International Journal of Engineering Research and Science & Technology

ISSN : 2319-5991
Vol. 3, No. 4
November 2014

www.ijerst.com

Email: editorijerst@gmail.com or editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 3, No. 4, November 2014

Research Paper

A SPEED OPTIMIZED DECODER ARCHITECTURE FOR LOSSLESS TRANSMISSION WIRELESS SENSOR NETWORKS

Yezarla Rajeev[1]* and G S J Mani Kumar[2]

Corresponding Author: **Yezarla Rajeev ✉ rajeevyezarla90@gmail.com*

With the increase in silicon densities, it is becoming feasible for compression systems to be implemented in a single chip. A 32-bit system with memory architecture is based on having data compression and decompression engines working on data at the same time. The objective of the project is to design a lossless data compression system which operates in high-speed to achieve high compression rate. By using architecture of compressor, the data compression rates are significantly improved. Also inherent scalability of architecture is possible especially for wireless sensor networks. The main parts of the system are the data compressor and the control blocks providing control signals for the Data compressor, allowing appropriate control of the routing of data into and from the system. The Data compressor can process four bytes of data into and from a block of data every clock cycle. This is to ensure that adequate data is present for compressor to process rather than being in an idle state. The decoder will decodes the data without any loss of encoded redundancy data with high relaiablity and high speed.

Keywords: WSN, Lossless, Compressor / Decompressor

INTRODUCTION

A wireless sensor network (WSN) of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

The WSN is built of “nodes” – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several)

[1] M.Tech. Student, Department of ECE, Chirala Engineering College, Chirala 523155, Prakasam Dt., AP.
[2] Associate Professor, Department of ECE, Chirala Engineering College, Chirala 523155, Prakasam Dt., AP.

sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. The cost of sensor nodes is similarly variable, ranging from a few to hundreds of dollars, depending on the complexity of the individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communications bandwidth. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network. The propagation technique between the hops of the network can be routing or flooding.

Area monitoring is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. A military example is the use of sensors detect enemy intrusion; a civilian example is the geo-fencing of gas or oil pipelines. Area monitoring is most important part.The medical applications can be of two types: wearable and implanted. Wearable devices are used on the body surface of a human or just at close proximity of the user. The implantable medical devices are those that are inserted inside human body. There are many other applications too e.g. body position measurement and location of the person, overall monitoring of ill patients in hospitals and at homes. Body-area networks can collect information about an individual's health, fitness, and energy expenditure.

There are many applications in monitoring environmental parameters, examples of which are given below. They share the extra challenges of harsh environments and reduced power supply.Wireless sensor networks have been deployed in several cities (Stockholm[citation needed], London[citation needed] and Brisbane[citation needed]) to monitor the concentration of dangerous gases for citizens. These can take advantage of the ad hoc wireless links rather than wired installations, which also make them more mobile for testing readings in different areas. A network of Sensor Nodes can be installed in a forest to detect when a fire has started. The nodes can be equipped with sensors to measure temperature, humidity and gases which are produced by fire in the trees or vegetation. The early detection is crucial for a successful action of the firefighters; thanks to Wireless Sensor Networks, the fire brigade will be able to know when a fire is started and how it is spreading.

A landslide detection system makes use of a wireless sensor network to detect the slight movements of soil and changes in various parameters that may occur before or during a landslide. Through the data gathered it may be possible to know the occurrence of landslides long before it actually happens.Water quality monitoring involves analyzing water properties in dams, rivers, lakes & oceans, as well as underground water reserves. The use of many wireless distributed sensors enables the creation of a more accurate map of the water status, and allows the permanent deployment of monitoring stations in locations of difficult access, without

the need of manual data retrieval.Wireless sensor networks can effectively act to prevent the consequences of natural disasters, like floods. Wireless nodes have successfully been deployed in rivers where changes of the water levels have to be monitored in real time.

ERROR DETECTION SCHEMES

Error detection is most commonly realized using a suitable hash function (or checksum algorithm). A hash function adds a fixed-length tag to a message, which enables receivers to verify the delivered message by recomputing the tag and comparing it with the one provided.There exists a vast variety of different hash function designs. However, some are of particularly widespread use because of either their simplicity or their suitability for detecting certain kinds of errors (e.g., the cyclic redundancy check's performance in detecting burst errors).Random-error-correcting codes based on minimum distance coding can provide a suitable alternative to hash functions when a strict guarantee on the minimum number of errors to be detected is desired. Repetition codes, described below, are special cases of error-correcting codes: although rather inefficient, they find applications for both error correction and detection due to their simplicity.

A repetition code is a coding scheme that repeats the bits across a channel to achieve error-free communication. Given a stream of data to be transmitted, the data is divided into blocks of bits. Each block is transmitted some predetermined number of times. For example, to send the bit pattern "1011", the four-bit block can be repeated three times, thus producing "1011 1011 1011". However, if this twelve-bit pattern was received as "1010 1011 1011" – where the first block is unlike the other two – it can be determined that an error has occurred.Repetition codes are very inefficient, and can be susceptible to problems if the error occurs in exactly the same place for each group (e.g., "1010 1010 1010" in the previous example would be detected as correct). The advantage of repetition codes is that they are extremely simple, and are in fact used in some transmissions of numbers stations.[4][5].A parity bit is a bit that is added to a group of source bits to ensure that the number of set bits (i.e., bits with value 1) in the outcome is even or odd. It is a very simple scheme that can be used to detect single or any other odd number (i.e., three, five, etc.) of errors in the output. An even number of flipped bits will make the parity bit appear correct even though the data is erroneous.

Extensions and variations on the parity bit mechanism are horizontal redundancy checks, vertical redundancy checks, and "double," "dual," or "diagonal" parity (used in RAID-DP).A checksum of a message is a modular arithmetic sum of message code words of a fixed word length (e.g., byte values). The sum may be negated by means of a ones'-complement operation prior to transmission to detect errors resulting in all-zero messages.Checksum schemes include parity bits, check digits, and longitudinal redundancy checks. Some checksum schemes, such as the Damm algorithm, the Luhn algorithm, and the Verhoeff algorithm, are specifically designed to detect errors commonly introduced by humans in writing down or remembering identification numbers.

A cyclic redundancy check (CRC) is a single-burst-error-detecting cyclic code and non-secure hash function designed to detect accidental changes to digital data in computer networks. It is not suitable for detecting maliciously introduced

errors. It is characterized by specification of a so-called generator polynomial, which is used as the divisor in a polynomial long division over a finite field, taking the input data as the dividend, and where the remainder becomes the result.Cyclic codes have favorable properties in that they are well suited for detecting burst errors. CRCs are particularly easy to implement in hardware, and are therefore commonly used in digital networks and storage devices such as hard disk drives.Even parity is a special case of a cyclic redundancy check, where the single-bit CRC is generated by the divisor x + 1.

The output of a cryptographic hash function, also known as a message digest, can provide strong assurances about data integrity, whether changes of the data are accidental (e.g., due to transmission errors) or maliciously introduced. Any modification to the data will likely be detected through a mismatching hash value. Furthermore, given some hash value, it is infeasible to find some input data (other than the one given) that will yield the same hash value. If an attacker can change not only the message but also the hash value, then a keyed hash or message authentication code (MAC) can be used for additional security. Without knowing the key, it is infeasible for the attacker to calculate the correct keyed hash value for a modified message.

DESIGN METHODOLOGY

The XMatchPro algorithm is efficient at compressing the small blocks of data necessary with cache and page based memory hierarchies found in computer systems.It is suitable for high performance hardware implementation. The XMatchPro hardware achieves a throughput 2-3 times greater than other high-performance hardware implementation. The core component of the system is the XMatchPro based Compression / Decompression system. The XMatchPro is a high-speed lossless dictionary based data compressor. The XMatchPro algorithm works by taking an incoming four-byte tuple of data and attempting to match fully or partially match the tuple with the past data.

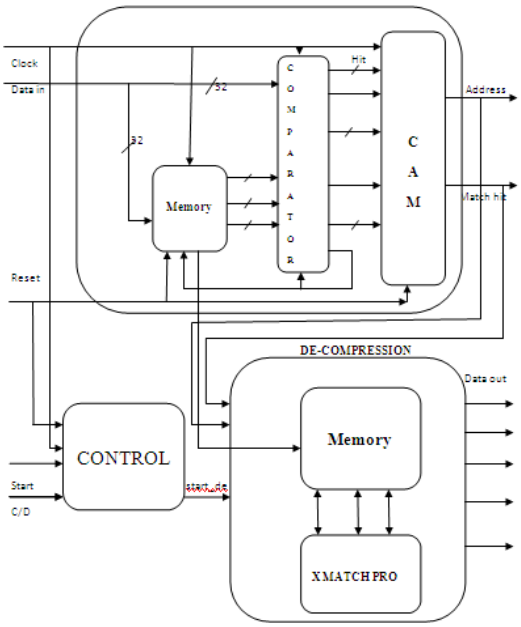
The XMatchPro algorithm maintains a dictionary of data previously seen and attempts to match the current data element with an entry in the dictionary, replacing it with a shorter code referencing the match location. Data elements that do not produce a match are transmitted in full (literally) prefixed by a single bit. Each data element is exactly 4 bytes in width and is referred to as tuple. This feature gives a guaranteed input data rate during compression and thus also guaranteed data rates during decompression, irrespective of the data mix. Also the 4-byte tuple size gives an inherently higher throughput than other algorithms, which tend to operate on a byte stream. The dictionary is maintained using move to front strategy, where by the current tuple is placed at the front of the dictionary and the other tuples move down by one location as necessary to make space. The move to front strategy aims to exploit locality in the input data. If the dictionary becomes full, the tuple occupying the last location is simply discarded.

A full match occurs when all characters in the incoming tuple fully match a dictionary entry. A partial match occurs when at least any tow of the characters in the incoming tuple match exactly with a dictionary entry, with the characters that do not match being transmitted literally. The use of partial matching improves the compression ratio when compared with allowing only 4 byte matches, but still maintains high throughput. If neither a full nor partial match

occurs, then a miss is registered and a single miss bit of '1' is transmitted followed by the tuple itself in literal form. The only exception to this is the first tuple in any compression operation, which will always generate a miss as the dictionary begins in an empty state. In this case no miss bit is required to prefix the tuple.

At the beginning of each compression operation, the dictionary size is reset to zero. The dictionary then grows by one location for each incoming tuple being placed at the front of the dictionary and all other entries in the dictionary moving down by one location. A full match does not grow the dictionary, but the move-to-front rule is still applied. This growth of the dictionary means that code words are short during the early stages of compressing a block. Because the XMatchPro algorithm allows partial matches, a decision must be made about which of the locations provides the best overall match, with the selection criteria being the shortest possible number of output bits.

Figure 1: Proposed Block Diagram



Array is of length of 64X32 bit locations. This is used to store the unmatched incoming data and when a new data comes, the incoming data is compared with all the data stored in this array. If a match occurs, the corresponding match location is sent as output else the incoming data is stored in next free location of the array & is sent as output. The last component is the cam comparator and is used to send the match location of the CAM dictionary as output if a match has occurred. This is done by getting match information as input from the comparator. Suppose the output of the comparator goes high for any input, the match is found and the corresponding address is retrieved and sent as output along with one bit to indicate that match is found. At the same time, suppose no match occurs, or no matched data is found, the incoming data is stored in the array and it is sent as the output. These are the functions of the three components of the Compressor. The hardware descriptions of these modules are done using VHDL Language. VHDL is an acronym for Very high-speed integrated circuits Hardware Description Language. It can be used to model a digital system at many levels of the abstraction, ranging from the algorithmic level to gate level.

The block diagram gives the details about the components of a single 32- bit compressor / decompressor. The Same design approach is used for designing a 64-bit Compression/ Decompression system which is essentially used for comparison of increased compression rates given by the 64-bit Lossless Parallel High-Speed Data Compression System. There are three components namely compressor, decompressor, control. The compressor has the following components which are comparator, array and cam comparator.The comparator is used to

compare two 32-bit data and to set or reset the output bit as 1 for equal and 0 for unequal. Array is of length of 64X32bit locations. This is used to store the unmatched in coming data and when the next new data comes, that data is compared with all the data stored in this array. If the incoming data matches with any of the data stored in array, the Comparator generates a match signal and sends it to Cam Comparator. The last component is the Cam comparator and is used to send the incoming data and all the stored data in array one by one to the comparator. Suppose output of comparator goes high for any input, then the match is found and the corresponding address (match based compressor).

At the same time, suppose no match is found, then the incoming data stored in the array is sent as output. These are the functions of Array, so it stores the data in the Array and if the match hit data is 1, it indicates the data is present in the Array, then it instructs to find the data from the Array with the help of the address input and sends as output to the data out location) is retrieved and sent as output along with the three components of the XMatchPro.The decompressor has the following components – Array and Processing Unit. Array has the same function as that of the array unit used in the Compressor. It is also of the same length. Processing unit checks the incoming match hit data and if it is 0, it indicates that the data is not present in the one bit to indicate the match is found.

RESULTS

Figure 2: Comparator Output Waveform

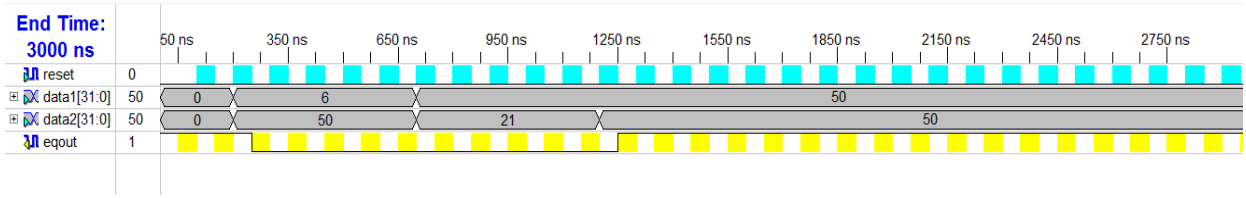


Figure 3: CAM Comparator

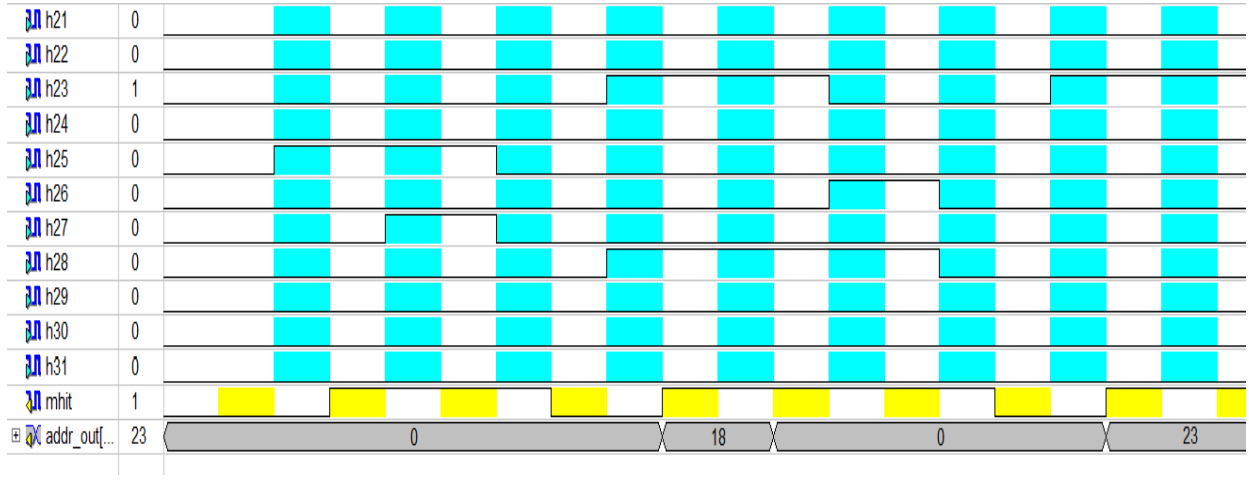


Figure 4: Content Addressable Memory Output Waveform

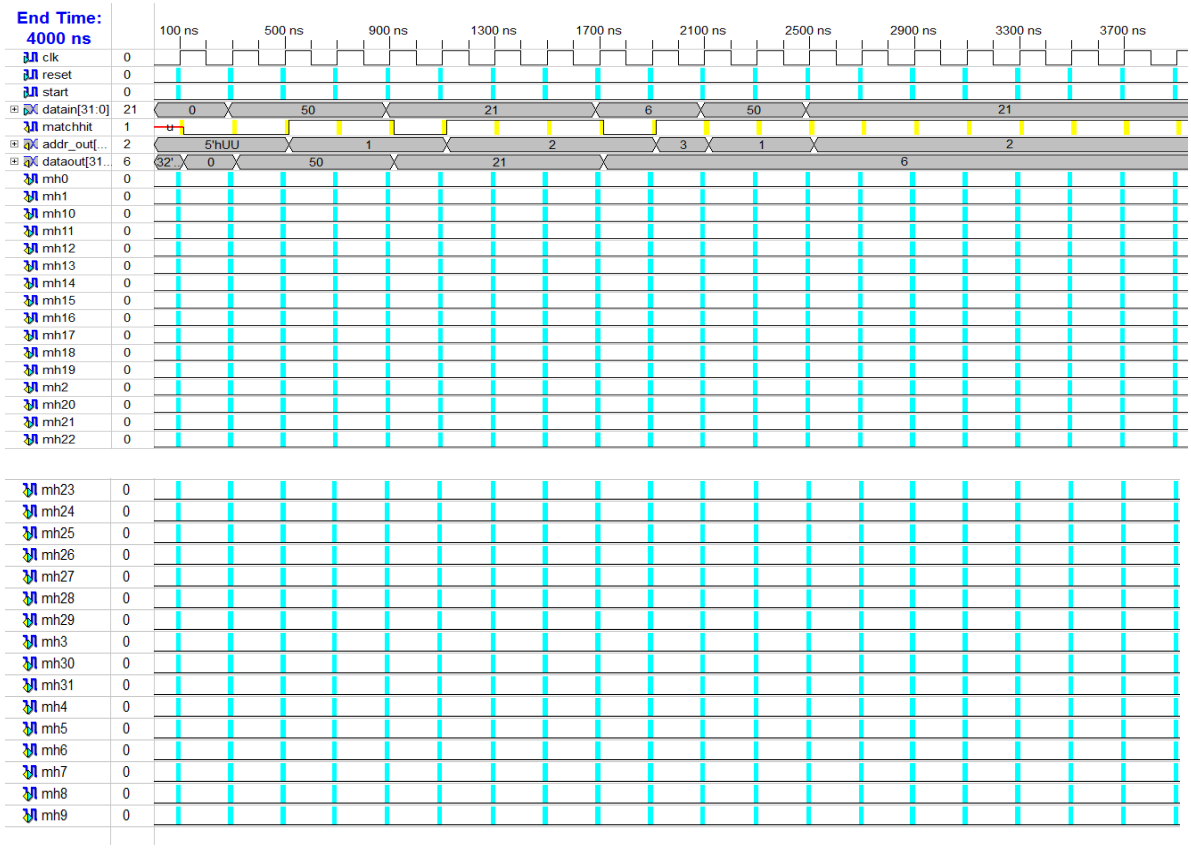


Figure 5 RTL View of Data Compression

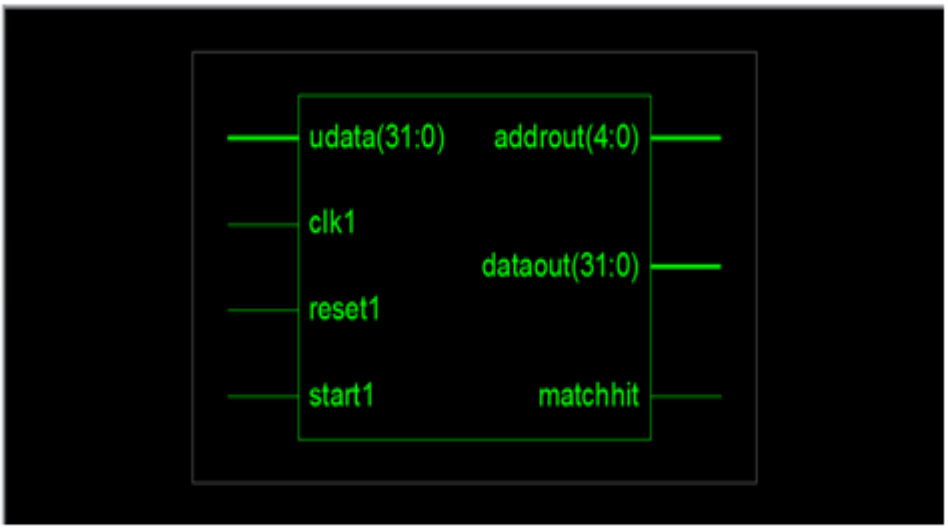


Figure 6: 32-bit Compression Output Waveform

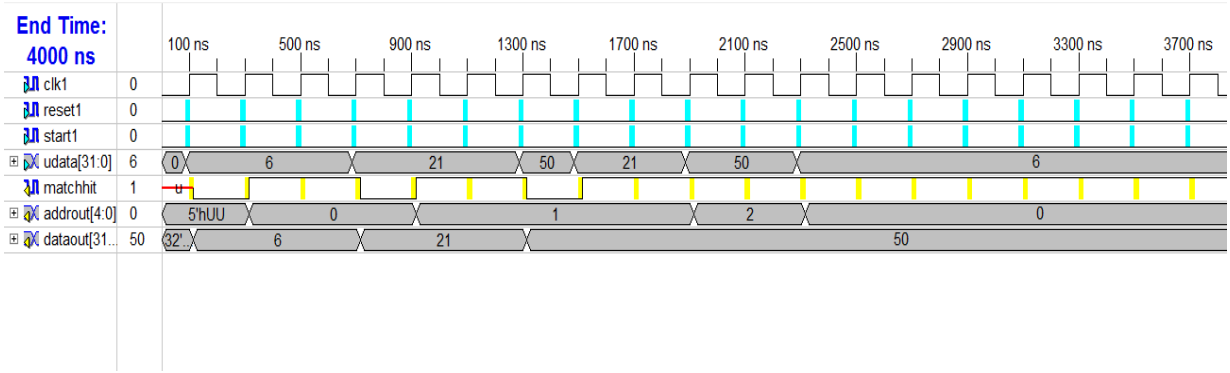


Figure 7: RTL View of Data De-compression

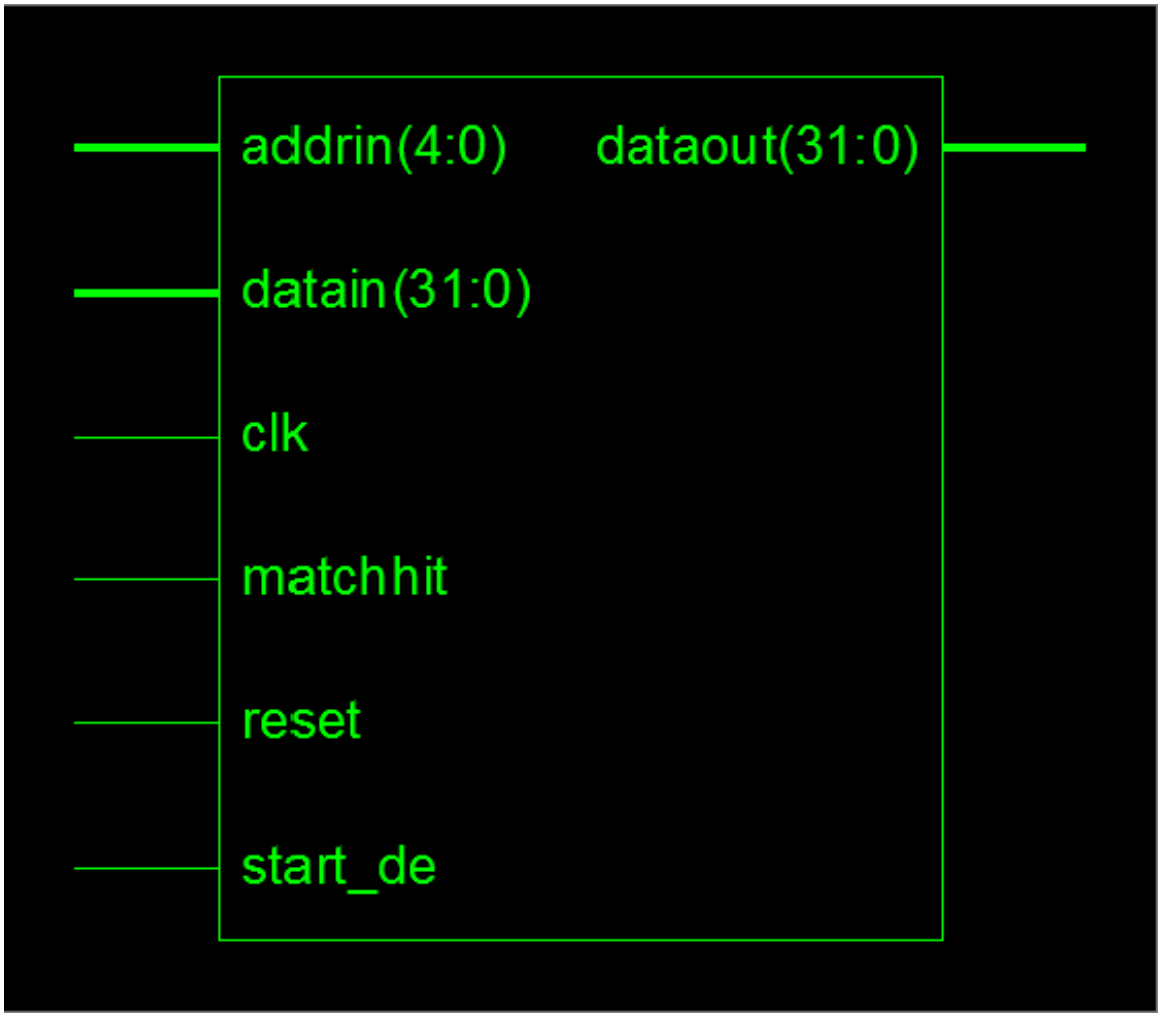
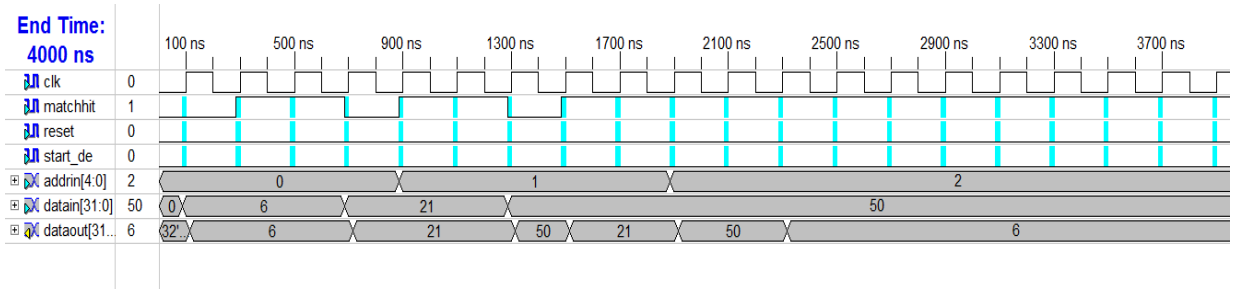


Figure 8: 32 bit Data Decompression Output Waveform



CONCLUSION

In this project "IMPLEMENTATION OF LOSS LESS DATA COMPRESSION" we presented a simple method to implement lossless data compression system which operates at high-speed to achieve high compression rate. By using Parallel architecture of compressors, the data compression rates are significantly improved and also inherent scalability of parallel architecture is possible. The algorithm "XMATCHPRO" used in this project is efficient at compressing and the flexibility provided by using this technology is of great interest, since the chip can be adapted to the requirements of a particular application easily. The functionality is verified using ISE simulator and the synthesis is carried out using XILINX ISE 12.3i with VERILOG HDL.

REFERENCES

1. Ang W-P and Garg H K (2001), "A New Iterative Channel Estimator for Thelog-MAP and Max-log-MAP Turbo Decoder in Rayleigh Fading Channel," in *Proc. Global Telecommun. Conf*., Vol. 6, pp. 3252–3256.
2. B. Koenemann (1991), "LFSR-coded Test Patterns for Scan Designs", *in Proc. Eur. Test Conf*., pp. 237–242.
3. Forney G D (1973), "The Viterbi Algorithm", *Proc. IEEE*, Vol. 61, No. 3, pp. 268–278.
4. May M, Ilnseher T, Wehn N and Raab W (2010), "A 150 Mbit/s 3GPP LTEturbo Code Decoder", *in Proc. Design, Autom. Test in Euro. Conf. Exhib.(DATE*), pp. 1420–1425.
5. Rajski J, Tyszer J, Kassab M and Mukherjee N (2004), "Embedded Deterministic Test", *IEEE Trans. Comput.-Aided Des. Integr. Circuit Syst*., Vol. 23, No. 5, pp. 776–792.
6. Robertson P, Hoeher P and Villebrun E (1997), "Optimal and Sub-optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding", *Euro.Trans. Telecommun*., Vol. 8, No. 2, pp. 119–125.
7. Studer C, Benkeser C, Belfanti S and Huang Q (2011), "Design and Implementation of a Parallel Turbo-decoder ASIC for 3GPP-LTE", *IEEE J.Solid-State Circuits*, Vol. 46, pp. 8–17.
8. Touba N A (2006), "Survey of Test Vector Compression Techniques", *IEEE Des.Test Comput*., Vol. 23, No. 4, pp. 294–303.
9. Wong C, Lee Y and Chang H (2009), "A 188-size 2.1 mm Reconfigurable Turbo Decoder Chip with Parallel Architecture for 3GPP LTE System", *in Proc. Symp. VLSI Circuits*, pp. 288–289.

International Journal of Engineering Research and Science & Technology
Hyderabad, INDIA. Ph: +91-09441351700, 09059645577
E-mail: editorijlerst@gmail.com or editor@ijerst.com
Website: www.ijerst.com