A 2014 Mbin/s Deeply Pipelined CABAC Decoder for HEVC

Yu-Hsin Chen, Vivienne Sze
Massachusetts Institute of Technology
Next-Generation Video Coding

**Ultra-HD (UHD) comes into play**

- 4K Streaming
- 4K Broadcasting

**Moving toward Mobile**

- 4K Recording
- S/W Support
Next-Generation Video Coding

Ultra-HD (UHD) comes into play

- 4K Streaming
- 4K Broadcasting

Moving toward Mobile

- 4K Recording
- S/W Support

High Efficiency Video Coding (HEVC)

- 2× higher coding efficiency than H.264/AVC
- High Throughput → 8K UHD @ 120 fps
- Low Power → prolonged battery life
HEVC Decoder

Encoded Bitstream

1001011

Entropy Decoding

Syntax Elements: prediction modes, motion vectors, etc.

Motion Compensation

Intra-Frame Prediction

Inv. Transform & Quantization

Filtering & Reconstruction

Decoded Video

Easily Parallelizable Pixel Operations
HEVC Throughput Bottleneck – CABAC

Context Adaptive Binary Arithmetic Coding (CABAC)*

- **Highly serial processing** → low CABAC throughput
- **Limits** decoder **throughput**
- **Limits** voltage scaling of the decoder for **low power**

* [D. Marpe, IEEE TCSVT, July 2003]
HEVC Throughput Bottleneck – CABAC

Context Adaptive Binary Arithmetic Coding (CABAC)*

- Highly serial processing → low CABAC throughput
- Limits decoder throughput
- Limits voltage scaling of the decoder for low power

High-throughput CABAC decoding is the key to HEVC decoder performance

* [D. Marpe, IEEE TCSVT, July 2003]
CABAC Coding Basics

- **Task**: To code two syntax elements $AB$ with CABAC

<table>
<thead>
<tr>
<th>Syntax Element A</th>
<th>Context Model $P_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messages</td>
<td></td>
</tr>
<tr>
<td>$a_1$</td>
<td>50%</td>
</tr>
<tr>
<td>$a_2$</td>
<td>30%</td>
</tr>
<tr>
<td>$a_3$</td>
<td>15%</td>
</tr>
<tr>
<td>$a_4$</td>
<td>5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax Element B</th>
<th>Context Model $P_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messages</td>
<td></td>
</tr>
<tr>
<td>$b_1$</td>
<td>25%</td>
</tr>
<tr>
<td>$b_2$</td>
<td>45%</td>
</tr>
<tr>
<td>$b_3$</td>
<td>30%</td>
</tr>
</tbody>
</table>
CABAC Coding Basics

➢ Task: To code two syntax elements $AB$ with CABAC

<table>
<thead>
<tr>
<th>Syntax Element $A$</th>
<th>Context Model $P_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>50%</td>
</tr>
<tr>
<td>$a_2$</td>
<td>30%</td>
</tr>
<tr>
<td>$a_3$</td>
<td>15%</td>
</tr>
<tr>
<td>$a_4$</td>
<td>5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax Element $B$</th>
<th>Context Model $P_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td>25%</td>
</tr>
<tr>
<td>$b_2$</td>
<td>45%</td>
</tr>
<tr>
<td>$b_3$</td>
<td>30%</td>
</tr>
</tbody>
</table>

Variable-Length Code: Use less bits to code higher-probability messages

syntax elements

bitstream

syntax elements

high prob. message: $a_1b_2$
low prob. message: $a_4b_1$
CABAC Coding Basics

**Task:** To code two syntax elements AB with CABAC

<table>
<thead>
<tr>
<th>Syntax Element A</th>
<th>Context Model $P_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>50%</td>
</tr>
<tr>
<td>$a_2$</td>
<td>30%</td>
</tr>
<tr>
<td>$a_3$</td>
<td>15%</td>
</tr>
<tr>
<td>$a_4$</td>
<td>5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax Element B</th>
<th>Context Model $P_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td>25%</td>
</tr>
<tr>
<td>$b_2$</td>
<td>45%</td>
</tr>
<tr>
<td>$b_3$</td>
<td>30%</td>
</tr>
</tbody>
</table>

**Variable-Length Code:** Use less bits to code higher-probability messages

Syntax elements:
- high prob. message: $a_1b_2$
- low prob. message: $a_4b_1$

Bitstream:
- 10
- 110110

Syntax elements:
- $a_1b_2$
- $a_4b_1$
CABAC Coding Basics

➢ Task: To code two syntax elements \( AB \) with CABAC

<table>
<thead>
<tr>
<th>Syntax Element ( A )</th>
<th>Context Model ( P_A )</th>
<th>Syntax Element ( B )</th>
<th>Context Model ( P_B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>50% ( \rightarrow ) 30%</td>
<td>( b_1 )</td>
<td>25%</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>30% ( \rightarrow ) 50%</td>
<td>( b_2 )</td>
<td>45%</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>15%</td>
<td>( b_3 )</td>
<td>30%</td>
</tr>
<tr>
<td>( a_4 )</td>
<td>5%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Variable-Length Code: Use less bits to code higher-probability messages

CABAC is **adaptive** to changing context models

```
syntax elements
a_1b_1
a_2b_1
```

```
bitstream
10  \( \rightarrow \) 110
110  \( \rightarrow \) 10
```
CABAC Coding Basics

➢ **Task**: To code two syntax elements **AB** with CABAC

<table>
<thead>
<tr>
<th>Syntax Element A</th>
<th>Context Model ( P_A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>50% ( \rightarrow ) 30%</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>30% ( \rightarrow ) 50%</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>15%</td>
</tr>
<tr>
<td>( a_4 )</td>
<td>5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax Element B</th>
<th>Context Model ( P_B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( b_1 )</td>
<td>25%</td>
</tr>
<tr>
<td>( b_2 )</td>
<td>45%</td>
</tr>
<tr>
<td>( b_3 )</td>
<td>30%</td>
</tr>
</tbody>
</table>

Variable-Length Code: Use less bits to code higher-probability messages

CABAC is **adaptive** to changing context models

CABAC processing is highly **effective** but also highly **adaptive**
Key Feedback Loops in CABAC Decoding

CABAC Decoder

- Context Modeling
  - Context Selection (CS)
  - Context Memory (CM)
- Decoding state
- Updated prob(.)
- prob(.)
- mvd = 5
- Decoded syntax elements
- Binary symbols (bin)
- Bitstream 10
- 1001011

- Video data path
- Control data path
- Feedback loops

- Arithmetic Decoder (AD)
- De-Binarization (DB)
Key Feedback Loops in CABAC Decoding

CABAC Decoder

Context Modeling

Context Selection (CS)

Context Memory (CM)

De-Binarization (DB)

Arithmetic Decoder (AD)

decoded syntax elements

mvd = 5

video data path

control data path

feedback loops

1001011

bitstream 10

prob(.)

updated prob(.)

decoding state

binary symbols (bin)

prob(.)

decoded video data path

control data path

feedback loops

bitstream 10
Key Feedback Loops in CABAC Decoding

- Adaptive context for accurate probability estimation (loop 1)

CABAC Decoder

- Context Modeling
  - Context Selection (CS)
  - Context Memory (CM)
- De-Binarization (DB)
- Arithmetic Decoder (AD)
- video data path
- control data path
- feedback loops

mvd = 5

decoding state
updated prob(.)
prob(.)

decoded syntax elements
binary symbols (bin)

bitstream 10

1001011
Key Feedback Loops in CABAC Decoding

- Adaptive context for accurate probability estimation (loop ①)
- Variable-length code contributes to bin-to-bin dependency (loop ②)
Key Feedback Loops in CABAC Decoding

CABAC Decoder

- Adaptive context for accurate probability estimation (loop ①)
- Variable-length code contributes to bin-to-bin dependency (loop ②)

Feedback loops restrict the parallelism of CABAC
Key Feedback Loops in CABAC Decoding

CABAC Decoder

- Context Modeling
  - Context Selection (CS)
  - Context Memory (CM)
- De-Binarization (DB)
- Arithmetic Decoder (AD)

Decoding of bins
- **Context-coded Bins:** Require a context model for each bin
- **Bypass Bins:** Do not require context modeling

mvd = 5

1001011

10
Proposed HEVC CABAC Decoder Architecture

- **Goal**: to achieve the **highest throughput possible** in bin/sec
Goal: to achieve the **highest throughput possible** in bin/sec
Proposed HEVC CABAC Decoder Architecture

- **Goal**: to achieve the **highest throughput possible** in bin/sec
Proposed HEVC CABAC Decoder Architecture

• **Goal**: to achieve the highest throughput possible in bin/sec
Proposed HEVC CABAC Decoder Architecture

**Goal:** to achieve the highest throughput possible in bin/sec

- bin/sec = \( \text{bin/cycle} \times \text{cycle/sec} \)
- increase the clock rate
- decode more bins per cycle
Feature 1: Deeply Pipelined Architecture

- Longest timing path in CABAC (w/o pipelining)
Feature 1: Deeply Pipelined Architecture

- Longest timing path in CABAC (w/o pipelining)

- A popular design: 2-stage pipelining

[Lin, ISCAS, 2009] [Liao, TCSVT, 2012]
Feature 1: Deeply Pipelined Architecture

- Longest timing path in CABAC (w/o pipelining)

- Proposed deeply pipelined architecture: 5-stage pipelining

[synthesized in 45nm SOI, only combinational logic included]

At least **2.2× higher clock rate** than 2-stage pipelined design
Feature 1: Deeply Pipelined Architecture

- Feedback loops introduce **Stalls**, reduce **bin/cycle**

**Stalls needs to be removed** to take advantage of pipelining
Feature 1: Pipelining with State Prefetch Logic

- FSM prefetches **next 2 possible states** based on the current bin binary value
- Optimized between **number of stalls** and **number of states**

**FSM Binary Tree:**

```
  ...            1   ...            0   ...            1   ...            0
  ...            1   State2_{11}    1   State1_{1}    1   State0
  ...            0   State2_{10}    0   State1_{0}    0   
  ...            1   State2_{01}    1   
  ...            0   State2_{00}    0
```

**Decoded Bins**

1 → 0 → 0 → ...

**State Prefetch Logic** removes majority of the stalls from pipelining
Feature 2: Latch-Based Context Memory

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>Compact</td>
<td>No Multi-R/W</td>
</tr>
<tr>
<td>Registers</td>
<td>Fast</td>
<td>Power/Area Hungry</td>
</tr>
<tr>
<td>Latches</td>
<td>Compact, Low Power</td>
<td><strong>Timing Issue</strong></td>
</tr>
</tbody>
</table>

2R1W per cycle → SRAM not suitable
Feature 2: Latch-Based Context Memory

Memory Type | Pros | Cons | Eq. Gate Count | Avg. Power (mW)*
---|---|---|---|---
SRAM | Compact | No Multi-R/W | 11.7k | 8.10
Registers | Fast | Power/Area Hungry | 20.2k | 13.10
Latches | Compact, Low Power | Timing Issue | 13.4k | 4.68

* at 2 GHz clock rate

Latch-based CM supports **multiple R/W with low power consumption**

Solution:

- Enable Decoder
- ED_N
- EN
- Latch N
- Clock
- wAddr
- wData
- D
- Q

2R1W per cycle \(\rightarrow\) SRAM not suitable

glitch-free protection
Feature 3: Separate FSM for Bypass Bins

- Bypass bins do **not** require context modeling
- HEVC groups bypass bins together in coding

```
C B C B C B B
```
Feature 3: Separate FSM for Bypass Bins

Separate FSM reduces the pipeline depth for bypass bins to avoid stalls
Feature 4: Multi-Bypass-Bin Decoding

- **High bitrate** bitstream tends to have **more grouped bypass bins**
- Decoding of bypass bins is fast
Feature 4: Multi-Bypass-Bin Decoding

Multi-bypass-bin decoding increases bin/cycle at the same clock rate
### Feature Summary

Throughput = $\text{bin/cycle} \times \text{cycle/sec}$

<table>
<thead>
<tr>
<th>Increase $\text{cycle/sec}$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>5-Stage Deep Pipelining</strong></td>
<td>2.2× faster than 2-stage design</td>
</tr>
<tr>
<td><strong>State Prefetch Logic</strong></td>
<td>reduce impact of stalls down to 12%</td>
</tr>
<tr>
<td><strong>Latch-based Memory</strong></td>
<td>multiple R/W at lower power</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Increase $\text{bin/cycle}$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Separate FSM</strong></td>
<td>Increase throughput by up to 33%</td>
</tr>
<tr>
<td><strong>2 Bypass Bins / Cycle</strong></td>
<td>Increase throughput by up to 15%</td>
</tr>
</tbody>
</table>

* All tested with common test bitstreams
Performance Evaluation

• High bitrate bitstream → More bypass bins → High throughput
• Reaches up to **1.06 bin/cycle** for sequence at 400 Mbps

![Graph showing the relationship between percentage of bypass bins and average bins/cycle for different bitrates. Each dot represents a single bitstream. The graph shows a trend where CABAC performance increases with higher bitrate sequences.](image)
Result Comparison

- Throughput up to **2000 Mbin/sec** at **1.9 GHz** clock rate
- **Real-time** decoding of **8K UHD @ 120 fps** bitstreams
- Can trade-off throughput for **low power** (voltage scaling)
Result Comparison

- Throughput up to **2000 Mbin/sec** at **1.9 GHz** clock rate
- **Real-time** decoding of **8K UHD @ 120 fps** bitstreams
- Can trade-off throughput for **low power** (voltage scaling)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard</strong></td>
<td>AVC</td>
<td>AVC</td>
<td>HEVC</td>
<td>HEVC</td>
</tr>
<tr>
<td><strong>Tech.</strong></td>
<td>UMC 90nm</td>
<td>UMC 90nm</td>
<td>Samsung 28nm</td>
<td>IBM 45nm SOI</td>
</tr>
<tr>
<td><strong>Gate Count</strong></td>
<td>82.4k</td>
<td>51.3k</td>
<td>100.4k&lt;sup&gt;1&lt;/sup&gt;</td>
<td>85.3k</td>
</tr>
<tr>
<td><strong>SRAM Size</strong></td>
<td>N/A</td>
<td>179B</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Max. Freq.</strong></td>
<td>222</td>
<td>264</td>
<td>333</td>
<td>1900</td>
</tr>
<tr>
<td><strong>(MHz)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bins/Cycle</strong></td>
<td>1.96</td>
<td>1.84&lt;sup&gt;2&lt;/sup&gt;</td>
<td>1.30</td>
<td>1.06&lt;sup&gt;3&lt;/sup&gt;</td>
</tr>
<tr>
<td><strong>Throughput</strong></td>
<td>435</td>
<td>486</td>
<td>433</td>
<td>2014</td>
</tr>
<tr>
<td><strong>(Mbin/s)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<sup>1</sup> without the bitstream parser buffer
<sup>2</sup> with the test bitstream bit-rate at 130 Mbps
<sup>3</sup> with the test bitstream bit-rate at 403 Mbps

Take-Home Messages

• A high-throughput CABAC decoder
  – *Deeply pipelined* (increase *cycle/sec*)
  – *Multi-bypass-bin decoding* (increase *bin/cycle*)

• **Throughput up to 2000 Mbin/s**
  – Real-time decoding of 8K UHD @ 120 fps

• **Trade-off throughput for low power**
  – Voltage scaling can be applied to the entire HEVC decoder
Thank you!

Contact: yhchen@mit.edu
Backup Materials
State Prefetch Logic

Case 1: with State prefetch
- Binary tree expanded
- No stalls left

Case 2: w/o State prefetch
- Stalls are kept
- Save dozens of states
Pipelining vs. Multi-Bin Decoding

- Deep multi-bin decoding suffers from higher complexity & reduced clock rate
- This work: increase the clock rate for all bins and further speed up processing of bypass bins

The Design Space

- cycle/sec
- higher throughput
- bin/cycle
The Design Space

Pipelining

Clock Frequency (MHz)

Average Bin/Cycle

This Work (HEVC)

Works using 45nm Process or below

Works using 90nm Process

Works using 130nm Process

Works using 180nm Process

line of equi-throughput

2000 MBin/s

1000 MBin/s

500 MBin/s

250 MBin/s

100 MBin/s

Multi-bin decoding