Binary Access Memory:

An Optimized Lookup Table for Successive Approximation Applications

Benjamin Hershberg*, Skyler Weaver*, Seiji Takeuchi†, Koichi Hamashita†, Un-Ku Moon*

*School of Electrical Engineering and Computer Science, Oregon State University

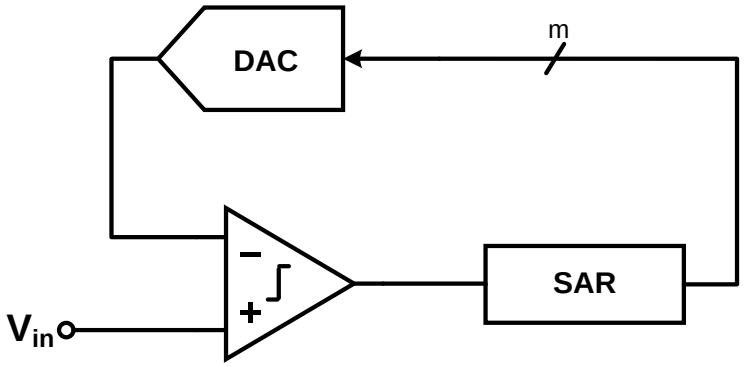
†Asahi Kasei EMD Corporation, Atsugi, Japan

Presentation Overview

- Introduction & Motivation
- Binary Access Memory (BAM)
  - Basic idea of BAM
  - Global pre-fetching
  - Local pre-charging
  - Asynchronous BAM
- Conclusion

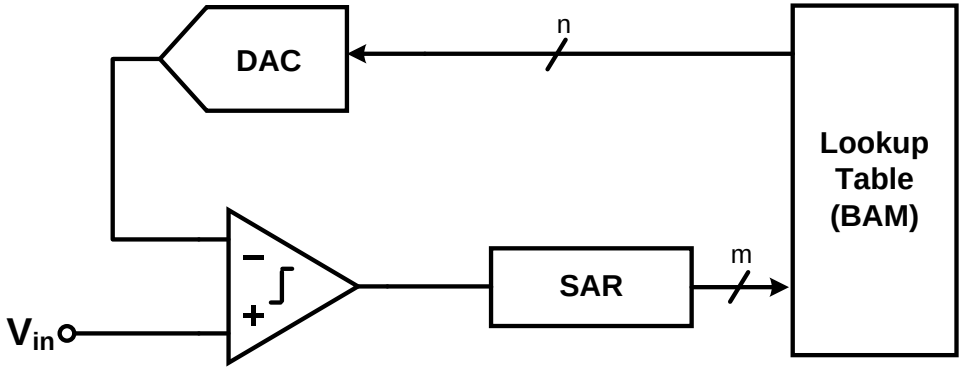
Introduction & Motivation

Typical SAR Error Correction



- Popular SAR error correction methods
  - Radix calibration
  - Trimming
  - Lookup table (LUT)
    - Outside the loop
    - Inside the loop

Generalized SAR Error Correction



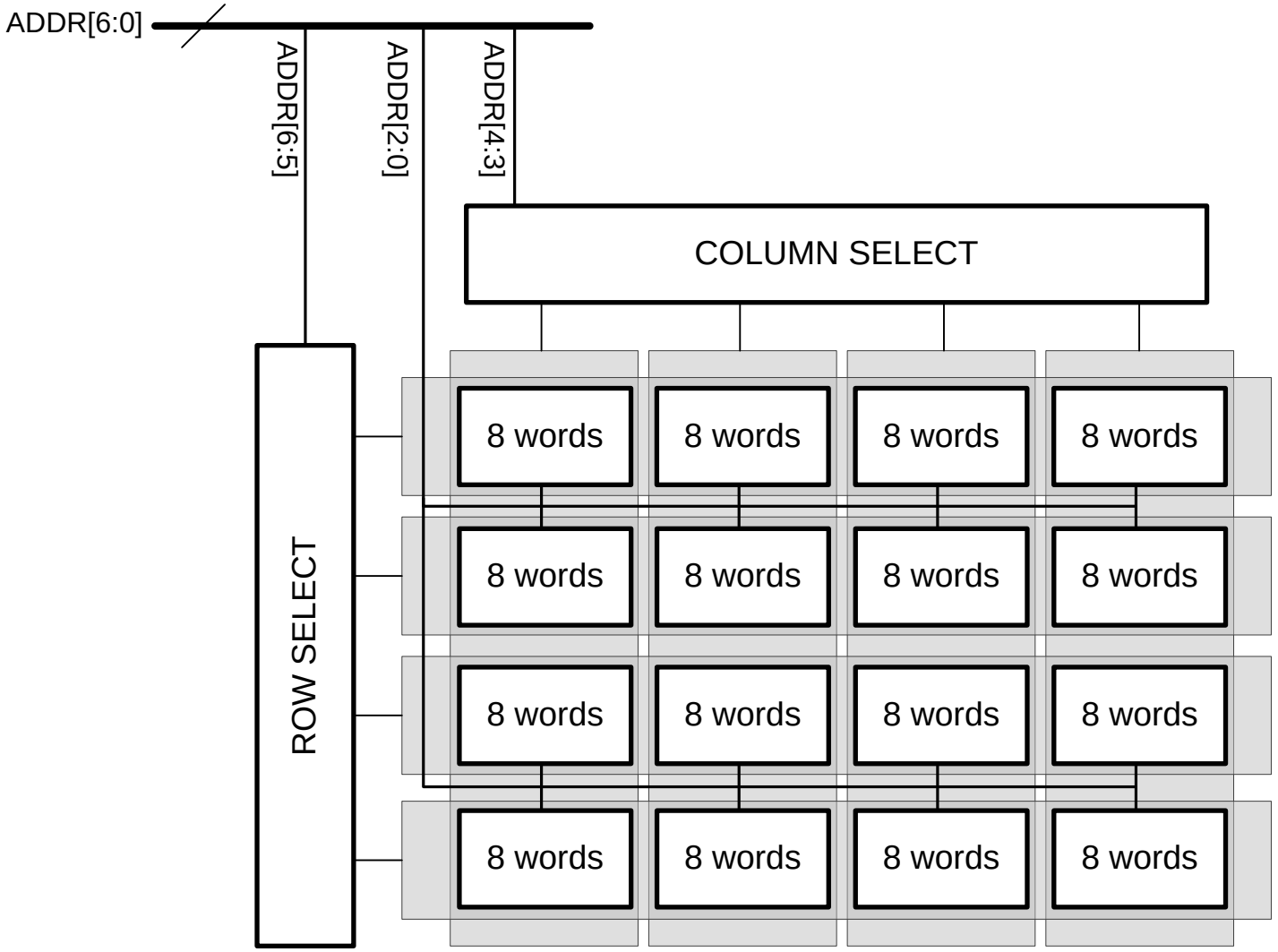
- Remaps each SAR code to *some* DAC code
- Payoff: enables new ways of implementing binary search
- Drawback: power, latency

Lookup Table Implementation

- RAM – Random Access Memory
  - But, binary search is *not* a random access pattern!
- BAM – Binary Access Memory
  - Exploit probabilistic aspects of binary search to reduce the latency and power requirements of the lookup table...

BAM memory organization

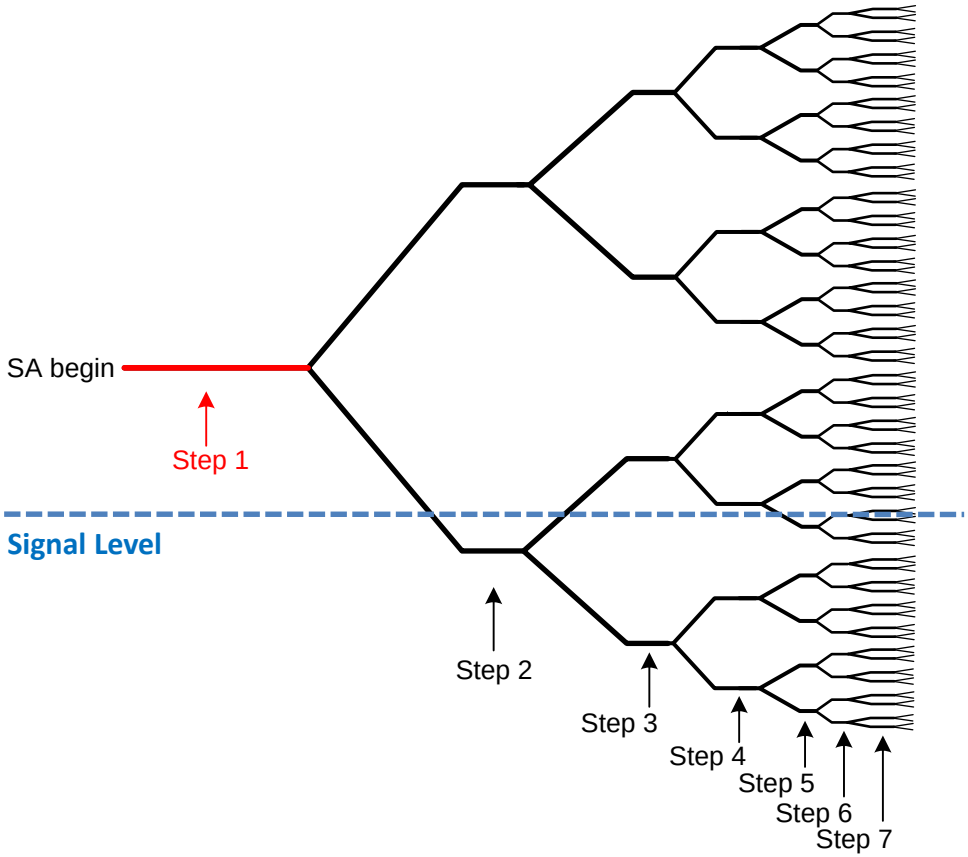
SRAM


ADDR[6:0]
ADDR[6:5]
ADDR[2:0]
ADDR[4:3]
COLUMN SELECT
ROW SELECT
8 words
8 words
8 words
8 words
8 words
8 words
8 words
8 words
8 words
8 words
8 words
8 words
8 words
8 words
8 words
8 words

Useful Properties of Binary Search

Property 1

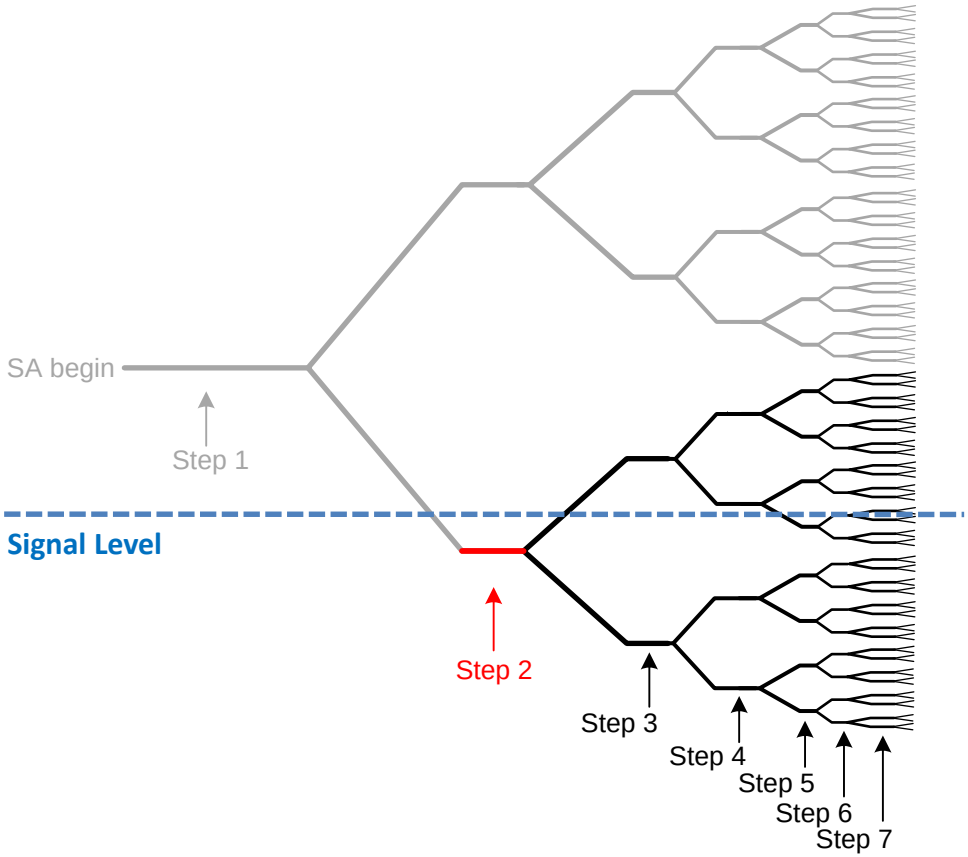
- A binary search is a one-way journey down the search tree



Useful Properties of Binary Search

Property 1

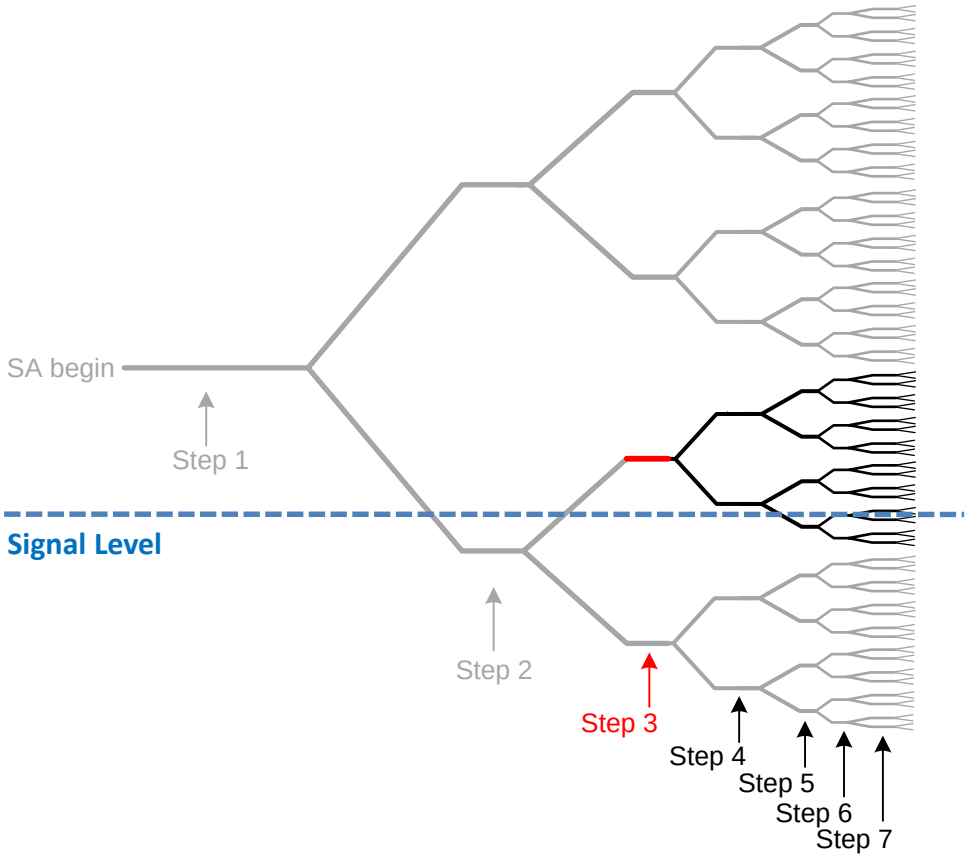
- A binary search is a one-way journey down the search tree



Useful Properties of Binary Search

Property 1

- A binary search is a one-way journey down the search tree



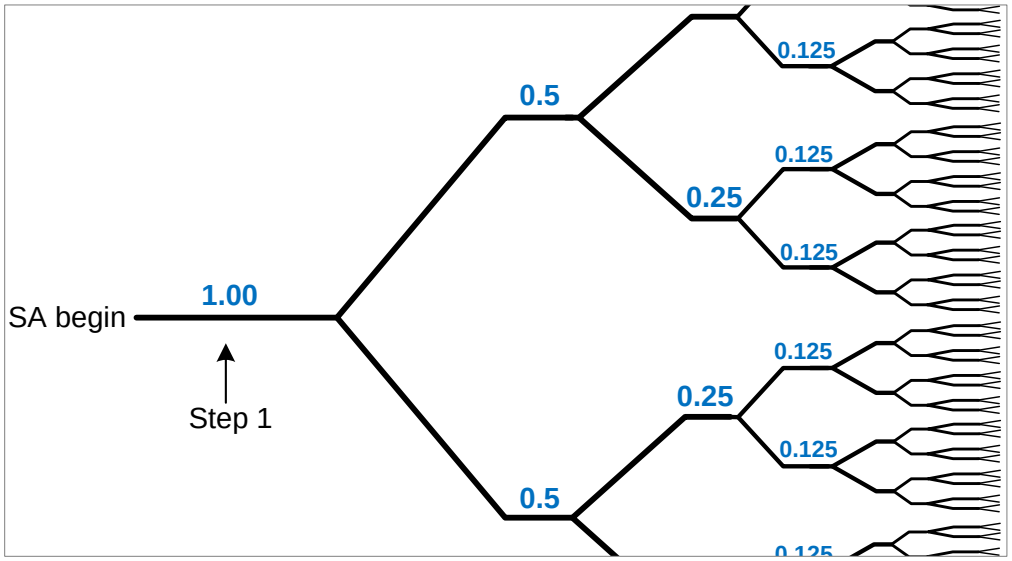
How can we improve the organization of data words in the memory?

- SRAM: organizes data according to similarity in address code
- BAM: organizes data according to similarity in location within the search tree

Useful Properties of Binary Search

Property 2

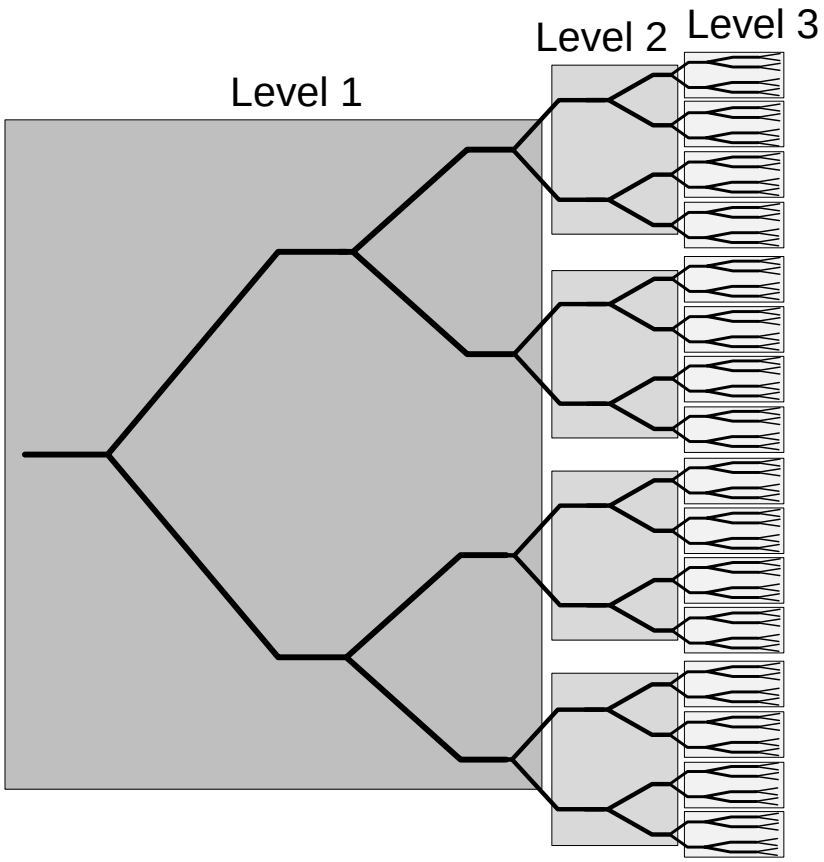
- The probability that a node will be visited during a binary search is a non-uniform distribution.



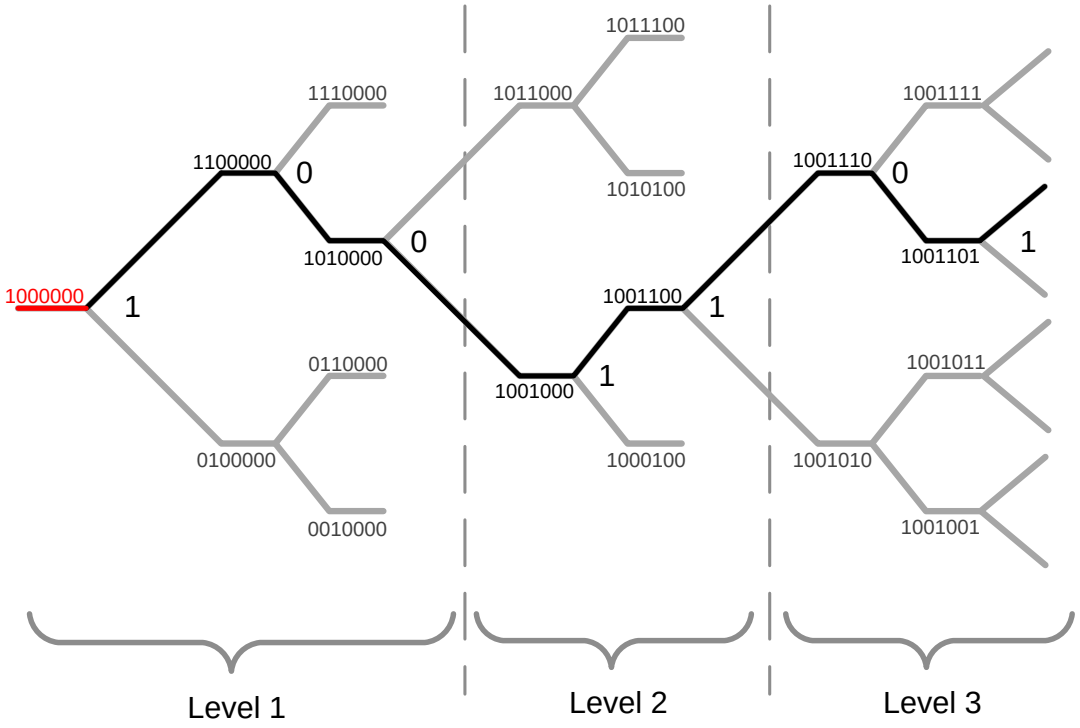
How can we improve the organization of data words in the memory?

- Make the nodes with the highest probability of being visited the “easiest” to access.
- Minimize average energy/bit and average latency.

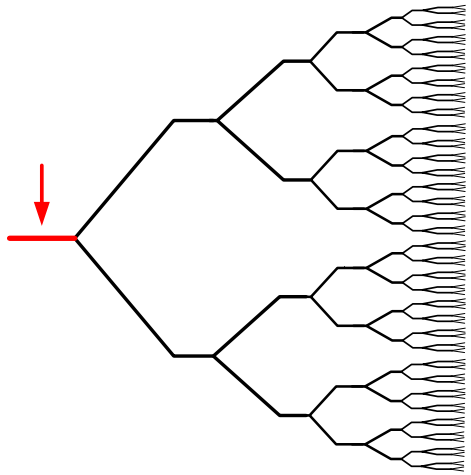
BAM memory organization



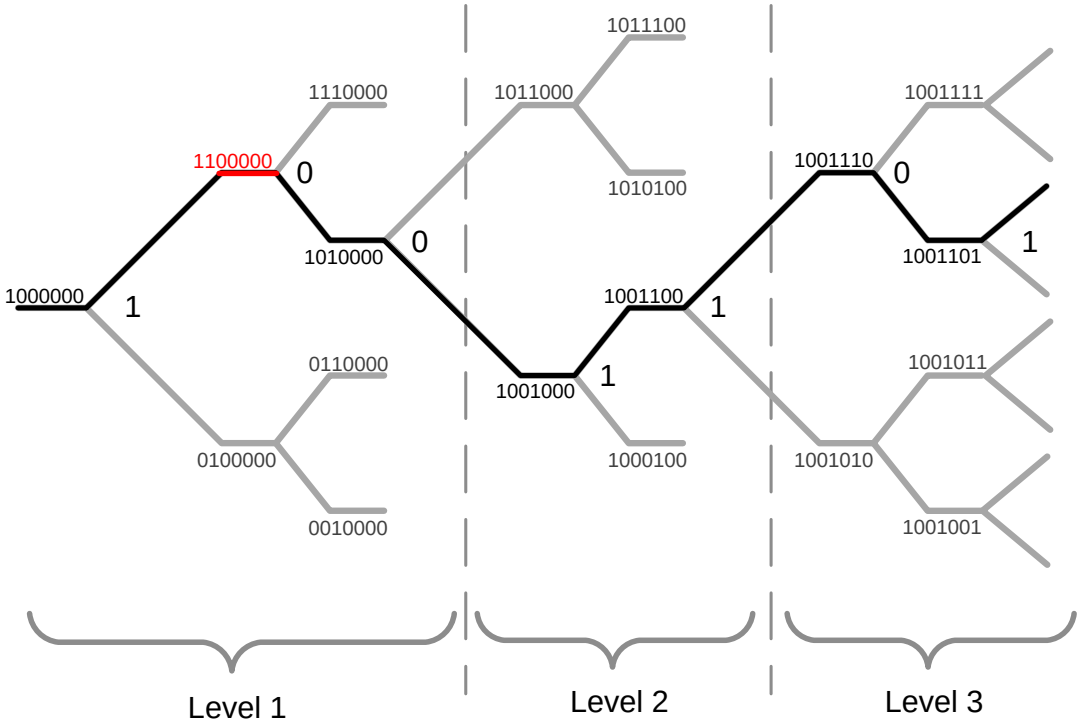
Basic Operation – Step 1



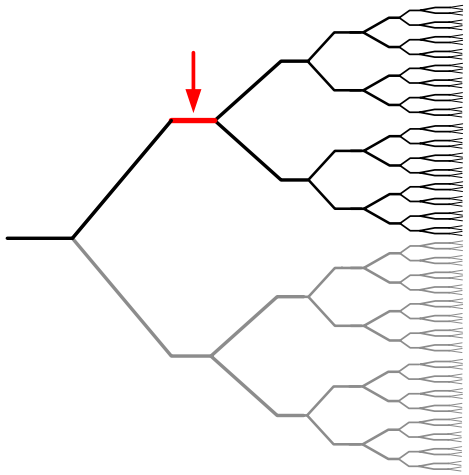
Step	Memory Address							
1	**1**	0	0	0	0	0	0	Level 1
2								
3								
4								Level 2
5								
6								Level 3
7								
	Level 1			Level 2		Level 3		



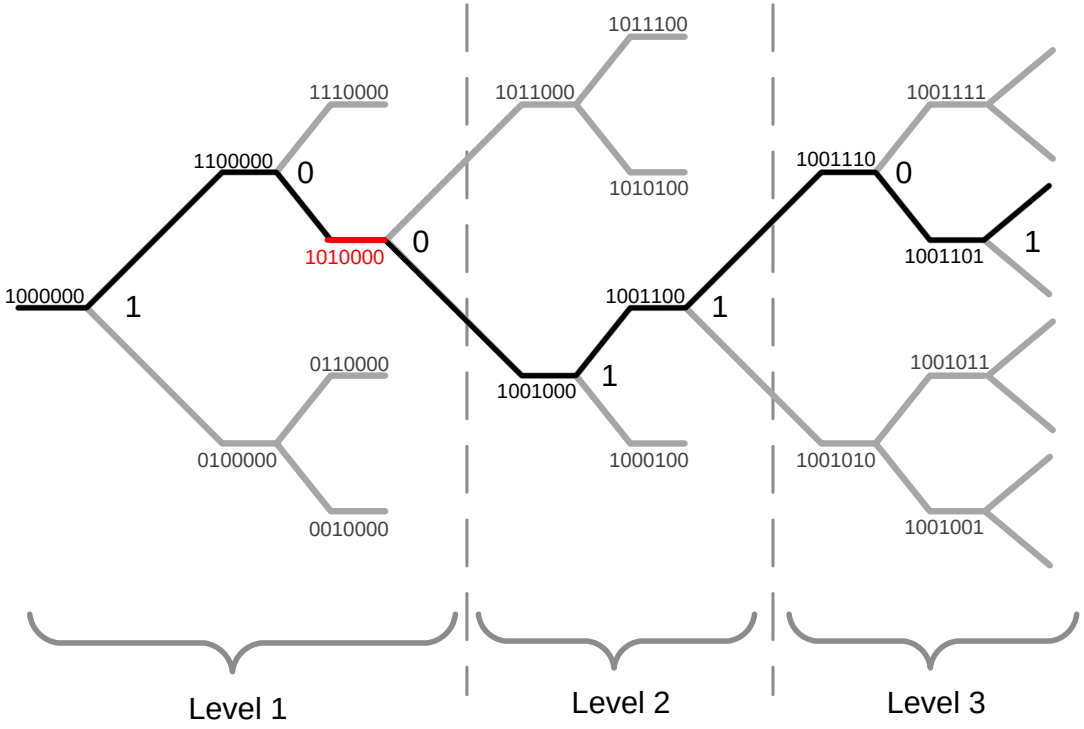
Basic Operation – Step 2



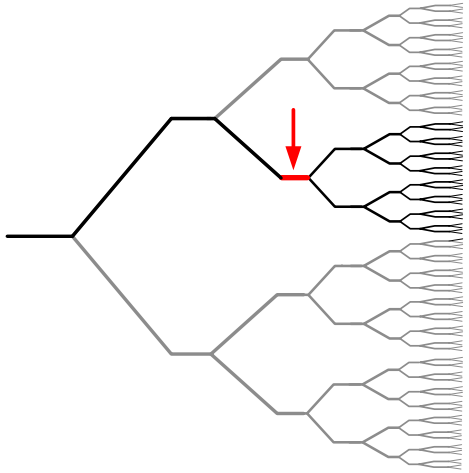
Step	Memory Address							
1	**1**	0	0	0	0	0	0	Level 1
2	1	**1**	0	0	0	0	0	
3								
4								Level 2
5								
6								Level 3
7								
	Level 1			Level 2		Level 3		



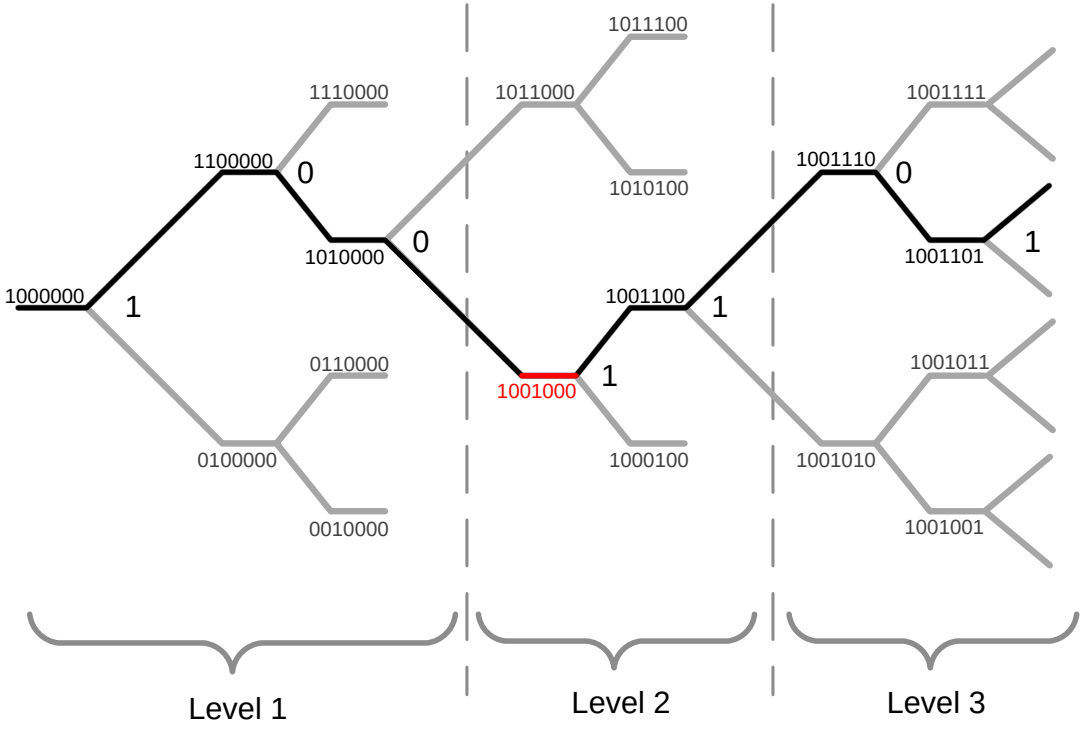
Basic Operation – Step 3



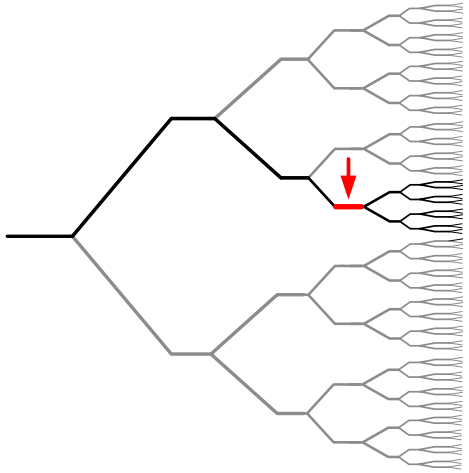
Step	Memory Address							
1	**1**	0	0	0	0	0	0	Level 1
2	1	**1**	0	0	0	0	0	
3	1	0	**1**	0	0	0	0	
4								Level 2
5								
6								Level 3
7								
	Level 1			Level 2		Level 3		



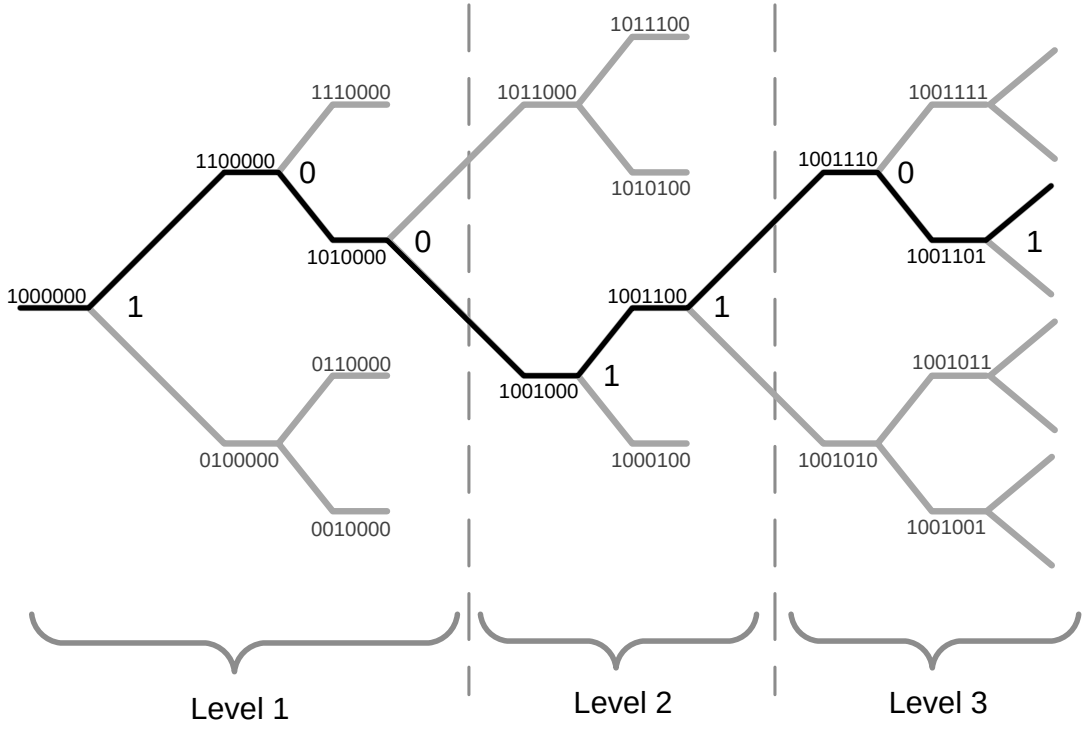
Basic Operation – Step 4



Step	Memory Address							
1	**1**	0	0	0	0	0	0	Level 1
2	1	**1**	0	0	0	0	0	
3	1	0	**1**	0	0	0	0	
4	1	0	0	**1**	0	0	0	Level 2
5								
6								Level 3
7								
	Level 1			Level 2		Level 3		



Basic Operation – Final Result



Step	Memory Address							
1	**1**	0	0	0	0	0	0	Level 1
2	1	**1**	0	0	0	0	0	
3	1	0	**1**	0	0	0	0	
4	1	0	0	**1**	0	0	0	Level 2
5	1	0	0	1	**1**	0	0	
6	1	0	0	1	1	**1**	0	Level 3
7	1	0	0	1	1	0	**1**	
	Level 1			Level 2		Level 3		

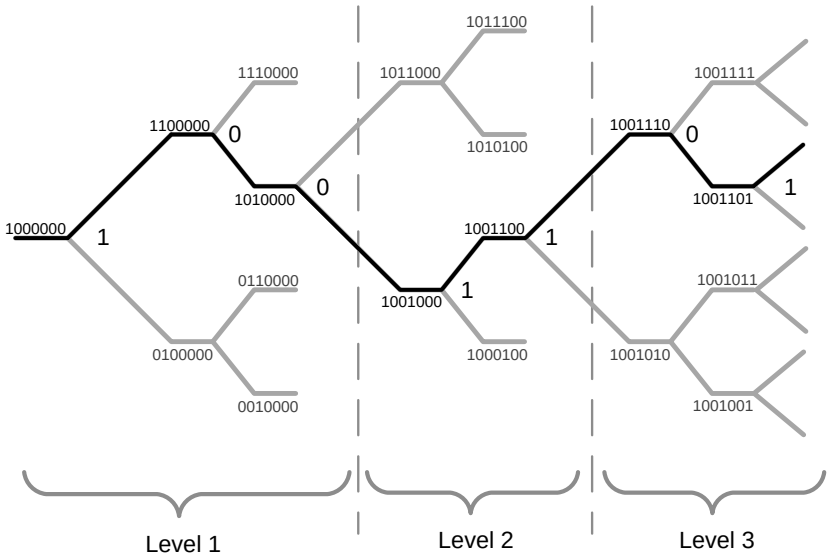
Basic Operation – Decode

Step	Memory Address							
1	**1**	0	0	0	0	0	0	Level 1
2	1	**1**	0	0	0	0	0	
3	1	0	**1**	0	0	0	0	
4	1	0	0	**1**	0	0	0	Level 2
5	1	0	0	1	**1**	0	0	
6	1	0	0	1	1	**1**	0	Level 3
7	1	0	0	1	1	0	**1**	
	Level 1			Level 2		Level 3		

- Simple decoding options
- Level Select
  - Determined by location of the ‘walking 1’
- Block Select
  - Use parent level’s address bits (either specific select lines from the parent level’s decoder or raw address bits will work)
- Block Decode
  - Use own level’s address bits

Reduced Number of Block Switches

- In BAM, number of block switches per conversion always equals the number of levels



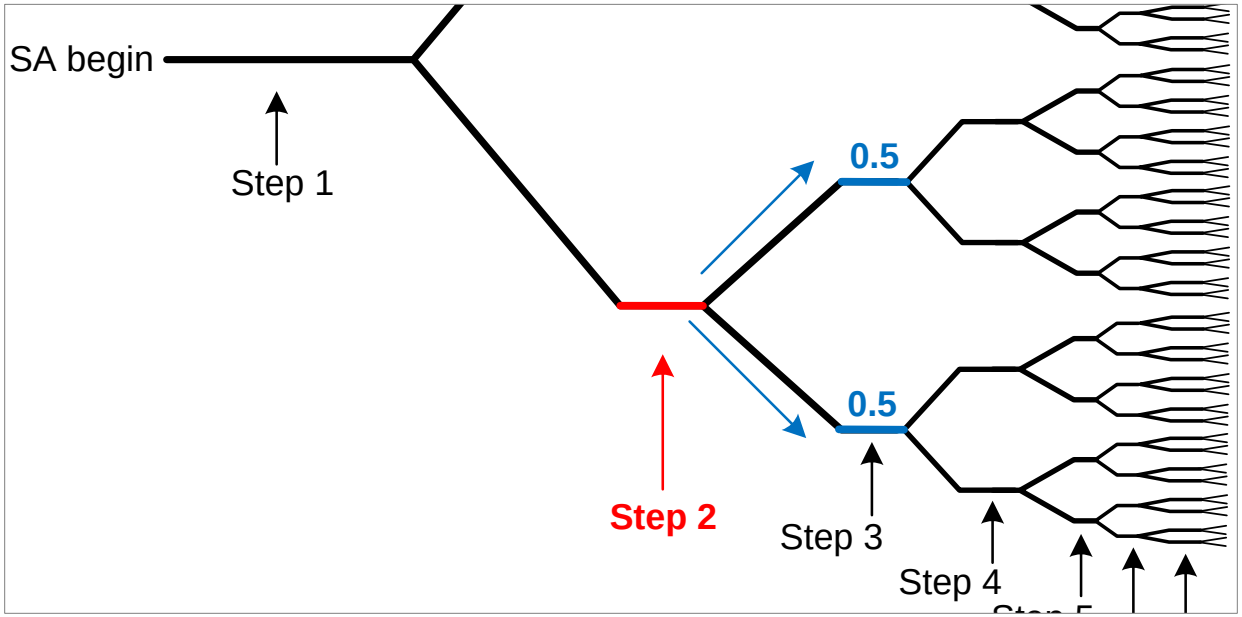
3-level Memory Depth and Organization	Average SRAM block switches	Average BAM block switches
7bit - 3x2x2	4.5	3
9bit – 3x3x3	5.5	3
12bit: 4x4x4	7.5	3
14bit: 4x4x6	8.5	3

Pre-fetching

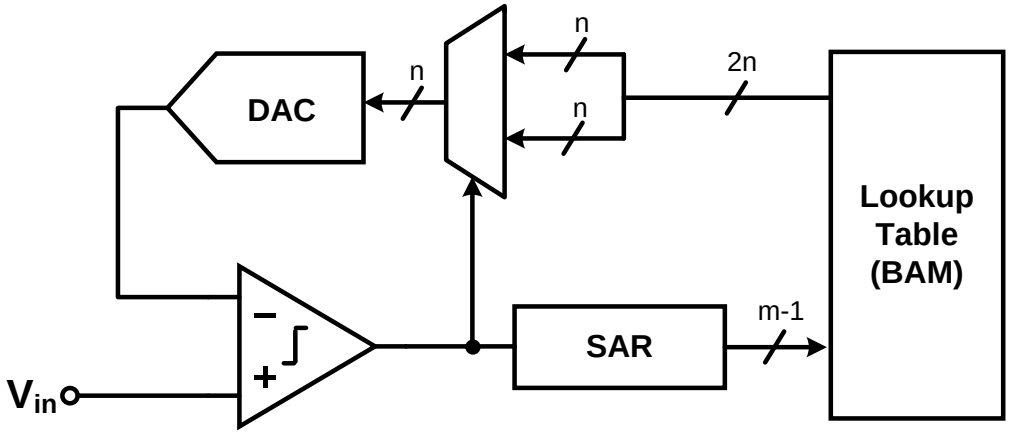
Useful Properties of Binary Search

Property 3

- Only the two children nodes directly below the current node have a chance of being accessed on the next step.
- Reduce latency by pre-fetching both possible children nodes during the parent's step



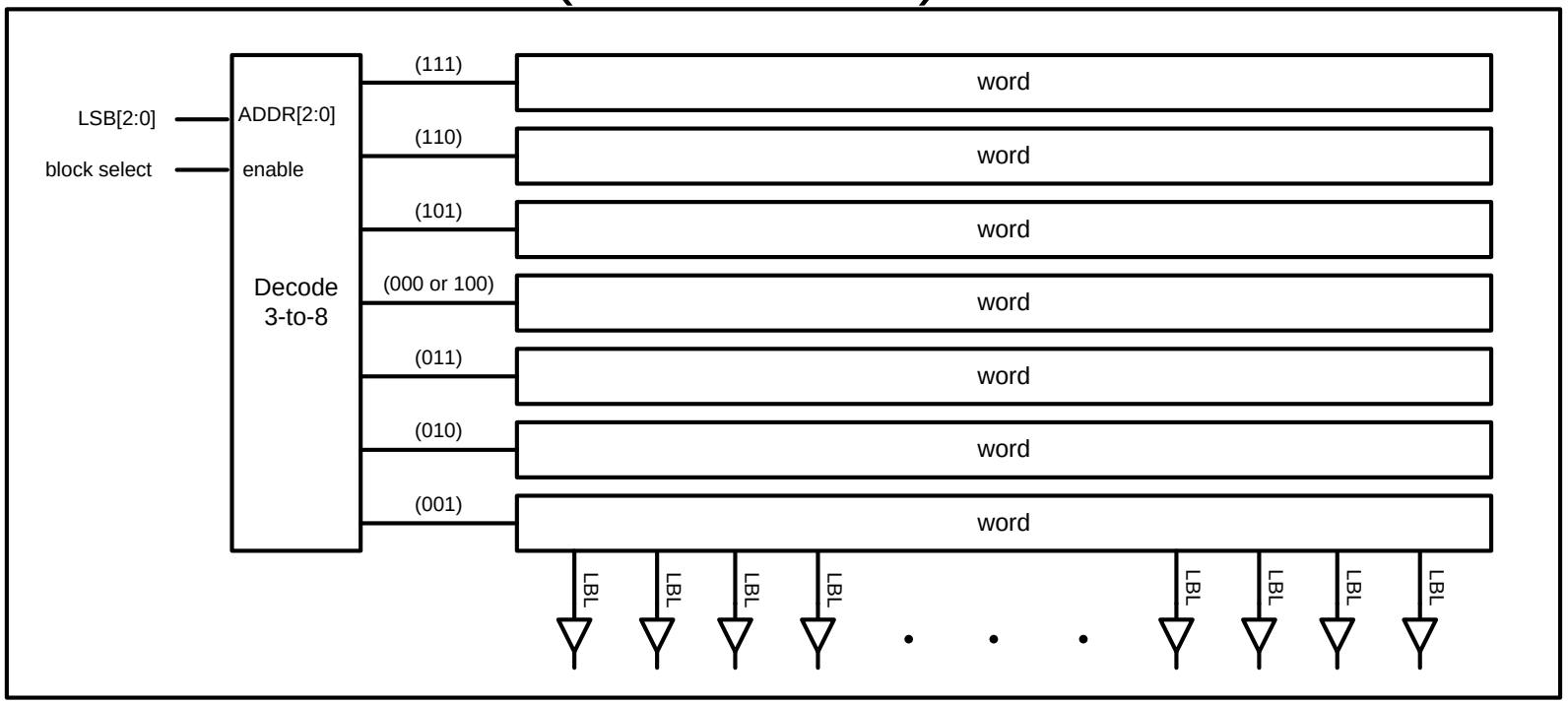
Pre-fetch Top Level Changes



- Reduce effective access latency
- Store both children words at the parent's address

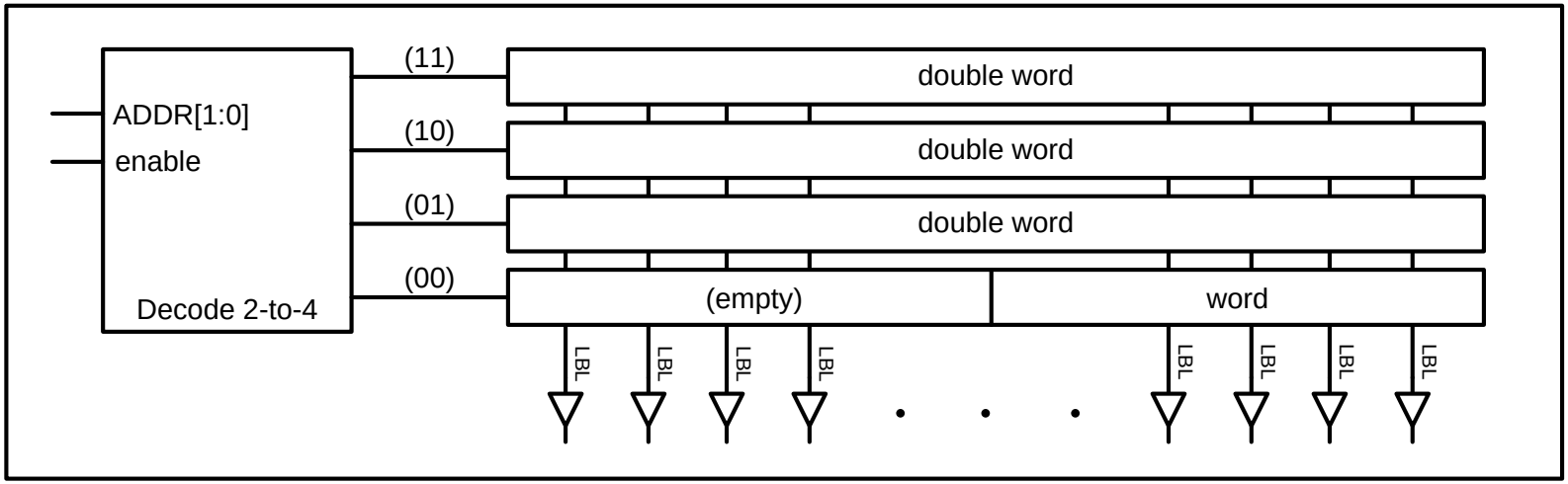
Sub-Block Re-Structuring for Pre-Fetch

Level 1 Sub-Block (No Prefetch)



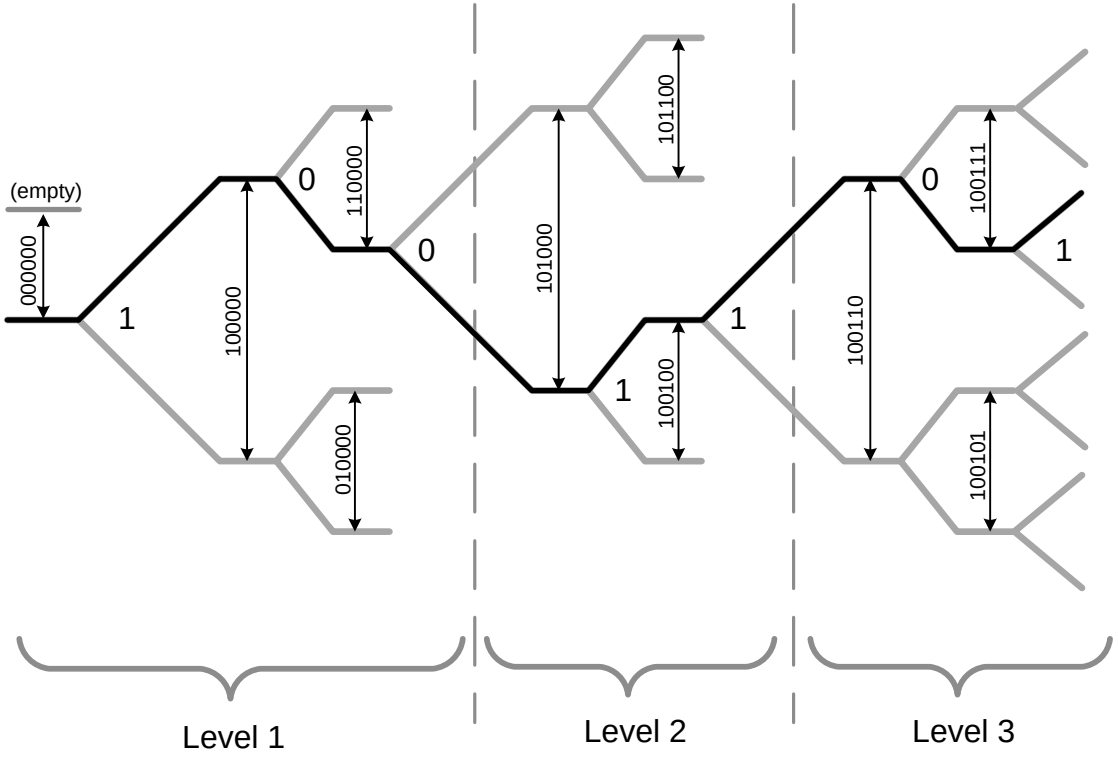
Sub-Block Re-Structuring for Pre-Fetch

Level 1 Sub-Block (Prefetch)



Pre-charging

Pre-Charging



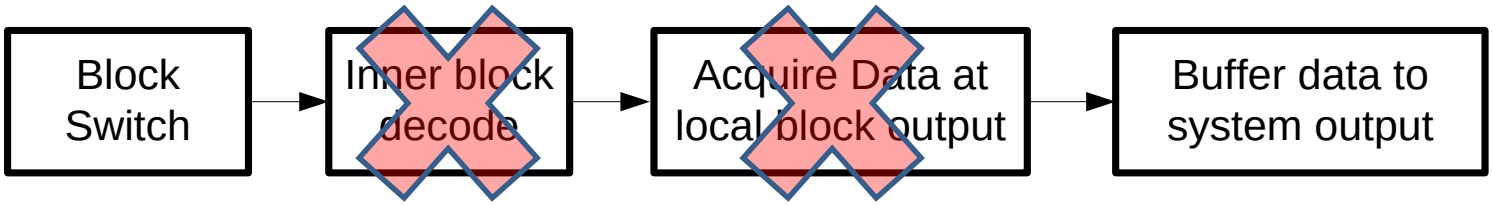
Step	Memory Address							
(7)	0	0	0	0	0	0		Level 1
1	**1**	0	0	0	0	0		
2	1	**1**	0	0	0	0		
3	1	0	**1**	0	0	0		Level 2
4	1	0	0	**1**	0	0		
5	1	0	0	1	**1**	0		Level 3
6	1	0	0	1	1	**1**		
7	0	0	0	0	0	**0**		
	Level 1		Level 2		Level 3			

- With pre-fetching implemented, there is now a word which is guaranteed to be the first accessed after a sub-block switch.

Pre-Charging

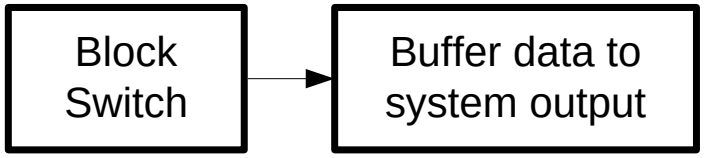
- Worst-case latency occurs when switching to a new sub-block
  - In some designs, output glitching can also occur
  - Solution: pre-charging
- When not selected, a sub-block's 'off state' is to pre-charges its local bit lines to the double-word which will always be requested first

Pre-Charging



- Worst-case latency occurs when switching to a new sub-block
  - In some designs, output glitching can also occur
  - Solution: pre-charging
- When not selected, a sub-block's 'off state' is to pre-charges its local bit lines to the double-word which will always be requested first

Pre-Charging



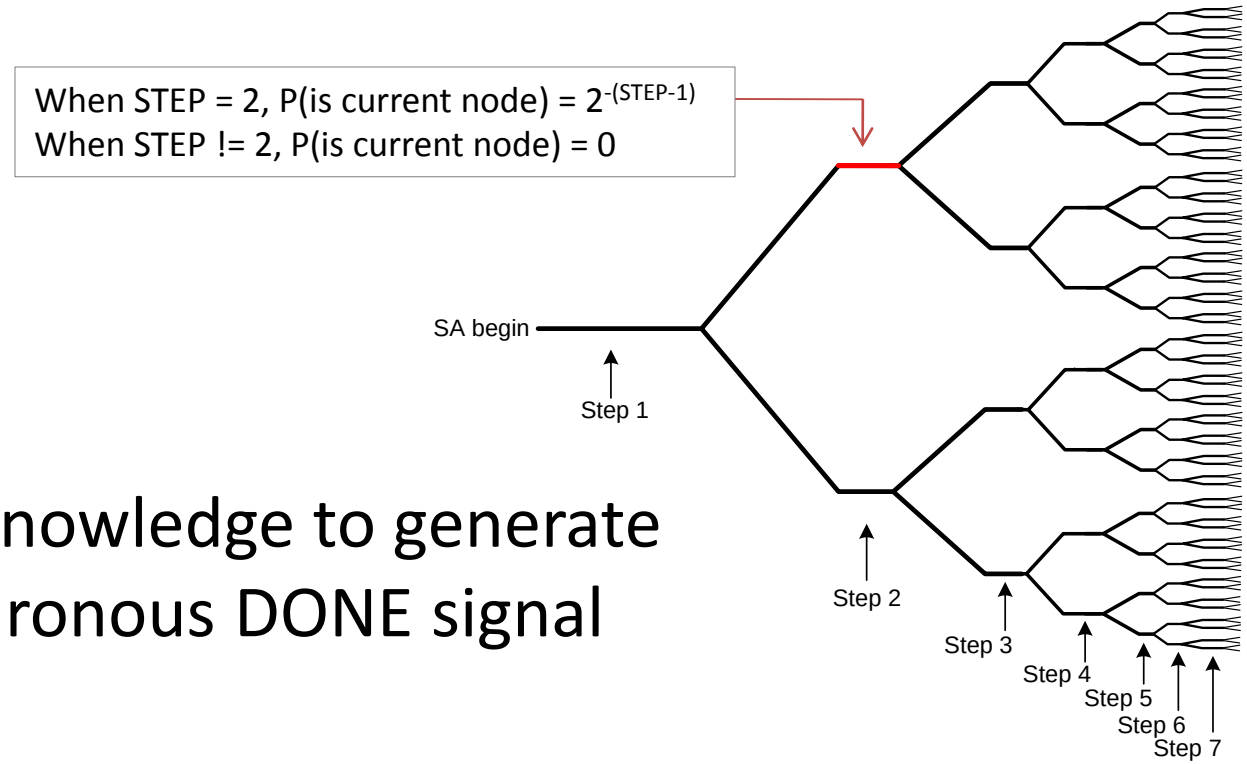
- Worst-case latency occurs when switching to a new sub-block
  - In some designs, output glitching can also occur
  - Solution: pre-charging
- When not selected, a sub-block's 'off state' is to pre-charges its local bit lines to the double-word which will always be requested first

Asynchronous BAM

Useful Properties of Binary Search

Property 4

- There is only one step number which a node can be visited during, and this step number is known for all nodes.

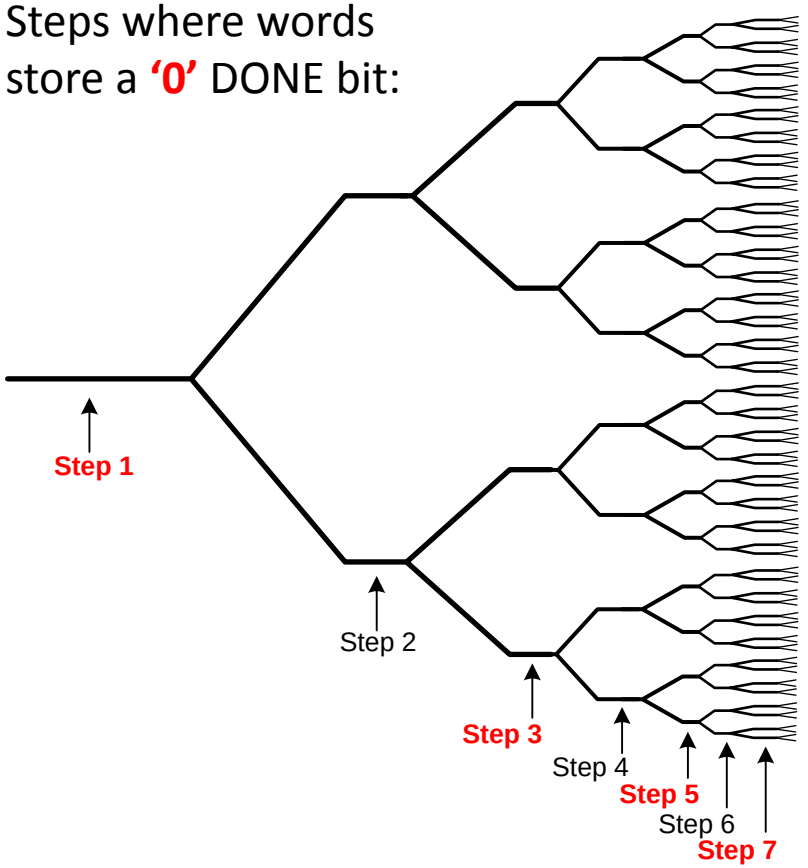


- Use this knowledge to generate an asynchronous DONE signal

Asynchronous BAM

Step	DONE bit value
1	0
2	1
3	0
4	1
5	0
6	1
7	0

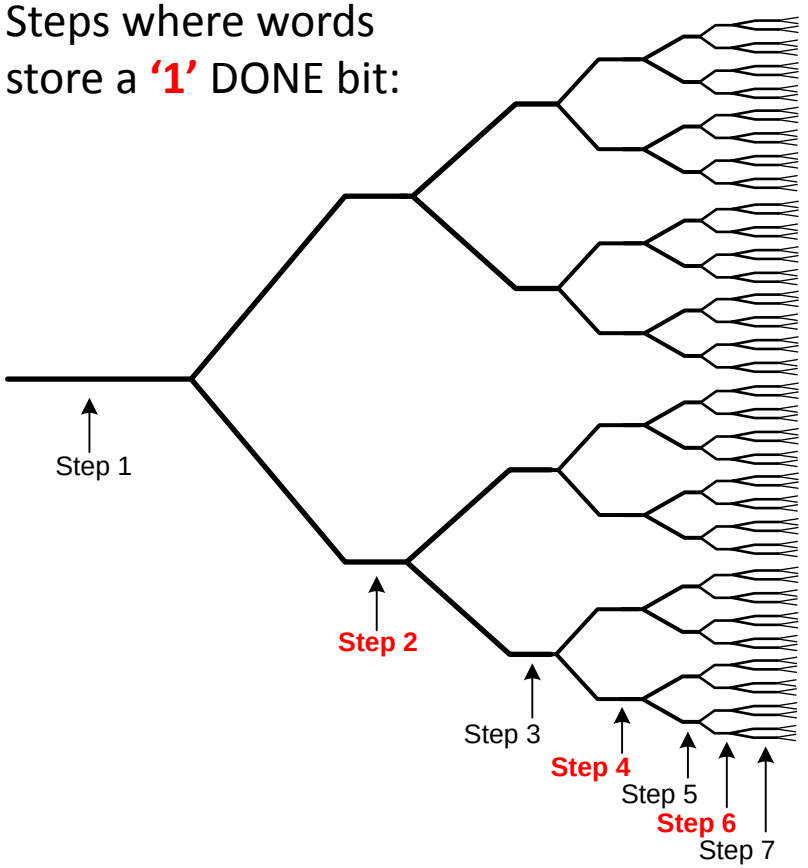
Steps where words store a **'0'** DONE bit:



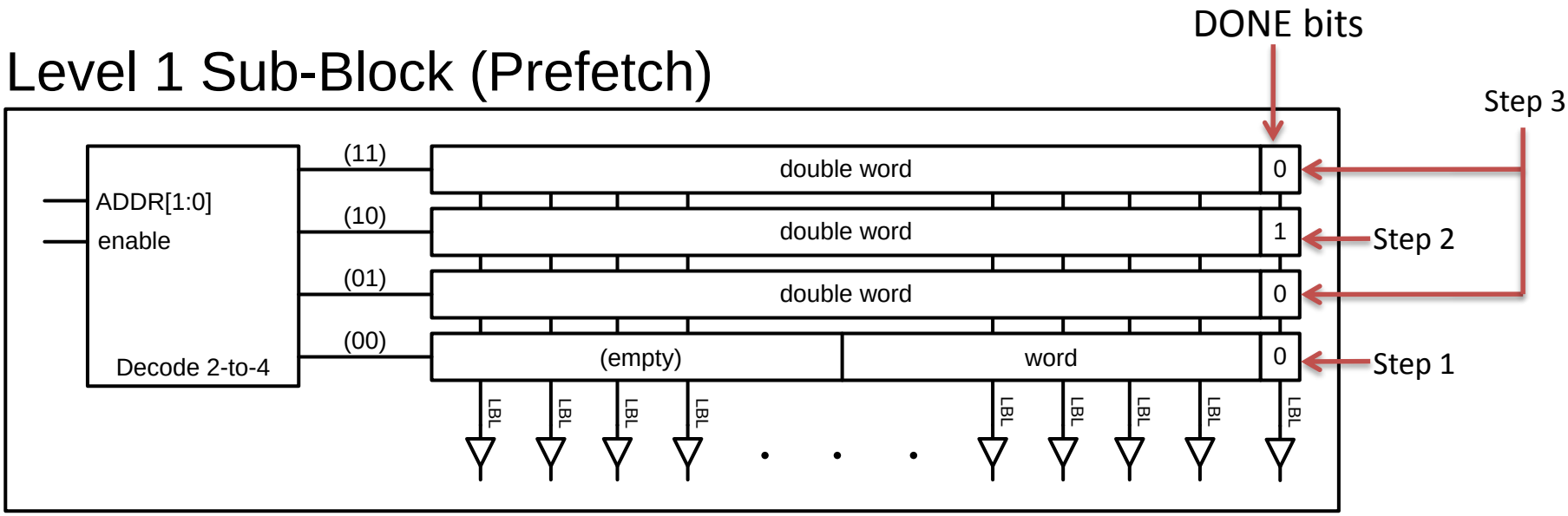
Asynchronous BAM

Step	DONE bit value
1	0
2	1
3	0
4	1
5	0
6	1
7	0

Steps where words store a **'1'** DONE bit:



Asynchronous BAM



Asynchronous BAM

- BAM access latency depends on where you are in the conversion
  - Early steps have low latency due to tree structuring.
- Fast early steps, slower later steps
  - Asynchronous BAM
  - Compatible with incomplete-settling, metastability-reduction, and other well known SAR techniques

Conclusion

Conclusion

- Binary Access Memory (BAM)
  - Improve performance (speed, power) by customizing lookup table for binary search tree memory access patterns
  - Key concepts
    - Blocks organized by location in tree rather than code
    - Most frequently accessed blocks are easiest to access
    - Prefetch the two possible ‘next’ codes in advance
    - Precharge bit-lines for fast access on block switch steps
    - Encode a DONE clock bit for asynchronous operation

Binary Access Memory:

An Optimized Lookup Table for Successive Approximation Applications

Benjamin Hershberg*, Skyler Weaver*, Seiji Takeuchi†, Koichi Hamashita†, Un-Ku Moon*

*School of Electrical Engineering and Computer Science, Oregon State University

†Asahi Kasei EMD Corporation, Atsugi, Japan