

**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΗΛΕΚΤΡΟΝΙΚΗΣ

**Σχεδιασμός Ενσωματωμένων Συστημάτων**

9ο εξάμηνο

4η Εργαστηριακή Άσκηση:

**Εργασία για High Level Synthesis σε FPGA**

03114681: Maliganis Nikolaos

# Άσκηση 1. Performance and resources measurement

|  |
| --- |
|  |
| Εικόνα 1. HW accelerated – unoptimized |

|  |
| --- |
|  |
| Εικόνα 2. Zybo results – unoptimized |

**void** **forward\_propagation**(**float** \*x, **float** \*y)

{

quantized\_type xbuf[N1];

l\_quantized\_type layer\_1\_out[M1];

l\_quantized\_type layer\_2\_out[M2];

//limit resources to max DSP number of Zybo - do not change

**#pragma** HLS ALLOCATION instances=mul limit=80 operation

read\_input:

**for** (**int** i=0; i<N1; i++)

{

**#pragma** HLS PIPELINE

xbuf[i] = x[i];

}

// Layer 1

layer\_1:

**for**(**int** i=0; i<N1; i++)

{

**#pragma** HLS unroll factor=9

**#pragma** HLS PIPELINE

**for**(**int** j=0; j<M1; j++)

{

l\_quantized\_type last = (i==0) ? (l\_quantized\_type) 0 : layer\_1\_out[j];

quantized\_type term = xbuf[i] \* W1[i][j];

layer\_1\_out[j] = last + term;

}

}

layer\_1\_act:

**for**(**int** i=0; i<M1; i++)

{

**#pragma** HLS PIPELINE

layer\_1\_out[i] = ReLU(layer\_1\_out[i]);

}

// Layer 2

layer\_2:

**for**(**int** i=0; i<M2; i++)

{

**#pragma** HLS PIPELINE

l\_quantized\_type result = 0;

**for**(**int** j=0; j<N2; j++)

{

l\_quantized\_type term = layer\_1\_out[j] \* W2[j][i];

result += term;

}

layer\_2\_out[i] = ReLU(result);

}

// Layer 3

layer\_3:

**for**(**int** i=0; i<M3; i++)

{

**#pragma** HLS PIPELINE

l\_quantized\_type result = 0;

**for**(**int** j=0; j<N3; j++)

{

l\_quantized\_type term = layer\_2\_out[j] \* W3[j][i];

result += term;

}

y[i] = tanh(result).to\_float();

}

}

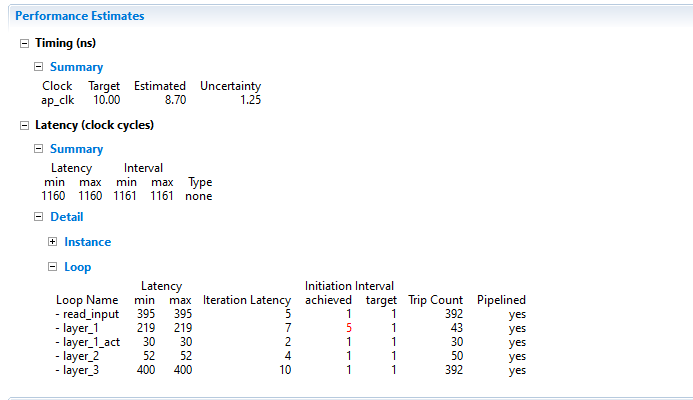
|  |
| --- |
|  |
| Εικόνα 3. HW accelerated – pipelined |

|  |
| --- |
|  |
| Εικόνα 4. Zybo results – pipelined |

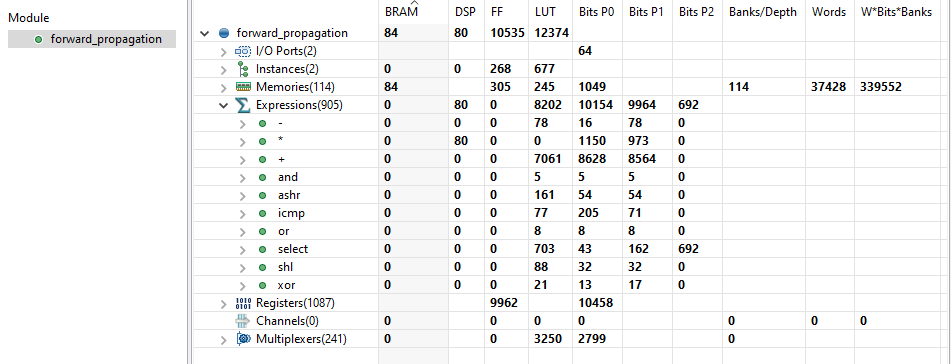
|  |
| --- |
|  |
| Εικόνα 5. HW accelerated – pipelined - unrolled |

|  |
| --- |
|  |
| Εικόνα 6. Zybo results – pipelined - unrolled |

|  |
| --- |
|  |
| Εικόνα 7. |



|  |
| --- |
|  |
| Εικόνα 8. |



# Άσκηση 2. Quality measurement

|  |
| --- |
|  |
| Εικόνα 10. Zybo results – pipelined – unrolled 4-Bit |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 4Bit – Index 10 | 4Bit – Index 11 | 4Bit – Index 12 |

4Bit – Index 10

Max pixel error: 255

Peak Signal-to-Noise Ratio: 14.051663945384881

4Bit – Index 11

Max pixel error: 249

Peak Signal-to-Noise Ratio: 14.634988266184102

4Bit – Index 12

Max pixel error: 255

Peak Signal-to-Noise Ratio: 13.525831164368576

|  |
| --- |
|  |
| Εικόνα 10. Zybo results – pipelined – unrolled 10-Bit |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 10Bit – Index 10 | 10Bit – Index 11 | 10Bit – Index 12 |

10Bit – Index 10

Max pixel error: 5

Peak Signal-to-Noise Ratio: 54.08543347142643

10Bit – Index 11

Max pixel error: 5

Peak Signal-to-Noise Ratio: 52.556099880280584

10Bit – Index 12

Max pixel error: 4

Peak Signal-to-Noise Ratio: 53.76982650203158

**void** **forward\_propagation**(**float** \*x, **float** \*y)

{

quantized\_type xbuf[N1];

l\_quantized\_type layer\_1\_out[M1];

l\_quantized\_type layer\_2\_out[M2];

//limit resources to max DSP number of Zybo - do not change

**#pragma** HLS ALLOCATION instances=mul limit=80 operation

read\_input:

**for** (**int** i=0; i<N1; i++)

{

**#pragma** HLS PIPELINE II=1

xbuf[i] = x[i];

}

// Layer 1

layer\_1:

**for**(**int** i=0; i<N1; i++)

{

**#pragma** HLS PIPELINE II=1

**for**(**int** j=0; j<M1; j++)

{

**#pragma** HLS PIPELINE II=1

**#pragma** HLS unroll factor=2

l\_quantized\_type last = (i==0) ? (l\_quantized\_type) 0 : layer\_1\_out[j];

quantized\_type term = xbuf[i] \* W1[i][j];

layer\_1\_out[j] = last + term;

}

}

layer\_1\_act:

**for**(**int** i=0; i<M1; i++)

{

**#pragma** HLS PIPELINE II=1

layer\_1\_out[i] = ReLU(layer\_1\_out[i]);

}

// Layer 2

layer\_2:

**for**(**int** i=0; i<M2; i++)

{

//#pragma HLS unroll factor=2

**#pragma** HLS PIPELINE II=1

l\_quantized\_type result = 0;

**for**(**int** j=0; j<N2; j++)

{

**#pragma** HLS PIPELINE II=1

**#pragma** HLS unroll factor=2

l\_quantized\_type term = layer\_1\_out[j] \* W2[j][i];

result += term;

}

layer\_2\_out[i] = ReLU(result);

}

// Layer 3

layer\_3:

**for**(**int** i=0; i<M3; i++)

{

**#pragma** HLS PIPELINE II=1

l\_quantized\_type result = 0;

**for**(**int** j=0; j<N3; j++)

{

**#pragma** HLS PIPELINE II=1

**#pragma** HLS unroll factor=2

l\_quantized\_type term = layer\_2\_out[j] \* W3[j][i];

result += term;

}

y[i] = tanh(result).to\_float();

}

}