Title : X-MatchPROvw Compression/Decompression FPGA Processor.

Preliminary Information (May 2002).

Author: José Luis Núñez

Features

- High-speed lossless data compressor supports compression and decompression in a single FPGA.
- Altera APEX20KE prototype implementation available on PCI board.
- Throughput up to 200 Mbytes/second compression/decompression with low latency clocking at 50 MHz on a FPGA.
- Full-duplex operation enables simultaneous compression/decompression for a combined performance of 400 Mbytes/s.
- Full-duplex architecture enables self-checking test mode using CRC (Cyclic Redundancy Check) codes.
- 32-bit high-performance coprocessor-style interface.
- Fully contained 32-bit architecture does not require any external components and supports operation on blocked data.
- Easy migration to gate array technology enables 3 times increase in performance.
- Compression ratio comparable to HiFn LZS and IBM ALDC using comparable dictionary sizes.

Applications

- Computer systems.
- Networking products.
- High performance storage devices.
- Data logging equipment.
- Remote sensing applications.

1.Benefits of data compression

The use of lossless data compression can bring about a number of increasingly important benefits to an electronic system. The term ‘lossless’ means that the original data can be exactly recreated after a

decompression operation, and should not be confused with audio and video compression systems (such as JPEG and MPEG) which are lossy and hence only recreate an approximation of the original data.
The most obvious benefit of data compression is a reduction in the volume of data which must be stored. This is important where the storage media itself is costly (such as memory) or other parameters, such as power consumption, weight or physical volume, are critical to product feasibility. Using data compression reduces the total storage requirement, thus effecting a cost saving.
There are also two other positive effects that data compression brings. The first of these is a reduction in the bandwidth required to transmit a given amount of data –less data must be transmitted when in compressed form, and hence less bandwidth is required. This can effect a cost saving in cabling operations, where a lower bandwidth link will be sufficient to meet demand. The second effect is that given a fixed bandwidth, the total time required to transmit compressed data is less than for uncompressed data. This can lead to a performance benefit, as the bandwidth of a link appears greater when transmitting compressed data and hence more data can be transmitted in a given amount of time.

2.X-MatchPROvw design architecture

The X-MatchPROvw compressor/decompressor processor is a fully contained unit having a simple architecture and uncomplicated interface – Figure 1 shows the global architecture together with the PCI interface.

The X-MatchPROvw design is a dictionary style compressor based around a dictionary implemented in the form of a content addressable memory (CAM). The length of the CAM varies with values ranging from 16 to 1024 tuples (4-byte locations) trading complexity for compression. Typically, the device complexity increases by a factor of 1.5 each time the dictionary doubles. Dictionary size is variable to be able to adapt algorithm complexity to the resources available in the selected FPGA. Each dictionary entry contains exactly 4 bytes. The dictionary adaptively stores the most recent phrases that have occurred in the data stream. Compression is achieved by replacing repeated phrases with references to the dictionary (these are codewords witch are sorter than the phrase itself).

The coding section is active during compression. This generates the required codewords and forms successive codewords into fixed 32-bit width words for writing to external medium. The decoding section is responsible for the reverse process – data is read from the external medium and generates the required dictionary references to allow the decompressed data to be recreated.

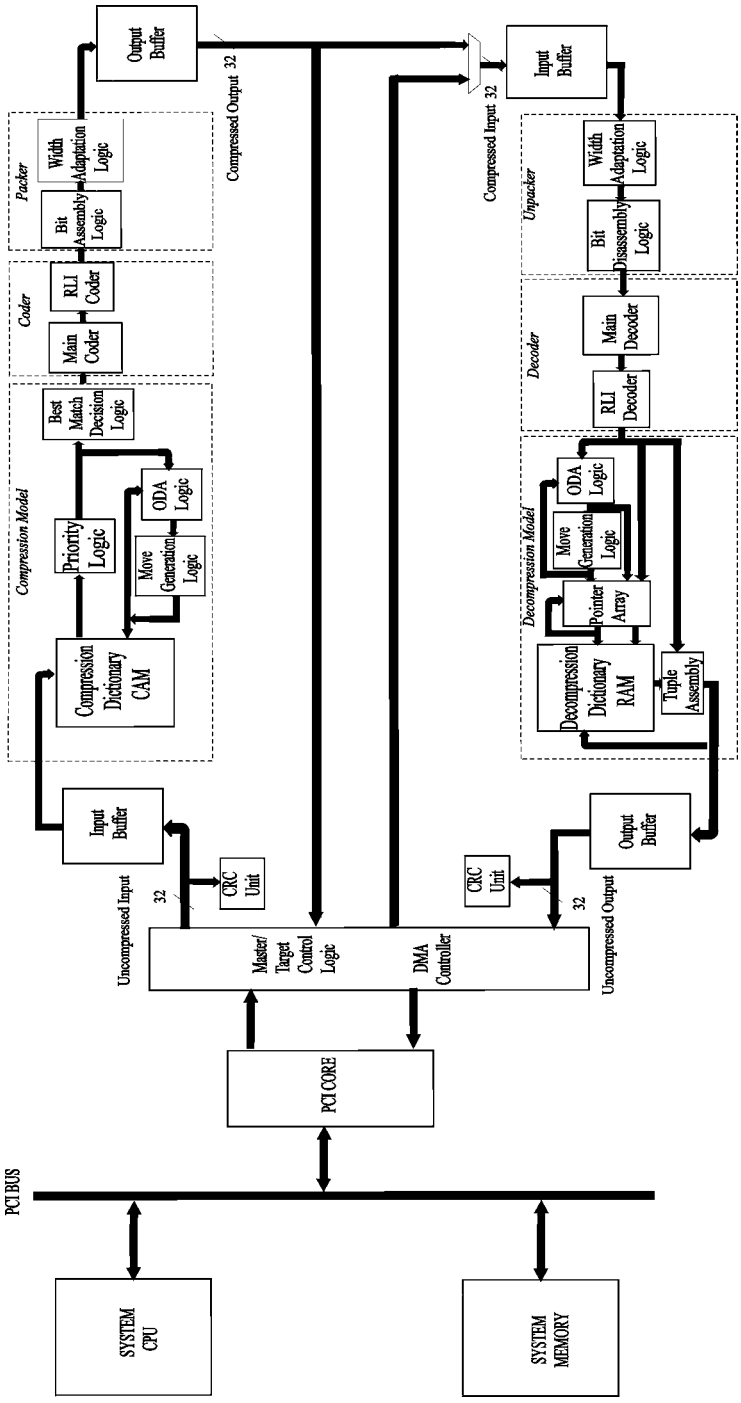

Figure 1. X-MatchPROvw plus PCI interface architecture.

3.Interface description

Figure 2 illustrates the X-MatchPROvw interface.

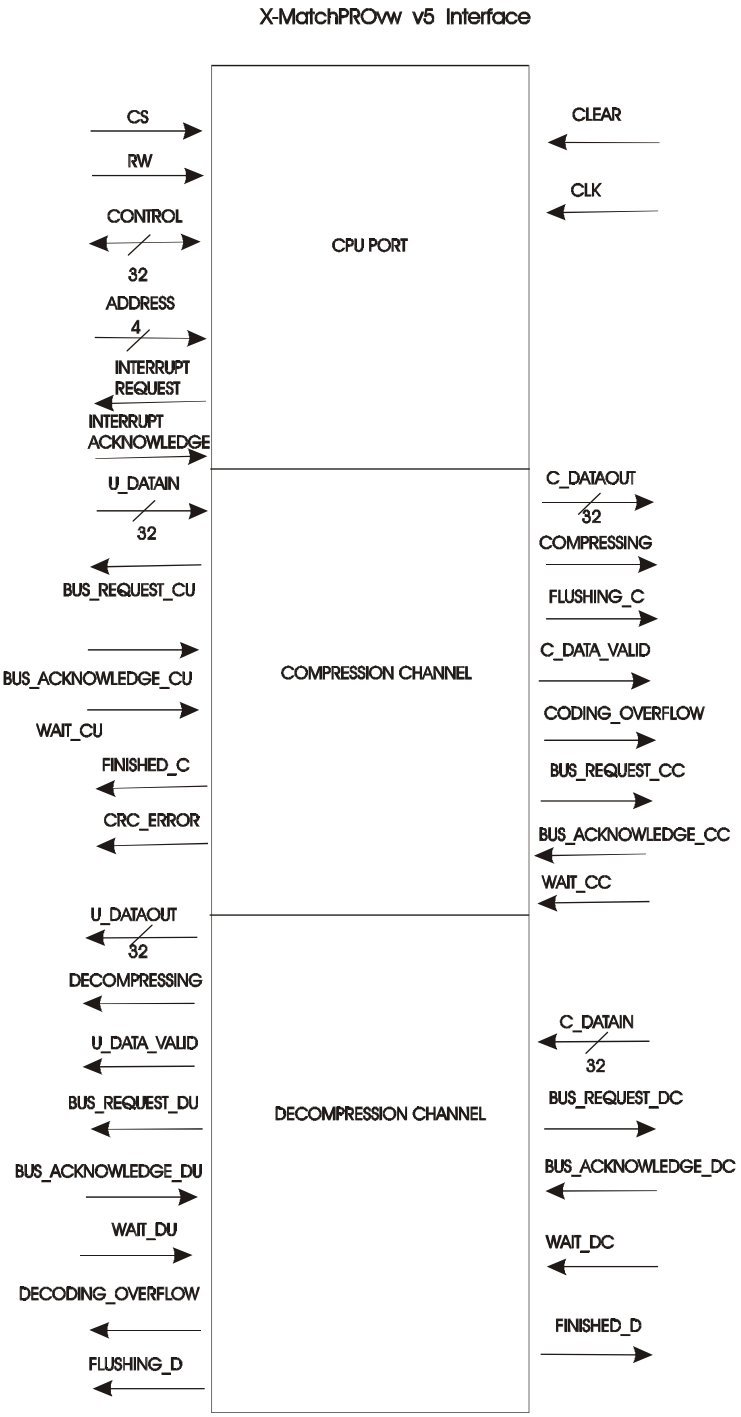

Figure 2. X-MatchPROvw interface.

Table 1 describes the functionality of these signals. All the signals are active low and fully synchronous.

Signal name	Direction	Width	Function
CS	IN	1	Enable access to the internal registers.
RW	IN	1	Enable reading or writing the internal registers.
ADDRESS	IN	4	Internal register address.
CLK	IN	1	System clock. Positive edge triggered.
CLEAR	IN	1	Asynchronous clear of all the storage elements.
BUS_ACKNOWLEDGE_CC	IN	1	The system grants the compressed data out bus during compression.
BUS_ACKNOWLEDGE_CU	IN	1	The system grants the uncompressed data in bus during compression.
BUS_ACKNOWLEDGE_DC	IN	1	The system grants the compressed data in bus during decompression.
BUS_ACKNOWLEDGE_DU	IN	1	The system grants the uncompressed data out bus during decompression.
BUS_REQUEST_CC	OUT	1	The chip requests the compressed data out bus during compression. Compressed data ready to be output.
BUS_REQUEST_CU	OUT	1	The chip request the uncompressed data in bus during compression.
BUS_REQUEST_DC	OUT	1	The chip requests the compressed data in bus during decompression. The chip request compressed data to be decompressed.
BUS_REQUEST_DU	OUT	1	The chip requests the uncompressed data out bus during decompression. The chip request compressed data to be decompressed.
FINISH_C	OUT	1	The chip signals end of a compression operation.
WAIT_CC	IN	1	Hold the output of compressed data during compression
WAIT_CU	IN	1	Wait for uncompressed data during compression
WAIT_DC	IN	1	Wait for compressed data during decompression

WAIT_DU	IN	1	Hold the output of uncompressed data during decompression.
FINISH_D	OUT	1	The chip signals end of a decompression operation.
CONTROL	IN/ OUT	16	Common system bus to issue compression and decompression commands to the chip. The control bus is also used to write or read the compressed and uncompressed block size registers if required together with the CRC registers and status registers.
U_DATA_IN	IN	32	Uncompressed data input during compression.
C_DATA_OUT	OUT	32	Compressed data output during compression.
CODING_OVERFLOW	OUT	1	Data overflow in the coding buffers. Error condition
C_DATA_VALID	OUT	1	Valid compressed data present in compressed data out bus.
COMPRESSING	OUT	1	Compression engine active.
FLUSHING_C	OUT	1	Compression engine inactive emptying the coding buffers.
C_DATA_IN	IN	32	Compressed data input during decompression.
U_DATA_OUT	OUT	32	Uncompressed data output during decompression.
DECODING_OVERFLOW	OUT	1	Overflow in the output buffers. Error condition.
U_DATA_VALID	OUT	1	Uncompressed data valid in the uncompressed data out bus.
DECOMPRESSING	OUT	1	Decompression engine active.
FLUSHING_D	OUT	1	Decompression engine inactive emptying the output buffers.
CRC_ERROR	OUT	1	Compression and decompression CRC codes do not match. Hardware failure.
INTERRUPT_REQUEST	OUT	1	The device issues an interrupt request because it has encountered and error or it has terminated the requested compression or decompression operations.
INTERRUPT_ACKNOWLEDGE	OUT	1	The interrupt request is acknowledged.

Table 1. Chip pinout.

4. Register bank description.

A total of 10 registers form the register bank that controls the compression/decompression engines and coding/decoding buffers. These registers are accessed using the address bus and the control bus and can be read or written. Figure 3 shows the format of these registers.

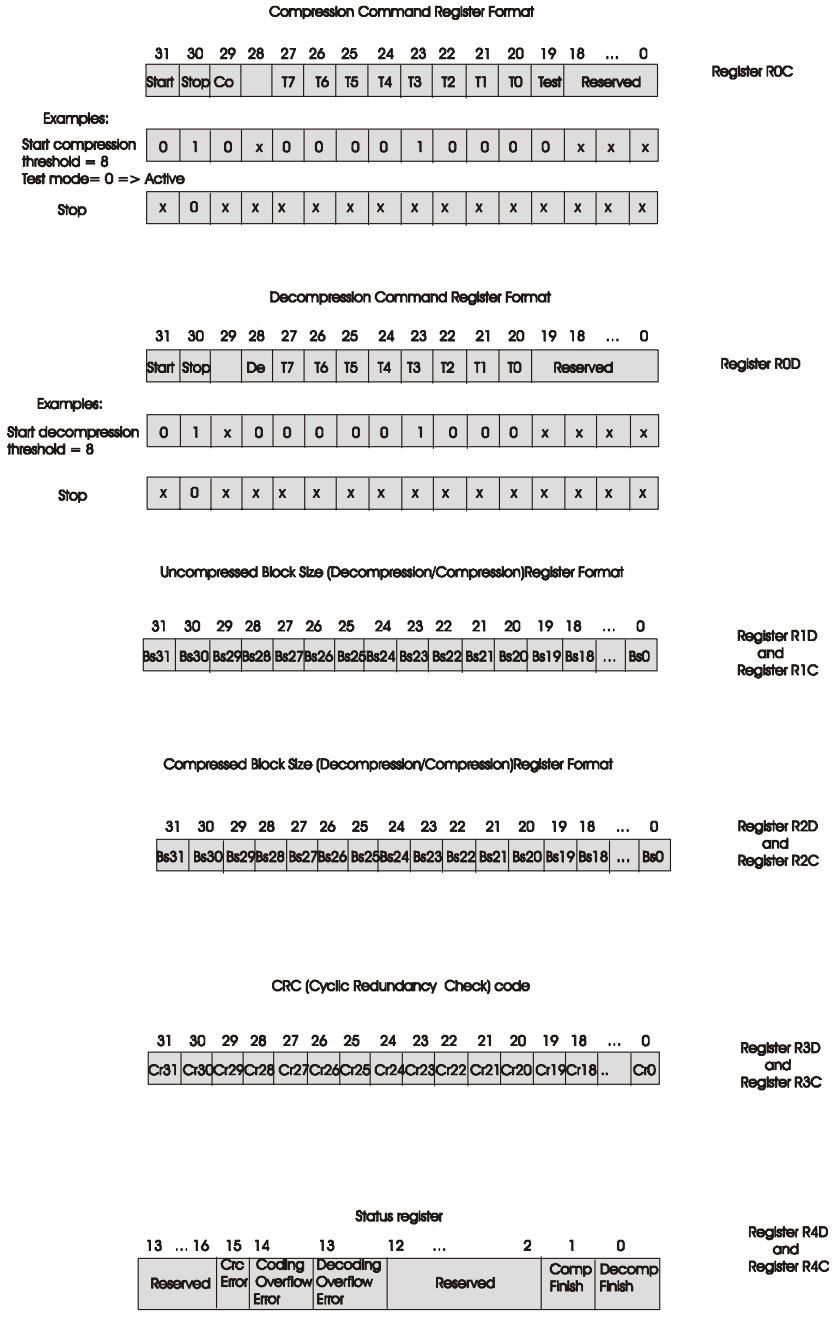

Figure 3. Register format.

Address	Channel	Register	Function
1000	Decompression	R0D Command register	Activates or stops the decompression channel
1001	Decompression	R1D Uncompressed block size register	Sets the number of bytes of the uncompressed block before decompression
1010	Decompression	R2D Compressed block size register	Reserved
1011	Decompression	R3D Decompression CRC	CRC code is stored here after completion of a decompression operation
0001	Decompression	R4D Decompression Status	Status information of the decompression channel
1100	Compression	R0C Command register	Activates or stops the compression channel
1101	Compression	R1C Uncompressed block size register	Sets the number of bytes of the uncompressed block before compression
1110	Compression	R2C Compressed block size register	Sets the number of bytes of the compressed block after compression
1111	Compression	R3C Compression CRC	CRC code is stored here after completion of a compression operation
0000	Compression	R4C Compression Status	Status information of the compression channel

Table 2. Register access description

5. X-MatchPROvw threshold value.

The threshold value is input with the command and written in the command register. It defines a programmable latency. A small value means a low latency but it is more probable that underflows in the output buffers will take place. A bigger value introduces more latency but these conditions are not so frequent. After an underflow in the output buffers the threshold value also defines the distance between write and read addresses before more compressed or uncompressed data is output or requested respectively. Underflow conditions are not error conditions but they will generate bubbles where valid data is not present in the compressed or uncompressed data out streams during compression or decompression respectively.
The threshold can have any value between 1 and 128. A threshold of 1 implies minimum latency => 1*64 bits of data are written in the buffer before the bus is requested during compression to output compressed data or 1*32 bits of data are written in the output buffers before the bus is requested during decompression. A threshold of 128 implies maximum latency or blocked operational mode => 128 * 64 bits of data are written in the buffer before the bus is requested during compression to output compressed data or 128*32 bits of data are written in the output buffer before the bus is requested during decompression.

6. X-MatchPROvw latency

In compression latency is defined as the number of cycles found between the moment the compression engine stops inputting data and the output buffers finish emptying the buffers (=> chip ready to start a new operation). The compression latency has two components one fix and one variable. The fixed component of 4 cycles is defined by the levels of registers located between the input search register and the output buffers

(5 levels) and the variable component is defined by how much data is in the buffers when the compression engine finishes its operation (flushing operation). The probability of having a long flushing operation is small when the threshold value setting is small. This variable component depends, however, in the input data. If the data expands the latency will grow because more data will be left in the buffers to be output during the flushing operation.

In decompression latency is also controlled by the threshold value. Latency can be defined as the number of cycles that elapse between the first tuple of compressed data enters the chip and the first tuple of uncompressed data leaves the chip. There are again two components. The levels of registers (5 levels) between the decoding buffers and the output register in the device introduced a fixed component of 4 cycles. The output buffer introduces the other component and it depends on the threshold value. A threshold value of 8 introduces a latency of 8 because 8 32-bit tuples must be written in the buffer before the number of 32-bit words exceeds the threshold value and the bus is requested to output uncompressed data.

7. X-MatchPROvw operational modes.

The device organizes the data block to be processed during compression and decompression operations in records of 512 bytes. This means that it will request the input data bus until one record has accessed the input buffers and then it will release the data bus and rearbitrate for new data if required until the whole block has accessed the input buffers. The compression and decompression engines are engaged shortly after the first input record has started accessing the bus and data will be available in the output buffers after a short latency. It is the responsibility of the system to service the requests originating in the output buffers to avoid having overflow errors in these buffers.

7.1 Compression mode

To start a compression operation the CPU must write two registers: The uncompressed block size register (UBSR) must be written first and the command register (CR) must be written second. The UBSR tells the compression engine when it must stop after processing all the bytes of data present in the block. The UBSR specifies the number of bytes present in the block and can be any value between 8 and 65536. The CR puts the device in compression mode and it also contains the threshold value to control the output buffer. It also sets the test bit that sets the device to self-checking test mode when 0 or to full-duplex mode when 1. The device requests the uncompressed data in bus after the command register has been set using the signal *bus request cu*. The system will grant the bus using *bus acknowledge cu* when data is ready for compression. Data must be available in the uncompressed data in bus one cycle after the bus has been granted. If data is not ready for the device the *wait cu* signal in the uncompressed data in bus can be asserted. The chip

requests the compressed bus when the number of 64-bit words available in the output buffer is bigger than the threshold value using the *bus request cc* signal and waits for the acknowledgement *bus acknowledge cc*. If data cannot be collected from the compressed data out bus the corresponding *wait cc* signal can be used to hold the outputting of data by the device. When the device produces compressed data in the compressed bus it asserts the *compressed data valid* signal active. The engine is known to be active because the *compressing* signal is active. The chip stops processing data when the value stored in UBSR is reached. Then a *flushing c* signal is activated to indicate that any remaining compressed data in the output buffers is being flushed out. When the buffers are emptied of their contents the device asserts the signal *finished c* active for one cycle and the *interrupt request* signal. The system can read the compressed block size register (CBSR) at the end of a compression operation to obtain the resulting compressed block size in bytes. This value could be compared with the original uncompressed block size to evaluate the compression efficiency. The system can also read the status register to monitor that an abnormal termination did not take place. After this cycle the device is ready to start a new compression operation. Figure 4 corresponds to a typical compression operation.

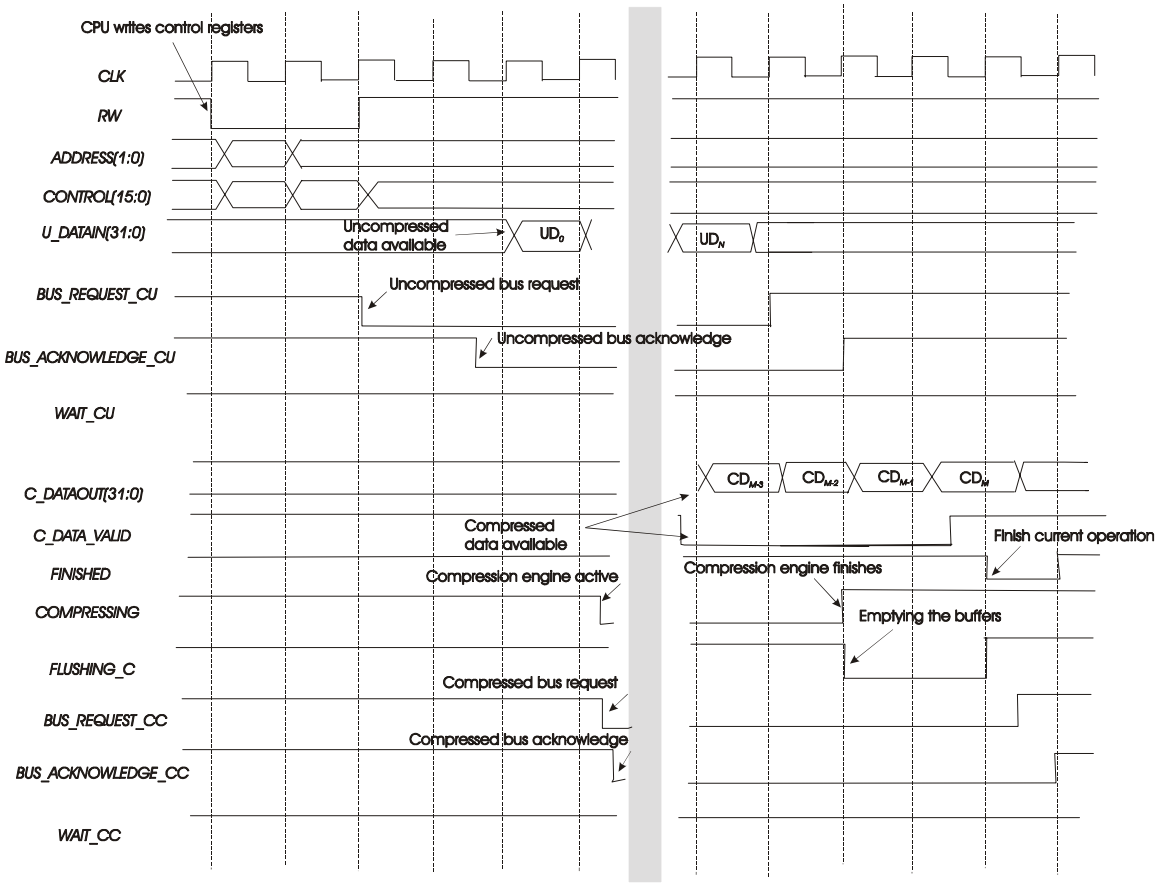

Figure 4. Compression operation

7.2 Decompression mode

To start a decompression operation the system must write 2 registers. The UBSR and the CR have the same function as in compression. The UBSR is used to indicate the device how much data must be decompressed before finishing the decompressed operation. Then, the system requests the compressed data in bus with the *bus request dc* signal and the bus is granted with the *bus acknowledge dc* signal. The *bus request dc* during decompression is equivalent to a compressed data request. Once the bus is granted the system is responsible to make available 32 bits of compressed data per cycle as long as the bus request signal is maintained active. The system can use the *wait dc* signal to insert wait cycles in the bus. The engine writes uncompressed data in the output buffers. Once the amount of data is larger than the threshold value the device asserts the bus *request du* signal requesting the uncompressed data out bus. The bus is granted with the *bus acknowledge du* signal. . When the device produces uncompressed data in the uncompressed data out bus it asserts the *uncompressed data valid* signal active. The engine is known to be active because the *decompressing* signal is active. When the output buffers are emptied of their contents the device asserts the signal *finished d* active for one cycle and the *interrupt request* signal. . The system can read the status register to monitor that an abnormal termination did not take place. After this cycle the device is ready to start a new decompression operation. Figure 5 shows a typical decompression cycle.

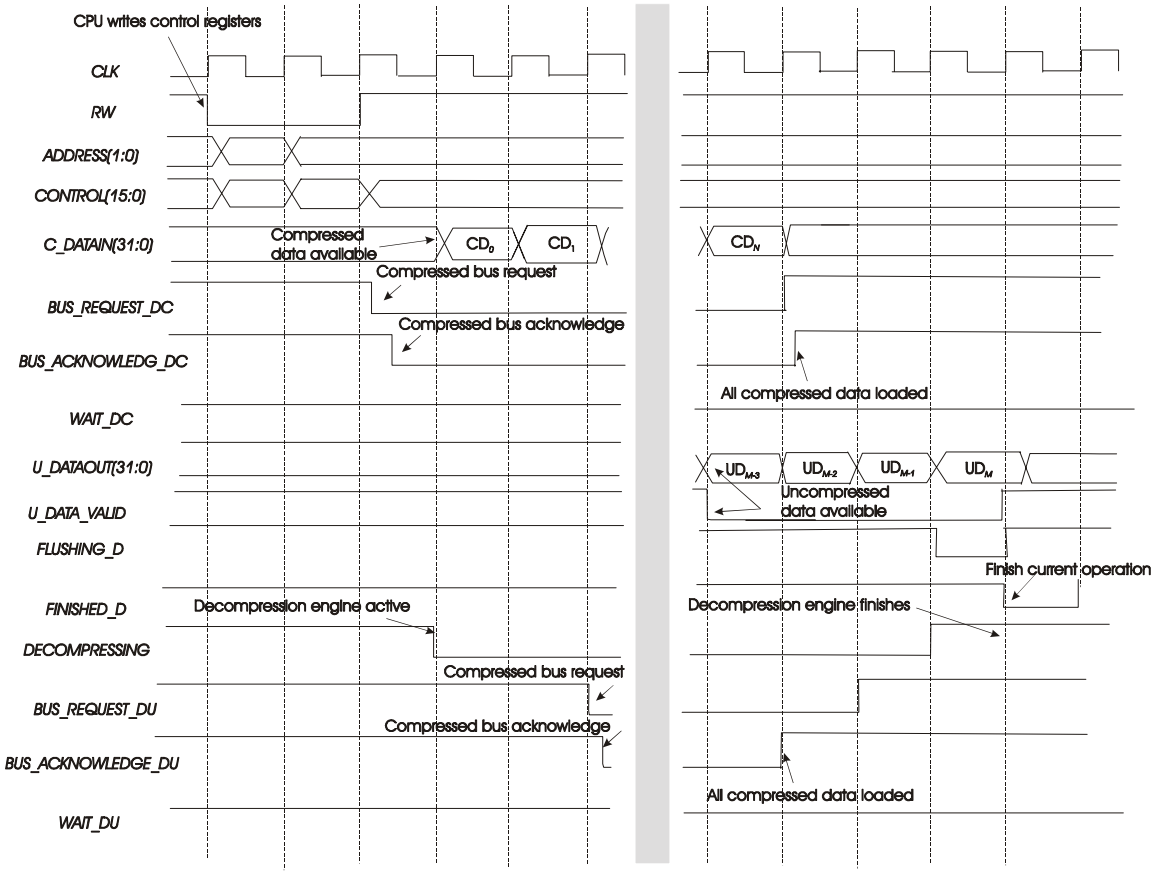

Figure 5. Decompression operation

7.3 X-MatchPRO Error conditions.

7.3.1 Output Buffer Coding Overflow and Output Buffer Decoding Overflow

Overflow errors should never be encountered under normal operation conditions. To avoid overflow errors the output bus that holds compressed data during compression and uncompressed data during decompression should be granted if it is being requested when the inputting of one data record has finished and before the inputting of a new data record starts.

7.3.2 CRC Error

A CRC error should never be encountered under normal operation conditions. The CRC error signal is used during compression in test mode. Both channels are active and a CRC code is calculated using all the data input to the compression channel and output by the decompression channel. A CRC error indicates a hardware failure because either the compression or the decompression channels failed to successfully perform its operation and there has been a mismatch in the calculated CRC's by each channel.

Electronic Systems Design Group,
Department of Electronic & Electrical Engineering,
Loughborough University, Loughborough, Leicestershire. LE11 3TU. UK.
Phone : +44 1509 227070 Fax: +44 1509 227014
E_mail: J.L.nunez-yanez@lboro.ac.uk, S.R.Jones@lboro.ac.uk