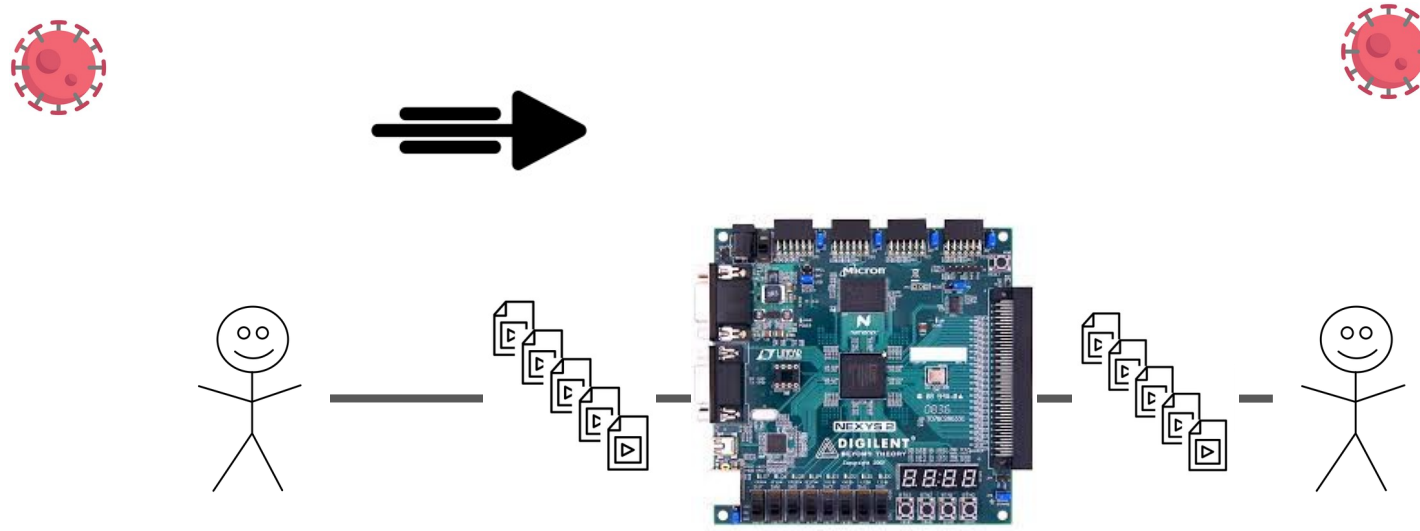
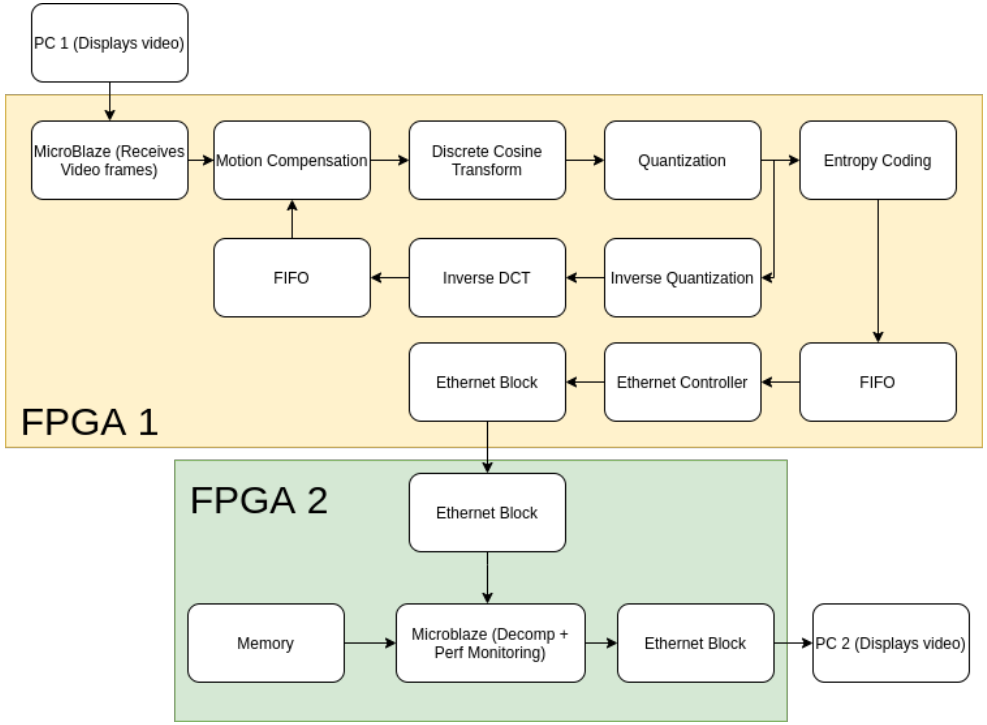
# H.263 Media Compression Broadcast on FPGAs Mid-project Demo

**Team 4 (Justin Hai, Yufeng Zhou, Isamu Poy)**

# Project overview

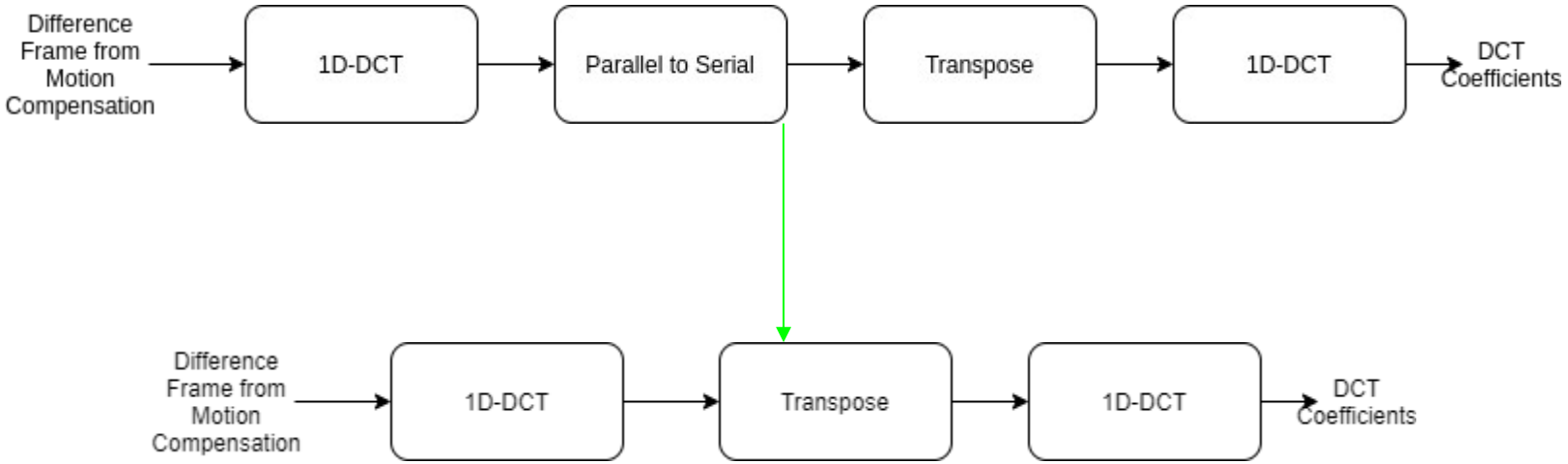


# System Block Diagram

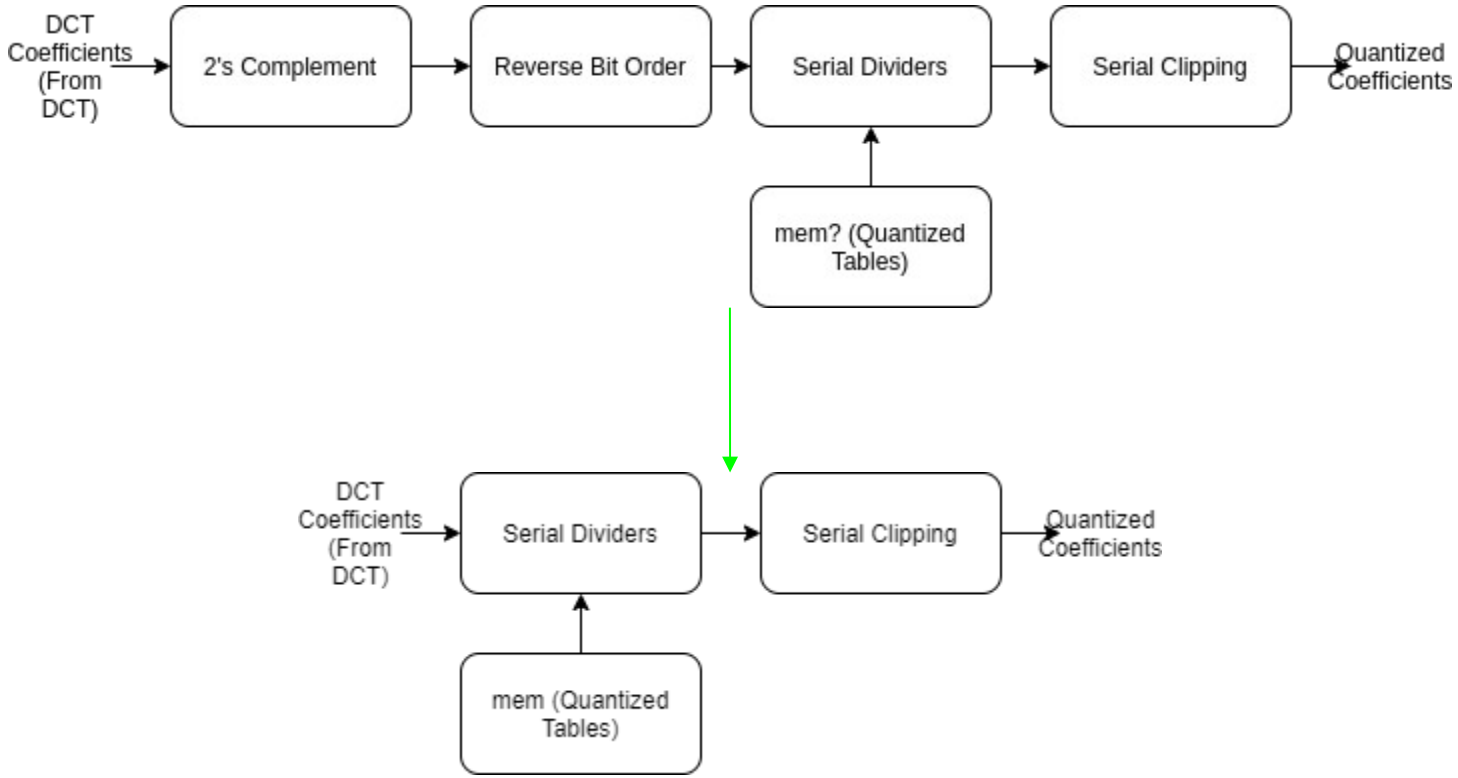


# Major Changes

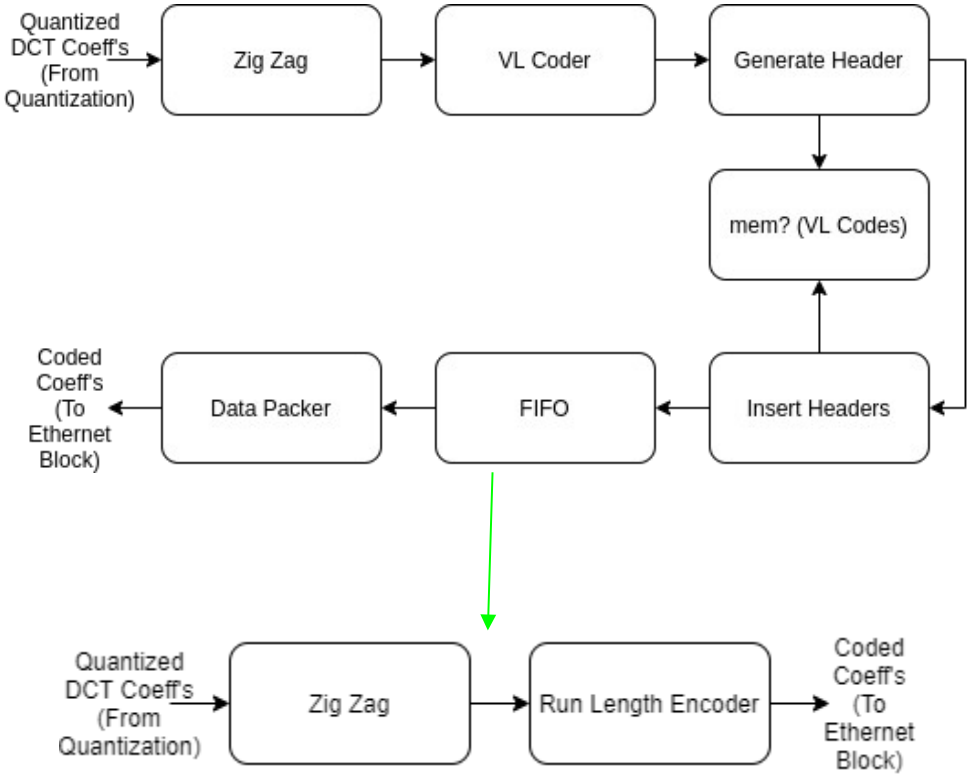
# Major Changes - DCT



# Major Changes - Quantizer



# Major Changes - Encoder



# Major Changes - Other

1. Switched to Fixed Point Arithmetic rather than Floats
2. Switched to using TEMAC adapter rather than Ethernet Lite
3. Utilize DDR memory on microblaze

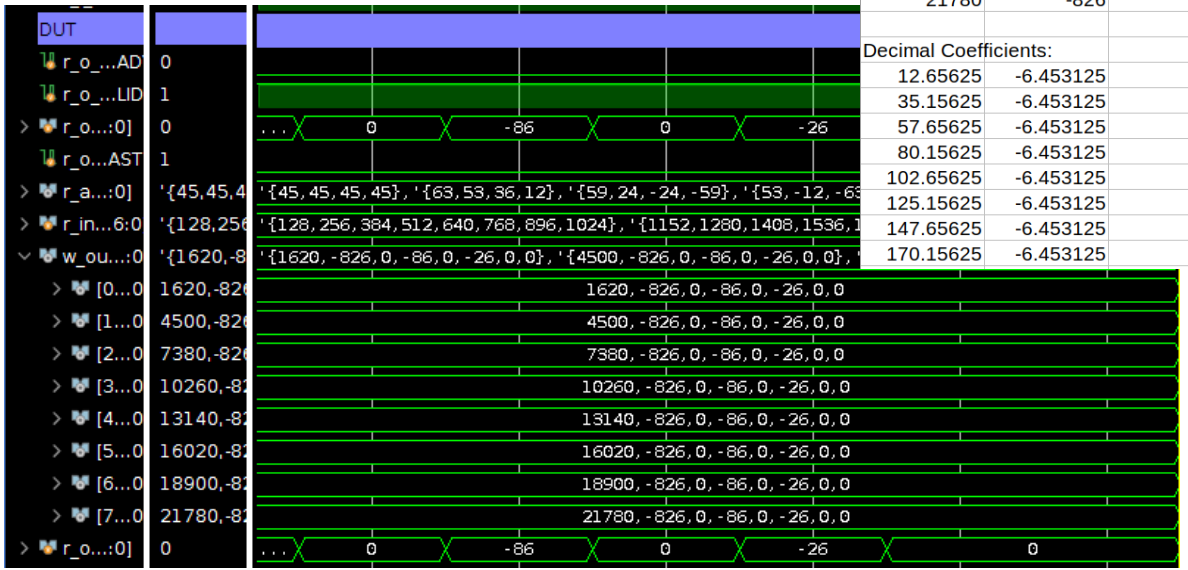
# Challenges

- Hard to understand the TEMAC manual (263 pages) -> Use provided example design to help understanding TEMAC
- Implementation of cores unclear -> Found reference designs online
- Reference designs unintuitive -> Write/draw test cases on paper
- Debugging segfaults in the software implementation -> Use GDB for debugging
- Debugging odd behaviour in embedded programming -> Get software working 100% correctly and compare outputs
- Limited time -> Build functional cores first, then improve performance
- Limited on-chip area -> Refactor implementations, reducing parallelism

# Demo Recap

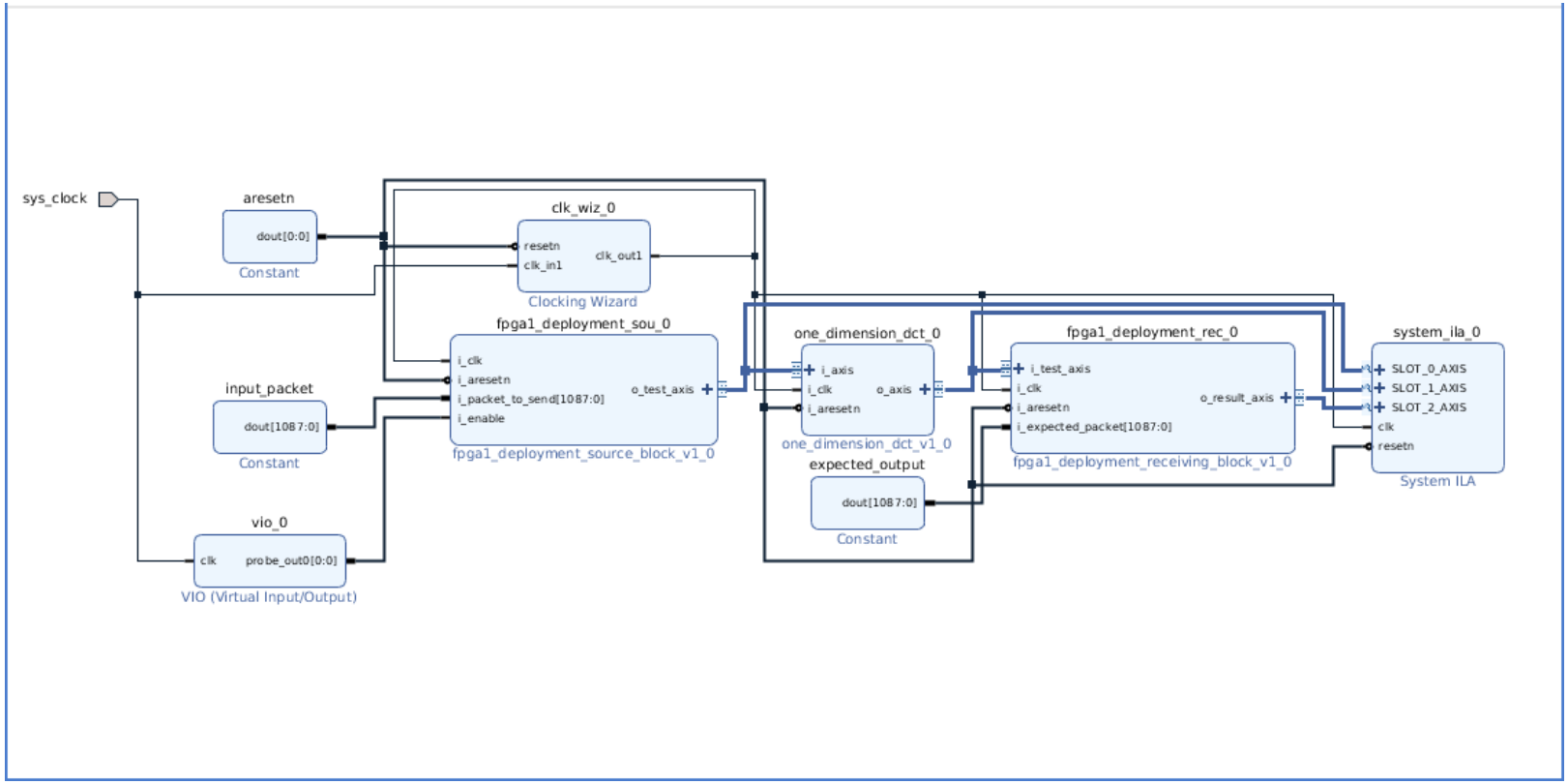
# Demo - Compression Algorithm Simulation

| Fixed Point Coefficients: | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1620 | -826 | 0 | -86 | 0 | -26 | 0 | 0 |
| 4500 | -826 | 0 | -86 | 0 | -26 | 0 | 0 |
| 7380 | -826 | 0 | -86 | 0 | -26 | 0 | 0 |
| 10260 | -826 | 0 | -86 | 0 | -26 | 0 | 0 |
| 13140 | -826 | 0 | -86 | 0 | -26 | 0 | 0 |
| 16020 | -826 | 0 | -86 | 0 | -26 | 0 | 0 |
| 18900 | -826 | 0 | -86 | 0 | -26 | 0 | 0 |
| 21780 | -826 | 0 | -86 | 0 | -26 | 0 | 0 |
| | | | | | | | |
| Decimal Coefficients: | | | | | | | |
| 12.65625 | -6.453125 | 0 | -0.671875 | 0 | -0.203125 | 0 | 0 |
| 35.15625 | -6.453125 | 0 | -0.671875 | 0 | -0.203125 | 0 | 0 |
| 57.65625 | -6.453125 | 0 | -0.671875 | 0 | -0.203125 | 0 | 0 |
| 80.15625 | -6.453125 | 0 | -0.671875 | 0 | -0.203125 | 0 | 0 |
| 102.65625 | -6.453125 | 0 | -0.671875 | 0 | -0.203125 | 0 | 0 |
| 125.15625 | -6.453125 | 0 | -0.671875 | 0 | -0.203125 | 0 | 0 |
| 147.65625 | -6.453125 | 0 | -0.671875 | 0 | -0.203125 | 0 | 0 |
| 170.15625 | -6.453125 | 0 | -0.671875 | 0 | -0.203125 | 0 | 0 |

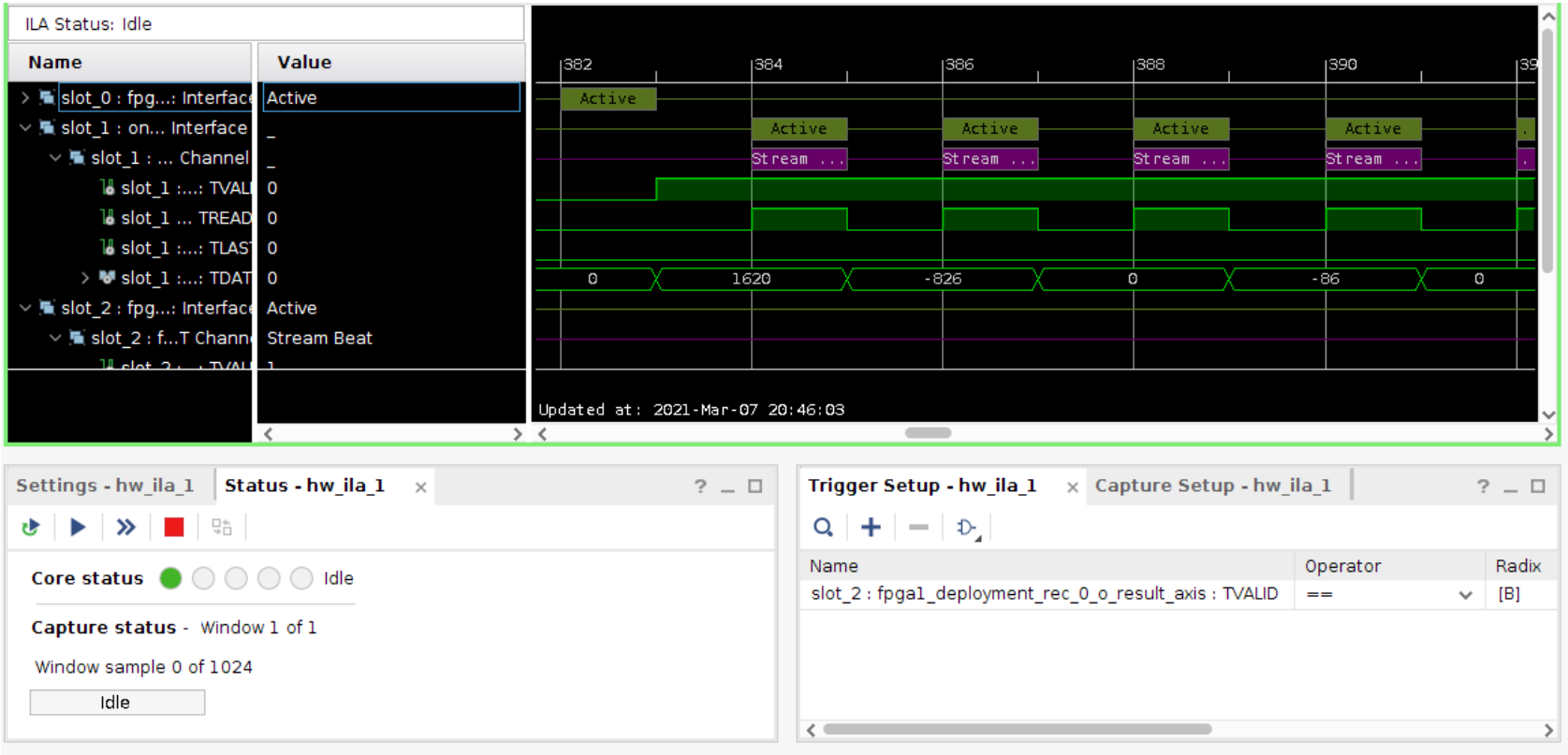



```
Output          1: 12.000000
Output          2: -7.000000
Output          3: 0.000000
Output          4: -1.000000
Output          5: 0.000000
Output          6: -1.000000
Output          7: 0.000000
Output          8: 0.000000
Output          9: 35.000000
Output         10: -7.000000
```

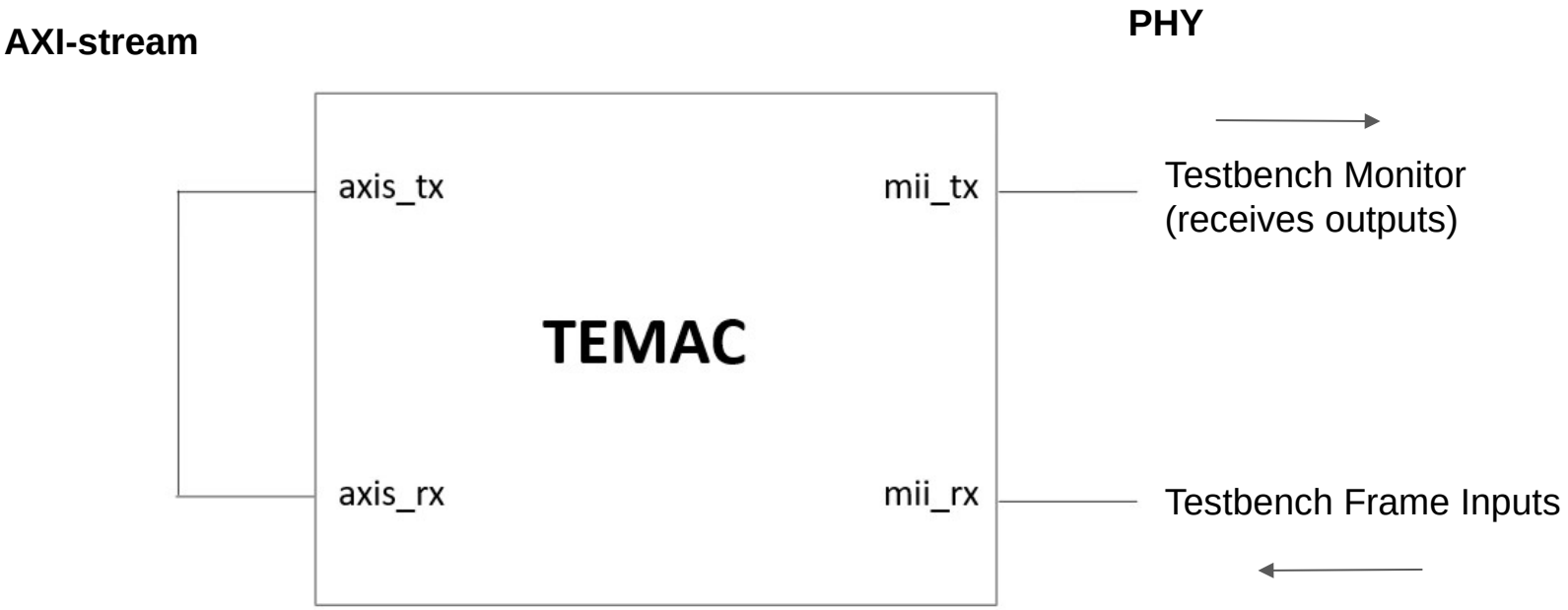
# Demo - Compression Algorithm Deployment



# Demo - Compression Algorithm Deployment



# TEMAC (tri-mode ethernet mac)



# Input frames

Five predefined input frames

- **Frame structure**: (Destination address|Source address|Data) <= Valid/Error bit
- **First frame**: Sending data of length 46
- **Second frame**: Sending data of type 8000
- **Third frame**: Sending data containing error bits
- **Fourth frame**: Sending data of length 3
- **Fifth frame**: Sending data of wrong addresses

# Frame data example

```
//-----------
    // Frame 0
    //-----------
    frame0.data[0]  = 8'hDA;  frame0.valid[0]  = 1'b1;  frame0.error[0]  = 1'b0; // Destination Address (DA)
    frame0.data[1]  = 8'h02;  frame0.valid[1]  = 1'b1;  frame0.error[1]  = 1'b0;
    frame0.data[2]  = 8'h03;  frame0.valid[2]  = 1'b1;  frame0.error[2]  = 1'b0;
    frame0.data[3]  = 8'h04;  frame0.valid[3]  = 1'b1;  frame0.error[3]  = 1'b0;
    frame0.data[4]  = 8'h05;  frame0.valid[4]  = 1'b1;  frame0.error[4]  = 1'b0;
    frame0.data[5]  = 8'h06;  frame0.valid[5]  = 1'b1;  frame0.error[5]  = 1'b0;
    frame0.data[6]  = 8'h5A;  frame0.valid[6]  = 1'b1;  frame0.error[6]  = 1'b0; // Source Address  (5A)
    frame0.data[7]  = 8'h22;  frame0.valid[7]  = 1'b1;  frame0.error[7]  = 1'b0;
    frame0.data[8]  = 8'h33;  frame0.valid[8]  = 1'b1;  frame0.error[8]  = 1'b0;
    frame0.data[9]  = 8'h44;  frame0.valid[9]  = 1'b1;  frame0.error[9]  = 1'b0;
    frame0.data[10] = 8'h55;  frame0.valid[10] = 1'b1;  frame0.error[10] = 1'b0;
    frame0.data[11] = 8'h66;  frame0.valid[11] = 1'b1;  frame0.error[11] = 1'b0;
    frame0.data[12] = 8'h00;  frame0.valid[12] = 1'b1;  frame0.error[12] = 1'b0;
    frame0.data[13] = 8'h2E;  frame0.valid[13] = 1'b1;  frame0.error[13] = 1'b0; // Length/Type = Length = 46
    frame0.data[14] = 8'h01;  frame0.valid[14] = 1'b1;  frame0.error[14] = 1'b0;
    frame0.data[15] = 8'h02;  frame0.valid[15] = 1'b1;  frame0.error[15] = 1'b0;
    frame0.data[16] = 8'h03;  frame0.valid[16] = 1'b1;  frame0.error[16] = 1'b0;
    frame0.data[17] = 8'h04;  frame0.valid[17] = 1'b1;  frame0.error[17] = 1'b0;
    frame0.data[18] = 8'h05;  frame0.valid[18] = 1'b1;  frame0.error[18] = 1'b0;
    frame0.data[19] = 8'h06;  frame0.valid[19] = 1'b1;  frame0.error[19] = 1'b0;
    frame0.data[20] = 8'h07;  frame0.valid[20] = 1'b1;  frame0.error[20] = 1'b0;
    frame0.data[21] = 8'h08;  frame0.valid[21] = 1'b1;  frame0.error[21] = 1'b0;
    frame0.data[22] = 8'h09;  frame0.valid[22] = 1'b1;  frame0.error[22] = 1'b0;
    frame0.data[23] = 8'h0A;  frame0.valid[23] = 1'b1;  frame0.error[23] = 1'b0;
    frame0.data[24] = 8'h0B;  frame0.valid[24] = 1'b1;  frame0.error[24] = 1'b0;
    frame0.data[25] = 8'h0C;  frame0.valid[25] = 1'b1;  frame0.error[25] = 1'b0;
    frame0.data[26] = 8'h0D;  frame0.valid[26] = 1'b1;  frame0.error[26] = 1'b0;
    frame0.data[27] = 8'h0E;  frame0.valid[27] = 1'b1;  frame0.error[27] = 1'b0;
    frame0.data[28] = 8'h0F;  frame0.valid[28] = 1'b1;  frame0.error[28] = 1'b0;
    frame0.data[29] = 8'h10;  frame0.valid[29] = 1'b1;  frame0.error[29] = 1'b0;
    frame0.data[30] = 8'h11;  frame0.valid[30] = 1'b1;  frame0.error[30] = 1'b0;
    frame0.data[31] = 8'h12;  frame0.valid[31] = 1'b1;  frame0.error[31] = 1'b0;
    frame0.data[32] = 8'h13;  frame0.valid[32] = 1'b1;  frame0.error[32] = 1'b0;
    frame0.data[33] = 8'h14;  frame0.valid[33] = 1'b1;  frame0.error[33] = 1'b0;
    frame0.data[34] = 8'h15;  frame0.valid[34] = 1'b1;  frame0.error[34] = 1'b0;
    frame0.data[35] = 8'h16;  frame0.valid[35] = 1'b1;  frame0.error[35] = 1'b0;
    frame0.data[36] = 8'h17;  frame0.valid[36] = 1'b1;  frame0.error[36] = 1'b0;
    frame0.data[37] = 8'h18;  frame0.valid[37] = 1'b1;  frame0.error[37] = 1'b0;
    frame0.data[38] = 8'h19;  frame0.valid[38] = 1'b1;  frame0.error[38] = 1'b0;
    frame0.data[39] = 8'h1A;  frame0.valid[39] = 1'b1;  frame0.error[39] = 1'b0;
    frame0.data[40] = 8'h1B;  frame0.valid[40] = 1'b1;  frame0.error[40] = 1'b0;
    frame0.data[41] = 8'h1C;  frame0.valid[41] = 1'b1;  frame0.error[41] = 1'b0;
    frame0.data[42] = 8'h1D;  frame0.valid[42] = 1'b1;  frame0.error[42] = 1'b0;
    frame0.data[43] = 8'h1E;  frame0.valid[43] = 1'b1;  frame0.error[43] = 1'b0;
    frame0.data[44] = 8'h1F;  frame0.valid[44] = 1'b1;  frame0.error[44] = 1'b0;
    frame0.data[45] = 8'h20;  frame0.valid[45] = 1'b1;  frame0.error[45] = 1'b0;
    frame0.data[46] = 8'h21;  frame0.valid[46] = 1'b1;  frame0.error[46] = 1'b0;
    frame0.data[47] = 8'h22;  frame0.valid[47] = 1'b1;  frame0.error[47] = 1'b0;
    frame0.data[48] = 8'h23;  frame0.valid[48] = 1'b1;  frame0.error[48] = 1'b0;
    frame0.data[49] = 8'h24;  frame0.valid[49] = 1'b1;  frame0.error[49] = 1'b0;
    frame0.data[50] = 8'h25;  frame0.valid[50] = 1'b1;  frame0.error[50] = 1'b0;
    frame0.data[51] = 8'h26;  frame0.valid[51] = 1'b1;  frame0.error[51] = 1'b0;
    frame0.data[52] = 8'h27;  frame0.valid[52] = 1'b1;  frame0.error[52] = 1'b0;
    frame0.data[53] = 8'h28;  frame0.valid[53] = 1'b1;  frame0.error[53] = 1'b0;
    frame0.data[54] = 8'h29;  frame0.valid[54] = 1'b1;  frame0.error[54] = 1'b0;
    frame0.data[55] = 8'h2A;  frame0.valid[55] = 1'b1;  frame0.error[55] = 1'b0;
    frame0.data[56] = 8'h2B;  frame0.valid[56] = 1'b1;  frame0.error[56] = 1'b0;
    frame0.data[57] = 8'h2C;  frame0.valid[57] = 1'b1;  frame0.error[57] = 1'b0;
    frame0.data[58] = 8'h2D;  frame0.valid[58] = 1'b1;  frame0.error[58] = 1'b0;
    frame0.data[59] = 8'h2E;  frame0.valid[59] = 1'b1;  frame0.error[59] = 1'b0;  // 46th Byte of Data
    // unused
    frame0.data[60] = 8'h00;  frame0.valid[60] = 1'b0;  frame0.error[60] = 1'b0;
    frame0.data[61] = 8'h00;  frame0.valid[61] = 1'b0;  frame0.error[61] = 1'b0;

    // No error in this frame
    frame0.bad_frame  = 1'b0;
```

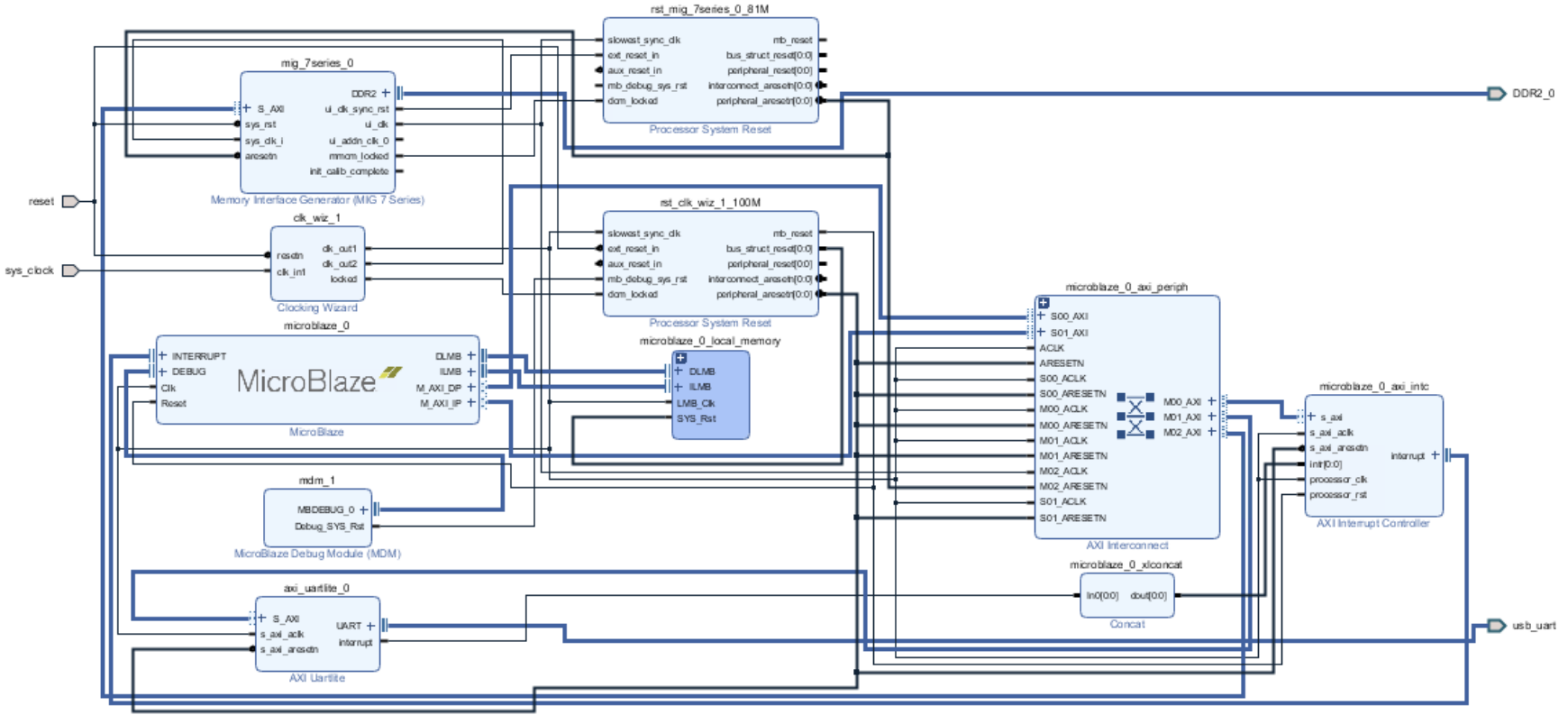
```
//-----------
// Frame 2
//-----------
frame2.data[0]  = 8'hDA;  frame2.valid[0]  = 1'b1;  frame2.error[0]  = 1'b0; // Destination Address (DA)
frame2.data[1]  = 8'h02;  frame2.valid[1]  = 1'b1;  frame2.error[1]  = 1'b0;
frame2.data[2]  = 8'h03;  frame2.valid[2]  = 1'b1;  frame2.error[2]  = 1'b0;
frame2.data[3]  = 8'h04;  frame2.valid[3]  = 1'b1;  frame2.error[3]  = 1'b0;
frame2.data[4]  = 8'h05;  frame2.valid[4]  = 1'b1;  frame2.error[4]  = 1'b0;
frame2.data[5]  = 8'h06;  frame2.valid[5]  = 1'b1;  frame2.error[5]  = 1'b0;
frame2.data[6]  = 8'h5A;  frame2.valid[6]  = 1'b1;  frame2.error[6]  = 1'b0; // Source Address  (5A)
frame2.data[7]  = 8'h22;  frame2.valid[7]  = 1'b1;  frame2.error[7]  = 1'b0;
frame2.data[8]  = 8'h33;  frame2.valid[8]  = 1'b1;  frame2.error[8]  = 1'b0;
frame2.data[9]  = 8'h44;  frame2.valid[9]  = 1'b1;  frame2.error[9]  = 1'b0;
frame2.data[10] = 8'h55;  frame2.valid[10] = 1'b1;  frame2.error[10] = 1'b0;
frame2.data[11] = 8'h66;  frame2.valid[11] = 1'b1;  frame2.error[11] = 1'b0;
frame2.data[12] = 8'h00;  frame2.valid[12] = 1'b1;  frame2.error[12] = 1'b0;
frame2.data[13] = 8'h2E;  frame2.valid[13] = 1'b1;  frame2.error[13] = 1'b0; // Length/Type = Length = 46
frame2.data[14] = 8'h01;  frame2.valid[14] = 1'b1;  frame2.error[14] = 1'b0;
frame2.data[15] = 8'h02;  frame2.valid[15] = 1'b1;  frame2.error[15] = 1'b0;
frame2.data[16] = 8'h03;  frame2.valid[16] = 1'b1;  frame2.error[16] = 1'b0;
frame2.data[17] = 8'h00;  frame2.valid[17] = 1'b1;  frame2.error[17] = 1'b0; // Underrun this frame
frame2.data[18] = 8'h00;  frame2.valid[18] = 1'b1;  frame2.error[18] = 1'b0;
frame2.data[19] = 8'h00;  frame2.valid[19] = 1'b1;  frame2.error[19] = 1'b0;
frame2.data[20] = 8'h00;  frame2.valid[20] = 1'b1;  frame2.error[20] = 1'b0;
frame2.data[21] = 8'h00;  frame2.valid[21] = 1'b1;  frame2.error[21] = 1'b0;
frame2.data[22] = 8'h00;  frame2.valid[22] = 1'b1;  frame2.error[22] = 1'b0;
frame2.data[23] = 8'h00;  frame2.valid[23] = 1'b1;  frame2.error[23] = 1'b1; // Error asserted
frame2.data[24] = 8'h00;  frame2.valid[24] = 1'b1;  frame2.error[24] = 1'b0;
frame2.data[25] = 8'h00;  frame2.valid[25] = 1'b1;  frame2.error[25] = 1'b0;
frame2.data[26] = 8'h00;  frame2.valid[26] = 1'b1;  frame2.error[26] = 1'b0;
frame2.data[27] = 8'h00;  frame2.valid[27] = 1'b1;  frame2.error[27] = 1'b0;
frame2.data[28] = 8'h00;  frame2.valid[28] = 1'b1;  frame2.error[28] = 1'b0;
frame2.data[29] = 8'h00;  frame2.valid[29] = 1'b1;  frame2.error[29] = 1'b0;
frame2.data[30] = 8'h00;  frame2.valid[30] = 1'b1;  frame2.error[30] = 1'b0;
frame2.data[31] = 8'h00;  frame2.valid[31] = 1'b1;  frame2.error[31] = 1'b0;
frame2.data[32] = 8'h00;  frame2.valid[32] = 1'b1;  frame2.error[32] = 1'b0;
frame2.data[33] = 8'h00;  frame2.valid[33] = 1'b1;  frame2.error[33] = 1'b0;
frame2.data[34] = 8'h00;  frame2.valid[34] = 1'b1;  frame2.error[34] = 1'b0;
frame2.data[35] = 8'h00;  frame2.valid[35] = 1'b1;  frame2.error[35] = 1'b0;
frame2.data[36] = 8'h00;  frame2.valid[36] = 1'b1;  frame2.error[36] = 1'b0;
frame2.data[37] = 8'h00;  frame2.valid[37] = 1'b1;  frame2.error[37] = 1'b0;
frame2.data[38] = 8'h00;  frame2.valid[38] = 1'b1;  frame2.error[38] = 1'b0;
frame2.data[39] = 8'h00;  frame2.valid[39] = 1'b1;  frame2.error[39] = 1'b0;
frame2.data[40] = 8'h00;  frame2.valid[40] = 1'b1;  frame2.error[40] = 1'b0;
frame2.data[41] = 8'h00;  frame2.valid[41] = 1'b1;  frame2.error[41] = 1'b0;
frame2.data[42] = 8'h00;  frame2.valid[42] = 1'b1;  frame2.error[42] = 1'b0;
frame2.data[43] = 8'h00;  frame2.valid[43] = 1'b1;  frame2.error[43] = 1'b0;
frame2.data[44] = 8'h00;  frame2.valid[44] = 1'b1;  frame2.error[44] = 1'b0;
frame2.data[45] = 8'h00;  frame2.valid[45] = 1'b1;  frame2.error[45] = 1'b0;
frame2.data[46] = 8'h00;  frame2.valid[46] = 1'b1;  frame2.error[46] = 1'b0;
frame2.data[47] = 8'h00;  frame2.valid[47] = 1'b1;  frame2.error[47] = 1'b0;
frame2.data[48] = 8'h00;  frame2.valid[48] = 1'b1;  frame2.error[48] = 1'b0;
frame2.data[49] = 8'h00;  frame2.valid[49] = 1'b1;  frame2.error[49] = 1'b0;
frame2.data[50] = 8'h00;  frame2.valid[50] = 1'b1;  frame2.error[50] = 1'b0;
frame2.data[51] = 8'h00;  frame2.valid[51] = 1'b1;  frame2.error[51] = 1'b0;
frame2.data[52] = 8'h00;  frame2.valid[52] = 1'b1;  frame2.error[52] = 1'b0;
frame2.data[53] = 8'h00;  frame2.valid[53] = 1'b1;  frame2.error[53] = 1'b0;
frame2.data[54] = 8'h00;  frame2.valid[54] = 1'b1;  frame2.error[54] = 1'b0;
frame2.data[55] = 8'h00;  frame2.valid[55] = 1'b1;  frame2.error[55] = 1'b0;
frame2.data[56] = 8'h00;  frame2.valid[56] = 1'b1;  frame2.error[56] = 1'b0;
frame2.data[57] = 8'h00;  frame2.valid[57] = 1'b1;  frame2.error[57] = 1'b0;
frame2.data[58] = 8'h00;  frame2.valid[58] = 1'b1;  frame2.error[58] = 1'b0;
frame2.data[59] = 8'h00;  frame2.valid[59] = 1'b1;  frame2.error[59] = 1'b0;
// unused
frame2.data[60] = 8'h00;  frame2.valid[60] = 1'b0;  frame2.error[60] = 1'b0;
frame2.data[61] = 8'h00;  frame2.valid[61] = 1'b0;  frame2.error[61] = 1'b0;

// Error this frame
frame2.bad_frame  = 1'b1;
```

# TEMAC testbench results

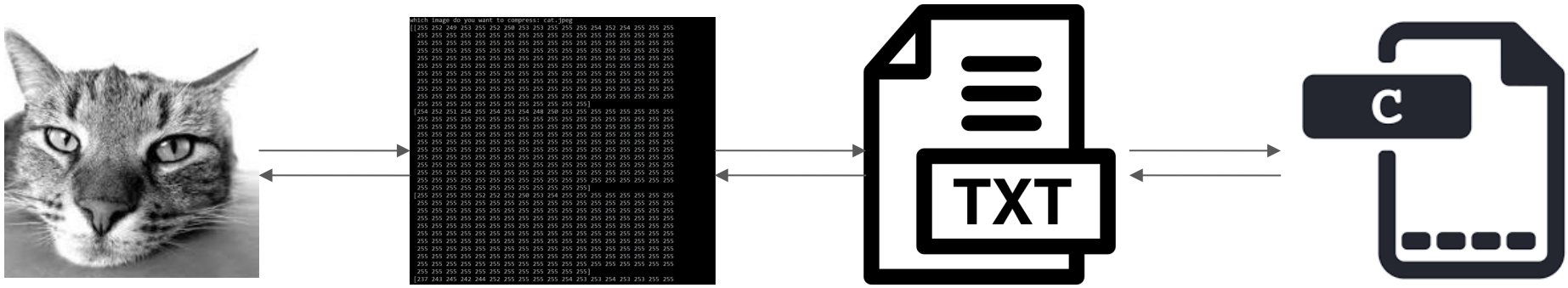

Blue: frame sent successfully

Red: frames unsent

# Microblaze Setup



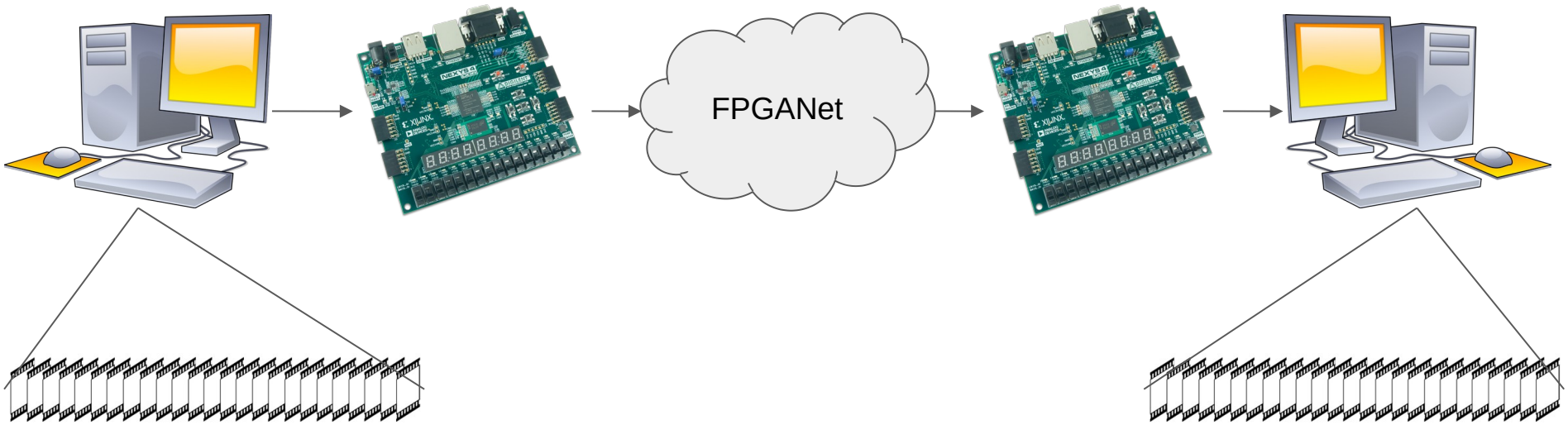
# Software I/O


which image do you want to compress: cat.jpeg
[[255 252 249 253 255 252 250 253 253 255 255 254 252 254 255 255 255
TXT
C

# Future goals

1. Compression algorithm:
    a. Implement and test transpose core, Quantizer, Encoder blocks in hardware
    b. Implement Motion Compensation cores on FPGA
    c. Integrate system
2. Integration on 2 FPGAs
    a. Implement UARTLITE to send pixel values from software to FPGA
3. Optimizations
    a. Reduce on-chip area of hardware blocks
    b. Improve performance by using DSPs
    c. Implement performance monitoring

# Final demo plan



Q/A