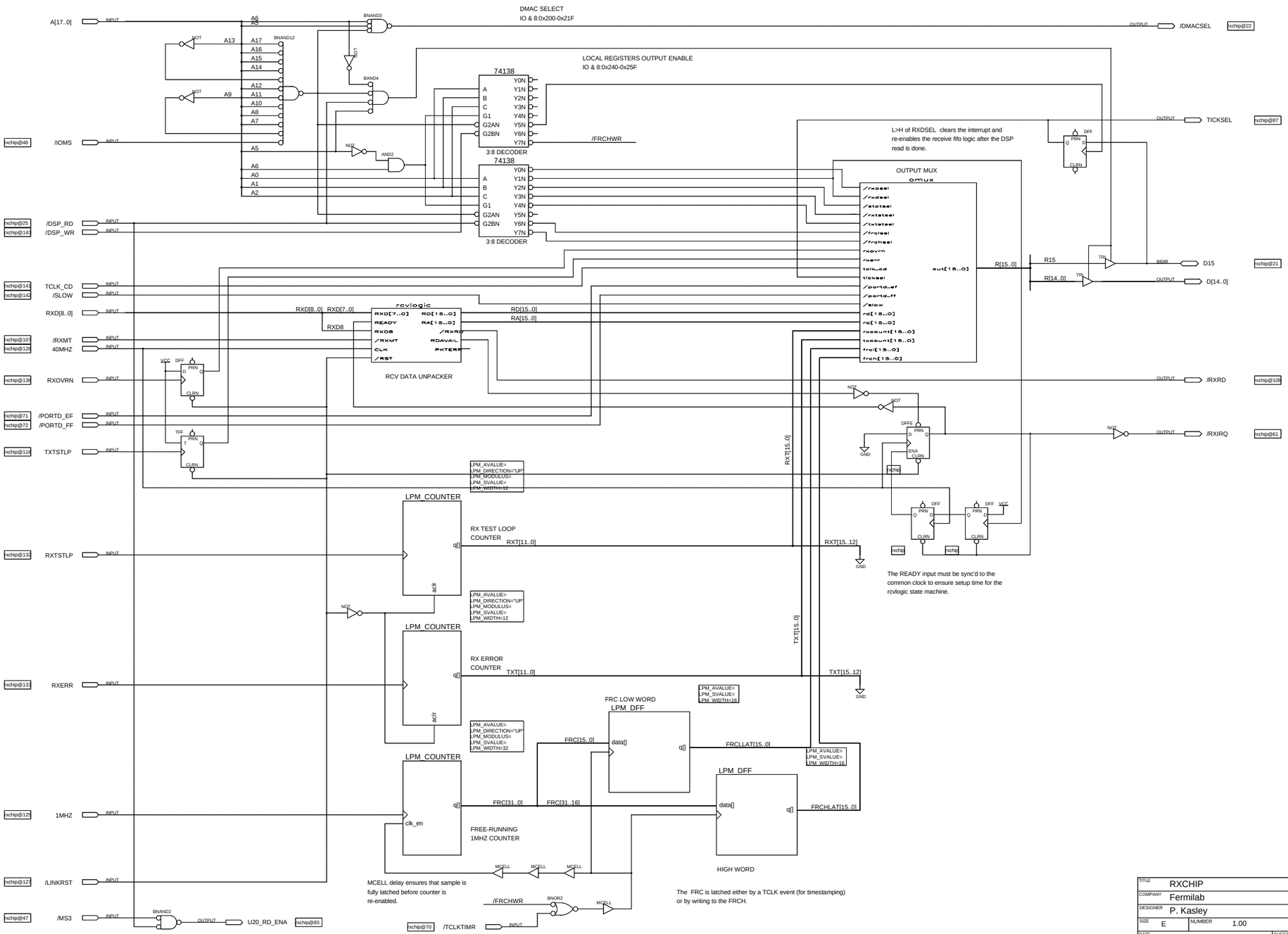
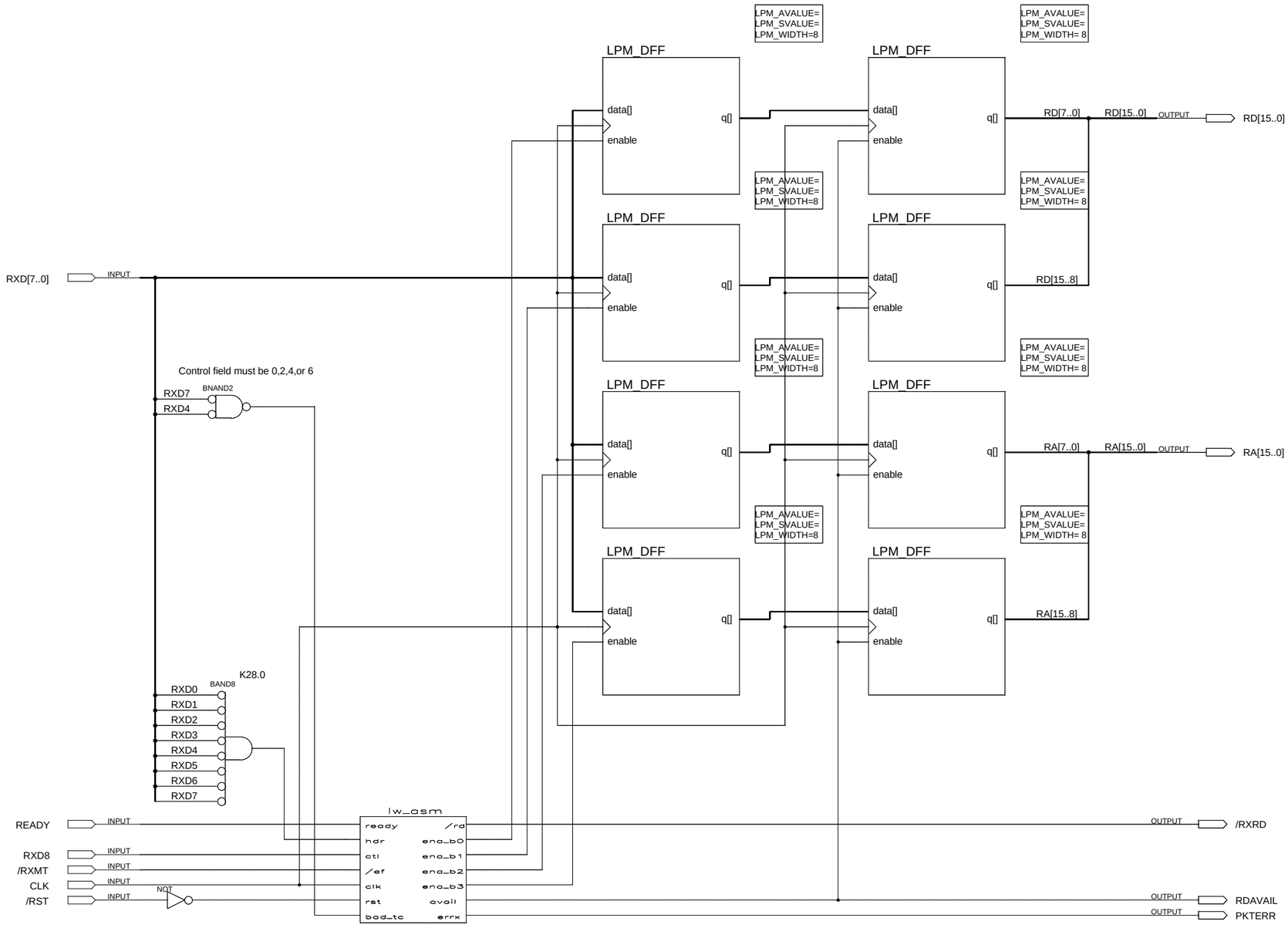


TITLE	RXCHIP
COMPANY	Fermilab
DESIGNER	P. Kasley
SIZE E	NUMBER 1.00 REV A
DATE 2:04p 9-29-2005	SHEET 1 OF 1



Scan data from the receiver FIFO until a packet header is found.
When a header is found, load each of the next four bytes from the FIFO into registers.
When the registers are full, signal Received Data Available and wait for the packet to be "consumed".
After the packet has been consumed, scan for the next header.
If an error occurs, signal Packet Error, discard the current data and begin scanning for a header.

TITLE	RCVLOGIC.GDF		
COMPANY	Fermilab		
DESIGNER	P. Kasley		
SIZE D	NUMBER 1.00		REV A
DATE 3:54p 10-22-2002		SHEET 1	OF 1

```
%*****************************************************************************
DAS-CENTRAL Hot Link Receiver Long Word Assembly
22 Oct 02 pak

Read the RX FIFO until a header is found. Retrieve the next four bytes from the
RX FIFO to a 32-bit register. Signal external logic that a frame is ready for
processing and wait for it to be accepted. Search for next header.

If header is found where data is expected, signal an error, discard the current
frame, and build a new frame. If control is found where data is expected, signal
error, discard the current frame, and search for a new header.
*****************************************************************************%

TITLE "Long Word Assembly";

SUBDESIGN lw_asm
(ready      : INPUT;        % DSP read the output buffer/ buffer is empty %
 hdr        : INPUT;        % Decode of FIFO data indicates start of data frame %
 ctl        : INPUT;        % SC/D* (control symbol received) %
 /ef        : INPUT;        % RX FIFO Empty flag %
 clk        : INPUT;        % clock %
 rst        : INPUT;        % reset %
 bad_tc     : INPUT;        % Invalid type code %
 /rd        : OUTPUT;       % RX FIFO read enable %
 ena_b0     : OUTPUT;       % Enable byte 0 latch (D7:0) %
 ena_b1     : OUTPUT;       % Enable byte 1 latch (D15:8) %
 ena_b2     : OUTPUT;       % Enable byte 2 latch (D23:16) %
 ena_b3     : OUTPUT;       % Enable byte 3 latch (D31:24) %
 avail      : OUTPUT;       % New frame available for DSP. %
 errx       : OUTPUT;       % Error flag %
)

VARIABLE
lw_asm: MACHINE OF BITS (q[3..0])
    WITH STATES (   wait0=H"0",
                    read0=H"1",
                    wait1=H"9",
                    read1=H"3",
                    wait2=H"b",
                    read2=H"2",
                    wait3=H"a",
                    read3=H"6",
                    wait4=H"e",
                    read4=H"7",
                    wait5=H"f",
                    xfrrd0=H"5",
                    xfr=H"4",
                    void3=H"c",
                    void4=H"d",
                    err=H"8");

BEGIN

% State machine clock and reset%
lw_asm.clk=clk;
lw_asm.reset=rst;

% Outputs %

% Wrap the MT flag back to the read input. Disable reading during these states. %
/rd = !/ef # err # void3 # void4 # wait5 # read4;

ena_b0=read1;
ena_b1=read2;
ena_b2=read3;
```

```
ena_b3=read4;
errx=err;
avail=xfr # xfrrd0;

% state transitions %
TABLE
    lw_asm,/ef,hdr,ctl,ready,bad_tc => lw_asm;

%   PS      /EF HDR CTL RDY BAD_TC  =>  NS %

%  WAIT0: idle %
    wait0,  0,  x,  x,  x,      x       =>  wait0; % Idle, waiting for data %
    wait0,  1,  x,  x,  x,      x       =>  read0; % FIFO Data %

%  READ0: pull from FIFO until frame header is found %
    read0,  0,  0,  x,  x,      x       =>  wait0; % Header not found, no further data %
    read0,  1,  0,  x,  x,      x       =>  read0; % Header not found, FIFO has data %
    read0,  0,  1,  1,  x,      x       =>  wait1; % Header found but no further data %
    read0,  1,  1,  1,  x,      x       =>  read1; % Header found, FIFO has data %

%  WAIT1: wait for LSB0 data %
    wait1,  0,  x,  x,  x,      x       =>  wait1; % Waiting for data %
    wait1,  1,  x,  x,  x,      x       =>  read1; % FIFO Data %

%  READ1: latch LSB0 data %
    read1,  0,  x,  x,  x,      x       =>  wait2; % Waiting further data %
    read1,  1,  x,  0,  x,      x       =>  read2; % FIFO has LSB1 %
    read1,  x,  x,  1,  x,      x       =>  err; % ERROR: Expected data, found control %

%  WAIT2: wait for LSB1 data %
    wait2,  0,  x,  x,  x,      x       =>  wait2; % Waiting for data %
    wait2,  1,  x,  x,  x,      x   =>  read2; % FIFO Data %

%  READ2: latch LSB1 data %
    read2,  0,  x,  x,  x,      x       =>  wait3; % Waiting further data %
    read2,  1,  x,  0,  x,      x       =>  read3; % FIFO has MSB0 %
    read2,  x,  x,  1,  x,      x       =>  err; % ERROR: Expected data, found control %

%  WAIT3: wait for MSB0 data %
    wait3,  0,  x,  x,  x,      x       =>  wait3; % Waiting for data %
    wait3,  1,  x,  x,  x,      x       =>  read3; % FIFO Data %

%  READ3: latch MSB0 data %
    read3,  0,  x,  x,  x,      x       =>  wait4; % Waiting further data %
    read3,  1,  x,  0,  x,      x       =>  read4; % FIFO has MSB1 %
    read3,  x,  x,  1,  x,      x       =>  err; % ERROR: Expected data, found control %

%  WAIT4: wait for MSB1 data %
    wait4,  0,  x,  x,  x,      x       =>  wait4; % Waiting for data %
    wait4,  1,  x,  x,  x,      x       =>  read4; % FIFO Data %

%   PS      /EF HDR CTL RDY BAD_TC  =>  NS %
%  READ4: latch MSB1 data %
    read4,  0,  x,  0,  1,      0       =>  xfr; % FIFO MT. Transfer frame to buffer and
idle %
    read4,  0,  x,  0,  0,      0       =>  wait5; % FIFO MT but memory busy. Wait. %
    read4,  1,  x,  0,  1,      0       =>  xfrrd0; % FIFO not MT and memory idle. Get to
work on next frame. %
    read4,  1,  x,  0,  0,      0       =>  wait5; % FIFO not MT but memory busy. Wait. %
    read4,  x,  x,  1,  x,      x       =>  err; % ERROR: Expected data, found control %
    read4,  x,  x,  0,  x,      1       =>  err; % ERROR: Bad typecode %

%  XFR: transfer from receiving distributor to holding register %
    xfr,    0,  x,  x,  x,      x       =>  wait0; % nothing in FIFO. Go to idle. %
```

```
    xfr,    1,  x,  x,  x,      x       =>  read0; % New data in FIFO. Go to search for
header. %

%  XFRRD0: transfer from receiving distributor to holding register and
    start into next frame %
    xfrrd0,0,   0,  x,  x,      x       =>  wait0; % Header not found, no further data %
    xfrrd0,0,   1,  0,  x,      x       =>  wait0; % Header not found, no further data %
    xfrrd0,0,   1,  1,  x,      x       =>  wait1; % Header found but no further data %
    xfrrd0,1,   0,  x,  x,      x       =>  read0; % Header not found, FIFO has data %
    xfrrd0,1,   1,  0,  x,      x       =>  read0; % Header not found, FIFO has data %
    xfrrd0,1,   1,  1,  x,      x       =>  read1; % Header found, FIFO has data %

%  WAIT5: wait for frame to be stored or discard invalid frame %
    wait5,  x,  x,  x,  0,      x       =>  wait5;
    wait5,  x,  x,  x,  1,      x       =>  xfr;

%  ERR: dummy state to make error output %
    err,    x,  x,  x,  x,      x       =>  wait0;

%  VOID: illegal states %
    void3,  x,  x,  x,  x,      x       =>  wait0;
    void4,  x,  x,  x,  x,      x       =>  wait0;

END TABLE;

END;
```

```
%*****************************************************************************
Rxchip output mux    3FEB03 PAK
*****************************************************************************%

TITLE "Rxchip Output Mux";

SUBDESIGN omux
(/rxasel    : INPUT;        % Receive adrs select %
 /rxdsel    : INPUT;        % Receive data select %
 /statsel   : INPUT;        % Status select %
 /rxtstsel  : INPUT;        % Rx test loop counter select %
 /txtstsel  : INPUT;        % Tx test loop counter select %
 /frclsel   : INPUT;        % Free running counter low select %
 /frchsel   : INPUT;        % Free running counter hi select %
 rxovrn     : INPUT;        % rx overrun %
 rxerr      : INPUT;        % rx error %
 tclk_cd    : INPUT;        % TeV clock carrier detect %
 ticksel    : INPUT;        % Read-back of frame tick select flop%
 /portd_ef  : INPUT;        % Port D FIFO empty flag %
 /portd_ff  : INPUT;        % Port D FIFO full flag %
 /slow      : INPUT;        % hot-link cable speed select %
 rd[15..0]  : INPUT;        % data word %
 ra[15..0]  : INPUT;        % address word %
 rxcount[15..0]: INPUT;     % rx count register %
 txcount[15..0]: INPUT;     % tx count register %
 frcl[15..0]: INPUT;        % free running counter low register %
 frch[15..0]: INPUT;        % free running counter high register %
 out[15..0] : OUTPUT;       % output bus %
)

BEGIN

IF (/rxasel == 0) THEN out[]=ra[]; END IF;
IF (/rxdsel == 0) THEN out[]=rd[]; END IF;
IF (/rxtstsel == 0) THEN out[]=rxcount[]; END IF;
IF (/txtstsel == 0) THEN out[]=txcount[]; END IF;
IF (/frclsel == 0) THEN out[]=frcl[]; END IF;
IF (/frchsel == 0) THEN out[]=frch[]; END IF;
IF (/statsel == 0) THEN
    out[0]=rxerr;
    out[1]=rxovrn;
    out[2]=/slow;
    out[3]=tclk_cd;
    out[15]=ticksel;
    out[4]=!/portd_ef;
    out[5]=!/portd_ff;
    out[14..6]=B"0";
    END IF;
END;
```