A HIGH THROUGHPUT CABAC ALGORITHM USING SYNTAX ELEMENT PARTITIONING

Vivienne Sze
Anantha P. Chandrakasan
2009 ICIP – Cairo, Egypt
Motivation

• **High demand** for video on mobile devices
• **Compression** to reduce storage and transmission
• **Battery capacity limited** by size, weight, and cost
• **Need low power video coding**
• **Achieve performance required for real time HD**
Low Power Video Coding

- Parallelism and voltage scaling shown to be effective in power reduction \( \Rightarrow \ > 10x \) power reduction
- However, certain algorithms inherently serial
  - E.g. Context Adaptive Binary Arithmetic Coding (CABAC)
- H.264/AVC High Profile uses CABAC for entropy coding
Arithmetic Coding

Example: Pr(A) = 0.6; Pr(B) = 0.4

Entropy Encoding Symbol Sequence: “A-B-B”

<table>
<thead>
<tr>
<th>Range</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.24</td>
<td>0.36</td>
</tr>
</tbody>
</table>

Output Binary Bitstream: .1001
(Binary Fraction)

- **Binary** Arithmetic Coding has binary symbols (‘bins’)
- Binarizer maps syntax elements to bins
- Range updated after every bin
Context-Adaptive

- Context (probability model)
- Adaptive estimation of probability (update context state)
- Context can be switched and updated *every* bin

Bin-to-bin dependencies $\rightarrow$ Cycle-to-cycle dependencies

**Diagram:**
- Syntax elements
- Binarizer
- BINS
- Arithmetic Coding Engine
- BITS: 0010...
- Pr(0) Pr(1)
- Context Modeler
- Probability Adaptation
- Context Selection

**Encoder:** Syntax Element $\rightarrow$ Bins $\rightarrow$ Bits
CABAC Challenges

**Decoder: Bits → Bins → Syntax Element**

- **Data Dependencies** (difficult to parallelize)
  - Contexts and Range are updated after every bin
  - At decoder, data feedback required
- **Context modeling and interval division tied to bins (not bits)**
  - Number of cycles proportional to number of bins
## Real-time H.264 CABAC Requirements

<table>
<thead>
<tr>
<th>Level</th>
<th>Max Frame Rate</th>
<th>Max Bins per Picture</th>
<th>Max Bit Rate</th>
<th>Peak Bin Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>fps</td>
<td>Mbins</td>
<td>25</td>
<td>275</td>
</tr>
<tr>
<td>5.1</td>
<td>26.7</td>
<td>17.6</td>
<td>300</td>
<td>2107</td>
</tr>
</tbody>
</table>

Max Bin Rate = (Max Bins per Picture) x (Max Frame Rate)

- For **real-time** decoding, decode frame within inter-frame time interval
- Frequency requirements reach **multi-GHz range**

---

*Parallelism needed to lower frequency to acceptable range*
H.264/AVC CABAC Parallelism

• Bin
  – Speculation required

• Frame
  – Buffering required for frames
  – Limited by latency requirement

• Slice
  – Coding Efficiency Penalty

Can we do better by changing the algorithm?
Entropy Slices

- Proposed by Sharp in 2008 [VCEG-AI32]
- Only entropy coding is independent
- Coding penalty overhead due to reduced training

Can we further reduce coding penalty?
Syntax Element Parallelism

- Place syntax elements in different groups
- Assign groups to different partitions and process partitions in parallel
- Allocation of syntax elements to partitions based on distribution (balance workload)

*E.g. Average distribution of bins (720p sequences QP=27)*

- 33% Macroblock Info
- 22% Prediction Mode
- 16% Coded Block Pattern
- 12% Significance Map
- 17% Coefficient Level
Reduce Cycle Count

H.264/AVC Slice

Syntax Element Partitions

MBINFO
PRED
CBP
SIGMAP
COEFF

LEGEND

Slice header
Start code

different syntax elements groups
macroblock

Cycles
Context Training for Coding Efficiency

- Coding efficiency depends on accuracy of bin probability estimate.
- Better estimate achieved with more bins (context training).
- Syntax element partitioning does not reduce number of bins used with each context.

### Improved Coding Efficiency

<table>
<thead>
<tr>
<th>Entropy Slices per frame [MB/slice]</th>
<th>Total Coding Penalty</th>
<th>Coding Penalty due to Reduced Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 [3600]</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>2 [1800]</td>
<td>0.30%</td>
<td>0.20%</td>
</tr>
<tr>
<td>3 [1200]</td>
<td>0.61%</td>
<td>0.41%</td>
</tr>
<tr>
<td>4 [900]</td>
<td>0.88%</td>
<td>0.57%</td>
</tr>
<tr>
<td>6 [600]</td>
<td>1.47%</td>
<td>0.95%</td>
</tr>
<tr>
<td>8 [450]</td>
<td>1.93%</td>
<td>1.20%</td>
</tr>
<tr>
<td>18 [200]</td>
<td>4.13%</td>
<td>2.38%</td>
</tr>
<tr>
<td>36 [100]</td>
<td>7.36%</td>
<td>3.87%</td>
</tr>
<tr>
<td>72 [50]</td>
<td>12.21%</td>
<td>5.50%</td>
</tr>
</tbody>
</table>

E.g. BigShips QP=27, IPPP
Area Cost (ASIC)

• Entire CABAC does NOT have to be replicated
  – Context selection, and context memory are not replicated
• Area increase due to
  – Replicated arithmetic decoder
  – Control and FIFO between engines
Experimental Results

- Validated with JM12.0 under common conditions across 720p: BigShips, City, Crew, Night, ShuttleStart

*For approx. same speed-up (~2.4 to 2.7x)*

<table>
<thead>
<tr>
<th>Prediction Structure</th>
<th>H.264/AVC Slices</th>
<th>Entropy Slices</th>
<th>Syntax Element Partitioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area Cost</td>
<td>3x</td>
<td>3x</td>
<td>1.5x</td>
</tr>
<tr>
<td>I only</td>
<td>0.87 2.43</td>
<td>0.25 2.43</td>
<td>0.06 2.60</td>
</tr>
<tr>
<td>IPPP</td>
<td>1.44 2.42</td>
<td>0.55 2.44</td>
<td>0.32 2.72</td>
</tr>
<tr>
<td>IBBP</td>
<td>1.71 2.46</td>
<td>0.69 2.47</td>
<td>0.37 2.76</td>
</tr>
</tbody>
</table>

2 to 4x reduction in coding penalty
Adaptive Bin Allocation (Varying QP)

- To reduce Start Code overhead – assign multiple groups to each partition and reduce partitions \((5 \rightarrow 3)\)
- Bin distribution changes with QP – combine adaptively

### Mode

<table>
<thead>
<tr>
<th>Mode</th>
<th>MBINFO</th>
<th>PRED</th>
<th>CBP</th>
<th>SIGMAP</th>
<th>COEFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low QP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>High QP</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Throughput Increase

Low QP

High QP

Switch to High QP

(Left to Right) BigShips, City, Crew, Night, ShuttleStart
Additional Parallelism

• Combine with slice level parallelism
  1. H.264/AVC Slices (8 slices)
  2. Entropy Slices (8 slices)
  3. Entropy Slices (4 slices) + Syntax Element Partitioning
Conclusions

• A new **CABAC** algorithm for **next generation standard** to **increase concurrency** by processing the bins of different **syntax elements in parallel**.

• Achieve a **throughput increase** of up to **3x** without sacrificing **coding efficiency, power, or delay** and **minimal area cost**.

• Can be combined with other approaches for improved coding efficiency and throughput/power.

**Acknowledgements:**

Funding from Texas Instruments and NSERC