DESIGN AND IMPLEMENTATION OF MULTI CHANNEL FRAME SYNCHRONIZATION IN FPGA

G. Prasad gurram_p@yahoo.com and N. Vasantha vasantha54321@gmail.com

ABSTRACT

Data from spacecraft are transmitted to stations on earth in serial and encoded form. The transmission channels are generally noisy and the encoding enables to correct errors during transmission. Lossless real time data acquisition of such spacecraft data is very crucial for providing data to meet the user need. Frame Synchronization is one method to ensure that the data received at ground is lossless. In serial frame synchronization that is when each frame starts with an identical sync code, false sync due to replicas of the code randomly generated by the data is completely eliminated by appropriate frame synchronization logic. In this paper a method of detection of the frame synchronization code from the received PCM data, a synchronization confirmation as a guard by detecting a repetition of the frame synchronization codes, a confirmation of the subsequent frame identification signal based on the synchronization confirmation, and separation of the received multiplex data into respective fields is presented.

1. INTRODUCTION

Frame synchronization is an essential element in digital communication systems. It determines the boundary between data frames so that the information can be recovered correctly from a stream of data. In modern communication systems data is not transferred as a simple stream of bits or bytes but in terms of frames or packets. In time multiplexed pulse code modulated (PCM) telemetry binary data signals from several sources are grouped into frames which have to be identified at the receiver in order to demultiplex the data. Frame synchronization is obtained by inserting in series (e.g. at the beginning of each frame), or in parallel (i.e. on a separate sync channel), a frame sync code (FSC). At the receiver, the frame synchronizer correlates the received signal with its own replica of the FSC for different bit shifts, until synchronization is acquired. The synchronization is then maintained by verifying the repetition of this code at each frame provided the frame length is fixed.
The frame synchronizer correlates the received FSC with the reference synchronization code. The main problem of false sync due to replicas or almost replicas of the FSC generated by the random data is avoided in this type of serial frame synchronization. Since the frame length is fixed the probability of false sync can be reduced by verifying the occurrences of the FSC on successive frames. In serial frame synchronization a portion of each frame (e.g. the first k bits of the frame) consists of a code sequence, repeated at each frame. After frame sync detection takes place and synchronization, individual measurands are identified according to the frame location. The decommutator identifies and extracts embedded asynchronous data stream (EADS) words. Thus frame synchronizer is a very crucial subsystem in the satellite data acquisition unit of satellite ground station. In this paper two channel frame synchronization logic is designed and implemented on a Stratix FPGA.

The hardware design consists of two major modules 1) LVDS receiver logic 2) Frame Synchronization logic which in turn has modules like Data Simulator, Flywheel & Frame Sync Strategy and Bit slip & correction logics. This design is realized in AHDL and VHDL and the software used is Altera’s Quartus. Decommutator will identify and separate the individual parameters from the incoming satellite PCM stream after frame synchronization. The validation of the modules is done with an inbuilt data simulator. The frame sync codes are selectable for catering to different satellites The FPGA used to implement the design is Stratix EP1S25F1020C5 which has a capacity of 25K logic elements, 6 PLLs, 10 DSP blocks, 1,944,576 Memory elements and 706 I/Os. Simulations and synthesis is done by the Quartus software tool provided by Altera. After successful total compilation the program output file is loaded into the FPGA using a JTAG connector.

2. QUARTUS SOFTWARE AND ALTERA HARDWARE DESCRIPTIVE LANGUAGE

The Altera Quartus II design software is a multiplatform design environment that easily adapts to specific needs in all phases of FPGA and CPLD design. Quartus II software delivers the highest productivity and performance for Altera FPGAs, CPLDs, and Hard Copy ASICs.

Quartus II software delivers superior synthesis and placement and routing, resulting in compilation time advantages. Compilation time reduction features include, Multiprocessor support, Rapid Recompile, Incremental compilation. Quartus II Analysis and Synthesis, together with the Quartus II Fitter, incrementally compiles only the parts of your design that change between compilations. By compiling only changed partitions, incremental compilation reduces compilation time by up to 70 percent. For small engineering change orders (ECOs), the Rapid Recompile feature maximizes your productivity by reducing your compilation time by 65 percent on average, and improves design timing preservation.

AHDL is a proprietary digital Hardware Description Language (HDL) from Altera Corporation for programming their Complex Programmable Logic Devices (CPLD) and Field Programmable Gate Arrays (FPGA). This language has an Ada programming language-like syntax and similar operation to VHDL or Verilog. It is supported by Altera’s Quartus and Max+ series of compilers. An advantage of AHDL is that all language constructs are synthesizable. AHDL is to Verilog much as assembly language is to a higher-level programming language: in AHDL, you have more control.
3. FPGA 90NM STRATIX EP1S25F1020C5

The Stratix FPGA is used to implement the four modules i.e. Decommutator, on chip memory, external fifo interface & control logic and PCI IP Master core. Stratix devices contain a two-dimensional row- and column-based architecture to implement custom logic. A series of column and row interconnects of varying length and speed provide signal interconnects between logic array blocks (LABs), memory block structures, and DSP blocks. The logic array consists of LABs, with 10 logic elements (LEs) in each LAB. An LE is a small unit of logic providing efficient implementation of user logic functions. LABs are grouped into rows and columns across the device.

M512 RAM blocks are simple dual-port memory blocks with 512 bits plus parity (576 bits). These blocks provide dedicated simple dual-port or single-port memory up to 18-bits wide at up to 318 MHz. M512 blocks are grouped into columns across the device in between certain LABs. M4K RAM blocks are true dual-port memory blocks with 4K bits plus parity (4,608 bits). These blocks provide dedicated true dual-port, simple dual-port, or single-port memory up to 36-bits wide at up to 291 MHz. These blocks are grouped into columns across the device in between certain LABs. M-RAM blocks are true dual-port memory blocks with 512K bits plus parity (589,824 bits). These blocks provide dedicated true dual-port, simple dual-port, or single-port memory up to 144-bits wide at up to 269 MHz. Several M-RAM blocks are located individually or in pairs within the device’s logic array.

Digital signal processing (DSP) blocks can implement up to either eight full-precision 9 × 9-bit multipliers, four full-precision 18 × 18-bit multipliers, or one full-precision 36 × 36-bit multiplier with add or subtract features. These blocks also contain 18-bit input shift registers for digital signal processing applications, including FIR and infinite impulse response (IIR) filters. DSP blocks are grouped into two columns in each device. Each Stratix device I/O pin is fed by an I/O element (IOE) located at the end of LAB rows and columns around the periphery of the device. I/O pins support numerous single-ended and differential I/O standards. Each IOE contains a bidirectional I/O buffer and six registers for registering input, output, and output-enable signals. When used with dedicated clocks, these registers provide exceptional performance and interface support with external memory devices such as DDR SDRAM, FCRAM, ZBT, and QDR SRAM devices. High-speed serial interface channels support transfers at up to 840 Mbps using LVDS, LVPECL, 3.3-V PCML, or Hyper Transport technology I/O standards.

4. LOW VOLTAGE DIFFERENTIAL SIGNALING.

When considering I/O standards, lower voltage and non-terminated standards typically yield lower power ratings. Any reduction in voltage has a quadratic effect on power. Static power is significant for terminated standards.

When the I/O buffer drives a high signal, the I/O supply voltage delivers power consumed in the external termination resistors. When driving a low signal, the chip dissipates power from the external termination voltage. Use the lowest driver strength I/O setting that meets the speed and waveform requirements to minimize I/O power when using terminated standards. Differential I/O standards such as LVDS (Low voltage differential signaling) with low switching voltage (typically 350mv) provide lower power consumption, better noise margin, smaller electromagnetic interference and overall better performance.

In our design 3.3V LVDS chips were used to interface to the differential inputs received from the Satellite data reception system and generate a single ended Low Voltage signals.
The LVDS receivers used are SN65LVDS1, which is single, low-voltage, differential line receivers in the small-outline transistor package. The outputs comply with the TIA/EIA-644A standard and provide a minimum differential output voltage magnitude of 247 mV into a 100-W load at signaling rates of 400 Mbps.

![Fig 1. LVDS RECEIVERS CONNECTED TO FPGA](image)

5. LOGIC IMPLEMENTED IN THE FPGA

The total design comprises of Data Simulator logic, Frame Synchronization and associated logic, are realized in the Stratix FPGA as shown in Fig 2.

![Fig 2. Block Diagram implemented in FPGA](image)
6. DATA SIMULATOR LOGIC

The Data Simulator is to simulate the data. Data Simulator logic generates the FS code, with variable line count in the aux field and fixed video data pattern in a single frame and for two channels. The required frequency is derived from a crystal oscillator. The twin channel’s serial data and clock are connected to a RJ45 connector. Cat 6 cable is used to connect the simulator outputs to the Frame synchronization logic inputs.

7. FRAME SYNCHRONIZATION LOGIC

Time Division Multiplexed (TDM) data is decommutated by first locating a fixed pattern and then determining that the pattern repeats at fixed intervals. The pattern is recognized by a correlator logic as shown in fig 3, that receives up to n sequential bits and compares them to a programmed or fixed reference pattern, using a programmable mask to exclude don’t care bit positions. Because telemetry data is often transmitted or stored imperfectly due to system noise constraints, absolute correlation is not always possible. When determining the sync pattern location, we must often allow a programmable number of conflicts to occur in an otherwise acceptable pattern. This number is referred to as the “sync pattern tolerance”. In our design the tolerance bits are from zero to seven bits. The n bit correlation function is realized in the FPGA. This correlation logic compares the digital pattern stored in a reference pattern register with the input data samples stored in the data input register. The number of matches is calculated for every clock cycle. This number is called the correlation sum and is compared with a threshold value. When the correlation score is >= the Threshold a Frame Sync Detect pulse is generated. To prevent false detects a flywheel logic is included with strategy which has a search, check and lock modes. When two consecutive syncs are detected the logic will change from search to check and later to lock mode. Likewise when a sync loss occurs the logic will change from lock to check and when two consecutive sync loss occur the logic will revert to search mode.

In our design the reference pattern register, mask register, correlation summation logic, threshold logic are realized using macrocells. The LUTs in the macrocell can be updated rapidly while the FPGA is in full operation. The design has been done for a frame sync pattern of 128 bit.

Eg: For a 128 bit frame sync code, the correlator compares all the 128bits of the input sequence with the 128bit stored in the reference data register. On each clock pulse, the incoming bit is clocked into the leftmost register and all bits are shifted by one register. Comparison is made bit by bit between the data input register and reference data register and the correlation sum is computed. A perfect match will give 1111 output at the correlation output. While in case of 1 bit error the correlation sum will be 1110 and it goes in this way until the tolerable limit of 0000 (i.e. 15 errors are tolerable for 127 bit frame sync code). If the number of errors are more than 15, than a loss pulse is generated to indicate that the frame sync code does not match with the reference pattern. If successive loss pulses are detected than the logic will go to loss mode.
8. FLY WHEEL AND FRAME SYNC STRATEGY

The flywheel logic provides reliable frame synchronization and data decommutation by using programmable frame sync strategy counters, a programmable bit-slip window, and programmable bit error tolerances. Frame synchronization occurs during all three states of SEARCH, VERIFY, and LOCK. The design incorporated for our application is a combination of Fixed, and Adaptive, strategies as shown in Fig 4. The Fixed strategy tests for a programmable number of good or bad sync patterns to determine when state changes should occur. The adaptive strategy is for use with noisy data when the sync pattern error tolerance is enabled.

The fixed strategy tests the number of errors in the sync pattern and uses this value, relative to previous values, to qualify a sync pattern. Four states have been identified as VERIFY, SEARCH, LOCK and CHECK for the frame sync logic to pass through. Initially the frame sync logic will be in check mode. In case 1 program for the sync strategy to transfer from SEARCH to VERIFY, or in case 2 directly to LOCK. When the transfer is to VERIFY, you can program additional flywheel frames (no acceptable sync pattern found in frame) before the strategy returns to SEARCH, and the number of consecutive good frames before the synchronizer will go into LOCK. This adds confidence that the real sync pattern has been found. In LOCK, you can program the number of consecutive bad frames before the synchronizer leaves LOCK. The transfer from LOCK is directly to SEARCH, thus allowing rapid sync reacquisition.

The Fixed strategy starts in SEARCH, where the bit stream is scanned for the programmed sync pattern. Upon detecting a good pattern (as qualified by the sync error tolerance) the synchronizer enters the VERIFY state. A window is generated at the end of the frame by the words per frame count and the bit slip window. If a good pattern is found within this window the consecutive Good Frames from VERIFY to LOCK count (2) is tested and, if equal, the synchronizer will enter LOCK. The count of zero allows the strategy to transfer directly from SEARCH to LOCK, bypassing the VERIFY state. When a good pattern is not found within the window, the consecutive Bad Frames from VERIFY to SEARCH (1) is tested and, if equal, the synchronizer will return to SEARCH. In LOCK the correlator pattern is tested in the End Of Frame window. If the number of consecutive frames with bad patterns matches the LOCK to SEARCH count (1), the synchronizer will return to SEARCH as shown in Fig 5.
Fig 4. Flow chart for the Frame synchronization logic

The flow chart shown in Fig 4. Indicates the sequence followed in detecting the frame synch and also in case the frame sync pattern is not detected, it goes into loss mode. The Frame synchronization States are presented below in Fig 5.
The flywheel and frame sync strategy allows us to record all the data received from the satellite irrespective of the frame sync state into the system hard disk for further processing. The states will also ensure that the data recorded in the storage may still contain valid data even if the frame sync is not detected since the flywheel logic is in place and in synchronization with the frame window. Thus the imagery data may be good even in case the Frame sync data does not match with the reference pattern. This states and strategy will avoid total frame loss thus improving the efficiency of data reception and enable post processing of data after real time data reception from the satellite. Thus the error tolerance will provide flexibility to the frame sync detection by enabling the allowable number of bit errors in the frame sync code and still acquire the data from the space craft. So to maximize the probability of synchronization lock with a particular acceptable errors and minimizing the lock probability if the data has got more errors will be the best suitable method for real time data acquisition from satellites.

The adaptive strategy functions as follows. In SEARCH, the frame sync logic will begin by searching for the frame sync pattern that meets the programmed frame sync error tolerance. When a matched pattern occurs, the number of detected errors in the frame sync pattern will replace the contents of the sync error tolerance register. The strategy sequencer remains in SEARCH as established by the words per frame count. The input stream is tested continually for a pattern with fewer errors than those stored, and if a sync pattern with less errors is found before end of frame, the words per frame counter is re-initiated and the new pattern error count replaces the sync pattern error tolerance. When a frame passes that does not contain a pattern that is better than the current sync pattern tolerance, and the pattern at the end of frame is acceptable, the frame sync unit will advance to VERIFY. The strategy now works as in fixed mode, with the error tolerance equal to the number of errors detected in the best pattern encountered in SEARCH. If the required number of good patterns is found in the frame sync window, the frame sync unit will advance to LOCK. If the frame sync errors are greater than the tolerance established (when leaving SEARCH), for the consecutive number of frames specified by the VERIFY to SEARCH count (when in VERIFY), or the LOCK to SEARCH count (when in LOCK), the frame sync unit reverts back to SEARCH mode. When SEARCH mode is re-entered, the sync error tolerance is set to the initially programmed value and the pattern search proceeds as previously described.

**Fig 5.** Frame Synchronization states

```
Verify >=2
Search >1 1
Lock >1
Check 1
Frame sync detected.
Frame sync loss.
```
9. BIT SLIP AND CORRECTION

Frame synchronization normally consists of first looking for a pattern anywhere in the stream while in SEARCH, and then applying a frame length window to avoid false sync patterns that may occur in the data stream. The frame is normally of fixed length and the pattern should recur at a specific bit interval. However, when the input to the bit synchronizer lacks sufficient transitions, and/or contains excessive noise, the bit synchronizer clock may drift and produce excessive or insufficient clock pulses during the frame. This results in a Frame Bit Slip condition, which can result in the loss of synchronization.

Bit Slip Correction permits to program the frame synchronizer to accept sync patterns occurring in bit positions adjacent to the expected position in the frame. The sync pattern may occur exactly at the expected location (one bit window); one bit position early or late (three bit window), two bit positions prior or after the expected position (five bit window), or three bit positions prior or after the expected position (seven bit window), and still be detected as an acceptable sync pattern. This feature enables the unit to maintain synchronization during excessive noise bursts or data dropouts in the input stream when the bit synchronizer cannot maintain synchronization with the PCM stream.

In our design the Bit Slip Window of 3 bits is designed (i.e. +/- 1 bit slip) may correct for the resultant one bit sync error. However, if the sync pattern is not properly chosen (when the telemetry encoding was designed), the pattern, shifted by several bit positions with random data, may look like an acceptable pattern (especially when the sync pattern error tolerance is large) and the results may resynchronize the frame erroneously. Therefore, the frame synchronizers must provide a programmable sync strategy that will achieve rapid synchronization while, at the same time, guarding against false sync patterns produced by the variable patterns encountered in the PCM stream.

RESULT

For a two channel frame synchronization and associated logic 617 logic elements i.e. 2.83% of the Stratix FPGA capacity and 7 of the FPGA I/O were used to implement. Operation frequency of 150 Mbits/sec was achieved. The power consumed was around 90milliamperes.

**Fig 6.** SIMULATION RESULTS
CONCLUSION

The multi channel Frame Synchronization in FPGA, and associated logic designed and developed is suitable for satellite data acquisition systems in the Ground segment. Since the hardware is compact and can be housed in any server this forms a embedded hardware and is thus suitable for fixed and mobile applications also. The frequency of operation achieved is 150Mbits/second thus it can cater to high speed data acquisition. Since the major modules are incorporated into the FPGA a power reduction of nearly 15% is achieved in the design. The frame synchronized data is fed to the decommutator which identifies and extracts embedded asynchronous data stream (EADS) words. The extracted data is written in the FIFO memory and to the host. The logic is validated with an inbuilt simulator so that the total chain involved in the design is completely tested. For future missions when more than two channel frame synchronization is required the same can be designed as per the requirements or multiple cards can be used.

REFERENCES

[9]. Christophe Cunat and Emmanuel Boutillon,(2007) Member IEEE. “Simplified Hardware Bit Correlator”.


