International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 9, September - 2013

Auto-Correlation of a Real-Time Remote Sensing Satellite Data

C. Yashitha
M.Tech., (Digital Systems and Computer Electronics)
Sreenidhi Institute of Science and Technology, Ghatkesar
Hyderabad, Andhra Pradesh

Abstract - **Remote Sensing Satellite images the earth with the help of Charge Coupled Devices (CCDs), and thus the EM waves are converted to an electric signals of one's and zero's to form a digital data in pixel form. When the data is being acquired at the ground station, identifying the frame synchronization code of the satellite data pattern is of prime importance. This includes correlating the satellite data pattern called frame sync code with a reference signal, before being processed and archived. To ensure valid data acquisition in real time the auto correlation of the incoming data with the same reference signal is performed in real time and archived on the data recorder.**

Keywords - Flywheel, frame sync, raw detect and loss pulse.

I. INTRODUCTION

Auto correlation is a mathematical representation of the degree of similarity between a given time series and a lagged version of itself over successive intervals of time. It is similar to that of calculating the correlation between two different time series, except that the same time series is used twice as once in its original form and once lagged one or more time periods. It can also be defined as cross-correlation of a signal with itself. Otherwise, it is the similarity between observations as a function of the time separation between them. It is a measure of how well a signal matches a time shifted version of itself, as a function of the amount of time shift. Correlation is to indicate a predictive relationship which is exploited in practice. Autocorrelation is useful for finding repetitive patterns in a signal, such as determining the presence of a periodic signal which has been buried under noise or identifying the missing fundamental frequency in a signal implied by its harmonic frequencies. Autocorrelation involves only one signal and provides information about the structure of the signal or its behavior in the time domain. The analysis of correlation is used to measure the spatial resolution of an image receptor with uniform white noise as the input, to the periodic patterns in noisy data and to characterize the similarity patterns in data compression.

While carrier and symbol clock may have been well established in a data transmission system, the boundary of long symbols or packets may not be known to a receiver. Such synchronization requires searching for some known pattern or known characteristic of the transmitted waveform to derive a phase error (which is typically measured in the number of sample periods or dimensions in error). Such synchronization, once established, is only lost if some kind of catastrophic failure of other mechanisms has occurred (or dramatic sudden change in the channel) because it essentially involves counting the number of samples from the recovered timing clock. Correlation functions are many which are denoted by r or *p*, used in the measurement of correlation degree. The most common of these is the Pearson Correlation coefficient. It is sensitive only to linear relationship between two variables. These two variables may exist even if one is a nonlinear function of the other. Other than this Pearson correlation coefficient, more correlation coefficients have been developed to be more sensitive to the nonlinear relationships.

Autocorrelation, sometimes is a correlation coefficient. However, it can be correlation of two different values of the same variable at different times instead of correlation between tow different variables. Given a signal, f(t) the continuous autocorrelation $R_{ff}(T)$ is most often defined as the continuous cross-correlation integral of f(t) with itself, at lag *T*.

$$R_{ff}(T) = \int_{-\infty}^{\infty} f(t+T) f^*(t)\, dt$$

where, f^* represents the complex conjugate and for a real function, $f^*=f$.

Auto correlation maintains all the properties of cross-correlation and is itself, periodic with the same period as autocorrelation is a specific type of cross correlation as mentioned above. The sum of the autocorrelation of each function separately is given as the autocorrelation result of the sum of two completely uncorrelated functions (for all *T*, the cross correlation is zero). These autocorrelation functions are exploited for predictions auto correlated time series is predictable and sometimes known because future values depend on past and present values. There are various tools for assessing the autocorrelation of time series out of which autocorrelation function is one apart from time series plot and lagged scatter plot.

The paper aims to design and develop an auto-correlator using a frame synchronizer which will ensure error free data acquisition and archiving high speed data. This will also validate individual elements of the Satellite Ground station before the actual data is being acquired on a daily basis. This pattern is to test the functionality of hardware and the system interfaces everyday to ensure the satellite data is received and archived properly. Due to huge number of multiplication and addition calculations in the correlator, a simple microprocessor cannot meet the real-time requirements. Therefore, a useful FPGA platform for high performance computing and real-time interactive signal processing is used to implement the correlator.

II. IMPLEMENTATION OF THE PROJECT

The hardware component used in the implementation of the correlator is FLEX 10K. Every FLEX 10K device consists of an embedded array for the implementation of memory and specialized logic functions, and also a logic array to implement general logic. Each embedded array consists of a series of EABs, which when implementing memory functions, each EAB provides 2,048 bits and in turn can be used to create ROM, RAM, dual-port RAM, or first-in first-out (FIFO) functions. When implementing a logic, each EAB can contribute 100 to 600 gates towards complex logic functions, such as multipliers, state machines, microcontrollers, and DSP functions. EABs can be used either independently or multiple EABs can be combined to implement larger functions. The logic array consists of logic array blocks (LABs). Each LAB contains eight LEs(Logic Elements) and a local interconnect. An LE consists of a 4-input look-up table (LUT), a programmable flip-flop, and dedicated signal paths for carry and cascade functions. Eight LEs together can be used to create medium-sized blocks of logic such as 8-bit counters, address decoders, or state machine or combined across LABs to create larger logic blocks. Each LAB represents up to 96 usable gates of logic. FLEX 10K devices provide six dedicated inputs that drive the flip-flops' control inputs to ensure the efficient distribution of high-speed, low-skew (less than 1.5 ns) control signals. These signals use dedicated routing channels that provide shorter delays and lower skews than the Fast Track Interconnect. Each of the four dedicated inputs drive four global signals respectively. These four global signals can also be driven by internal logic, providing an ideal solution for a clock divider or an internally generated asynchronous clear signal that clears many registers in the device.

The EAB provides advantages over FPGAs, which implement on-board RAM as arrays of small, distributed RAM blocks. These FPGA RAM blocks contain delays which are less predictable as the size of the RAM increases. In addition, FPGA RAM blocks are prone to routing problems because small blocks of RAM must be connected together to make larger blocks. In contrast, EABs can be used to implement large, dedicated blocks of RAM that eliminate these timing and routing concerns. If needed, all EABs in a device can be cascaded to form a single RAM block. EABs can be cascaded to form RAM blocks of up to 2,048 words without impacting timing. Altera's software automatically combines EABs to meet a designer's RAM specifications.

The LE, the smallest unit of logic in the FLEX 10K architecture, has a compact size that provides efficient logic utilization. Each LE contains a four-input LUT, which is a function generator that can quickly compute any function of four variables. The programmable flip-flop in the LE can be configured for D, T, JK, or SR operation. The clock, clear, and preset control signals on the flipflop can be driven by global signals, general-purpose I/O pins, or any internal logic. For combinatorial functions, the flipflop is bypassed and the output of the LUT drives the output of the LE.

Max+Plus II offers a full spectrum of logic design capabilities and a variety of design entry methods for hierarchical designs, powerful logic synthesis, timing-driven compilation, partitioning, functional and timing simulation, linked multi-device simulation, timing analysis, automatic error location, and device programming and verification. The software both reads and writes Altera Hardware Description Language (AHDL) files besides reading Xilinx net list files and writing Standard Delay format (SDF) files for a convenient interface to other industry-standard CAE software. It offers many features and commands such as opening files, entering device, pin, and logic cell assignments and compiling the current project. The design editors like Graphic, Text, and Waveform, and auxiliary editors like Floorplan and Symbol editors share numerous features. User can enter, edit, and delete the types of resource, device and parameter assignments that control project compilation, including logic synthesis, partitioning, and fitting irrespective of project design file or application window is open or not. The software allows the user to open multiple design files and transfer information between them while compiling or simulating another project.

The timing analyzer helps to tag multiple source and destination nodes so that they are included in an analysis after which a source or destination node can be selected and all delay paths associated with it are listed. An automatic error location and extensive documentation on error and warning messages make design modifications quick and easy. The compiler applies a variety of techniques to increase the efficiency of the project and minimize device resource usage. If the project is too large to fit into a single device, it partitions into multiple devices from the same device family, thus reducing the number of connections between devices. The simulator tests the logical operation and internal timing of the project and allows to model a circuit design before it is programmed into a device.

III. DESIGN OF A CORRELATOR

Digital Correlator is a digital logic design that correlates the input bit sequence with a standard reference sequence (Frame Sync). This design consists of two independently clocked shift registers, a reference holding latch, and an independently clocked digital summing network. It correlates the received frame Sync Code with the reference synchronization code. Since the frame length is fixed, the probability of false sync can be reduced by verifying the occurrences of the FSC on successive frames. The incoming data is compared with the reference frame sync code. When the correlation score is >= the Threshold a Frame Sync Detect pulse is generated. Otherwise a loss pulse is generated.

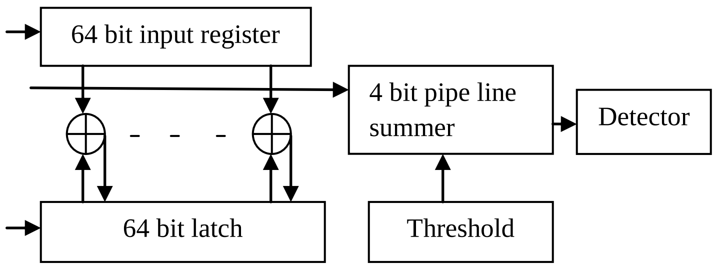

Figure 1: Design of A Correlator

In the design of the correlator shown in figure 1, the 64-bit transparent latch is controlled by an input load. A HIGH level on the load input causes the latch to be transparent, allowing the contents of the register to be applied directly to the correlator array. When the load input is LOW, the data in the latch is held, so that the register may be loaded with a new correlation reference without affecting the current reference value stored in Latch. Each bit in the input register is Exclusive ORed with the corresponding bit in the latch, implementing a single bit multiplication at each bit position. This 64-bit vector is applied to a pipelined digital summer which contains four pipeline stages and calculates the total number of ones in the vector (the correlation score). The output of the summer represents the count of number of matching positions in the input data stream. In the design the reference pattern register, correlation summation logic and threshold logic are realized using macrocells. The Look-Up Tables in the macrocell can be updated rapidly while the FPGA is in full operation. The code below is written for assigning a threshold value which is used to control the allowable number of errors in the data stream of each frame. The threshold value can be adjusted by using the three bit threshold value code given below.

```
bin_vector_value[]=bin_vector_reg[];
thresh_reg[]=thresh_ip[];
delay_reg[0]=in_data_reg[63];
delay_reg[9..1]=delay_reg[8..0];
in_data_op=delay_reg[9];

if thresh_reg[] == 0 & sum_op[] == 64 then
raw_reg = TRUE;
elsif thresh_reg[] == 1 & sum_op[] >= 63 then
raw_reg = TRUE;
elsif thresh_reg[] == 2 & sum_op[] >= 62 then
raw_reg = TRUE;
elsif thresh_reg[] == 3 & sum_op[] >= 61 then
raw_reg = TRUE;
else raw_reg = FALSE;
end if;
```

The threshold holds the value of number of permissible errors in the input data. A maximum of three errors are allowed in the frame which can be adjusted by the 2 bit threshold value. This output detects the arrival of loss pulse if error occurred is more than the error bits in the threshold value. If there is no error which falls under the threshold value, a raw detect pulse occurs indicating the data received is valid. A valid data also occurs when the error occurred is less than or equal to the adjusted threshold value. Otherwise, a loss pulse is detected instead of a raw detect which indicates an error has occurred in the respective frame.

As the frame sync code for any remote sensing satellite is unique and fixed, it can be stored in memory location and can be called for every start of frame. The frame sync code contributes 8 words out of the total frame length of 2400 words. The description is for only one channel and the same procedure is repeated for the second channel too. The code below is for the 8 words frame sync code for one channel and the length of it is one byte each. We use a 74195IC as an up counter and the code is as following.

```
LNCTR :lpm_counter with (lpm_width=12,
lpm_direction = "up",lpm_modulus=2400);
LFCTR :lpm_counter with (lpm_width=24,
lpm_direction = "up");
```

The start of frame code can be loaded into memory as shown below.

```
a1[]=H"0c";     % loading frame sync register with fs-code %
a2[]=H"28";
a3[]=H"f2";
a4[]=H"2c";
a5[]=H"ed";
a6[]=H"7d";
a7[]=H"0e";
a8[]=H"24";
```

The same procedure can be repeated to generate another frame sync code for the second channel.

IV Flywheel and its Strategy

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. To prevent false detects a flywheel logic is included with strategy which has a search, check and lock modes as shown in figure 2. 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. For a fixed strategy of flywheel search mode is the initial state where the data stream is scanned for the given programmed sync pattern. When an expected pattern is detected, the synchronizer enter into verify state. A window is generated at the end of the frame and a bit slip window. If a good pattern is found within the window the state is shifted from verify to lock mode. Otherwise, the identification of bad pattern shifts the state from verify to search. In lock state the correlator pattern is tested at the end of frame window. The code below is for generating a window at the end of frame and bit slip window at 19198 bit which uses fwf and dg2 as the D-filpflops and fcout is the output. The figure 2 below describes the states of flywheel.

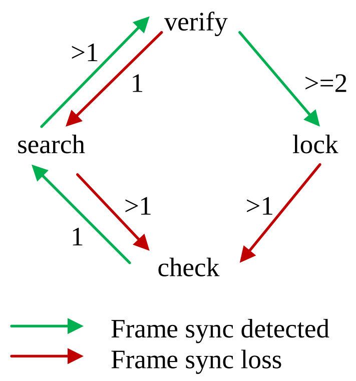

Figure 2: Flywheel and its Strategy

```
fwf[].d=fwf[].q+1;
fw=(Fwf[]==19198);
fcout[]=fwf[];
dg2.clk=clk;
dg2.d=fw;
```

The search mode looks for a possible frame sync pattern. When a known pattern is tentatively identified, a window is set at the expected time of reoccurrence of the sync pattern, and is checked for several frames after which the synchronizer advances to lock. In the lock mode, the mode continues to look for the frame sync pattern in the sync window and will only revert to a previous mode if the sync pattern fails to occur in the window for a given number of frames. Once the frame synchronization is established, commutated and super commutated measures can be identified since the position of the data values are relatively known to the frame sync pattern. Finally, after being in lock, if an expected frame sync pattern is not detected, the state is the converse of the "Verify" mode. The frame sync strategy and flywheel allows to record all the data received from the satellite irrespective of the frame sync state into the system hard disk for further processing. These states of the flywheel avoid total frame loss, improving the efficiency of the data reception and enable post processing of the data after real time data reception from the satellite.

V. Simulation Results

The entire code for the correlation with frame synchronization is written in Altera's Hardware Description Language (AHDL) and implemented using FLEX 10k device with the help of Altera's Maxplus II software. The entire design occupies approximately 30% of the total device resource and achieved a frequency up to 70MHz for each channel.

The fixed data pattern for multiple frames is generated as shown in figure 3. It is of length 16 bytes, attached to the video data in a frame of 2400bytes. Thus any number of frames can be occurred depending on the length of the video data sent from the satellite.

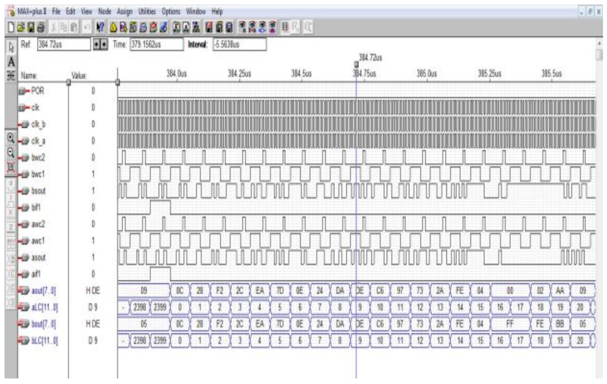

Fig 3: Pattern Generation Waveforms

The simulation results of the correlator is as shown below in figure 4. The loss pulse mentioned as lp in the output is enabled for the threshold values 0, 1 and 2. The raw detect is enabled for the threshold value set at 3 with up to 3 allowable errors.

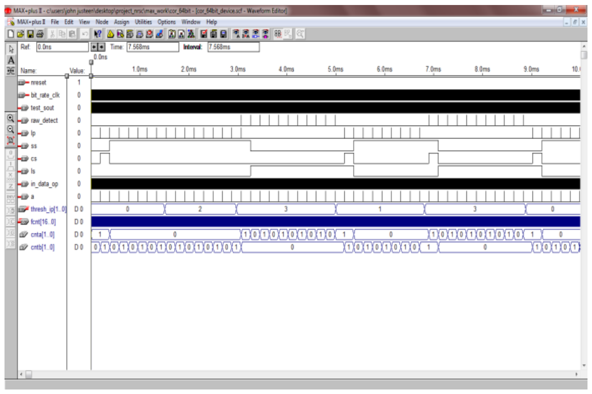

Fig 4: Correlator waveform

In the design a Bit Slip Window of 3 bits is designed which corrects the resultant one bit sync error. As the frame sync code is fixed and should reoccur at specific bit interval, a bit slip correction is developed to allow one it sync error. The fly wheel strategy output with a 3 bit slip window is shown in figure 5 below.

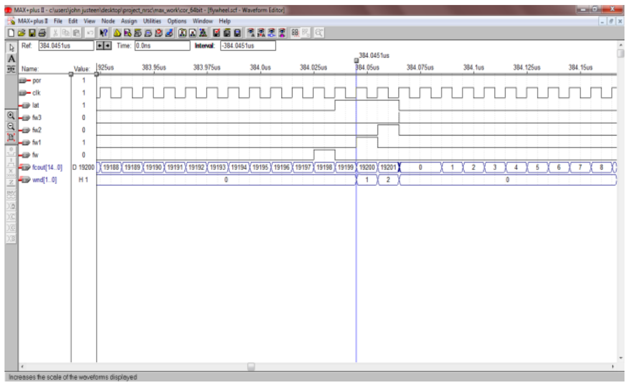

Fig 5: Flywheel Output

Hence, the sync pattern may occur one bit position early or late. 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.

The Floorplan Editor as shown in figure 6, provides a convenient method to enter and edit physical device resource assignments of the project. Two displays are available. They are the device view and LAB view where all pins on a device package with their functions and the interior of the device, including all Logic Array Blocks(LABs) with individual logic cells within each LAB are shown respectively. In devices that include Embedded Array Blocks (EABs), we can view individual embedded cells within each EAB. In MAX 9000, FLEX 6000, FLEX 8000, and FLEX 10K devices, I/O cell locations are also displayed. In addition, pins are displayed around the edges of the device packages. The Floorplan Editor provides a list of unassigned node and pin names of the project. Each name has a handle that you can drag to an individual pin, logic cell, I/O cell, or embedded cell in the Device View or LAB View display. You can also drag a node or pin with an existing assignment back to the list of unassigned nodes or to a different location on the device

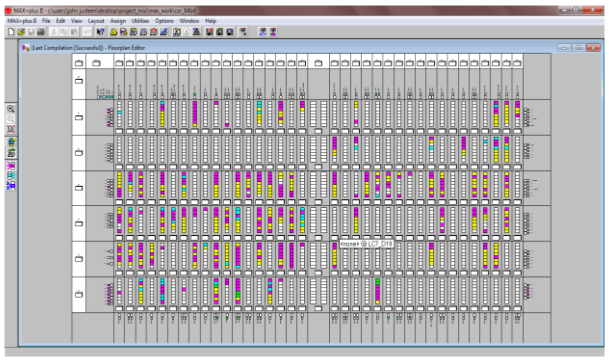

Figure 6: Floor Plan view

A color legend clearly indicates unassigned and assigned pins, logic cells, and I/O cells; the type of fan-out from each item; and VCC, GND, and reserved pins. The white, blue red and yellow indicate unassigned, device-wide fan-out, local LAB fan-out and unrouted. One can automatically display the fan-in and fan-out of any selected item(s), or the paths between multiple selected items and also view detailed routing statistics for selected item(s) and for the most congested area of a chip.

VI. CONCLUSION

During high speed data acquisition, validity of the data recorded is to be maintained. Hence real time auto correlation enables us as the first phase of identifying the data and subsequently recording the data. By implementing this in the hardware, considerable amount of load in the software is reduced, thus giving sufficient time for packing the data in the format required for further processing and display.

VII. REFERENCE

[1] Hand Book of Telemetry & remote control by Royal Signals.
[2] Advanced FPGA Design: Architecture, Implementation, and Optimization By Steve Kilts.
[3] International Journal of Electronics and Communication Engineering & Technology (IJECET), ISSN 0976–6464(Print), ISSN 0976=6472(Online) Volume 4, Issue 1, January-February, (2013), © IAEME.
[4] Altera Max plus II User Guide.
[5] www.altera.com/literature/ds/archives/dsf**10k**.pdf.
[6] Flow Measurement with Digital Correlator Realized by FPGA,Advanced Materials Research Vol.462(2012) pp 641-646.
[7] IACSIT International Journal of Engineering and Technology, Vol.3, No.3, June 2011.
[8] John G. Proakis and Dimities G. Manolakis, Digital Signal Processing Principles, Algorithms and Applications", Prentice-Hall India Publications, Third Edition, ISBN-81-203-1129-9.
[9] Integrated Electronics: Analog and Digital Circuits and Systems by Jacob Millman and Christos C.Halkias, Tata McGraw-Hill Edition.
[10] Digital Logic and Computer Design by M.Morris Mano, Prentice-Hall of India.
[11] http://www.ltrr.arizona.edu/~dmeko/notes_3/pdf
[11] www.stanford.edu/group/cioffi/book/chap/pdf.
[12]www.princeton.edu/~achaney/tmve/wiki100k/docs/Autocorrelation.html.
[13]www.physics.ohio-state.edu/~hughes/cdf_osu/xft/documents/dsconf/pdf.
[14] Communication Systems Security, AppendixA, Draft, L.Chen and G.Gong, 2008.