure 12: Timing Circuit Timing

- *Control Outputs* (`t19, t20, t22, trot, tshift`) – Control outputs lasting one clock cycle starting at phi2. except for tshift which is the time when data can be shifted into or out of the buffers and trot which lasts two clock cycles.

The timing control circuit consists of a counter, to count the clock cycle in the begining of a packet cycle, and a dynamic PLA to decode the various times and guard conditions. The outputs of the counter are fed into the decoding PLA. The PLA outputs indicate the various clock times. The PLA outputs are formed by ORing together some product terms. They are then connected to a set of R-S flip flops used to generate the enables lines. The enable lines are the buffered outputs of the flip-flops. The reset signal is latched into a flip-flop and the circuit is reset until the next packet time arrives. The TCC was generated using a timing circuit generator designed by George Robbert [Ge88].

The timing diagram for the timing circuit is shown in figure 12.

3.6. Output Control Circuit

The output control circuit is shown in Figure 13.

The interface signals to the output control circuit are:

- Inputs
  - *Output Port Requests* (`r[NO1][AB]`) – The 3-bit request vectors from the header logic of each switch side (`ra, rb`). The interpretation of the vector is: 000=(no port), 100=(either port), 110=(port 0), 101=(port 1), 111=(both ports).

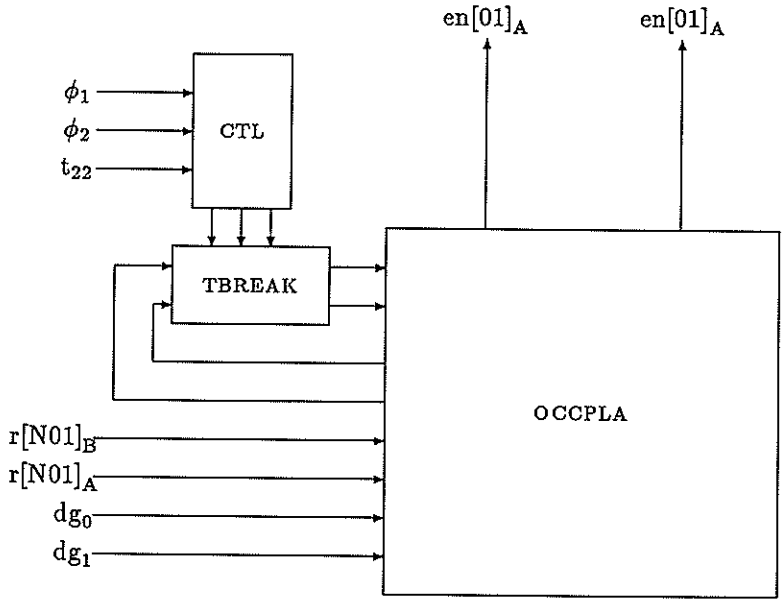

Figure 13: Output Control Circuit

- *Downstream Grants* (dg[10]) – Grant signals from downstream neighbors, indicating that permission has been given to transmit a packet on the corresponding dd port during the next packet cycle.
- *Latch Tiebreaker signals* (t20) – A control signal indicating that the uI, u0 inputs should be latched for the next packet cycle.
- *Clock* (phi1, phi2) – Clock signals used for the tiebreak latch.

- Outputs
  - *Output Enables* (e[10][AB]) – Switch element output port enable signals to each output side of the switch. The interpretation of the vector is: 00=(no port), 01=(port 0), 10=(port 1), 11=(both ports).

The output control circuit block OCC consists of a OCCPLA to generate the outputs, along with random logic to to control the latch for the tiebreak signals.

The OCCPLA uses the request vectors r[N01][AB] and additional state vectors uI, u0 to generate the output enables e[ab].

The additional state vectors is the last-used toggle (uI, u0. This is a two bit code indicating the most recently used switch input (uI=0:a, uI=1:b) and output (u0=0, u0=1). These allow the routing policies of arbitrary input tiebreaking, and uniform output distribution (under the proper circumstances) to be followed. During each packet cycle, new values are generated (newu[I0]) based on the inputs and outputs used.

The full specification for enable generation is presented in appendix A.

4. Implementation

This section discusses the implementation of the PSE as a 2μm CMOS chip.

4.1. Integrated Circuit Design

The PSE chip was designed using a set of CAD tools from the University of California at Berkeley, described in [ScMa86]. The circuit layout tool used is MAGIC, which is a graphical layout editor providing a hierarchical cell structure. Technology based design rules are checked by MAGIC, and are based on a manhattan Mead-Conway design style with λ=1.0μm [MeCo80]. This results in a minimum line width of 2μm. Chip fabrication technology is a bulk p-well CMOS process, with facilities provided by MOSIS [USCa]. Various references used during the design of the PSE are listed in the bibliography.

4.2. Circuit Design Verification

The PSE functional design was executed in a top-down hierarchical manner, but the chip was implemented bottom-up. As each cell was designed, it was simulated before incorporation into higher level cells. The various tools which were available for PSE simulation will be briefly described.

Circuit-level simulation was performed using FACTS [MCNC]. Due to the computational intensity of these simulations, it was only used up to the major blocks and not on the whole chip except for the clock signals and major control signals. All low level components were simulated in this manner, particularly the basic flip-flops, shift register cells, and clock line drivers. FACTS simulations gave us a good way to do timing analysis.

Logic-level simulation was executed using ESIM [ScMa86], which is efficient enough to allow simulation of all PSE logic functions, to the top level.

VESIM, a front-end interface to ESIM, was written by Tony Mazraani, and allows test vector input and simulation results to be printed in a vertical format. This enables test vectors considerably longer than 80 to be easily specified, and results interpreted, since the width of a terminal or print line is not a limit. It also gives a good way to compare esim output vectors with DAS output vectors, since the chip will be tested with the Tektronics DAS 9100 digital analysis system.

4.3. Packaging and Pin Asignment

The chip is implemented in a 108 pin grid array package. This package is Kyocera KD-P87938-A or equivalent with a 0.450" cavity (topside). The package is 1.200" square with the pins arranged on an 12 $\times$ 12 pin grid at 0.1" centers. There are three full rows of pins around the outside. Figure 14 depicts the bounding pad to pin connectivity as viewed from the top of the package (or the top of the socket onto which the package is plugged), with the index at the upper right corner. The columns of the grid are labeled alphabetically and the rows numerically. The signal names associated with the pins are also shown. The full description of pins can be found in Appendix C.

	M	L	K	J	H	G	F	E	D	C	B	A
1	sn2 (82)	sn1 (81)	sn0 (80)	tm1 (77)	tdA1 (74)	tdA3 (71)	tdA5 (68)	tdA7 (65)	tbo1A (62)	tbi1 (59)	tpo1A (57)	wdt (55)
2	phi1 (84)	hrst (83)	phi3 (79)	tm0 (76)	tdA0 (73)	tdA2 (70)	tdA4 (67)	tdA6 (64)	tbo0A (61)	tpo0A (58)	tsten (56)	rrf (54)
3	udA2 (86)	udA1 (85)	udA0 (78)	ten (75)	tsi0 (72)	tsi1 (69)	tso1A (66)	tso0A (63)	tbi0 (60)	dd02 (51)	dd01 (52)	dd00 (53)
4	udA5 (89)	udA4 (88)	udA3 (87)							dd05 (48)	dd04 (49)	dd03 (50)
5	udA8 (92)	udA7 (91)	udA6 (90)							dd08 (45)	dd07 (46)	dd06 (47)
6	ugA (95)	gnd (94)	vdd (93)							gnd (42)	vdd (43)	dg0 (44)
7	ugB (98)	vdd (97)	gnd (96)							vdd (39)	gnd (40)	dg1 (41)
8	udB6 (101)	udB7 (100)	udB8 (99)							dd16 (36)	dd17 (37)	dd18 (38)
9	udB3 (104)	udB4 (103)	udB5 (102)							dd13 (33)	dd14 (34)	dd15 (35)
10	udB0 (107)	udB1 (106)	udB2 (105)	tld (6)	tso1B (9)	tsoOB (12)	tmei0 (15)	tmei1 (18)	tbo1B (21)	dd10 (24)	dd11 (31)	dd12 (32)
11	phi2 (108)	om0 (2)	om1 (4)	tso (7)	tdB0 (10)	tdB2 (13)	tdB4 (16)	tdB6 (19)	tpo1B (22)	tpi0B (25)	tpi1A (29)	tboOB (30)
12	sub (1)	pt (3)	srst (5)	tsi (8)	tdB1 (11)	tdB3 (14)	tdB5 (17)	tdB7 (20)	tpoOB (23)	tpi1B (26)	tpi0A (27)	err (28)

Figure 14: Pin Assignment of the PSE Chip

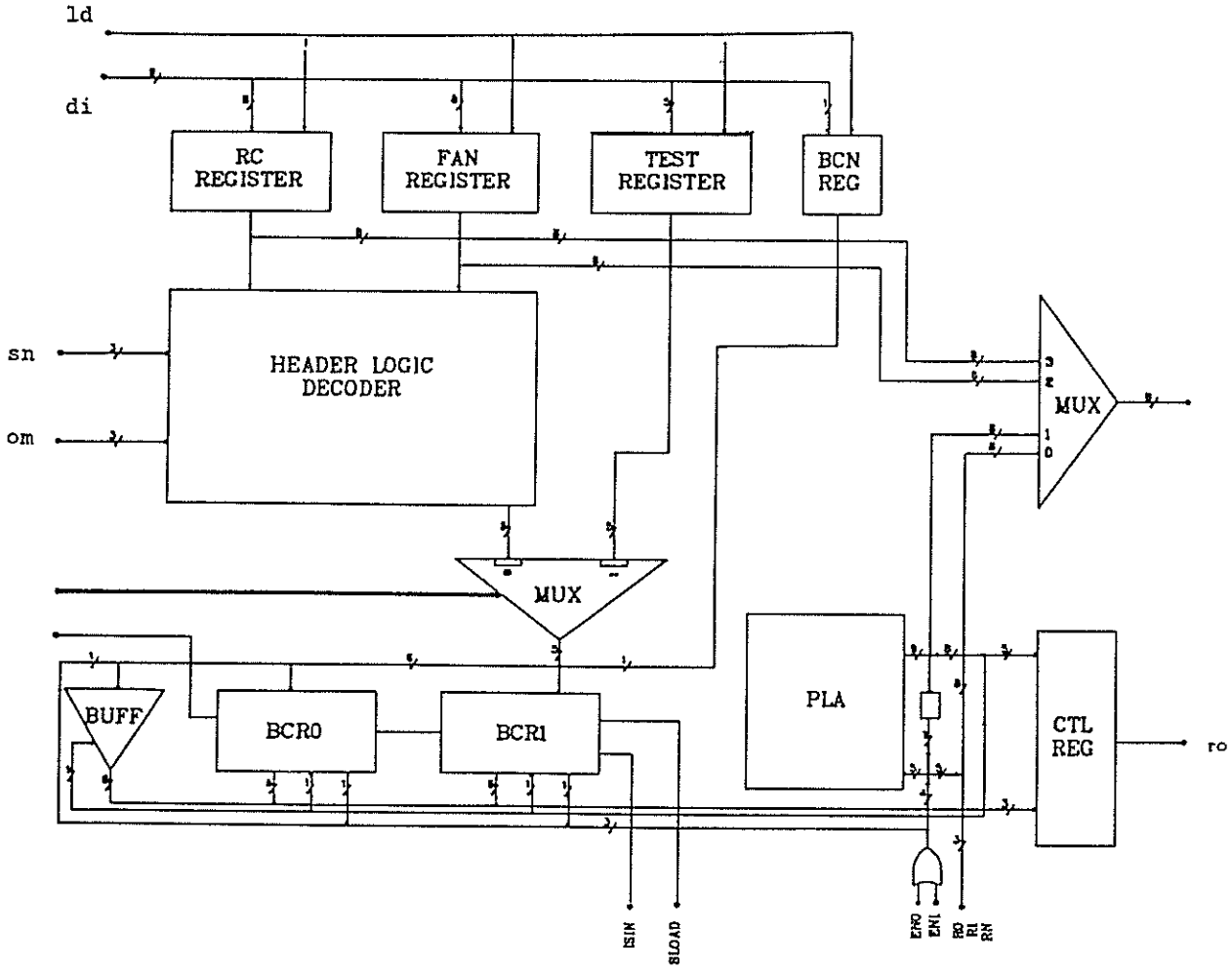

Figure 15: Test circuitry in Input Control Circuit

4.4. Testability features

A very important part of the design was testability considerations. This included pins for observation of critical signals and circuit design to disable parts of the circuit and control the rest externally. To obtain this, almost half of the 108 pins of the chip are used for testing purposes.

The main concern was being able to access the ICC. The ICC with the additional testing circuitry is shown in Figure 15.

A $8 \times 4 : 1$ multiplexor makes it possible to observe the header registers and the input and output from the `pla` which has the overall control of the packet flow through the PSE. Also the request vector `r[NO1]` is observed. The multiplexor output is `td[AB][7654321]`, and is enabled by the following:

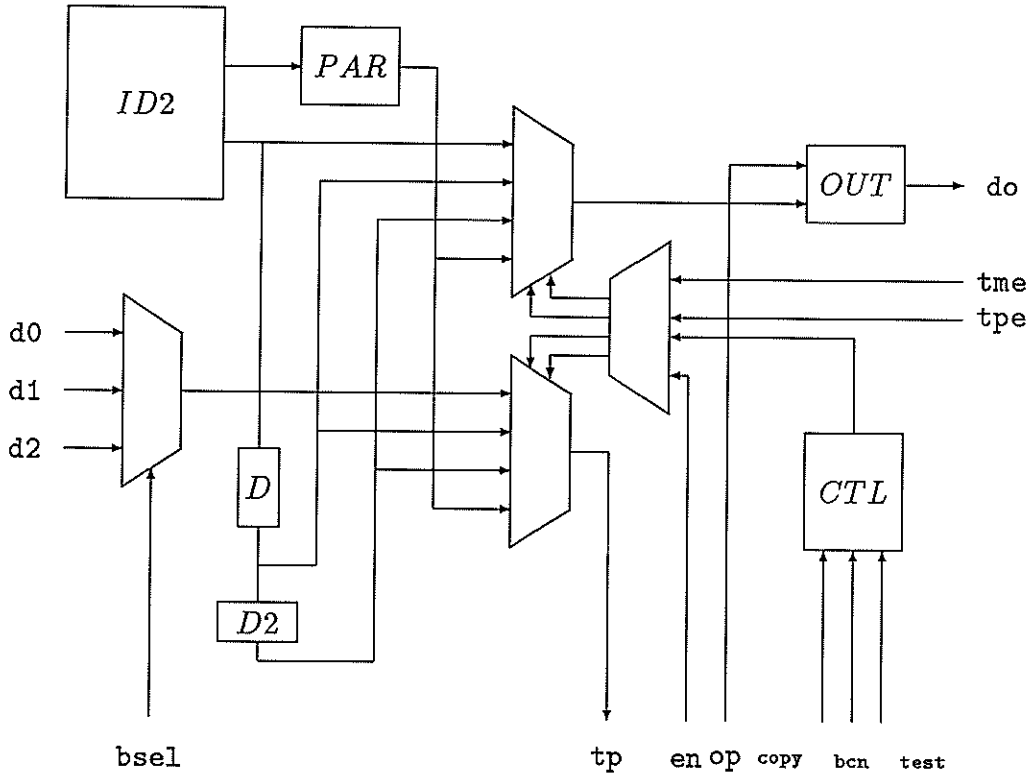

Figure 16: Test Circuitry in Header Modification Circuit

- tm[10] = 00 — Output of pla.
- tm[10] = 01 — Input of pla and the request vector. td[765] = r[N01].
- tm[10] = 10 — Output of FAN register.
- tm[10] = 11 — Output of RC register.

An additional test register is used to bypass the header logic decoder and load the request vector straight in. A $5 \times 2 : 1$ multiplexor is used to select either the decoder or the test register. ten = 0 selects the decoder and ten = 1 selects the test register. Furthermore there is also included the possibility to serial load the buffer control registers BCR. If signal pin tld is high the data on tsi is serially loaded and tso is the output serial data. The BCR for the two ports are serially connected in the following order from the input tsi: $\mathrm{BSR0_B, BSR1_B, BSR0_A, BSR1_A}$.

The shift signal for the buffer shift registers can be observed with tso[10][AB] and the buffers can also be controlled externally with tsi0 for BSR0 for both ports and tsi1 for BSR1 for both

ports. Similarly the select signal of the mainpath multiplexor can be observed at `tbo[10][AB]` and controlled externally with `tbi[10]`, same signal controls both ports. The enable signals to enable the ports can be observed at `tpo[10][AB]` and controlled externally with `tpi[10][AB]`. The multiplexors in HMC can also be controlled with `tmei[10]` as follows:

- `tmei[10]` = `00` — select divide by two
- `tmei[10]` = `01` — select increment and divide by two
- `tmei[10]` = `10` — select delayed path
- `tmei[10]` = `11` — select undelayed path

These testing featured were accomplised by adding some multiplexors in ICC and HMC, the circuit of HMC is shown in Figure 16. The testing input is enabled when `tsten` is asserted high.

References

[Di88] Dillinger, Thomas E., *VLSI Engineering*, Prentice-Hall, Engelwood Cliffs, N.J., 1988.

[GlDo85] Glasser, Lance A., and Daniel W. Dobberpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley, Reading, Mass., 1985.

[He88] Heinbuch, Dennis V., ed., *CMOS3 Cell Library*, Addison-Wesley, Reading, Mass., 1988.

[Ma87] Maly, W., *Atlas of IC Technologies: An Introduction to VLSI Processes*, Benjamin Cummings, Menlo Park, Calif., 1987.

[MCNC] *VIVID System FACTS Fast Circuit Simulator Designer Tutorial Version 1.3*, and, *VIVID System FACTS Fast Circuit Simulator Designer Reference, Version 1.3*, Microelectronics Center of North Carolina, Research Triangle Park, N.C., 1985.

[MeCo80] Mead, Carver, and Lynn Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, Mass., 1980.

[Mu86] Mukherjee, Amar, *Introduction to nMOS and CMOS VLSI Systems Design*, Prentice-Hall, Engelwood Cliffs, N.J., 1986.

[NeMa83] Newkirk, John and Robert Mathews, *The VLSI Designer's Library*, Addison-Wesley, Reading, Mass., 1983.

[Ou85] Ousterhout, John, "Using Crystal for Timing Analysis", in, *1986 VLSI Tools*, University of California at Berkeley, Computer Science Division (EECS), Report No. UCB/CSD 86/272, Berkeley, Calif., Dec. 1986

[PuEs88] Pucknell, Douglas A., and Kamran Eshraghian, *Basic VLSI Design: Systems and Circuits*, Prentice-Hall, Engelwood Cliffs, N.J., 1988.

[ScMa86] Scott, Walter S., Robert N. Mayo, Gordon Hamachi, and John K. Ousterhout, ed., *1986 VLSI Tools*, University of California at Berkeley, Computer Science Division (EECS), Report No. UCB/CSD 86/272, Berkeley, Calif., Dec. 1986

[Sh88] Shoji, Masakazu, *CMOS Digital Circuit Technology*, Prentice-Hall, Engelwood Cliffs, N.J., 1988.

[Ge88] Robbert, George., *A Circuit Generator for Synchronus Streams Processors*, Washington University Computer Science Department, MS thesis, St. Louis, May 1988

[St88a] Sterbenz, James P. G., *Design of a VLSI Packet Switch Element*, Washington University Computer Science Department, WUCS-88-5, St. Louis, April 1988

[St88b] Sterbenz, James P. G., *Design and Implementation of a VLSI Packet Switch Element*, Washington University Computer and Communications Research Center, WUCCRC-88-1, St. Louis, April 1988

[Tu85] Turner, Jonathan S., *Design of a Broadcast Packet Switching Network*, Washington University Computer Science Department, WUCS-85-4, St. Louis, March 1985

[Tu86] Turner, Jonathan S., "Design of a Broadcast Packet Network," *Proceedings of IEEE Infocom'86*, IEEE Computer Society, Los Angeles, April 1986, pp. 667-675.

[Tu87] Turner, Jonathan S., *Specification of Integrated Circuits for Broadcast Packet Network*, Washington University Computer Science Department, WUCS-87-5, St. Louis, April 1987

[USCa] *MOSIS User's Manual*, University of Southern California Information Sciences Institute, Marina Del Rey, Calif., (no date)

[USCb] *MIT-SCP3U-PADS.CIF*, University of Southern California Information Sciences Institute, Marina Del Rey, Calif., (no date)

[VlZh] Vladimirescu, A., Kaihe Zhang, A.R. Newton, D.O. Pederson, and A. Sangiovanni-Vincentelli, *SPICE User's Guide*, University of California at Berkeley, Department of Electrical Engineering and Computer Sciences, Berkeley, Calif., Oct. 1983

[WeEs85] Weste, Neil H. E., and Kamran Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley, Reading, Mass., 1985.

A. Packet Switch Element Routing Policy

This section documents the PSE routing policy in detail. First, all of the required inputs, outputs, and internal state variables are described. Then a set of tables describing the policy implementation is given. A general description of the routing policy is given in §2.1 of this technical report.

The tables are used as input to the MPLA program to generate the PLA ENGEN, which is used in the output enable circuit ENABLE to generate the e vectors.

An entry of "X" for an input variable indicates a "don't care" condition. An entry of "–" for an output variable indicates that its output state will not change.

A.1. State variable definition

Output request vectors:

ra	ra[N10]	three bit code specifying output ports requested by input a
rb	rb[N10]	three bit code specifying output ports requested by input b
r	r[N10]	three bit code specifying output ports requested by an input

rN specifies that an output port is needed
r1 specifies output port 1 is needed
r0 specifies output port 0 is needed

0XX specifies no ports requested
100 specifies either (one) output port needed
101 specifies output port 0 needed
110 specifies output port 1 needed
111 specifies both output ports needed

Output enable vectors:

ea	ea[10]	two bit code specifying output ports enabled for input a
eb	eb[10]	two bit code specifying output ports enabled for input b

e	e[10]	two bit code specifying output ports enabled for an input

e1 specifies output port 1 will be enabled
e0 specifies output port 0 will be enabled

00 specifies no ports enabled
01 specifies port 0 enabled
10 specifies port 1 enabled
11 specifies both ports enabled

Output port availability:

p	p[10]	two bit code specifying output ports available

p1 specifies that output port 1 is available
p0 specifies that output port 0 is available

00 specifies no ports are available
01 specifies only port 0 is available
10 specifies only port 1 is available
11 specifies both ports are available

Last used:

u	u[I0]	two bit code specifying most recently used switch ports

uI=0 specifies port a was most recently used single input
uI=1 specifies port b was most recently used single input
uO=0 specifies port 0 was most recently used single output
uO=1 specifies port 1 was most recently used single output

A.2. Routing policy control tables

rb N10	ra N10	p 10	u I0	eb 10	ea 10	u I0	DESCRIPTION
							no outputs are available
XXX	XXX	00	XX	00	00	--	enable no outputs
							neither input has requests
0XX	0XX	XX	XX	00	00	--	enable no outputs
							input b has no requests
0XX	100	01	XX	00	01	--	input a needs either output only 0 available: enable 0
0XX	100	10	XX	00	10	--	only 1 available: enable 1
0XX	100	11	X0	00	10	-1	both available: enable $\overline{u0}$=1
0XX	100	11	X1	00	01	-0	both available: enable $\overline{u0}$=0
0XX	101	X0	XX	00	00	--	input a needs output 0 unavailable
0XX	101	X0	XX	00	01	--	available:enable 0
0XX	110	0X	XX	00	00	--	input a needs output 1 unavailable
0XX	110	1X	XX	00	10	--	available: enable 1
0XX	111	0X	XX	00	00	--	input a needs both outputs 1 unavailable
0XX	111	X0	XX	00	00	--	0 unavailable
0XX	111	11	XX	00	11	--	both available: enable both
							input a has no requests
100	0XX	01	XX	01	00	--	input b needs either output only 0 available: enable 0
100	0XX	10	XX	10	00	--	only 1 available: enable 1
100	0XX	11	X0	10	00	-1	both available: enable $\overline{u0}$=1
100	0XX	11	X1	01	00	-0	both available: enable $\overline{u0}$=0
101	0XX	X0	XX	00	00	--	input b needs output 0 unavailable
101	0XX	X1	XX	01	00	--	available:enable 0
110	0XX	0X	XX	00	00	--	input b needs output 1 unavailable
110	0XX	1X	XX	10	00	--	available: enable 1
111	0XX	0X	XX	00	00	--	input b needs both outputs 1 unavailable
111	0XX	X0	XX	00	00	--	0 unavailable
111	0XX	11	XX	11	00	--	both available: enable both

rb N10	ra N10	p 10	u I0	eb 10	ea 10	u I0	DESCRIPTION
							both inputs need different specific outputs
101	110	01	XX	01	00	--	input a needs output 1, b needs output 0 only 0 available: enable 0 for b
101	110	10	XX	00	10	--	only 1 available: enable 1 for a
101	110	11	XX	01	10	--	both available: enable both
110	101	01	XX	00	01	--	input a needs output 0, b needs output 1 only 0 available: enable 0 for a
110	101	10	XX	10	00	--	only 1 available: enable 1 for b
110	101	11	XX	10	01	--	both available: enable both
							both inputs need the same specific output
101	101	X0	XX	00	00	--	both inputs need output 0 unavailable
101	101	X1	0X	00	01	1-	only 0 available, favor $\overline{uI}$=0: enable for a
101	101	X1	1X	01	00	0-	only 0 available, favor $\overline{uI}$=1: enable for b
110	110	0X	XX	00	00	--	both inputs need output 1 unavailable
110	110	1X	0X	00	10	1-	only 1 available, favor $\overline{uI}$=1: enable for a
110	110	1X	1X	10	10	0-	only 1 available, fovor $\overline{uI}$=0: enable for b

rb N10	ra N10	p 10	u I0	eb 10	ea 10	u I0	DESCRIPTION
							input b needs either output
							input a needs output 0
100	101	01	XX	00	01	--	only 0 available: enable for a
100	101	10	XX	10	00	--	only 1 available: enable for b
100	101	11	XX	10	01	--	both available
							input a needs output 1
100	110	01	XX	01	00	--	only 0 available: enable for b
100	110	10	XX	00	10	--	only 1 available: enable for a
100	110	11	XX	01	10	--	both available
							input a needs either output
							input b needs output 0
101	100	01	XX	01	00	--	only 0 available: enable for b
101	100	10	XX	00	10	--	only 1 available: enable for a
101	100	11	XX	01	10	--	both available
							input b needs output 1
110	100	01	XX	00	01	--	only 0 available: enable for a
110	100	10	XX	10	00	--	only 1 available: enable for b
110	100	11	XX	10	01	--	both available
							both inputs need either output
100	100	01	0X	00	01	1-	only 0 available, favor $\overline{uI}$=1: enable for a
100	100	01	1X	00	01	0-	only 0 available, favor $\overline{uI}$=0: enable for b
100	100	10	0X	10	00	1-	only 1 available, favor $\overline{uI}$=1: enable for a
100	100	10	1X	00	10	0-	only 1 available, favor $\overline{uI}$=0: enable for b
100	100	11	X0	10	01	-0	both available
100	100	11	X1	01	10	-1	both available

rb N10	ra N10	p 10	u IO	eb 10	ea 10	u IO	DESCRIPTION
							one input needs both outputs, other input needs either
100	111	01	XX	01	00	--	input a needs both outputs, input b either only 0 available: enable for b
100	111	10	XX	10	00	--	only 1 available: enable for b
100	111	11	XX	00	11	--	both available: enable both for a
111	100	01	XX	00	01	--	input b needs both outputs, input a either only 0 available: enable for a
111	100	10	XX	00	10	--	only 1 available: enable for a
111	100	11	XX	11	00	--	both available: enable both for b
							one input needs both outputs, other a specific output
101	111	01	XX	01	00	--	input a needs both, input b needs output 0 only 0 available: enable for b
101	111	10	XX	00	00	--	0 unavailable
101	111	11	XX	00	11	--	both available: enable both for a
110	111	01	XX	00	00	--	input a needs both, input b needs output 1 1 unavailable
110	111	10	XX	10	00	--	only 1 available: enable for b
110	111	11	XX	00	11	--	both available: enable both for a
111	101	01	XX	00	01	--	input b needs both, input a needs output 0 only 0 available, enable for a
111	101	10	XX	00	00	--	0 unavailable
111	101	11	XX	11	00	--	both available: enable for b
111	110	01	XX	00	00	--	input b needs both, input a needs output 1 1 unavailable
111	110	10	XX	00	10	--	only 1 available, enable for a
111	110	11	XX	11	00	--	both available: enable for b
							both inputs need both outputs
111	111	01	XX	00	00	--	only 0 available
111	111	10	XX	00	00	--	only 1 available
111	111	11	0X	11	00	1-	both available, favor $\overline{\text{uI}}$=1: enable both
111	111	11	1X	00	11	0-	both available, favor $\overline{\text{uI}}$=0: enable both

B. Input Control Circuit PLA State Table

This section documents the ICC PLA state table in detail. First, all of the required inputs, outputs, and internal state variables are described. Then a set of tables describing the state implementation is given. A general description of the state table is given in §3.

The tables are used as input to the MPLA program to generate the PLA, which is used in the input control circuit ICC for owerall control of data flow.

B.1. State variable definition

Input to the PLA:

en	Output ports available for current request vector
dp_2	Incoming packet is requesting an output port
dp_1	Packet in BSR1 requesting an output port
dp_0	Packet in BSR0 requesting an output port
fb	Which packet came in first

Output from the PLA:

s1	Select signal for BSR1 1
s0	Select signal for BSR0 1
bsel[10]	Select multiplexor for data path through buffers or cut-through
nbf	Which packet came in first update
oe_2	Enable incoming request vector
oe_1	Enable BCR1 as request vector
oe_0	Enable BCR0 as request vector
ug	Upstream grant signal

B.2. PLA State Table

en	dp_2	dp_1	dp_0	fb	s_1	s_1	bsel[10]	nfb	oe_2	oe_1	oe_0	ug
0	0	0	0	x	0	0	11	x	1	0	0	1
0	0	0	1	x	0	0	11	0	0	0	1	1
0	0	1	0	x	0	0	11	1	0	1	0	1
0	0	1	1	x	0	0	11	-	0	0	1	0
0	1	0	0	x	0	1	11	0	1	0	0	1
0	1	0	1	x	1	0	11	0	0	0	1	0
0	1	1	0	x	0	1	11	1	0	1	0	0
0	1	1	1	x	x	x	xx	x	0	0	0	0
1	0	0	0	x	0	0	11	x	1	0	0	1
1	0	0	1	x	0	1	00	x	0	0	1	1
1	0	1	0	x	1	0	01	x	0	1	0	1
1	0	1	1	0	0	1	00	1	0	0	1	1
1	0	1	1	1	1	0	01	0	0	1	0	1
1	1	0	0	x	0	0	10	x	1	0	0	1
1	1	0	1	x	0	1	00	0	0	0	1	1
1	1	1	0	x	1	0	01	1	0	1	0	1
1	1	1	1	x	x	x	xx	x	0	0	0	0

C. Pin Assignment and Signal Names

This section documents the pin assignment and signal names in detail.

Pin	I/O	signal name	comment
1	blank	sub	Substrate
2	input	om0	operational mode bit 0
3	input	pt	packet time
4	input	om1	operational mode bit 1
5	input	srst	soft reset
6	input	tld	load serial BCR
7	output	tso	serial output for BCR
8	input	tsi	serial input for BCR
9	output	tso1B	enable ICC(B) data mux bit 1
10	output	tdB0	test data from ICC(B)
11	output	tdB1	test data form ICC(B)
12	output	tso0B	enable ICC(B) data mux bit 0
13	output	tdB2	test data form ICC(B)
14	output	tdB3	test data form ICC(B)
15	input	tmei0	mux enable for output ports bit 0
16	output	tdB4	test data form ICC(B)
17	output	tdB5	test data form ICC(B)
18	input	tmei1	mux enable for output ports bit 1
19	output	tdB6	test data form ICC(B)
20	output	tdB7	test data form ICC(B)
21	output	tbo1B	bsel1 for port B
22	output	tpo1B	port enable signal bit 1 for port B
23	output	tpo0B	port enable signal bit 0 for port B
24	output	ddB0	output data on port B
25	input	tpi0B	port enable signal bit 0 for port B
26	input	tpi1B	port enable signal bit 1 for port B
27	input	tpi0A	port enable signal bit 0 for port A
28	output	err	parity or RC error
29	input	tpi1A	port enable signal bit 1 for port A
30	output	tbo0B	bsel0 for port B
31	output	ddB1	output data on output port 1
32	output	ddB2	output data on output port 1
33	output	ddB3	output data on output port 1
34	output	ddB4	output data on output port 1
35	output	ddB5	output data on output port 1
36	output	ddB6	output data on output port 1

Pin	I/O	signal name	comment
37	output	ddB7	output data on output port 1
38	output	ddB8	output data on output port 1
39	vdd	Vdd	power suppply voltage
40	gnd	GND	reference voltage; ground
41	input	dg1	downstream grant for output port 1
42	gnd	GND	reference voltage; ground
43	vdd	Vdd	power sypply voltage
44	input	dg0	downstream grant for output port 0
45	output	ddA8	output data on output port 0
46	output	ddA7	output data on output port 0
47	output	ddA6	output data on output port 0
48	output	ddA5	output data on output port 0
49	output	ddA4	output data on output port 0
50	output	ddA3	output data on output port 0
51	output	ddA2	output data on output port 0
52	output	ddA1	output data on output port 0
53	output	ddA0	output data on output port 0
54	output	rrf	rotating routing field
55	output	wdt	watchdog timer
56	input	tsten	enable to bypass control signals
57	output	tpo1A	port enable signal bit 1 for port A
58	output	tpo0A	port enable signal bit 0 for port A
59	input	tbi1	buffer select bit 1
60	input	tbi0	buffer select bit 0
61	output	tbo0A	buffer select bit 0
62	output	tbo1A	buffer select bit 1
63	output	tso0A	BSR 0 shift signal
64	output	tdA6	test data for ICC(A)
65	output	tdA7	test data for ICC(A)
66	output	tso1A	BSR 1 shift signal
67	output	tdA4	test data for ICC(A)
68	output	tdA5	test data for ICC(A)
69	input	tsi1	BSR 1 shift signal
70	output	tdA2	test data for ICC(A)
71	output	tdA3	test data for ICC(A)
72	input	tsi0	BSR 0 shift signal

Pin	I/O	signal name	comment
73	output	tdA0	test data from ICC(A)
74	output	tdA1	test data for ICC(A)
75	input	ten	enable bypass decoder mux
76	input	tm0	enable ICC(A) data mux bit 0
77	input	tm1	enable ICC(A) data mux bit 1
78	output	udA0	input data on port A
79	input	phi3	clock phase 3
80	input	sn0	stage number bit 0
81	input	sn1	stage number bit 1
82	input	sn2	stage number bit 2
83	input	hrst	hard reset
84	input	phi1	clock phase 1
85	input	udA1	input data for input port A
86	input	udA2	input data for input port A
87	input	udA3	input data for input port A
88	input	udA4	input data for input port A
89	input	udA5	input data for input port A
90	input	udA6	input data for input port A
91	input	udA7	input data for input port A
92	input	udA8	input data for input port A
93	vdd	Vdd	power supply voltage
94	gnd	GND	reference voltage; ground
95	output	ugA	upstream grant for input port A
96	gnd	GND	reference voltage; ground
97	vdd	Vdd	power supply voltage
98	output	ugB	upstream grant for input port B
99	input	udB8	input data for input port B
100	input	udB7	input data for input port B
101	input	udB6	input data for input port B
102	input	udB5	input data for input port B
103	input	udB4	input data for input port B
104	input	udB3	input data for input port B
105	input	udB2	input data for input port B
106	input	udB1	input data for input port B
107	input	udB0	input data for input port B
108	input	phi2	clock phase 2

D. Interconnection of Circuit Blocks

This section shows a detailed diagram of the chip.

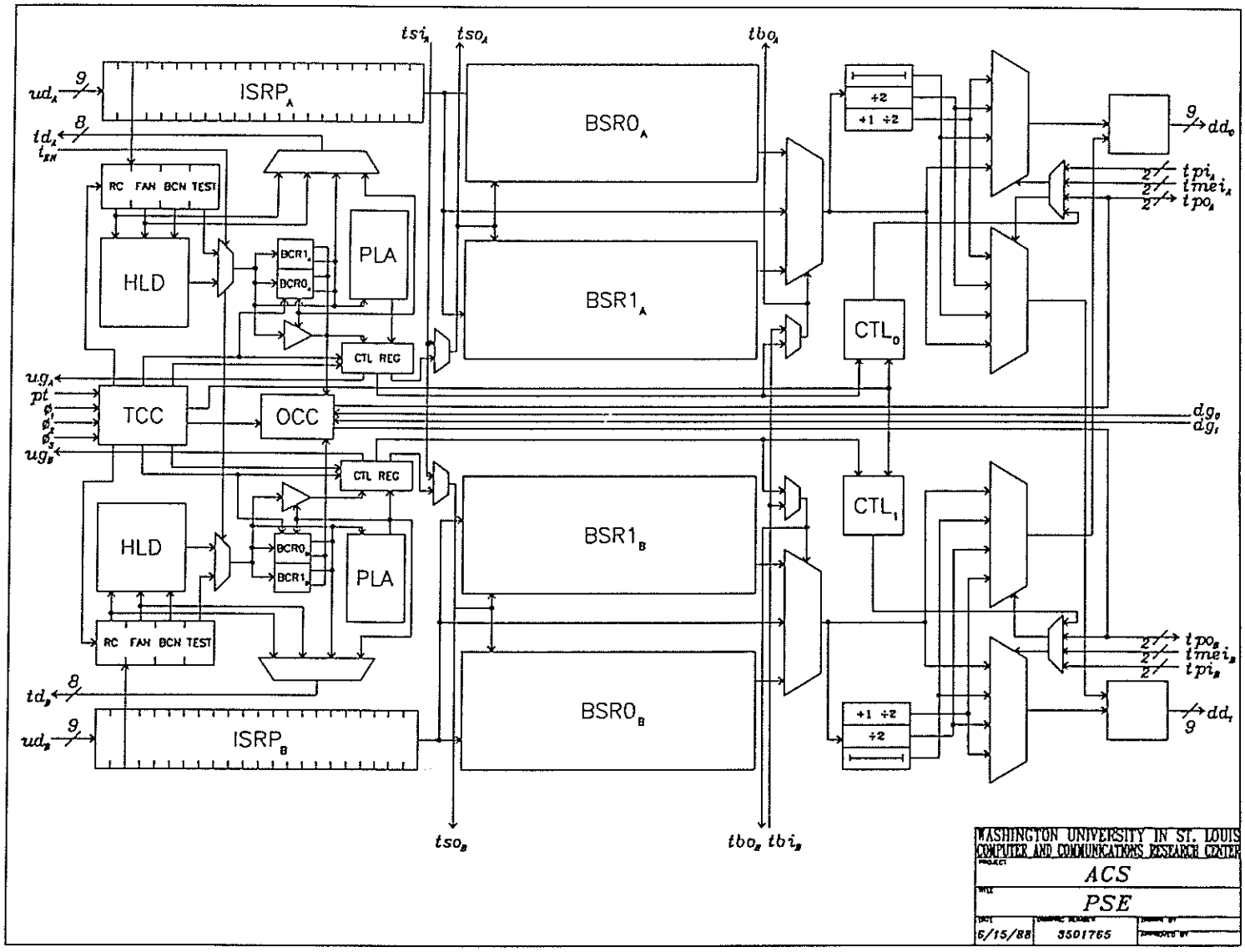

Figure 17: Chip block diagram