ETC2: Texture Compression using Invalid Combinations

Jacob Ström, Martin Pettersson
Ericsson Research
Outline

- Motivation, Previous work
- ETC1, advantages and shortcomings
- Invalid Codes and their use
- ETC 2 = ETC1 + three new modes
- Results compared to ETC 1 and DXTC
Why 3D Graphics... on a Mobile Phone?

- Man-Machine Interfaces
- Screen Savers
- Games
- Maps, Messaging, Browsing and more...
Why is 3D Graphics Hard on a Mobile Phone?

Limited resources:
Why is 3D Graphics Hard on a Mobile Phone?

- Small amount of memory
- Little memory bandwidth
- Little chip area for special purpose
- Powered by batteries
Texture Compression Helps

- **Small amount of memory**
  - More texture data can fit in the limited amount of memory

- **Little memory bandwidth**
  - More texturing possible for same amount of bandwidth

- **Little chip area for special purpose**
  - A texture cache using compressed data can be made smaller

- **Powered by batteries**
  - Reduced bandwidth means lower energy consumption
Previous Work

- CCC [Campbell et al. ’86]
Previous Work

- CCC [Campbell et al. ’86]
Previous Work

- CCC [Campbell et al. ’86]
Previous Work

- CCC [Campbell et al. ’86]
- S3TC/DXTC [Iourcha et al. ’99]
Previous Work

- CCC [Campbell et al. ’86]
- S3TC/DXTC [lourcha et al. ’99]
Previous Work

- CCC [Campbell et al. ’86]
- S3TC/DXTC [Iourcha et al. ’99]
Previous Work

- CCC [Campbell et al. ’86]
- S3TC/DXTC [lourcha et al. ’99]
Previous Work

- CCC [Campbell et al. ’86]
- S3TC/DXTC [Iourcha et al. ’99]
- PVR-TC [Fenney ’03]
Previous Work

- CCC [Campbell et al. ’86]
- S3TC/DXTC [lourcha et al. ’99]
- PVR-TC [Fenney ’03]
Previous Work

- CCC [Campbell et al. ’86]
- S3TC/DXTC [lourcha et al. ’99]
- PVR-TC [Fenney ’03]
Previous Work

- CCC [Campbell et al. ’86]
- S3TC/DXTC [Iourcha et al. ’99]
- PVR-TC [Fenney ’03]
Previous Work

- CCC [Campbell et al. ’86]
- S3TC/DXTC [lourcha et al. ’99]
- PVR-TC [Fenney ’03]
- Compressed Lossless Texture Representation and Caching [Inada and McCool 06’]
  - uses special purpose caches to allow for variable bit rate
Of the fixed rate systems, S3TC/DXTC achieved the best quality.

Could a equally good system of lower complexity be built?

PACKMAN [Strom and Akenine-Moller ’03]
  - very simple but considerably lower quality (around 2.5 dB)

iPACKMAN/Ericsson Texture Compression (ETC) [Strom and Akenine-Moller ’05]
  - still simple and quality on par with S3TC/DXTC

Could ETC be enhanced to surpass S3TC/DXTC in quality?
Recap ETC1

- The human visual system is more sensitive to luminance than to chrominance.
- The idea is to specify the base color for an entire 2x4 block (base color marked with a blue circle)
- The luminance can then be changed per pixel by moving along the intensity direction (1,1,1)
ETC1 Recap

- On a macro level, it can look like this

```
"base color" + per-pixel luminance = resulting image
```
ETC1 Recap

- This is all fine, if the variation inside a sub-block is aligned more or less with the intensity direction.
ETC1 Weaknesses

- However, if the block contains a number of pixels with very different chrominance, the results will be poor.
ETC1 Weaknesses

- However, if the block contains a number of pixels with very different chrominance, the results will be poor.
ETC1 Weaknesses

- However, if the block contains a number of pixels with very different chrominance, the results will be poor.
ETC1 Weaknesses

- However, if the block contains a number of pixels with very different chrominance, the results will be poor.
ETC1 Weaknesses

- However, if the block contains a number of pixels with very different chrominance, the results will be poor.
Weaknesses

ETC 1.0

- Another weakness is smooth transitions between two colors of equal luminance.
Weaknesses

ETC 1.0

- Another weakness is smooth transitions between two colors of equal luminance.
- Since only one color per sub-block is possible, block artifacts are more pronounced than for S3TC/DXTC for such blocks.
How to Improve ETC1

- We have realized the need to improve ETC1 for certain blocks, but how do we do it?
- Each 4x4 blocks takes 64 bits in ETC1. One way would be to add another bit to signal new modes for problematic blocks.
- But 65 bits per block is less than ideal...
Redundant Bit Sequences

- In S3TC, there are coded bit sequences that produce exactly the same output block.
Redundant Bit Sequences

- In S3TC, there are coded bit sequences that produce exactly the same output block.
In S3TC, there are coded bit sequences that produce exactly the same output block.
Redundant Bit Sequences

- In S3TC, there are coded bit sequences that produce exactly the same output block.

- This is of course wasteful, so instead the "darkest" colour is forced to be the first one. If not, the block is decoded with one color being black:
Redundant Bit Sequences

- In S3TC, there are coded bit sequences that produce exactly the same output block.

- This is of course wasteful, so instead the "darkest" colour is forced to be the first one. If not, the block is decoded with one color being black:
Redundant Bit Sequences

- In S3TC, there are coded bit sequences that produce exactly the same output block.

- This is of course wasteful, so instead the ”darkest” colour is forced to be the first one. If not, the block is decoded with one color being black:

- This technique was also used by Munkberg et al. under the name the ”ordering trick”.

![Images showing color sequence examples]
Redundant Bit Sequences

- In S3TC, there are coded bit sequences that produce exactly the same output block.

- This is of course wasteful, so instead the "darkest" colour is forced to be the first one. If not, the block is decoded with one color being black:

- This technique was also used by Munkberg et al. under the name the "ordering trick".

- We looked for redundant bit combinations in ETC1...
Invalid Bit Sequences
and their use

- ... and found nothing exploitable.
Invalid Bit Sequences
and their use

- ... and found nothing exploitable.
- But what if some 64-bit sequences do not produce valuable blocks? They can then be used for new modes.
Invalid Bit Sequences
and their use

- ... and found nothing exploitable.
- But what if some 64-bit sequences do not produce valuable blocks? They can then be used for new modes.

All 64-bit sequences

0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010 .
1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111
Invalid Bit Sequences
and their use

- ... and found nothing exploitable.
- But what if some 64-bit sequences do not produce valuable blocks? They can then be used for new modes.
- So we started to look for invalid bit sequences instead

All 64-bit sequences

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010
```

? ← invalid
Invalid Bit Sequences in ETC1

- In some blocks in ETC1, the base color of the right sub-block is coded differentially w.r.t. the left sub-block.
Invalid Bit Sequences in ETC1

- In some blocks in ETC1, the base color of the right sub-block is coded differentially w.r.t. the left sub-block.
Invalid Bit Sequences in ETC1

- In some blocks in ETC1, the base color of the right sub-block is coded differentially w.r.t. the left sub-block.

left: encode as RGB555

right: encode as left + dRGB333
Invalid Bit Sequences in ETC1

- In some blocks in ETC1, the base color of the right sub-block is coded differentially w.r.t. the left sub-block.
- Right_RED = Left_RED + dR
Invalid Bit Sequences in ETC1

- In some blocks in ETC1, the base color of the right sub-block is coded differentially w.r.t. the left sub-block.
- Right_RED = Left_RED + dR
- But if Left_RED is 0, a negative dR would mean a negative color (physically impossible). Such a bit sequence is possible but would never be used by the encoder.
Invalid Bit Sequences in ETC1

- In some blocks in ETC1, the base color of the right sub-block is coded differentially w.r.t. the left sub-block.
- \( \text{Right\_RED} = \text{Left\_RED} + \text{dR} \)
- But if \( \text{Left\_RED} \) is 0, a negative \( \text{dR} \) would mean a negative color (physically impossible). Such a bit sequence is possible but would never be used by the encoder.
- These bit sequences can be detected and the bits can be decoded a different way.
Schematic of an ETC2 decoder
Schematic of a ETC2 decoder

ETC2 decoder

ETC1 decoder

auxiliary mode decoder

compressed bits

1111
0010
1110
0110
1100
1001
0010
0110
1100
1001
0010
0110
1100
1001
0010
Schematic of a ETC2 decoder

- ETC1 can always be used – ETC2 better or same
- Decoder is backward compatible
How many bits can we recover?

- How much data can be transmitted using bit sequences that overflow in the red component?

![Diagram showing signals, differential mode, overflow, and remaining bits]

- 55 bits remaining bits can be used for a new mode
How many bits can we recover?

- How much data can be transmitted using bit sequences that overflow in the red component?
- But $R + dR$ can overflow (or underflow) in exactly 16 ways, which means we can signal 4 more bits for the new mode.

55 bits remaining can be used for a new mode.
How many bits can we recover?

- How much data can be transmitted using bit sequences that overflow in the red component?
- But R+dR can overflow (or underflow) in exactly 16 ways, which means we can signal 4 more bits for the new mode.

![Diagram showing signal transmission and bit recovery](image-url)
How many bits can we recover?

- How much data can be transmitted using bit sequences that overflow in the red component?
- But $R + dR$ can overflow (or underflow) in exactly 16 ways, which means we can signal 4 more bits for the new mode.

```
59 bits can be used for new mode
```

```
55 bits
```

```
remaining bits can be used for a new mode
```

```
signals differential mode
```

```
signals overflow and carries four bits
```

```
signals
```

```
R dR
```

```
59 bits can be used for new mode
```

```
55 bits
```

```
```
More Modes

- But the Green Component can also overflow, so we can get another mode.
More Modes

- But the Green Component can also overflow, so we can get another mode.
- We must first make sure the red does not overflow, otherwise the decoder will think it is that mode.
More Modes

- But the Green Component can also overflow, so we can get another mode.
- We must first make sure the red does not overflow, otherwise the decoder will think it is that mode.
- \( R + dR \) can avoid to overflow in 256-16 ways, so we can safely store 7 bits in \( R + dR \)
More Modes

- But the Green Component can also overflow, so we can get another mode.
- We must first make sure the red does not overflow, otherwise the decoder will think it is that mode.
- R+dR can avoid to overflow in 256-16 ways, so we can safely store 7 bits in R+dR
- And 4 bits in G+dG
More Modes

- But the Green Component can also overflow, so we can get another mode.
- We must first make sure the red does not overflow, otherwise the decoder will think it is that mode.
- \( R + dR \) can avoid to overflow in 256-16 ways, so we can safely store 7 bits in \( R + dR \)
- And 4 bits in \( G + dG \)
The same can be done for the blue component and we have three new modes:

- Mode 1: 59 bits payload
- Mode 2: 58 bits payload
- Mode 3: 57 bits payload

We want three new modes that targets blocks that ETC1 has most problems with:

- Colors in block have very different chrominances
- Smooth transitions between several colors in the block

The first problem was addressed by us in a previous paper published at a small national conference.
Mode 1: The "T-Mode"

- The first mode targets blocks where some pixels are of a very different chrominance.
Mode 1: The "T-Mode"

- The first mode targets blocks where some pixels are of a very different chrominance.
- Two colors are stored in the block.
Mode 1: The "T-Mode"

- The first mode targets blocks where some pixels are of a very different chrominance.
- Two colors are stored in the block.
Mode 1: The ”T-Mode”

- The first mode targets blocks where some pixels are of a very different chrominance.
- Two colors are stored in the block.
Mode 1: The "T-Mode"

- The first mode targets blocks where some pixels are of a very different chrominance.
- Two colors are stored in the block.
- Two more colors are obtained by modulating the first color along the intensity direction.

---

original block

---

![Diagram](image)
Mode 1: The "T-Mode"

- The first mode targets blocks where some pixels are of a very different chrominance.
- Two colors are stored in the block.
- Two more colors are obtained by modulating the first color along the intensity direction.
Mode 1: The "T-Mode"

- The first mode targets blocks where some pixels are of a very different chrominance.
- Two colors are stored in the block.
- Two more colors are obtained by modulating the first color along the intensity direction.
Mode 1: The "T-Mode"

- The first mode targets blocks where some pixels are of a very different chrominance.
- Two colors are stored in the block.
- Two more colors are obtained by modulating the first color along the intensity direction.

![Diagram showing original block and T-mode transformation]
Mode 1: The “T-Mode”

- The first mode targets blocks where some pixels are of a very different chrominance.
- Two colors are stored in the block.
- Two more colors are obtained by modulating the first color along the intensity direction.
Mode 1: The "T-Mode"

- The first mode targets blocks where some pixels are of a very different chrominance.
- Two colors are stored in the block.
- Two more colors are obtained by modulating the first color along the intensity direction.
T-Mode Decompression

- The first color is expanded from RGB444 to RGB888.
The first color is expanded from RGB444 to RGB888. Three bits are then used to select one of eight intensity modifiers.

```
204,170,51

[3,6,11,16,23,32,41,64]
```
T-Mode Decompression

- The first color is expanded from RGB444 to RGB888.
- Three bits are then used to select one of eight intensity modifiers.
- This value is then used additively and subtractively to get two more colors.

$$204,170,51 + 32$$

$$[3, 6, 11, 16, 23, 32, 41, 64]$$
The first color is expanded from RGB444 to RGB888. Three bits are then used to select one of eight intensity modifiers. This value is then used additively and subtractively to get two more colors.
T-Mode Decompression

- The first color is expanded from RGB444 to RGB888.
- Three bits are then used to select one of eight intensity modifiers.
- This value is then used additively and subtractively to get two more colors.

\[ \begin{align*}
172,138,19 & \quad -32 \quad 204,170,51 \quad +32 \quad 204,170,51
\end{align*} \]
T-Mode Decompression

- The first color is expanded from RGB444 to RGB888.
- Three bits are then used to select one of eight intensity modifiers.
- This value is then used additively and subtractively to get two more colors.
- The second color is then expanded to RGB888.
T-Mode Decompression

- Two bits per pixel decides which of the four colors to choose from.
T-Mode Decompression

- Two bits per pixel decides which of the four colors to choose from.

<table>
<thead>
<tr>
<th>col0</th>
<th>col1</th>
<th>col2</th>
<th>col3</th>
</tr>
</thead>
<tbody>
<tr>
<td>172,138,19</td>
<td>204,170,51</td>
<td>204,170,51</td>
<td>68,68,68</td>
</tr>
</tbody>
</table>

12 bit RGB444 12 bit RGB444
T-Mode Decompression

- Two bits per pixel decides which of the four colors to choose from.

```
col0  col1  col2  col3
172,138,19  204,170,51  204,170,51  68,68,68
```
T-Mode Decompression

- Two bits per pixel decides which of the four colors to choose from.

```
<table>
<thead>
<tr>
<th>col0</th>
<th>col1</th>
<th>col2</th>
<th>col3</th>
</tr>
</thead>
<tbody>
<tr>
<td>172,138,19</td>
<td>204,170,51</td>
<td>204,170,51</td>
<td>68,68,68</td>
</tr>
</tbody>
</table>
```

12 bit RGB444 12 bit RGB444
T-Mode Decompression

- Two bits per pixel decides which of the four colors to choose from.
T-Mode Decompression

- Two bits per pixel decides which of the four colors to choose from.
- All in all 59 bits which fits into the first mode.

```
col0  col1  col2  col3
172,138,19  204,170,51  204,170,51  68,68,68
```

```
12 bit RGB444  12 bit RGB444  3 bits  32 bits pixel indices
```
Mode 2: The "H-Mode"

- The second mode targets blocks where there are two groups of pixels that can be intensity modulated.
Mode 2: The "H-Mode"

- The second mode targets blocks where there are two groups of pixels that can be intensity modulated.
- Two colors are stored in the block.
Mode 2: The "H-Mode"

- The second mode targets blocks where there are two groups of pixels that can be intensity modulated.
- Two colors are stored in the block.
- Both colors are modulated in the intensity direction...
Mode 2: The "H-Mode"

- The second mode targets blocks where there are two groups of pixels that can be intensity modulated.
- Two colors are stored in the block.
- Both colors are modulated in the intensity direction... and clamped.
Mode 2: The "H-Mode"

- The second mode targets blocks where there are two groups of pixels that can be intensity modulated.
- Two colors are stored in the block.
- Both colors are modulated in the intensity direction... and clamped.
- These four colors are used to build up the block.

![Diagram showing original block and H-mode transformation](image-url)
Mode 2: The "H-Mode"

- The second mode targets blocks where there are two groups of pixels that can be intensity modulated.
- Two colors are stored in the block.
- Both colors are modulated in the intensity direction... and clamped.
- These four colors are used to build up the block.
The H mode needs 59 bits just as the T-mode.
However, only 58 bits are available.
But since the two colors are interchangeable, we can use the “ordering trick” to signal an extra bit:
  – “Darkest” color first signals a 0
  – “Brightest” color first signals a 1
This way we can fit the H-mode into the 58 bit slot.
Last Mode: The Planar mode

- Some blocks have very slowly varying colors. These are not always well approximated with ETC or S3TC/DXTC.
Last Mode: The Planar mode

- Some blocks have very slowly varying colors. These are not always well approximated with ETC or S3TC/DXTC.
- Such blocks contain very little information – can be handled well with a special mode.
Last Mode: The Planar mode

- Some blocks have very slowly varying colors. These are not always well approximated with ETC or S3TC/DXTC.
- Such blocks contain very little information – can be handled well with a special mode.
- Three colors are stored per block: $C_0$, $C_H$ and $C_V$ in RGB676. The color is interpolated colinearly (using a planar model) in between.
Last Mode: The Planar mode

- Some blocks have very slowly varying colors. These are not always well approximated with ETC or S3TC/DXTC.
- Such blocks contain very little information – can be handled well with a special mode.
- Three colors are stored per block: $C_0$, $C_H$ and $C_V$ in RGB676. The color is interpolated colinearly (using a planar model) in between.
Results

- ETC2 was tested on 64 textures, each texture on all mipmap sizes between 512x512 and 8x8 pixels.
- The textures were both photographic images and game textures.
- The system has been compared to
  - ETC1
  - S3TC/DXTC
  - ATI-TC
Results

- For the highest mipmap:
  - 0.8 dB higher quality than S3TC/DXTC (same bitrate)
  - 1.0 dB higher quality than ETC1 (same bitrate)
  - 1.8 dB higher quality than ATI-TC (same bitrate)
Results – All Mipmaps
margin to next best varies between 0.8 dB and 1.3 dB
Results

<table>
<thead>
<tr>
<th>original</th>
<th>S3TC/DXTC</th>
<th>ETC1</th>
<th>ETC2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results

<table>
<thead>
<tr>
<th>original</th>
<th>S3TC/DXTC</th>
<th>ETC1</th>
<th>ETC2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Score 1:</th>
<th>S3TC/DXTC</th>
<th>ETC 1</th>
<th>ETC2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results

<table>
<thead>
<tr>
<th></th>
<th>original</th>
<th>S3TC/DXTC</th>
<th>ETC1</th>
<th>ETC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image 1</td>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="image1.png" alt="Image 1" /></td>
</tr>
<tr>
<td>Image 2</td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
</tr>
<tr>
<td>Image 3</td>
<td><img src="image3.png" alt="Image 3" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
</tr>
<tr>
<td>Image 4</td>
<td><img src="image4.png" alt="Image 4" /></td>
<td><img src="image4.png" alt="Image 4" /></td>
<td><img src="image4.png" alt="Image 4" /></td>
<td><img src="image4.png" alt="Image 4" /></td>
</tr>
</tbody>
</table>

Legend:
- ETC1
- T-mode
- H-mode
- Planar

ERICSSON
Results cont.

<table>
<thead>
<tr>
<th>original</th>
<th>S3TC/DXTC</th>
<th>ETC1</th>
<th>ETC2</th>
</tr>
</thead>
</table>

[Images of original, S3TC/DXTC, ETC1, ETC2]
Results cont.

<table>
<thead>
<tr>
<th></th>
<th>original</th>
<th>S3TC/DXTC</th>
<th>ETC1</th>
<th>ETC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planar</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H-mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T-mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETC1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETC2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results cont.

<table>
<thead>
<tr>
<th>Original</th>
<th>S3TC/DXTC</th>
<th>ETC1</th>
<th>ETC2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Original Image" /></td>
<td><img src="image2" alt="S3TC/DXTC Image" /></td>
<td><img src="image3" alt="ETC1 Image" /></td>
<td><img src="image4" alt="ETC2 Image" /></td>
</tr>
<tr>
<td><img src="image5" alt="Original Image" /></td>
<td><img src="image6" alt="S3TC/DXTC Image" /></td>
<td><img src="image7" alt="ETC1 Image" /></td>
<td><img src="image8" alt="ETC2 Image" /></td>
</tr>
<tr>
<td><img src="image9" alt="Original Image" /></td>
<td><img src="image10" alt="S3TC/DXTC Image" /></td>
<td><img src="image11" alt="ETC1 Image" /></td>
<td><img src="image12" alt="ETC2 Image" /></td>
</tr>
</tbody>
</table>
Conclusion

- We have presented ETC2
- It is backward compatible with ETC1 – new hardware will automatically decompress both correctly
- Three new modes are added without changing the old modes – thus it is guaranteed to always be better or equal to ETC1
- Tests show that it is 0.8 dB better than S3TC/DXTC which is a significant improvement
- Visual improvements are especially pronounced for blocks with sharp chrominance changes and for smooth regions.
Thank You
Decompression Complexity

- Due to the new modes, ETC2 is more complex than ETC1.
- We have not implemented the two algorithms in VHDL in order to compare their complexity.
- The extra cost for the T- and H- mode is mostly control logic (which is simple), seven multiplexors per color channel and one 12-bit comparator.
- The extra cost for the planar mode is five adders per color channel, and multiplexors.
ETC2 was tested on 64 textures, each texture on all mipmap sizes between 512x512 and 8x8 pixels. The images were contained both photographic images and game textures. The system has been compared to
- ETC1 (compressed exhaustively)
- S3TC/DXTC (compressed using ATI’s The Compressonator with weights set to 1,1,1 to maximize PSNR)
- ATI-TC (compressed with ATI’s The Compressonator)