



# A VLSI Implementation of ADPCM Voice Encoder

Li Xu, Mark Yun, Saad Mahmoud

ELEC 422 Project

December 7, 2000

Null Group

# Outline

- Project overview
- ADPCM algorithm
- Block diagram
- Subcell (register, mux, adder)
- Control PLA
- Layout, floor planning, routing
- Design decisions
- Summary

# Project Overview

- Implement the ADPCM voice encoding algorithm in silicon
- Chip compresses 16-bit input voice data to 4-bit output

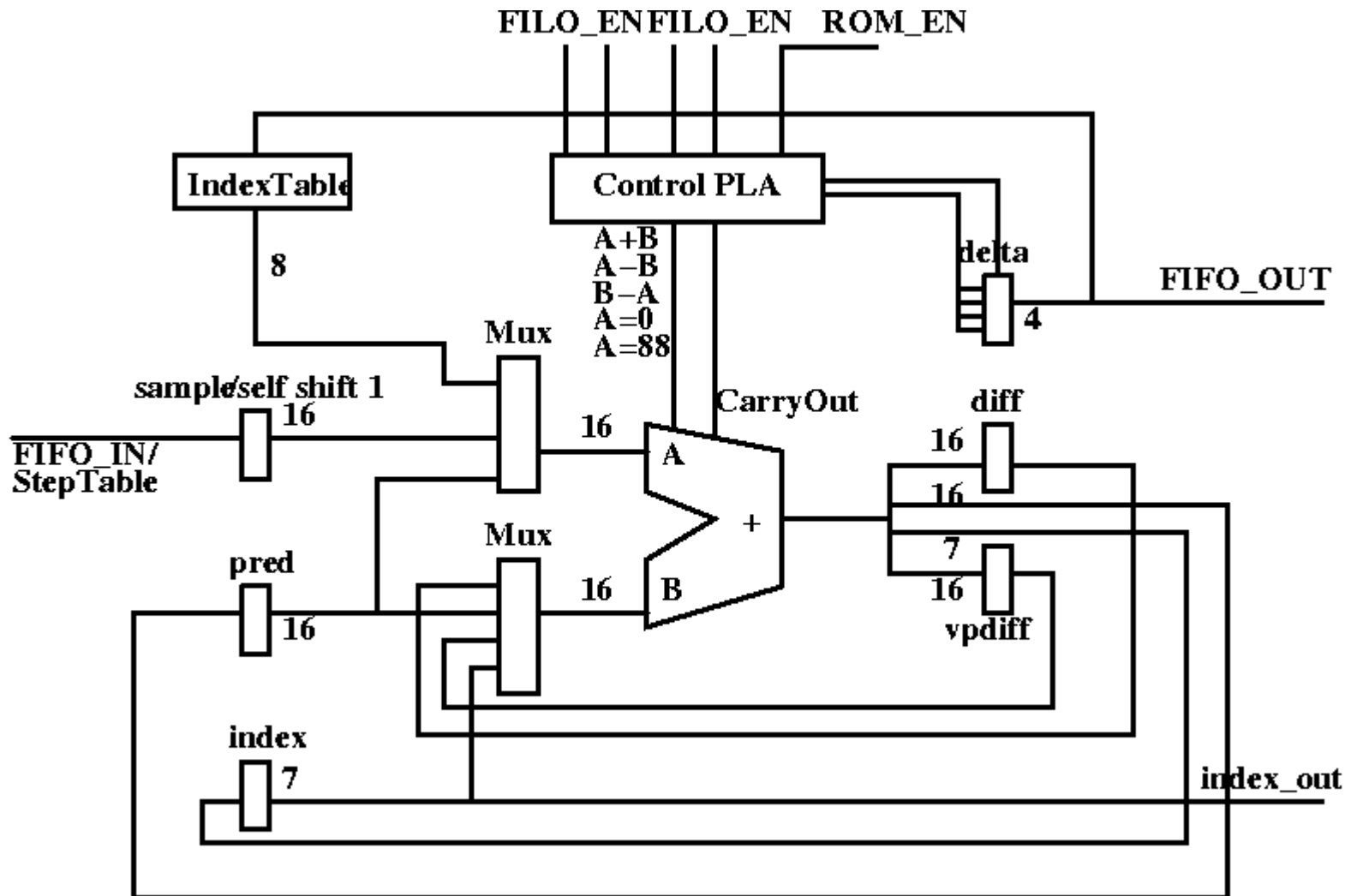
# ADPCM Algorithm

- ADPCM: Adaptive Differential Pulse Code Modulation
- Encode 16-bit linear PCM voice sample into 4-bit sample
- Compute difference between adjacent voice sample (16-bit) and mapping the difference to a constant lookup table entry.
- Store the entry index (4-bit)

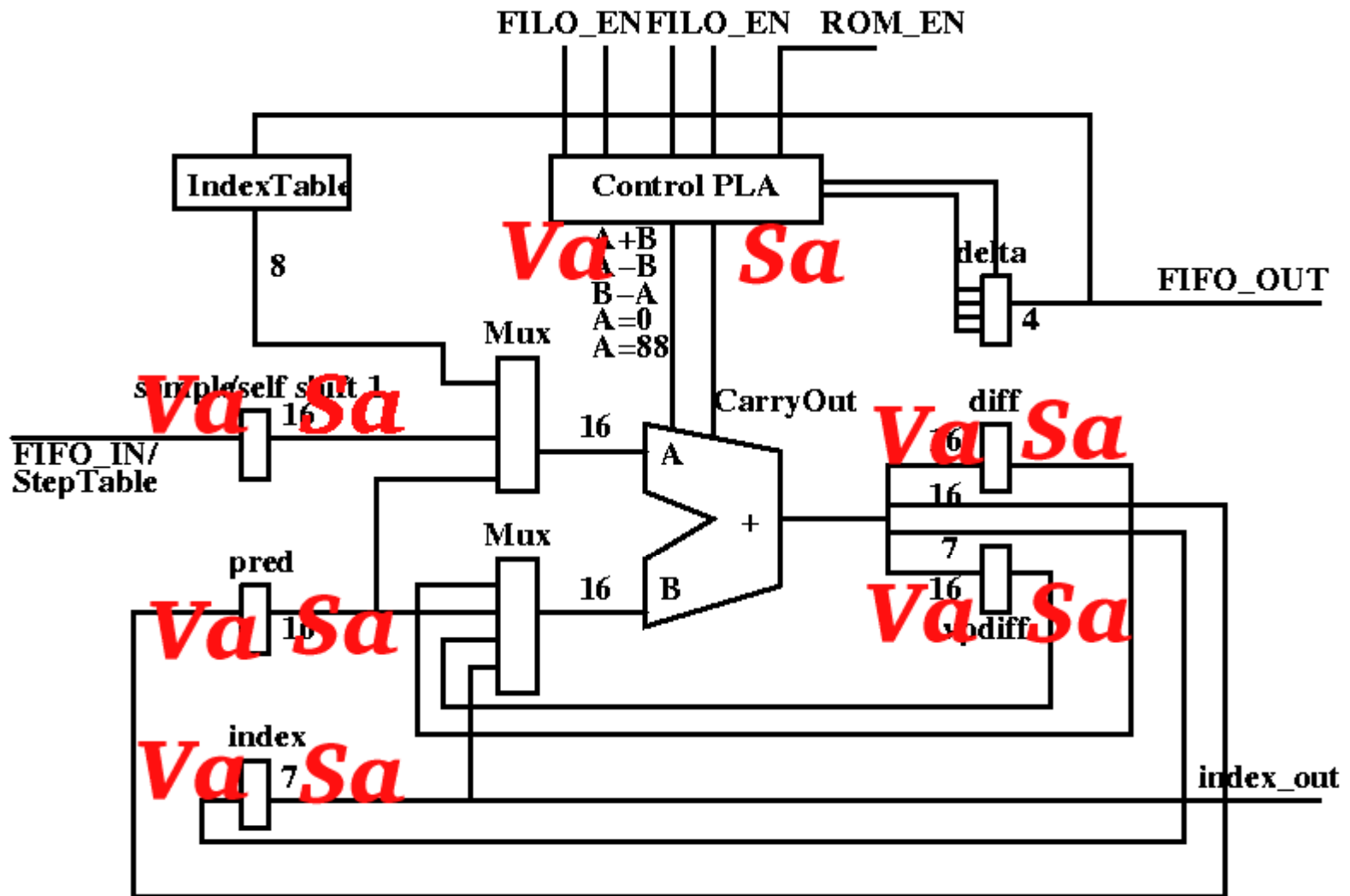
# The Gory Details

- 0) Initialize registers:  $\text{delta}=0$ ,  $\text{index}=0$ ,  $\text{pred}=0$ ,  $\text{step}=7$  (Reset)
- 1) Load data from input FIFO(external) to  $\text{vpdiff}$
- 2)  $\text{diff} = \text{vpdiff} - \text{pred}$ ;  $\text{delta}[3] = \text{Adder\_CarryOut}$ ;  $\text{diff} = |\text{diff}|$
- 3)  $\text{vpdiff} = \text{step} \gg 3$
- 4) If  $\text{diff} \geq \text{step}$  then {  $\text{delta}[2] = 1$ ,  $\text{diff} = \text{diff} - \text{step}$ ;  $\text{vpdiff} = \text{vpdiff} + \text{step}$  }
- 5)  $\text{step} = \text{step} \gg 1$
- 6) If  $\text{diff} \geq \text{step}$  then {  $\text{delta}[1] = 1$ ,  $\text{diff} = \text{diff} - \text{step}$ ;  $\text{vpdiff} = \text{vpdiff} + \text{step}$  }
- 7)  $\text{step} = \text{step} \gg 1$
- 8) If  $\text{diff} \geq \text{step}$  then {  $\text{delta}[0] = 1$ ,  $\text{diff} = \text{diff} - \text{step}$ ;  $\text{vpdiff} = \text{vpdiff} + \text{step}$  }
- 9) If  $\text{delta}[3] == 1$  then {  
     $\text{pred} = \text{pred} - \text{vpdiff}$   
} else {  
     $\text{pred} = \text{pred} + \text{vpdiff}$   
}
- 10) Clamp pred:  
If  $\text{pred} \leq -32767$ ,  $\text{pred} = -32767$ .  
If  $\text{pred} \geq 32767$ ,  $\text{pred} = 32767$
- 11)  $\text{index} = \text{index} + \text{indexTable}[\text{delta}]$
- 12) Clamp index.  
If  $\text{index} < 0$ ,  $\text{index} = 0$ .  
If  $\text{index} > 88$ ,  $\text{index} = 88$
- 13)  $\text{step} = \text{stepsizeTable}[\text{index}]$
- 14) Output delta to output FIFO
- 15) Goto Step 1)
- **StepTable carefully computed**

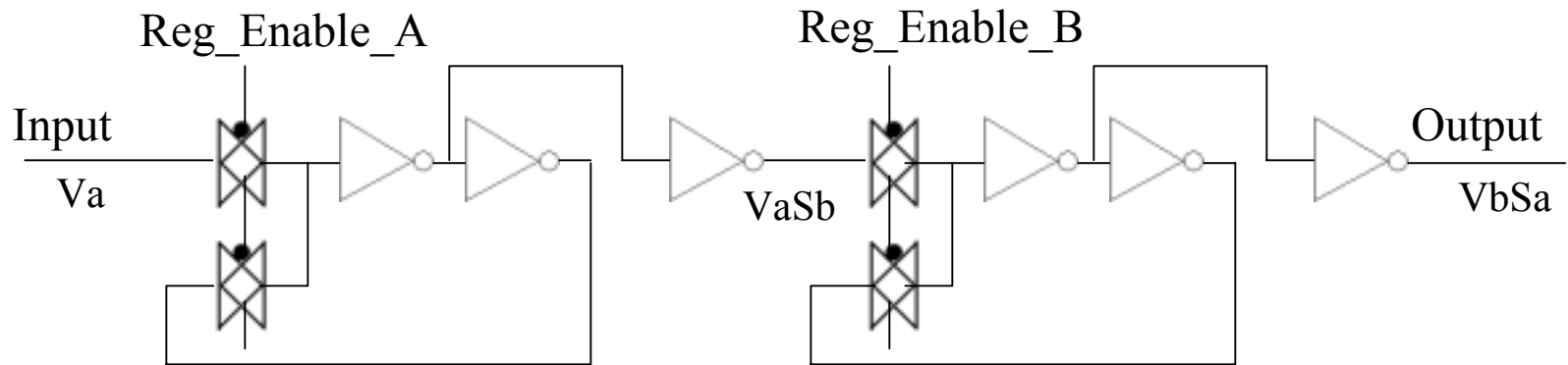
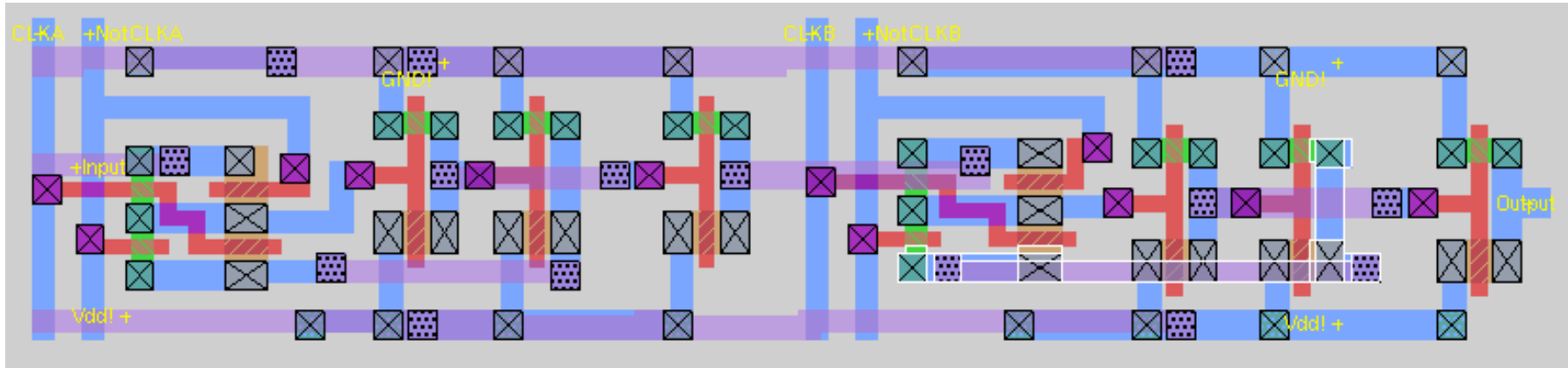
# Block Diagram



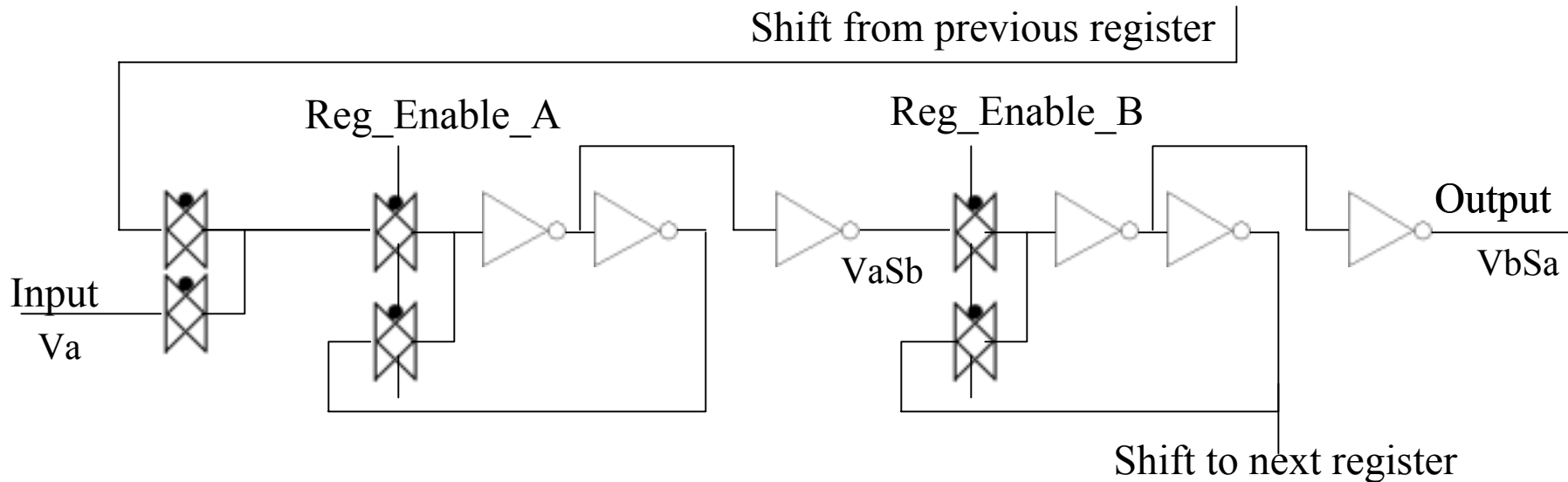
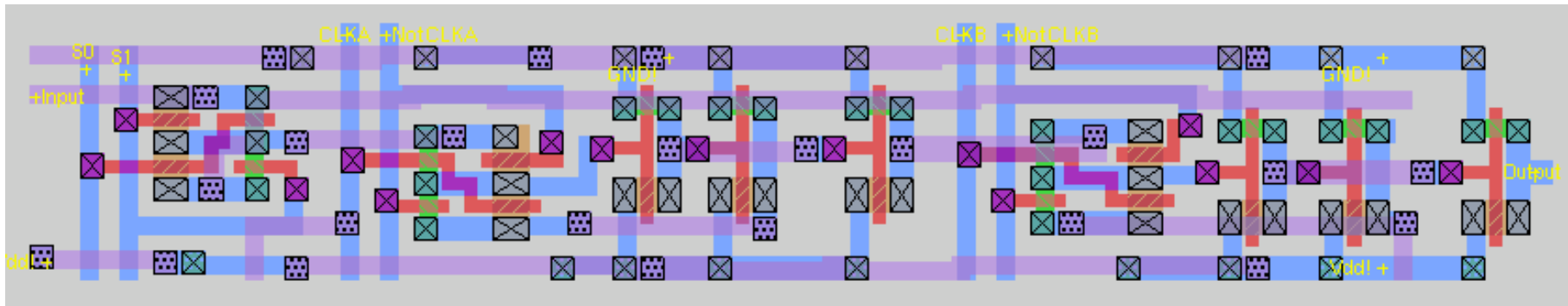
# Timing-Benefits of True Registers



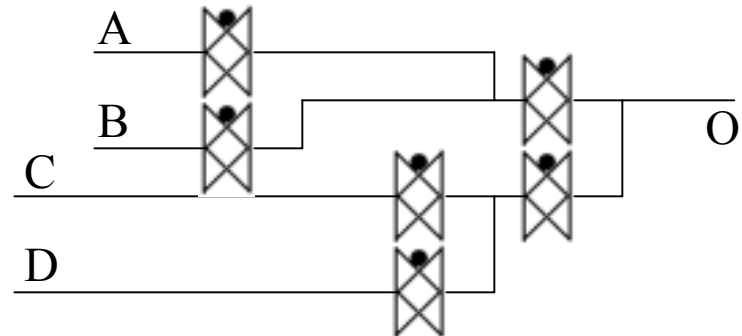
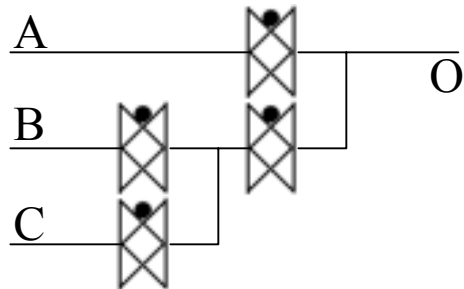
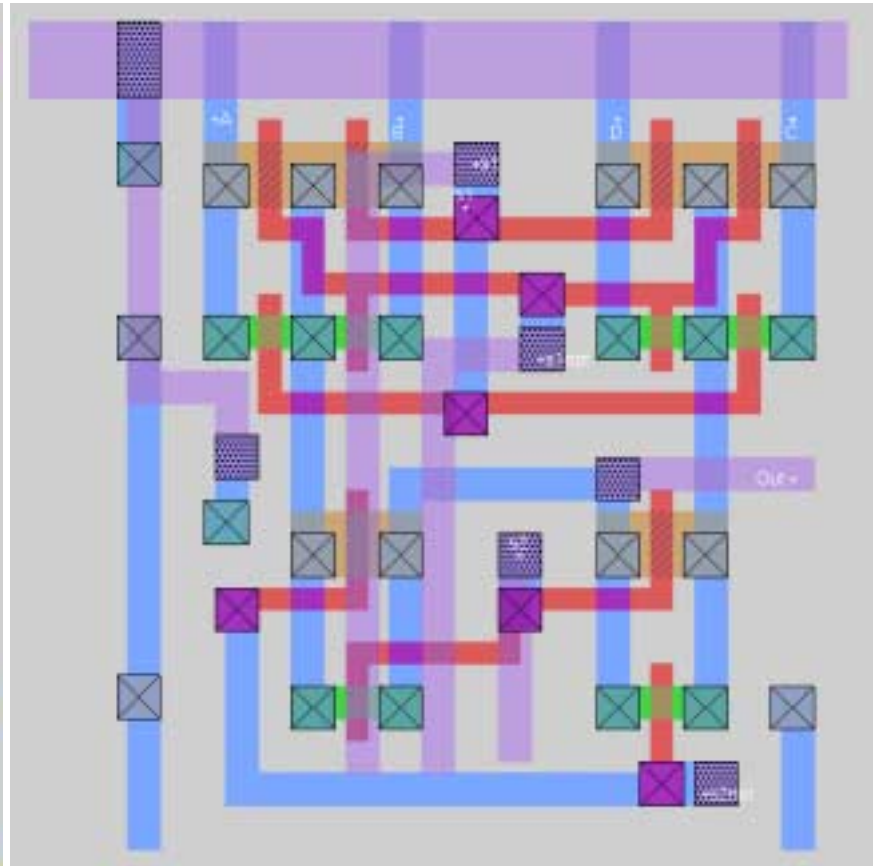
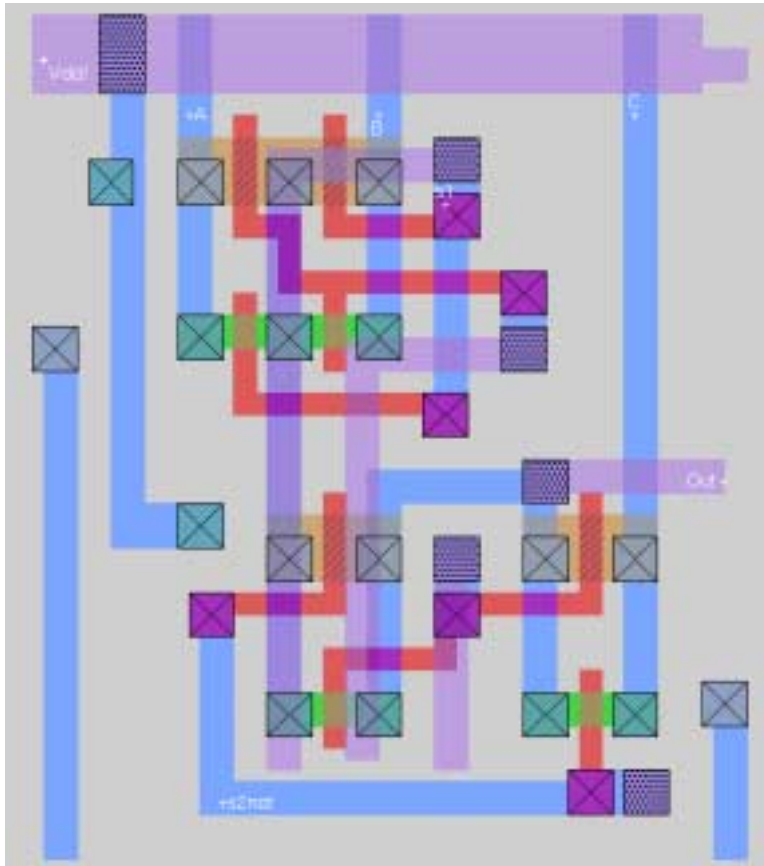
# Subcell: Normal Register



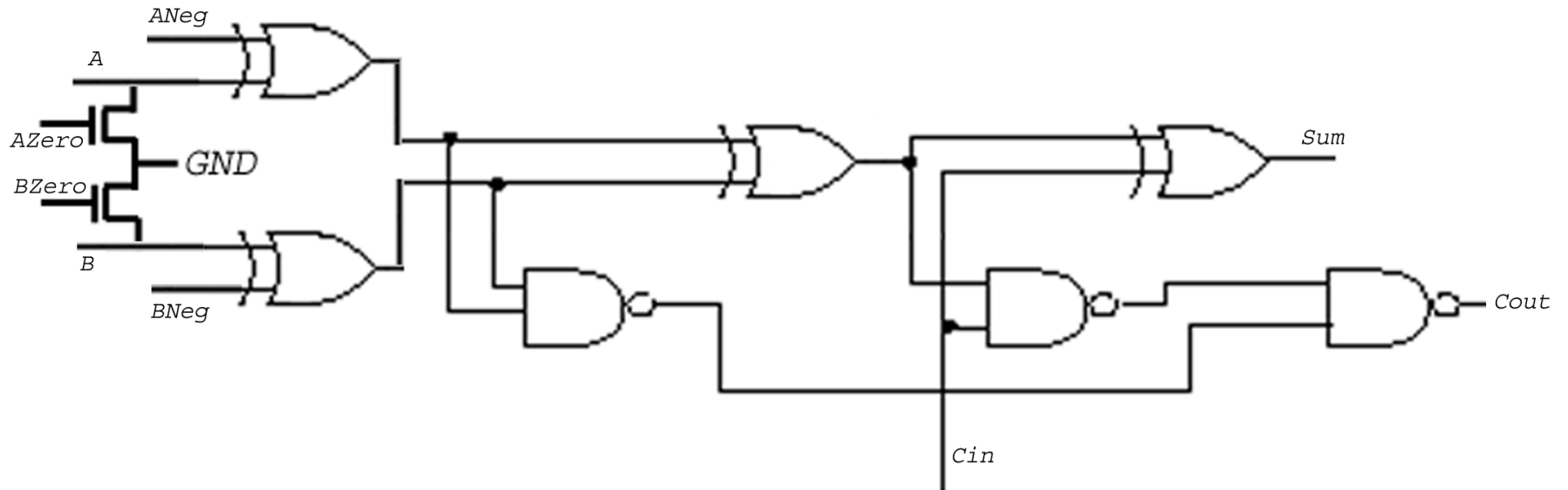
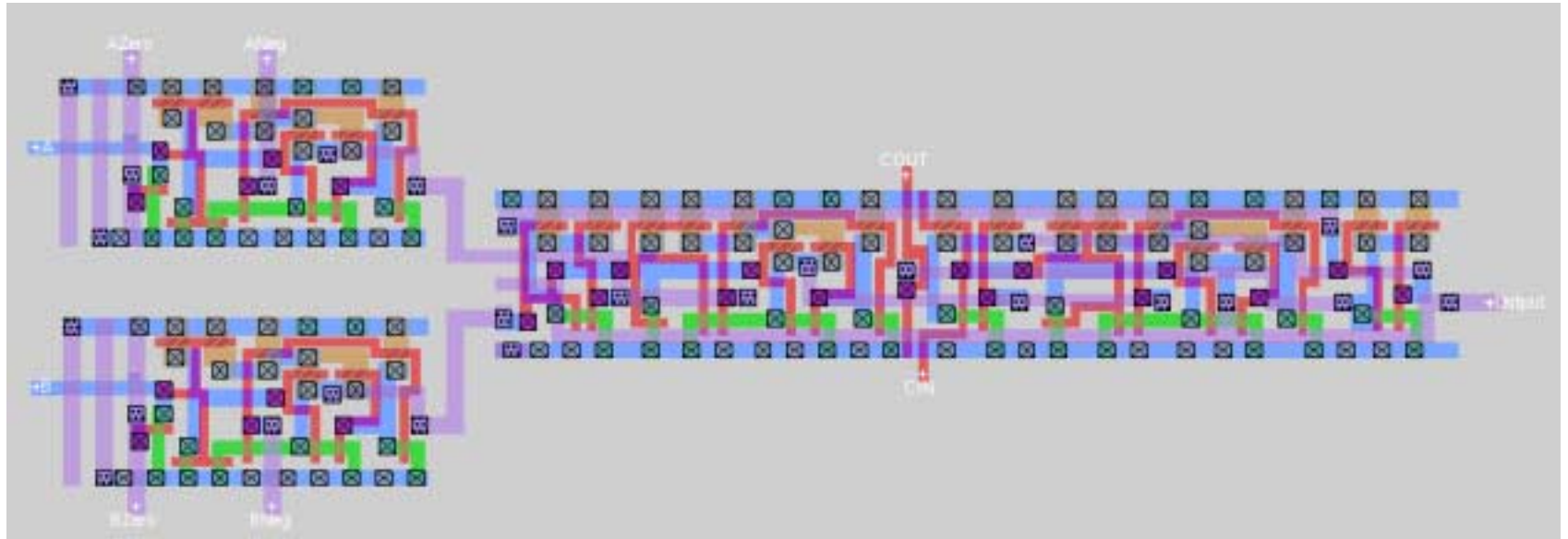
# Subcell: Right Self-shift Register



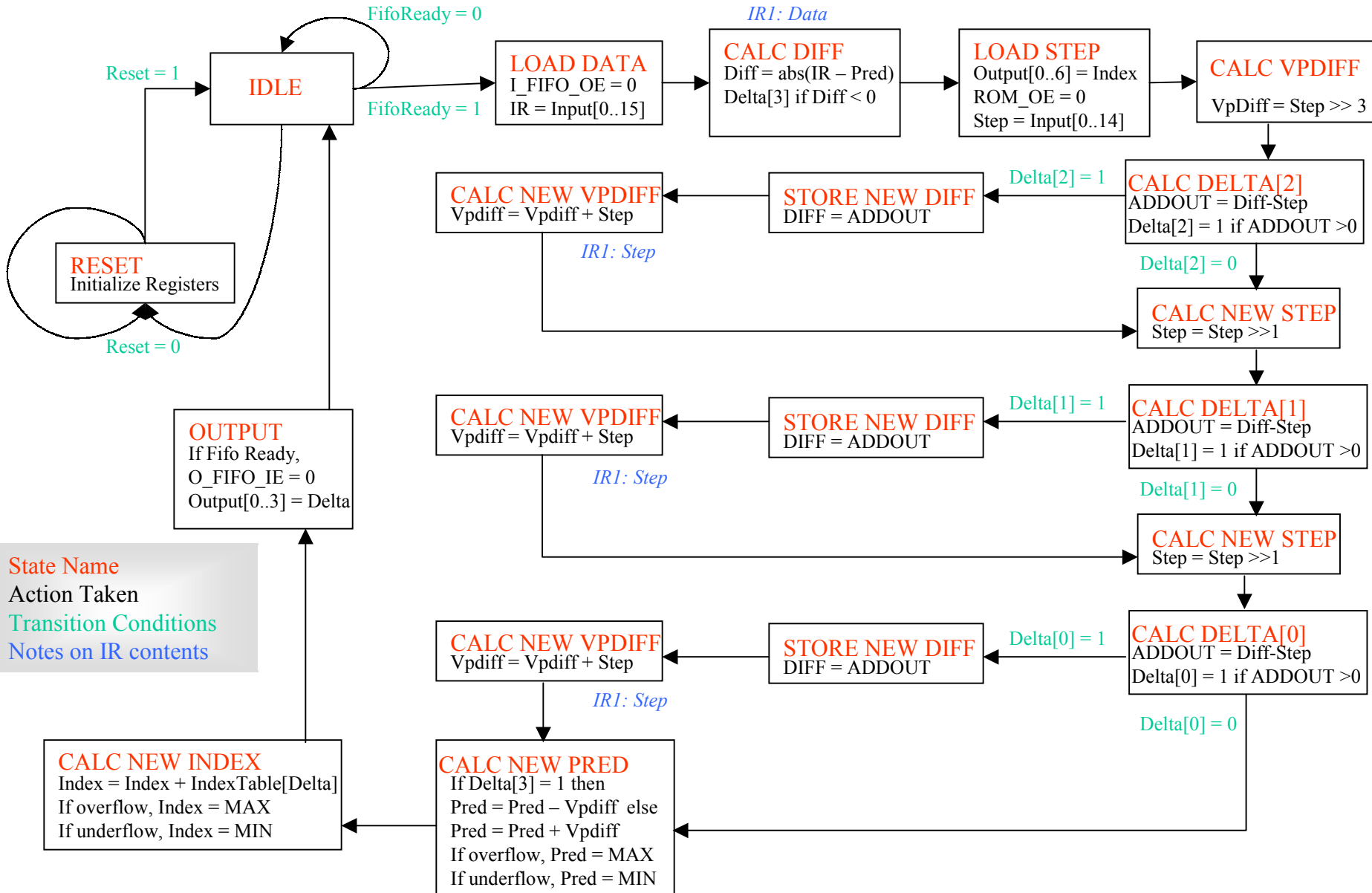
# 3 and 4 Input MUXEs

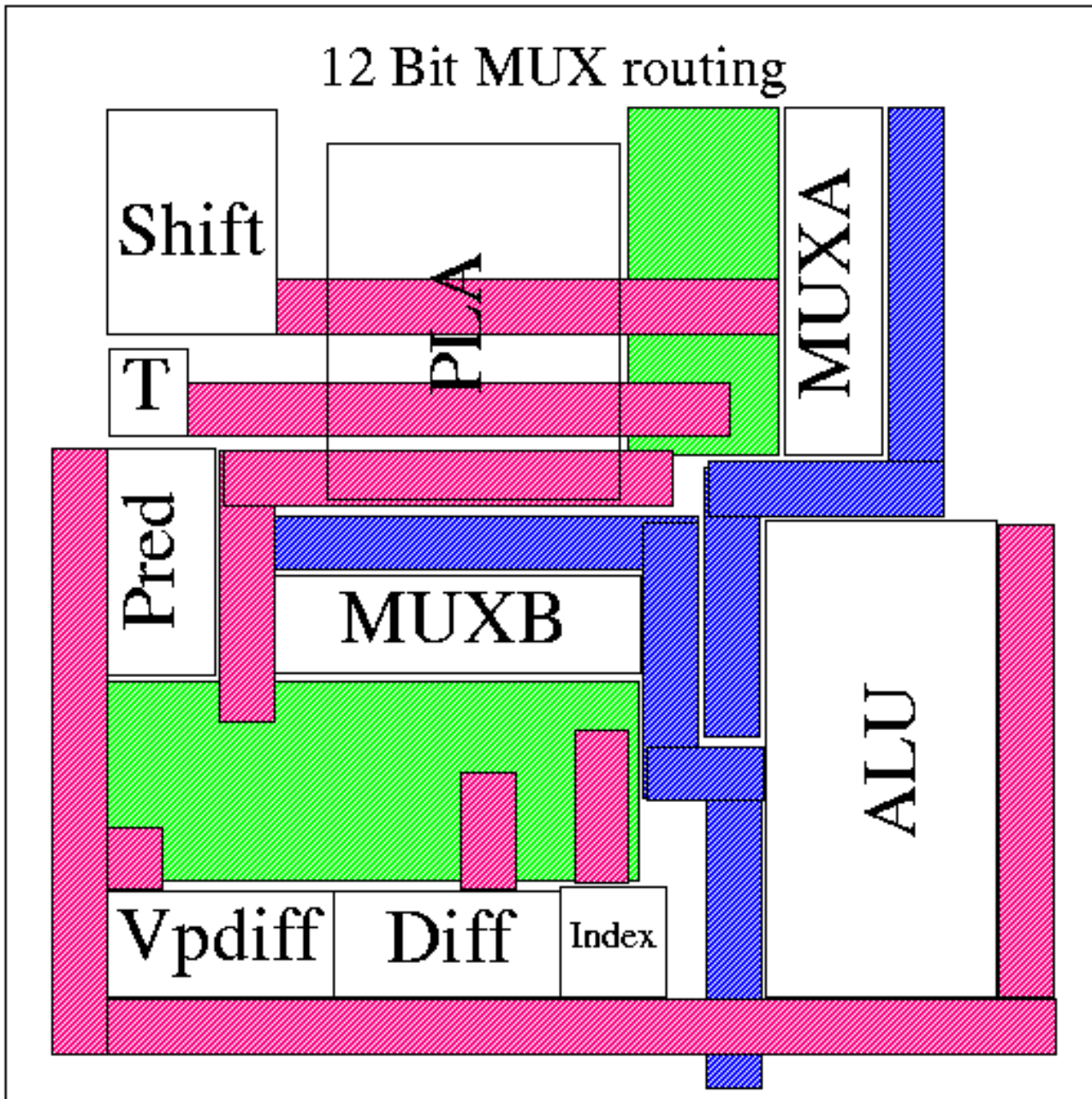


# Adder

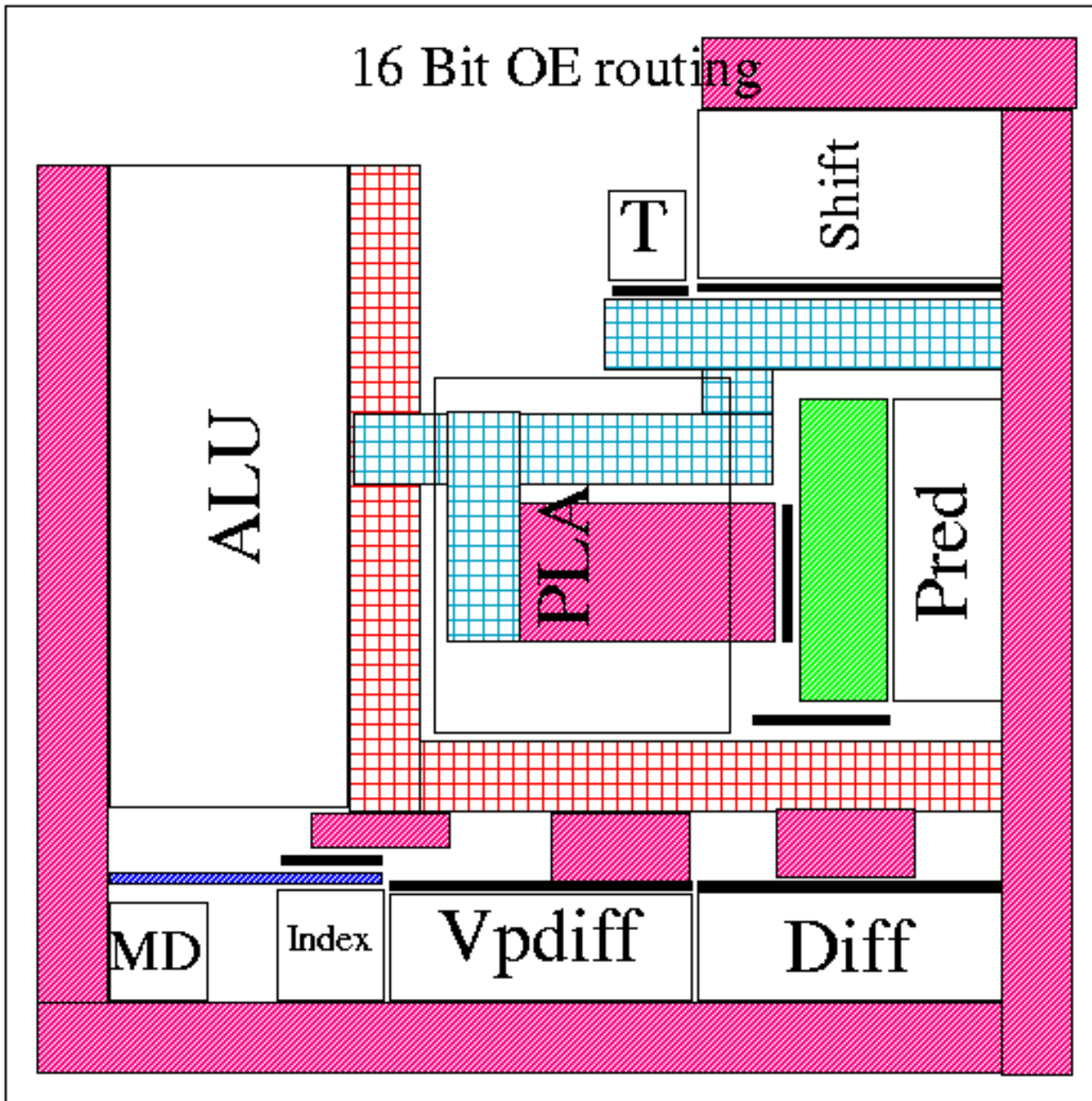


# State Machine





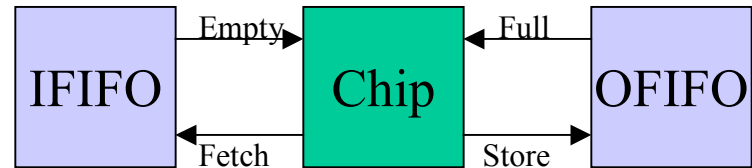
Picture to Scale. Bounding box is  $2200 \lambda$



Picture to Scale. Bounding box is  $2200 \lambda$

# Design Considerations

- Removal of Register
- I/O Design
- Pin Configuration
  - Much pin sharing involved
- ROM Table on/off chip
  - Size Constraints
- Scaling of the Algorithm
- Decoding was left off of the chip



# Summary

- 4:1 compression
- 16 bit ALU and dual bus routing
- Current testing suggest that it is fast enough for real time voice compression
- Companion chip can be made easily from current chip structure