

Two-dimensional Discrete Wavelet Transform Memory Architectures

Ibrahim Saeed Koko, *Member, IAENG* and Herman Agustiawan

Abstract—Until now, the external memory architectures of the two-dimensional discrete wavelet transform (2-D DWT) such as the external RAM and the memory between DWT unit and compression unit, call it, subband memory, which are in need of architecting, have been overlooked in the literature. Since, 2-D DWT memory architectures are equally important as DWT processor architectures commonly covered in the literature, in this paper, two novel VLSI memory architectures for lifting-based 5/3 and 9/7 DWT are proposed. The first proposed memory architecture, the RAM, is read and written by DWT unit only, whereas, the second proposed memory architecture, the subband memory, is written by DWT unit and is read by compression unit.

Index Terms—DWT memory architecture, LL-RAM, subband memory, lifting scheme, and VLSI architecture

I. INTRODUCTION

The 2-D DWT considered in this paper is part of a compression system. The general structure of a compression system is shown in Fig. 1. The DWT unit generally consists of a row-processor (RP) and a column-processor (CP) [1, 2]. RP reads LL-RAM, while CP writes into LL-RAM and subband memory.

DWT decomposes an $N \times M$ image into subbands, as shown in Fig. 2 for 3 decomposition levels [7]. These subbands must be stored by DWT unit in a memory such that they can be manipulated effectively by compression unit for compression purposes. Therefore, a memory architecture, which allows DWT unit to perform efficiently both, reads and writes and compression unit to perform reads is necessary.

Fig. 2 shows that the first decomposition generates 4 subbands labeled HL₁, HH₁, LH₁, and LL₁. The coefficients of the first 3 subbands would be stored in a memory, call it subband memory, which would contain memory blocks labeled HL₁, HH₁, and LH₁. The compression unit can then read the 3 subbands and compress each independently. While the LL₁ subband would be stored in another memory, call it, LL-RAM or just RAM, for further decompositions.

The second decomposition generates 4 subbands, labeled HL₂, HH₂, LH₂, LL₂, by reading subband LL₁ coefficients stored in the LL-RAM. The coefficients of the 3 subbands HL₂, HH₂, and LH₂ would be stored also in the subband memory blocks labeled HL₂, HH₂, and LH₂, while subband LL₂ would be stored in the RAM for further decomposition.

In the discussion above, two memory components have

been identified, the LL-RAM and the subband memory that need to be designed such that DWT unit can perform effectively both read and write operations in the LL-RAM and write only into subband memory, while compression unit can read subband memory.

This paper is organized as follows. In sections II and III the proposed external RAM and subband memory architectures are presented. Conclusions are given in section IV.

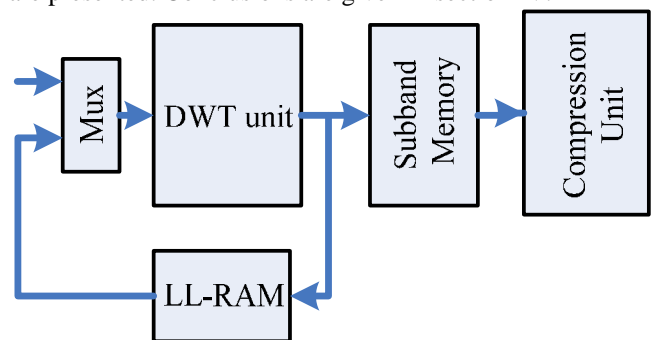


Fig. 1 General structure of a compression system

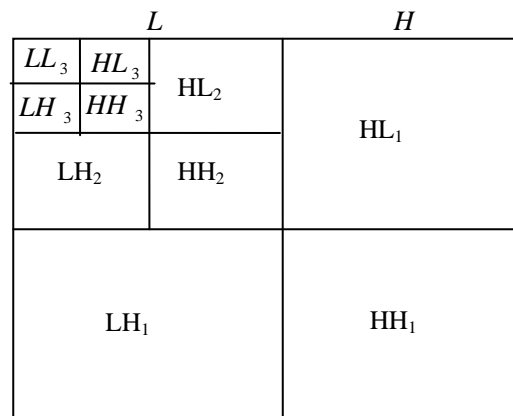


Fig. 2 Subband decomposition of an $N \times M$ image into 3 levels

II. PROPOSED LL-RAM ARCHITECTURE

The LL-RAM is used by the DWT unit to store the coefficients of the LL subband that it generates in each decomposition level, for further decompositions. In the DWT unit, the RP scans (reads) the LL-RAM, and the CP writes the LL subband coefficients in the LL-RAM. The generalized scan method proposed in [2] requires the RAM to be scanned in every clock cycle with frequency f_l , where $l=1, 2, 3$ denote single, 2- or 3-parallel DWT architectures, and to be written according to the order in which each scan method generates its output coefficients. Which implies that reads operations would coincide with writes operations. Therefore, the RAM architecture should be designed such that both read and write can take place in the same clock cycle. Thus, the first half

The authors are with the Electrical and Electronic Engineering Department, Universiti Teknologi PETRONAS, Perak, Tronoh, Malaysia (emails: kokois12@hotmail.com, herman_agustiawan@petronas.com.my).

cycle of clock f_i will be reserved for read and the second half cycle for write.

The RAM, which can be viewed as a 2-dimensional memory of size $N/2 \times M/2$, where $N = 2^n$ and $M = 2^m$, can be readily constructed from $M/2$ modules with each module having $N/2$ locations.

The block diagram of the memory module that would be used in forming the RAM architecture is shown in Fig. 3. The E signal, which is active high, enables the module for reads and writes. The module is read and the result is placed in the output bus when the signal labeled $\overline{R/W}$ is low, otherwise, it is written. The address bus is used in addressing each location in the module for read or write. The control signal labeled MS (module select) is useful when several modules are used in forming a memory. It allows through a decoder one module to be selected for read or write. The module can be read or written only when both signal E and MS are asserted high.

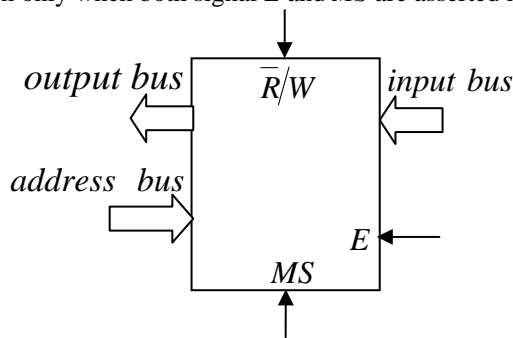


Fig. 3 Block diagram of the memory module.

The complete architecture of the RAM that facilitates both reads and writes is shown in Fig. 4(a) and (b). This architecture is based on the first scan method [1, 2]. However, the architecture can be easily modified to handle other (or higher) scan methods, as will be explained later. The decoder labeled $dcodms$ is responsible for selecting modules for reads or writes. When the architecture performs read operations, the register labeled $RMSR$ (read module select register) determines through $muxs$ which modules to be enabled. When it performs write operations, register $WMSR$ (write module select register) is used for selecting modules. Both registers are $(m-2)$ -bit counters with control signal clr (clear) and inc (increment) and operate with frequency f_i .

The multiplexer labeled $muxs$, its control signal is shown connected to clock f_i . When f_i is low, read operation takes place and $RMSR$ controls the decoder. On the other hand, when f_i is high, write operation takes place and $WMSR$ controls the decoder.

The circuit in the upper left corner of Fig. 4 (a), consisting of 3 multiplexers labeled $muxb$, a NOR gate, signal RE and the register labeled WER (write enable register), is in charge of generating signal values for the read and write signal labeled $\overline{R/W}$. The control signals of the 3 multiplexers are also shown connected to the clock f_i . When f_i is low, modules are enabled for read, otherwise, are enabled for write. Signals generated by this circuit will be described later in details.

The address bus is managed by two registers labeled $RMAR$ (read module address register) and $WMAR$ (write module address register) through $muxa$. When the

multiplexer control signal is driven low by clock f_i , read operation takes place and $RMAR$ provides addresses to modules, otherwise, write operation takes place and $WMAR$ provides address to modules.

In the following, read and write operations will be described in details. First, read operations will be described followed by write operations.

A. The LL-RAM read operations

The LL-RAM is read according to the first overlapped scan method proposed in [1, 2]. This scan method requires reading every clock cycle 3 pixels simultaneously, one from each module as follows. When f_i is low, the 3 multiplexers labeled $muxb$ pass signal RE , which is active low, to the output signals $Y0$, $Y2$, and $Y1$. The three output signals enable all memory modules for read. However, the scan method requires that in every run 3 modules should be enabled for read as follows. First, modules 1, 2, and 3 should be enabled. Then, modules 3, 4, and 5 followed by modules 5, 6, and 7 and so on. Thus, the role of the decoder labeled $dcodms$ is to guarantee that modules are enabled in the order specified above. First, the output of the decoder labeled 0 will be activated to enable modules 1, 2, and 3. Then, using the address bus, the first location of each enabled module is read into the output buses. When f_i makes a positive transition, the 3 pixels in the output buses are loaded into a temporary register labeled DL (Data latch). Then the negative transition of the clock f_i loads the 3 pixels into the RPs latches. To address the second location in each module, the negative transition of f_i increments also register $RMAR$. This process is repeated until the 3 enabled modules are read.

To enable the next 3 modules, register $RMSR$ is incremented by one, which asserts the second output of the decoder high. The decoder output labeled 1 enables modules 3, 4, and 5 for read. When all 3 modules are read, the decoder output line labeled 2 is activated by incrementing $RMSR$ again by one to enable the set that contains modules 5, 6, and 7. This process is repeated until the whole RAM is scanned.

In Fig. 4 (b), the output of modules 3 and 5 are shown connected to $mux0$ and $mux1$, respectively. These multiplexers are necessary because all modules with odd numbers, except the first, are scanned twice. For example, in the first run, when locations of module 3 are scanned they are placed in the bus labeled $bus2$, whereas in the second run they are placed in another bus labeled $bus0$. Thus, to allow these multiplexers to switch between $bus2$ and $bus0$ their control signals are connected to decoder $dcodms$ output lines labeled 0 and 1 and so on.

B. The LL-RAM write operations

How the LL-RAM should be written can be determined by examining the scan method of the DWT architecture under consideration. For example, examination of the first scan method shows that the CP would generate output coefficients column-by-column, which implies that the RAM should be written module-by-module.

In general, the RAM can be written as follows. When f_i is high, the outputs $b1$ and $b2$ of the 2-bit register labeled WER (write enable register), which is initially cleared to zero, and the output of the NOR gate are passed through multiplexers to

the outputs labeled $Y1$, $Y3$, and $Y0$, respectively. Since WER is initially zero, only $Y0$ will be asserted high, which enable for write all modules $1+3i$, where $i = 0, 1, 2 \dots, m-1$. For example, if $m=3$, then modules 1, 4, and 7 will be enabled. However, the RAM is required to be written module-by-module and in order, i.e, first module 1, then 2 followed by 3 and so on and the function of the decoder

labeled $dcodms$ is to provide this module-by-module control. Thus, the decoder output labeled 0 will be first asserted high through $WMSR$ to enable only module number 1 for write. Note that a module is enabled for write when its both signals MS and \bar{R}/W are asserted high and all modules are disabled when signal E of $dcodms$ is low.

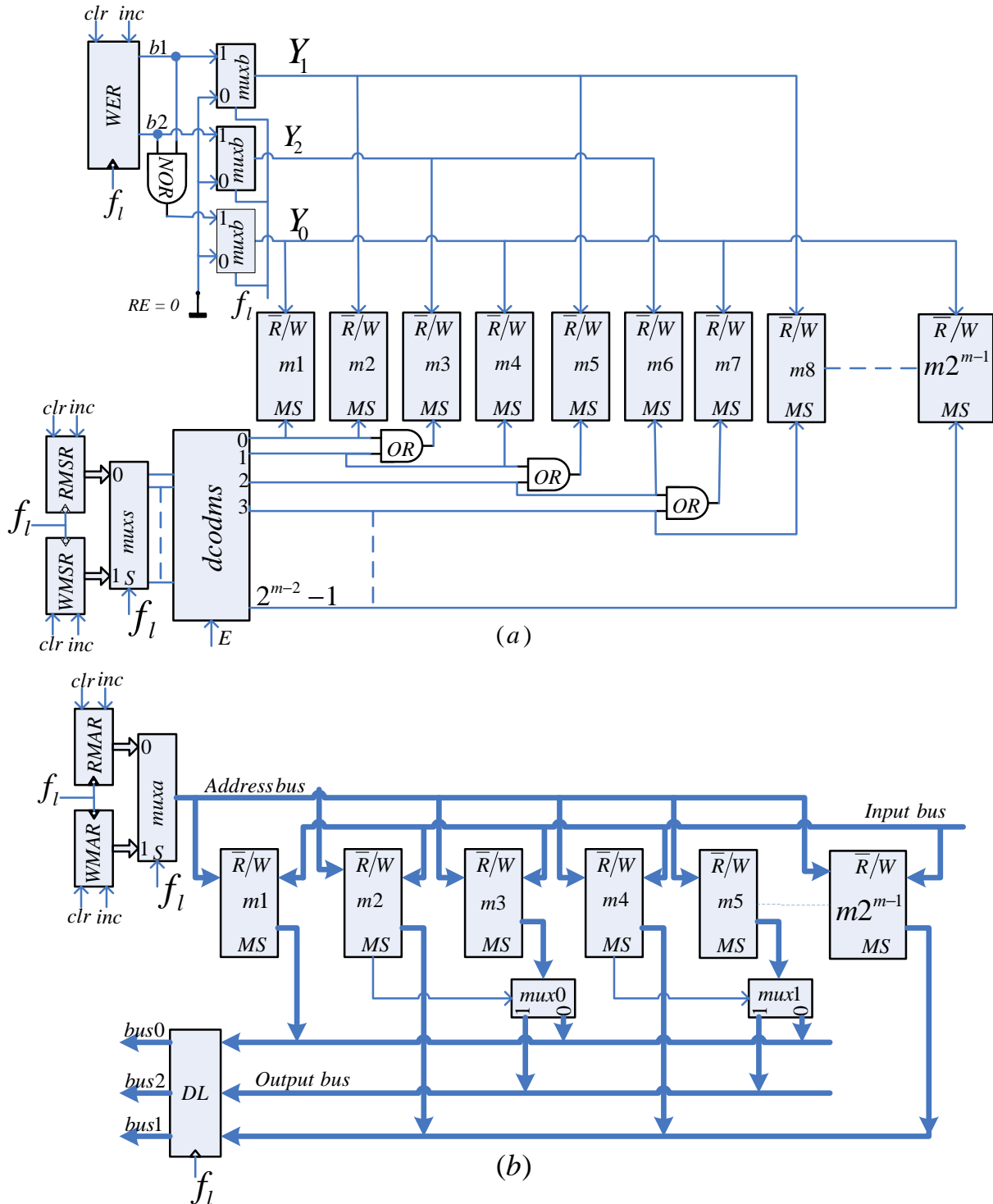


Fig. 4 (a) and (b) RAM architecture using modules

When all locations of module 1 are written, WER is incremented by one to assert only $Y1$ high. $Y1$ enables all modules labeled $2+3i$, but since the first output of the decoder is still high, only module 2 will be selected for write. When all locations of module 2 are written, WER is incremented again by one to assert this time $Y2$ high. $Y2$

enables all modules labeled $3+3i$ but since the first output of the decoder is still high, only modules 3 will be selected for write. When all locations of module 3 are written, WER is cleared to zero to set $Y0$ high, and $WMSR$ is incremented by one to assert the decoder second output labeled 1 high. Assertion of both $Y0$ and the second output of the decoder

enables only module 4 for write. This process is repeated until all modules are written.

Note that *WER* is a 2-bit register that count from 0 to 2 and repeats. Furthermore, the amount of data to be written in each decomposition level including number of modules and number of locations to be written in each module, can be determined in advance from the knowledge of the height (*N*) and width (*M*) of the image that will be processed.

A. RAM architecture modifications for higher scan methods

The RAM architecture shown in Fig. 4 can be easily modified to handle other scan method. The circuits in the upper corner of the RAM architecture, consisting of register *WER* and multiplexers labeled *muxb*, remain unchanged. However, modifications for a specific scan method in general, can be obtained by eliminating some of the OR gates whose outputs are connected to signal *MS*, as follows. For example, the second scan method, which requires 5 modules to be considered for read and two modules for write at a time, would require eliminating the first OR gate and connecting the first output of the decoder labeled *dcodms* to signal *MS* of modules *m1*, *m2*, and *m3*. While, connections to modules *m4* and *m5*, remain unchanged. Then, the connection pattern of the first 5 modules *m1*, *m2*, *m3*, *m4*, and *m5* is repeated in the next 5 modules *m5*, *m6*, *m7*, *m8*, and *m9* and so on.

Similarly, the third scan method, which requires 7 modules to be considered for read and 3 modules for write at a time, would require eliminating the first and the second OR gates and then connecting the first output of the decoder to signal *MS* of modules *m1*, *m2*, and *m3* and that of the second output to *MS* of modules *m4*, and *m5*, while connections to modules *m6* and *m7*, remain the same. The connection pattern of the first 7 modules is repeated in the next 7 modules *m7*, *m8*, *m9*, *m10*, *m11*, *m12*, and *m13* and so on.

Now, let's see how read operations are performed on the RAM architecture based on the second scan method. Since, the second scan method requires 5 modules to be considered for read at a time, the modules labeled *m1*, *m2*, *m3*, *m4*, and *m5* will be considered first. Thus, to read these modules location-by-location, registers *RMAR* and *RMSR* are reset 0. This will allow register *RMAR* to address the first location in each module and register *RMSR* to enable modules *m1*, *m2*, and *m3* through the decoder *dcodms*. Then, in the first clock cycle, when *f_i* is low, the \overline{R}/W signals of modules *m1*, *m2*, and *m3* are activated for read. This will allow the first location of each modules *m1*, *m2*, and *m3* to be read into the buses labeled *bus0*, *bus1*, and *bus2*, respectively. Then register *RMSR* is incremented by one to enable modules *m4*, and *m5* for read. When *f_i* is low, again in the second clock cycle, the first location in each modules *m4* and *m5* are read into bus1 and bus2, respectively. When this is done, register *RMAR* is incremented by one to point at the second location in each module. Register *RMSR* is reset 0 to enable again modules *m1*, *m2*, and *m3*. When *f_i* is low in the third cycle, the second location in each modules *m1*, *m2*, and *m3* are read into the buses. Then, register *RMSR* is incremented by one to enable modules *m4* and *m5* and disable *m1*, *m2*, and *m3* though the decoder labeled *dcodms*. Again, when *f_i* is low in

the fourth clock cycle, the second location of each modules *m4* and *m5* are read into bus1 and bus2, respectively. Then, register *RMAR* is incremented by one to address the third location of each module and register *RMSR* is reset 0 to enable again modules *m1*, *m2*, and *m3*. This process is repeated until the first 5 modules are read. Then the same process is applied on the next 5 modules *m5*, *m6*, *m7*, *m8*, and *m9* and so on.

Similarly, the RAM architectures for third and fourth scan method etc. can be read in the same manner described above. Note that, in the read operations described above for the second scan method, after each read operation performed on modules *m4* and *m5*, the control should return to module *m1* and repeat the process. The same situation also occurs when the next 5 modules *m5*, *m6*, *m7*, *m8*, and *m9* are considered for read and so on. That is, returning to module *m5* from module *m9* should be remembered by the control. Therefore, register *XR* is added to serve this purpose and it can be connected to register *RMSR* as shown in Fig.13. Note that in Fig. 13 register *RMSR* replaces *SMSR*. A similar problem occurs with write operations using registers *WMSR* and *WER*, and the solution shown in Fig.13, which is described in details in section III (C), can be used.

This RAM architecture would work well in DWT architectures, where pixels are required to be scanned in parallel as in parallel architectures. But, if a DWT architecture is required to scan RAM pixel-by-pixel, then in that case all OR gates in Fig. 4 (a) are eliminated and each output of decoder *dcodms* is connected only to signal *MS* of one module and the output buses are reduced to one bus.

On the other hand, how the RAM should be written depends on the scan method adopted. The first scan method requires the RAM to be written module-by-module, whereas, the second scan method requires writing into 2 modules at a time, as follows. Initially, registers *WER*, *WMSR*, and *WMAR* are set 0. Setting *WER* and *WMSR* 0 while *f_i* is high enable module 1 for write, and *WMAR* addresses the first location of module 1. This will let the first output coefficient, *LL0*, 0 to be stored in the first location of module 1. When the negative transition of clock *f_i* ending the cycle occurs, it will increment *WER* by one to enable module 2 for write. During the high pulse of the second cycle of clock *f_i*, the second coefficient labeled *LL0*, 1 is stored in the first location of module 2, while the negative transition of clock *f_i* ending the cycle clears *WER* to enable again module 1 for write and increments *WMAR* to address the second location of module 1. In this location, the third output coefficient, *LL1*, 0 is stored during the high pulse of the third cycle of clock *f_i*. The negative transition of clock *f_i* ending the third cycle, increments *WER* by one to enable module 2 again for write. During the high pulse of the fourth cycle, the fourth output coefficient, *LL1*, 1 is stored in the second location of module 2. This process is repeated until all required locations in the two modules are written. Then the same process is applied on the next 2 modules *m3* and *m4* and so on. Note that writing into the RAM does not take place every clock cycle as reading but when it occurs it coincides with reading and the order of writing coefficients occur as described above.

Similarly, the third scan method requires writing into 3 modules at a time. In general, the *i*th scan method would

require writing into i modules at a time.

B. RAM architecture using banks

The decoder labeled $dcodms$, in the RAM architecture shown in Fig. 4, is a very large decoder. This large decoder can slow down the LL-RAM's operations and can degrade its performance in terms of speed and power. Therefore, it is necessary to reduce the size of the decoder to a practical level. Furthermore, the signal labeled $Y0$, $Y1$, and $Y2$, each is shown in Fig. 4 connected to drive read/write signal labeled \overline{R}/W of several modules. Driving this large capacitive load in this way can also negatively affect the performance of the RAM. For these reasons, the bank method is introduced in Fig. 5 (a) to alleviate these problems.

Fig. 5 (a) shows a bank structure with 8 modules. The bank can contain any number of 2^b modules where $b = 1, 2, 3, \dots, m-2$. Read and write operations in the bank can be performed in the same way as described for Fig. 4. Fig. 5 (b) shows the block diagram of the bank. This block diagram is used in building the RAM architecture shown in Fig. 6. This architecture can be thought formed by dividing the architecture in Fig. 4, which can be considered as one big bank holding 2^{m-1} modules, into several smaller independent banks each holding 2^b modules. Inside the smaller banks reads and writes are performed as in the big bank but faster and more efficient.

The architecture performs read or write operations bank-by-bank and in order, $b1$ first, $b2$ second followed by $b3$ and so on. In the architecture shown in Fig. 4, the decoder labeled $dcodms$ is used for selecting modules, whereas the decoder labeled $dcodbs$ in Fig. 6 is used for selecting banks. When read operation takes place, register $RBSR$ (read bank select register) controls the decoder but when write operation takes place, the register labeled $WBSR$ (write bank select register) controls the decoder. Both registers are $(m-4)$ -bit counters with control signals clr (clear) and inc (increment).

The decoders which are attached to the banks labeled $b1$ and $b2$ etc. in the RAM architecture shown in Fig. 6, each is responsible for selecting modules when its bank is enabled by decoder $dcodbs$. When the architecture performs read operations in bank $b1$, for example, the register labeled $RMSR$ (read module select register) controls the decoder output through mux . When it performs write operations, the register labeled $WMSR$ (write module select register) controls output of the enabled decoder. Registers $RMSR$ and $WMSR$ both are 2-bit counters that count from 0 to 3 and repeats. When all modules in a bank are either read or written, the signals labeled zbr or $zbrw$ will be asserted high, respectively, indicating that the next bank can now be enabled by $dcodbs$.

To see how effective the bank method in reducing the decoder size, consider the following. Suppose, $M=2^m$ is the largest image width that can be processed by the DWT unit. Then, the maximum number of modules in the RAM will be

2^{m-1} modules with decoder size $m-2$: 2^{m-2} . Now, if each bank is structured to contain 2^b modules, then

$$2^{m-1}/2^b = 2^{m-b-1}$$

represents number of banks and number of decoder $dcodbs$ output lines. Whereas,

$$2^{m-b-1}/2^{m-2} = 2^{-b+1}$$

gives the reduction in the decoder size. Thus, if $b=3$, the decoder size decreases by a factor of 4.

III. PROPOSED SUBBAND MEMORY ARCHITECTURE

The basic architecture of the subband memory is shown in Fig. 7. The architecture is developed with two objectives in mind to achieve, that is, write operations by the DWT unit and read operations by compression unit, which are somewhat complex operations, should be performed effectively.

The strategy adopted for managing subband memory architecture for an $N \times M$ image is as follow. The first decomposition, which consist of subbands HL1, HH1, and LH1, are stored in the memory blocks labeled $HLL1$, $HHH1$, and $LHL1$, respectively. Then, the compression unit is informed to read these memory blocks. The compression unit can read each subband memory block code-block by code-block for EBCOT (Embedded Block Coded with Optimized Truncation) coding as required by JPEG2000 standard [3, 4, 5, 6]. The compression unit applies compression algorithm on each code-block independently. The compression unit first reads contents of $HLL1$, then $HHH1$, and last $LHL1$. While, the LL1 subband coefficients, which are stored in the RAM, are scanned by the RPs for further decomposition.

Subbands of the second decomposition HL2, HH2, and LH2, are stored in the subband memory blocks labeled $HLL2$, $HHH2$, and $LHL2$. While, subbands of the third decomposition are stored in the subband memory blocks labeled $HLL3$, $HHH3$, and $LHL3$, and so on. However, subbands of the last decomposition are stored in the subband memory labeled HL_{jmax} , HH_{jmax} , LH_{jmax} , and LL_{jmax} .

When the LL1 subband is decomposed into the required number of decomposition levels, the compression unit is again informed. Thus, the compression unit is informed twice during the whole decomposition process. First, when subbands of the first decomposition are available in subband memory blocks $HLL1$, $HHH1$, and $LHL1$. Second, when all subsequence decompositions of LL1 subband are completed and are stored in their respective subband memory blocks.

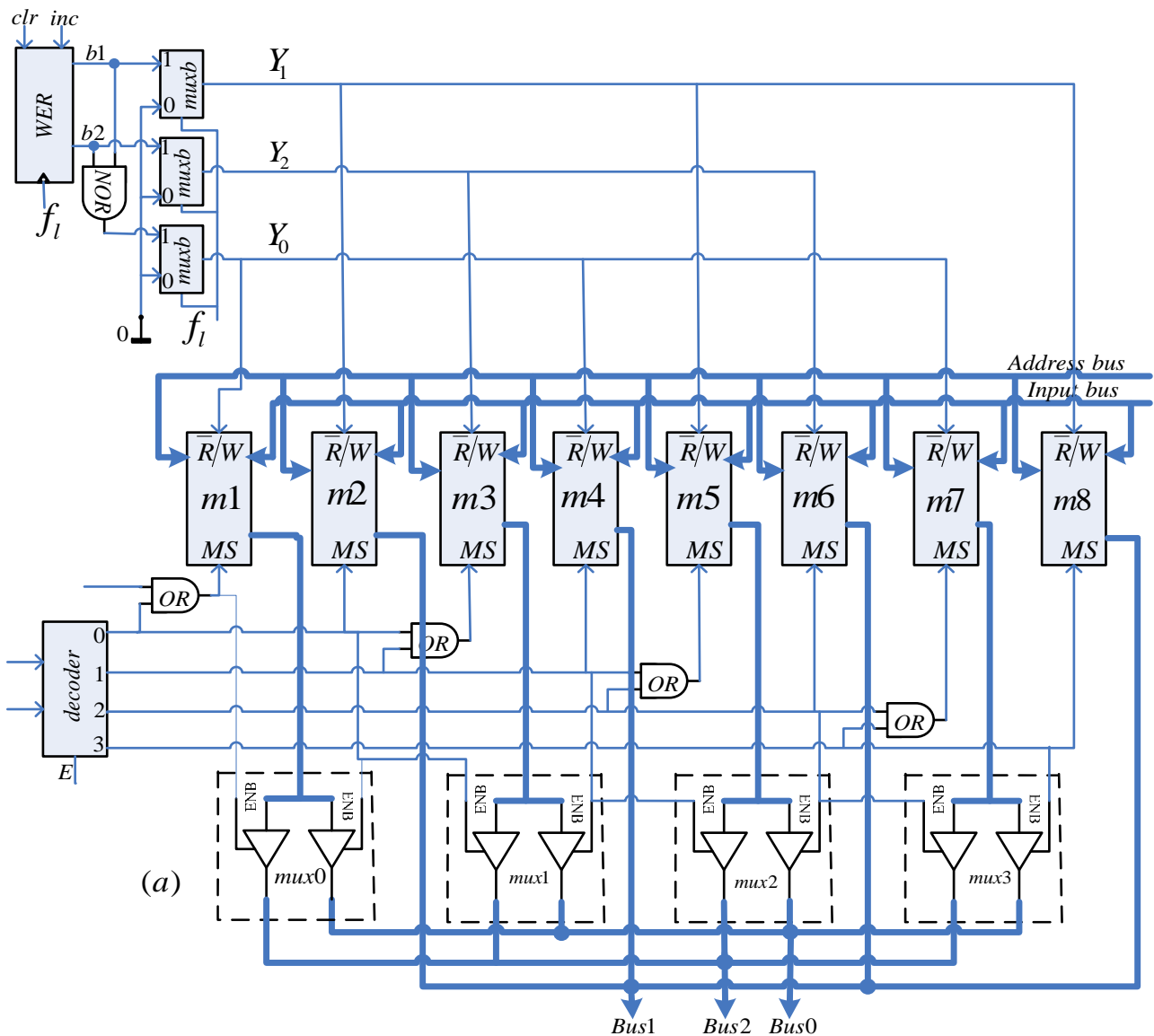


Fig. 5 (a) Bank architecture with 8 modules

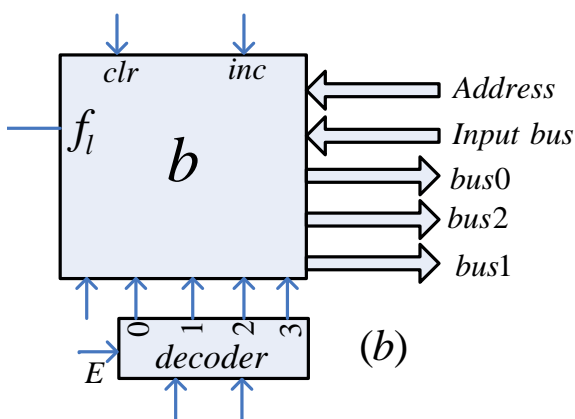


Fig. 5 (b) Block diagram of Fig. 4 (a)

A. The bank structure used in forming subband memory

In Fig. 7, each block of the subband memory labeled *HLL*, *HHI*, etc. is a 2-dimensional memory block, size $2^{n-j} \times 2^{m-j}$, where $j=1, 2, 3 \dots jmax$ and $jmax$ is the maximum number of decomposition levels allowed. Two methods of forming a bank containing modules are shown Figs. 8 and 9. The first

bank shown in Fig. 8 contains 2^b modules. When *EM* is asserted high, it enables the bank for both read and writes operations. Which module to read or write is determined by the decoder and the address lines are used to address each location in the selected module starting from location zero to location $2^{n-j}-1$. The block diagram of the bank is shown in Fig. 8 (b).

The second bank is illustrated in Fig. 9(a) and its block diagram is shown in Fig. 9 (b). It consists of two small banks, the upper and the lower banks, which in turn form a larger bank. The second bank method reduces the decoder size by $\frac{1}{2}$ as compared with the first bank method, and allows more packing of modules into a bank. The number of modules in the larger bank is 2^b , while the lower and upper banks each contains (2^{b-1}) modules as indicated in Fig. 9 (a). Reads or writes into the bank take place module-by-module. Modules in the upper bank are read (or written) first followed by the lower bank modules. When signal *E* is enabled, the upper bank is selected by asserting the signal *EUB* (enable upper bank), whereas the lower bank is selected by asserting the signal *ELB* (enable lower bank). Modules in the upper or lower banks are selected by the decoder. Modules are

selected in the order specified by the decoder, which selects a module at a time.

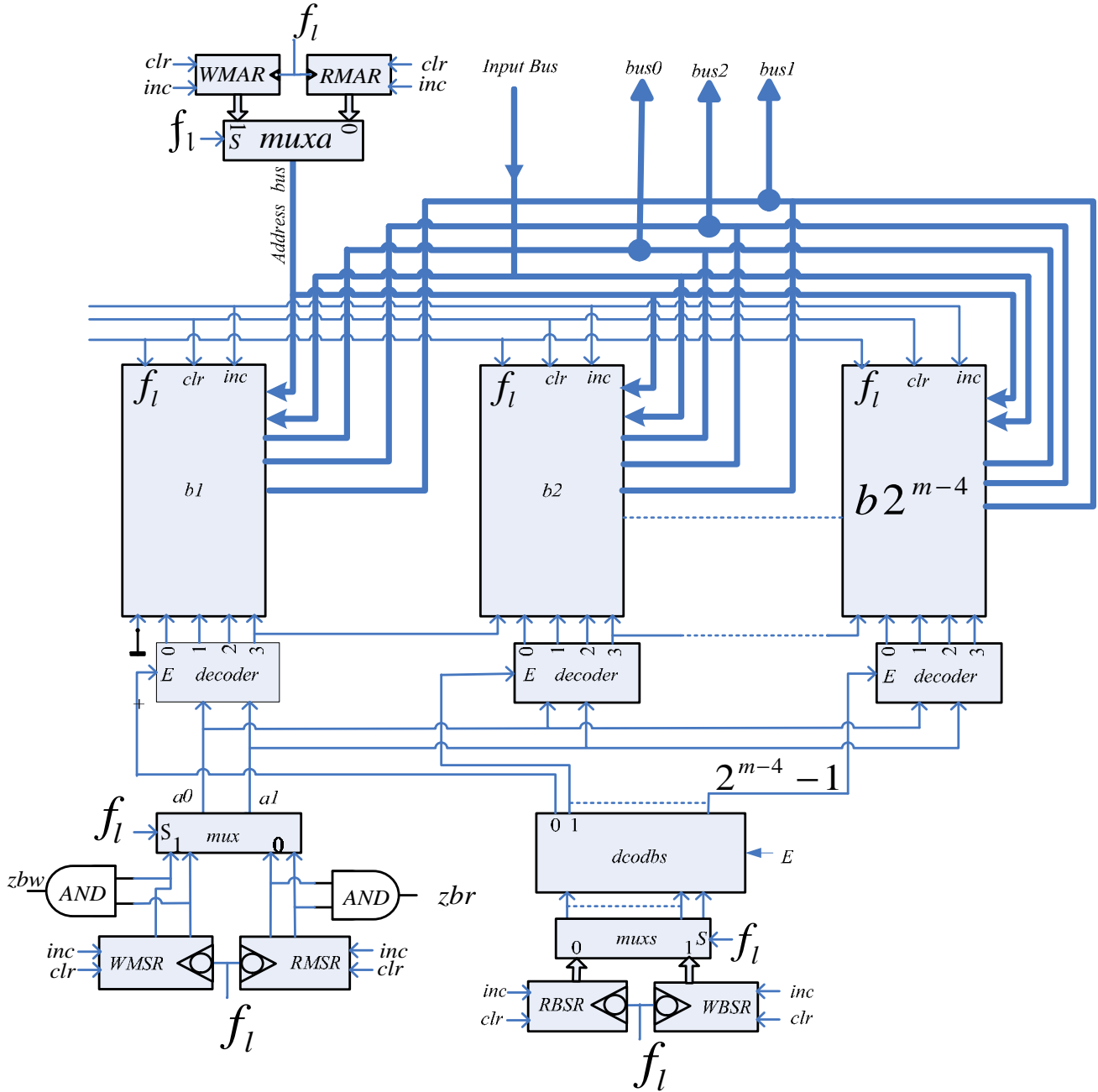


Fig. 6 RAM architecture using bank

Using the block diagram of the second bank, the subband memory block architecture shown Fig. 10 (a) is formed. The architecture consists of 2^{m-b_j} banks; each bank contains 2^b memory modules and each module with 2^{n_j} locations.

The decoder labeled *dcodbs* in Fig. 10(a) selects one bank at a time for reads or writes. Banks are selected in order, first *b1*, and second *b2* and so on. The modules inside a selected bank are enabled one at a time through the lines labeled *MS* (module select). The line labeled \overline{UB}/LB enables the upper bank when asserted low and the lower bank when asserted high. Reads or writes occur when signal *E* of decoder *dcodbs* is high, otherwise, no reads or writes occur.

The block diagram of the architecture is shown in Fig. 10(b). This block is further used for forming the subband memory architecture shown in Fig. 7. That is, each block in Fig. 7 is replaced by the block diagram shown in Fig. 10(b).

Suppose, for instant, the largest image size that can be processed is $N=M=2^{10}$, *b* is 3, and the maximum number of decomposition levels, *jmax* is 7. Then, this implies that the subband memory blocks labeled *HL1*, *HH1*, and *LH1* for *j=1*, each should be designed to contain 64 banks and each module in a bank should have 2^9 locations. The blocks of the second level labeled *HL2*, *HH2*, and *LH2* for *j=2*, each should contain 32 banks and each module in a bank should contain 2^8 memory locations. Similarly, the sizes of the subband memory blocks for third and fourth to *jmax*th levels can be determined. Note that the blocks of the last level labeled *HL_{jmax}*, *HH_{jmax}*, *LH_{jmax}*, and *LL_{jmax}* for *j=jmax=7*, each must be designed with one bank with each module in the bank having 2^3 memory locations. That is each block size will be 8x8.

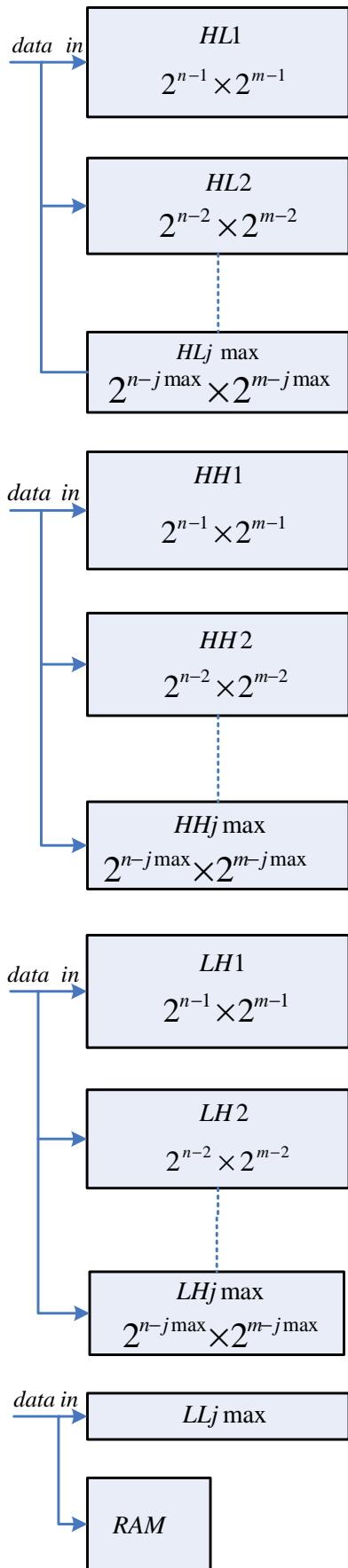


Fig. 7 Subband memory architecture

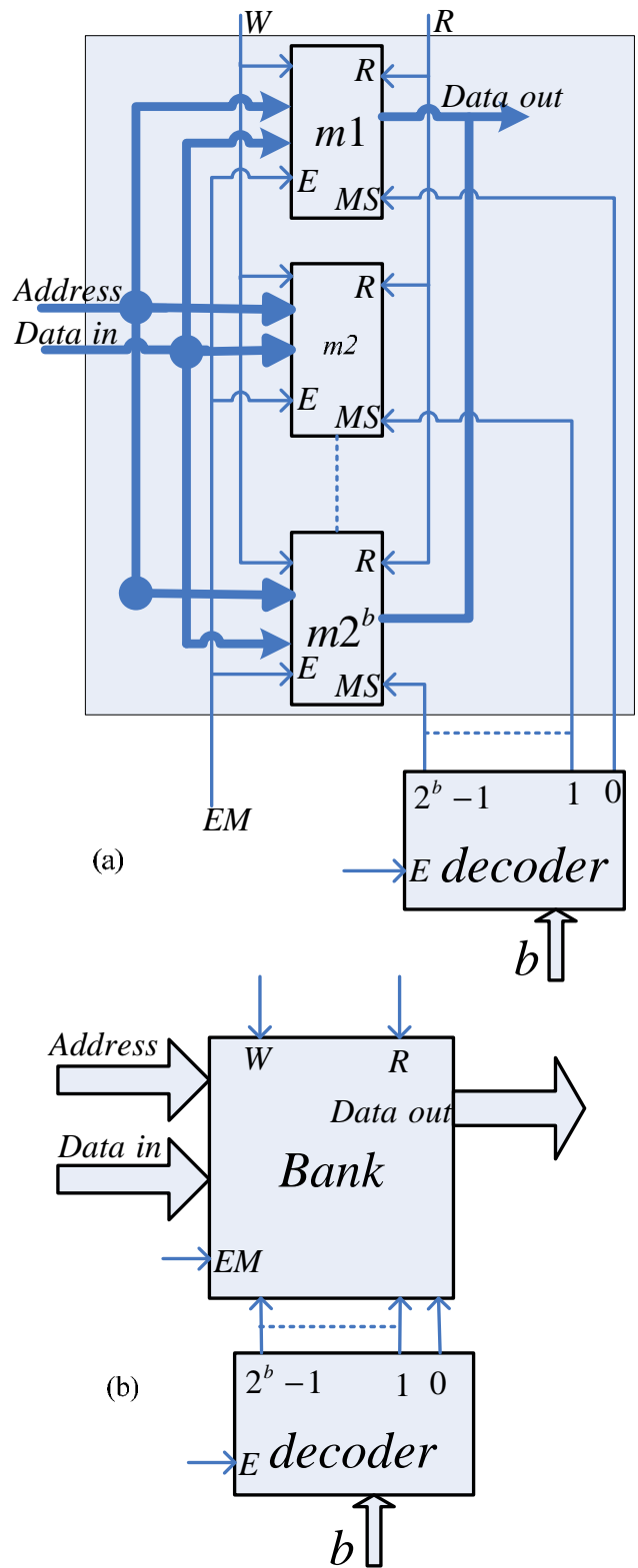


Fig. 8 (a) Structure of the first bank (b) its block diagram

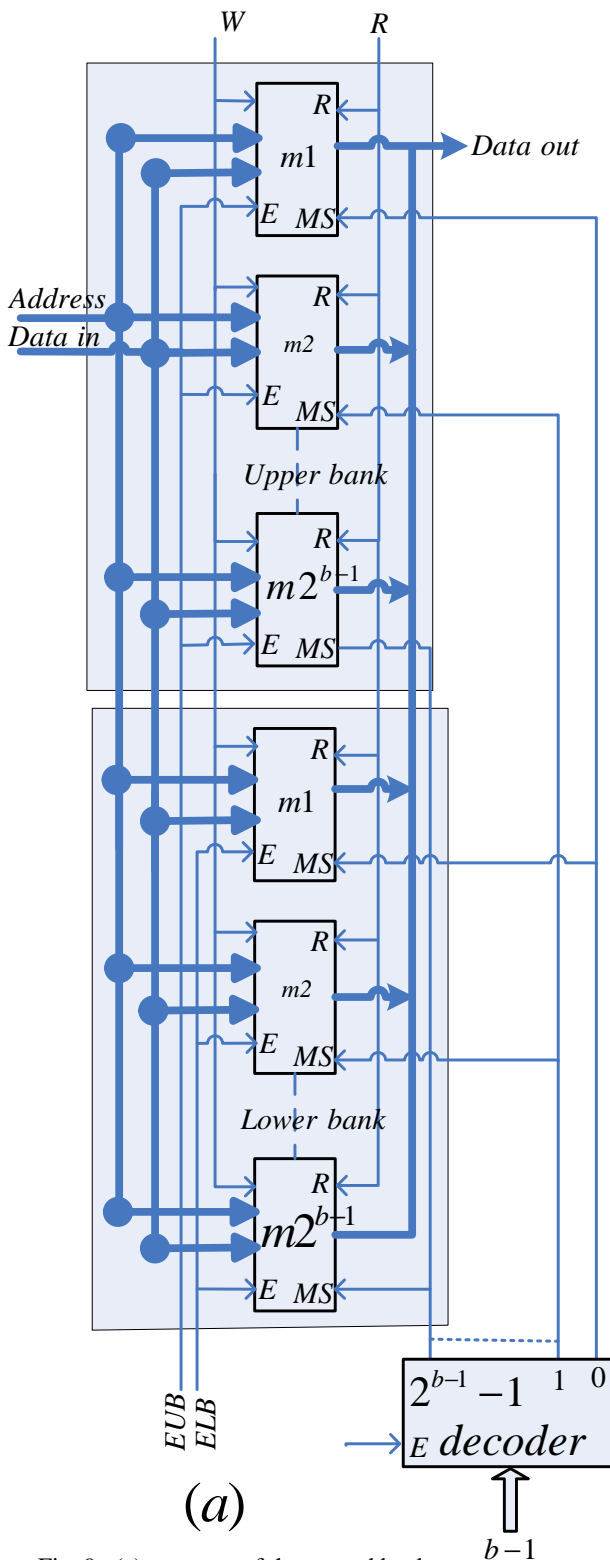


Fig. 9 (a) structure of the second bank

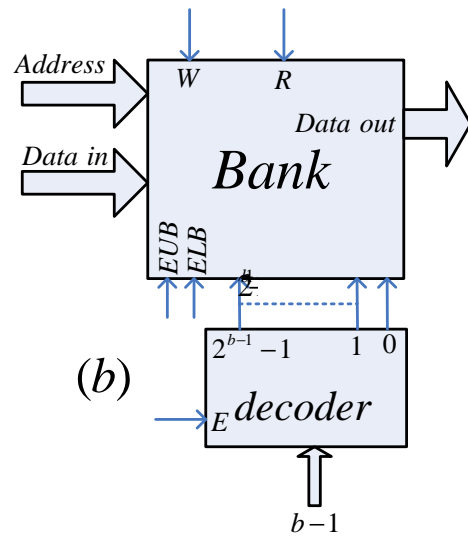


Fig. 9 (b) Block diagram of Fig. 9 (a)

A. Details of the subband memory architecture

The details of the subband memory architecture and its interconnections are shown in Figs. 11 and 12. These two figures together give the complete architecture of the subband memory. The architecture is designed to allow the DWT unit to write into subband memory and the compression unit to read it.

The two sets of registers labeled MAR1 and MAR2 in Fig. 11 supply address to modules that are selected for reads or writes. MAR1, which is an $(n-1)$ -bit counter, provides addresses to modules of the first level memory blocks labeled *HL1*, *HH1*, and *LH1*. While MAR2, an $(n-2)$ -bit counter, provides addresses to all memory blocks that lay below the first level. Note that in Fig. 12, the 3 signals labeled *BS*, $\overline{UB/LB}$, and *MS* are grouped together and are connected to the output of the register labeled *SMSR* (subband module select register), where *BS* and *MS* occupy the least and the most significant bits positions, respectively. Grouping of these 3 signals in this way facilitate banks and modules within a bank to be accessed successively. These signals can be generated by register *SMSR*, which is a simple counter. This register will drive these signals and will determine their values by simply counting from 0 to 2^{m-j} , where 2^{m-j} represents number of modules to be written (or read) in each subband memory block. The value in the *SMSR* gives, when a block of the subband memory is enabled for reads (or writes), the bank number and the module number selected in the upper or lower bank. *SMSR1* is an $(m-1)$ -bit register and is used along with *MAR1* to address only subband memory blocks of the first level. Whereas *SMSR2*, which is an $(m-2)$ -bit register, is used along with *MAR2* to address all subband memory blocks that lay after the first level.

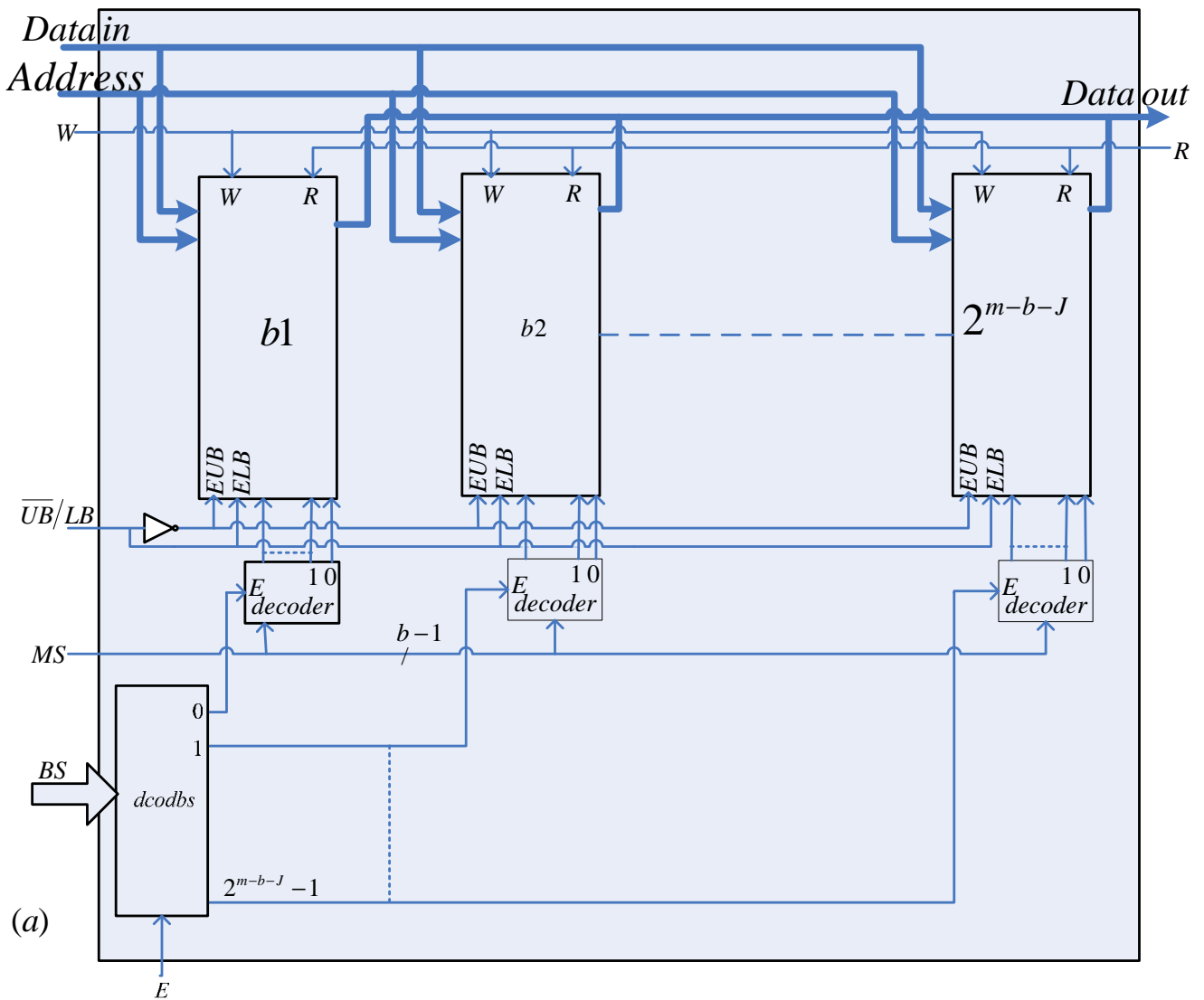


Fig. 10 (a) subband memory block architecture built using the block diagram of the second bank

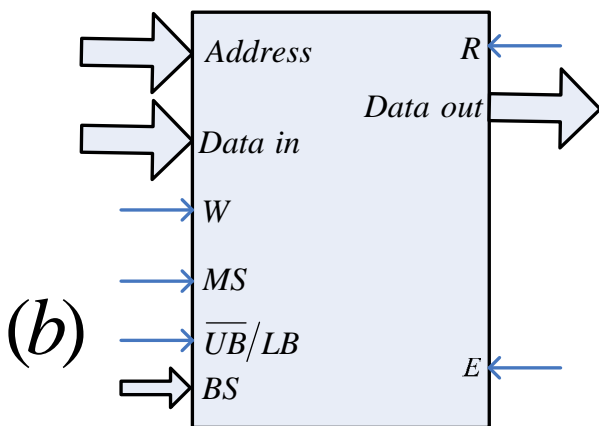


Fig. 10 (b) Block diagram of Fig. 10 (a)

Single pipelined architectures as in [1, 2] generate two output coefficients each clock cycle, reference to the processor's clock. The two output coefficients might belong to either subbands LH and LL or subbands HL and HH. In the first case, one coefficient (the high coefficient) is stored in the subband memory block *LH* using group *B* registers, while the other coefficient (low coefficient) is passed to LL-RAM where it is stored. In the second case, simultaneously, the low and high coefficients are stored in the subband memory blocks *HL* and *HH*, respectively, using group *A* registers. On the other hand, the parallel architectures generate 4 output coefficients every clock cycle that belong to subbands HL, HH, LH, and LL. The 3 coefficients of subbands HL, HH, and LH are stored in the subband memory blocks *HL*, *HH*, and *LH* using both groups *A* and *B* registers, while coefficient of subband LL is passed to LL-RAM.

Figs. 11 and 12 also show two groups of registers labeled *A* and *B*. These registers would make it possible to control storing of output coefficients in the subband memory by either single or parallel pipelined 2-D DWT architectures.

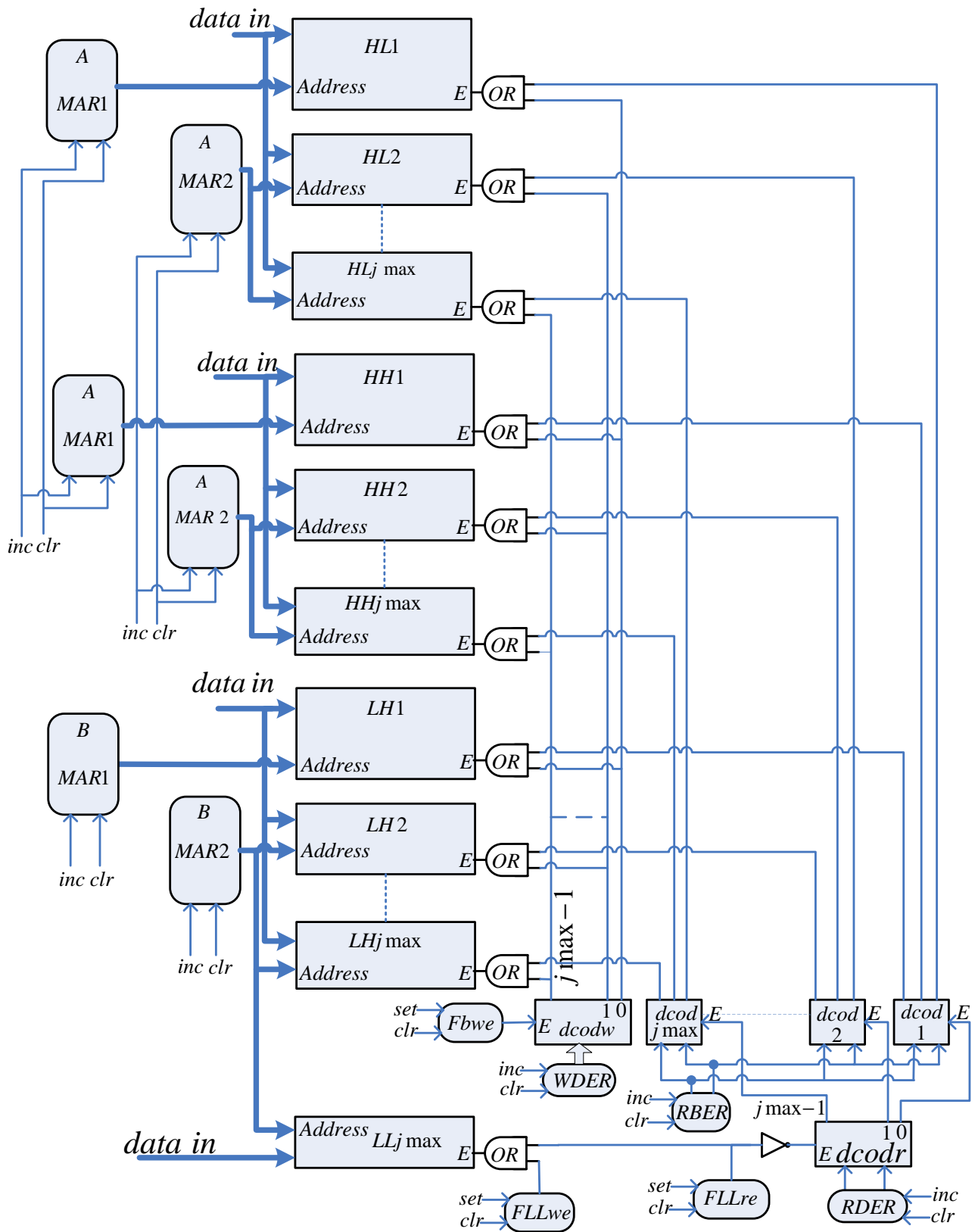


Fig. 11 Architecture of the subband memory

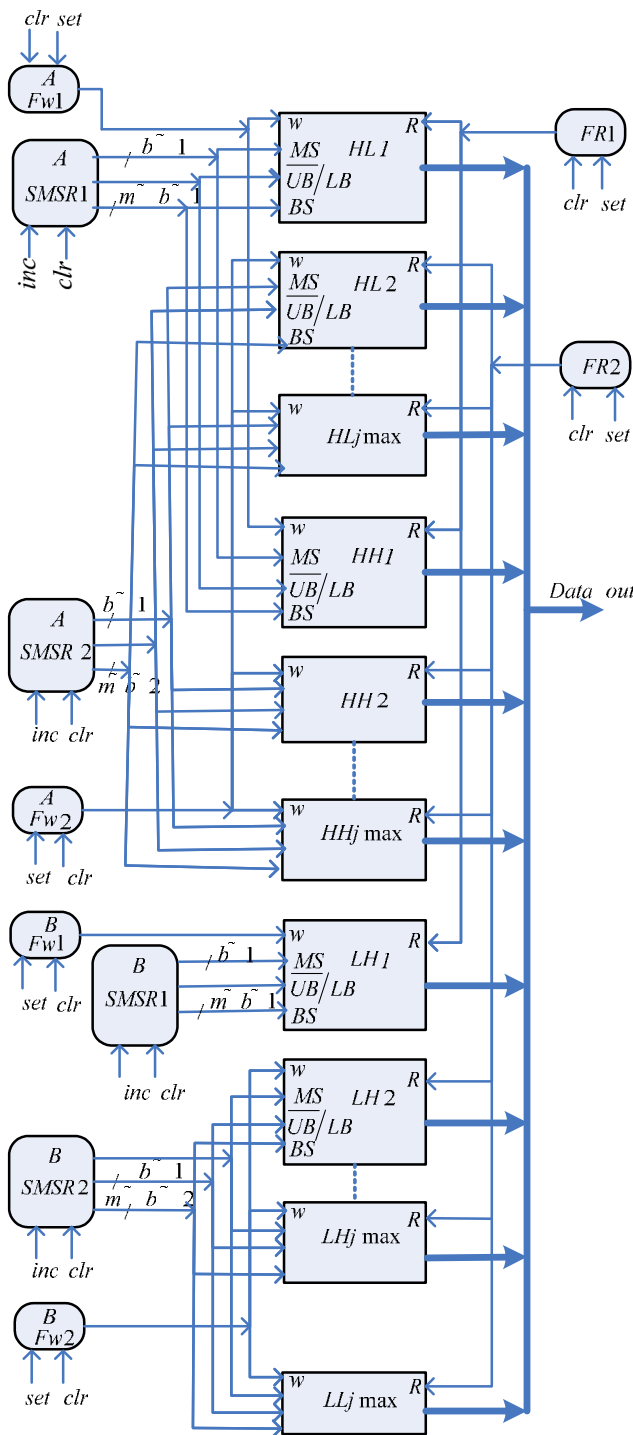


Fig. 12 Architecture of subband memory

Suppose, now the DWT unit is requested to process, for example, a 256x200 image and since the largest image size that can be process is $N=M=2^{10}$ and the maximum number of decomposition levels allowed, j_{max} is 7, then this image can be decomposed into 5 levels of decomposition. The first decomposition will generate 4 subbands, each of size 128x100. The 3 subbands HL1, HH1, and LH1 will be written into the subband memory blocks HL1, HH1, and LH1. That is, in each subband memory blocks HL1, HH1, and LH1, 100 modules will be written and each module addresses range from 0 to 127. SMSR1 selects a bank and a module in the bank to be written, while MARI generates addresses for accessing locations in the selected module.

The second decomposition generates also 4 subbands images, each of size 64x50. The 3 subbands HL2, HH2, and LH2 will be written into the subband memory blocks HL2, HH2, and LH2. In each subband memory block, SMSR2 is used for selecting a bank and a module in the bank and MAR2 is used for generating addresses for accessing each location in the module.

The third decomposition generates 4 subbands HL3, HH3, LH3, and LL3 each of size 32x25. The first 3 subbands are stored in the subband memory blocks HL3, HH3, and LH3, respectively.

The fourth decomposition generates 4 subbands HL4, HH4, LH4, and LL4. Subbands HL4 and HH4 each is of size 16x12, while subbands LH4 and LL4 each is of size 16x13. The first 3 subbands are stored in the subband memory blocks HL4, HH4, and LH4, respectively.

The fifth decomposition, which is the last decomposition, generates 4 subbands HL5, HH5, LH5, and LL5. Subbands HL5 and HH5 each is of size 8x6, while subbands LH5 and LL5 each is of size 8x7. These 4 subbands are stored in the subband memory blocks HL5, HH5, LH5, and LLjmax, respectively. Note that the LLj subband of the last decomposition should always be stored in the subband memory block labeled LLjmax.

The decoder labeled $dcodw$ along with the register labeled WDER and the FFs labeled Fbwe, Fw1, Fw2, and FLLwe are used for enabling subband memory for writes. Whereas the decoder labeled $dcodr$ along with the two registers labeled RDER and RBER, and the FFs labeled FR1, FR2, and FLLre are used by compression unit for enabling subband memory for reads.

The two registers labeled WDER (write decomposition register) and RDER (read decomposition register) both are counter that count from 0 to $j-1$. These registers are initially designed to count from 0 to $j_{max}-1$, where j_{max} is the maximum number of decomposition allowed. In a decomposition process, the required number of decompositions, j desired should be provided by loading j into a register. Moreover, the order of writing into the subband memory blocks are controlled by WDER, whereas the order of reading them by compression unit are controlled by the two registers labeled RDER and RBER.

To write subbands coefficients of the first level decomposition into subband memory, the DWT unit initially clears registers SMSR1, MARI, WDER, and the flip-flop (FF) labeled FLLre to zero and sets the FFs labeled Fw1 and Fbwe 1. Fbwe enables the decoder labeled $dcodw$ and since WDER is 0, the first output of the decoder labeled 0 is activated. Activation of this output signal enables subband memory blocks HL1, HH1, and LH1 for write. The value in register SMSR1 determines the bank number and the module number to be written in each enabled subband memory block. While register MARI is used for addressing each location in the 3 selected modules. When all locations of the 3 modules are written, register SMSR1 is incremented by one to select the next 3 modules, one from each enabled blocks. This process is repeated until all modules in the 3 enabled subband memory blocks are written. Then, the DWT unit resets FF Fw1 0 and informs the compression unit. The compression

unit responds by reading contents of the subband memory blocks $HL1$, $HH1$, and $LH1$, and compresses them independently.

Meanwhile, the DWT unit moves to the second level in the subband memory by incrementing register $WDER$ and setting $Fw2$ 1. This allows the DWT unit to write subbands coefficients of the second decomposition into the subband memory. Incrementing register $WDER$ by one activates the second output of the decoder labeled $dcodw$. This output enables subband memory blocks labeled $HL2$, $HH2$, and $LH2$ for write. In addition, registers $SMSR2$ and $MAR2$ are reset zero. Resetting $SMSR2$ zero, selects the first bank in each one of the 3 enabled blocks and enables the first module in each selected bank for write. Register $MAR2$ is used for addressing each location in the 3 enabled modules. The process of writing into these modules proceeds as that of the first level. When all modules in the 3 enabled subband memory blocks are written, the third level in the subband memory is enabled by incrementing $WDER$ by one. This activates the third output of the decoder, which enables blocks $HL3$, $HH3$, and $LH3$ for write. This process is continued until the last decomposition level is reached. When all subbands coefficients of the last decomposition are written, the DWT unit will inform again the compression unit. It will also reset $Fbwe$ and $Fw2$ zero to disable subband memory for writes, until it read by compression unit.

On the other hand, reading of subband memory by compression unit proceeds as follows. As soon as the compression unit receives the first signal from DWT unit, confirming that the first level decomposition is completed and its subbands coefficients are available in the subband memory blocks $HL1$, $HH1$, and $LH1$, the compression unit clears registers $RDER$, $RBER$, $SMSR1$, and $MAR1$ to zero and sets $FFFR1$ 1. Resetting $RDER$ and $RBER$ zero enable the subband memory block labeled $HL1$ for read. While resetting $SMSR1$ selects the first bank in block $HL1$ and enables the first module in the bank. Then $MAR1$ is used for addressing each location in the module for read. The next module is enabled by incrementing $SMSR1$ by one. The compression unit continues in this fashion until all $HL1$ modules are read. Then $RBER$ is incremented by one to enable $HH1$ for read and $SMSR1$ and $MAR1$ are reset zero to select the first bank and enable the first module in the bank. Then, reading of block $HH1$ proceeds as that of $HL1$.

To enable block $LH1$, the compression unit increments again $RBER$ by one and resets $SMSR1$ and $MAR1$ zero. When all modules in $LH1$ are read and the second signal from DWT unit is received to confirm that all subband coefficients, starting from the second level decomposition, are available in their respective subband memory blocks, register $RDER$ is incremented by one to enable the second decoder ($dcod2$) and $RBER$ is reset zero to activate the first output of the decoder. In addition, $FR1$ is reset zero and $FR2$ is set 1. Activation of the first output of the second decoder enables block $HL2$ for read. Then compression unit uses registers $SMSR2$ and $MAR2$ to read block $HL2$ module-by-module as described in the first level. After $HL2$ is read, $HH2$ is enabled for read then $LH2$. The compression unit reads subband memory level-by-level and each level is read block-by-block and each block is read bank-by-bank and each bank is read module-by-module until

it reaches the last subband memory block labeled LL_{jmax} . To read block LL_{jmax} , the compression unit sets $FLLre$ 1 to enable this block for read and then uses registers $SMSR2$ and $MAR2$ to read its contents.

Subband memory architecture for higher scan methods

With first scan method, writing into each subband memory block takes place module-by-module. That is, only one module in each block will be enabled for write at a time. The second and the third scan methods require writing into 2 and 3 modules at time in each block, respectively. In general, the i th scan method requires writing into i modules in each subband memory block.

To see how this can take place, consider, for example, 2-parallel architecture based on the third scan method. This architecture will yield 4 output coefficients every clock cycle, reference to the processor clock $f_2/2$. The first 3 output coefficients labeled $HH0, 0, HL0, 0$, and $LH0, 0$ should be stored in the first location of the first module in each subband memory blocks $HH1, HL1$, and $LH1$, respectively. The second output coefficients $HH0, 1, HL0, 1$, and $LH0, 1$ should be stored in the first location of the second module in each subband memory blocks $HH1, HL1$, and $LH1$, respectively. The third output coefficients $HH0, 2, HL0, 2$, and $LH0, 2$ should be stored in the first location of the third module in each subband memory blocks $HH1, HL1$, and $LH1$, respectively. The fourth output coefficients $HH1, 0, HL1, 0$, and $LH1, 0$ should be stored in the second location of the first module in each subband memory blocks $HH1, HL1$, and $LH1$, respectively.

It is obvious, after the third output coefficients are stored, the process returns to the first module in each block to repeat the process until the first 3 modules in each subband memory blocks $HH1, HL1$, and $LH1$ are written. Similarly, the next 3 modules in each subband memory blocks $HH1, HL1$, and $LH1$ are written and so on. When all modules in the subband memory blocks $HH1, HL1$, and $LH1$ are written, the process moves to the second level of the subband memory blocks $HH2, HL2$, and $LH2$ to store subbands coefficients of the second decomposition level. However, in order for the control to move effectively between 3 modules, the first modules number should be remembered by the control. For this reason, register XR is added and is connected to register $SMSR$ as shown in Fig. 13.

Initially, registers $SMSR$, MAR and XR are reset 0. When $SMSR$ is reset, BS enables the first bank in each subband memory blocks $HH1, HL1$, and $LH1$, while \overline{UB} and MS enable the upper bank and the first module in each bank, respectively. This will allow the first 3 output coefficients $HH0, 0, HL0, 0$, and $LH0, 0$ to be stored in the first location of each module in blocks $HH1, HL1$, and $LH1$, respectively, addressed by MAR . Then register $SMSR$ is incremented by one to enable the second module in each subband memory blocks $HH1, HL1$, and $LH1$. This will allow the second output coefficients $HH0, 1, HL0, 1$, and $LH0, 1$ to be stored in the first location of the second module in each subband memory blocks $HH1, HL1$, and $LH1$, respectively. To store the third output coefficients $HH0, 2, HL0, 2$, and $LH0, 2$ in

the first location of the third module in each block, register *SMSR* is again incremented by one.

Since, the fourth output coefficients *HH1*, 0, *HL1*, 0, and *LH1*, 0 should be stored in the second location of the first module in each subband memory blocks *HH1*, *HL1*, and *LH1*, respectively, register *XR*, which is 0, is loaded into *SMSR* while *MAR* is incremented by one to address the second location in each module. This process is repeated until the first 3 modules in each block are written. At that point where run 2 begins, *SMSR* will be 2, indicating that the third module is the last module written in each block. To enable the fourth module in each block, register *SMSR* is incremented by one and the result is loaded into *XR* so that this module number can be remembered, while *MAR* is reset 0 to address the first location in each module. This will allow the first 3 output coefficients of run 2 to be stored in the first location of each module enabled in the subband memory blocks *HH1*, *HL1*, and *LH1*. Then, register *SMSR* is incremented by one to enable the fifth module in each block. When the first location of each module is written, register *SMSR* is incremented again by one to enable the sixth module in each block. When the first location of each module is written, register *MAR* is incremented by 1 and register *XR* is loaded into *SMSR* to enable again the fourth module in each block and the process repeats. When all modules in the first level are written, the subband memory blocks *HH2*, *HL2*, and *LH2*, in the second level, are enabled and writing into these blocks proceeds as in the first level.

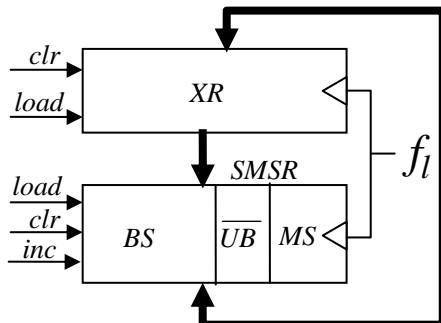


Fig.13 Incorporation of register XR

The total subband memory size required is NM , which is half of the memory size ($2NM$) required in the architecture proposed in [5]. However, in order to make the subband memory size practically reasonable and to allow each subband block equals or matches the maximum code-block size 64×64 required in the JPEG2000 standard [6, 4], we suggest a maximum tile image size of 128×128 to be processed by the DWT unit. Then each subband block (code-block) can be scanned and processed by any scheme such as the 4-Square Compression (4-SqC) scheme proposed in [6].

I. CONCLUSIONS

In this paper, two novel VLSI memory architectures for lifting-based 2-D DWT architectures for $5/3$ and $9/7$ are proposed. First, based on the first scan method, the LL-RAM and subband memory architectures are developed. Then how the two memory architectures can be modified for high scan methods is illustrated. In addition, banking technique is used

to form more efficient memory architectures in terms of speed and power consumption. The advantage of two proposed memory architectures is that they can be easily incorporated into single or parallel DWT architectures.

REFERENCES

- [1] I. Saeed, H. Agustawan., "Lifting-based VLSI architectures for 2-dimensional discrete wavelet transform for effective image Compression," in: Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol. 1, IMECS'08, Hong Kong, PP. 339-347, Newswood Limited, 2008.
- [2] Ibrahim Saeed and Herman Agustawan, "high-speed and power Efficient lifting-based VLSI architectures for two-dimensional discrete Wavelet transform," proceedings of the IEEE Second Asia International Conference on Modelling and Simulation, AMS 2008, PP. 998-1005.
- [3] R. Jain, P. Ranjan Panda, "Memory architecture exploration for power efficient 2D-discrete wavelet transform," in: proceedings of the IEEE 20th International Conference on VLSI Design, 2007, PP. 813-818.
- [4] M. Y Chiu, K. B Lee, C. W. Jen, "Optimal, Optimal data transfer and buffering schemes for JPEG2000 encoder," in: Proc. IEEE Workshop on Signal Processing Systems (SIPS'03), pp. 177-182, August 2003.
- [5] G. C. Jung, S. M. Park, J. H. Kim, "An efficient VLSI architecture for JPEG2000 encoder," in: proceedings of ISCAS, PP. 1203-1206, IEEE, 2007.
- [6] A. K. Gupta, S. Nooshabadi, D. Taubman, "Efficient data transfer Techniques and VLSI architecture for DWT-block coder integration of JPEG2000 encoder," in: proceedings of ISCAS, PP. 1365-1368, IEEE, 2007.
- [7] David S. and Michael W. "JPEG 2000 image compression fundamentals, standards and practice," Kluwer Academic publishers, 2002