A Hardware Implementation of a High-Performance Memory Allocator

Khushwinder Jasrotia

Electrical and Computer Engineering

University of Toronto

Oct 26, 2001

ksj@eecg.toronto.edu

Outline

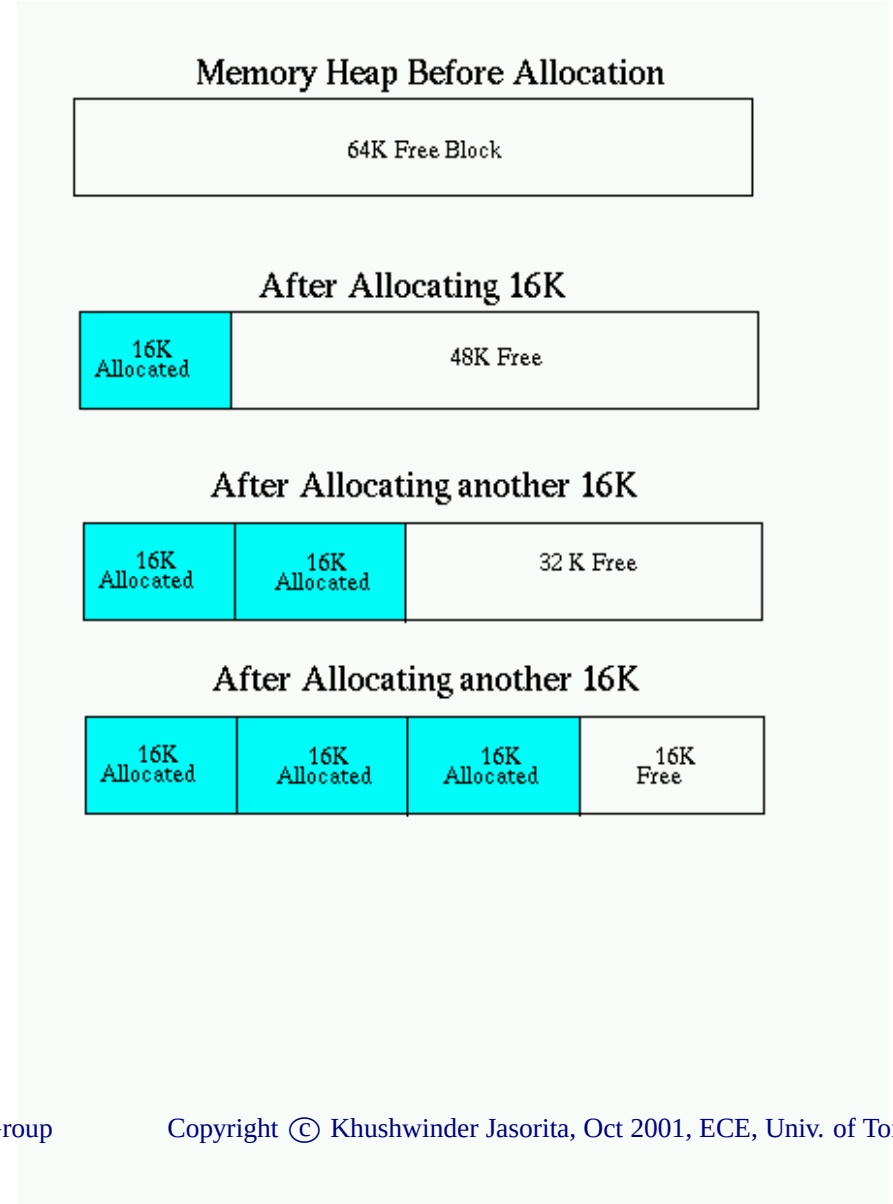
- Motivation
- Description of Software Memory Allocators
- Description of Hardware Memory Allocator
- Results
- Conclusion

What is memory allocation ?

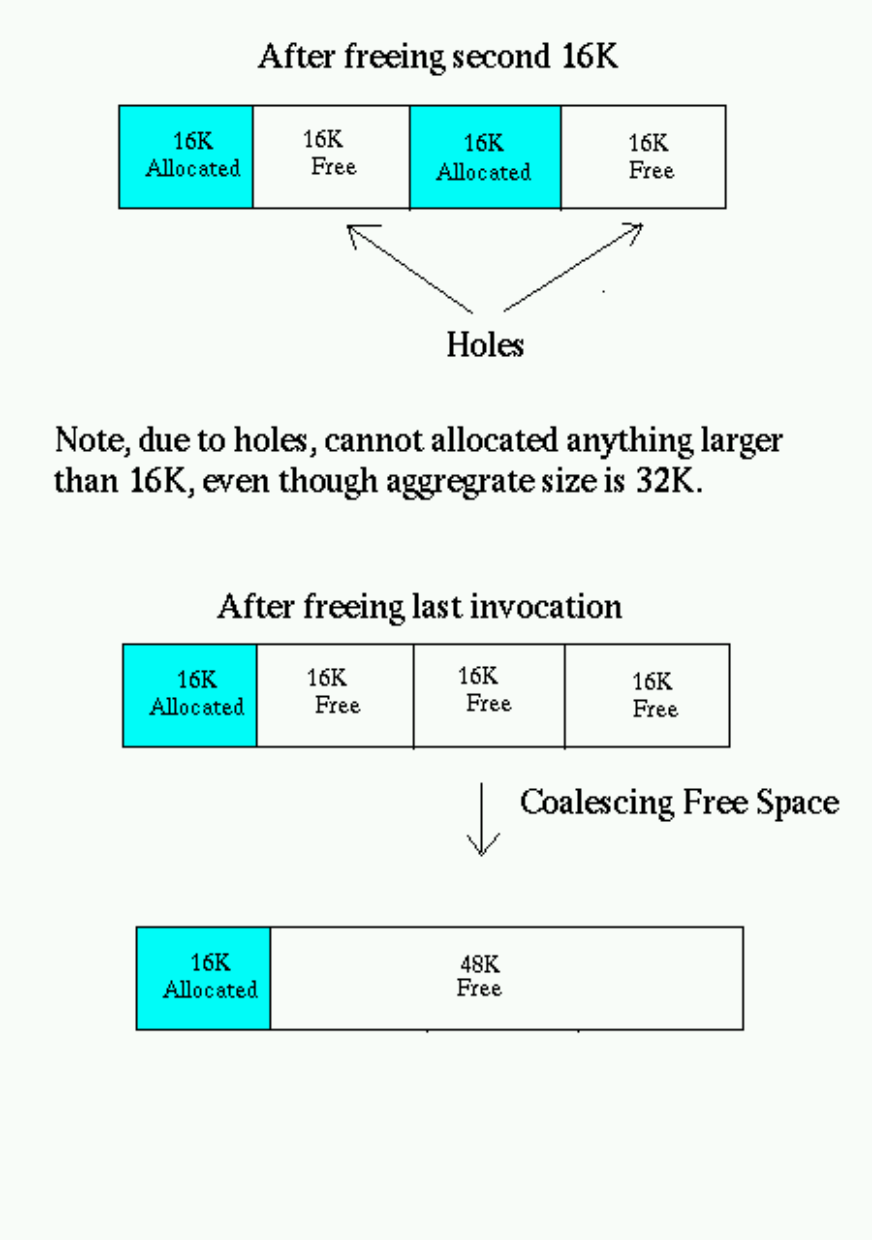
- Memory allocation is the process of assigning blocks of memory on request.
- Examples of this in C are *malloc()* and *free()*
- A memory allocator keeps track of which part of memory in use, and which parts are free
- It must make any returned blocks available for reuse.
- Goal of allocator design is to minimize wasted space without undue time costs.
- Why *dynamic* memory allocation? Size of data structures do not have to be statically defined.

Example of allocating and freeing memory

- Memory is assigned from a free heap area
- Can only deal with memory that is free, and only choose where in free memory to allocate the next requested block.



Example of allocating and freeing memory - Continued



Problems with memory allocation: Fragmentation

- Application programs may free blocks in any order, creating ”holes” amid live objects.
- If these holes are too numerous and small, thay cannot be used to satisfy future requests for a larger blocks.
- This problem is known as *fragmentation*.
- No reliable algorithm for ensuring efficient memory usage, and *none is possible*.

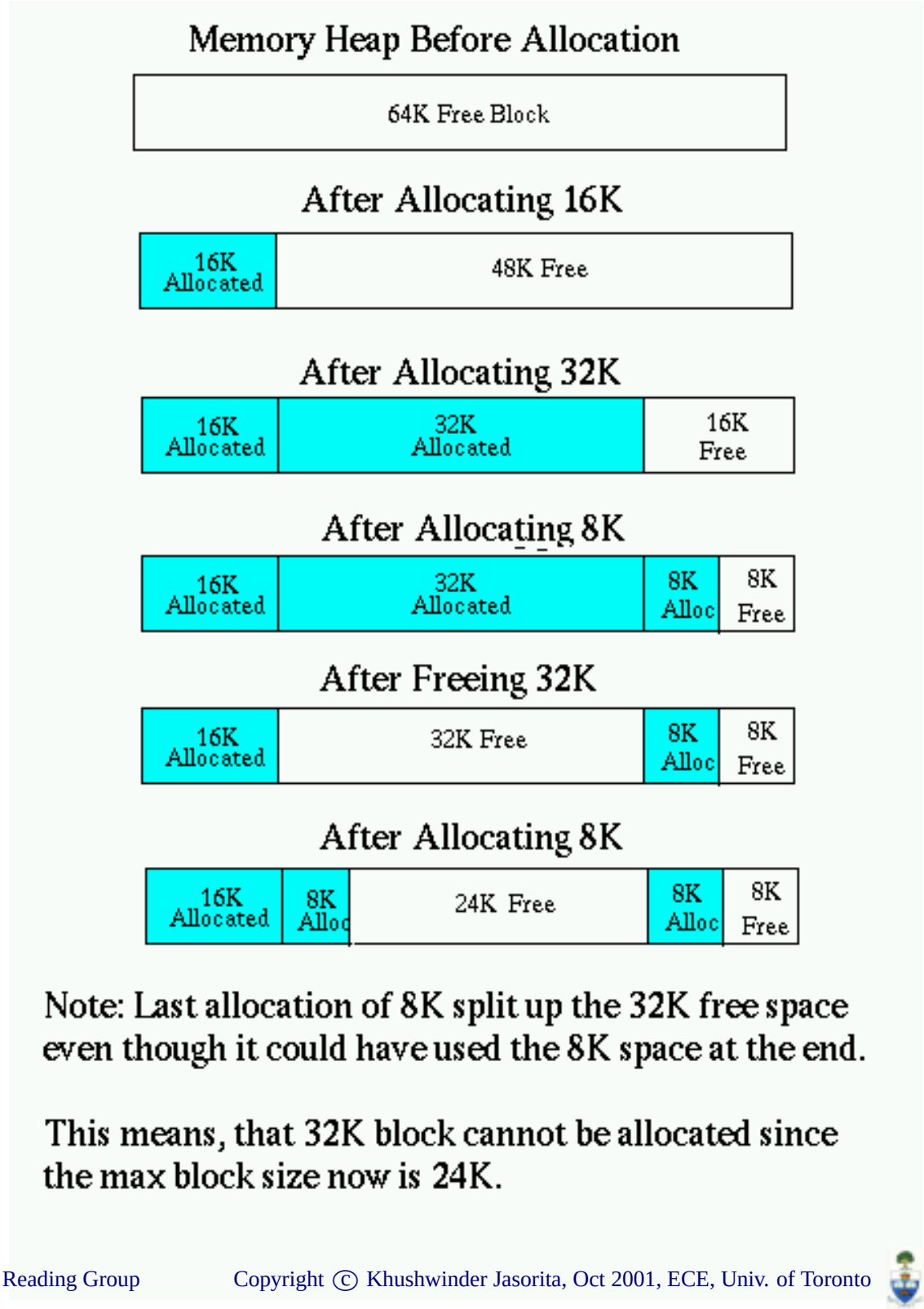
Different Software memory allocation mechanisms

- Different memory allocator mechanisms are used in software .
- Each mechanism has its pros and cons with regards to efficient memory usage and excecution speed.
- There are three popular mechanisms:
  1. Sequential First Fit
  2. Segregated Free Lists
  3. Buddy System

Sequential First Fit

- In this algorithm, the allocator keeps a list of free blocks(known as the free list).
- On receiving a request for memory, is scans along the list for the first block that is large enough to satisfy the request.
- If chosen block is larger than requested, it is usually split, and the remainder is addded to the list as another free block.
- If a block that is freed is next to an already free block, the two blocks are coaleseced to into one big free block.

Sequential First Fit



Note: Last allocation of 8K split up the 32K free space even though it could have used the 8K space at the end.

This means, that 32K block cannot be allocated since the max block size now is 24K.

Advantages and Disadvantages of First Fit

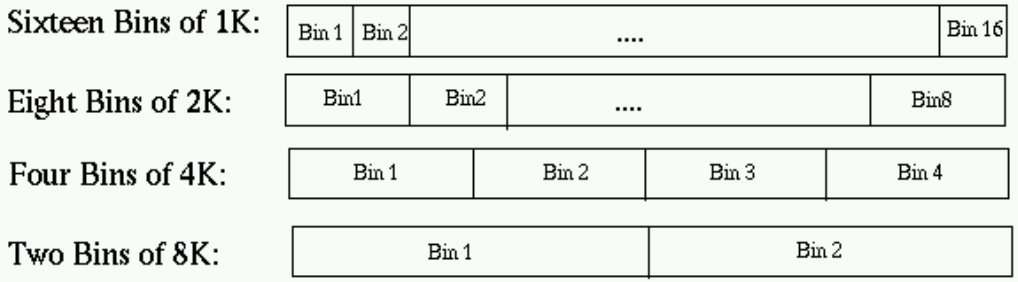
- Advantages: Exhibits good memory usage.
- Disadvantages: Increased search time due to fragmentation in the begnning of heap. The search for free block must go past them each time a larger block is requested.

Segregated free Lists

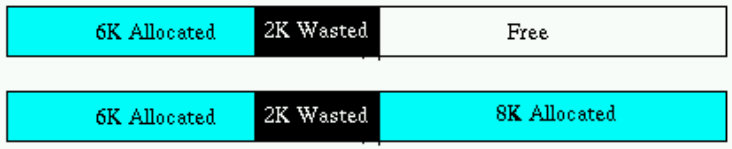
- Uses an array of free lists, where each list holds free blocks of a particular size.
- When a block of memory is freed, it is simply pushed onto the free list for that size.
- When a request is serviced, the free list for the appropriate size is used to satisfy the request.
- Requested are usually rounded up to powers of 2.

Segregated free Lists

Original memory heap of 64K split up into four bins as follows:



Malloc 6K: Note, 6K gets rounded up to closest bin which is 8K. It gets stored in the 8K bin.



The wasted space is known as internal fragmentation.
Cannot allocated another 8K block since no more bins, even though enough memory exists

Advantages and Disadvantages of Segregated free Lists

- Advantages:
  - Segregated storage is quite fast in the usual case.
- Disadvantages:
  - Wasted space since requests are rounded up to powers of 2.This is called internal fragmentation.
  - Severe external fragmentation, since none of memory allocated to one size block can be resued for another.

Buddy System

- In buddy system, the allocator will only allocated blocks of certain sizes.
- The permitted sizes are usually either powers of two, such that any block except the smallest can be divided into two smaller blocks of permitted sizes.

Buddy System

- When allocator receives a request for memory, it rounds the requested size up to a permitted size, and returns the first block from that size's free list.
- If free list for that size is empty, the allocator splits a block from a larger size and returns one of the pieces, adding the other to the appropriated free list.

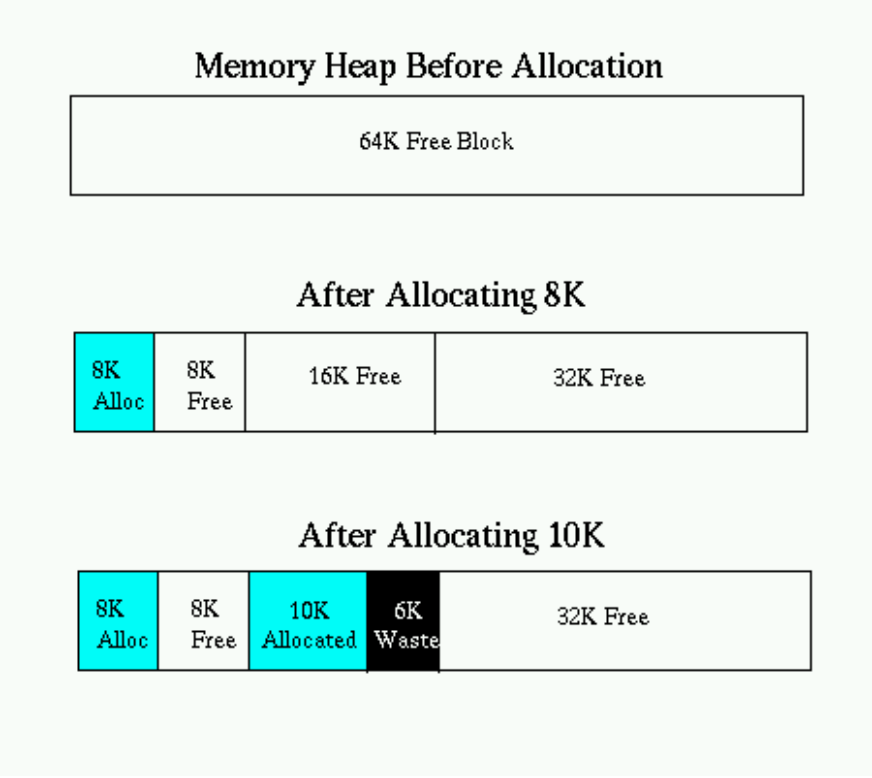


Figure 2: Example of Sequential Fits

Advantages and Disadvantages of the Buddy System

- Advantages: Coalescing is relatively fast because the ”buddy” of any free block can be calculated from its address.
- Disadvantages: Rounding typically leads to a significant amount of wasted memory.

Hardware memory allocator

- Hardware allocator as described in *A High-Performance Memory Allocator for Object-Oriented Systems* by Chang and F. Gehringer.
- Why hardware:
  - Density of logic devices increasing - becoming more attractive to map basic software algorithms into hardware.
  - Research reports dynamic storage management consumes 23-38 percent time in six allocation-intensive C programs.
  - Speed of storage management more crucial to system performance.

Hardware memory allocator advantages - Continued

- If storage allocated in contigues blocks, can hold allocation in a bit-vector. Easy to implement in hardware.
- Bit-mapped approach provides opportunity to implement hardware version of the buddy system.
- Operation of splitting and coalescing memory dominates cost of buddy system in software implementation.
- Hardware-maintained bit-map approach eliminates need for splitting and coaslescing operations.
- Allocation done using haredware-maintained binary tree that finds free blocks using combinational logic.

Hardware memory allocator advantages - Continued

- Binary buddies assign blocks in powers of 2 - wasted space due to internal fragmentation.
- Bit-vector allocation uses unused portion of the block.
- Eg. if buddy systems allocates eight blocks for a 5-block request, hardware can mark exactly five of the eight bits.

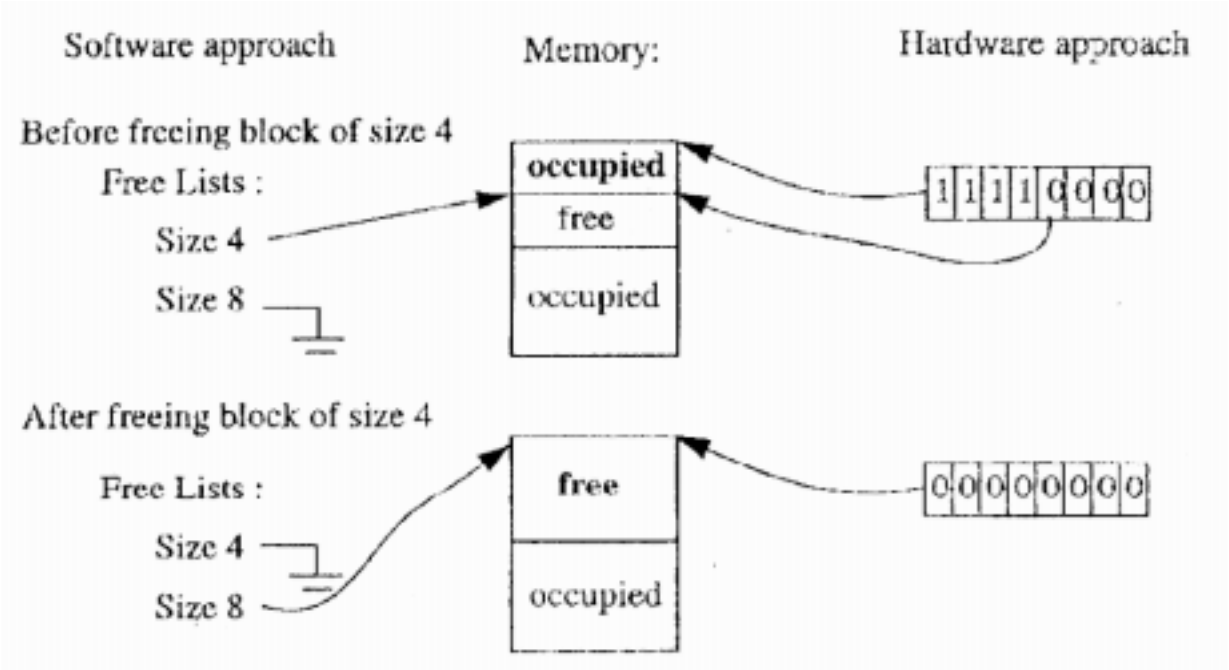


Fig. 1. An example of coalescing in software and hardware approach.

Memory Allocator Based on the Buddy System

- Hardware allocator must do three things:
  1. Determine if there is large enough block for allocation.
  2. If so, find beginning address of that memory chunk
  3. flip the bits corresponding to the memory chunk in the bit-vector.

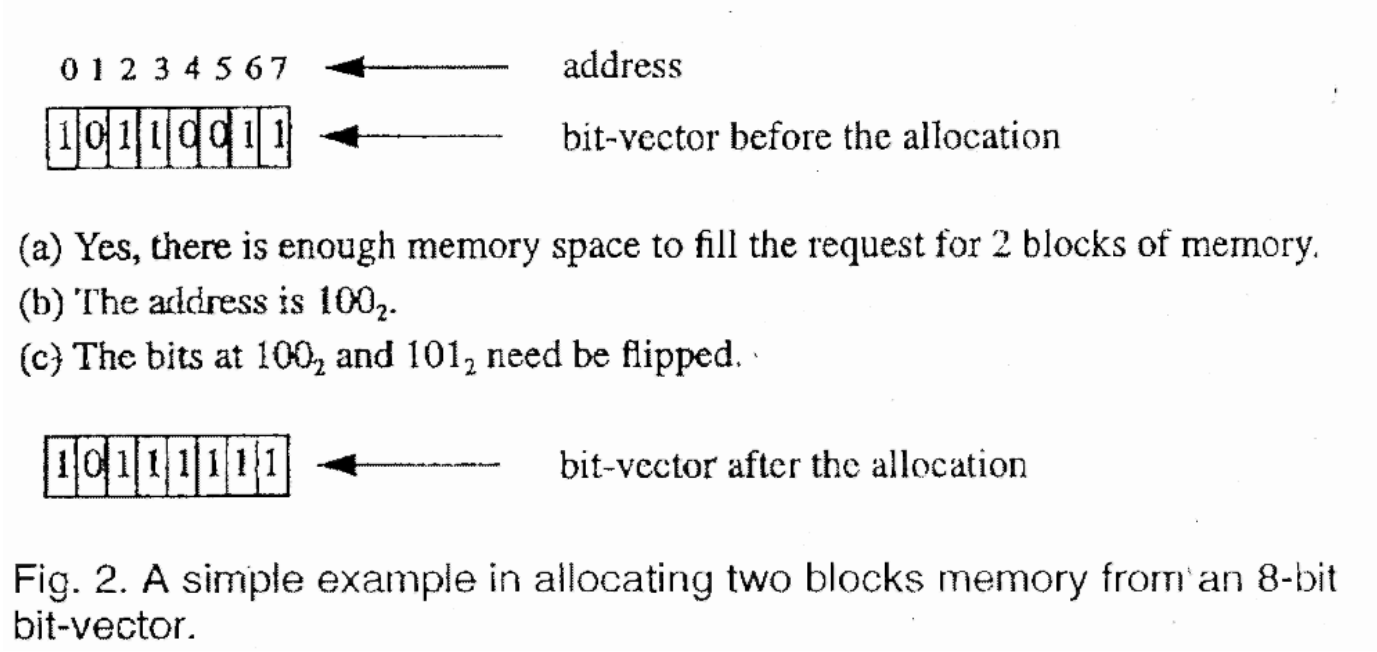


Fig. 2. A simple example in allocating two blocks memory from an 8-bit bit-vector.

Memory Allocator Based on the Buddy System Cont..

- Use an or-gate tree to perform function 1.
- Use an and-gate tree to perform function 2.
- Use a bit-flipper circuitry to perform function 3.

Locating Free Blocks: The Or-Gate Tree

- A bit-vector contains bits that represent the status of the memory blocks. (1 for used, 0 for free).
- Build a complete binary tree (CBT) ”on top” of a bit vector.
- Leaves of the tree are the bits in the bit-vector.
- Each CBT node is simply a two-input OR gate.
- The output of the OR gate shows the allocation status of the subtree

Locating Free Blocks: The Or-gate tree - Continued

- For memory with 2^N blocks, this requires $2^N - 1$ OR gates.
- An availibilty value of zero at level *l* means a memory chunk of size 2^{N-l} is available.

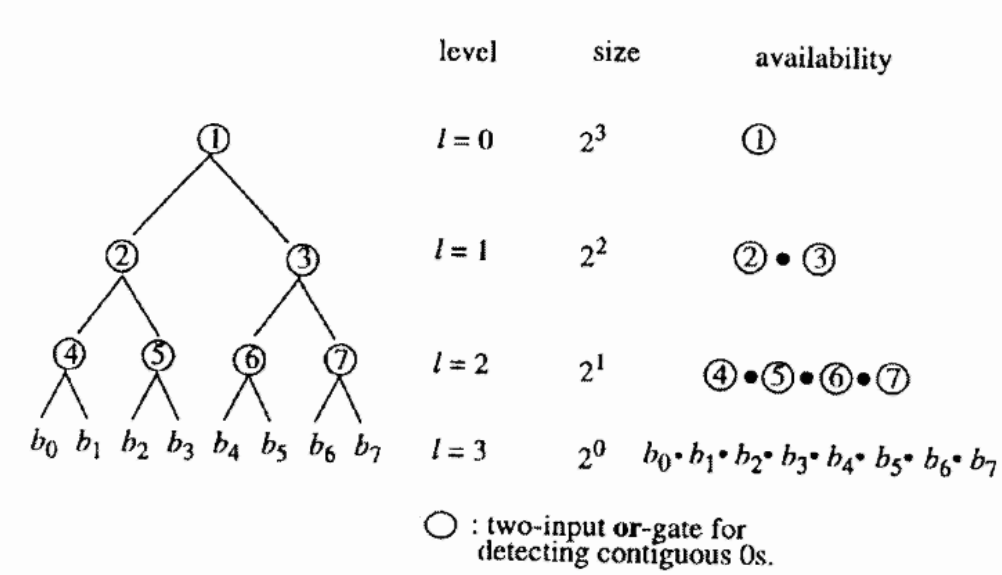
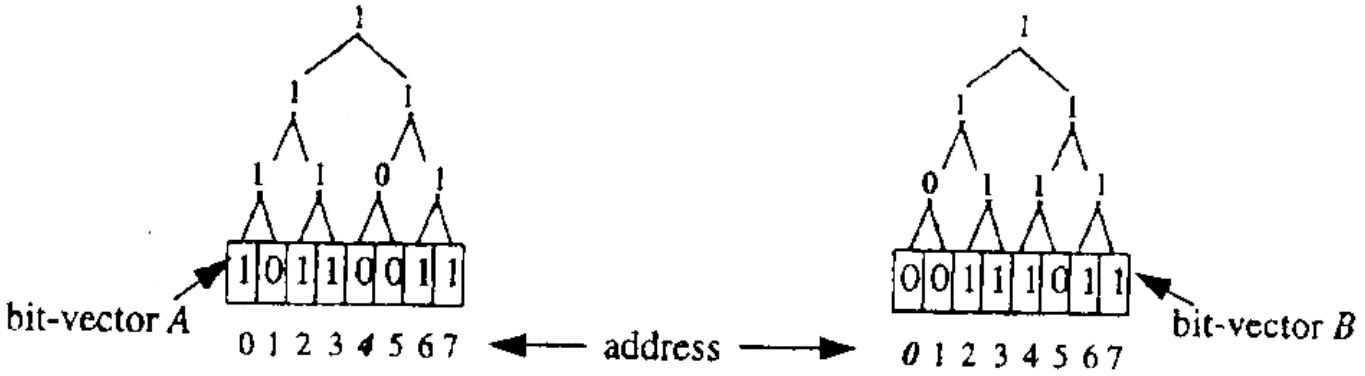


Fig. 3. Building a complete binary tree (CBT) for an 8-bit bit-vector.

Example of address locator tree



For the two-block request, the address of free memory space:

100 (for bit-vector *A*)

000 (for bit-vector *B*)

Fig. 4. Example of locating an address from the or-gate tree in two different bit-vectors.

Finding the First Zero and Its address: The And-Gate tree

- This tree is constructed from And gates.
- It is used to determine the address of the first zero (at level *l* of the or-gate tree) to be *n*.
- The address of the free memory block in the bit-vector is *n* X 2^{N-l}

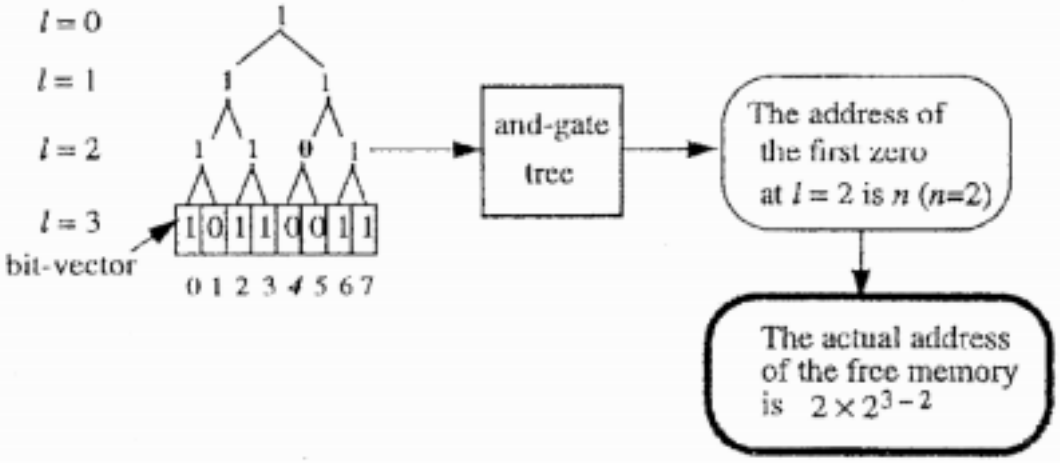


Fig. 5. Example of allocating two-block memory from an 8-bit ($N = 3$) bit-vector.

Nonbacktracking Binary Search Algorithm

- Iterative search could be peformed in a given level, however this requires backtracking.
- Each node has a one-bit value called the *propogation bit* (or, the P-bit).
- The P-Bit is the otput of an AND gate at each node.
- To avoid backtracking, the P-bit is augmented with a new *address bit* (or A bit).

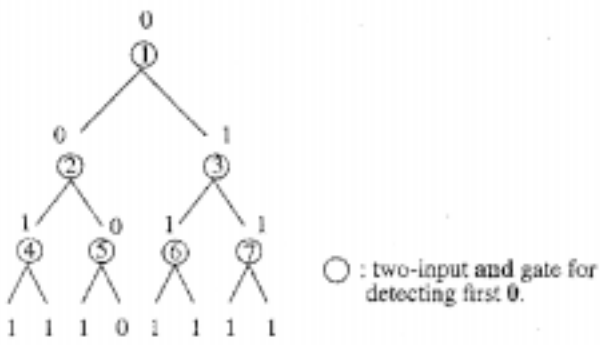


Fig. 6. Searching for the first "0" in an "and" binary tree built from an 8-bit bit vector.

Nonbacktracking Binary Search Algorithm

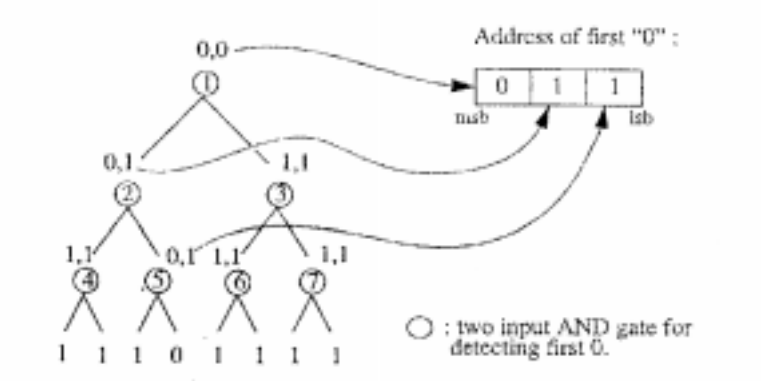


Fig. 7. Searching for the first "0" in an "and" binary tree built on an 8-bit bit vector.

- The A-bit is a copy of the P-bit of the left child node.
- It serves to direct the search to a particualr child node.
- Search is always directed to the subnode that has enough free space, making backtracking unnecessary.
- A-bits encountered on a traversal from root towards leaves makes up the binary address.

Connecting AND an OR Trees

- Connection scheme between or-gate and AND-gate is specified below:

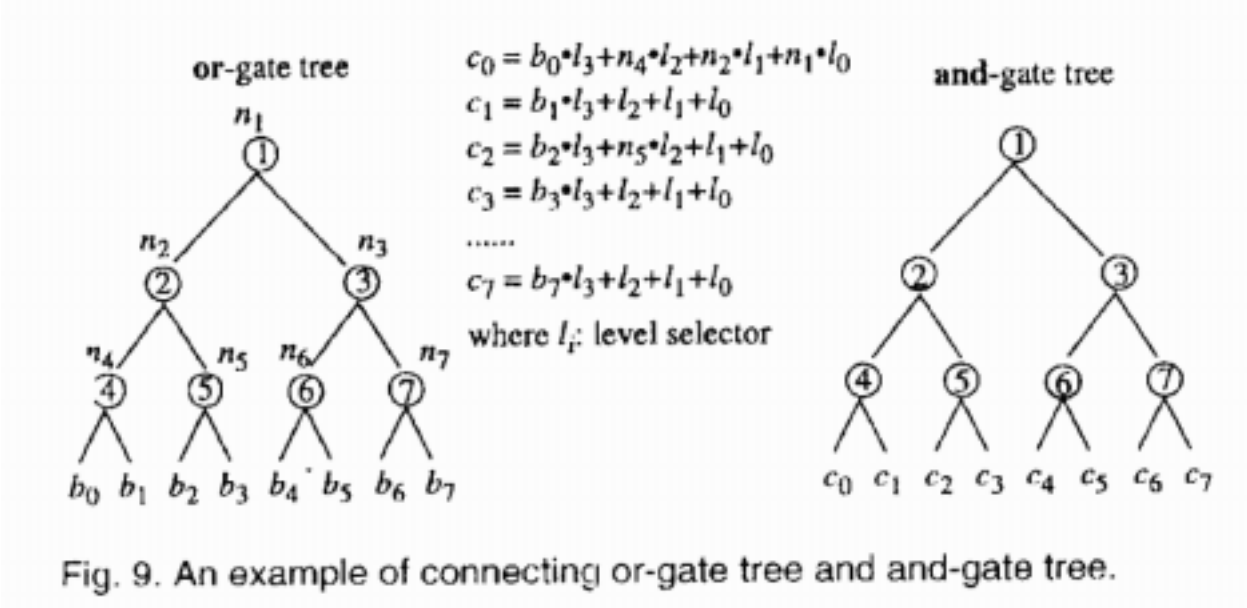


Fig. 9. An example of connecting or-gate tree and and-gate tree.

- Only one of the *li*(level selectors) will be asserted at any one time.
- This is the *li* corresponding to the size of memory requested.

Marking Allocated Memory: The Bit-Flipper

- Bit is bit-vector need to be marked as occupied or un-occupied.
- The bit flipper inverts bits, from 0 to 1 for allocations, or from 1 to 0 for deallocation.
- Bit-flipper propogates signals from root of the tree to the leaves.

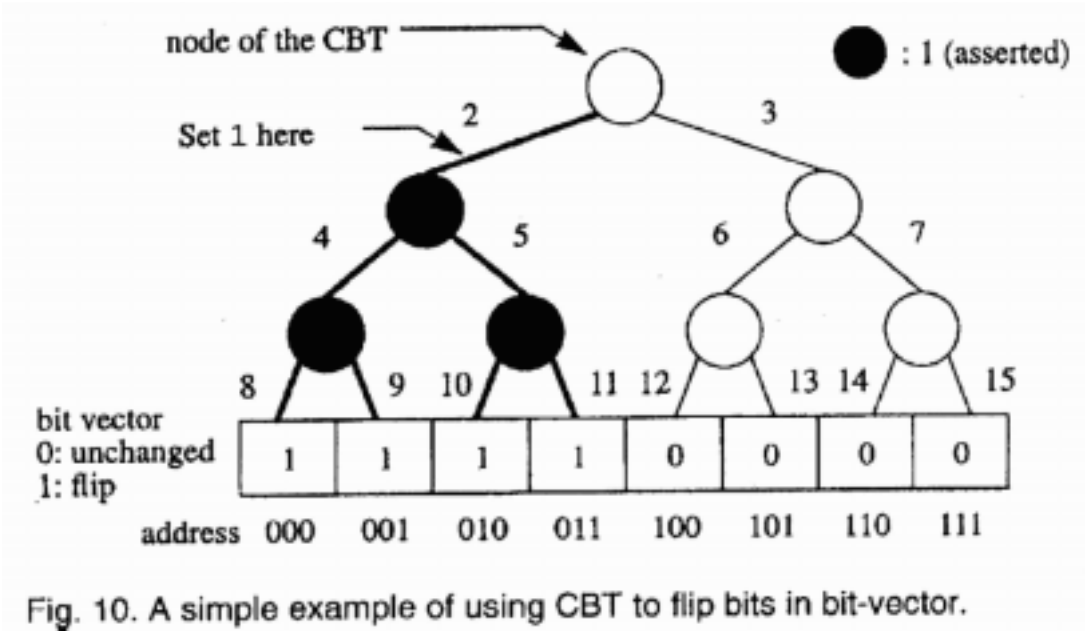


Fig. 10. A simple example of using CBT to flip bits in bit-vector.

Marking Allocated Memory: The Bit-Flipper - Continued

- Must determine which subtree we want to alloated from
- This depends on the staring address and the number of bits that need to be flipped.
- Use external logic or build logic into nodes to let it determine how to propogate bit-flipping.
- Building intelligent node that can propogate information has advantage of limiting hardware complexity to O(n).
- Provides good regularity which is key to VLSI design.

Propagating Allocation information down a Tree

- Need at least two signals to link between a node and its children.
- The *flip* signal, if asserted, indicates that all allocation bits in the subtree rooted at the node are to be set to 1.
- If *flip* is not asserted, the *route* signal controls which, if any, of the allocation bits in the subtree are to be set.

Propagating Allocation information down a Tree - Continued

- Also need two control signals to each *level* of the tree.
- The *size* bit controls the total number of bits to be flipped.
- The *address control* bit gives the starting address of the bits to be flipped.

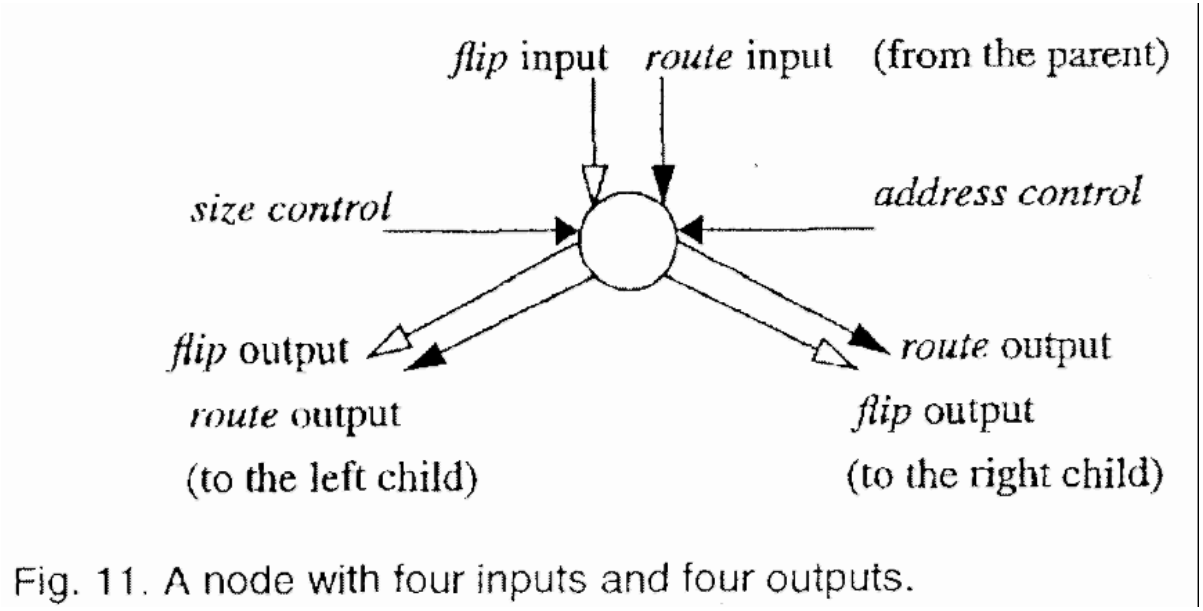


Fig. 11. A node with four inputs and four outputs.

Propagating Allocation information down a Tree - Continued

Table 1: Truth Table for the Node With Four Inputs and Four Outputs

flip input	route input	size control	address control	flip output (left)	route output (left)	flip output (right)	route output (right)
1	X	X	X	1	X	1	X
0	0	X	X	0	0	0	0
0	1	0	0	0	1	0	0
0	1	0	1	0	0	0	1
0	1	1	0	1	X	0	0
0	1	1	1	0	0	1	X

Tree for Propogating Allocation information down a Tree - Continued

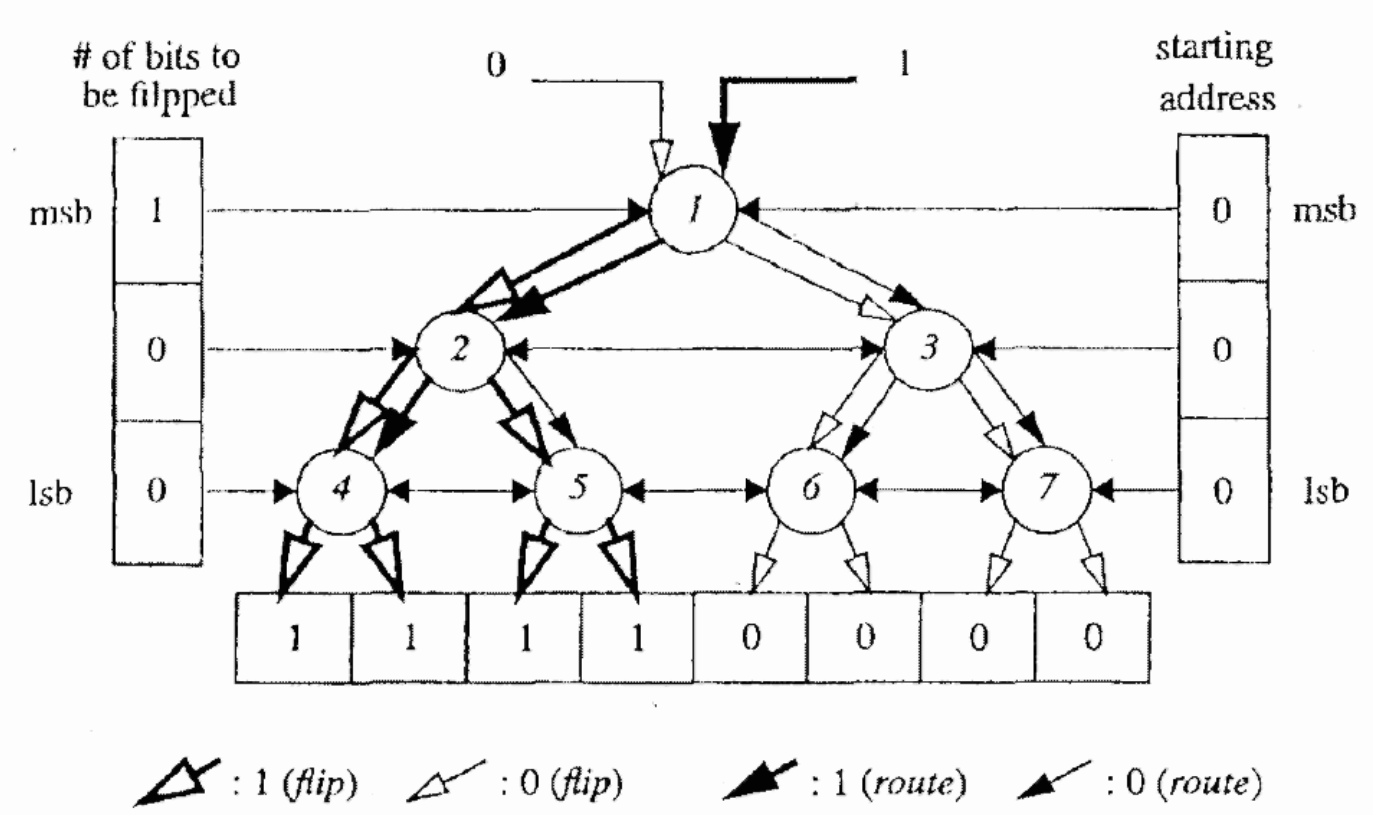


Fig. 12. Flipping four bits in an 8-bit bit-vector.

Marking Arbitrary Number of Bits

- Try flipping 101_2 consective bits staring at address 000_2.
- According to truth table, only four, not five bits will be flipped.
- Decompose 101_2 into $100_2 + 001_2$
- To accomposh this, as we march down the subtree for the first component, 100_2, must *activate* its right counterpart.

Table 2: Truth Table for the Node With Four Inputs and Four Outputs

flip input	route input	size control	address control	flip output (left)	route output (left)	flip output (right)	route output (right)
1	X	X	X	1	X	1	X
0	0	X	X	0	0	0	0
0	1	0	0	0	1	0	0
0	1	0	1	0	0	0	1
0	1	1	0	1	X	0	*1*
0	1	1	1	0	0	1	X

Tree for Propogating Allocation information down a Tree for Arbitrary Bits

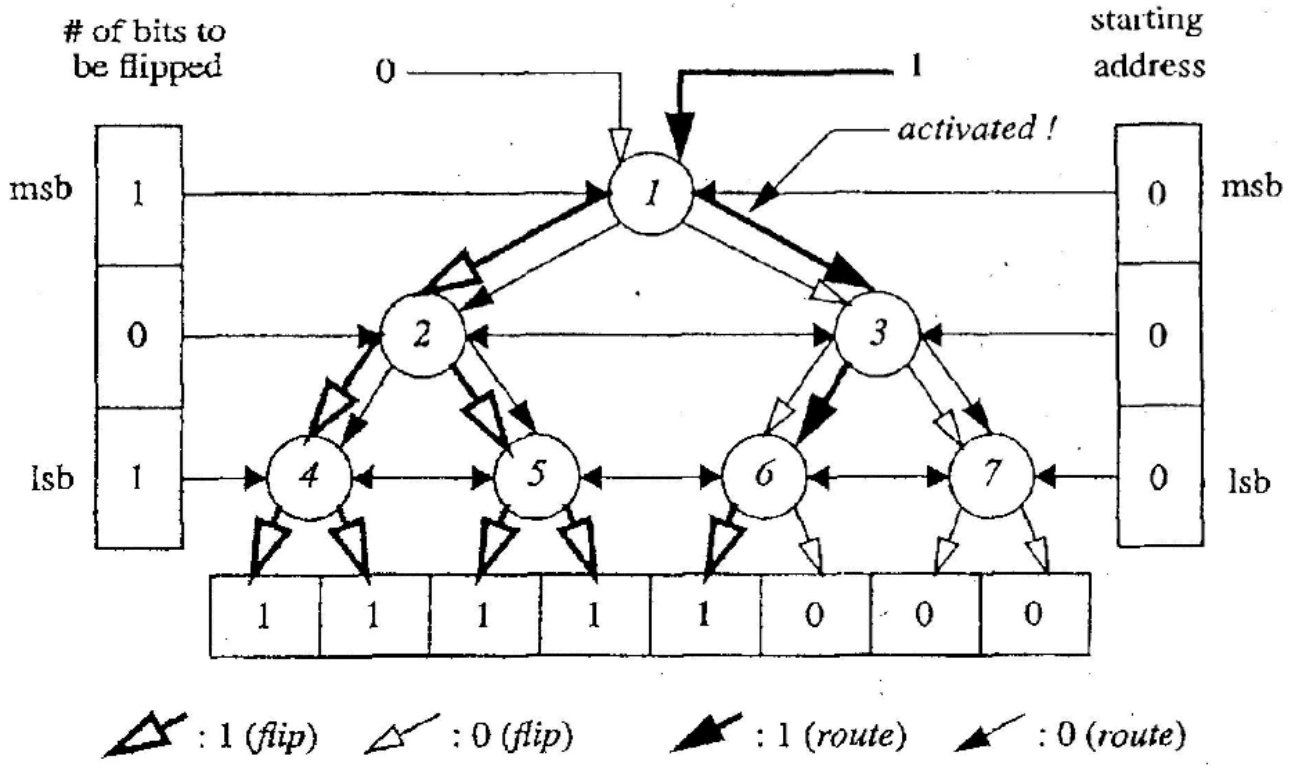


Fig. 13. An example of flipping n ($n = 5$) bits.

Evaluation Methodology

- Evalute scheme on traces of memory allocations an deallocations from actual C progrsm.
- Use programs from different applications *gawk; perl; espresso*
- Instrumented the programs by modifying *malloc* and *free* to generate traces.
- Fed traces into analysis program which emulates the functionalitys of the proposed system.
- Analysis program provides two measurements: *memory space allocated* and *hightest address* allocated.
- hightest address allocated = memory space allocated + fragmentation

Results

- Modified buddy system uses an average of 19.7 percent less (maximum) memory space.
- Highest address ever allocated decreases by an average of 7.7 percent.

Conclusion

- This paper has presented the deisgn and hardware realization of a new non-backingtracking algorithm for address generation.
- This saves considerable time over software aproach.
- Also describes a ”bit-flipper” for turning unused portions of a block for availiable storage.