Using Actel FPGAs to Implement the 100 Mbit/s Ethernet Standard

One of the more recent entrants into the high-speed networking standards battle is 100Base-X—Ethernet operating at 100 Mbit/s. This standard is supported by the Fast Ethernet Alliance and sponsored by several key networking companies such as Intel, National Semiconductor, Sun Microsystems, and 3Com. This proposed standard involves many of the types of digital logic functions facing high-speed network designers and, as will be shown in this application note, can be readily implemented using Actel FPGA devices.

The emerging 100 Mbit Ethernet market is expected to mushroom as network performance requirements continue to grow. Network users are expected to have almost doubled between 1991 and 1994, and networks will need to provide these new users with just as much (if not more) bandwidth. A high-speed Ethernet network could solve the bandwidth problems for many classes of users while maintaining compatibility with current equipment and software.

100Base-X Network Standards

The 100Base-X proposal uses two established networking standards to support the 100 Mbit data rate required to implement the tenfold increase in the 10 Mbit rate of the current Ethernet standard. The 100Base-X standard keeps the Media Access Control (MAC) layer the same as the current Ethernet standard, but it raises the data rate to 100 Mbit/s. Since the MAC layer was defined independently of performance level, this increase can be accomplished relatively easily, and the well-proven behavioral dynamics of the Ethernet MAC can be retained. The only change required is to reduce the physical network span to 1/10 of the 10 Mbit/s distance, resulting in a span of about 250 meters.

This reduced span fits well within current structured wiring methodologies. Building-floor wiring in modern installations of Ethernet, such as 10Base-T, are organized as physical stars with a centralized wiring closet and cable runs of less than 100 meters. For LANs, this results in a hub-station architecture with interconnections of less than 100 meters.

At the physical layer, 100Base-X leverages off the proven FDDI standard for 100 Mbit/s communications using a full-duplex 125 Mbit/s Physical Media-Dependent (PMD) sublayer. This supports fiber optic, shielded twisted-pair (STP) and unshielded twisted-pair (UTP) wiring. Combining the MAC layer of Ethernet to the PMD layer of FDDI requires a convergence sublayer (CS) between them. Using the CS, 100Base-X maps the PMD's constant signaling system to the packet-oriented half-duplex system imposed by the Ethernet MAC.

Convergence Sublayer Interfaces

The MAC transmits data to the convergence sublayer in the form of 4-bit words (Figure 1). This data is then encoded into 5-bit groups, serialized, and transmitted by the CS to the PMD sublayer as the transmitPMD signal.

Received data is sent from the PMD to the CS as the receivePMD signal and is synchronized with the 125 MHz clock. Note that the PMD also generates signalDetect when data is detected on the line. The CS decodes the serial data, converting the input 5-bit code groups into 4-bit hex characters and sends it to the MAC as the receiveMAC signal. Note that the PMD extracts the clock from the serial bit stream input. The 125 MHz frequency is recovered from the input data stream by the PMD clock circuits in the CS. In addition, receiveError is generated by the CS to indicate to the MAC that an error has occurred during reception. The carrierSense signal is provided to the MAC to indicate that the line is active. The collisionDetect signal notifies the MAC if a collision has occurred.

This application note will show you how to use Actel FPGAs to develop a complete convergence sublayer. It will subdivide the CS into its functional divisions and will show you how each can be implemented using Actel ACT 3 FPGAs.

Convergence Sublayer Functions

Figure 2 shows the basic data flow in the convergence sublayer. The CS receives transmit data from the MAC as 4-bit words designated transmitMAC. These 4-bit words are encoded into 5-bit symbols (designated TxSYM) that are shifted out to the PMD at the 125 MHz clock rate.

Received data at a 125 Mbit/s rate is sent from the PMD to the CS as the receivePMD signal. The CS formats input data to produce 5-bit symbol groups. Detection of the two-symbol sequence, J and K, marks the beginning of a packet and starts the synchronization of the input data stream. The 5-bit groups are then decoded by the 5B4B decoder and sent to the MAC as a stream of 4-bit words until the packet's end is detected by the reception of the end-of-packet delimiter characters, T and R.

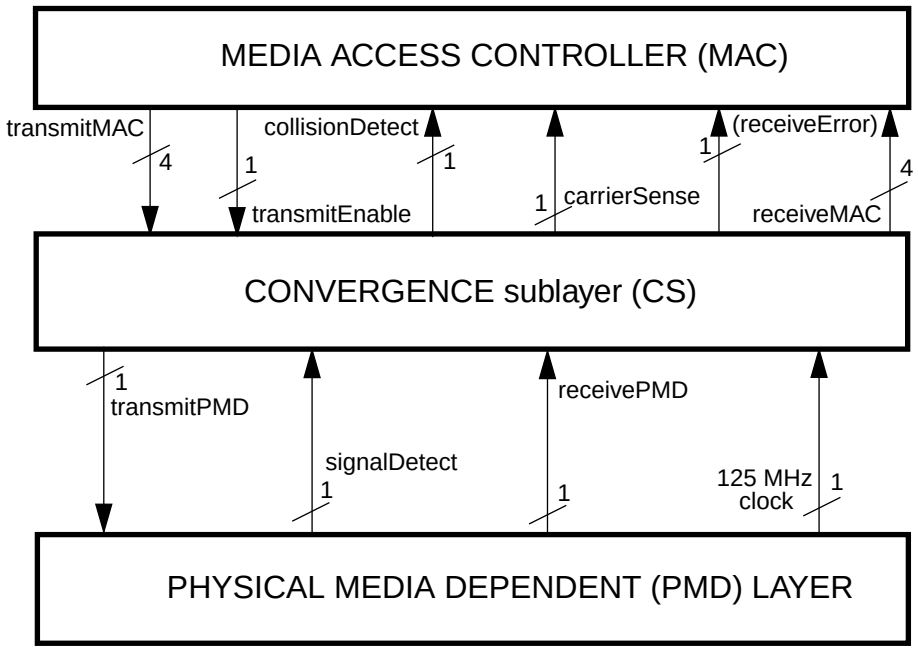

Figure 1 • *Convergence Sublayer Interfaces*

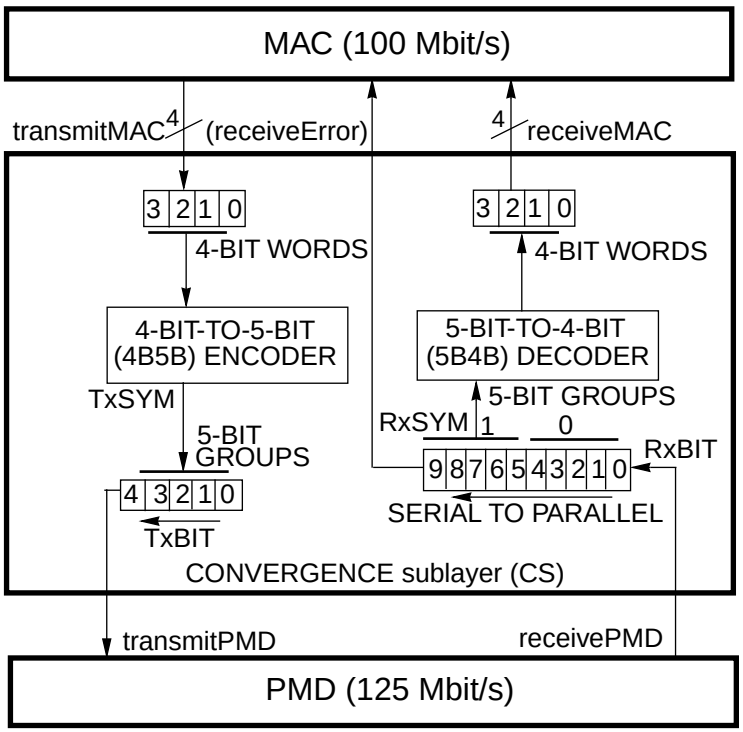

Figure 2 • *Data Flow in Convergence Sublayer*

The 4B5B encoding/decoding method, which is a subset of the standard FDDI 4B5B encoding method, employs 5-bits to encode/decode both 16 data (hex) characters and the signaling symbols required to indicate the start and end of the data packet. In addition to the 16 valid 4-bit-binary code groups shown in Table 1, there are five special control signals used to indicate start of packet (J followed by K), end of packet (T followed by R), and idle (I). A number of other 5-bit combinations are designated as invalid and represent channel errors or repeater collision artifacts. Thus, the physical line idles until the start of a data packet is indicated by a J symbol followed by a K. Data symbols then follow with the end of data being indicated by a T symbol followed by an R. Idle symbols immediately follow. The job of the convergence sublayer is to extract the control characters or idles from the packet and then send a data-only packet to the MAC. Thus, the MAC never receives idle, JK, or TR symbols. When receiving, the CS reverses the process, encapsulating and encoding the data from the MAC for transmission by the PMD.

Convergence Sublayer Data Flow

The block diagram of the convergence sublayer is shown in Figure 3. The receive state machine generates receiveMAC data and receiveError for the MAC based on the receivePMD data input from the media. The transmit state machine accepts transmitMAC and transmitEnable from the MAC and generates the transmitPMD data to the physical layer. The collisionDetect function is generated by the transmit state machine, based on transmitEnable and the receive state machine's receiving signal.

The Carrier Sense function asserts the carrierSense signal when the convergence sublayer is either transmitting or receiving, based upon the two corresponding internal signals generated by the Transmit and Receive functions. The Link Monitor function generates linkTestFail based on the PMDs signalDetect. LinkTestFail is an internal signal unused by the MAC and can optionally be used by your network management entity.

Transmitted data, shown as the transmitMAC signal in Figure 3, indicates that MAC data is available and is registered in the convergence sublayer logic. Groups of 4 data bits in the transmit bit stream are converted to 5-bit code groups by 4B5B encoding prior to transmission on the 125 Mbit/s PMD. The TxDATA, TxSYM, and TxBIT signals are all different views of the same data, but at different data rates.

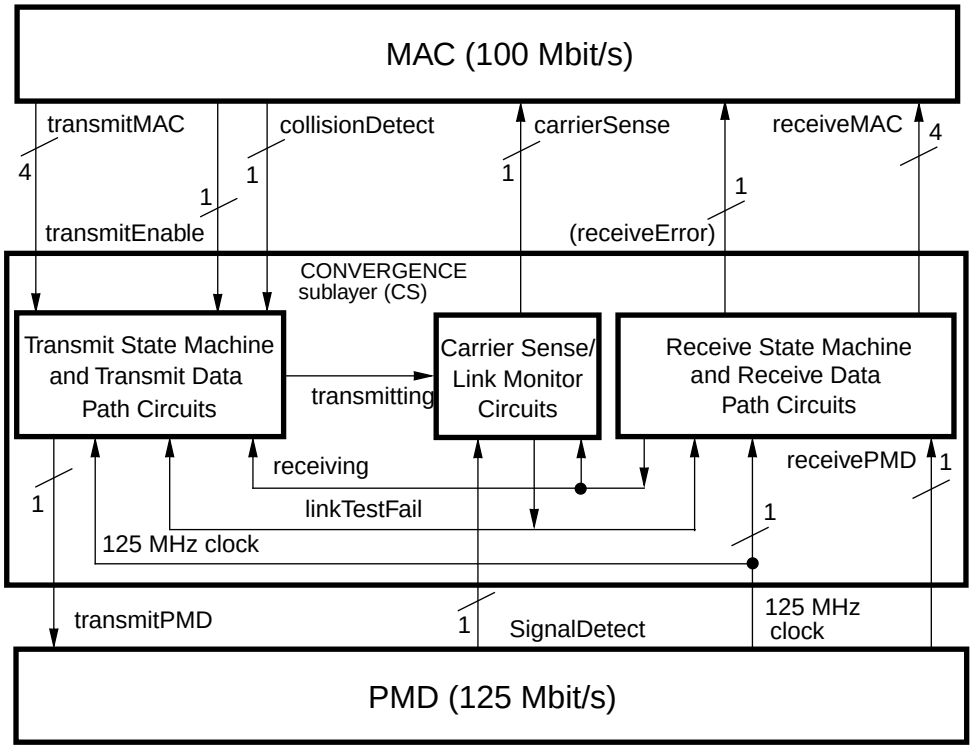

Figure 3 • *Convergence Sublayer Functional Block Diagram*

Using FPGAs to Implement a 100Base-X Convergence Sublayer

As will be seen in the following sections, the 100Base-X convergence sublayer can be implemented as eight functional blocks, each of which forms the subject of a separate application discussion. These are listed under the three main headings: the transmit function, the receive function, and the carrier-sense and link-monitor circuits.

Convergence Sublayer Transmit Function

The design of the transmit function shows some common design techniques used in high-speed FPGA applications. The main function of this block is to provide the requested symbol data to the PMD at the 125 Mbit/s serial rate. This requires the 4-bit MAC data words to be 4B5B encoded and then shifted out using the 125 MHz clock. In addition, the serial data stream needs to be framed using the leading /J/K/ symbol pair and trailing /T/R/ symbol pair. When data is not transmitting, it is replaced by the constant transmission of idle symbols. The transmit function is divided into two blocks as shown in Figure 4:

- The Transmit State Machine
- The Data Path

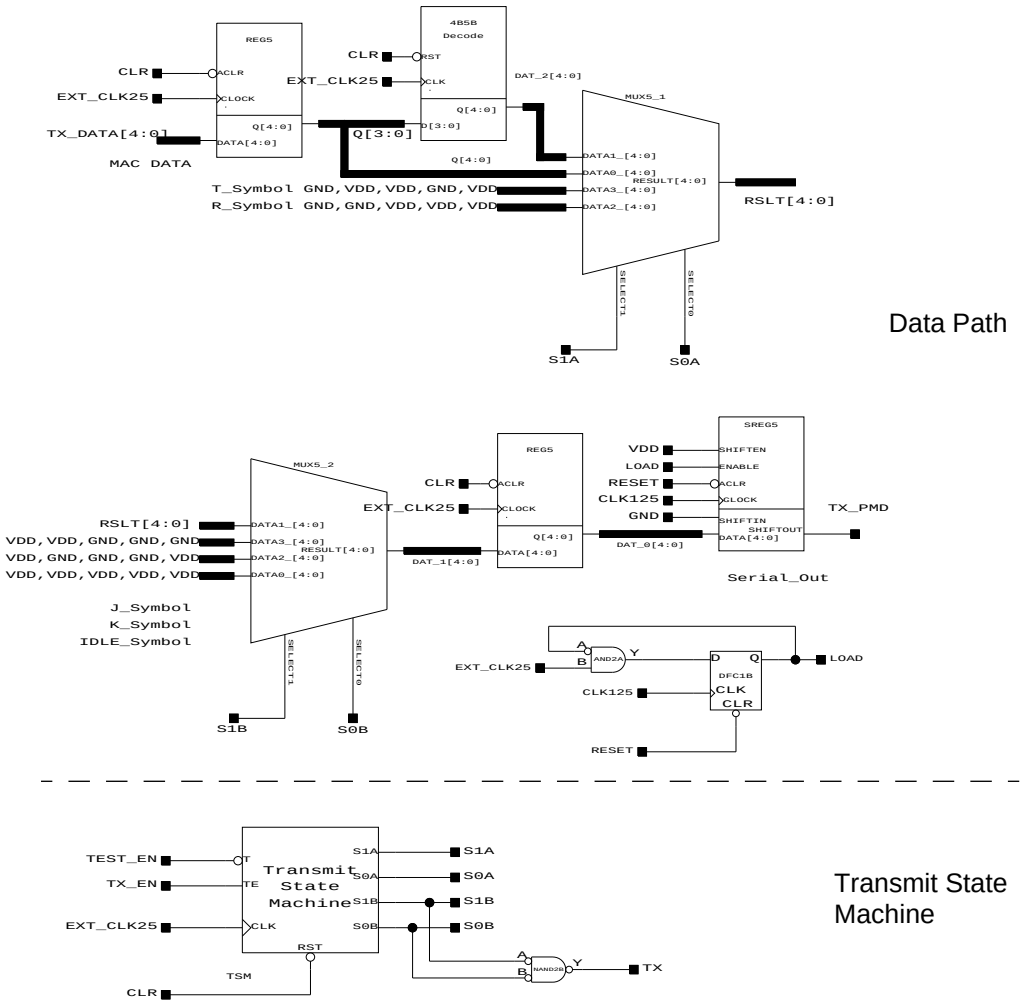

Figure 4 • *Convergence Sublayer Transmit Functions*

Data Path

The data path portion of the transmit block is shown in Figure 4. The main flow of data comes from the MAC at 25 MHz with a 4-bit-wide data word and a control signal that enables transmission. When MAC data is not being transmitted, the convergence sublayer sends continuous idle (I) symbols to the PMD. When the transmitEnable signal becomes active, a /J/K/ symbol pair is transmitted to indicate the beginning of a data packet. MAC data symbols then follow and are encoded into 5-bit symbols using the 4B5B encoding scheme shown in Table 1. The end of MAC data is indicated by the transmitEnable signal going inactive and a /T/R/ symbol pair is inserted at the end of the data packet. Finally, the CS logic returns to transmitting idle symbols.

Table 1 • *4B5B Symbol Coding*

Symbol	5-bit Code Group (in Convergence sublayer)	4-Bit-Binary Code Group (in MAC)	Interpretation/Function
0	11110	0000	Data character: 0H
1	01001	0001	Data character: 1H
2	10100	0010	Data character: 2H
3	10101	0011	Data character: 3H
4	01010	0100	Data character: 4H
5	01011	0101	Data character: 5H
6	01110	0110	Data character: 6H
7	01111	0111	Data character: 7H
8	10010	1000	Data character: 8H
9	10011	1001	Data character: 9H
A	10110	1010	Data character: AH
B	10111	1011	Data character: BH
C	11010	1100	Data character: CH
D	11011	1101	Data character: DH
E	11100	1110	Data character: EH
F	11101	1111	Data character: FH
I	11111	--	Idle character transmitted between packets
J	11000	--	First control character in start-of-packet delimiter
K	10001	--	Second control character in start-of-packet delimiter
T	01101	--	First control character in end-of-packet delimiter
R	00111	--	Second control character in end-of-packet delimiter
V	00000	--	Invalid character
V	00001	--	Invalid character
V	00010	--	Invalid character
V	00011	--	Invalid character
V	00100	--	Invalid character
V	00101	--	Invalid character
V	00110	--	Invalid character
V	01000	--	Invalid character
V	01100	--	Invalid character
V	10000	--	Invalid character
V	11001	--	Invalid character

The implementation of the described functions involves selecting six different symbol sources for PMD data: the I (idle), J, K, T, and R symbols and the 4B5B encoded MAC data. In addition, a TestData input can be used to provide raw unencoded data to the PMD for use in diagnostics and testing. This selection is accomplished via the multiplexer in front of the output shift register. One multiplexer selects from the I, J, and K symbols or from another multiplexer output. The other multiplexer selects from the T and R symbols and the encoded MAC data. Note the additional path around the encoder, which allows raw (unencoded) data to be provided to the PMD. This is used for system test and diagnostics and is the only way to inject known errors into the system, simulating collision remnants and exercising the boundary conditions of the standard.

The Actel logic implements multiplexers directly in a single logic module so, by inspection, the path through the multiplexer tree requires only two module delays and can easily meet the 25 MHz performance requirement. The 4B5B encode block also requires only two logic levels and can be

designed via schematics or via equations. The equations can be automatically compiled using Actel's ACTmap VHDL Synthesis tool and then incorporated into the schematic.

4B5B Encoder

Symbol encoding of the 4-bit data words transmitted from the MAC into the 5-bit coded groups required by the convergence sublayer and the PHY layer employs a modified version of the coding used in FDDI-based systems. The differences from FDDI are that the symbols S, Q, and H are not used and that R is now used as part of the /T/R/ end-of-packet delimiter character group.

Table 1 lists all 32 5-bit data- and special-symbol codes that the PMD can send to the convergence sublayer. The 16 data characters—0 through F (hex)—are shown in Table 1, both as 5-bit code groups and as their 4-bit binary equivalents, as sent by the CS to the MAC. The idle character I and the control characters J, K, T, and R are shown in Table 1 in 5-bit form only, because they are not used in the MAC. The same applies to the remaining 11 possible 5-bit combinations that might be received on the media, all of which have no meaning to the decoder and hence are treated as invalid. For simplicity, each of the 11 invalid symbols is designated as V.

Encoding of 4-bit data words into 5-bit symbols can be accomplished in a few simple logic equations, as shown in the PALASM2 entry format shown in Figure 5.

The above equations translate each bit in sequence. Bits D0-D3 are the 4-bit data word input to the decoder, and B0-B4 define the 5-bit output symbols from the decoder. Thus, in the first equation, bit D0 is always the same as the bit B0, as can be seen by inspection of Table 1. The decoding equations for the remaining output bits (bits B4 through B1) are derived in a comparable fashion.

ACTmap VHDL Synthesis

These equations are then processed by ACTmap VHDL Synthesis, a computer-aided design tool for working with the Actel families of FPGAs. It performs three basic functions:

- PALASM2, VHDL to netlist translation
- Netlist-optimized mapping
- I/O insertion

ACTmap reads the PALASM2 or VHDL source file and translates it into either an EDIF or an ADL (Actel Design Language) output file or Verilog netlist. The output file that it generates is optimized for a specific family of Actel FPGAs (ACT 1, ACT 2, 1200XL, 3200DX, or ACT 3).

You can specify whether the design should be optimized for area or speed. The PALASM2 description for the 4B5B encoder shown in Figure 5 was processed by ACTmap VHDL, and the following results where achieved:

- Area = 9 modules
- Estimated worst-case delay = 8.80 ns

```
;Encoder for 4B to 5B
;Used in 100 Mbit Ethernet application
CHIP 4b5b generic
clk rst d3 d2 d1 d0 q4 q3 q2 q1 q0

EQUATIONS

q4 := d3 + (/d2 * d1) + (/d2 * /d0)
q3 := d2 + (/d3 * /d1)
q2 := d1 + (/d3 * /d2 * /d0)
q1 := (/d1 * /d0) + (d3 :+: d2) + (d2 * /d1)
q0 := d0

q4.clkf = clk
q3.clkf = clk
q2.clkf = clk
q1.clkf = clk
q0.clkf = clk

q4.rstf = /rst
q3.rstf = /rst
q2.rstf = /rst
q1.rstf = /rst
q0.rstf = /rst
```

Figure 5 • PALASM2 Description for the 4B5B Encoder

These results easily meet the 25 MHz requirements of the transmit function and show the speed and capacity capabilities of the ACT 3 architecture. The schematic logic implementation of the 4B5B encoder (see Figure 6) shows the compact nature of the final implementation.

Transmit Operation

Transmit operational states are shown in block diagram form in Figure 7.

The actions shown in Figure 7 are assumed to be instantaneous, although, for simplicity, some time-sequenced events are contained in single states. Unconditional state transitions are unlabeled. Conditional state transitions occur when explicitly shown by the accompanying condition; a state is repeated until some transitional condition is detected. States are atomic in that conditions are evaluated only at the completion of the state's actions. Transitions shown without source states, notably linkTestFail, are evaluated at the completion of every state and take precedence over other transition conditions.

Convergence Sublayer Transmit Operation

The transmit state block diagram begins with the IDLE state. The transmitting and collisionDetect signals are initialized as FALSE and the IDLE symbol is continuously supplied to the PMD. Once the MAC has data to transmit, it asserts transmitEnable and the START state is entered. The transmitting signal is asserted (set to TRUE) to indicate to the Carrier Sense function that data is being transmitted. In addition, collisionDetect is set to the level of the receiving signal. The receiving signal comes from the Receive function; if it is also asserted, a collision has occurred. The waitNibble function synchronizes the MAC data with the PMD clock. The first 8-bits of the MAC preamble are replaced with the /J/K/ symbol pair. If transmitEnable becomes FALSE, the machine makes a transition back to IDLE. If transmitEnable stays asserted, the next state becomes TRANSMIT. During TRANSMIT state, collisionDetect is still set to receiving. The MAC data (TxDATA) is encoded using the 4B5B function, and encoding continues until transmitEnable is disabled. Once transmitEnable is deasserted, the machine makes a transition to the END state. In the END state, transmitting and collisionDetect are both FALSE. The /T/R/ symbols are transmitted to indicate the end of data, and the machine moves to the IDLE state. The assertion of linkTestFail (by the Link Monitor function) causes an immediate transition to the IDLE state and takes precedence over any MAC request.

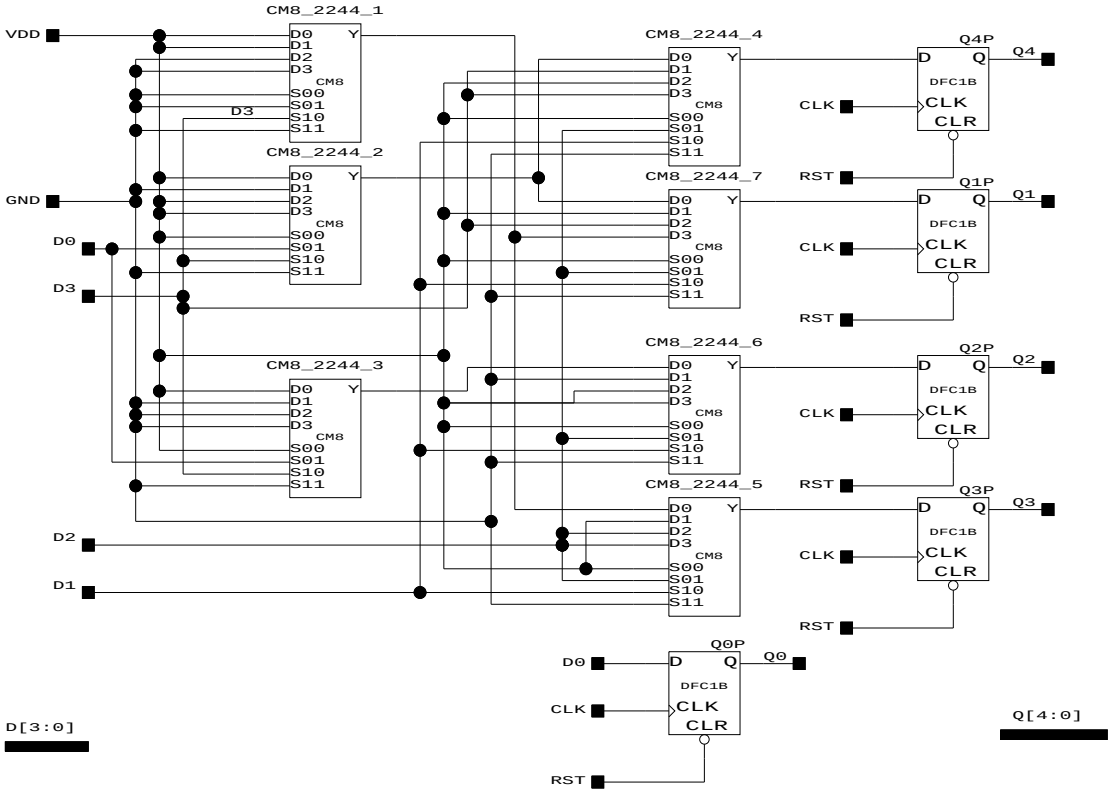

Figure 6 • *Transmit Path: 4B5B Encoder FPGA Logic*

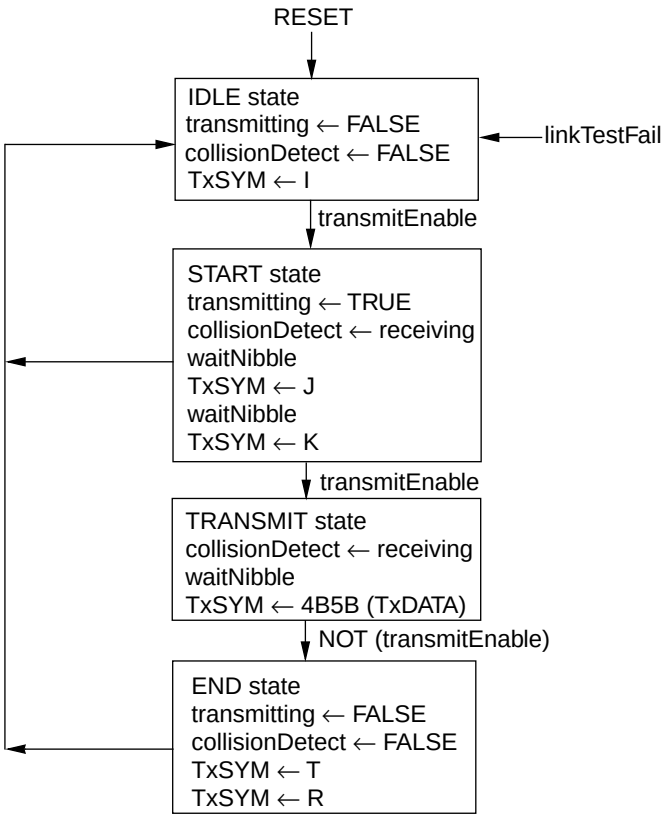

Figure 7 • *Convergence Sublayer Transmit Operation*

Transmit State Machine

The state-diagram implementation of transmit operation is shown in Figure 8. The state machine starts in the IDLE state and transmits the idle symbol (I) until transmitEnable (TE) is TRUE. As long as TE is TRUE, the machine proceeds through the J and K states, sending first the J symbol and then the K symbol to indicate the start of a data packet. The machine then transmits data until TE goes FALSE (i.e., transmit not enabled (/TE)), after which a T and an R are transmitted, indicating the end of the data packet. The state machine then returns to the IDLE state and waits for the next data packet. The test mode may be entered from the IDLE state by asserting Test mode (TM). In this mode, any 5-bit code symbol may be transmitted, thus allowing known error conditions to be injected onto the network.

The logic implementation of the transmit state machine is shown in Figure 9. Each state is encoded into the transmit state machine flip-flops to allow symbol selection in the transmit multiplexer. These transitions are controlled by the input logic for each flip-flop and depend only on the TE signal and the current state.

The resulting design employs only 11 logic modules and runs well in excess of the required 25 MHz speed.

Note: *This state-machine implementation differs from the commonly seen one-hot approach in that the states are encoded into four D flip-flops rather than a single flip-flop per state. Also, the state-machine encoding shown in this application is more efficient than the one-hot approach because the state flip-flops can drive the data-path multiplexer directly, eliminating the additional encoding logic that would be required to handle the one-hot state variables and to select desired multiplexer sources.*

Convergence Sublayer Receive Function

The receive functions of the convergence sublayer are shown in the block diagram in Figure 10. These functions are discussed in the following sections.

- Shift register, sync detect, and squelch
- Clock generation
- 5B4B symbol decoder
- Receive state machine

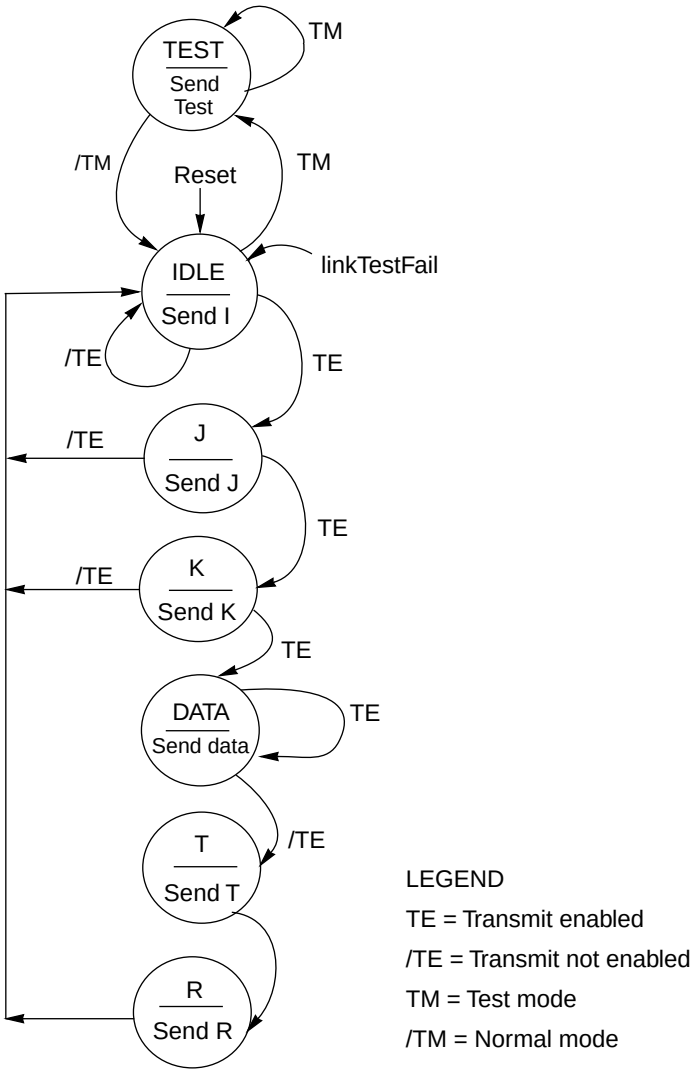

Figure 8 • *Transmit State Machine Diagram*

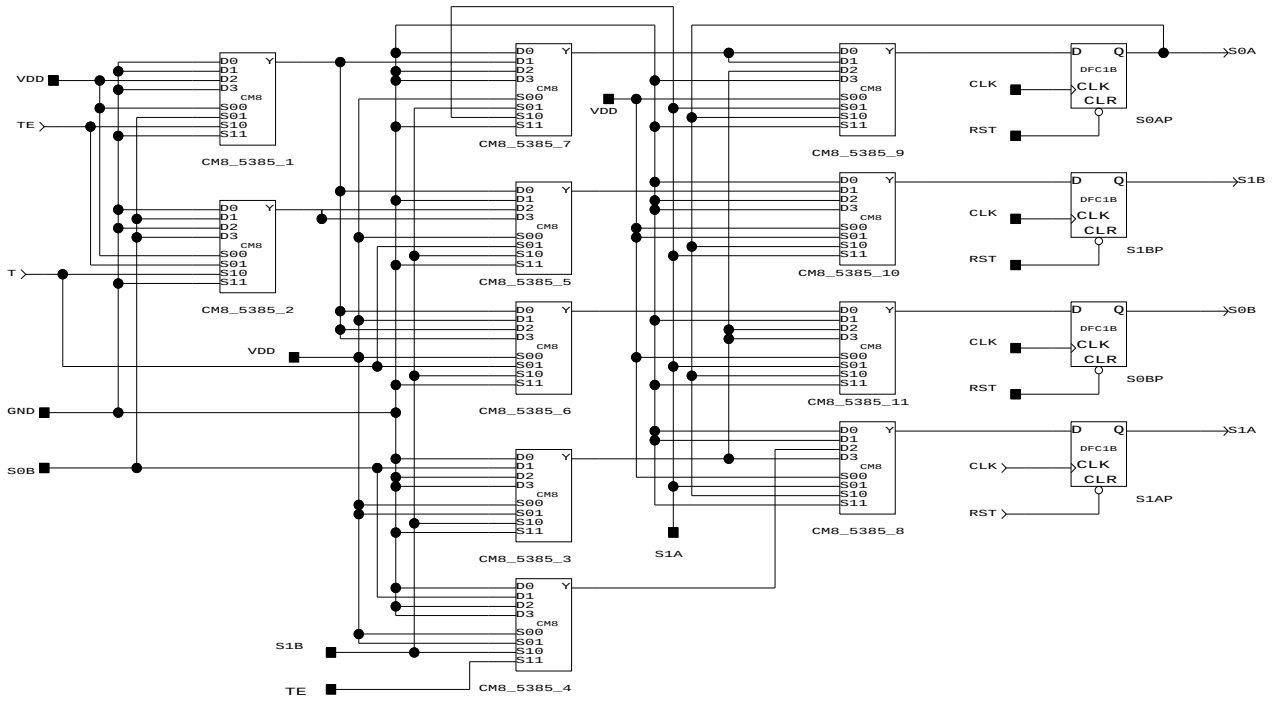

***Figure 9** • Transmit State Machine Schematic*

Receive Operation

The sequence of receive states is shown in the receive-operation diagram, Figure 11. The receive state machine tracks the received symbols to ensure that a complete packet has been received and indicates the current line state to the next layer of the protocol. The receive process (see Figure 11) involves two separate sets of states. The constituents in the first set—the IDLE, SCAN, CARRIER, and ALIGN states—are prealigned and operate on the raw input bits using RxBIT. The remaining states are aligned and operate on the input data stream as symbols (RxSYM). Output data, designated RxDATA, is sent directly to the MAC in these states.

The RECEIVE state sequence begins with the IDLE state. The receiving signal and the optional receiveError signal are initialized to FALSE. The SCAN state is entered next and the waitBit function synchronizes the machine to the received data stream. At this point, the squelch function filters out noise events by not allowing a transition to the CARRIER state unless two nonconsecutive zeros are detected. Because carrierSense is used by the MAC for deferral purposes, it must be asserted on the detection of any received signal (i.e., received energy, or non-IDLE input) whether or not it's an actual packet. Since carrierSense is also used to detect collisions, it's important to avoid triggering on noise, specifically a single-bit event. If CARRIER is entered, RxDATA is initialized to all zeros (0000) and receiving is set to true.

The system enters the ALIGN state next. In ALIGN, the start of packet symbols /J/K/ is searched for. If at least two idle symbols (1111111111) are found instead, no start of packet has been detected and the machine moves to the IDLE state. If the /J/K/ symbols are successfully found, the START state is entered. In START, the MAC preamble data (55) is substituted for the received /J/K/ symbols. The waitQuint function assures that MAC data is not overwritten. The RECEIVE state is entered next. Usually, in the RECEIVE state, valid data is received and a transition to the DATA state is made. In the DATA state, receiveError is deasserted and 4B5B decoded data is sent to the MAC.

From the DATA state, the machine returns to the RECEIVE state. If, during the RECEIVE state, two idle symbols are received, the PREMATURE END state is entered, receiveError is asserted, and IDLE is reentered. If, in the RECEIVE state, invalid data is received, the DATA ERROR state is entered, receiveError is asserted, and RECEIVE is reentered. Invalid data is not transmitted to the MAC. If, in the RECEIVE state, a /T/R/ symbol pair is detected, the END state is entered, receiving is deasserted, and IDLE state is reentered.

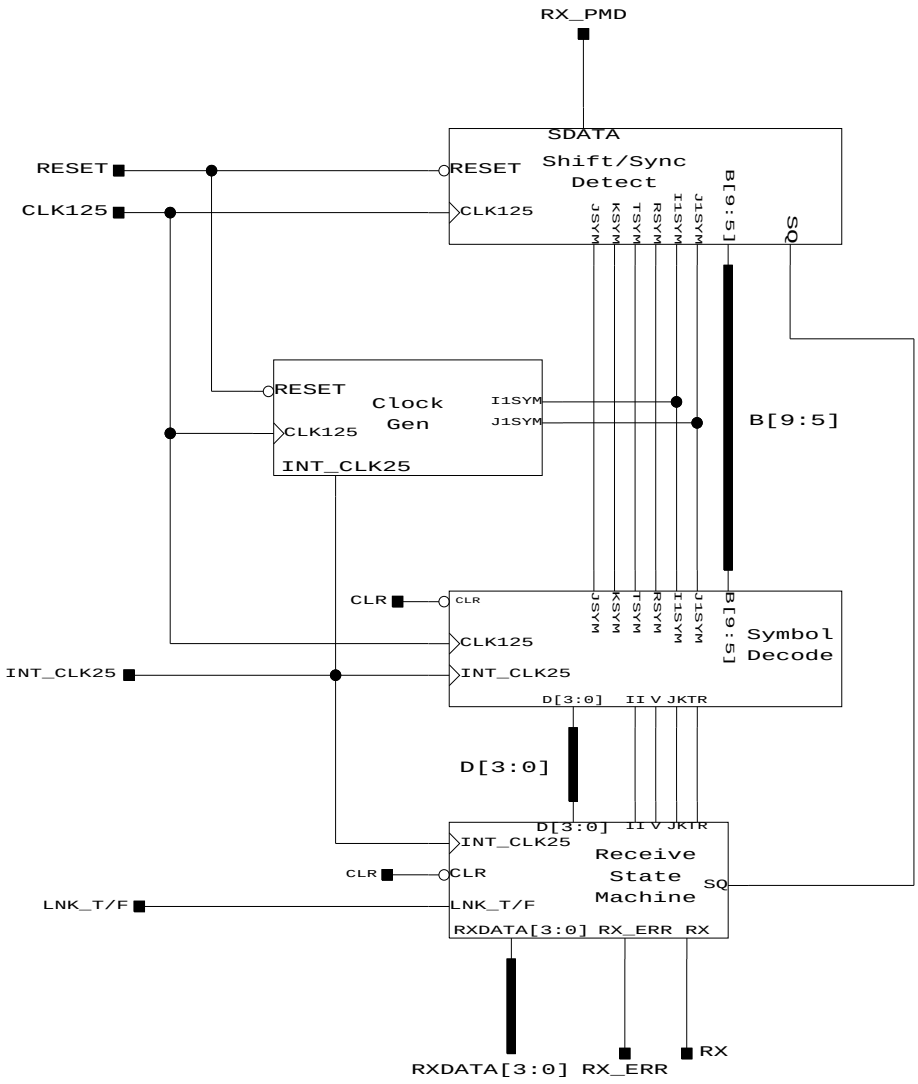

Figure 10 • *Convergence Sublayer Receive Functions*

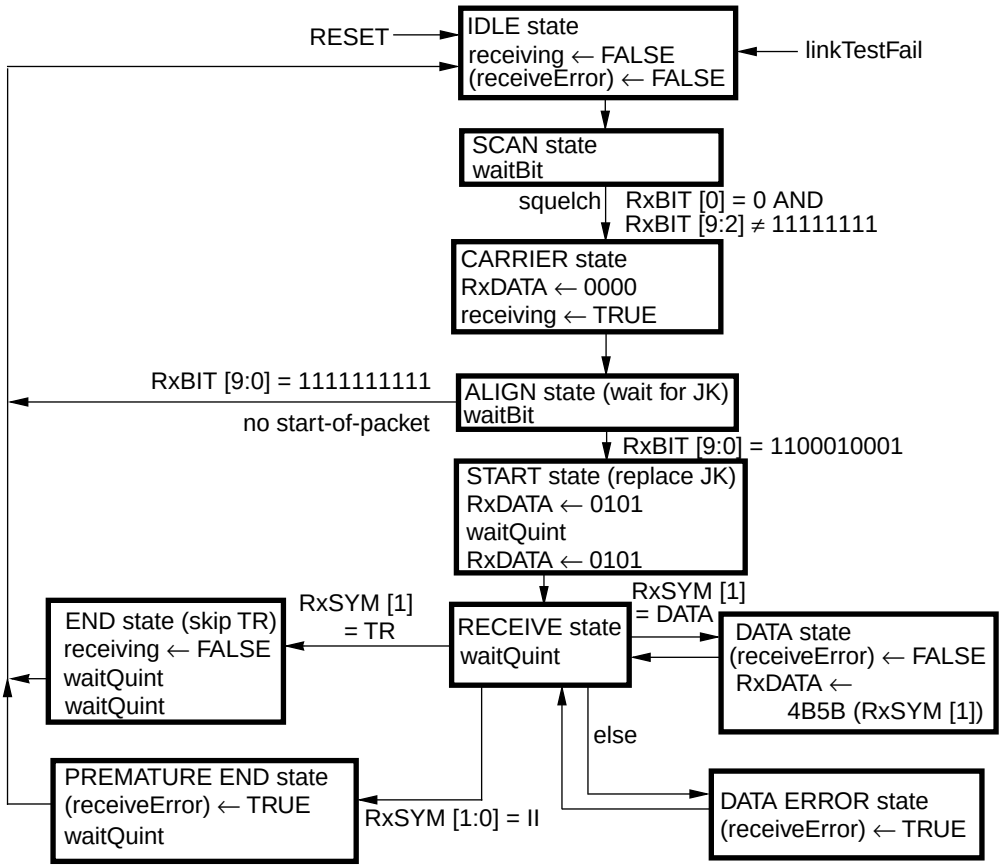

Figure 11 • *Convergence Sublayer Receive Operation*

Shift Register, Sync Detect, and Squelch

The shift register, sync detect, and squelch circuits (Figure 12) are responsible for shifting serial data at the 125 MHz line rate and detecting clock synchronization symbols. Once a sync symbol is detected, the clock generation state machine adjusts the 25 MHz symbol clock by stretching it the required number of 125 MHz clocks to align it with an input symbol. Control symbols in the input data stream can then be captured correctly by the 25 MHz clock and decoded by the 4B5B decode block.

Serial data is clocked into the shift register and sync detect block by using the 125 MHz clock. Sync symbols are detected as the data shifts. As shown in Figure 12, only a single logic level is required to detect each of the five important sync signals (J, K, T, R, and I). The code groups corresponding to each of these symbols are shown in Table 1.

Note: *The shift register generates both the true and complemented versions of bits B3, B4, B8, and B9. This is required to implement single-level decode for the I symbol because the ACT 3 logic module implements five-input AND/NAND gates with at least two inverted inputs. The technique of providing additional registers with inverted outputs is common when implementing logic functions using fine-grained antifuse FPGAs. The additional registers cost little because of the fine-grained logic module, and they can be used where needed to provide additional logic signals. The abundant routing resources available with antifuse FPGAs also supply the additional routing required to create these additional logic signals.*

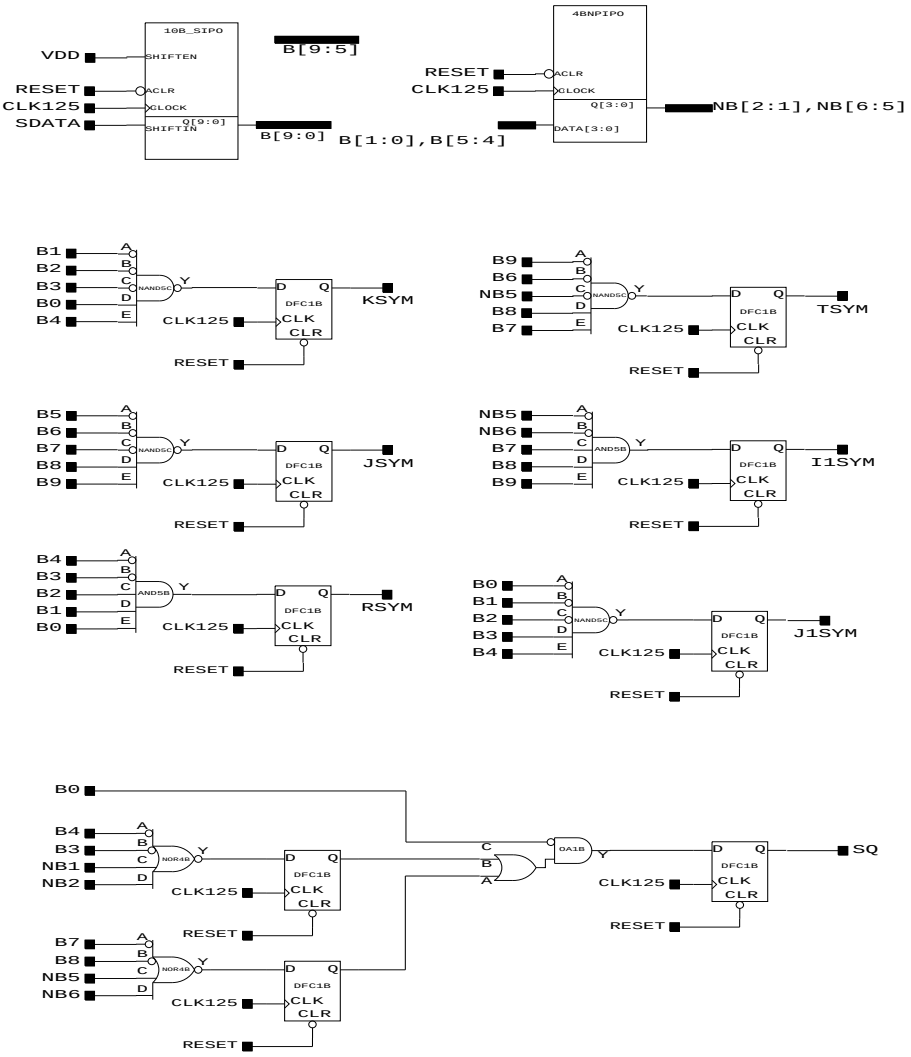

Figure 12 • *Shift Register, Sync Detect, and Squelch Schematic Diagram*

The squelch function filters out noise events from the received data stream. Zeros are ignored unless there are two noncontiguous zeros within the first 10 bits. At first glance, it would appear that the logic to detect two noncontiguous zeros in a 10-bit word should be quite extensive. However, once it is observed that this function is used only on a serial data stream, several simplifying logic reductions can be made. First, check the least significant bit (B0) for a zero, and then check that at least one of the higher-order bits (only B2 through B9, since B1 is contiguous) is also zero. Any other combination is simply a shift from B0. However, the resulting logic equation for a squelch state (S),

S:= /B0*(/B2+/B3+/B4+/B5+/B6+/B7+/B8+/B9)

is too large to implement in a single FPGA logic level.

To simplify this approach, note that because data is being serially shifted in, higher-order terms can be precomputed and then combined with the critical B0 signal using a single logic level. The logic that results can be expressed by the following three equations:

S1:= /(/B1+/B2+/B3+/B4)

S2:= /(/B5+/B6+/B7+/B8)

S:= /B0*(/S1+/S2)

These three equations are the ones actually implemented in FPGA form. (See Figure 12.)

Note: *As shown in Figure 12, bits B1–B4 and bits B5–B8 are used with registers to develop the two intermediate terms S1 and S2. These two are then ORed with bit B1 to develop the final squelch function (S). This form of pipelined operation works well in serial data applications and will almost always result in faster and more area-efficient FPGA designs.*

Also, notice that the extra inversions on the S1 and S2 terms (Figure 12) are used because NOR functions with inverted inputs map more easily into a single ACT 3 logic module. Synthesis software like Actel's ACTmap VHDL Synthesis program figures this out automatically, allowing the designer to focus on architectural and functional issues instead. Thus, what initially looks like a difficult decoding problem can be significantly simplified to only three logic modules that operate easily at the serial data rate.

Clock Generation

The clock generation state diagram and the clock generation schematic diagram are shown in Figures 13 and 14, respectively. The clock generation logic divides the 125 MHz serial clock by 5 to generate the 25 MHz symbol clock, Clock25. The Clock25 signal (Figure 13) is stretched when a sync symbol is detected, to align it with the 5-bit symbols. This is accomplished by a transition to the Q0 state when the JK signal (start of packet) is active. Entry into Q0 synchronizes the 25 MHz clock (Clock25, the output from states Q2 and Q3) and the load signal. The load signal is active every 5 clocks after synchronization, which captures the 5-bit symbol from the aligned data stream. The symbol can then be safely captured by the 25 MHz clock, Clock25 in the aligned symbol register (ASR). The schematic implementation for this process is shown in Figure 14.

Note: *Each state in the machine uses a single register. This one-hot (i.e., one register at a time) type of state machine design uses the register-intensive nature of fine-grained, antifuse-based FPGAs to reduce the logic complexity required to determine next-state transitions. In traditional encoded designs every state bit is needed to determine which state the machine is in. This can make for large transition terms in complex state machines. FPGAs, on the other hand, can use the additional register available to reduce the logic complexity, because only a single register output is required to determine the state of the machine. Thus, the FPGA's narrow, high-speed logic module can be used to generate the transition terms efficiently. In fact, on closer examination, the implementation of the state machine maps very closely to the state diagram. Transitions from one state to another result in a connection from the starting-states register to the entered-states register. Logic complexity can easily be estimated directly from the state diagram. Because only a single logic module is required to implement even the most complex transition, the entire machine runs easily at the 125 MHz clock rate.*

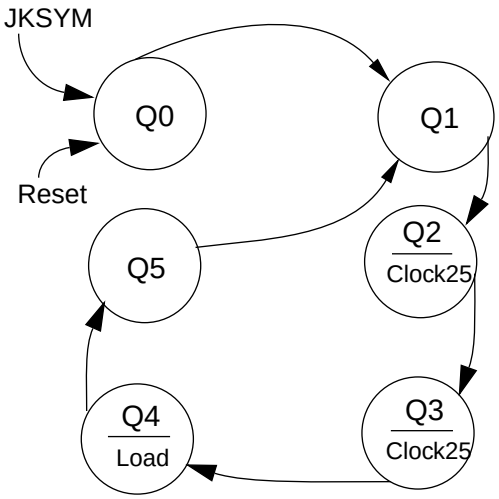

Figure 13 • *Clock Generation State Diagram*

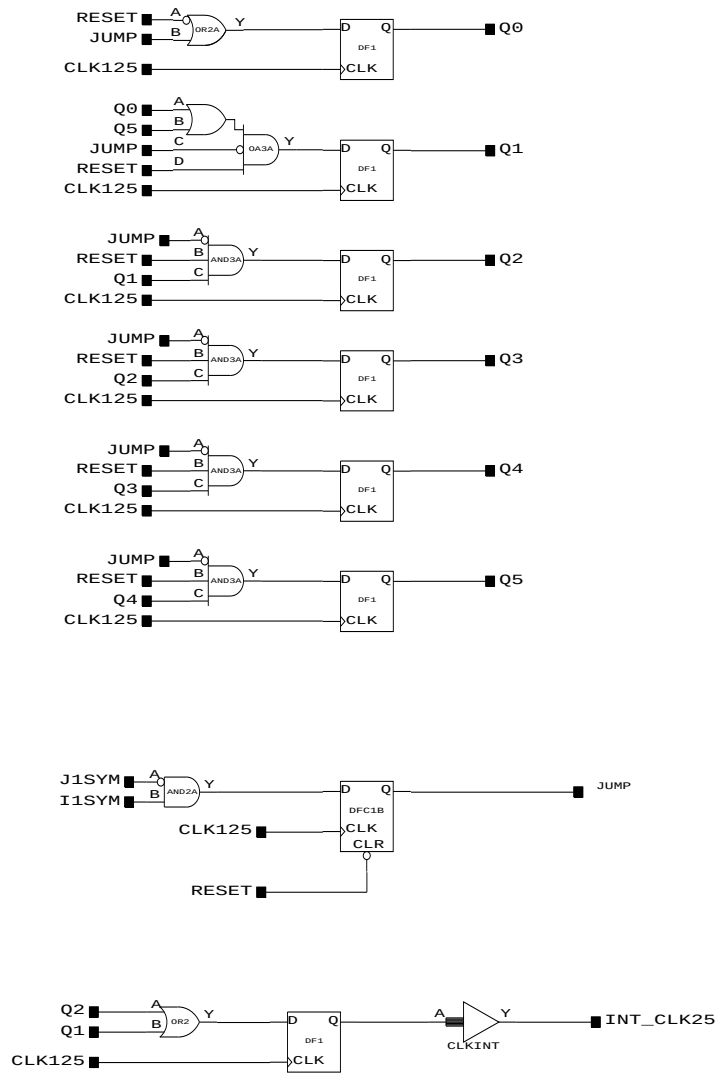

Figure 14 • *Clock Generation State Machine Schematic*

5B4B Symbol Decoder

Once a symbol has been aligned, the data must be extracted by converting the 5-bit input from the PHY into a 4-bit data word that is sent to the MAC. The logic diagram for this decoder is shown in Figure 15. Symbol conversion is done in accordance with the 4B5B decode table, Table 1. Implementation of the decoder in the ACT 3 family is automatically generated from the logic equations developed from the encoding table by using the ACTmap VHDL tool. As shown in Figure 15, the full decode requires only 24 modules and only two levels of logic, easily meeting the speed required for the 25 MHz clock.

Receive State Machine Logic

As shown in Figure 17, the schematic implementation of the RECEIVE state machine requires only 4 logic modules and two levels of logic. It easily meets the 25 MHz clock rate required for this portion of the design.

Receive State Machine Diagram

The RECEIVE state diagram is shown in Figure 16. The machine begins in the START state and waits for the reception of a JK symbol pair. The RECEIVE state is entered upon the reception of this pair and exited only under one or more of the following conditions:

- Reception of a TR symbol pair (end of data packet)
- Reception of an idle (I) symbol (premature packet end)
- Reception of an invalid symbol (error condition)

Note that if an invalid symbol is received, the ERROR state is entered to capture the event. This state is cleared only by resetting the state machine.

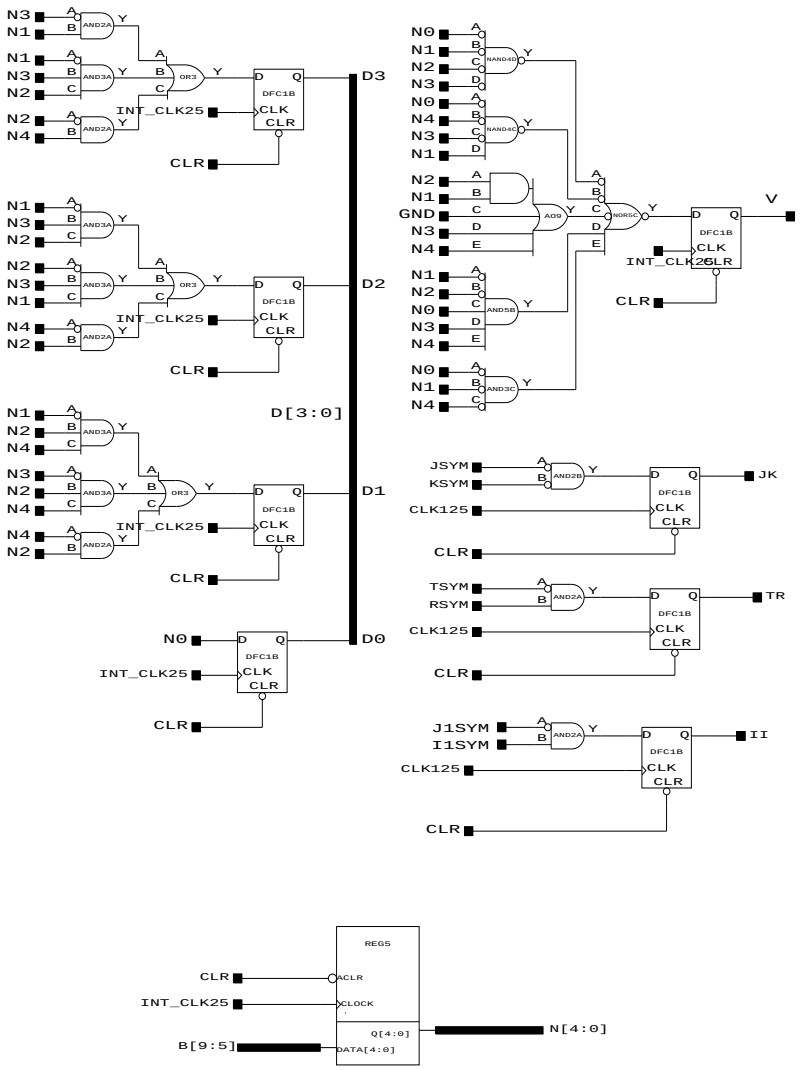

Figure 15 • *5B4B Decoder Schematic Diagram*

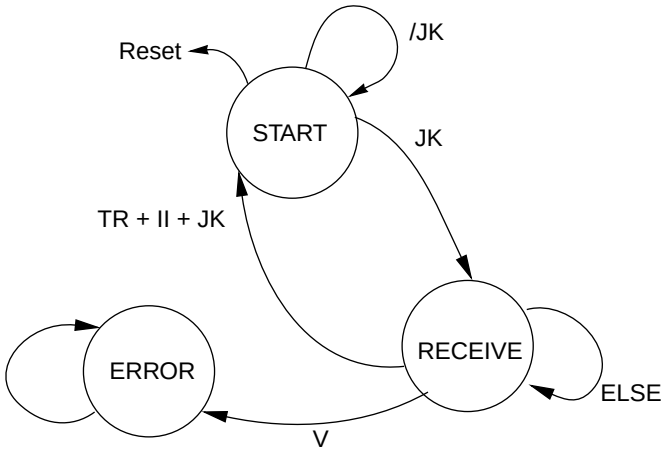

Figure 16 • *Receive State Machine Diagram*

Carrier Sense and Link Monitor Circuits

The carrier sense and link monitor circuits combine outputs from the transmit, receive, and PMD blocks to develop the receiveError, carrierSense, and collisionDetect signals. The logic for implementing this process is shown in Figure 18.

Conclusion

This application note has described the complete design of a 100Base-X convergence sublayer. Each building block has been fully tested and documented and is available on disk to those contemplating similar or related applications.

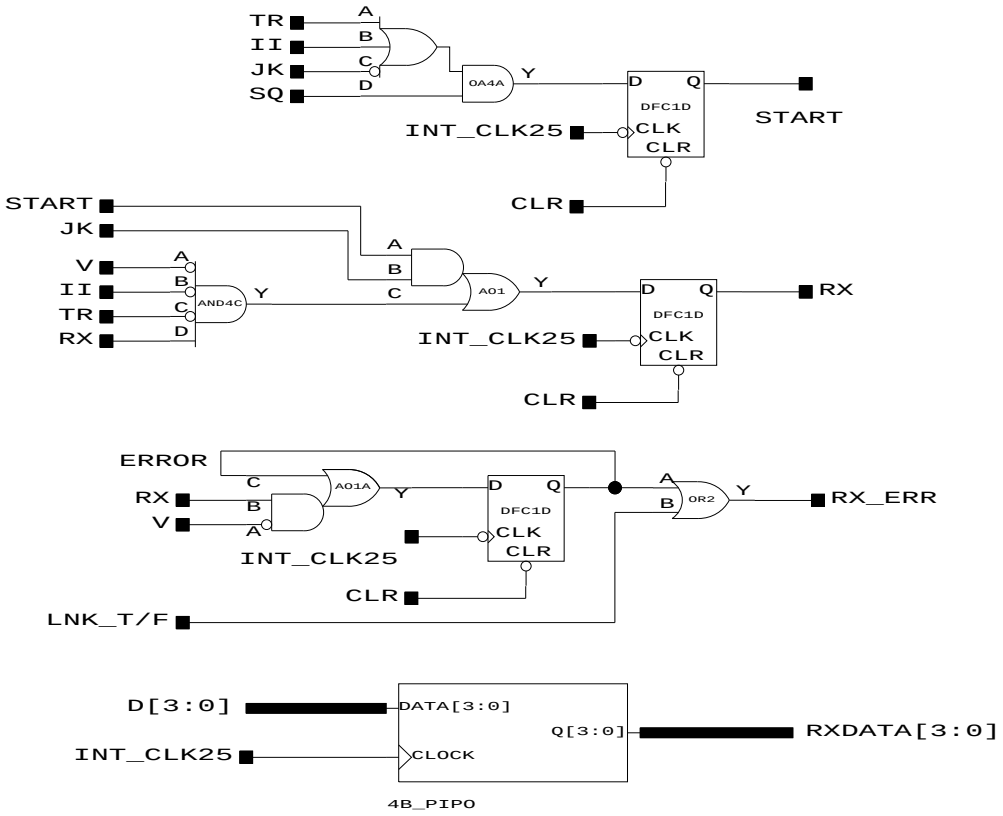

Figure 17 • *Receive State Machine Schematic Diagram*

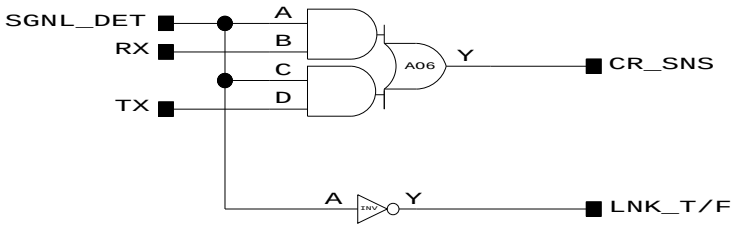

Figure 18 • *Carrier Sense and Link Monitor FPGA Logic Schematic*