Config – je to trochu jednoduší, opičárny s přehazováním řeší už sám readout:

public void ChipConfig(string filename)

{

if (mask\_frame == null)

mask\_frame = new PixelFrame();

mask\_frame\_matrix = new byte[256, 256];

mask\_frame.data\_array.Clear();

if (localThframe == null)

localThframe = new PixelFrame();

localThframe.data\_array.Clear();

PixelValue tmp\_mask\_pixel = new PixelValue();

chip\_config\_object = new ChipConfigClass();

byte[] pixel\_conf\_bytes = new byte[1];

try

{

pixel\_conf\_bytes = System.IO.File.ReadAllBytes(filename);

}

catch

{

MessageBox.Show("Config file error!");

};

for (int i = 0; i < pixel\_conf\_bytes.Length; i++)

{

byte tmp = pixel\_conf\_bytes[i];

uint y = (uint)(i / 256);

uint x = (uint)(i % 256);

//pixetformat[y, x] = tmp;

chip\_config\_object.SetValue(i,tmp);

mask\_frame\_matrix[x, y] = 0;

// mask frame

if (chip\_config\_object.mask\_bit[i])

{

tmp\_mask\_pixel = new PixelValue();

tmp\_mask\_pixel.coordinateX = (int)x;

tmp\_mask\_pixel.coordinateY = (int)y;

tmp\_mask\_pixel.ToT\_value = 1;

mask\_frame.data\_array.Add(tmp\_mask\_pixel);

mask\_frame\_matrix[x, y] = 1;

};

// local th frame

PixelValue tmp\_th\_pixel = new PixelValue();

tmp\_th\_pixel.coordinateX = (int)x;

tmp\_th\_pixel.coordinateY = (int)y;

tmp\_th\_pixel.count\_value = (byte)(chip\_config\_object.local\_threshold[i]);

localThframe.data\_array.Add(tmp\_th\_pixel);

}

// Send config data

udpSender.SetAllConfigRegister(chip\_config\_object.GetByteArray());

// save into chip

udpSender.ReadoutUnitHwCmdDebug(6, 0);

}

public class ChipConfigClass

{

public uint[] local\_threshold;

public bool[] mask\_bit;

public bool[] test\_bit;

public ChipConfigClass()

{

local\_threshold = new uint[256 \* 256];

mask\_bit = new bool[256 \* 256];

test\_bit = new bool[256 \* 256];

}

public void SetValue(int index, byte pixel\_config\_data)

{

local\_threshold[index] = (uint)(pixel\_config\_data&0x1F);

mask\_bit[index] = ((uint)pixel\_config\_data & 0x80) > 0 ? true : false;

test\_bit[index] = ((uint)pixel\_config\_data & 0x40) > 0 ? true : false;

}

public byte[] GetByteArray()

{

byte[] tmp;

tmp = new byte[256 \* 256];

for (int i = 0; i < (256 \* 256); i++)

{

tmp[i] = 0;

tmp[i] |= (byte)(0x1F&local\_threshold[i]);

tmp[i] |= (byte)(mask\_bit[i] ? 1 << 7 : 0);

tmp[i] |= (byte)(test\_bit[i] ? 1 << 6 : 0);

}

return tmp;

}

}

public void SetAllConfigRegister(byte[] byte\_chip\_array)

{

int MessageID = 0x12;

Byte[] senddata = new Byte[8];

for (int i = 0; i < senddata.Length; i++)

senddata[i] = 0;

senddata[6] = (Byte)MessageID;

udpClient.Send(senddata, senddata.Length);

MemoryStream mem\_stream = new MemoryStream(byte\_chip\_array);

using (BinaryReader reader = new BinaryReader(mem\_stream))

{

for (int i = 0; i < 16; i++)

{

udpClient.Send(reader.ReadBytes(4096), 4096);

Thread.Sleep(3);

}

}

if (!WaitResponse(MessageID))

MessageBox.Show("NACK - " + MessageID.ToString());

}

public void ReadoutUnitHwCmdDebug(int ID, byte data)

{

int MessageID = 0x7;

Byte[] senddata = new Byte[8];

for (int i = 0; i < senddata.Length; i++)

senddata[i] = 0;

senddata[6] = (Byte)MessageID;

senddata[1] = (Byte)data;

senddata[0] = (Byte)ID;

udpClient.Send(senddata, senddata.Length);

if (!WaitResponse(MessageID))

MessageBox.Show("NACK - " + MessageID.ToString());

}

DAC – zápis:

0: Ibias\_Preamp\_ON

1: Ibias\_Preamp\_OFF

4: VPreamp\_NCAS

8: Ibias\_Ikrum

5: Vfbk

7: Vthreshold\_fine

6: Vthreshold\_coarse

2: LS\_ON

3: LS\_OFF

9: DISCPMOS  
10: DISCNMOS

11: DIsc

12: THS

13: VGND

14: VTP\_coarse

15: VTP\_fine

16: VCM\_SLVS

17: BIAS\_SLVS

// DAC values sending

// timhle to hodíš do readoutu

public void InternalDACSetting(int dacID, int dacValue)

{

int MessageID = 0x4;

Byte[] senddata = new Byte[8];

for (int i = 0; i < senddata.Length; i++)

senddata[i] = 0;

senddata[6] = (byte)MessageID;

senddata[4] = (Byte)dacID;

senddata[0] = (Byte)(dacValue & 0xFF);

senddata[1] = (Byte)((dacValue >> 8) & 0xFF);

senddata[2] = (Byte)((dacValue >> 16) & 0xFF);

senddata[3] = (Byte)((dacValue >> 24) & 0xFF);

udpClient.Send(senddata, senddata.Length);

if (!WaitResponse(MessageID))

MessageBox.Show("NACK - " + MessageID.ToString());

}

// DAC Update – takhle se to loadne přímo do čipu

udpSender.ReadoutUnitHwCmd(1);

DACs – čtení:  
0: Ibias\_Preamp\_ON

1: Ibias\_Preamp\_OFF

4: VPreamp\_NCAS

8: Ibias\_Ikrum

5: Vfbk

7: Vthreshold\_fine

6: Vthreshold\_coarse

2: LS\_ON

3: LS\_OFF

9: DISCPMOS  
10: DISCNMOS

11: DIsc

12: THS

13: VGND

14: VTP\_coarse

15: VTP\_fine

16: VCM\_SLVS

17: BIAS\_SLVS

25: BandGap

VDD = 3 \* ([19] - [20])

VDDA = 3 \* ([21] - [22])

Jak to načíst?

float[] tmp = udpSender.GetAllDACscan();

public float[] GetAllDACscan()

{

Byte[] senddata = new Byte[8];

for (int i = 0; i < senddata.Length; i++)

senddata[i] = 0;

senddata[6] = 0x14;

udpClient.Send(senddata, senddata.Length);

return WaitResponseAllGetDACscan();

}

public float[] WaitResponseAllGetDACscan()

{

UInt32 i = 0;

UInt64 tmp\_data = 0;

bool ack = false;

float dacscan\_value = 0;

int j = 0;

float[] dac\_scan\_results = new float[27];

//Thread.Sleep(5);

for (i = 0; i < 90000000; i++)

{

Application.DoEvents();

if (ReceiverCommandQueue.TryDequeue(out tmp\_data))

{

byte[] tmp\_data\_byte = BitConverter.GetBytes(tmp\_data);

if (tmp\_data\_byte[6] == 0xF)

{

dac\_scan\_results[j] = System.BitConverter.ToSingle(tmp\_data\_byte, 0);

j++;

if (j == 27)

{

ack = true;

break;

}

}

}

}

if (ack) return dac\_scan\_results;

else return (null);

}

Column trigger:

public void SetColumnTrgiger(int number\_columns, bool column\_trigger\_en)

{

int MessageID = 0x27;

Byte[] senddata = new Byte[8];

for (int i = 0; i < senddata.Length; i++)

senddata[i] = 0;

senddata[6] = (Byte)MessageID;

if(column\_trigger\_en)

senddata[2] = 1;

senddata[1] = (Byte)((number\_columns>>8)&0x1);

senddata[0] = (Byte)(number\_columns & 0xFF);

udpClient.Send(senddata, senddata.Length);

if (!WaitResponse(MessageID))

MessageBox.Show("bbbbNACK - 9");

}

Je v pohodě, že se mi teplota čipu ukazuje jako 16 degC? Nemá to ADC nějakou jinou aditivní/multiplikativní konstantu?

Heleď, to měření na Tpx2 je nějak pomrdaný, v ABS číslech to ukazuje asi blbosti, relativně to nějak sedí, již jsem řešil s návrhářem čipu, nic moc neporadil.

ToT, ToA frequncies:

Volime ToT (tu vybírá readout 5-120MHz), podle zvolené ToT f. pak odvodíme možnosti pro ToA.

double[] tot\_clocks = new double[] { 120, 100, 80, 50, 25, 10, 5 };

toa\_clocks = new double[32];

Pro zvolení tot\_clock z pole tot\_clocks pak pocitam toa\_clocks (klasické dělení)

//double toa\_cloc

for(int i=0; i < 31; i++)

{

toa\_clock = tot\_clock / (double)(1 << i);

toa\_clocks[i] = toa\_clock;

if (toa\_clock > 1)

comboBoxToAClock.Items.Add(toa\_clock.ToString("0.##") + "MHz");

else if (toa\_clock > 0.1)

comboBoxToAClock.Items.Add((toa\_clock \* 1000).ToString("0.##") + "kHz");

else

comboBoxToAClock.Items.Add((toa\_clock \* 1000000).ToString("0.##") + "Hz");

}

// -----------------------------------------

// Frequencies

int mclock = comboBoxMclock.SelectedIndex; // ToT

int toaclock = comboBoxToAClock.SelectedIndex;

float toa\_clock\_value = (float)(toa\_clocks[toaclock]);

if(toaclock==0)

toaclock = 0x1E;

else if(toaclock==31)

toaclock = 0x1F;

else

toaclock = toaclock-1;

udpSender.SetFrequencies(toa\_clock\_value, toaclock, mclock);

MClock:

0: 120MHz

1: 100MHz

2: 80MHz

3: 50MHz

4: 25MHz

5: 10MHz

6: 5MHz

7: CLK OFF

public void SetFrequencies(float toa\_value, int toa\_data, int mclock\_data)

{

Byte[] senddata = new Byte[8];

int MessageID = 0x6;

CommandQueueClear();

for (int i = 0; i < senddata.Length; i++)

senddata[i] = 0;

senddata[6] = (byte)MessageID;

byte[] tmp\_array = BitConverter.GetBytes(toa\_value);

senddata[2] = tmp\_array[0];

senddata[3] = tmp\_array[1];

senddata[4] = tmp\_array[2];

senddata[5] = tmp\_array[3];

senddata[0] = (byte)toa\_data;

senddata[1] = (byte)mclock\_data;

udpClient.Send(senddata, senddata.Length);

if (!WaitResponse(MessageID))

MessageBox.Show("NACK");

}

Asi ti přijde nelogické proč do readoutu posílat tu float hodnotu, ale z nějakého důvodu je to tam užitečné…