IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 2, Issue 2 (July-Aug. 2012), PP 09-17
www.iosrjournals.org

ASIC implementation of STM-1 Framer and De-Framer

Anand T[1], Dr. Siva S Yellampalli [2]

[1,2]*(VLSI Design and Embedded System, VTU Extn. Centre, UTL Technologies, INDIA)*

Abstract : *In this paper we present the ASIC implementation of STM-1 Framer and De-Framer. This paper mainly focuses on multiplexing digital data, transmitting and receiving the STM-1 frame. The design is implemented using Verilog HDL, simulated on Modelsim and Synthesized on Xilinx ISE 13.2. For power analysis and area calculation, the designed framer and de-framer are analysed using Cadence version 6.1.5. For debugging Chipscope Analyser has been used. The designed framer can be used for generation and analysis of STM-1 frame that has a data rate of 155.52Mbps.*

Keywords - *PRBS, STM-1 frame, Scrambler, Descrambler, BIP, SOH, POH.*

I. INTRODUCTION

The SDH is a hierarchical set of digital transport structures, standardized for the transport of suitably adapted payloads over physical transmission networks. The SDH defines a structure which enables plesiochronous signals to be combined together and encapsulated within a standard SDH signal. The ITU-T recommendations define a number of transmission rates within the SDH. The first of these is 155.52 Mbit/s, normally referred to as STM-1 (where STM stands for 'Synchronous Transport Module') [1]. The recommendations also defines a multiplexing structure whereby an STM-1 signal can carry a number of lower bit rate signals as payload, thus allowing existing PDH signals to be carried over a synchronous network. The SDH defines a number of "containers" each corresponding to an existing plesiochronous rate. Information from the plesiochronous container is mapped into the relevant container. The way in which this is done is similar to the bit stuffing procedure carried out in a conventional PDH multiplexer. SDH is currently the dominant choice for metropolitan-area networks as well as for accessing wavelength division multiplexing networks in wide-area networks [7]. The purpose of this paper is to carry multiple digital signals on a single medium. The paper also describes whether the data that is transmitted is received correctly or not. The STM-1 frame is capable of transporting any PDH tributary signal ($\leq$ 140 Mbit/s). The frame comprises of section overhead (SOH), pointer and the payload as shown in fig 1.

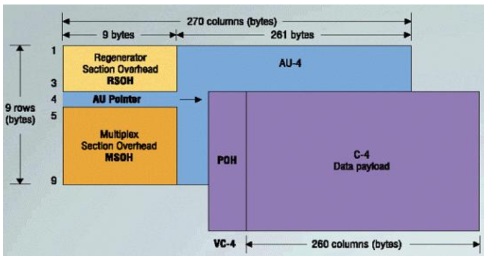


Figure 1: stm-1 frame structure [9]

II. DESIGN AND IMPLEMENTATION

According to the ITU-T standard G.707 the specification of STM-1 Frame are as follows:

Number of rows in a frame = 9 rows
Number of columns in a frame = 9+261 = 270 columns
Number of bytes/frame = 9*270 = 2,430 bytes
Number of bits/frame = 9*270*8 = 19440 bits
Number of bits per second = 9*270*8*8000
= 155,552,000 bits per second
= 155.52 Mbits/s.

The implementation has Framer and the De-Framer module which is sub-divided as follows:
For the Framer Module:

1. Design of the 23-bit PRBS Generator Module.
2. Design of Framer Generator Module.
3. Design of the Scrambler Module.

4. Design of B1 Calculation Module
5. Design of B2 Calculation Module
6. Design of B3 Calculation Module

For the De-Framer Module:

1. Design of Head Detector Module.
2. Design of Descrambler Module.
3. Design of Overhead detector and extractor.
4. Design of PRBS Detector Module.
5. Design of Clock Divider Module.
6. Design of BIP Error Detector Module

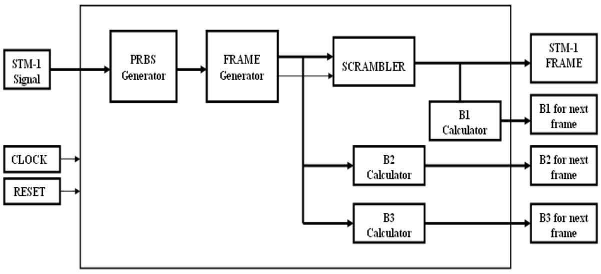


Figure 2: stm-1 framer

The figure 3.1 shows the block diagram of how an STM-1 frame is generated. The 8 bit PRBS data is used to fill the payload section of the STM-1 frame. At every positive edge of the clock the PRBS data is fed into the payload block. When the RESET signal is high the frame will be reset to the initial value. The Frame Generator block adds the overhead bytes at appropriate positions and fills the remaining payload section with the prbs data.

The B3 calculation block will calculate the even parity over all the bytes of the payload section of the previous frame before scrambling and will be placed in the B3 location of the current frame before scrambling. The Regenerator section overhead and Multiplex section overhead values are also sent into the frame generator at appropriate locations to complete the frame.

The B2 calculation block will calculate the parity of Multiplex Section and the payload section except for the first 3 rows of the Regenerator Section Overhead, and place the calculated value in the B2 bytes of the current frame before scrambling.

The B1 Calculation block calculates the even parity over all the bytes of the previous frame after scrambling, and places the calculated value in B1 location of the current frame before scrambling.
Finally the Scrambler block will scramble the incoming data and will then transmit the scrambled data.
The fig 3 shows the block diagram of STM-1 Deframer. The STM-1 Frame signal is fed to the Head Detector block which performs frame synchronization as per ITU-T G.707 [1]. It generates Loss-of-Frame (LOF) and Out-of-Frame (OOF) alarms.

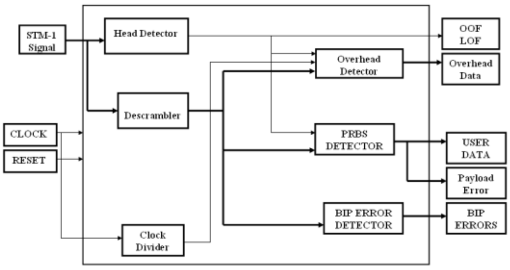


Figure 3: stm-1 de-framer

The Received signal is descrambled and then fed to the Overhead detector which will detect the overhead**.** The overhead detector extracts the Regenerator Section Overhead and outputs it from the core. It also extracts the Multiplex Section Overhead and outputs it from the core. The block extracts Higher Order Path Overhead for all the configured VCs.

The PRBS detector block will compare the received payload data with the original data and will indicate if the data received is correct or not.

The BIP error detector block verifies the incoming B1 value and indicates the presence and number of any B1 errors. The BIP error detector block verifies the incoming B2 value and indicates the presence and number of any B2 errors. It also calculates and verifies the B3 value for all VCs and indicates the presence and number of B3 errors.

III. Submodules Description

3.1 23 bit PRBS Generator.

Figure 4 shows the schematic of a 23 bit PRBS generator. Here the PRBS data is intialized to 1 when it is reset. At every clock cycle the values are shifted to the left and the xored values of the tappings are inserted to the first bit. Then the first eight bits of the PRBS sequence is fed as the input to the STM-1 frame.

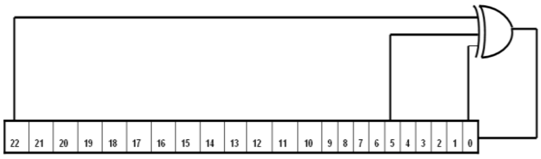


Figure 4: 23 bit prbs generator

The output of the LFSR is controlled by three parameters: clock, tap positions, and the initial value that is loaded into the LFSR or seed. For the 23 bit PRBS generator the seeds are used at 23rd bit, 6th bit and first bit according to [10].

3.2 STM-1 Frame Generator

When the reset signal is high the count is reset to zero and the memory is cleared. The scrambled data is sent as the output of the system and whenever the scrambled data changes the output is updated. The data that are being sent to the output depends on the value of count. The data's are sent in left to right fashion and are based on the ITU.T standard G707 [1].

3.3 SCRAMBLER

Scrambling of the bits in a synchronous transport module (SONET) frame is needed to keep the frequency content of the transmitted signals near the actual line rate [6].

There are two main reasons why scrambling is used

- To eliminate long sequences of zeros and ones.
- It eliminates the dependence of a signal's power spectrum upon the actual transmitted data, making it more dispersed to meet maximum power spectral density requirements.

The scrambler used in this implementation is a parallel scrambler shown in figure 5. The scrambler is reset to 1111111 at the start of the frame; by loading all seven flip flops with 1's.

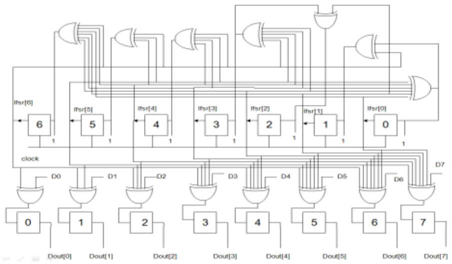


Figure 5: 8 bit parallel scrambler

3.4 B2 Calculation Module

The B2 bytes are allocated for a multiplex section error monitoring function. This function shall be a Bit Interleaved Parity 24 code using even parity. The BIP-24 is computed over all bits of the previous STM-N frame except for the first three rows of SOH and is placed in bytes B2 of the current frame.

3.5 B3 Calculation Module

The B3 byte is allocated in each virtual container for path error monitoring function. This module calculates the Bit Interleaved Parity 8 using even parity over all the bits of the previous virtual container and is placed in the B3 byte of the current virtual container.

3.6 B1 Calculation Module

One byte is allocated for regenerator section error monitoring. This function shall be a Bit Interleaved Parity 8 (BIP-8) code using even parity. The BIP-8 is computed over all bits of the previous STM-N frame after scrambling and is placed in byte B1 of the current frame before scrambling.

3.7 Head Detector Module.

The Head Detector block is used at the Receiving end to determine whether the frame has started or not. This is done by comparing the first six bytes with A1 and A2 which is the Frame Alignment Word and is used to recognize the beginning of an STM-N frame. A1 has a default value of F6h and A2 has a default value of 28h. The head detector will wait for these values and whenever it encounters all the six values the head signal will be asserted to indicate the start of the frame. Once the head signal is asserted the STM data will be extracted [5, 8]. The state machine for implementing Head Detector module is shown in figure 6.

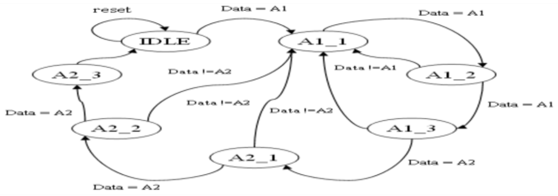


Figure 6: fsm for head detector

3.8 Descrambler

The Descrambler is required to retrieve the actual bytes that comprises SDH frame. It is similar to the scrambler used in the design. The descrambling operation is inverse of scrambling. The descrambler is used only after the first row of the scrambled SDH frame, until the end of the frame.

3.9 Clock Divider

The clock divider module is used to divide the system clock into different clock signals. These clock signals are used to output the overhead bytes at the receiver end.

3.10 Overhead Detector

The overhead detector module will detect the overhead bytes at their respective locations, extract the overhead byte and display them at the output. It also verifies the incoming BIP values with the calculated BIP value and indicates the presence and number of errors.

3.11 PRBS Detector

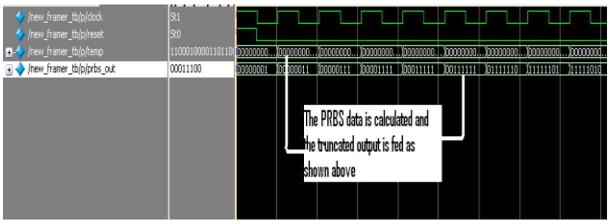
The PRBS detector module compares the incoming payload data with the original data and determines whether the data obtained is correct or not. If there is a difference between the received data and the calculated data of the payload section, the detector gives the number of bit that are inverted

3.12 BIP Error Detector

The BIP Error detector module compares the calculated value of BIP's and the extracted values of BIP and determines the type of error being encountered. If there is an error in B1 byte it tells the number of BIP violations happening in the frame. The error detection for B2 and B3 is similar to that of B1. The Error Detector also tests whether the received payload data is correct or not and if there is an error in the signal a Loss of data signal is indicated.

IV. Verification And Simulation

The verification objective is the development of test cases to ensure that the design implements defined functionality. It was performed in each of the sub-modules of the architecture. Simulation tests were performed using Modelsim and Cadence NcLaunch. The design was synthesized on Xilinx 13.2 for device utilization summary, and implemented on Cadence RTL compiler for area and power calculations. Fig 7 shows the output of the prbs generator which is truncated and fed as the input to the STM-1 framer.



```
/new_framer_tb/p/clock      St1
/new_framer_tb/p/reset      St0
/new_framer_tb/p/temp       1100010000110110  00000000... 00000000... 00000000... 00000000... 00000000... 00000000... 00000000... 00000000... 00000000...
/new_framer_tb/p/prbs_out   00011100          00000001 00000011 00000111 00001111 00011111 00111111 01111110 11111101 11111010
```

The PRBS data is calculated and the truncated output is fed as shown above

Figure 7: Verilog output of 23-bit PRBS

Fig 8 shows the structure and the output of STM-1 frame and the completion of a frame in 125us. It also shows the start of the next frame.

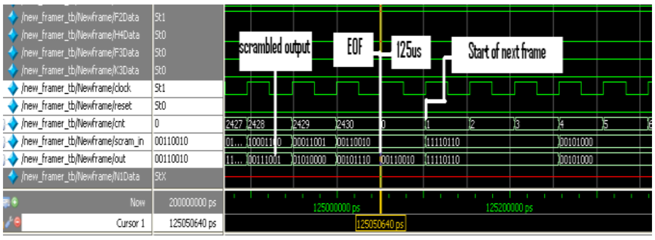


Figure 8: Verilog output of stm-1 framer

Fig 9 shows the output of the scrambler whose operation is as described in section 3.3. The STM-1 signal is fed as the input to the scrambler and the scrambled output is sent as the output of the STM-1 frame.

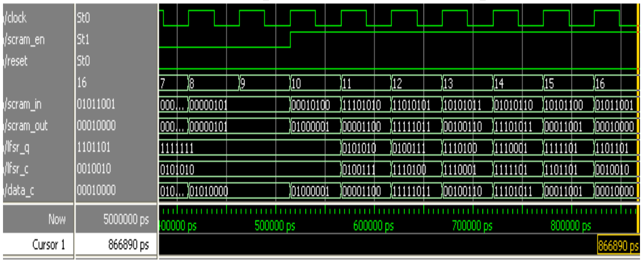


Figure 9: verilog output of scrambler

The figure 10 shows the output of detection and extraction of STM-1 overhead bytes. It shows the extraction of all the overhead bytes.

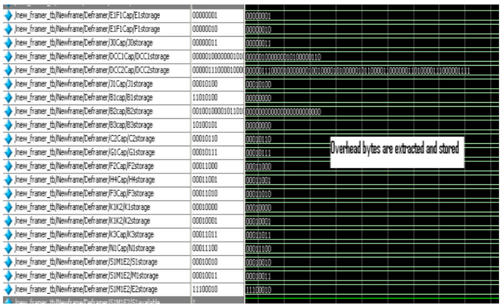


Figure 10: verilog output of overhead detector

Fig 11 shows the output of PRBS Detector, where the output error is zero for the whole frame indicating that the data has received correctly.

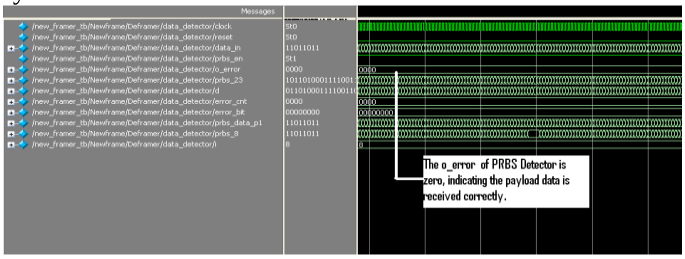

The o_error of PRBS Detector is zero, indicating the payload data is received correctly.

Figure 11: verilog output of prbs detector without error

Fig 12 shows the error being detected in the received data, where the error has been injected at the input and the output error will show the number of bits that are erroneous.

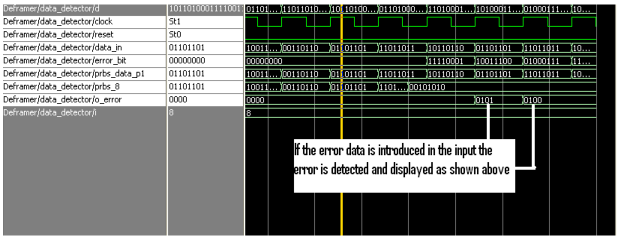
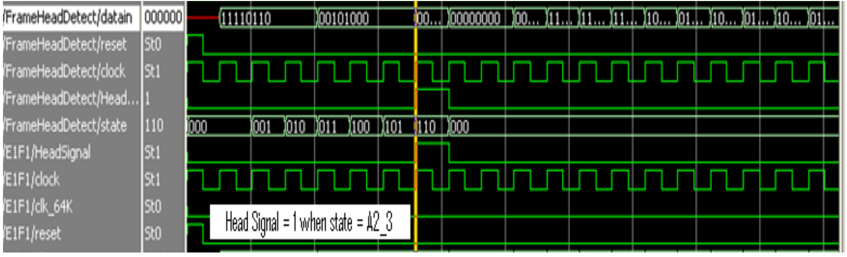

If the error data is introduced in the input the error is detected and displayed as shown above

Figure 12: verilog output of prbs detector with error

Fig 13 shows the output of Head Detector Block, where the Head signal will be set when a sequence of 3 A1's and 3 A2's are detected, indicating the start of the frame.

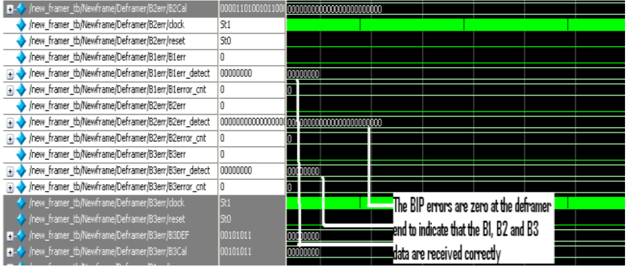


```
/FrameHeadDetect/datain   000000  11110110  00101000  00...  00000000  00...  11...  11...  11...  10...  01...  10...  01...  10...  01...
/FrameHeadDetect/reset    St0
/FrameHeadDetect/clock    St1
/FrameHeadDetect/Head...  1
/FrameHeadDetect/state    110     000  001  010  011  100  101  110  000
/E1F1/HeadSignal          St1
/E1F1/clock               St1
/E1F1/clk_64K             St0
/E1F1/reset               St0
```

Head Signal = 1 when state = A2_3

Figure 13: verilog output of head detector

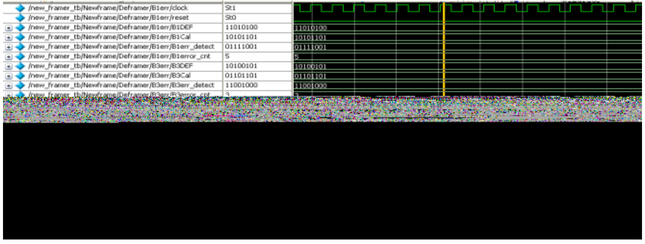
Fig 14 depicts the simulation result of BIP Error Detector where the error is zero indicating that the calculated and received BIP values are the same. Fig 15 depicts the simulation result of BIP error detector where the errors are forced manually and the difference can be seen between the received data and the calculated data. The difference is calculated and reported.



```
/new_framer_tb/Newframe/Deframer/B2err/B2Cal          0000110100101100  000000000000000000000000
/new_framer_tb/Newframe/Deframer/B2err/clock          St1
/new_framer_tb/Newframe/Deframer/B2err/reset          St0
/new_framer_tb/Newframe/Deframer/B1err/B1err          0
/new_framer_tb/Newframe/Deframer/B1err/B1err_detect   00000000          00000000
/new_framer_tb/Newframe/Deframer/B1err/B1error_cnt    0                 0
/new_framer_tb/Newframe/Deframer/B2err/B2err          0
/new_framer_tb/Newframe/Deframer/B2err/B2err_detect   0000000000000000  00 0000000000000000000000
/new_framer_tb/Newframe/Deframer/B2err/B2error_cnt    0                 0
/new_framer_tb/Newframe/Deframer/B3err/B3err          0
/new_framer_tb/Newframe/Deframer/B3err/B3err_detect   00000000          00000000
/new_framer_tb/Newframe/Deframer/B3err/B3error_cnt    0                 0
/new_framer_tb/Newframe/Deframer/B3err/clock          St1
/new_framer_tb/Newframe/Deframer/B3err/reset          St0
/new_framer_tb/Newframe/Deframer/B3err/B3DEF          00101011          00000000
/new_framer_tb/Newframe/Deframer/B3err/B3Cal          00101011          00000000
```

The BIP errors are zero at the deframer end to indicate that the BI, B2 and B3 data are received correctly

Figure 14: output of bip error detector without errors



```
/new_framer_tb/Newframe/Deframer/B1err/clock          St1
/new_framer_tb/Newframe/Deframer/B1err/reset          St0
/new_framer_tb/Newframe/Deframer/B1err/B1DEF          11010100   11010100
/new_framer_tb/Newframe/Deframer/B1err/B1Cal          10101101   10101101
/new_framer_tb/Newframe/Deframer/B1err/B1err_detect   01111001   01111001
/new_framer_tb/Newframe/Deframer/B1err/B1error_cnt    5          5
/new_framer_tb/Newframe/Deframer/B3err/B3DEF          10100101   10100101
/new_framer_tb/Newframe/Deframer/B3err/B3Cal          01101101   01101101
/new_framer_tb/Newframe/Deframer/B3err/B3err_detect   11001000   11001000
/new_framer_tb/Newframe/Deframer/B3err/B3error_cnt    3
```

Figure 15: output of bip error detector with errors

The figure 16 shows the RTL schematic of STM-1 framer implemented on Xilinx ISE.

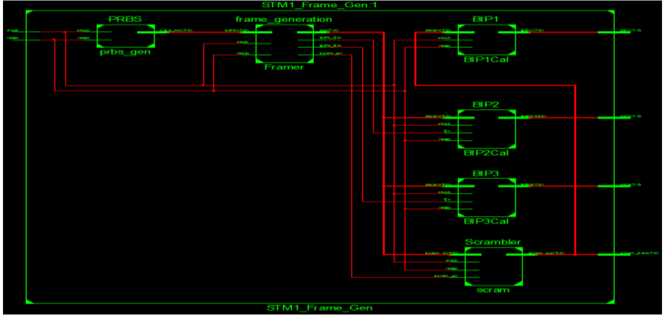
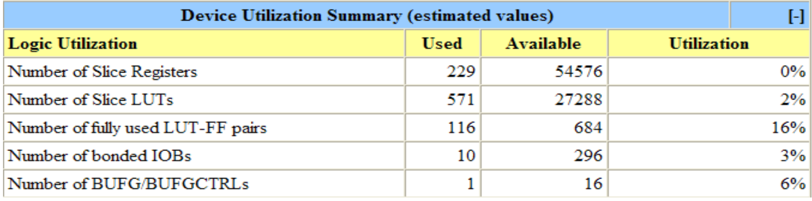


Figure 16: rtl schematic of stm-1 framer

Fig 17 shows the device utilization summary of STM-1 Framer implemented in Xilinx ISE 13.2 on Spartan 6 Evaluation board. It shows the number of LUT's occupied by the STM-1 framer.



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	229	54576	0%
Number of Slice LUTs	571	27288	2%
Number of fully used LUT-FF pairs	116	684	16%
Number of bonded IOBs	10	296	3%
Number of BUFG/BUFGCTRLs	1	16	6%

Figure 17: device utilization summary of stm-1 framer

Fig 18 shows the output of the STM-1 framer which is implemented on Chipscope, which is the tool used for debugging the output of a program.

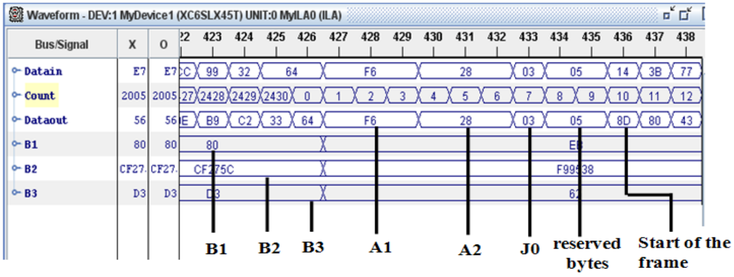
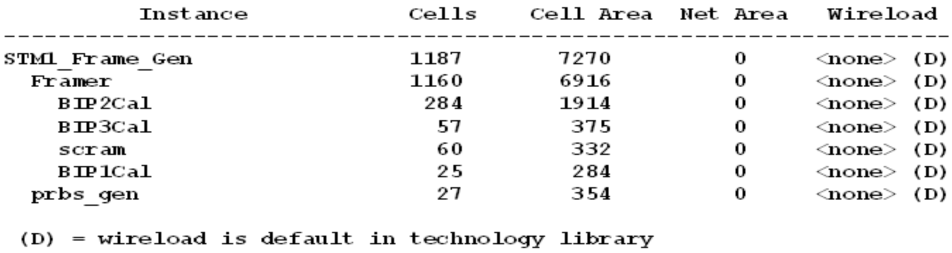


Figure 18: chipscope output of stm-1 framer

The fig 19 shows the area report of STM-1 Framer generated using Cadence RTL Compiler. The total area occupied by the STM-1 Framer is 7270um^2.

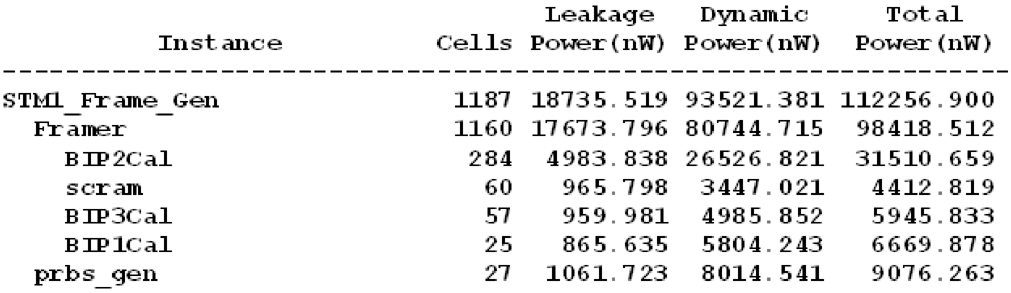


```
          Instance             Cells   Cell Area  Net Area   Wireload
-----------------------------------------------------------------------
STM1_Frame_Gen                  1187      7270        0     <none> (D)
  Framer                        1160      6916        0     <none> (D)
    BIP2Cal                      284      1914        0     <none> (D)
    BIP3Cal                       57       375        0     <none> (D)
    scram                         60       332        0     <none> (D)
    BIP1Cal                       25       284        0     <none> (D)
  prbs_gen                        27       354        0     <none> (D)

 (D) = wireload is default in technology library
```

Figure 19: area report of stm-1 framer

Figure 20 shows the power consumption of STM-1 Framer generated using RTL Compiler. The total power consumed by STM-1 Framer is 0.112mW.



```
                                   Leakage    Dynamic      Total
        Instance            Cells Power(nW)  Power(nW)   Power(nW)
-----------------------------------------------------------------
STM1_Frame_Gen               1187 18735.519  93521.381 112256.900
  Framer                     1160 17673.796  80744.715  98418.512
    BIP2Cal                   284  4983.838  26526.821  31510.659
    scram                      60   965.798   3447.021   4412.819
    BIP3Cal                    57   959.981   4985.852   5945.833
    BIP1Cal                    25   865.635   5804.243   6669.878
  prbs_gen                     27  1061.723   8014.541   9076.263
```

Figure 20: power report of stm-1 framer

Figure 21 shows the RTL Schematic of STM-1 De-Framer implemented on Cadence RTL Compiler.

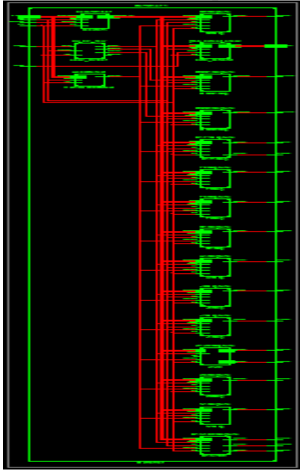
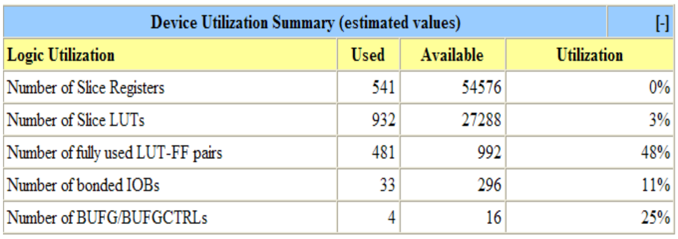


Figure 21: rtl schematic of stm-1 de-framer

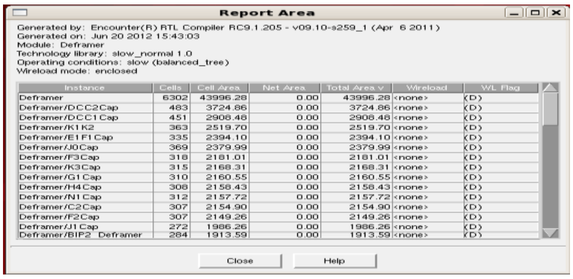
Figure 22 shows the device utilization summary of STM-1 De-Framer implemented on Spartan 6 FPGA in Xilinx ISE 13.2.



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	541	54576	0%
Number of Slice LUTs	932	27288	3%
Number of fully used LUT-FF pairs	481	992	48%
Number of bonded IOBs	33	296	11%
Number of BUFG/BUFGCTRLs	4	16	25%

Figure 22: device utilization summary of stm-1 de-framer

The fig 23 shows the area report of STM-1 De-Framer generated using Cadence RTL Compiler. The total area occupied by the STM-1 De-Framer is 43996um^2.

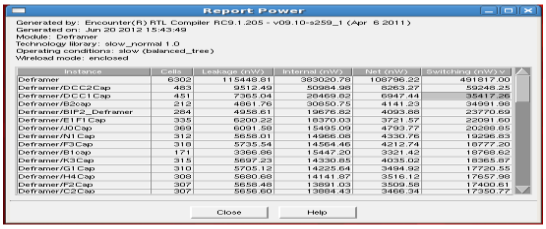


```
Generated by:  Encounter(R) RTL Compiler RC9.1.205 - v09.10-s259_1 (Apr  6 2011)
Generated on:  Jun 20 2012 15:43:03
Module:  Deframer
Technology library:  slow_normal 1.0
Operating conditions:  slow (balanced_tree)
Wireload mode:  enclosed
```

Instance	Cells	Cell Area	Net Area	Total Area v	Wireload	WL Flag
Deframer	6302	43996.28	0.00	43996.28	<none>	(D)
Deframer/DCC2Cap	483	3724.86	0.00	3724.86	<none>	(D)
Deframer/DCC1Cap	451	2908.48	0.00	2908.48	<none>	(D)
Deframer/K1K2	363	2519.70	0.00	2519.70	<none>	(D)
Deframer/E1F1Cap	335	2394.10	0.00	2394.10	<none>	(D)
Deframer/J0Cap	369	2379.99	0.00	2379.99	<none>	(D)
Deframer/F3Cap	318	2181.01	0.00	2181.01	<none>	(D)
Deframer/K3Cap	315	2168.31	0.00	2168.31	<none>	(D)
Deframer/G1Cap	310	2160.55	0.00	2160.55	<none>	(D)
Deframer/H4Cap	308	2158.43	0.00	2158.43	<none>	(D)
Deframer/N1Cap	312	2157.72	0.00	2157.72	<none>	(D)
Deframer/C2Cap	307	2154.90	0.00	2154.90	<none>	(D)
Deframer/F2Cap	307	2149.26	0.00	2149.26	<none>	(D)
Deframer/J1Cap	272	1986.26	0.00	1986.26	<none>	(D)
Deframer/BIP2_Deframer	284	1913.59	0.00	1913.59	<none>	(D)

Figure 23: area report of stm-1 de-framer

Fig 24 shows the power consumption of STM-1 De-Framer generated using RTL Compiler. The total power consumed by STM-1 De-Framer is 0.61mW.



```
Generated by:  Encounter(R) RTL Compiler RC9.1.205 - v09.10-s259_1 (Apr  6 2011)
Generated on:  Jun 20 2012 15:43:49
Module:  Deframer
Technology library:  slow_normal 1.0
Operating conditions:  slow (balanced_tree)
Wireload mode:  enclosed
```

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW) v
Deframer	6302	115448.81	383020.78	108796.22	491817.00
Deframer/DCC2Cap	483	9512.49	50984.98	8263.27	59248.25
Deframer/DCC1Cap	451	7365.04	28469.82	6947.44	35417.26
Deframer/B2cap	212	4861.76	30850.75	4141.23	34991.98
Deframer/BIP2_Deframer	284	4958.61	19676.82	4093.88	23770.69
Deframer/E1F1Cap	335	6200.22	18370.03	3721.57	22091.60
Deframer/J0Cap	369	6091.58	15495.09	4793.77	20288.85
Deframer/N1Cap	312	5658.01	14966.08	4330.76	19296.83
Deframer/F3Cap	318	5735.54	14564.46	4212.74	18777.20
Deframer/B1cap	171	3366.86	15447.20	3321.42	18768.62
Deframer/K3Cap	315	5697.23	14330.85	4035.02	18365.87
Deframer/G1Cap	310	5705.12	14225.64	3494.92	17720.55
Deframer/H4Cap	308	5680.68	14141.87	3516.12	17657.98
Deframer/F2Cap	307	5658.48	13891.03	3509.58	17400.61
Deframer/C2Cap	307	5656.60	13884.43	3466.34	17350.77

Figure 24: power report of stm-1 de-framer

Fig 25 shows the physical layout of STM-1 Framer generated using Cadence SOC Encounter.

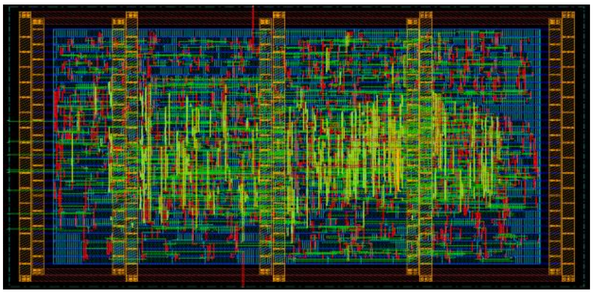


Figure 25: physical layout stm-1 framer

Fig 26 shows the physical layout of STM-1 Framer generated using Cadence SOC Encounter.

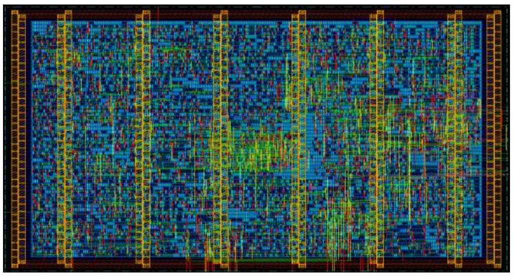


Figure 26: physical layout stm-1 de-framer

V. Conclusion

The development of the STM-1 framer core offers encapsulation of Plesiochronous Digital Hierarchy standards or support Asynchronous Transport Mode. It can be used for moving voice and data. The STM-1 Framer has been designed and can accommodate 2340 user data's that will be multiplexed in a single Frame. The STM-1 De-Framer has been designed and will extract the overhead bytes. The De-framer checks whether the obtained data is error free or not and also check the overhead bytes to determine the errors in the Section Overhead.

VI. Acknowledgement

We acknowledge Dr. V. Venkateswarlu, Principal, UTL Technologies for his guidance and suggestion and UTL Technologies for providing lab facility during the design and implementation.

References

[1] ITU-T Recommendation G.707/Y.1322, Network node interface for Synchronous Digital Hierarchy (SDH), Telecommunication Standardization Sector of ITU. Geneva, January 2007.

[2] Zhang Xiaoru and Zeng Lieguang, "An SDH STM-1 termination IC", ASIC 2nd International Conference, pp 179-182, 1996.

[3] ITU-T Rec. G.783, Characteristics of synchronous digital hierarchy (SDH) equipment functional blocks, Telecommunication Standardization Sector of ITU. Geneva, March 2006.

[4] G. Cariolaro, G. Iudicello, "Signal theory for SONET STM-1" Telecommunication symposium 1990. Its'90 symposium record, Sep 1990, pp 26 - 32.

[5] D. Bajic, J. Stojanovic, FTN Novi Sad, Fruskogorska, Mobtel Beograd, "Frame Alignment Monitoring For STM-1 Frame", IEEE International Conference on Communications, 2001. ICC 2001, Vol. 10, pp. 2953- 2957.

[6] Srini W. Seetharam, Gary J. Minden, Joseph B. Evans, "A parallel SONET scrambler/descrambler architecture", Circuits and Systems, ISCAS, IEEE International Symposium, May 1993, vol 3, pp. 2011 - 2014.

[7] R. Clauberg "Data aggregation architectures for single chip SDH/SONET framers", IBM J. RES & DEV. VOL. 47 NO.2/3 March/May 2003.

[8] D.W. Choi, "Frame Alignment in a Digital Carrier System - A Tutorial", IEEE Comm. Mag., pp. 47-54, Vol. 28, February 1990.

[9] Cisco Systems, "Synchronous Digital Hierarchy (SDH) Graphical Overview", Oct 2006.

[10] Wei-Zen Chen, Guan-Sheng Huang, "A Parallel Multi- pattern PRBS Generator and BER Tester for 40+ Gbps Serdes Applications", IEEE Asia-Pacific Conference on Advanced System Integrated Circuits, Aug 2004, pp. 318-321.