Report Number: WUCS-89-42

1989-10-01

Design of PP3 a Packet Processor Chip

Authors: Hai-Feng Bi

This paper describes the design of the PP3 packet processor chip. PP3 is one of the four component chips in a packet processor used in the high speed broadcast packet switching network [Tu88]. Together with the other three component chips, PP3 provides the interface between the fiber optic links and the switch fabric. PP3 in currently being fabricated in 2 μm CMOS technology.

DESIGN OF PP3
A PACKET PROCESSOR CHIP

Hai-Feng Bi

WUCS-89-42

October 1989

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 PP3 packet processor chip. PP3 is one of the four component chips in a packet processor used in the high speed broadcast packet switching network [Tu88]. Together with the other three component chips, PP3 provides the interface between the fiber optic links and the switch fabric. PP3 is currently being fabricated in 2 μm CMOS technology.

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

Contents

List of Figures

DESIGN OF PP3
A PACKET PROCESSOR CHIP

Hai-Feng Bi
bi@wuccrc.wustl.edu

1. Introduction

This paper describes the design of the PP3 packet processor chip. PP3 is one of the four component chips in a packet processor used in the high speed broadcast packet switching network. Together with the other three component chips, PP3 provides the interface between the fiber optic links and the switch fabric. PP3 is currently being implemented in 2 μm CMOS technology.

1.1 Switch Module

The Advanced Networks Group is developing a prototype fast packet switching system supporting link speeds of 100 Mb/s and a general multicast capability, which has been named the *Broadcast Packet Switch.* The overall structure of the prototype packet switch [Tu88] is shown in Figure 1. The switch module terminates 15 fiber optic links. The Connection Processor (CP), shown at the top of the figure, is a general purpose computer that provides the overall control of the system, including connection establishment. The heart of the system is a 16-port Switch Fabric (SF), comprising a Copy Network (CN), a set of Broadcast Translation Circuits (BTCs), a Distribution Network (DN) and a Routing Network (RN). The CN, DN and RN are composed of binary Packet Switch Elements (PSE) that perform replication, load balancing and packet routing. The interface between the SF and the high speed Fiber Optic Links (FOLs) is provided through a set of Packet Processors (PPs).

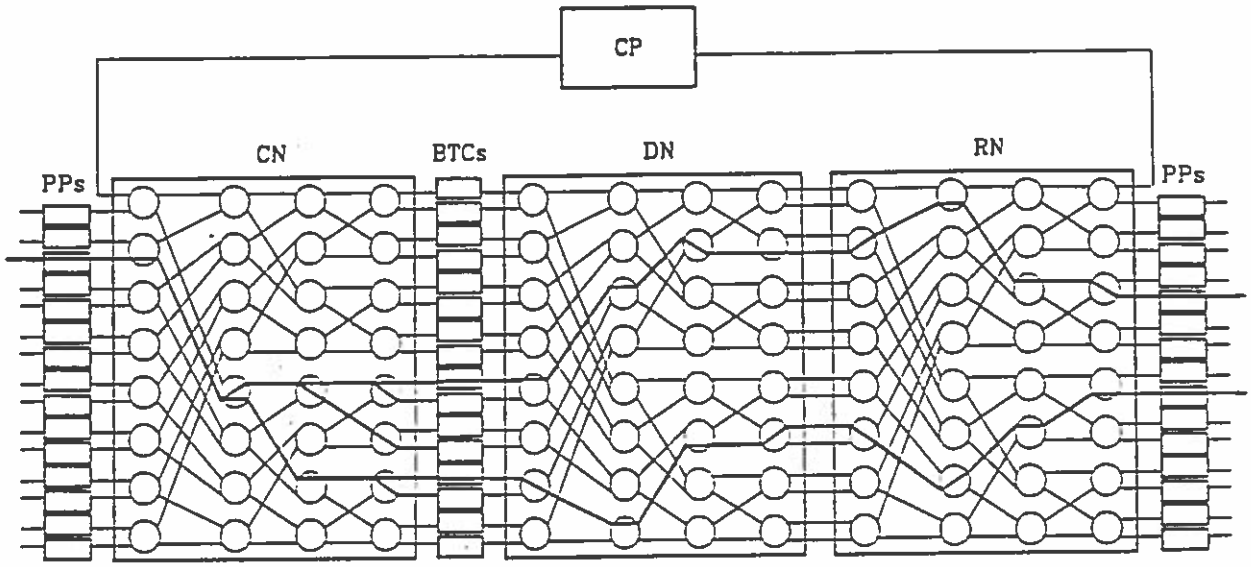

Figure 1. *Switch Module*

1.2 Packet Processor

The Packet Processors (PPs) form the interface between the external fiber optic links and the switch module's internal data paths. They perform all the link level protocol functions, including the determination of how packets are routed. A block diagram of the PP appears in Figure 2.

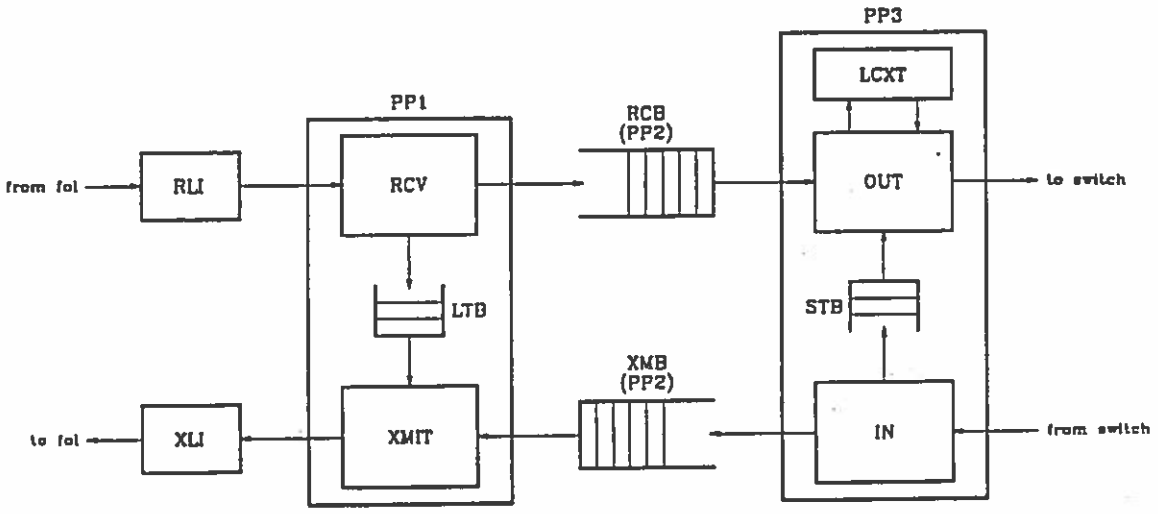

Figure 2. *Packet Processor*

- **Receive Link Interface** (RLI). Converts the incoming optical signal to an 8-bit electrical format, and provides a clock recovered from the incoming data stream.
- **Receive Circuit** (RCV). Checks incoming packets for errors, adds parity, strips off frame check, routes test packets to the LTB and other packets to the RCB.
- **Receive Buffer** (RCB). A FIFO which buffers up to 32 external packets entering the switch module from the FOL.
- **Link Test Buffer** (LTB). Provides path for test packets used to verify the operation of the FOL. The LTB can hold 2 packets.
- **Output Circuit** (OUT). Adds five bytes of header information to the front of each packet received from the RCB. Performs logical channel translation and sends packets to the SF. Also reads switch test packets and LCXT read/write packets from the STB and processes them appropriately.
- **Switch Test Buffer** (STB). Provides path for test packets used to verify the operation of the SF. The STB can hold 2 packets.
- **Logical Channel Translation Table** (LCXT). A lookup table used to translate an incoming logical channel number to the routing information needed by the SF.
- **Input Circuit** (IN). Routes internal data packets to the XMB, removing the first five bytes of header information and routes all other packets to the STB.
- **Transmit Buffer** (XMB). A FIFO which buffers up to 32 external packets exiting the switch module from the SF.
- **Transmit Circuit** (XMIT). Takes packets from the XMB, adds the SYNC field, strips parity and computes the frame check. Also processes test packets from the LTB.
- **Transmit Link Interface** (XLI). Converts from 8-bit electrical format to optical format.

Note that Figure 2 also shows how the components are divided among the 4 integrated circuit chips. The RCV, XMIT and LTB are placed on the PP1 chip. PP1 also synchronizes the data streams on its receive side and its transmit side. The RCB and XMB each consists of a PP2 chip and the IN, OUT, STB and LCXT are placed on the PP3, the chip under discussion here.

For reference purposes, the external and internal packet formats and the definitions of the fields are provided in Figures 3 and 4.

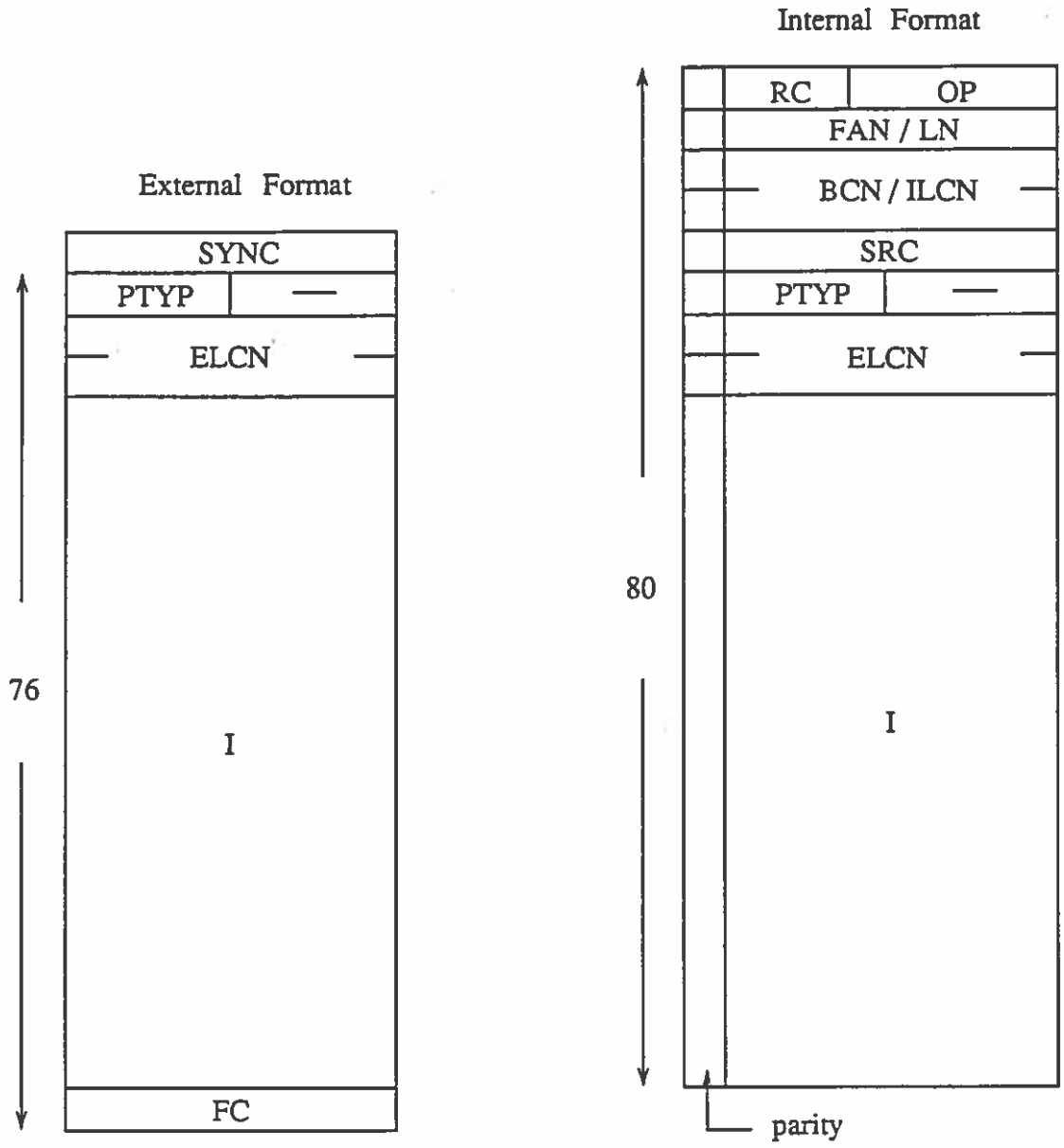

Figure 3. *Packet Formats*

```
                        /* external packet types */
#define DATA 1          /* ordinary data packet */
#define LTEST 2         /* link test packet */
#define CTL 4           /* control packet */

                        /* routing control types */
#define IDLE 0          /* unused packet slot */
#define PPNT 1          /* point to point packet */
#define MPNT 2          /* multipoint */
#define SPATH 4         /* specific path */
#define PENDING 7       /* waiting for logical channel translation */

                        /* internal packet op codes */
#define VANILLA 0       /* no control functions */
#define RLCXT 1         /* read LCXT entry */
#define WLCXT 2         /* write LCXT entry */
#define RBTT 3          /* read broadcast translation table entry */
#define WBTT 4          /* write broadcast translation table entry */
#define STEST1 5        /* switch test, first leg */
#define STEST2 6        /* switch test, second leg */
#define STEST3 7        /* switch test, third leg */

typedef struct {                    /* external packet */
        bit     fill[5];            /* unused */
        bit     ptyp[3];            /* packet type */
        bit     elcn[16];           /* external logical channel number */
        bit     info[72][8];        /* information field */
} ext_pkt;

typedef struct {                    /* internal packet */
        bit     op[5];              /* op field */
        bit     rc[3];              /* routing control field */
        bit     fan_ln[8];          /* fanout/ln */
        bit     bcn_ilcn[16];       /* bcn/lcn */
        bit     src[8];             /* source of packet */
        ext_pkt     extp;
} int_pkt;
```

Figure 4. *Packet Format Definitions*

1.3 PP3

Figure 5 is an illustration of PP3. Packets from the link side enter the chip at the RCB, pass through the OUT1 circuit, a delay line and the OUT2 circuit and then pass on to the switch fabric. When such a packet passes through the OUT1 circuit, its logical channel number is passed to the LCXT, which extracts the selected entry and delivers it to the OUT2 circuit in time to be inserted into the packet. The OUT1 and OUT2 circuits also cooperatively process switch test packets and LCXT read/write packets. Packets coming from the switch fabric pass through the INC circuit and are steered to either the STB or the XMB. Packets going to the XMB are stripped of the additional header information added when the packet was first received. The darker lines in the figure indicate the paths of data flow.

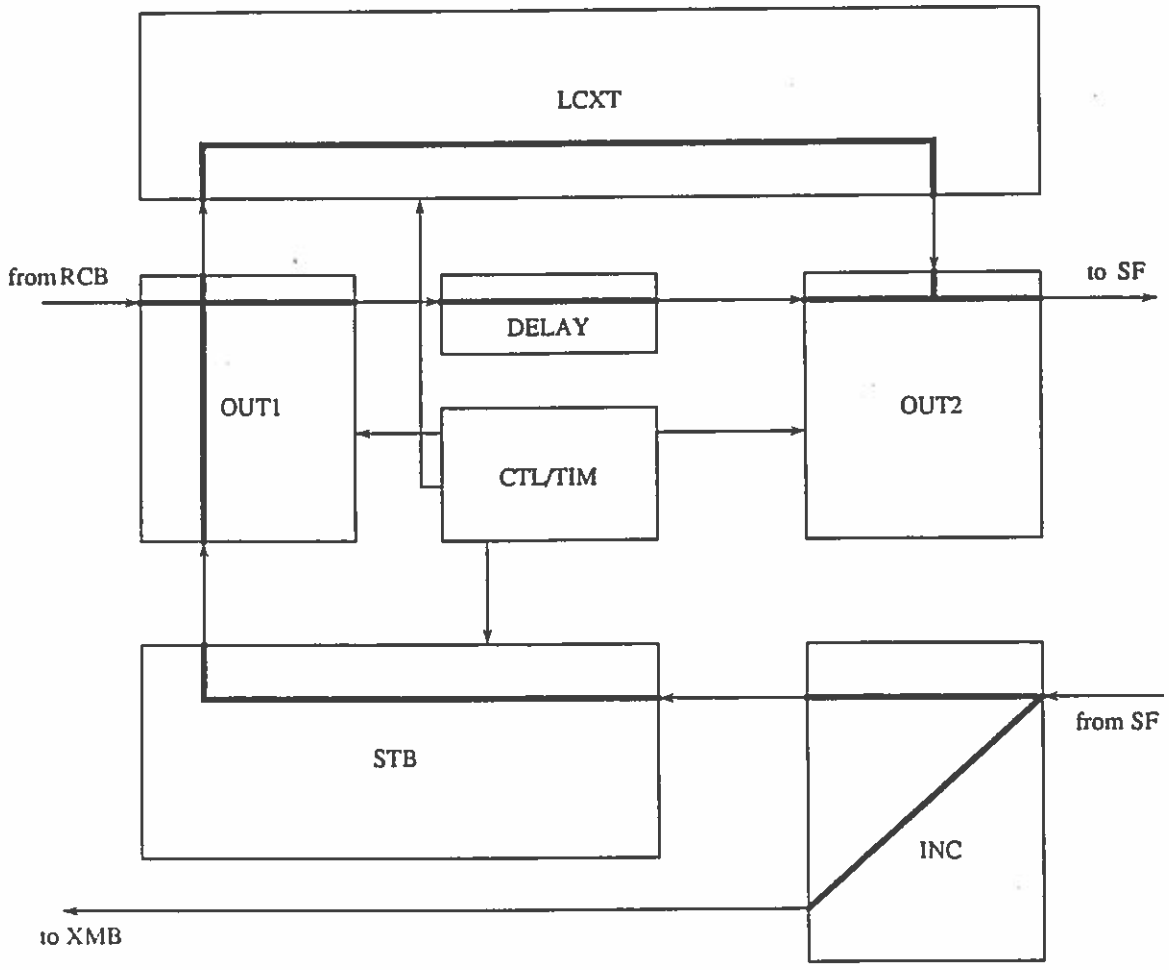

Figure 5. *PP3 Illustration*

2. Chip Specification

2.1 Operation

PP3 sends and receives packets, performs necessary format conversion and routes the packets according to their packet types. The types of packets that a PP3 chip expects to see are listed below.

Receiving at OUT1 :

- **Data Packet** Pass the logical channel number to LCXT, change the routing control field to PENDING, pass the packet in internal format through the delay line, insert the LCXT entry at OUT2 and send the packet on to SF.
- **Non-data Packet** Change the routing control field to point-to-point (PPNT) and the address field to 0, pass the packet in internal format through the delay line and OUT2, then send the packet on to SF.

Receiving at INC :

- **Vanilla** Copy the ILCN field to the ELCN field, strip off the header, and send the packet to the XMB in external format.
- **Read LCXT** Send the packet through INC, STB and OUT1 where the entry address is passed to the LCXT. The entry is then extracted by the LCXT and returned to the packet at OUT2.
- **Write LCXT** Process in the same way as Read LCXT packets, except the LCXT writes the given contents into the entry, allowing the LCXT to be updated.
- **Specific Path Test** Pass the packet through INC, STB, OUT1, the delay line and in OUT2, modify the packet's routing control field, the operation field and other relevant fields.

Any operation involving the participation of OUT1 assumes the presence of the grant signal from the down-stream SF (sf_dg). In absence of this grant signal, no packet will be accepted by OUT1.

External packets entering OUT1 when the sf_dg is not present will not be acknowledged and thus are required to be retransmitted. Internal non-data packets, entering PP3 through INC when the STB is full, are lost. Internal data packets, entering PP3 through INC when XMB is full, are also lost.

Packet acceptance is assumed only when no parity error is detected at various check points in PP3, otherwise the packet is discarded at the point where the error is detected.

The current implementation restricts the continuous use of STB. Successive packets entering the STB are required to be separated by at least two full packet cycles.

The entire circuit needs an initialization period at the beginning of operation, during which the LCXT completes its pre-charge cycle. All input or output data present during this period is treated as invalid.

The PP3 chip allows isolated operation of each of its functional blocks. Each functional block can be operated in two modes, normal (or watching) mode and test (or driving) mode. These blocks operate in their normal mode under most circumstances. Detailed discussion on this is found in a later section.

Three clocks are required for the operation of PP3. They are named $\phi 1$, $\phi 2$ and $\phi 3$. The three clocks satisfy the phase relations depicted in Figure 6. In addition, we require that data be stable during $\phi 1$ and may only be allowed to change during $\phi 2$.

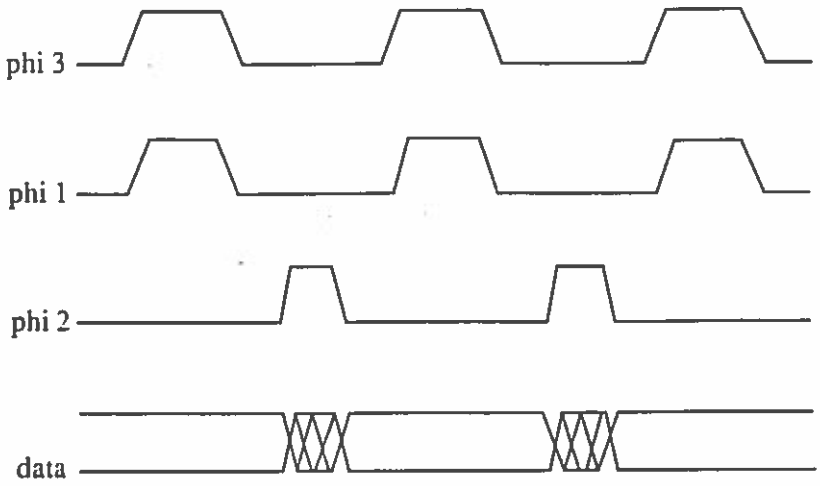

Figure 6. *Clock Phase Relationship*

2.2 PP3 Interface

The external interface signals of the PP3 chip are shown in Figure 7 and are described briefly below.

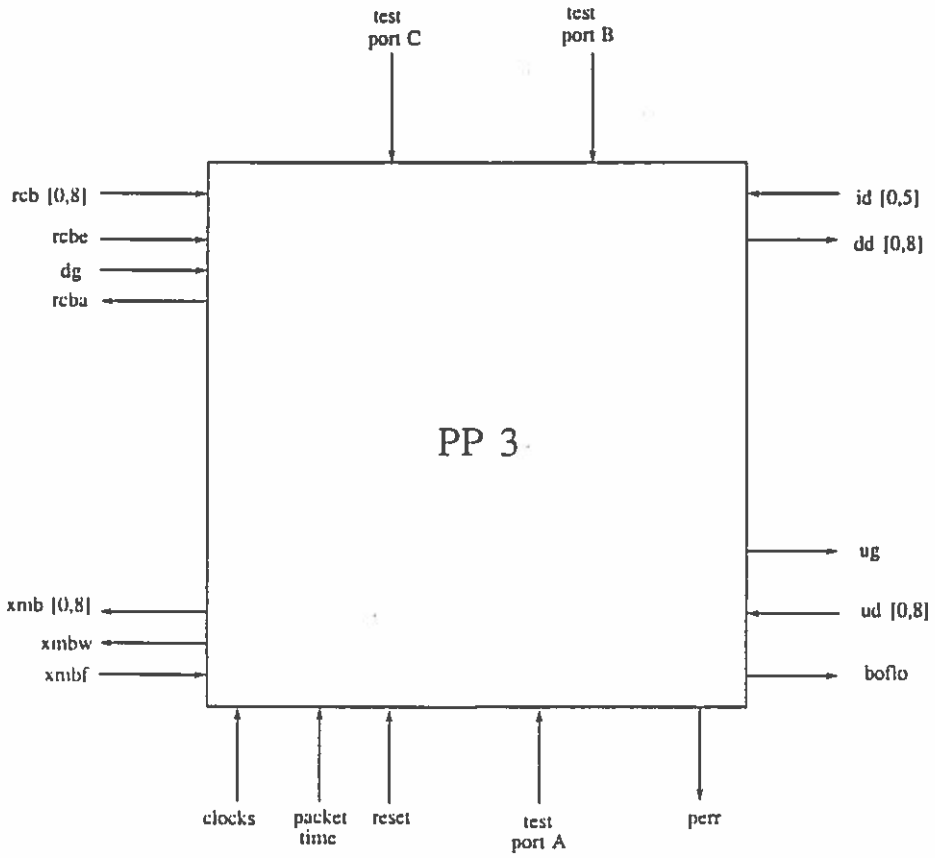

Figure 7. *PP3 Interface*

The following is a list of major PP3 interface signals and their description.

• *Receive Buffer Data* (rcb[0,8]).	Input data from the receive buffer, 9-bit wide with parity at bit-8.
• *Receive Buffer Empty* (rcbe).	Input signal indicating the status of the Receive Buffer.
• *Down Stream Grant* (dg).	Input grant signal from down-stream neighboring SF, indicating the permission to transmit a packet during the next packet cycle.
• *Receive Buffer Acknowledge* (rcba).	Output signal, acknowledging the receipt of a packet sent by the RCB.
• *Transmit Buffer Data* (xmb[0,8]).	Output signals, destined for the XMB from SF, 9-bit wide with parity at bit-8.
• *Transmit Buffer Write* (xmbw).	Output signal, indicating whether or not a packet is approaching the XMB and needs to be written by the XMB.
• *Transmit Buffer Full* (xmbf).	Input signal, indicating whether or not the XMB is full.
• *Clocks* (ϕ1, ϕ2, ϕ3).	Input signals, two-phase non-overlapping clocks ϕ1 and ϕ2. ϕ3 overlaps ϕ1 and is used in control and timing circuits.
• *Packet Time* (gtm2, itm2).	Input signals, signifies that the next global or internal packet cycle starts in 2 ticks.
• *Reset* (reset, srst).	Input signal, providing a hardware reset or a soft reset signal, the latter only resets the parity error.
• *Test Ports* (tA, tB, tC).	Bidirectional signals, providing the capability of either watching or driving a certain set of signals.
• *Parity Error* (perr).	Output signal, indicating whether or not a parity error has been detected.
• *Buffer Over Flow* (boflo).	Output signal, indicating overflow at either STB or XMB.
• *Up Stream Data* (ud[0,8]).	Input data from the SF, 9-bit wide with parity at bit-8.
• *Up Stream Grant* (ug).	Ouput signal to up stream neighboring SF, indicating permission to transmit a packet during the next packet cycle. It is tied high in this design.
• *Down Stream Data* (dd[0,8]).	Output data, destined for SF, 9-bit wide with parity at bit-8.
• *PP ID* (id[0,5]).	Input signals, allowing each PP to be identified with a 6-bit pattern.

Figure 8. *PP3 Interface Signals*

3. Design of Functional Blocks

The PP3 block diagram is shown in Figure 9. The major functional blocks include,

1) Input circuit (INC)
2) Switch test buffer (STB)
3) Output circuits (OUT1,OUT2)
4) Logical channel translation table (LCXT)
5) Delay line (DELAY)
6) Control/timing circuit (CTL/TIM)
7) Parity checker/generator (PCHK/PGEN)
8) Parity indicator
9) Tap (tA, tB, tC)

Each block operates with reference to its own packet cycle time. Although the packet cycles are all of the same length, they don't start and end at the same time. The start of a local packet cycle is signified by the presence of a *packet time* (or tm2) signal. Each local clock starts counting an 86-tick cycle two ticks after the presence of *tm2*. The CTL/TIM block is responsible for generating the *tm2* signals to each of the individual blocks, coordinating the operation in the entire chip. Figure 9 on the next page shows the PP3 block structure.

3.1 Input Circuit

The input circuit (INC) interfaces with the SF on one side and the XMB on the other. There are two types of packets it may expect to see coming from the SF end, data packets and non-data packets. Data packets are converted to external format and sent on to the XMB. Non-data packets are passed on to the STB. The buffer overflow flag is set when either the STB or the XMB is detected to be full. Writing to these buffers is only allowed when the involved buffer is not full and the parity flag indicates no error. The input and output of this block are tabulated as follows. The time column indicates when a signal is activated with respect to its local packet cycle. Local clocks run from tick 0 (t0) to tick 85 (t85). Note that the presence of the clock signals, the *tm2* signal, power and ground are assumed and are thus omitted in the table.

Signal	Direction	Width	Time	Description
sf_d	input	8	t0	data from SF
stb_f	input	1	t0	STB full flag
ipe	input	1	t80	parity error
xmb_d	output	8	t8	data to XMB
xmb_w	output	1	t85	XMB write signal
stb_d	output	8	t4	data to STB
stb_w	output	1	t85	STB write signal
boflo	output	1	t80	buffer overflow

Table 1. *INC Input and Output*

INC is generated with the *Synchronous Streams Processor (ssp)* Circuit Generator [Ro88]. The SSP Circuit Generator takes a high-level circuit description and generates the circuit at mask level. The INC circuit description is given on the page that follows. (Refer to [Ro88] for syntax and conventions used in this specification language.)

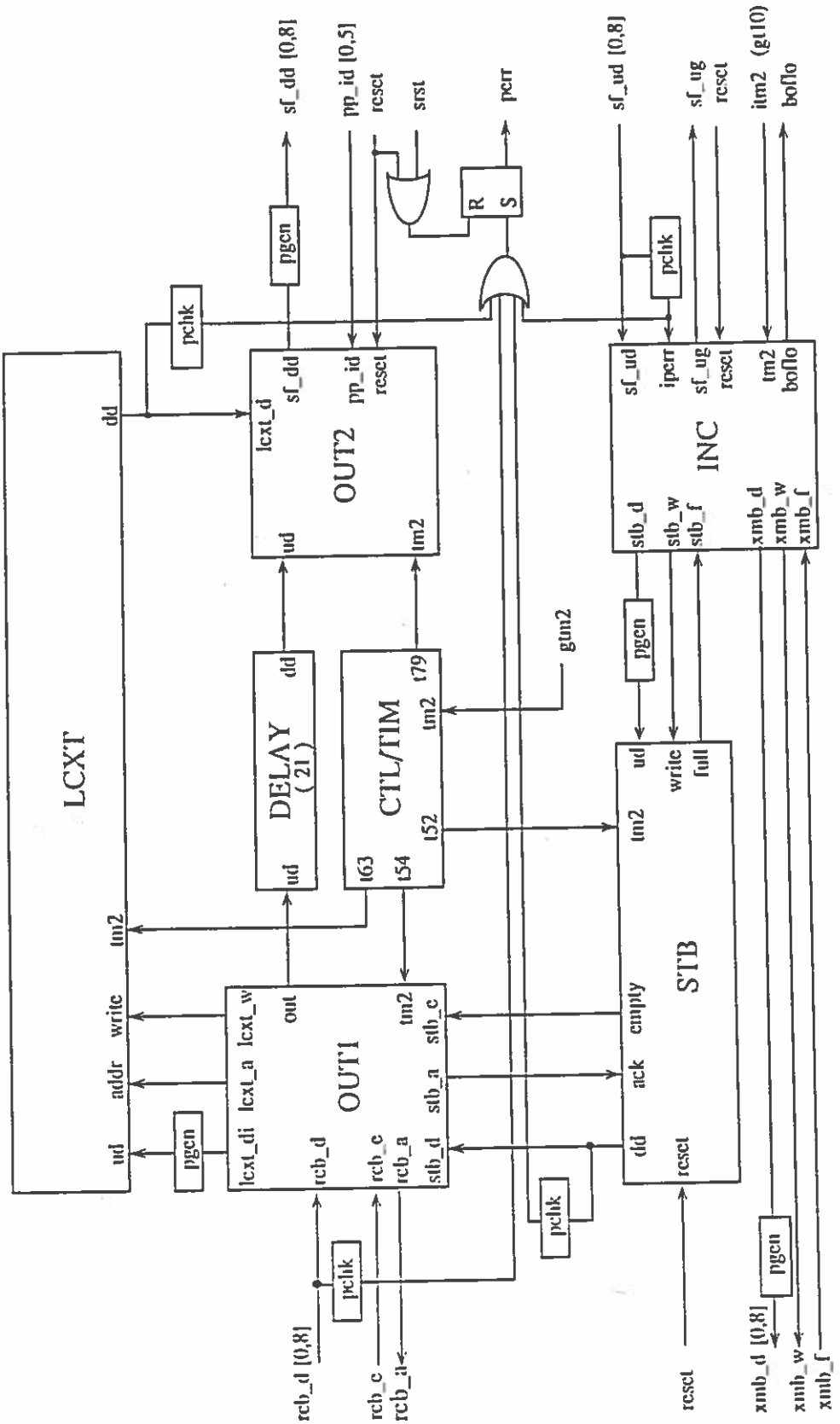

Figure 9. *PP3 Block Diagram*

```
/* Synchronous streams processor definition for the input circuit used
/* within the PP3 chip.
/*
/* Inc interfaces to the switch fabric, the stb and xmb and input
/* parity checker. On each cycle it expects a packet to arrive from
/* the switch fabric and it steers the packet to the appropriate port.
/* For xmb packets, it strips the extra header information.
/* Packets with errors are not written to buffers.
*/

#include "pktfmt.sih"

inc@86( port[8] int_pkt          <sf_d@0;        /* data from fabric */

        port[8] ext_pkt          >xmb_d@8;       /* data to xmb */
        port[1] bit[1]           >xmb_w@85;      /* xmb write signal */
        port[1] bit[1]           <xmb_f@0;       /* xmb full */

        port[8] int_pkt          >stb_d@4;       /* data to stb */
        port[1] bit[1]           >stb_w@85;      /* stb write signal */
        port[1] bit[1]           <stb_f@0;       /* stb full */

        port[1] bit[1]           >boflo@80;      /* buffer overflow */
        port[1] bit[1]           <ipe@80)        /* parity error */
{
        stb_w = 0;
        xmb_w = 0;
        boflo = 0;

        if sf_d.rc != IDLE & sf_d.op == VANILLA ->
                xmb_d = sf_d.extp;
                xmb_d.elcn = sf_d.bcn_ilcn;
                if xmb_f == 1 ->
                        boflo = 1;
                |  xmb_f == 0 & ipe == 0 ->
                        xmb_w = 1;
                fi;
         | sf_d.rc != IDLE & sf_d.op != VANILLA ->
                stb_d = sf_d;
                if stb_f == 1 ->
                        boflo = 1;
                |  stb_f == 0 & ipe == 0 ->
                        stb_w = 1;
                fi;
        fi;
}
```

Figure 10. *INC SSP Circuit Description*

3.2 Switch Test Buffer

The switch test buffer (STB) resides in between the INC and the OUT1. The purpose of doing this is two-fold. One, packets destined for OUT1 may not be able to enter OUT1 during a particular packet cycle, because downstream SF doesn't permit OUT1 to send packet at current time. Secondly, INC and OUT1 have different local packet times. Packets coming out of INC are therefore buffered until, at least, they are synchronized to OUT1's packet time. The block diagram of STB appears in Figure 11.

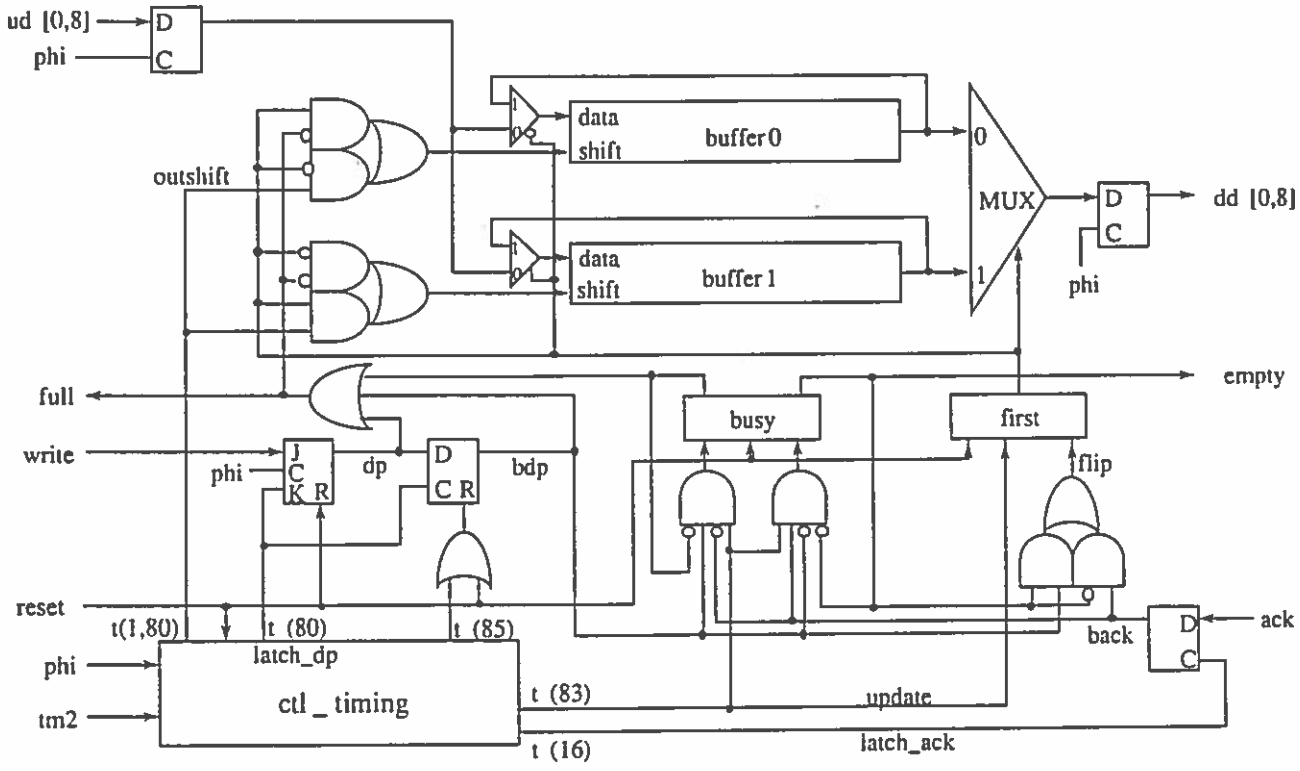

Figure 11. *STB Block Diagram*

There are a number of components in STB. Most importantly are the two packet buffers, each capable of holding one 80-byte long packet. The buffers are implemented with a chain of shift register cells. A feedback path connects the output of a buffer back to its input end, allowing a packet not permitted to exit during the current cycle to recirculate in the buffer. The output multiplexer selects alternately between the two buffers. The Control/Timing block provides the timing signals to all the blocks in STB.

The block BUSY contains mainly a two-bit binary counter. The counter is incremented when a new packet is accepted at STB and decremented when a packet leaves STB. The *busy* signal is generated when the output of the counter reaches 2 which is the maximum capacity of STB, and the *empty* signal is asserted when the output indicates a zero.

The block FIRST determines which buffer is to shift out data during the next packet cycle. The output of FIRST is toggled if one of the following condition is satisfied.

1) buffers are currently empty and a new packet is arriving,
2) a packet currently occupying a buffer has been transmitted and has been acknowledged.

The current implementation of the STB deviates from this schematic slightly. In the current implementation, only one buffer (buffer 1) is being used. The output mux is tied to high allowing output only from buffer 1. Accordingly, the *busy* signal from BUSY is taken to be the complement of *empty*, i.e. *busy* is asserted as soon as buffer 1 is taken. Figure 12 shows this change.

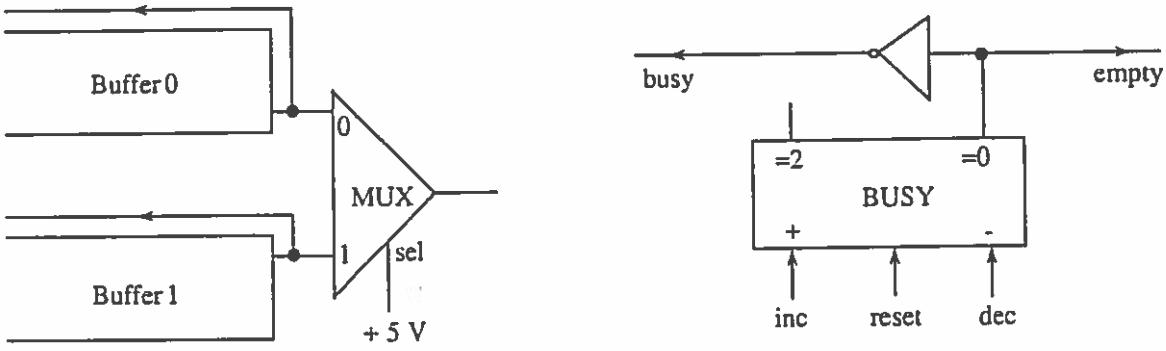

Figure 12. *STB Modifications*

A table of input and output of STB is provided below.

Signal	Direction	Width	Time	Description
ud	input	9	t(ud)	data coming from INC
write	input	1	t(ud)+80	write to STB from INC
ack	input	1	t10	acknowledgement from OUT1
dd	output	9	t2	data going to OUT1
full	output	1	vary	STB buffer full flag
empty	output	1	t84	STB buffer empty flag

Table 2. *STB Input and Output*

3.3 Output Circuits

The output circuit is divided into two parts, OUT1 and OUT2. This division is necessary due to the limitations in the current implementation of the *SSP* Circuit Generator.

OUT1

OUT1 receives packets from the RCB or the STB. If the packet coming from the RCB is a data packet, OUT1 passes the logical channel number to the LCXT and sends the rest of the packet on through the delay line. If the packet is a non-data packet, the packet is sent on to the delay line with a few modifications. For packets coming from the STB, regardless of their types, OUT1 routes them to the delay line without any change. The OUT1 inputs and outputs are listed below.

Signal	Direction	Width	Time	Description
rcb_d	input	8	t0	data coming from RCB
rcb_e	input	1	t0	RCB empty flag
stb_d	input	8	t0	data coming from STB
stb_e	input	1	t0	STB empty flag
sf_dg	input	1	t0	grant from down-stream SF
out	output	8	t4	data sent to delay line
rcb_a	output	1	t8	acknowledge RCB packets
stb_a	output	1	t8	acknowledge STB packets
lcxt_a	output	8	t13	LCXT address, only 6 bits used
lcxt_di	output	8	t12	LCXT data
lcxt_w	output	1	t6	write to LCXT

Table 3. *OUT1 Input and Output*

The *ssp* specification for the OUT1 circuit appears on the next page.

OUT2

OUT2 accepts packets from the delay line, originally sent out by OUT1. In the default mode, OUT2 copies the incoming packet to the outgoing packet, adding the processor id number to the source field (src). If the packet was a data packet originally from RCB, OUT2 replaces its first 4 bytes with the entry LCXT has delivered. If the packet is to read an entry out of the LCXT, OUT2 sticks the contents of the entry in the designated field. For specific path test packets, OUT2 updates the related fields and passes them along.

Signal	Direction	Width	Time	Description
ud	input	8	t0	data coming from the delay line
lcxt_do	input	8	t0	entry LCXT delivers
pp_id	input	8	t0	processor id, 6 bits used
dd	output	8	t4	data exiting PP3

Table 4. *OUT2 Input and Output*

The *ssp* specification for OUT2 is shown on the page that follows.

```
/* Synchronous streams processor definition for the out1 circuit used
/* within the PP3 chip.
/*
/* Out1 interfaces to the rcb, ltb, lcxt and a delay line. It also
/* receives the grant from the switch fabric. On each cycle it expects
/* packets to arrive from the two buffers and a grant signal to be
/* present at time 0.
*/

#include "pktfmt.sih"

out1@86(port[8] ext_pkt          <rcb_d@0;
        port[1] bit              <rcb_e@0;
        port[1] bit[16]          >rcb_a@8;

        port[8] int_pkt          <stb_d@0;
        port[1] bit              <stb_e@0;
        port[1] bit[16]          >stb_a@8;

        port[8] int_pkt          >out@4;

        port[8] bit[8]           >lcxt_a@13;
        port[8] bit[4][8]        >lcxt_di@12;
        port[1] bit[16]          >lcxt_w@6;

        port[1] bit              <sf_dg@0)
{
        out.rc = 0;
        rcb_a = 0;
        stb_a = 0;

        lcxt_w = 0;
        lcxt_a = stb_d.extp.info[0];
        lcxt_di:(0..31) = stb_d.extp.info:(8..39);

        if sf_dg == 1 & rcb_e == 0 & stb_e == 1 ->
                out.extp = rcb_d;
                rcb_a = 0xffff;
                if rcb_d.ptyp == DATA ->         /* start logical channel */
                        out.rc = PENDING;        /* translation           */
                        lcxt_a = rcb_d.elcn:(0..7);
                 | rcb_d.ptyp != DATA ->         /* route non-data packets */
                        out.rc = PPNT;           /* to CP.                */
                        out.fan_ln = 0;
                fi;
        | sf_dg == 1 & stb_e == 0 ->
                stb_a = 0xffff;
                if stb_d.op == RLCXT ->          /* start lcxt read */
                        out = stb_d;
                 | stb_d.op == WLCXT ->          /* write lcxt */
                        lcxt_w = 0xffff;
                 | stb_d.op == STEST1 | stb_d.op == STEST2 ->
                        out = stb_d;             /* out2 does rest */
                fi;
        fi;
}
```

Figure 13. *OUT1 SSP Circuit Description*

```
/* Synchronous streams processor definition for the out2 circuit used
/* within the PP3 chip.
/*
/* Out2 interfaces to an upstream delay line, the lcxt and the
/* switch fabric. It also has a port connected to the pins that
/* give the pp's number. On each cycle it expects a packet to
/* arrive from the delay line and if needed, data from the lcxt.
*/

#include "pktfmt.sih"

out2@86(port[8] int_pkt          <ud@0;          /* upstream data */
        port[8] bit[4][8]        <lcxt_do@0;     /* info from lcxt */
        port[8] bit[8]           <pp_id@0;       /* pp number */
        port[8] int_pkt          >dd@4)          /* downstream data */
{
        dd = ud;
        dd.src = pp_id;
        if ud.rc == PENDING ->
                dd:(0..31) = lcxt_do:(0..31);
         | ud.op == RLCXT ->
                dd.extp.info:(8..39) = lcxt_do:(0..31);
         | ud.op == STEST1 ->            /* rotate bytes 1-4 by 1 */
                dd.rc = SPATH;
                dd.op = STEST2;
                dd:(8..31) = ud:(16..39);
                dd:(32..39) = ud:(8..15);
         | ud.op == STEST2 ->            /* return it to CP */
                dd.rc = PPNT;
                dd.op = STEST3;
                dd.fan_ln = 0;
        fi;
}
```

Figure 14. *OUT2 SSP Circuit Description*

3.4 Logical Channel Translation Table

The logical channel translation table (LCXT) is a memory component with 64 entries. LCXT allows both reading from or writing to a location identified by a 6-bit address. Writing and reading are done through the WLCXT and RLCXT control packets. The signal *r/w* is set to low when read, and high when write. LCXT requires a pre-charge cycle before its normal operation.

Signal	Direction	Width	Time	Description
in	input	9	t4	data to be written into
a	input	6	t4	6-bit entry address
out	output	9	t16	data to be read out

Table 5. *LCXT Input and Output*

3.5 Delay Line

The delay line (DELAY) is a chain of 21 shift register cells. Packets passing through DELAY will experience a delay of 21 ticks. The reason a delay line is placed in between OUT1 and OUT2 is that when an external data packet enters PP3, it needs to find an entry from the LCXT. The delay allows processing time required by the LCXT to fetch an entry. By the time the head of the packet exits the delay line and reaches OUT2, the entry from LCXT will be ready, waiting at OUT2.

3.6 Control/Timing Circuit

The control/timing (CTL/TIM) is the heart of the entire PP3. It provides the packet cycle start-signals to all the other blocks, maintaining the overall synchronization. The CTL/TIM circuit is mainly a 7-bit counter, counting from 0 to 85 in a wrap-around fashion. Timing signals are sent to other blocks when the counter output reaches some pre-determined values.

The CTL/TIM circuit is once again implemented using the *ssp* Circuit Generator facility. The specification appears below. Four timing signals are being specified here. They are activated at ticks 52, 54, 63 and 79 in an 86-tick packet cycle.

```
condinps C0
regouts L0
muxouts t52 t54 t63 t79
cntsize 7
cntmax 86
cntena  - 86 <MAX>
sigset  - 52 t52
sigrst  - 53 t52
sigset  - 54 t54
sigrst  - 55 t54
sigset  - 63 t63
sigrst  - 64 t63
sigset  - 79 t79
sigrst  - 80 t79
```

Figure 15. *CTL/TIM Specification*

3.7 Other Circuits

There are many small circuits involved in PP3. These include, clock drivers, parity generator, parity checker, parity indicator, taps, etc. Here we talk about the design of two of the small circuits, the parity indicator and the taps.

Parity Indicator

The parity indicator provides a mechanism for reporting a parity error. The indicator can be cleared upon a hardware reset or a soft reset. The schematic is shown as follows. It consists of a J/K flip flop and two OR gates, providing the J and K signals. The indicator is set when at least one of the 4 parity check points reports an error. The indicator stays high until a hardware reset or a soft reset is issued to clear it.

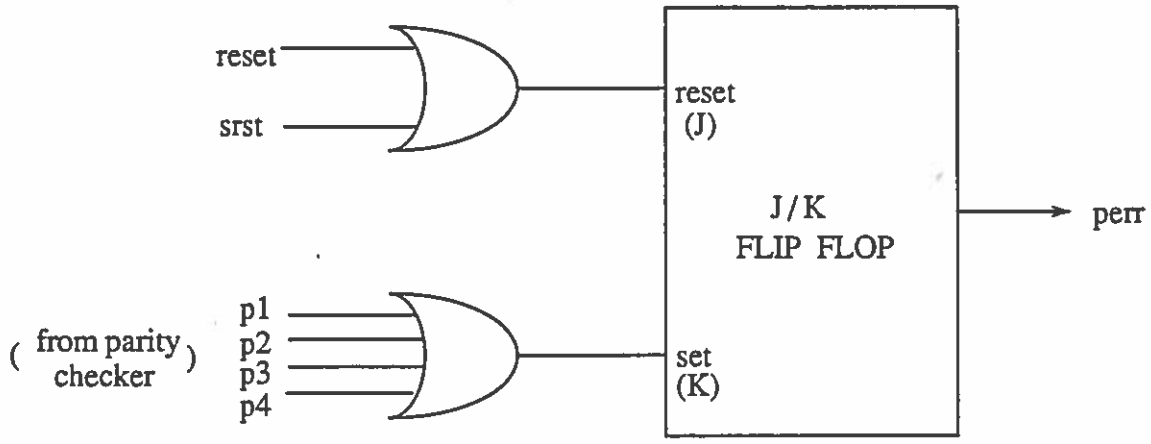

Figure 16. *Parity Indicator*

The structure of the J/K flip flop used in the parity indicator is shown below. The J/K flip flop is 2-phase clocked. Changes at J or K are only latched in during $\phi 1$ and Q is only updated during $\phi 2$. A transition table is also provided.

J	K	Q
0	0	previous state
0	1	1 (set)
1	-	0 (clear)

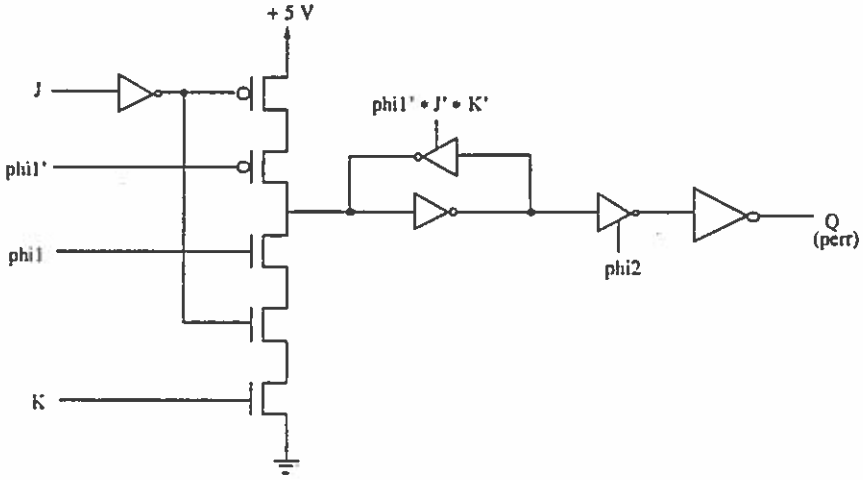

Figure 17. *J/K Flip-flop in Parity Indicator*

Taps

Taps are circuits attached to a data line or a group of data lines allowing the designer to either monitor or drive the line(s). A tap has two operation modes, the watching mode and the driving mode. In the watching mode, a tap simply sends a copy of what passes through the line without altering the data. This provides the designer with the capability of monitoring some critical internal signals that are normally no accessible from the package pins. In the driving mode, a tap attached to a line breaks the line, stops the upstream data and supplies the downstream data with its own. This effectively allows the downstream circuit to be able to operate independently. In case of upstream circuit failure, the downstream circuit can remain operational if data is supplied through these taps. The structure of a tap is shown below.

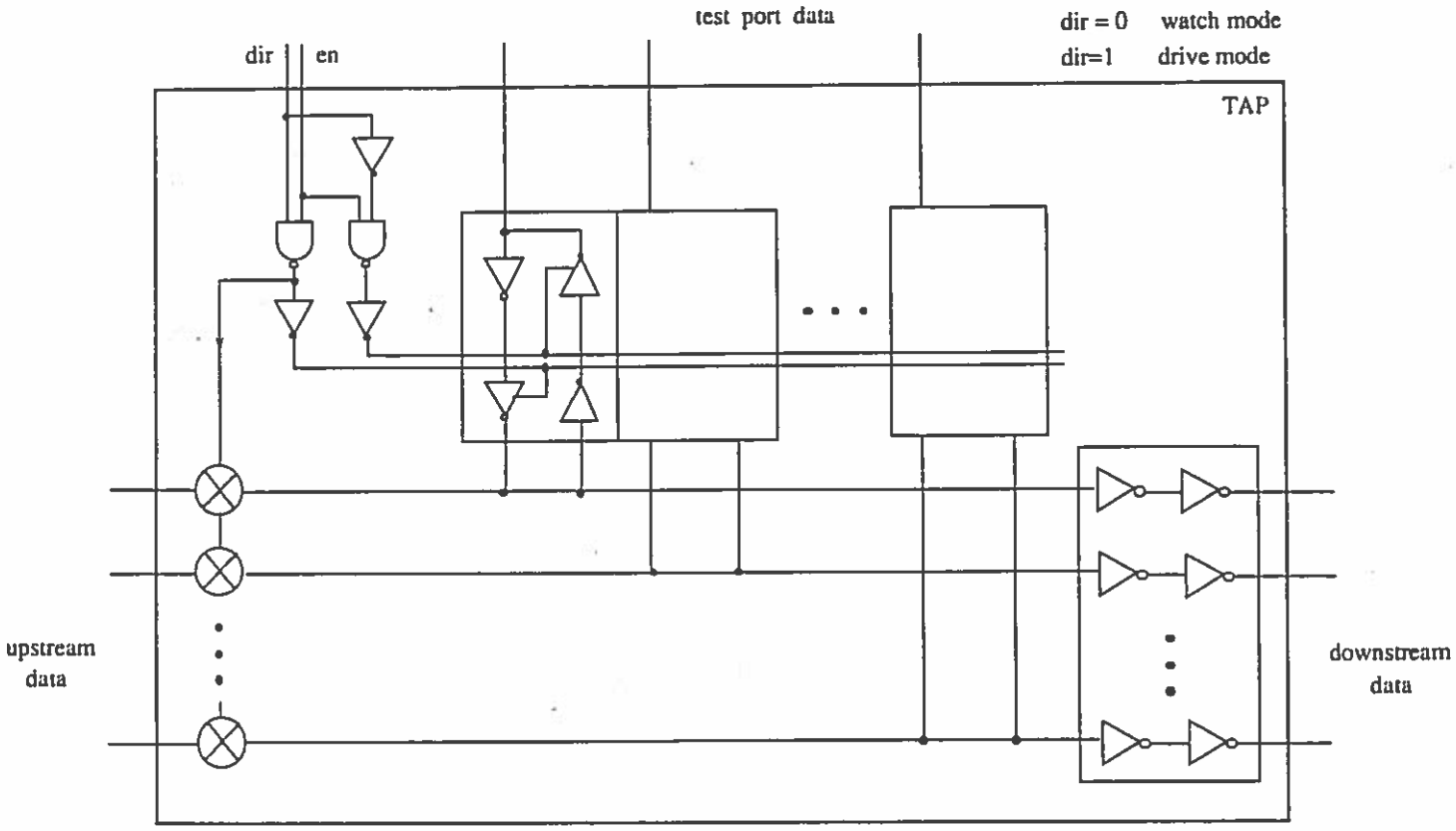

Figure 18. *Structure of a Tap*

A tap is enabled when the *EN* signal is asserted. It is in the watching mode if the direction bit is set to 0 and driving mode if set to 1. A tap allows a maximum of 10 data lines to be tapped at a time.

In PP3, there are 3 groups of taps, named tA, tB and tC. Each group consists of 4 taps, tapping at different points in the circuit. In within each group, only one can be activated at a time. This is done through a 2-to-4 decoder. Each tap group provides its decoder with two select bits, ts1 and ts0. The output of the decoder then selects the tap in the group and make it enabled. The signal *tin* provides the direction a tap needs. Note however, the tap direction bit is taken to be the inversion of the *tin*. For example, if in group A, we set *tin* = 0, *ts1* = 1, and *ts0* = 0, then tap 2 is enabled and the direction is set to 1 (*tin'*), indicating the driving mode. Taps are useful for testing and debugging purposes. In Figure 19 on the next page, we show how the signals in PP3 are tapped.

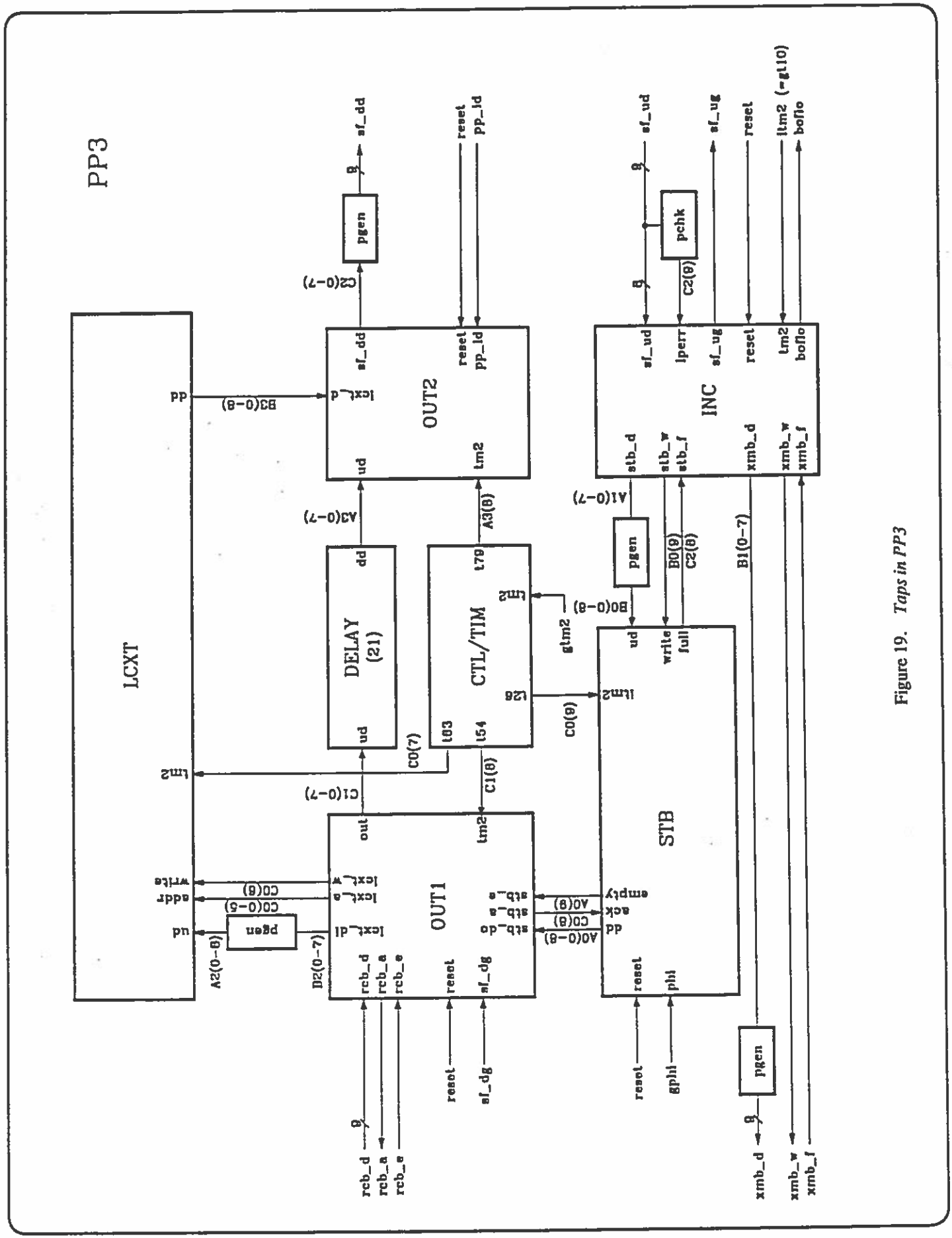

Figure 19. *Taps in PP3*

4. Implementation

4.1 Integrated Circuit Design

The PP3 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, a graphical layout editor capable of supporting a hierarchical cell structure. Technology based design rules are checked by Magic and these are based on a Manhattan design style with λ=1.0 µm. The minimum line width is 2 µm. The chip fabrication technology is an n-well CMOS process, facilities being provided by MOSIS. The chip contains a total of approximately 48,000 transistors.

The diagram below shows the layout of the major functional blocks viewed from the topmost cell. Notice the placement of taps among the functional blocks

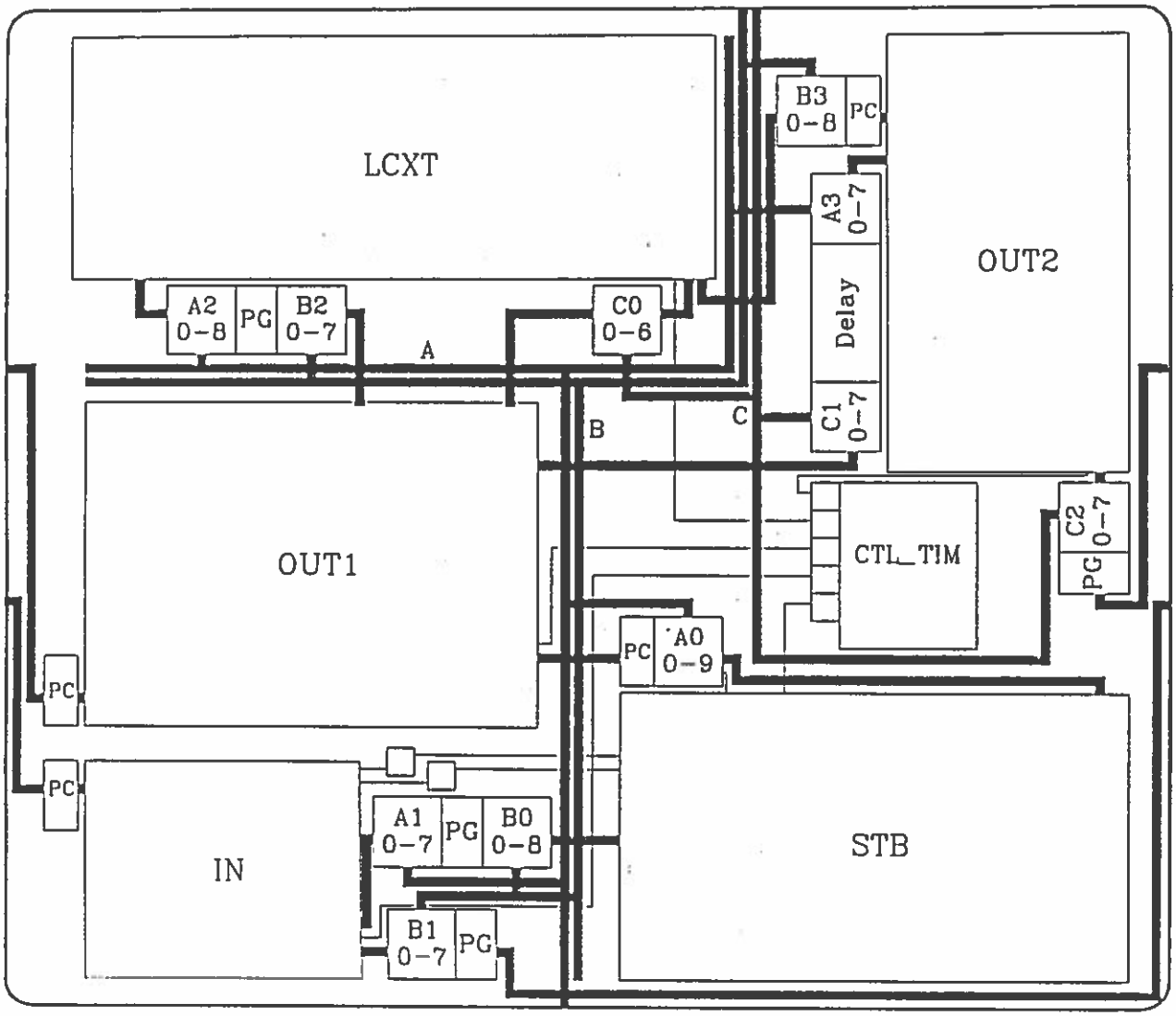

Figure 20. *PP3 Block Level Layout*

4.2 Circuit Simulation

The simulation tools used include ESIM [ScMa86] and FACTS [MCNC]. Esim provides a logic simulator while FACTS performs timing analyses in a nice graphical form. Each packet is 80-bytes long and the inter-packet separation is 6-bytes long. Most of these logic simulation require input with 4 or more packets. Consequently, a simulation input file has to contain a large number of vectors. VESIM is used in this case to provide a front-end interface to ESIM, allowing test vectors to be displayed in a vertical format merely for visual convenience. Due to the computational intensity of the simulation, FACTS is used to simulate circuits only up to the functional block level. Clock rates of 42 MHz are used in all the timing simulations.

Each functional block was simulated for correct operation after it had been designed. The PP3 highest-level cell was then simulated with the confidence that all sub-blocks work properly. In simulating the highest-level cell, a set of 8 input files were used to test the circuit, providing a variety of conditions and different types of packets. The layout had passed all of the eight tests before it was submitted for fabrication. These eight tests have provided a sufficient variety of cases that PP3 is expected to deal with. I believe that an exhaustive test is neither practical nor necessary. These eight tests are listed below.

Test Name	Path	Packet Type	Description
ppnt	OUT1-DELAY-OUT2	non-data (ext)	process RCB nod-data packet
pend	OUT1-DELAY/LCXT-OUT2	data (ext)	process RCB data packet
wrlcxt	INC-STB-OUT1-DELAY/LCXT-OUT2	wlcxt/rlcxt (int)	access LCXT entries
xmb	INC	vanilla (int)	send packet from SF out to XMB
stest1	INC-STB-OUT1-DELAY-OUT2	stest1 (int)	process specific path test packet
stest2	INC-STB-OUT1-DELAY-OUT2	stest2 (int)	process specific path test packet
stb	STB (isolated)	(int)	test STB isolated operation
drvc2	OUT2		drive a tap

Table 6. *PP3 Simulation*

Taps are also included in these simulations. Every tap is made sure to have been simulated at least once, either watched or driven. If a tap works in one mode, one can be convinced that the direction of tap works fine and also the tap bus can deliver data properly and conclude that this tap works. The following table lists how each tap is tested. (D indicates that a tap is tested in its driving mode and W in its watching mode).

Test Name	A0	A1	A2	A3	B0	B1	B2	B3	C0	C1	C2	C3	tin ts1 ts0 (A B C)
ppnt	W	-	-	-	W	-	-	-	-	W	-	unused	100 100 101
pend	-	W	-	-	-	-	-	W	W	-	-	unused	101 111 100
wrlcxt	-	-	W	-	-	-	W	-	W	-	-	unused	110 110 100
xmb	-	-	-	-	-	W	-	-	-	-	-	unused	1-- 101 1--
stest1	-	-	-	W	D	-	-	-	-	W	-	unused	111 000 101
stest2	-	-	-	W	D	-	-	-	-	-	W	unused	111 000 110
drvc2	-	-	-	-	-	-	-	-	-	-	D	unused	1-- 1-- 010

Table 7. *Testing the Taps*

4.3 Packaging and Pin Assignment

The PP3 chip is implemented in a 108 pin-grid-array package. The size of the chip is 7900 mm × 9200 mm. Figure 21 below shows the pin arrangement as viewed from top of the package. A full description of the pins and the signals follows.

	M	L	K	J	H	G	F	E	D	C	B	A
1	GND (82)	tCs1 (81)	tCs0 (80)	tC8 (77)	tC5 (74)	tC2 (71)	tB9 (68)	tB6 (65)	tB3 (62)	tB0 (59)	tBs1 (57)	GND (55)
2	rcbe (84)	tCin (83)	tC9 (79)	tC7 (76)	tC4 (73)	tC1 (70)	tB8 (67)	tB5 (64)	tB2 (61)	tBin (58)	tBs0 (56)	boflo (54)
3	rcb2 (86)	rcb1 (85)	rcb0 (78)	tC6 (75)	tC3 (72)	tC0 (69)	tB7 (66)	tB4 (63)	tB1 (60)	dd2 (51)	dd1 (52)	dd0 (53)
4	rcb5 (89)	rcb4 (88)	rcb3 (87)							dd4 (48)	dd4 (49)	dd3 (50)
5	rcb8 (92)	rcb7 (91)	rcb6 (90)							dd8 (45)	dd7 (46)	dd6 (47)
6	rcba (95)	gnd (94)	vdd (93)							gnd (42)	vdd (43)	dg (44)
7	ug (98)	vdd (97)	gnd (96)							vdd (39)	gnd (40)	xmbf (41)
8	ud6 (101)	ud7 (100)	ud8 (99)							xmb6 (36)	xmb7 (37)	xmb8 (38)
9	ud3 (104)	ud4 (103)	ud5 (102)							xmb3 (33)	xmb4 (34)	xmb5 (35)
10	ud0 (107)	ud1 (106)	ud2 (105)	itm2 (6)	tA0 (9)	tA3 (12)	tA6 (15)	tA9 (18)	tAin (21)	xmb0 (24)	xmb1 (31)	xmb2 (32)
11	perr (108)	phi1 (2)	phi3 (4)	reset (7)	tA1 (10)	tA4 (13)	tA7 (16)	tAs0 (19)	id0 (22)	id2 (25)	id5 (29)	xmbw (30)
12	sub (1)	phi2 (3)	gtm2 (5)	srst (8)	tA2 (11)	tA5 (14)	tA8 (17)	tAs1 (20)	id1 (23)	id3 (26)	id4 (27)	Vdd (28)
	M	L	K	J	H	G	F	E	D	C	B	A

Figure 21. *PP3 Pin Arrangement*

Pin	Signal	Direction	Description
1	sub	-	substrate
2	phi1	input	clock phase 1
3	phi2	input	clock phase 2
4	phi3	input	clock phase 3
5	gtm2	input	global packet time
6	itm2	input	internal packet time
7	reset	input	hardware reset
8	srst	input	soft reset
9	tA0	i/o	tap A, data bit 0
10	tA1	i/o	tap A, data bit 1
11	tA2	i/o	tap A, data bit 2
12	tA3	i/o	tap A, data bit 3
13	tA4	i/o	tap A, data bit 4
14	tA5	i/o	tap A, data bit 5
15	tA6	i/o	tap A, data bit 6
16	tA7	i/o	tap A, data bit 7
17	tA8	i/o	tap A, data bit 8
18	tA9	i/o	tap A, data bit 9
19	tAs0	input	tap A, select bit 0
20	tAs1	input	tap A, select bit 1
21	tAin	input	tap A, direction
22	id0	input	processor id bit 0
23	id1	input	processor id bit 1
24	xmb0	output	data to xmb, bit 0
25	id2	input	processor id bit 2
26	id3	input	processor id bit 3
27	id4	input	processor id bit 4
28	Vdd	input	power
29	id5	input	processor id bit 5
30	xmbw	output	write to xmb signal
31	xmb1	output	data to xmb, bit 1
32	xmb2	output	data to xmb, bit 2
33	xmb3	output	data to xmb, bit 3
34	xmb4	output	data to xmb, bit 4
35	xmb5	output	data to xmb, bit 5
36	xmb6	output	data to xmb, bit 6

Pin	Signal	Direction	Description
37	xmb7	output	data to xmb, bit 7
38	xmb8	output	data to xmb, bit 8
39	Vdd	input	power
40	GND	input	ground
41	xmbf	input	xmb full flag
42	GND	input	ground
43	Vdd	input	power
44	dg	input	down stream grant
45	dd8	output	down stream data bit 8
46	dd7	output	down stream data bit 7
47	dd6	output	down stream data bit 6
48	dd5	output	down stream data bit 5
49	dd4	output	down stream data bit 4
50	dd3	output	down stream data bit 3
51	dd2	output	down stream data bit 2
52	dd1	output	down stream data bit 1
53	dd0	output	down stream data bit 0
54	boflo	output	stb or xmb overflow flag
55	GND	input	ground
56	tBs0	input	tap B, select bit 0
57	tBs1	input	tap B, select bit 1
58	tBin	input	tap B, direction
59	tB0	i/o	tap B, data bit 0
60	tB1	i/o	tap B, data bit 1
61	tB2	i/o	tap B, data bit 2
62	tB3	i/o	tap B, data bit 3
63	tB4	i/o	tap B, data bit 4
64	tB5	i/o	tap B, data bit 5
65	tB6	i/o	tap B, data bit 6
66	tB7	i/o	tap B, data bit 7
67	tB8	i/o	tap B, data bit 8
68	tB9	i/o	tap B, data bit 9
69	tC0	i/o	tap C, data bit 0
70	tC1	i/o	tap C, data bit 1
71	tC2	i/o	tap C, data bit 2
72	tC3	i/o	tap C, data bit 3

Pin	Signal	Direction	Description
73	tC4	i/o	tap C, data bit 4
74	tC5	i/o	tap C, data bit 5
75	tC6	i/o	tap C, data bit 6
76	tC7	i/o	tap C, data bit 7
77	tC8	i/o	tap C, data bit 8
78	rcb0	input	data from rcb, bit 0
79	tC9	i/o	tap C, data bit 9
80	tCs0	input	tap C, select bit 0
81	tCs1	input	tap C, select bit 1
82	GND	input	ground
83	tCin	input	tap C, direction
84	rcbe	input	rcb empty flag
85	rcb1	input	data from rcb, bit 1
86	rcb2	input	data from rcb, bit 2
87	rcb3	input	data from rcb, bit 3
88	rcb4	input	data from rcb, bit 4
89	rcb5	input	data from rcb, bit 5
90	rcb6	input	data from rcb, bit 6
91	rcb7	input	data from rcb, bit 7
92	rcb8	input	data from rcb, bit 8
93	Vdd	input	power
94	GND	input	ground
95	rcba	output	rcb data acknowledge
96	GND	input	ground
97	Vdd	input	power
98	ug	output	up stream grant
99	ud8	input	up stream data, bit 8
100	ud7	input	up stream data, bit 7
101	ud6	input	up stream data, bit 6
102	ud5	input	up stream data, bit 5
103	ud4	input	up stream data, bit 4
104	ud3	input	up stream data, bit 3
105	ud2	input	up stream data, bit 2
106	ud1	input	up stream data, bit 1
107	ud0	input	up stream data, bit 0
108	perr	output	parity error

References

[Tu88] Turner, Jonathan S., *Advanced Communications System*, annual progress report, WUCS-88-28, Dept. of Computer Science, Washington University, St. Louis, MO. 1988.

[Ro88] Robbert, George., *A Circuit Generator for Synchronous Stream Processors*, MS thesis, Dept. of Computer Science, Washington University, St. Louis, MO. 1988.

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

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

[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, NC. 1985.*