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: delta=0, index=0, pred=0, step=7 (Reset)
• 1) Load data from input FIFO(external) to vpdiff
• 2) diff = vpdiff - pred; delta[3] = Adder_CarryOut; diff = |diff|
• 3) vpdiff = step >> 3
• 4) If diff >= step then { delta[2] = 1, diff = diff - step; vpdiff = vpdiff + step }
• 5) step = step >> 1
• 6) If diff >= step then { delta[1] = 1, diff = diff - step; vpdiff = vpdiff + step }
• 7) step = step >> 1
• 8) If diff >= step then { delta[0] = 1, diff = diff - step; vpdiff = vpdiff + step }
• 9) If delta[3]==1 then {
  pred = pred - vpdiff
} else {
  pred = pred + vpdiff
}
• 10) Clamp pred:
  If pred <= -32767, pred = -32767.
  If pred >= 32767, pred = 32767
• 11) index = index + indexTable[delta]
• 12) Clamp index.
  If index < 0, index = 0.
  If index > 88, index = 88
• 13) step = stepsizetable[index]
• 14) Output delta to output FIFO
• 15) Goto Step 1)

• StepTable carefully computed
Timing-Benefits of True Registers
Subcell: Normal Register

```
Reg_Enable_A
Input Va

Reg_Enable_B

VaSb

Output VbSa
```
Subcell: Right Self-shift Register

Shift from previous register

Input
Va

Reg_Enable_A

Output
VbSa

Shift to next register

Reg_Enable_B
3 and 4 Input MUXes
Adder
**State Machine**

**State Name**
- IDLE
- OUTPUT
- RESET

**Action Taken**
- LOAD DATA
- CALC DIFF
- LOAD STEP
- CALC VPDIFF
- STORE NEW DIFF
- CALC NEW VPDIFF
- CALC NEW INDEX
- CALC NEW PRED

**Transition Conditions**
- If FIFO Ready, O_FIFO_IE = 0
- Output[0..3] = Delta

**Notes on IR contents**
- IR1: Data
- IR1: Step
- IR1: Step

**State Diagram**
- **IDLE**
  - LOAD DATA
    - I_FIFO_OE = 0
    - IR = Input[0..15]
  - CALC DIFF
    - Diff = abs(IR – Pred)
    - Delta[3] if Diff < 0
  - LOAD STEP
    - Output[0..6] = Index
    - ROM_OE = 0
    - Step = Input[0..14]
  - CALC VPDIFF
    - VpDiff = Step >> 3
  - STORE NEW DIFF
    - DIFF = ADDOUT
  - CALC NEW VPDIFF
    - Vpdiff = Vpdiff + Step
  - CALC NEW STEP
    - Step = Step >>1
  - CALC NEW INDEX
    - Index = Index + IndexTable[Delta]

- **OUTPUT**
  - If FIFO Ready, O_FIFO_IE = 0
  - Output[0..3] = Delta

- **RESET**
  - Initialize Registers
  - Reset = 1
  - FIFOReady = 0

- **FifoReady = 1**
  - LOAD DATA
  - CALC DIFF
  - LOAD STEP
  - CALC VPDIFF
  - STORE NEW DIFF
  - CALC NEW VPDIFF
  - CALC NEW STEP
  - CALC NEW INDEX

- **FifoReady = 0**
  - LOAD DATA
  - CALC DIFF
  - LOAD STEP
  - CALC VPDIFF
  - STORE NEW DIFF
  - CALC NEW VPDIFF
  - CALC NEW STEP
  - CALC NEW INDEX

**ADDOUT**
- Delta[2] = 1
- ADDOUT = Diff-Step
- Delta[2] = 1 if ADDOUT >0
- Delta[2] = 0

**ADDOUT**
- Delta[1] = 1
- ADDOUT = Diff-Step
- Delta[1] = 1 if ADDOUT >0
- Delta[1] = 0

**ADDOUT**
- Delta[0] = 1
- ADDOUT = Diff-Step
- Delta[0] = 1 if ADDOUT >0
- Delta[0] = 0

**ADDOUT**
- Delta[3] = 1
- ADDOUT = Diff-Step
- Pred = Pred – Vpdiff
- Pred = Pred + Vpdiff
- If overflow, Pred = MAX
- If underflow, Pred = MIN

**ADDOUT**
- Delta[2] = 0
- ADDOUT = Diff-Step
- Pred = Pred – Vpdiff
- Pred = Pred + Vpdiff
- If overflow, Pred = MAX
- If underflow, Pred = MIN

**ADDOUT**
- Delta[1] = 0
- ADDOUT = Diff-Step
- Pred = Pred – Vpdiff
- Pred = Pred + Vpdiff
- If overflow, Pred = MAX
- If underflow, Pred = MIN

**ADDOUT**
- Delta[0] = 0
- ADDOUT = Diff-Step
- Pred = Pred – Vpdiff
- Pred = Pred + Vpdiff
- If overflow, Pred = MAX
- If underflow, Pred = MIN
Picture to Scale. Bounding box is 2200 λ
16 Bit OE routing

ALU

 PLA

 T

 Shift

 Pred

MD

Index

Vpdiff

Diff

Picture to Scale. Bounding box is 2200 λ
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