

Hardware Implementation of *Realloc* function

Witawas Srisa-an, C.D. Lo, and Morris Chang

**Department of Computer Science
Illinois Institute of Technology**

IWV'99

Objectives

- To analyze all the steps necessary to complete reallocation process in hardware domain.
- To provide the designs of all necessary components to implement *realloc* and *expand* functions. These new components should be elegant in functionalities and can easily be integrated into the existing designs.
- To reuse as many components as possible to accomplish reallocation process.

Outline

- Motivation
- Previous work
- Reallocation function overview
- Implementation of *expand()*
- Completing reallocation process
- Conclusion

Motivation

- The memory intensive nature of Object-Oriented applications creates the need for high performance dynamic memory manager.
 - + Java spends 20% of execution time in GC.
 - + Frequency of heap-allocation in C++ can be as much as 10 times more than comparable C application.
- Hardware approach can improve the performance of dynamic memory management in both speed and deterministic turnaround time.
 - + Advancement in VLSI technology allows instruction specific applications to be mapped into hardware (MMX, 3-D now).

- + Dynamic memory management functions can be implemented in hardware as well. Substantial improvement can be gained.
- Our long term goal is to perform dynamic memory management in hardware. In order to do so, all of the dynamic memory management functions need to be designed (malloc, free, realloc, and GC). The designs of *malloc()* and *free()* have already been proposed; however, the design of *realloc()* and GC has not.
- Our design can be incorporated into:
 - + CPU (as DMMU co-processor)
 - + Hardware implemented JAVA chip
 - + Self managed memory system (for embedded system)

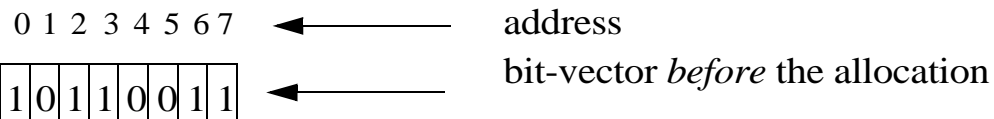
Previous work

- In 1996, Chang and Gehringer proposed a hardware implementation of *malloc* function.
 - + The allocation status is kept on bit-map.
 - + The allocation is done through modified buddy system.
 - + It has four major components.
 - bit map is used to maintain allocation status for each memory block
 - **or-gate** tree is used to search for the first available memory chunk for a particular size
 - **and-gate** tree is used to find the address of to be allocated memory chunk
 - *bit-flipper* is used to mark allocated bits on the bit map

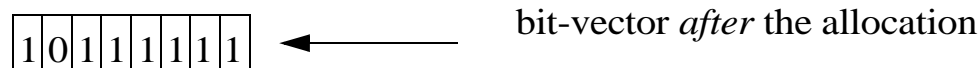
Previous work (cont.)

- In order to allocate a memory chunk, three steps are needed.
 - + Locating free block: the **or-gate** tree
 - + Finding the first zero and its address: the **and-gate** tree
 - + Marking allocated memory: the *bit-flipper*

An example of allocating 2 blocks of memory

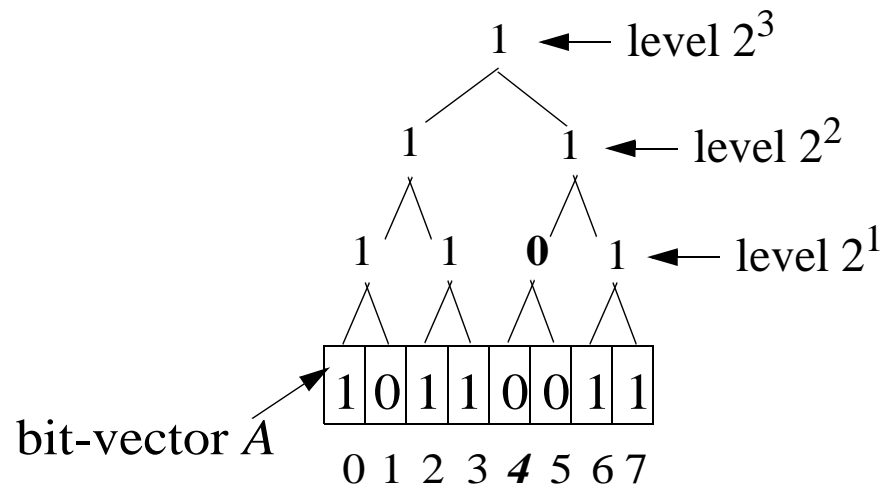


- (a) Yes, there is enough memory space to fill the request for 2 blocks of memory.
(b) The address is 100_2 .
(c) The bits at 100_2 and 101_2 need be flipped.



Previous work (cont.)

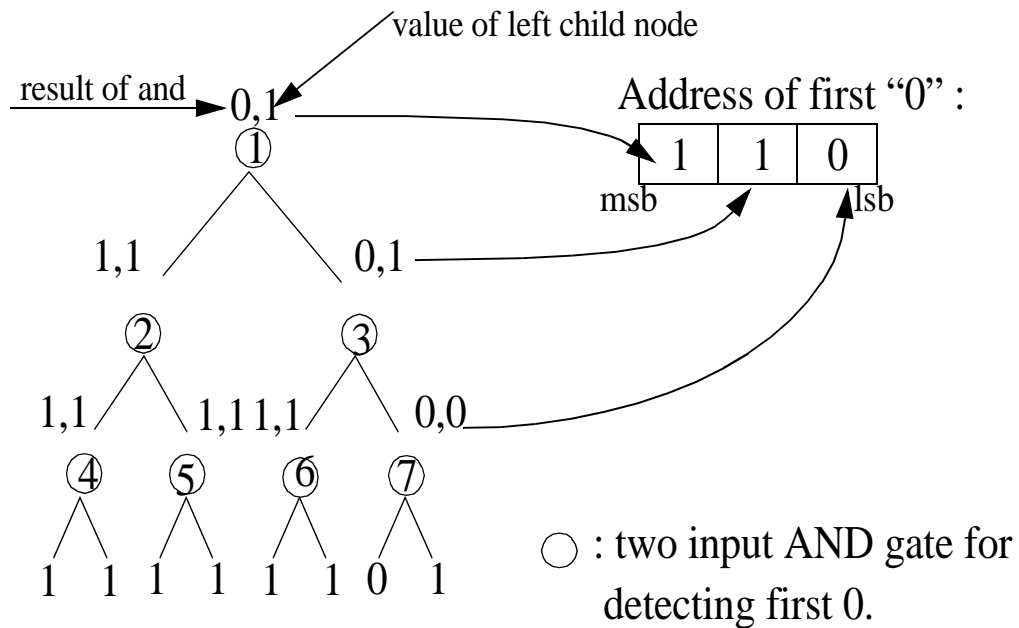
- Locating free blocks: the **or-gate** tree
-



For the two-block request, the address of free memory space = 100.

Previous work (cont.)

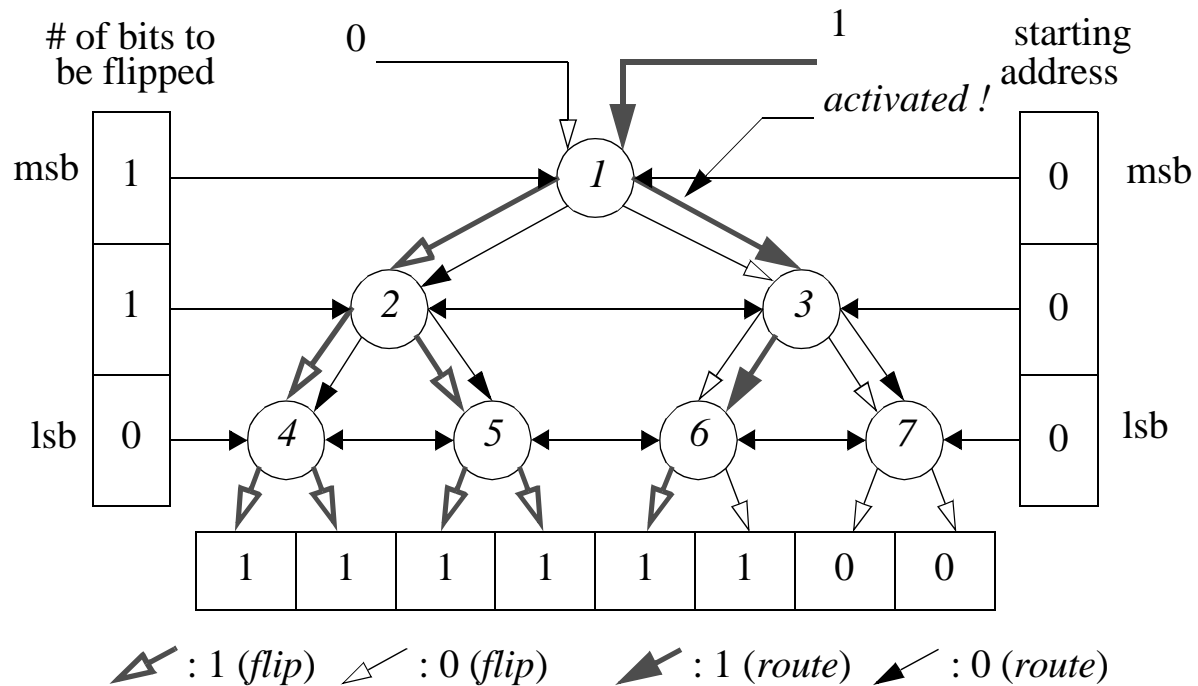
- Finding the first zero and its address: the **and-gate** tree



Previous work (cont.)

- Marking allocated memory: the *bit-flipper*

Example of flipping 6 bits



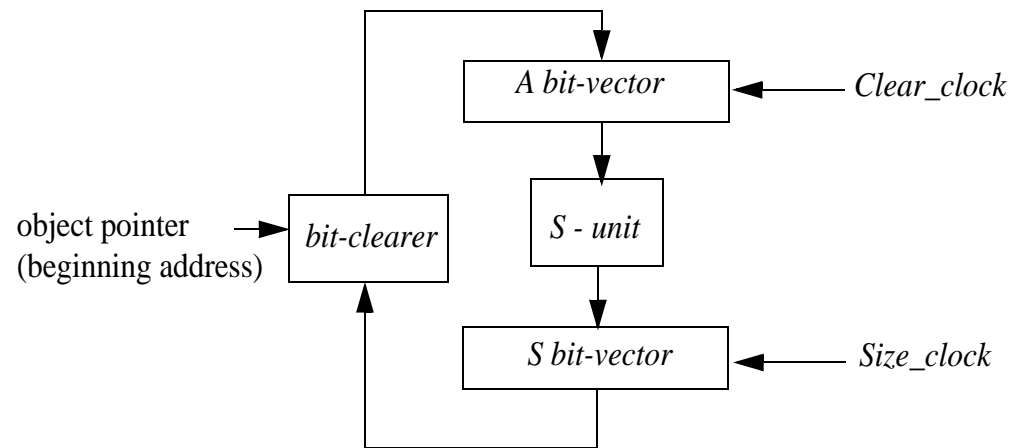
Previous work (cont.)

- In summer of 1998, Chang *et al.* proposed a hardware implementation of the *free* function.
 - + The size information is kept on the size bit-vector (*S-bit vector*).
 - + The size information is maintained by S bit-vector generation hardware circuit (*S unit*).
 - + It has three major components.
 - *S-bit vector* is used to store size information for each memory chunk.
 - *S unit* is used to update size information directly from the allocation bit vector (*A-bit vector*)
 - *Bit-clearer* is used to clear the bits in the *A bit-vector* which represents the memory blocks to be freed

Previous work (cont.)

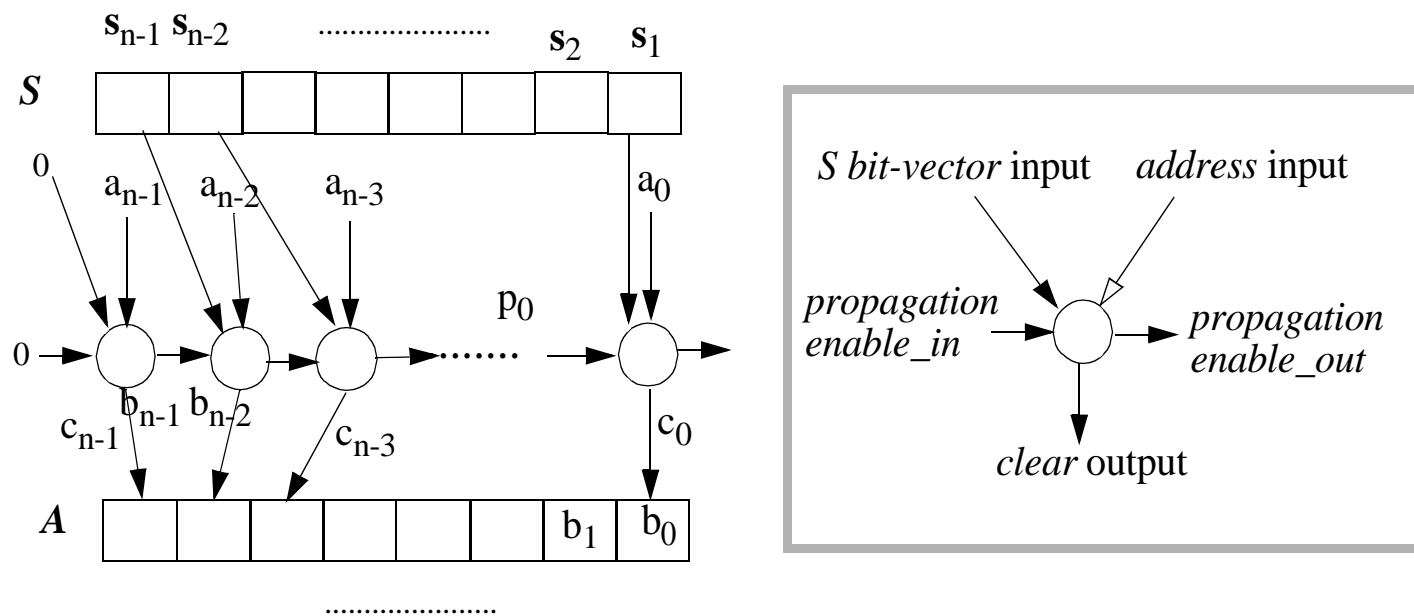
- In order to deallocate a memory chunk, two steps are needed
 - + Clear the bits in the *A bit-vector*: *bit-clearer*
 - + Adjust the *S bit-vector*: *S unit*

Overview of the deallocation system



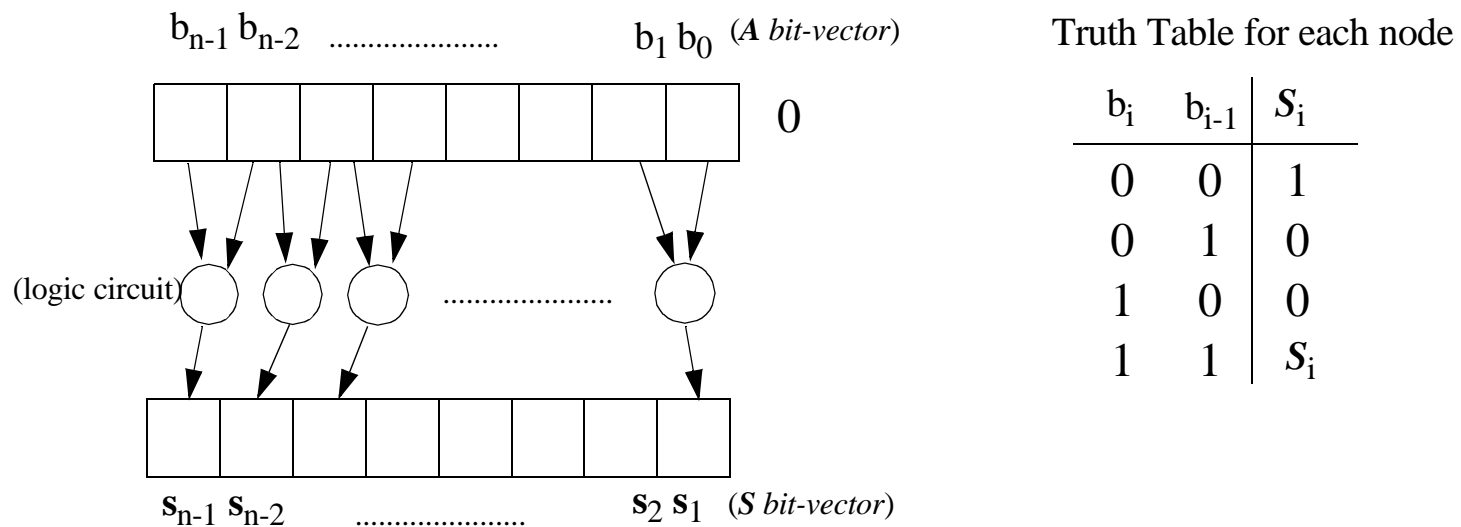
Previous work (cont.)

- Clear the bits in the A bit-vector: *bit-clearer*



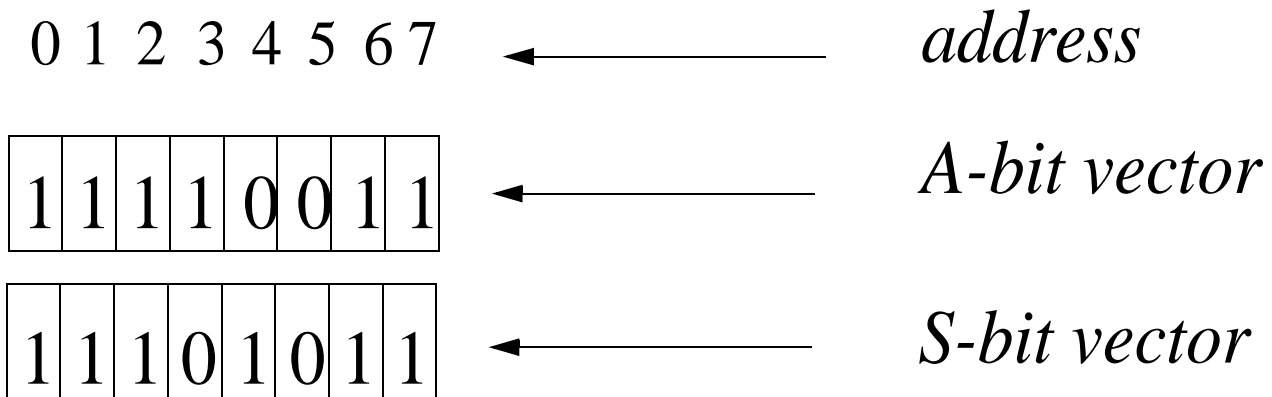
Previous work (cont.)

- Adjust the S bit-vector: S unit



Recording Information

- Basic configuration



Reallocation functions

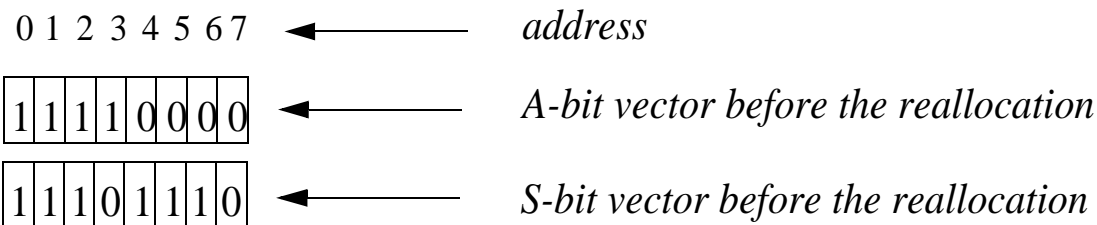
- Functions such as *realloc* or *expand* require two arguments, address pointer and size.
- With these two parameters, there can be three scenarios
 - + If the address pointer is NULL, realloc function as *malloc()*.
 - + If the size is NULL, realloc function as *free()*.
 - + If address pointer and size are not NULL then
 - if new size is smaller, shrink the block.
 - if new size is larger
 - > if block can be enlarged, inplace reallocation is possible.
 - > otherwise, new allocation is needed.

Differences between *realloc()* and *expand()*

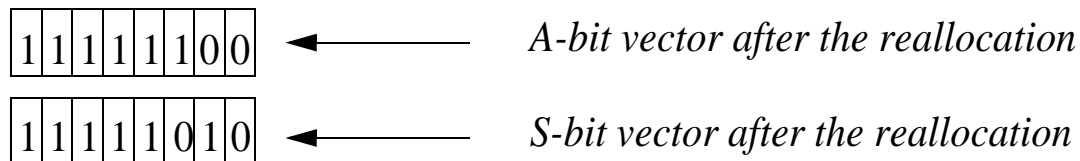
- *Expand()*
 - + if the block can be enlarged, *expand()* would enlarge the block and return the same address pointer.
 - + if the block can not be enlarged, *expand()* would return NULL.
- *Realloc()*
 - + if the block can be enlarged, *realloc* would enlarge the block and return the same address pointer.
 - + if the block can not be enlarged.
 - call *malloc* to allocate new block.
 - call *bit-wise* copy to copy the contents.
 - call *free* to deallocate old block.

Implementation of *expand* function

- In a nutshell



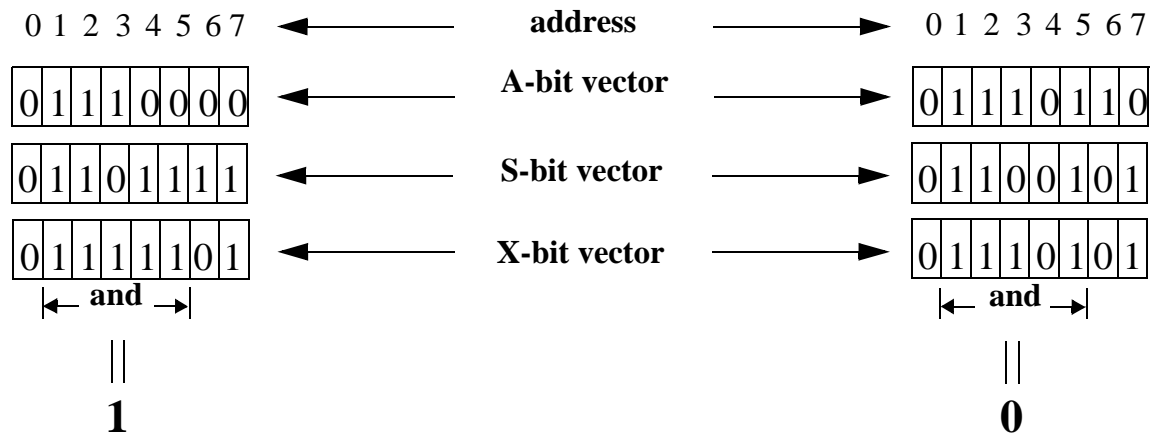
- (a) Yes, it is possible to reallocate two additional block of memory at address 0.
(b) Adjust *S-bit vector* to record block size information
(c) Adjust *A-bit vector* to record allocation status.



Is it possible?

- To determine whether inplace allocation is possible an auxiliary bit vector is introduced so that the projection of *S-bit vector* can be made.

expand(1, 6) / request new size of six blocks starting at address 1 */*



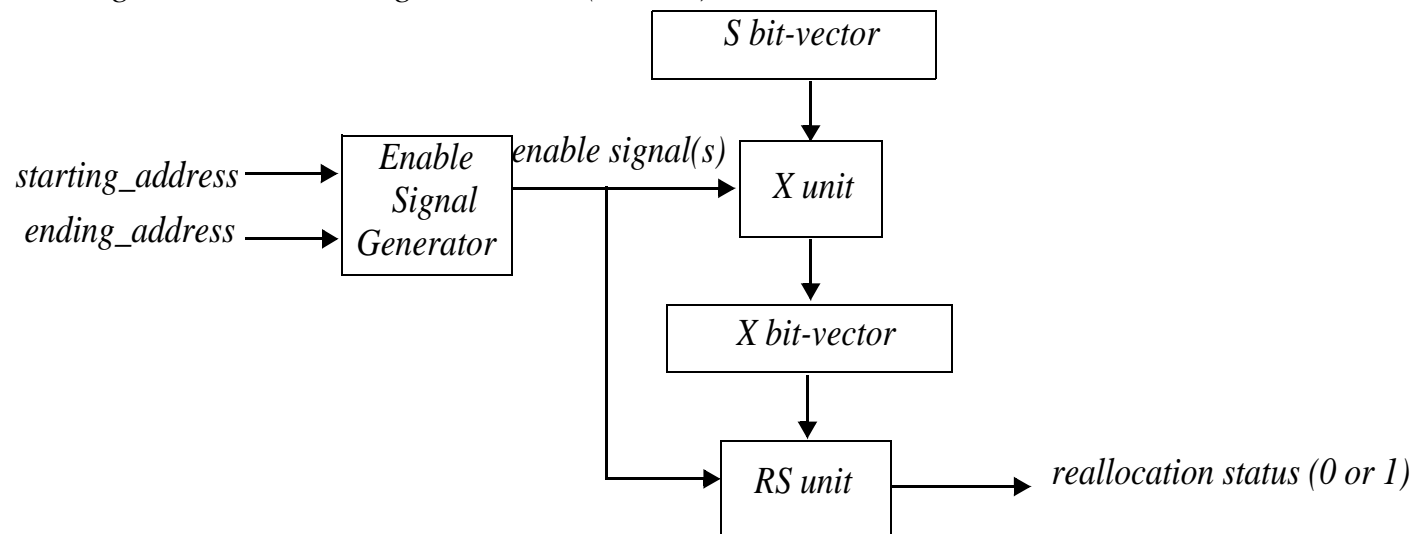
case a. reallocation is possible

case b. reallocation is not possible

Overview

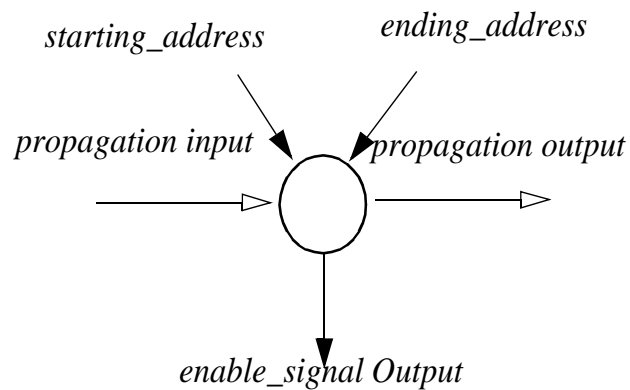
- The following diagram illustrates the components needed to determine inplace reallocation status.

$$\text{ending_address} = \text{starting_address} + (\text{size} - 1)$$



Enable Signal Generator Unit (*ESG unit*)

- is used to “highlighted” the targeted reallocation region of the *S-bit vector*.

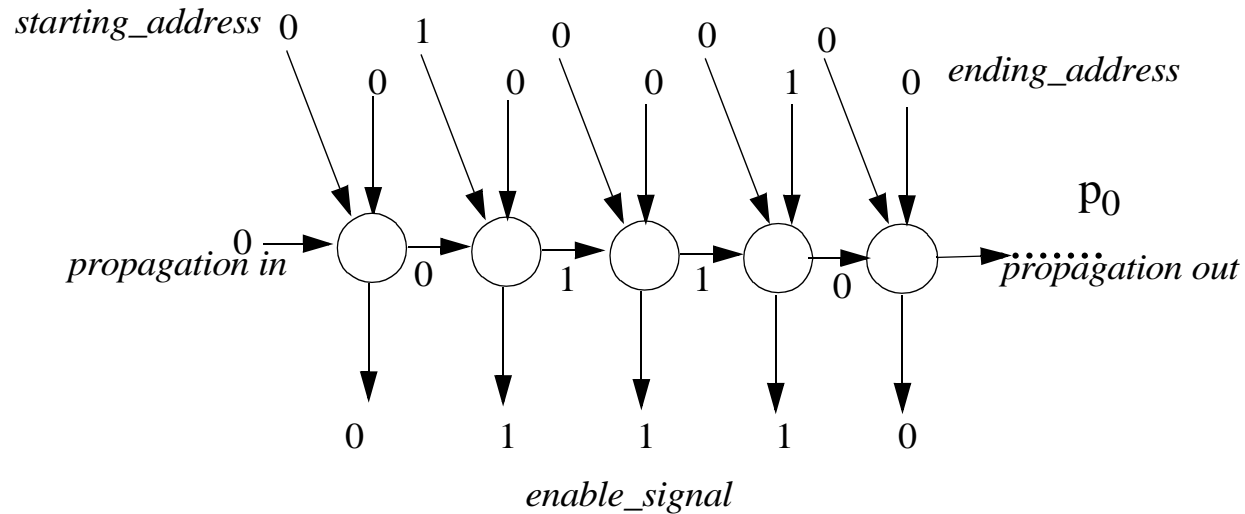


Enable_signal Generator Truth Table

<i>starting_</i> <i>address</i>	<i>propagation</i> <i>Input</i>	<i>ending_</i> <i>address</i>	<i>enable_</i> <i>signal</i>	<i>propa-</i> <i>gation</i>
0	0	0	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0

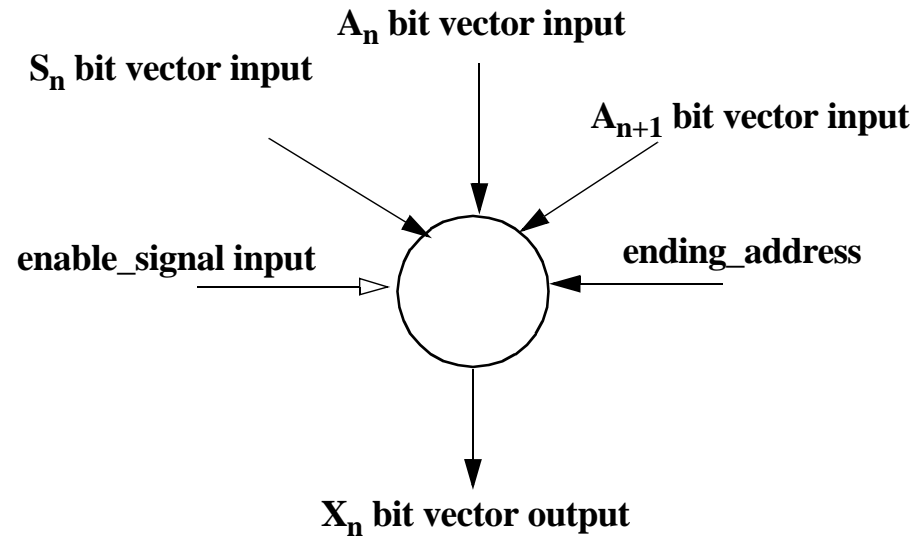
ESG unit (cont.)

- An example of *expand(1,3)*



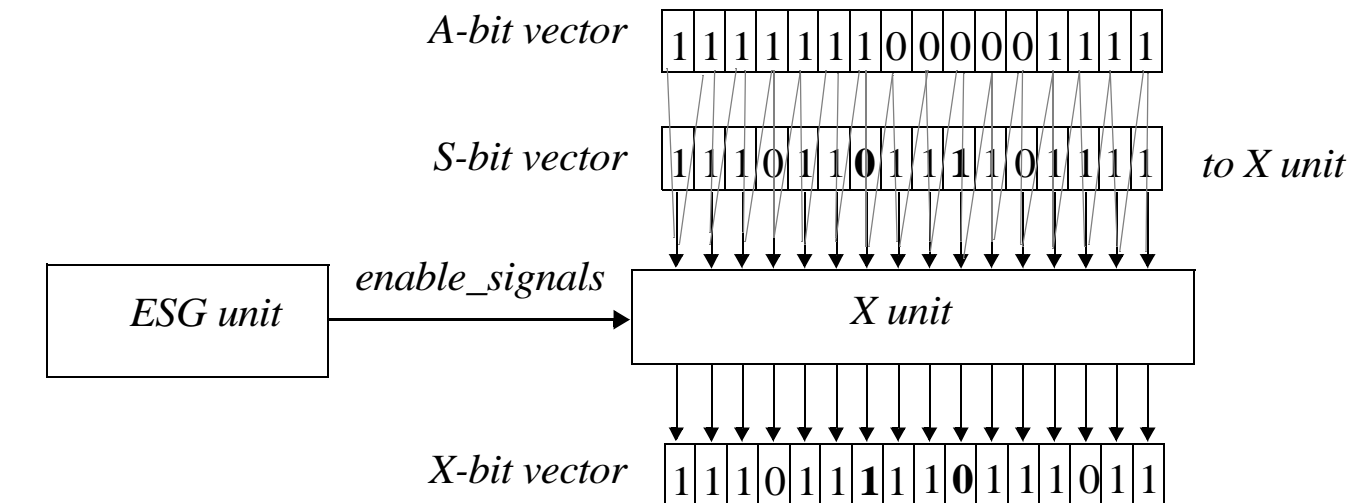
X-bit vector generation hardware (*X unit*)

- is used to project the future value of *S-bit vector* if the reallocation were to take place.



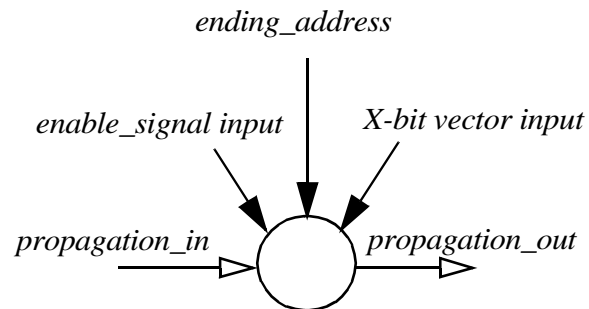
X unit (cont.)

- The contents of *S-bit vector* and *X-bit vector* for function call *expand(4,6)*.



Reallocation Status Unit (*RS unit*)

- is used to identify whether the inplace reallocation is possible or not.



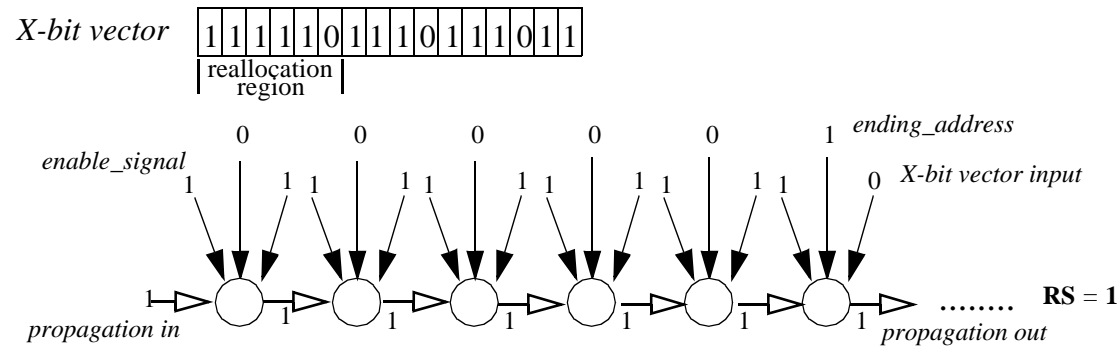
Reallocation Status truth Table

<i>ending_address</i>	<i>Propa- gation_in</i>	<i>enable_ signal</i>	<i>X bit vector input</i>	<i>Propa- gation_out</i>
0	0	0	X	0
0	1	0	X	1
0	1	1	1	1
1	1	1	0	1

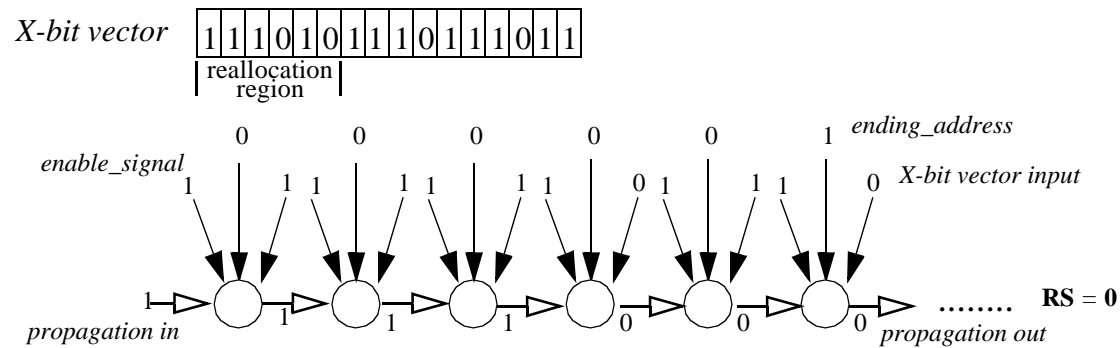
Note: initial value of propagation input is 1

RS Unit (cont.)

An example of the RS unit with reallocation status (RS) = 1

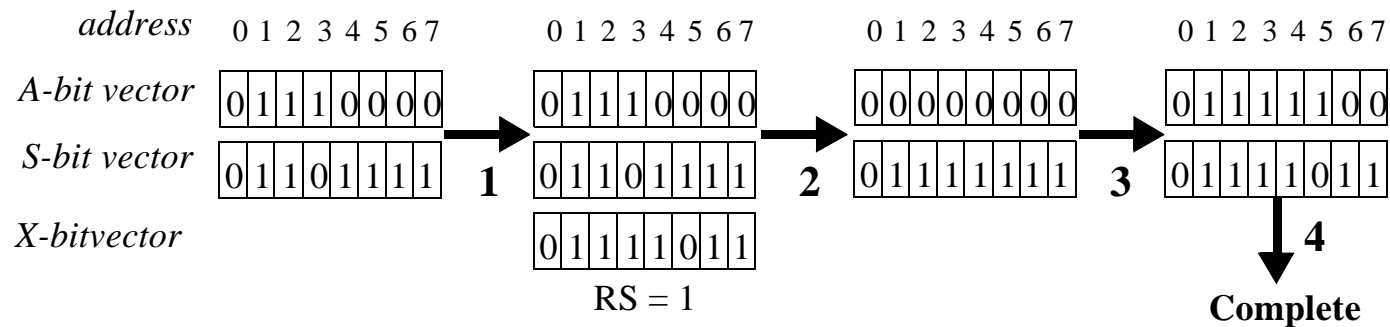


An example of the RS unit with reallocation status (RS) = 0



Completing reallocation process (cont.)

- Detailed description of inplace reallocation process.



- Step 1.* Determine the reallocation status. If $RS = 1$, go to step 2. Else go to step 4.
- Step 2.* Free the memory chunk pointed to by address pointer (1 in this case) input
- Step 3.* Allocate memory chunk pointed to by address pointer 1 in this case) until the given size (5 in this case) is reached.
- Step 4.* Complete the process return address pointer if succeeds or NULL if fails.

Conclusion

- Hardware implementation of *expand()* allows the inplace reallocation to be done in constant time.
- Hardware complexities of *ESG unit*, *X unit*, and *RS unit* are $O(n)$.
- X-bit vector can also be used for other purpose such as marking phase for mark and sweep garbage collection.