

ESE5320: System-on-a-Chip Architecture

Day 17: Oct. 30, 2023
LZW, Associative Maps



Penn ESE5320 Fall 2023 -- DeHon

1

Today

- Part 1: LZW Compression Setup (Part 1)
- Part 2" LZW Compression Algorithm (Part 2)
- Associative Memory
 - Custom (part 3)
 - FPGA (Part 4, time permitting, probably not)

Penn ESE5320 Fall 2023 -- DeHon

2

Message

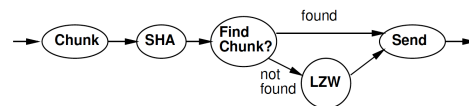
- Can adaptively compress data using recurring substrings
- Rich design space for Maps

Penn ESE5320 Fall 2023 -- DeHon

3

3

Part 1: LZW Compression Setup



Penn ESE5320 Fall 2023 -- DeHon

4

4

Preclass 1

- I AM S<2,3><5,4><0,4>
- Message?

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
I		A	M		S											

Penn ESE5320 Fall 2023 -- DeHon

5

5

Preclass 2, 3

- Bits in unencoded (decoded) message?
 - Assume 8b char
- Bits for encoded message?
 - Assume 9b for character
 - 1 bit to say is a character, then 8b char
 - And 9b for <x,y> pair
 - 1 bit char, 4b for each of x and y

Penn ESE5320 Fall 2023 -- DeHon

6

6

Idea

- Use data already sent as the dictionary
 - Give short names to things in dictionary
 - Don't need to pre-arrange dictionary
 - Adapt to common phrases/idioms in a particular document

Penn ESE5320 Fall 2023 -- DeHon

7

7

Encoding

- Greedy simplification
 - Encode by successively selecting the longest match between the head of the remaining string to send and the current window

Penn ESE5320 Fall 2023 -- DeHon

8

8

Algorithm Concept

- While data to send
 - Find largest match in window of data sent
 - If length too small (length=1)
 - Send character
 - Else
 - Send $\langle x, y \rangle = \langle \text{match-pos}, \text{length} \rangle$
 - Add data encoded into sent window

Penn ESE5320 Fall 2023 -- DeHon

9

9

Preclass 4

- How many comparisons per invocation?

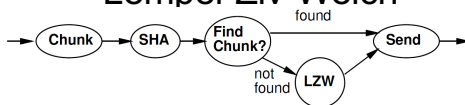
```
#define DICT_SIZE 4096
#define LENGTH 256
// clen<=LENGTH
int longest_match(char dict[DICT_SIZE], char candidate[LENGTH], int clen) {
    int best_len=0;
    int best_loc=-1;
    for (int i=0; i<DICT_SIZE-clen; i++) {
        j=0;
        while((candidate[j]==dict[i+j]) && (j<clen)) {
            j++;
        }
        if (j>best_len) {
            best_len=j;
            best_loc=i;
        }
    }
    return((best_loc<<8)|best_len);
}
```

Penn ESE5320 Fall 2023 -- DeHon

10

10

Part 2: LZW Compression Lempel-Ziv-Welch



Penn ESE5320 Fall 2023 -- DeHon

11

11

Idea

- Avoid $O(\text{Dictionary-size})$ work
 - Only need to match against positions that start with the character(s) in string to encode
 - Separate dictionary for each?

0	1	2	3	4	5	6	7	8	9	10
I		A	M		S					

Only check position 0 for "starts with I"

Penn ESE5320 Fall 2023 -- DeHon

12

12

Idea

- Avoid $O(\text{Dictionary-size})$ work
 - Only need to match against positions that start with the character(s) in string to encode
 - Separate dictionary for each?
- T-dictionary:
 - Tap, The, Their, Then, There, Tuesday

Penn ESE5320 Fall 2023 -- DeHon

13

13

Idea

- Avoid $O(\text{Dictionary-size})$ work
 - Only need to match against positions that start with the character(s) in string to encode
 - Separate dictionary for each?
- T-dictionary:
 - Tap, **The**, **Their**, **Then**, **There**, Tuesday
- If prefix same, why check redundantly?
 - Generalize: Store things with common prefix together
 - Share prefix among substrings

Penn ESE5320 Fall 2023 -- DeHon

Represent all strings as prefix tree

14

14

Idea

- Avoid $O(\text{Dictionary-size})$ work
 - Only need to match against positions that start with the character(s) in string to encode
 - Separate dictionary for each?
- If prefix same, why check redundantly?
 - Store things with common prefix together
 - Share prefix among substrings
 - Represent all strings as prefix tree
- Follow prefix trees with **fixed** work per input character

Penn ESE5320 Fall 2023 -- DeHon

15

15

Tree Algorithm

- Tree Root for each character
- Follow tree according to input until no more match
- Send <name of last tree node>
 - Dictionary entry
 - Named sequentially by insertion
- Extend tree (dictionary) with new character
- Start over with this character

Penn ESE5320 Fall 2023 -- DeHon

16

16

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			

Penn ESE5320 Fall 2023 -- DeHon

17

17

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	???	I	???	???

Penn ESE5320 Fall 2023 -- DeHon

18

18

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
<spc>	73	I	256	I<-spc
I				

I
↓
<spc>
↓
256

Penn ESE5320 Fall 2023 -- DeHon

19

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
<spc>	73	I	256	I<-spc
A	???	spc	???	???

I
↓
<spc>
↓
256

Penn ESE5320 Fall 2023 -- DeHon

20

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
<spc>	73	I	256	I<-spc
A	32	spc	257	spc<-A

I <spc>
↓ ↓
<spc> A
↓ ↓
256 257

Penn ESE5320 Fall 2023 -- DeHon

21

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
<spc>	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	??	A	???	???

I <spc>
↓ ↓
<spc> A
↓ ↓
256 257

Penn ESE5320 Fall 2023 -- DeHon

22

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
<spc>	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M

I <spc>
↓ ↓
<spc> A
↓ ↓
256 257

A
↓
M
↓
258

Penn ESE5320 Fall 2023 -- DeHon

23

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
<spc>	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	???	M	???	???

I <spc>
↓ ↓
<spc> A
↓ ↓
256 257

A
↓
M
↓
258

<spc>
↓
A
↓
257

Penn ESE5320 Fall 2023 -- DeHon

24

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc

Penn ESE5320 Fall 2023 -- DeHon 25

25

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	???	spc	???	???

Penn ESE5320 Fall 2023 -- DeHon 26

26

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S

Penn ESE5320 Fall 2023 -- DeHon 27

27

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	???	S	???	???

Penn ESE5320 Fall 2023 -- DeHon 28

28

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A

Penn ESE5320 Fall 2023 -- DeHon 29

29

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	???	A	???	???

Penn ESE5320 Fall 2023 -- DeHon 30

30

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		

Penn ESE5320 Fall 2023 -- DeHon 31

31

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		
	???	AM	???	???

Penn ESE5320 Fall 2023 -- DeHon 32

32

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		
	258	AM	262	258<-spc

Penn ESE5320 Fall 2023 -- DeHon 33

33

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		
	258	AM	262	258<-spc

Penn ESE5320 Fall 2023 -- DeHon 34

34

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		
	258	AM	262	258<-spc
S	???	spc	???	???

Penn ESE5320 Fall 2023 -- DeHon 35

35

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		
	258	AM	262	258<-spc
S	<nothing>	spc		

Penn ESE5320 Fall 2023 -- DeHon 36

36

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		
	258	AM	262	258<-spc
S	<nothing>	spc		
A	???	spc S	???	???

Penn ESE5320 Fall 2023 -- DeHon 37

37

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		
	258	AM	262	258<-spc
S	<nothing>	spc		
A	<nothing>	spc S	263	260<-A

Penn ESE5320 Fall 2023 -- DeHon 38

38

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		
	258	AM	262	258<-spc
S	<nothing>	spc		
A	260	spc S	263	260<-A
M	???	A	???	???

Penn ESE5320 Fall 2023 -- DeHon 39

39

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		
	258	AM	262	258<-spc
S	<nothing>	spc		
A	260	spc S	263	260<-A
M	<nothing>	A		

Penn ESE5320 Fall 2023 -- DeHon 40

40

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		
	258	AM	262	258<-spc
S	<nothing>	spc		
A	260	spc S	263	260<-A
M	<nothing>	A	???	???

Penn ESE5320 Fall 2023 -- DeHon 41

41

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>	A		
	258	AM	262	258<-spc
S	<nothing>	spc		
A	260	spc S	263	260<-A
M	<nothing>	A		

Penn ESE5320 Fall 2023 -- DeHon 42

42

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
A	73	I	256	I<-spc
M	65	A	258	A<-M
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>			
S	258	AM	262	258<-spc
A	<nothing>			
M	260	spc S	263	260<-A
I	<nothing>	A		
	<nothing>	AM		
	<nothing>	AM spc	???	???

Penn ESE5320 Fall 2023 -- DeHon 43

43

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
A	73	I	256	I<-spc
M	65	A	258	A<-M
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>			
S	258	AM	262	258<-spc
A	<nothing>			
M	260	spc S	263	260<-A
I	<nothing>	A		
	<nothing>	AM		
	262	AM spc	264	262<-I

Penn ESE5320 Fall 2023 -- DeHon 44

44

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
A	73	I	256	I<-spc
M	65	A	258	A<-M
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>			
S	258	AM	262	258<-spc
A	<nothing>			
M	260	spc S	263	260<-A
I	<nothing>	A		
	<nothing>	AM		
	262	AM spc	264	262<-I

Penn ESE5320 Fall 2023 -- DeHon 45

45

Large Memory Implementation

- int encode[SIZE][256];
- Name tree node by insertion order
- c is a character
- Encode[x][c] holds the next tree node that extends tree node x by symbol c
 - Or NONE if there is no such tree node

Penn ESE5320 Fall 2023 -- DeHon 46

46

Memory Tree Algorithm

```

curr=0; // pointer into input chunk
nextdict=NUM_SYMBOLS;
// dict[i]= symbol i for i<NUM_SYMBOLS
while (curr<chunk_size)
  while(encode[x][input[curr]]!=NONE)
    x=encode[x][input[curr]]; curr++;
  send x
  tree[nextdict].parent=x;
  tree[nextdict].sym=input[curr];
  encode[x][input[curr]]=nextdict++;
  x=input[curr]; curr++;

```

Penn ESE5320 Fall 2023 -- DeHon 47

47

Memory Tree Algorithm

```

curr=0; // pointer into input chunk
nextdict=NUM_SYMBOLS;
// dict[i]= symbol i for i<NUM_SYMBOLS
while (curr<chunk_size)
  while(encode[x][input[curr]]!=NONE)
    x=encode[x][input[curr]]; curr++;
  send x
  tree[nextdict].parent=x;
  tree[nextdict].sym=input[curr];
  encode[x][input[curr]]=nextdict++;
  x=input[curr]; curr++;

```

Follow Tree

Penn ESE5320 Fall 2023 -- DeHon 48

48

Memory Tree Algorithm

```
curr=0; // pointer into input chunk
nextdict=NUM_SYMBOLS;
// dict[i]= symbol i for i<NUM_SYMBOLS
while (curr<chunk_size)
    while(encode[x][input[curr]]!=NONE)
        x=encode[x][input[curr]]; curr++;
    send x
    tree[nextdict].parent=x;
    tree[nextdict].sym=input[curr];
    encode[x][input[curr]]=nextdict++;
    x=input[curr]; curr++;
```



Penn ESE5320 Fall 2023 -- DeHon

49

49

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
I	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
M	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>			
M	258	AM	262	258<-spc
S	<nothing>			
A	260	spc S	263	260<-A
M	<nothing>	A		
M	<nothing>	AM		
I	262	AM spc	264	262<-I

Penn ESE5320 Fall 2023 -- DeHon

50

50

Memory Tree Algorithm

```
curr=0; // pointer into input chunk
nextdict=NUM_SYMBOLS;
// dict[i]= symbol i for i<NUM_SYMBOLS
while (curr<chunk_size)
    while(encode[x][input[curr]]!=NONE)
        x=encode[x][input[curr]]; curr++;
    send x
    tree[nextdict].parent=x;
    tree[nextdict].sym=input[curr];
    encode[x][input[curr]]=nextdict++;
    x=input[curr]; curr++;
```

How much work per character to encode? Hint: between curr++ executions?

Penn ESE532.0 Fall 2023 -- DeHon

51

51

Map

- In Part 3 will talk about higher-level maps
- Version of encode/decode in
 - Geeks-for-Geeks description
 - Decoder we provide
- Based on Maps of strings
 - Simpler to state
 - Likely not O(1) per symbol
 - More expensive implementation in hardware
 - Version here better for hardware (keeps simple)

Penn ESE5320 Fall 2023 -- DeHon

52

52

Algorithm with Map

```
curr=0; // pointer into input chunk
nextdict=NUM_SYMBOLS;
// dict initialized symbol i for i<NUM_SYMBOLS
while (curr<chunk_size)
    while(dict.lookup(x+input[curr])!=NONE)
        x=x+input[curr]; curr++;
    send dic.lookup(x);
    dict.insert(x+input[curr],nextdict++);
    x=String(input[curr]); curr++;
```

Penn ESE5320 Fall 2023 -- DeHon

53

53

I AM SAM SAM I AM

Input	Send	Dict	Add Entry	Add What
I	<nothing>			
I	73	I	256	I<-spc
A	32	spc	257	spc<-A
M	65	A	258	A<-M
M	77	M	259	M<-spc
S	32	spc	260	spc<-S
A	83	S	261	S<-A
M	<nothing>			
M	258	AM	262	258<-spc
S	<nothing>			
A	260	spc S	263	260<-A
M	<nothing>	A		
M	<nothing>	AM		
I	262	AM spc	264	262<-I

Penn ESE5320 Fall 2023 -- DeHon

54

54

Compact Memory

- `int encode[SIZE][256];`
- How many entries in this table are not NONE?
 - Maximum number of times execute:
`encode[x][input[curr]]=nextdict++;`
 - Maximum number of entries added to encode?
 - Maximum number of NONE entries in encode?

Penn ESE5320 Fall 2023 -- DeHon

55

55

Compact Memory

- `int encode[SIZE][256];`
- Table is very sparse
- If store as associative memory
 - At most SIZE entries
- Look at how to implement associative memories next

Penn ESE5320 Fall 2023 -- DeHon

56

56

LZW So Far – 4KB chunks

- Brute Force
 - Needs one byte per byte = 4KB = 1 BRAM
 - `DICT_SIZE=4096` comparisons per byte
- Dense memory `encode[SIZE][256]`
 - 12b key (1.5B)
 - Need $1.5 * 4096 * 256 = 384 * 4KB$
= 384 BRAMs
 - 1 comparison and lookup per byte
 - (maybe should be `SIZE+256`)

Penn ESE5320 Fall 2023 -- DeHon

57

57

4K Chunk LZW Search

	BRAMs	Operations/byte
Brute Search	1	4K
Tree with Dense RAM	384	1

36Kb BRAMs on ZU3EG = 216

Penn ESE5320 Fall 2023 -- DeHon

58

58

Complexity

- Reminder from Day 15
 - Optimized implementations tend to be more complex
 - Large problems may force more complexity
 - (e.g. Large window, deal with limited BRAMs on chip)
 - Seeing examples of that today...

Penn ESE5320 Fall 2023 -- DeHon

59

59

Associative Memories

Part 3

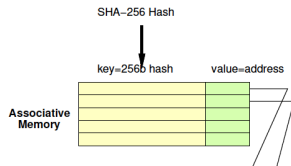
Penn ESE5320 Fall 2023 -- DeHon

60

60

Associative Memory

- Maps from a key to a value
- Key not necessarily dense
 - Contrast simple RAM
 - Cannot afford 2^{256} word memory



Penn ESE5320 Fall 2023 -- DeHon

61

Associative Memories

- Use for deduplication
- Also may use in LZW to reduce BRAMs
 - Just saw
 - **Problem:** Simple 2D tree table requires too many BRAMs
 - **Opportunity:** Tree table sparse

Penn ESE5320 Fall 2023 -- DeHon

62

Custom Hardware Associative Memory

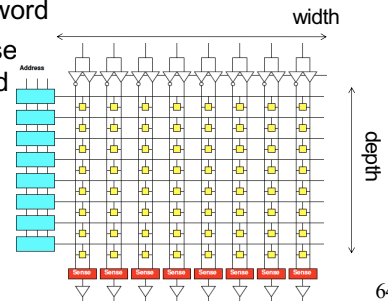
Penn ESE5320 Fall 2023 -- DeHon

63

63

Memory Block Review

- Match on address
- Select wordline for a row
- Reads out a word
- Address dense and hardwired
- One row for each 2^A bits values



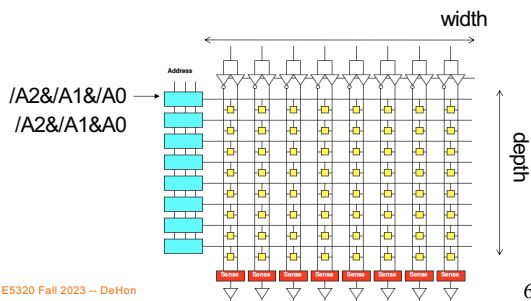
Penn ESE5320 Fall 2023 -- DeHon

64

64

Address Blocks

- Each address match is AND

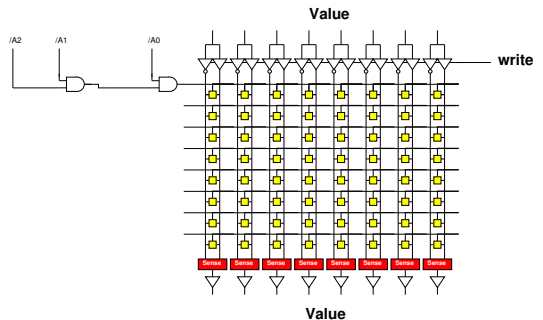


Penn ESE5320 Fall 2023 -- DeHon

65

65

Address Blocks



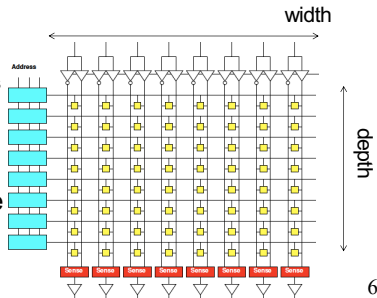
Penn ESE5320 Fall 2023 -- DeHon

66

66

Memory Block Associative

- Want address as key
- Word is value
- Key sparse
- Rows $< 2^{\text{keybits}}$
- Entries $< 2^{\text{keybits}}$
- Key programmable



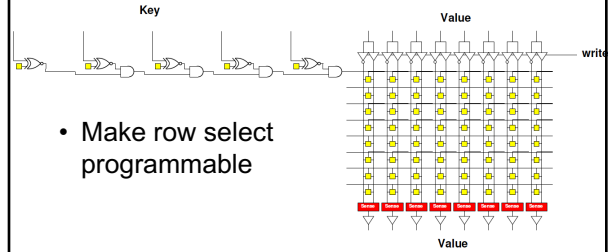
Penn ESE5320 Fall 2023 -- DeHon

67

67

Programmable Key

- Make row select programmable

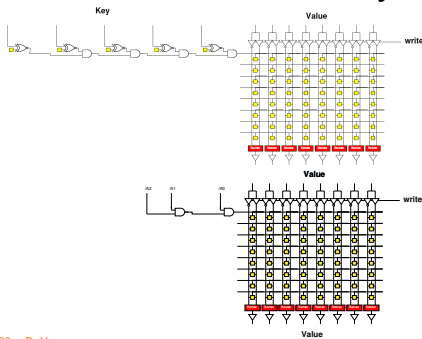


Penn ESE5320 Fall 2023 -- DeHon

68

68

Contrast Assoc. and Dense Memory

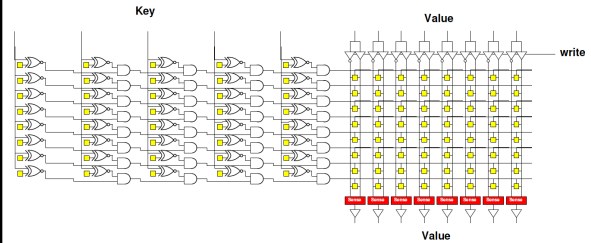


Penn ESE5320 Fall 2023 -- DeHon

69

69

Associative Memory Bank



Penn ESE5320 Fall 2023 -- DeHon

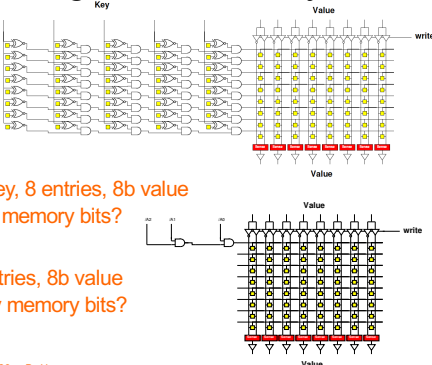
70

70

Programmable Key

Assoc: 5b key, 8 entries, 8b value
How many memory bits?

Direct: 8 entries, 8b value
How many memory bits?



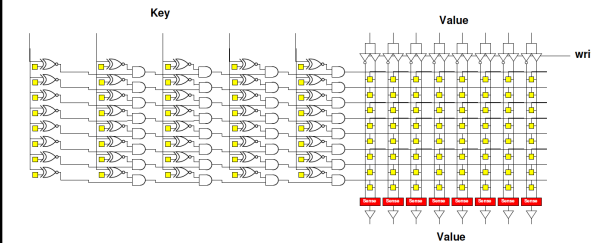
Penn ESE5320 Fall 2023 -- DeHon

71

71

Associative Memory Bank

- Memory cells = entries*(keybits+valuebits)

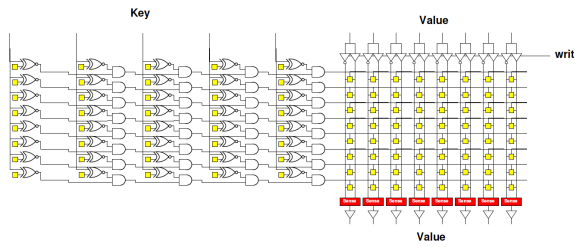


Penn ESE5320 Fall 2023 -- DeHon

72

72

Associative Memory Bank



- Will need to be able to write into key
 - Another “fixed” decoder to generate key-word line for programming

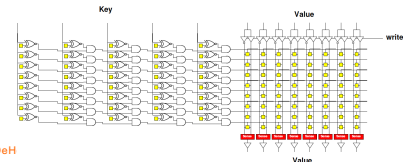
Penn ESE5320 Fall 2023 -- DeHon

73

73

Associate Memory Cost

- More expensive than equal capacity SRAM memory bank
 - Memory cells in decoder
 - Need to support write into key



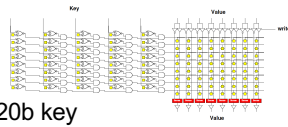
Penn ESE5320 Fall 2023 -- DeH

74

74

Associate Memory Cost

- Physical associative memory for 4KB LZW Chunk tree encode
 - 4K entries
 - 12b (pos) output
 - 12b (pos)+8b (char)=20b key
- Memory cells assoc.?
- Compare direct 4Kx12 memory (cells)?
 - How much larger is assoc. for same capacity?
- Compare 4096*256 with 12b result for dense LZW case (cells)?
 - How much larger to solve same problem



Penn ESE5320 Fall 2023 -- DeHon

75

75

Associative Memories FPGA

Part 4
(probably for Day 18)

[Skip wrapup](#)

Penn ESE5320 Fall 2023 -- DeHon

76

76

FPGA

- Has BRAMs – normal memories, not associative
- 36Kb BRAM
 - 512x72
- Can be 9b key → 72b value assoc.
 - Just using the memory sparsely
- Or interpret as programmable decoder with 72 match lines

Penn ESE5320 Fall 2023 -- DeHon

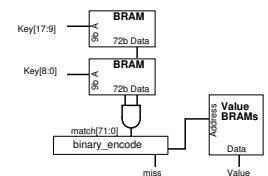
77

77

Assoc. Mem from BRAM

For wider match

- Cover 9b of key with each BRAM
- Use 72 output bits to indicate if one of 72 entries match
- AND together corresponding entries
- Get 72 match bits
- Re-encode match bits to lookup value



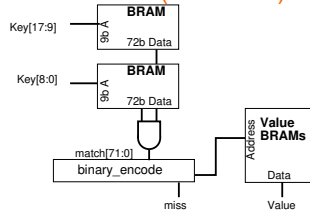
Penn ESE5320 Fall 2023 -- DeHon

78

78

BRAM Associative Memory

- Previous slide expands match width
- How would we expand capacity?
 - Hint: how get a wider word (144b word)?

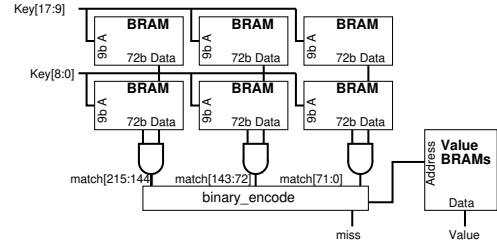


Penn ESE5320 Fall 2023 -- DeHon

79

79

BRAM Associative Memory



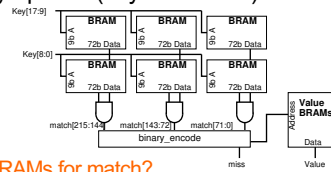
Penn ESE5320 Fall 2023 -- DeHon

80

80

Associative Memory Cost

- Match unit
 - Requires 1 BRAM per 9b of key per 72 entries
 - $\lceil \text{keylen}/9\text{b} \rceil \times \lceil \text{entries}/72 \rceil$
 - Asymptotically optimal ($\text{keylen} \times \text{entries}$)
 - But large constants
- LZW
 - 4K entries
 - 20b key
 - How many BRAMs for match?



Penn ESE5320 Fall 2023 -- DeHon

81

81

4K LZW Chunk Search: Fully associative

- Match BRAMs:
 - Match key: 20b
 - Entries: 4096
- Value BRAMs:
 - 12b (state [position])
 - 12b x 4096 entries
 - Takes 2 BRAMs

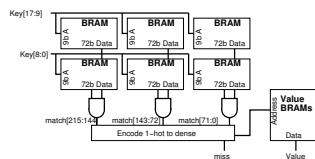
Penn ESE5320 Fall 2023 -- DeHon

82

82

Example Stored Values

Key[17:9]	Key[8:0]	Value
0x001	0x014	0x01
0x001	0x01	0x34
0x0F0	0x014	0xE3
0x0C8	0x113	0xCC



Penn ESE5320 Fall 2023 -- DeHon

83

83

Memory Contents

Key[17:9] Match BRAM

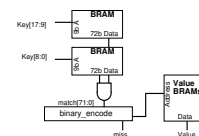
Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	0	0	0	1	1
0x014	0	0	0	0	0	0	0	0
0x0C8	0	0	0	0	1	0	0	0
0x0F0	0	0	0	0	0	1	0	0
0x113	0	0	0	0	0	0	0	0

Value BRAM

Addr	Value
0x00	0x01
0x01	0x34
0x02	0xE3
0x03	0xCC
0x04	
0x05	
0x06	

Key[8:0] Match BRAM

Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	0	0	0	1	0
0x014	0	0	0	0	0	1	0	1
0x0C8	0	0	0	0	0	0	0	0
0x0F0	0	0	0	0	0	0	0	0
0x113	0	0	0	0	1	0	0	0



Penn ESE5320 Fall 2023 -- DeHon (only show bottom 8 b; rest 0's)

84

84

Code Snippet

```

ap_uint<72> key_low[512];
ap_uint<72> key_high[512];
int value[72];

match_low=key_low[key%512];
match_high=key_high[(key>>9)%512];
match=match_low & match_high;
addr=binary_encode(match);
res=value[addr];

```

Penn ESE5320 Fall 2023 -- DeHon

85

How Lookup Work?

Key[17:9]	Key[8:0]	Value
0x001	0x014	0x01
0x001	0x01	0x34
0x0F0	0x014	0xE3
0x0C8	0x113	0xCC

Lookup 0x214 = 0x001 0x014

Penn ESE5320 Fall 2023 -- DeHon

86

Code Snippet

```

ap_uint<72> key_low[512];
ap_uint<712> key_high[512];
int value[72];

match_low=key_low[key%512];
match_high=key_high[(key>>9)%512];
match=match_low & match_high;
addr=binary_encode(match);
res=value[addr];

```

Penn ESE5320 Fall 2023 -- DeHon

87

Memory Contents

Key[17:9] Match BRAM

Addr	7	6	5	4	3	2	1	0	
0x001	0	0	0	0	0	0	1	1	match_low=key_low[key%512];
0x014	0	0	0	0	0	0	0	0	match_high=key_high[(key>>9)%512]
0x0C8	0	0	0	0	1	0	0	0	
0x0F0	0	0	0	0	0	1	0	0	
0x113	0	0	0	0	0	0	0	0	

Key[8:0] Match BRAM

Addr	7	6	5	4	3	2	1	0	
0x001	0	0	0	0	0	0	1	0	
0x014	0	0	0	0	0	1	0	1	
0x0C8	0	0	0	0	0	0	0	0	
0x0F0	0	0	0	0	0	0	0	0	
0x113	0	0	0	0	1	0	0	0	

What match_low, match_high?

(only show bottom 8 b; rest 0's)

Penn ESE5320 Fall 2023 -- DeHon

88

Memory Contents

Key[17:9] Match BRAM

Addr	7	6	5	4	3	2	1	0	
0x001	0	0	0	0	0	0	1	1	match_low=key_low[key%512];
0x014	0	0	0	0	0	0	0	0	match_high=key_high[(key>>9)%512]
0x0C8	0	0	0	0	1	0	0	0	match=match_low & match_high;
0x0F0	0	0	0	0	0	1	0	0	
0x113	0	0	0	0	0	0	0	0	

Key[8:0] Match BRAM

Addr	7	6	5	4	3	2	1	0	
0x001	0	0	0	0	0	0	1	0	
0x014	0	0	0	0	0	1	0	1	
0x0C8	0	0	0	0	0	0	0	0	
0x0F0	0	0	0	0	0	0	0	0	
0x113	0	0	0	0	1	0	0	0	

What match?

(only show bottom 8 b; rest 0's)

Penn ESE5320 Fall 2023 -- DeHon

89

What does binary_encode do?

binary_encode(0000000...010000)=0x40

- 10000...0 → 71
- 0000...01 → 0
- 0000...010 → 1
- for (i=0<i<72;i++)
 - If (bit[i]==1) return i
- Return(MISS); // if not find (i.e., all 0's)
- Technicalities – maybe check only one 1

Penn ESE5320 Fall 2023 -- DeHon

90

Memory Contents

Key[17:9] Match BRAM

Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	0	0	0	1	1
0x014	0	0	0	0	0	0	0	0
0x0C8	0	0	0	0	1	0	0	0
0x0F0	0	0	0	0	0	1	0	0
0x113	0	0	0	0	0	0	0	0

```
match_low=key_low[key%512];
match_high=key_high[(key>>9)%512];
match=match_low & match_high;
addr=binary_encode(match);
```

Key[8:0] Match BRAM

Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	0	0	0	1	0
0x014	0	0	0	0	0	1	0	1
0x0C8	0	0	0	0	0	0	0	0
0x0F0	0	0	0	0	0	0	0	0
0x113	0	0	0	0	1	0	0	0

What addr?

Penn ESE5320 (only show bottom 8 b; rest 0's)

91

91

Memory Contents

Key[17:9] Match BRAM

Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	0	0	0	1	1
0x014	0	0	0	0	0	0	0	0
0x0C8	0	0	0	0	1	0	0	0
0x0F0	0	0	0	0	0	1	0	0
0x113	0	0	0	0	0	0	0	0

Value BRAM

Addr	Value
0x00	0x01
0x01	0x34
0x02	0xE3
0x03	0xCC
0x04	
0x05	
0x06	

Key[8:0] Match BRAM

Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	0	0	0	1	0
0x014	0	0	0	0	0	1	0	1
0x0C8	0	0	0	0	0	0	0	0
0x0F0	0	0	0	0	0	0	0	0
0x113	0	0	0	0	1	0	0	0

res=value[addr];

What res?

Penn ESE5320 (only show bottom 8 b; rest 0's)

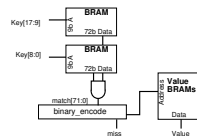
92

92

How Lookup Work?

Key[17:9]	Key[8:0]	Value
0x001	0x014	0x01
0x001	0x01	0x34
0x0F0	0x014	0xE3
0x0C8	0x113	0xCC

Lookup 0x214 = 0x001 0x014



Penn ESE5320 Fall 2023 -- DeHon

93

93

Memory Contents

Key[17:9] Match BRAM

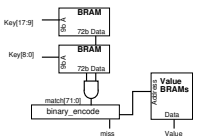
Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	0	0	0	1	1
0x014	0	0	0	0	0	0	0	0
0x0C8	0	0	0	0	1	0	0	0
0x0F0	0	0	0	0	0	1	0	0
0x113	0	0	0	0	0	0	0	0

Value BRAM

Addr	Value
0x00	0x01
0x01	0x34
0x02	0xE3
0x03	0xCC
0x04	
0x05	
0x06	

Key[8:0] Match BRAM

Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	0	0	0	1	0
0x014	0	0	0	0	0	1	0	1
0x0C8	0	0	0	0	0	0	0	0
0x0F0	0	0	0	0	0	0	0	0
0x113	0	0	0	0	1	0	0	0



Penn ESE5320 (only show bottom 8 b; rest 0's)

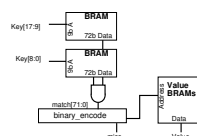
94

94

Add another entry

match	Key[17:9]	Key[8:0]	Value
0	0x001	0x014	0x01
1	0x001	0x01	0x34
2	0x0F0	0x014	0xE3
3	0x0C8	0x113	0xCC
4	0x0C8	0x01	0x2B

How BRAM contents change to add this entry for 0x19001



Penn ESE5320 Fall 2023 -- DeHon

95

95

Memory Contents

Key[17:9] Match BRAM

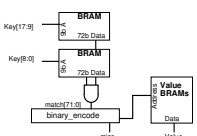
Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	0	0	0	1	1
0x014	0	0	0	0	0	0	0	0
0x0C8	0	0	0	0	1	0	0	0
0x0F0	0	0	0	0	0	1	0	0
0x113	0	0	0	0	0	0	0	0

Value BRAM

Addr	Value
0x00	0x01
0x01	0x34
0x02	0xE3
0x03	0xCC
0x04	
0x05	
0x06	

Key[8:0] Match BRAM

Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	0	0	0	1	0
0x014	0	0	0	0	0	1	0	1
0x0C8	0	0	0	0	0	0	0	0
0x0F0	0	0	0	0	0	0	0	0
0x113	0	0	0	0	1	0	0	0



Penn ESE5320 (only show bottom 8 b; rest 0's)

96

96

Memory Contents

Key[17:9] Match BRAM

Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	0	0	0	1	1
0x014	0	0	0	0	0	0	0	0
0x0C8	0	0	0	1	1	0	0	0
0x0F0	0	0	0	0	0	1	0	0
0x113	0	0	0	0	0	0	0	0

Key[8:0] Match BRAM

Addr	7	6	5	4	3	2	1	0
0x001	0	0	0	1	0	0	1	0
0x014	0	0	0	0	0	1	0	1
0x0C8	0	0	0	0	0	0	0	0
0x0F0	0	0	0	0	0	0	0	0
0x113	0	0	0	0	1	0	0	0

Value BRAM

Addr	Value
0x00	0x01
0x01	0x34
0x02	0xE3
0x03	0xCC
0x04	0x2B
0x05	
0x06	

Penn ESE5320 Fall 2023 -- DeHon

97

97

4K Chunk LZW Search

	BRAMs	Operations/byte
Brute Search	1	4K
Tree with Dense RAM	384	1
Tree with Full Assoc	173	1

36Kb BRAMs on ZU3EG = 216

Penn ESE5320 Fall 2023 -- DeHon

98

98

Checkpoint

- Notice levels of mapping:
 - Prefix Tree algorithm
 - Formulated on a 2D memory
 - Then implemented in assoc. memory
 - (later with Tree ... hash table)

Penn ESE5320 Fall 2023 -- DeHon

99

99

Big Ideas

- Can adaptively compress data using recurring substrings
 - With constant work for symbol
- Rich design space for engineering associative map solutions

Penn ESE5320 Fall 2023 -- DeHon

100

100

Admin

- Feedback (including HW7)
- Reading for Wednesday on Web
- First project milestone due Friday
 - Including teaming

Penn ESE5320 Fall 2023 -- DeHon

101

101