Upgading BCM1f:

Deconvolution

FPGA

Piotr Burtowy
PH-CMX-DS

23.04.2013r

Agenda

1. Motivation

2. Deconvolution

- Function in Root
- SIMPAS - Succesive Identify Max Position And Subtract
- Deconvolution vs. SIMPAS

3. FPGA

- Data flow – block diagram
- Peak generator
  - Non-saturated peaks
  - Saturated peaks, moving averages
- Other blocks
- Speed of calculations

4. Summary – what is done

5. Next steps

Agenda

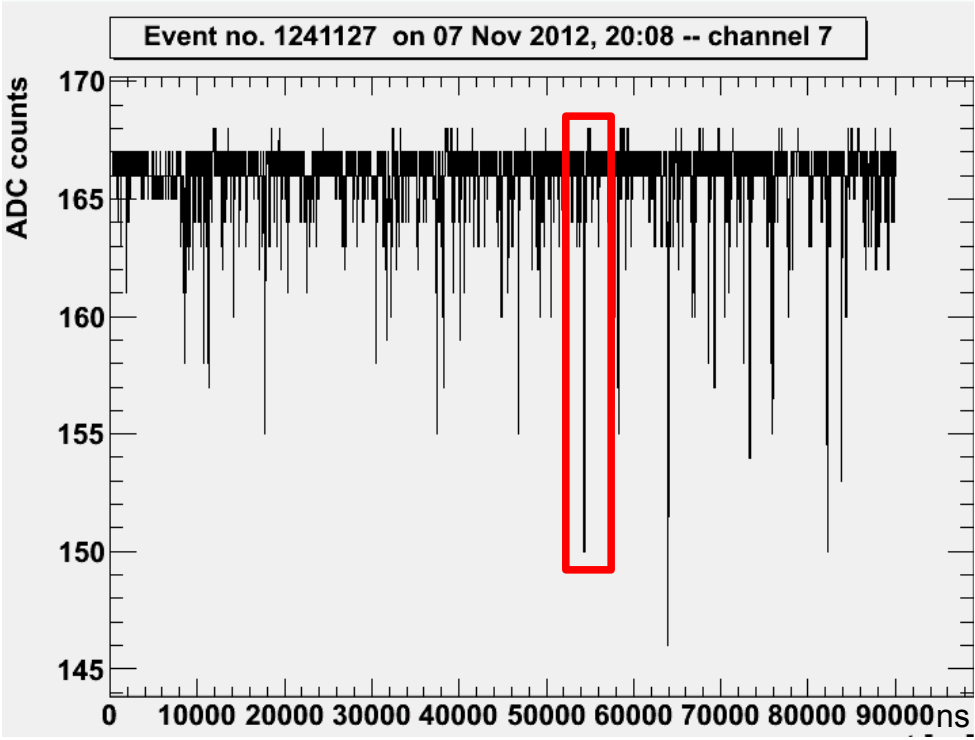
1. Motivation
2. Deconvolution
   - Function in Root
   - SIMPAS - Succesive Identify Max Position And Subtract
   - Deconvolution vs. SIMPAS
3. FPGA
   - Data flow – block diagram
   - Peak generator
     - Non-saturated peaks
     - Saturated peaks, moving averages
   - Other blocks
   - Speed of calculations
4. Summary – what is done
5. Next steps

Motivation

1. Implement algorithm to identify peaks:
   - time arrival
   - pulse height
2. Choose algorithm:
   - Real time – **deadtime free**
   - Efficient – identify overlapping peaks
3. Algorithm should be:
   - **Reliable**
   - **Stable**

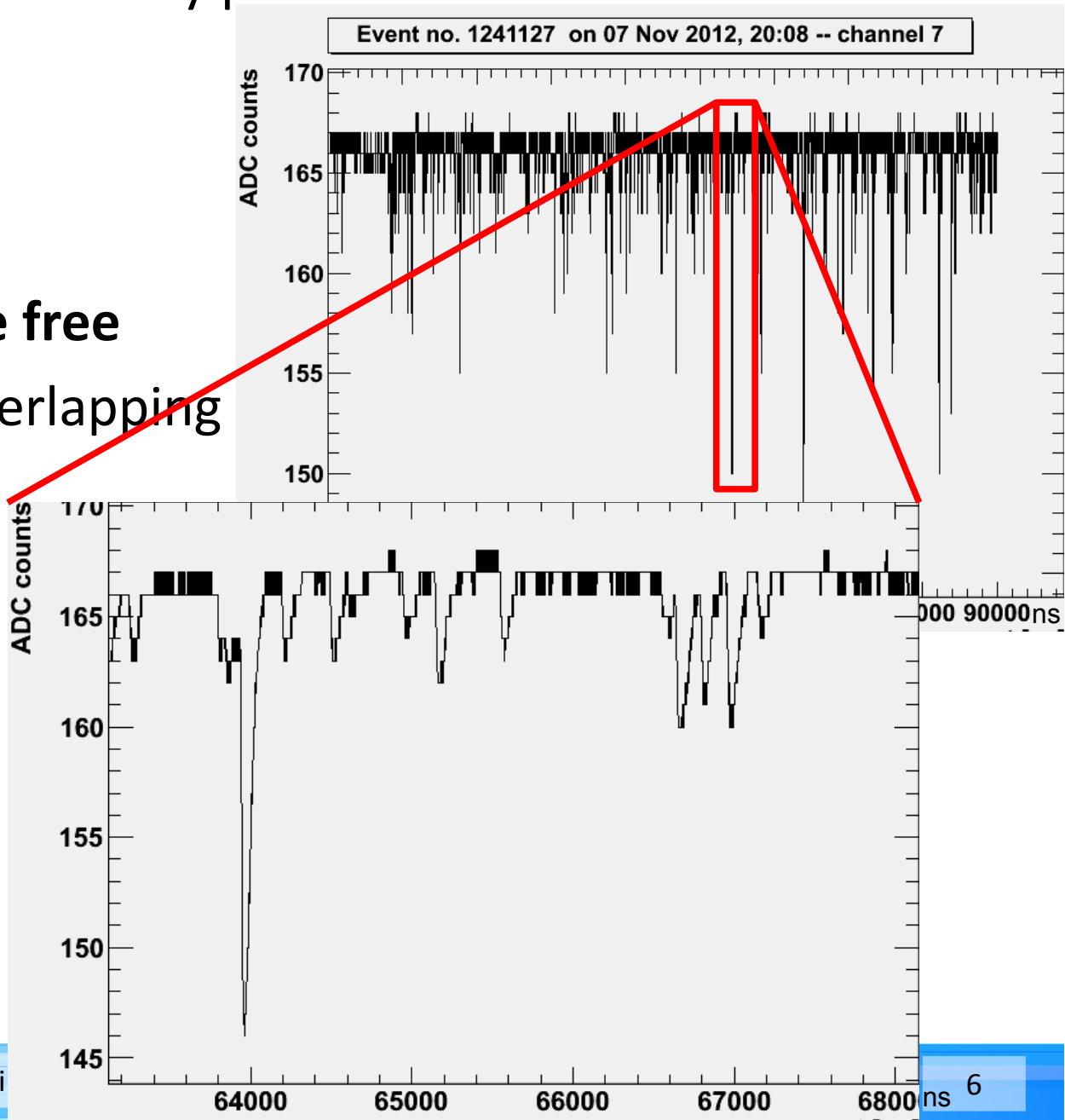
Motivation

1. Implement algorithm to identify peaks:
   - time arrival
   - pulse height
2. Choose algorithm:
   - Real time – **deadtime free**
   - Efficient – identify overlapping peaks
3. Algorithm should be:
   - **Reliable**
   - **Stable**



Motivation

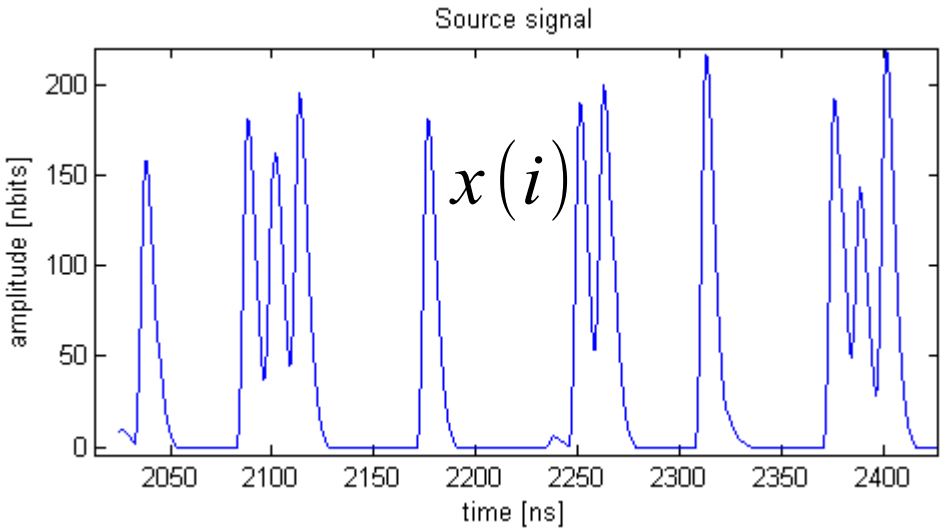
1. Implement algorithm to identify peaks:
   - time arrival
   - pulse height
2. Choose algorithm:
   - Real time – **deadtime free**
   - Efficient – identify overlapping peaks
3. Algorithm should be:
   - **Reliable**
   - **Stable**



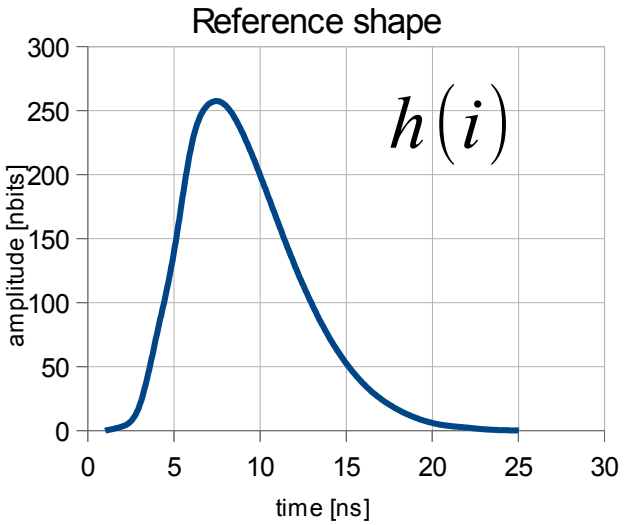
Agenda

1. Motivation
2. **Deconvolution**
   - Function in Root
   - SIMPAS - Succesive Identify Max Position And Subtract
   - Deconvolution vs. SIMPAS
3. FPGA
   - Data flow – block diagram
   - Peak generator
     - Non-saturated peaks
     - Saturated peaks, moving averages
   - Other blocks
   - Speed of calculations
4. **Summary – what is done**
5. **Next steps**

Deconvolution - Root

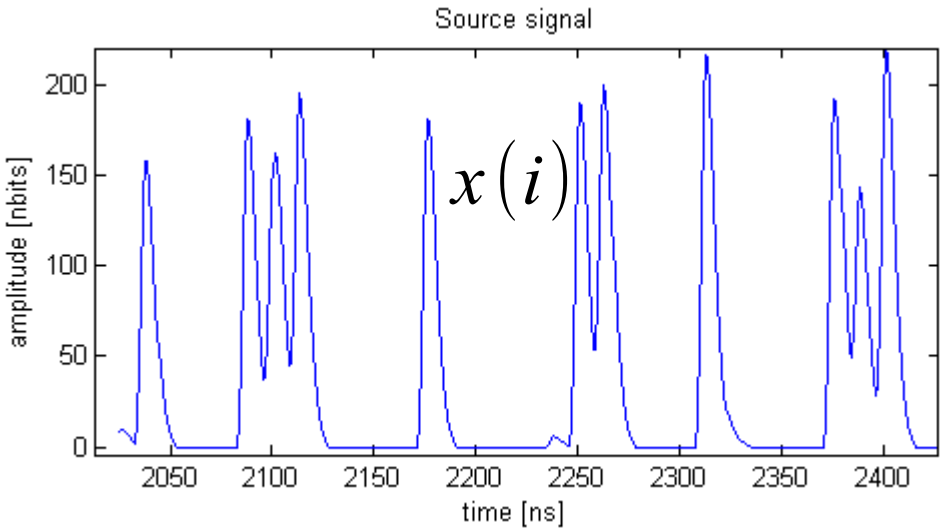


$y(i)$ – Output spectrum
$h(i)$ – Reference signal
$x(i)$ – Source signal



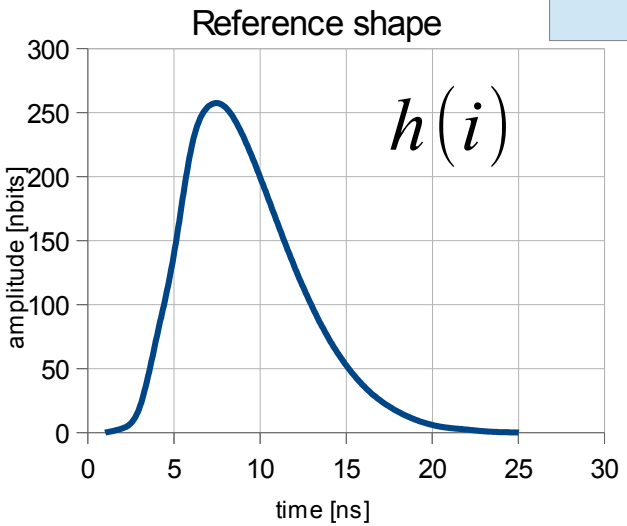
Thanks R. Walsh for initial example codes!!

Deconvolution - Root



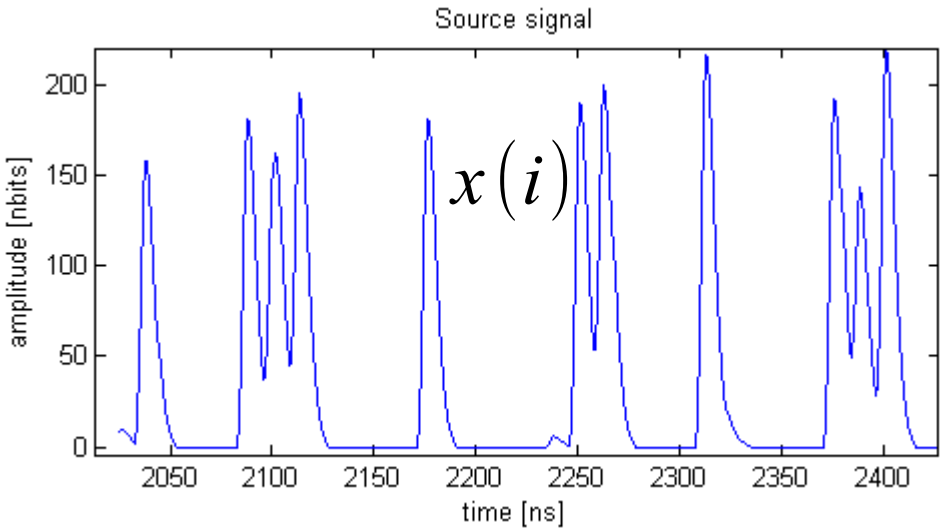
$y(i)$ – Output spectrum
$h(i)$ – Reference signal
$x(i)$ – Source signal

$$y(i) = \sum_{k=0}^{N-1} h(i-k)\,x(k),$$
$$i = 0, 1, 2, \ldots N-1$$



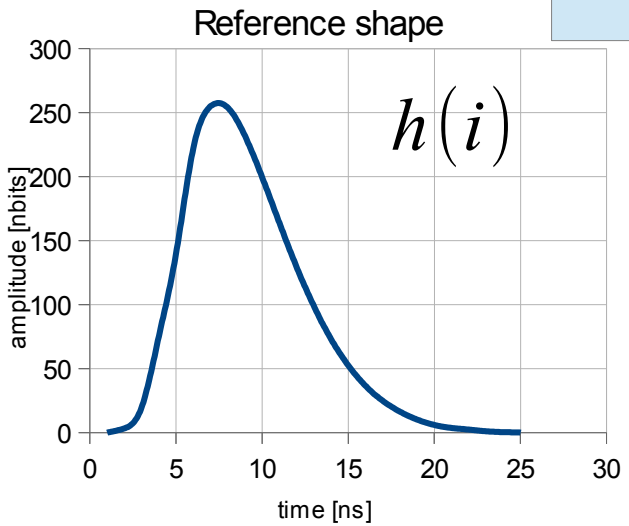
Thanks R. Walsh for initial example codes!!

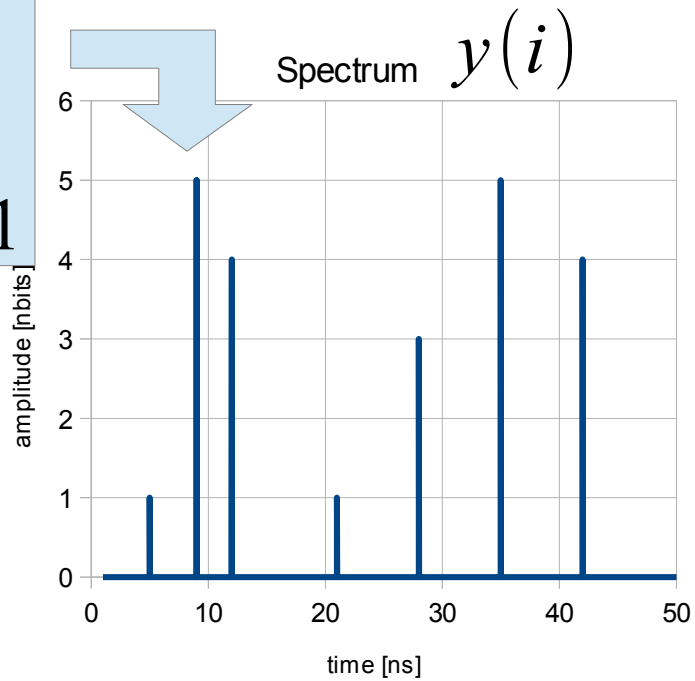
Deconvolution - Root



$y(i)$ – Output spectrum
$h(i)$ – Reference signal
$x(i)$ – Source signal

$$y(i) = \sum_{k=0}^{N-1} h(i-k)\,x(k),$$
$$i = 0,1,2,\ldots N-1$$





Thanks R. Walsh for initial example codes!!

Deconvolution - Root

Algorithm:

0. set x^0, nRepetitions, nIterations, boost

1. repeat nRepetitions times:

   1a. $x_i^{(k)} := [x_i^{(k)}]^{boost}$

   1b. Repeat nIterations times: $x^{(k+1)} := x^{(k)} \dfrac{y'}{\sum_{m=0}^{N-1} H'_{im} x_m^{(k)}}$

Deconvolution - Root

Algorithm:

0. set x^0, nRepetitions, nIterations, boost

1. repeat nRepetitions times:

   1a. $x_i^{(k)} := [x_i^{(k)}]^{boost}$

   1b. Repeat nIterations times: $x^{(k+1)} := x^{(k)} \dfrac{y'}{\sum_{m=0}^{N-1} H'_{im} x_m^{(k)}}$

Deconvolution(reference,source,size,nIterations,nRepetition,boost)

	nIterations	nRepetitions	boost	Total = nIterations*nRepetitions	quality
#1	1000	1	1	1000	ok
#2	50	5	3	250	ok
#3	10	10	1	100	
#4	10	10	5	100	
#5	10	5	3	50	ok

Total number of iterations = time consuming!

SIMPAS - Succesive Identify Max Position And Subtract

0. Get Reference and Source signal, thereshold //= expected noise
1. max(Source) // seeking from right side of Source
2. Source = Source – Reference
   // Xpos(max(Reference)) = Xpos(max(Source))
3. Do (1 and 2) until (Source >= thereshold)

SIMPAS - Succesive Identify Max Position And Subtract

0. Get Reference and Source signal, thereshold //= expected noise
1. max(Source) // seeking from right side of Source
2. Source = Source – Reference
   // Xpos(max(Reference)) = Xpos(max(Source))
3. Do (1 and 2) until (Source >= thereshold)

Reference = [2 3 1]

Source = [	0	6	9	5	3	1	2	3	]
Peaks = [	0	0	**3**	0	**1**	0	0	**1**	]

SIMPAS - Succesive Identify Max Position And Subtract

0. Get Reference and Source signal, thereshold //= expected noise
1. max(Source) // seeking from right side of Source
2. Source = Source – Reference
   // Xpos(max(Reference)) = Xpos(max(Source))
3. Do (1 and 2) until (Source >= thereshold)

Reference = [2 3 1]

Source = [0 6 9 5 3 1 2 3]
Peaks = [0 0 **3** 0 **1** 0 0 **1**]

1. Source = [0 6 **9** 5 3 1 2 3] // maxX = 3

maxX	N times
3	1

SIMPAS - Succesive Identify Max Position And Subtract

0. Get Reference and Source signal, thereshold //= expected noise
1. max(Source) // seeking from right side of Source
2. Source = Source – Reference
   // Xpos(max(Reference)) = Xpos(max(Source))
3. Do (1 and 2) until (Source >= thereshold)

Reference = [2 3 1]

Source = [0 6 9 5 3 1 2 3]
Peaks = [0 0 **3** 0 **1** 0 0 **1**]

1. Source = [0 6 **9** 5 3 1 2 3] // maxX = 3
   - Reference 2 3 1
2. Source = [0 4 **6** 4 3 1 2 3] // maxX = 3

maxX	N times
3	2

SIMPAS - Succesive Identify Max Position And Subtract

0. Get Reference and Source signal, thereshold //= expected noise
1. max(Source) // seeking from right side of Source
2. Source = Source – Reference
   // Xpos(max(Reference)) = Xpos(max(Source))
3. Do (1 and 2) until (Source >= thereshold)

Reference = [2 3 1]

Source = [0 6 9 5 3 1 2 3]
Peaks = [0 0 **3** 0 **1** 0 0 **1**]

1. Source = [0 6 **9** 5 3 1 2 3] // maxX = 3
   - Reference 2 3 1
2. Source = [0 4 **6** 4 3 1 2 3] // maxX = 3
   - Reference 2 3 1
3. Source = [0 2 3 3 3 1 2 **3**] // maxX = 8

maxX	N times
3	2
8	1

SIMPAS - Succesive Identify Max Position And Subtract

0. Get Reference and Source signal, thereshold //= expected noise
1. max(Source) // seeking from right side of Source
2. Source = Source – Reference
   // Xpos(max(Reference)) = Xpos(max(Source))
3. Do (1 and 2) until (Source >= thereshold)

Reference = [2 3 1]

Source = [0 6 9 5 3 1 2 3]
Peaks = [0 0 **3** 0 **1** 0 0 **1**]

1. Source = [0 6 **9** 5 3 1 2 3] // maxX = 3
   - Reference 2 3 1
2. Source = [0 4 **6** 4 3 1 2 3] // maxX = 3
   - Reference 2 3 1
3. Source = [0 2 3 3 3 1 2 **3**] // maxX = 8
   - Reference 2 3 (1)
4. Source = [0 2 3 3 **3** 1 0 0] // maxX = 5

maxX	N times
3	2
5	1
8	1

SIMPAS - Succesive Identify Max Position And Subtract

0. Get Reference and Source signal, thereshold //= expected noise
1. max(Source) // seeking from right side of Source
2. Source = Source – Reference
   // Xpos(max(Reference)) = Xpos(max(Source))
3. Do (1 and 2) until (Source >= thereshold)

Reference = [2 3 1]

Source = [0 6 9 5 3 1 2 3]
Peaks = [0 0 **3** 0 **1** 0 0 **1**]

1. Source = [0 6 **9** 5 3 1 2 3] // maxX = 3
 - Reference 2 3 1
2. Source = [0 4 **6** 4 3 1 2 3] // maxX = 3
 - Reference 2 3 1
3. Source = [0 2 3 3 3 1 2 **3**] // maxX = 8
 - Reference 2 3 (1)
4. Source = [0 2 3 3 **3** 1 0 0] // maxX = 5
 - Reference 2 3 1
5. Source = [0 2 **3** 1 0 0 0 0] // maxX =3

maxX	N times
3	3
5	1
8	1

SIMPAS - Succesive Identify Max Position And Subtract

0. Get Reference and Source signal, thereshold //= expected noise
1. max(Source) // seeking from right side of Source
2. Source = Source – Reference
   // Xpos(max(Reference)) = Xpos(max(Source))
3. Do (1 and 2) until (Source >= thereshold)

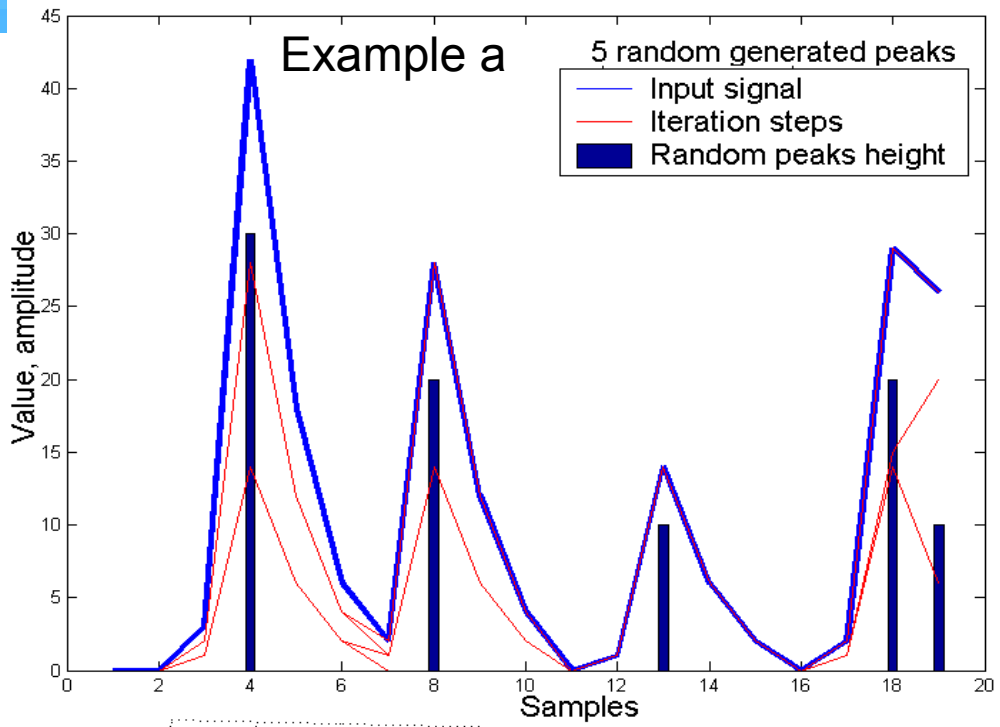
Reference = [2 3 1]

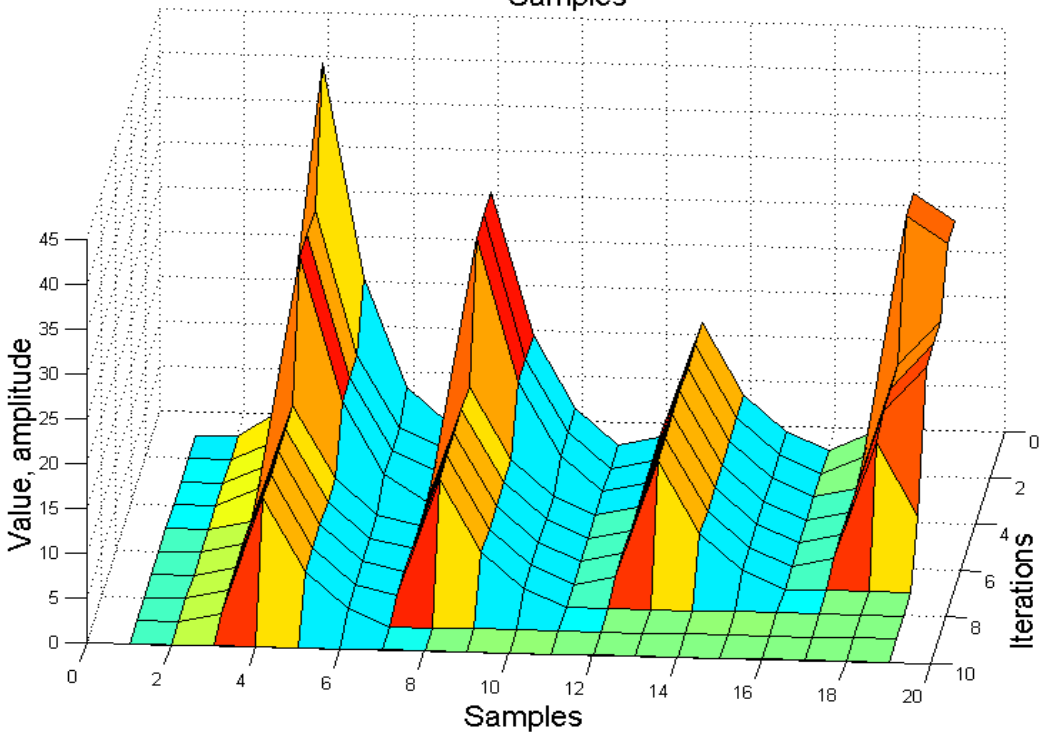
Source = [0 6 9 5 3 1 2 3]
Peaks = [0 0 **3** 0 **1** 0 0 **1**]

1. Source = [0 6 **9** 5 3 1 2 3] // maxX = 3
   - Reference 2 3 1
2. Source = [0 4 **6** 4 3 1 2 3] // maxX = 3
   - Reference 2 3 1
3. Source = [0 2 3 3 3 1 2 **3**] // maxX = 8
   - Reference 2 3 (1)
4. Source = [0 2 3 3 **3** 1 0 0] // maxX = 5
   - Reference 2 3 1
5. Source = [0 2 **3** 1 0 0 0 0] // maxX =3
   - Reference 2 3 1
6. Source = [0 0 0 0 0 0 0 0] // finish

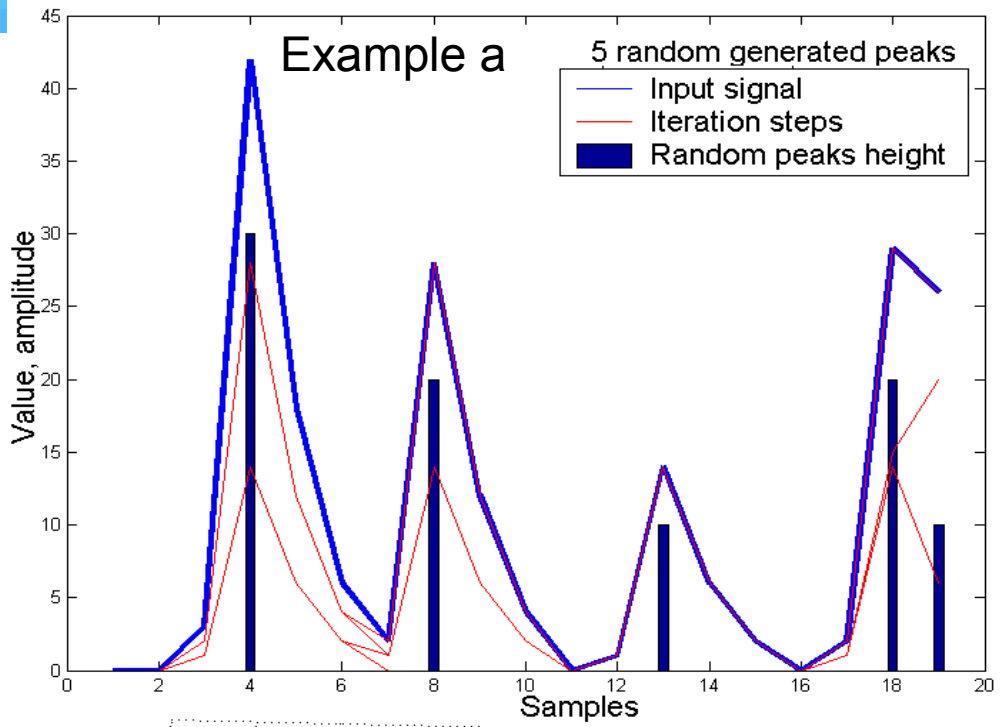
maxX	N times
3	3
5	1
8	1

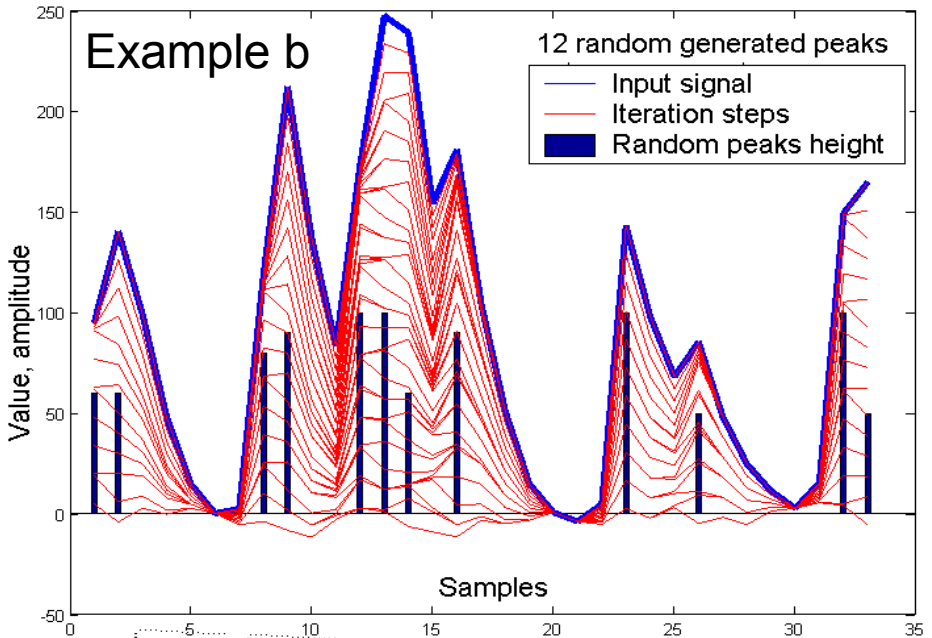
SIMPAS - Succesive Identify Max Position And Subtract

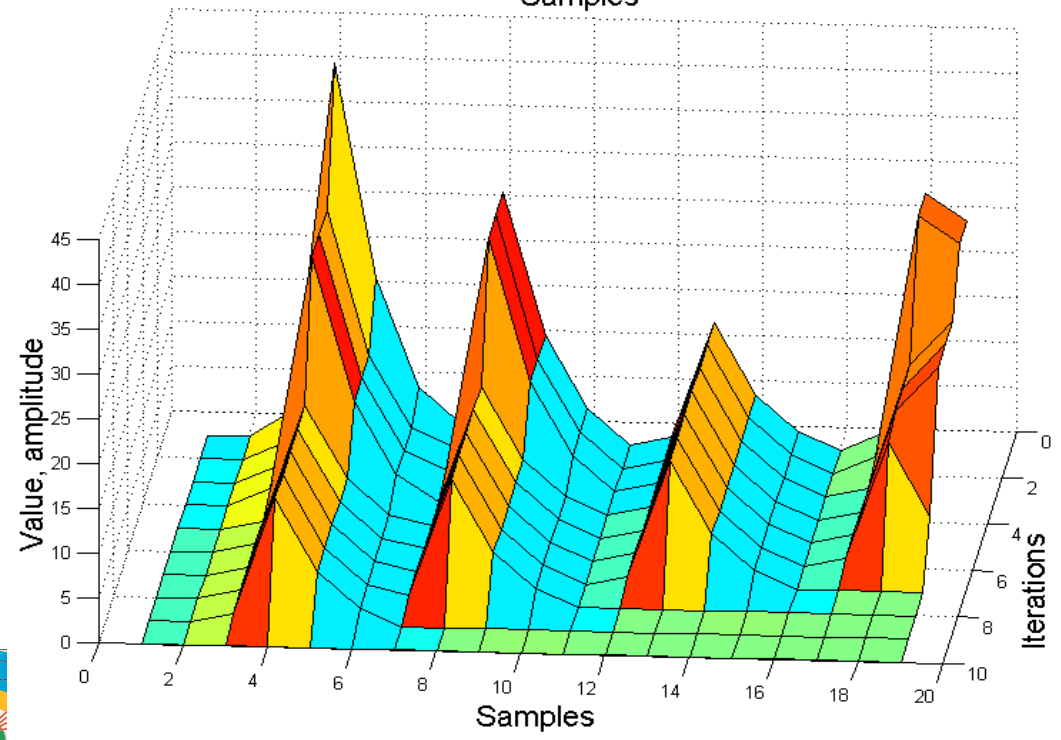


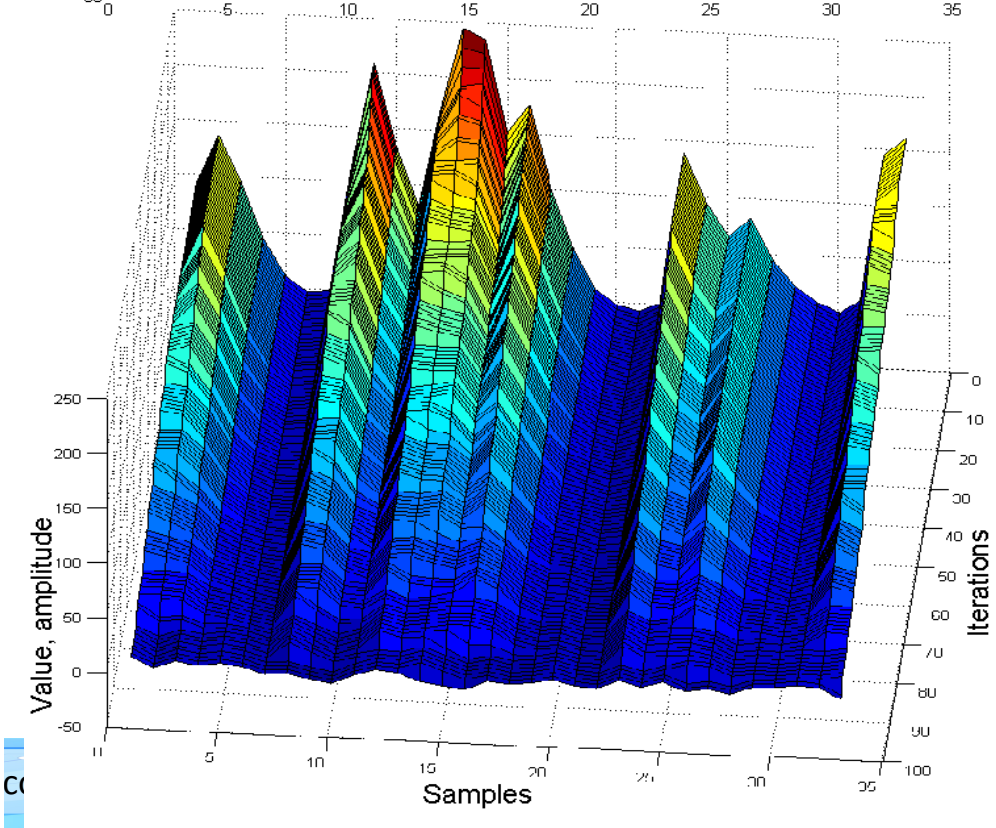


SIMPAS - Succesive Identify Max Position And Subtract



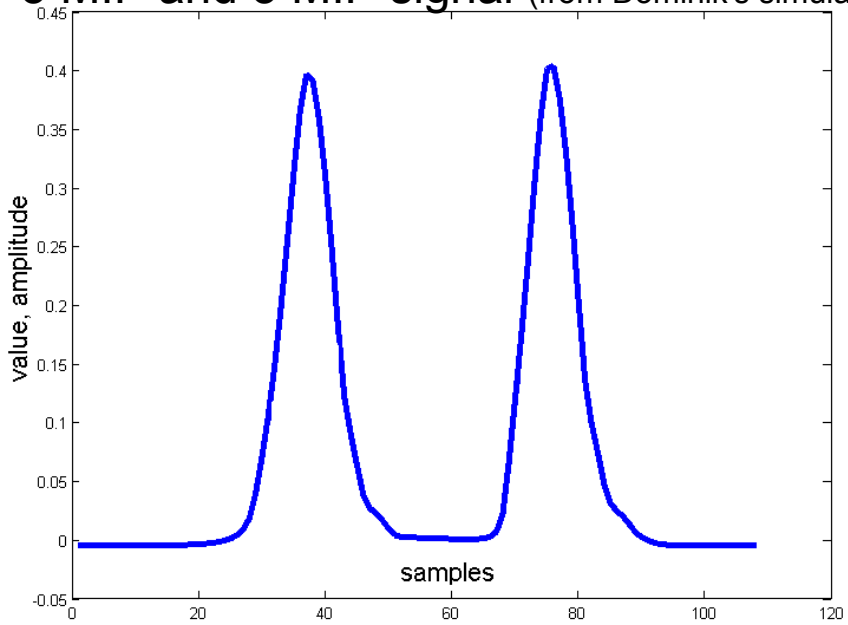


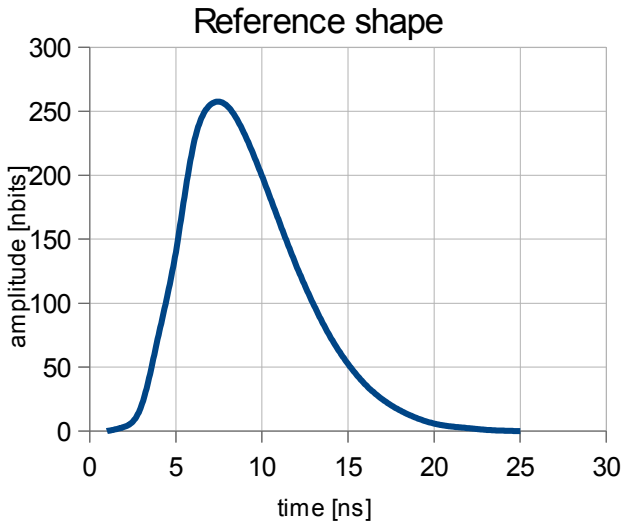




SIMPAS - Succesive Identify Max Position And Subtract Example

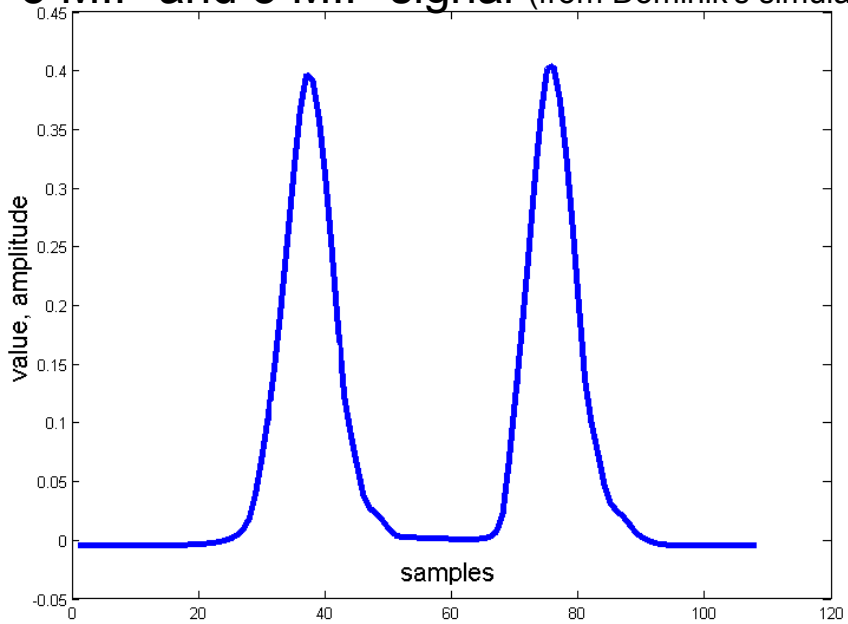
3 MIP and 3 MIP signal (from Dominik's simulations)

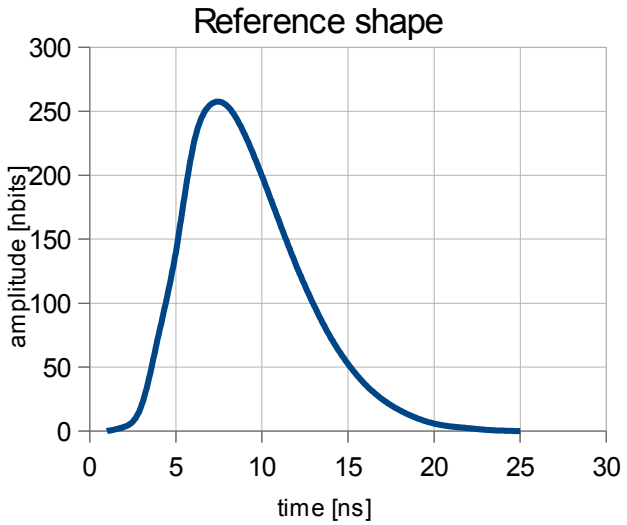




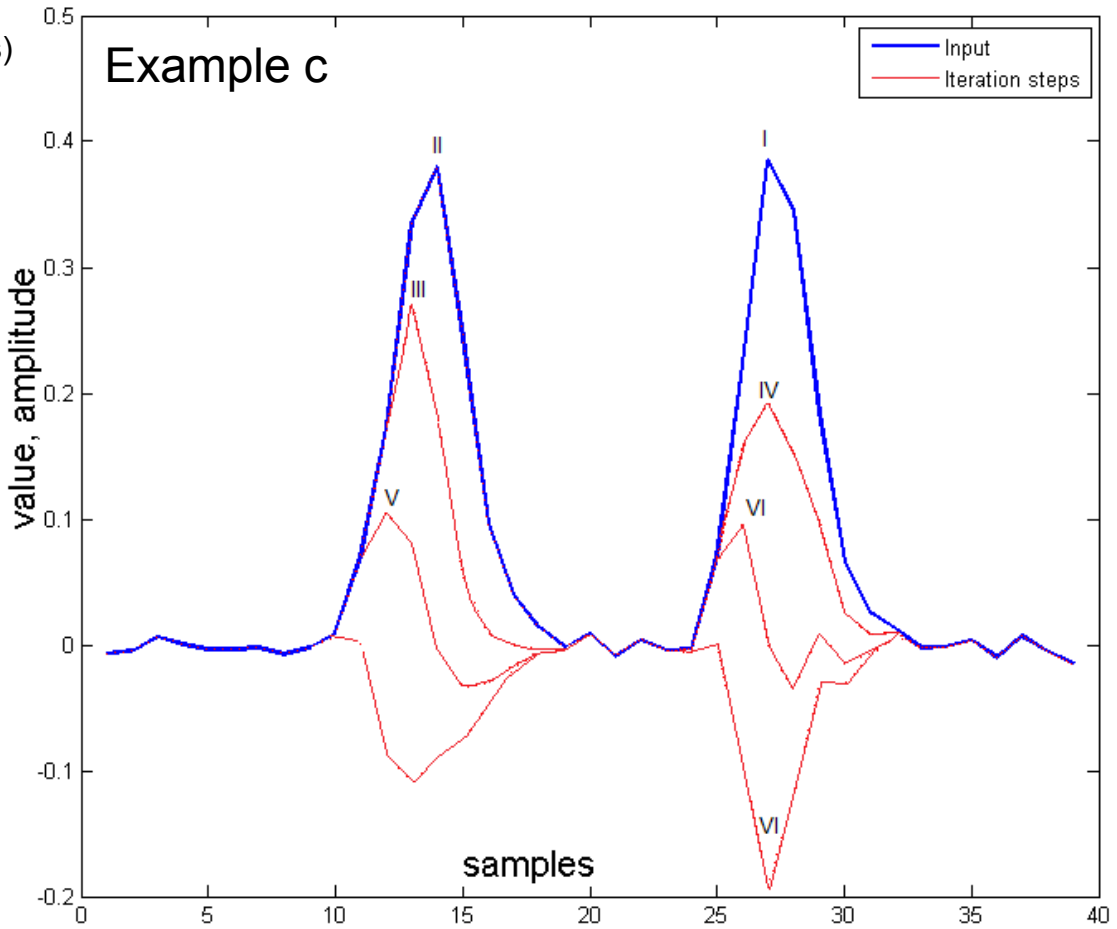
SIMPAS - Succesive Identify Max Position And Subtract

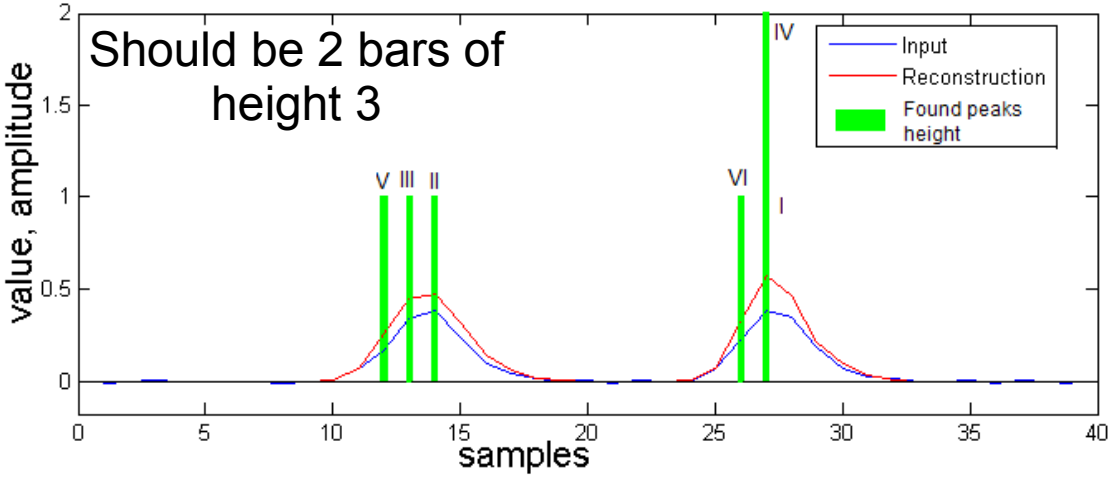
3 MIP and 3 MIP signal (from Dominik's simulations)





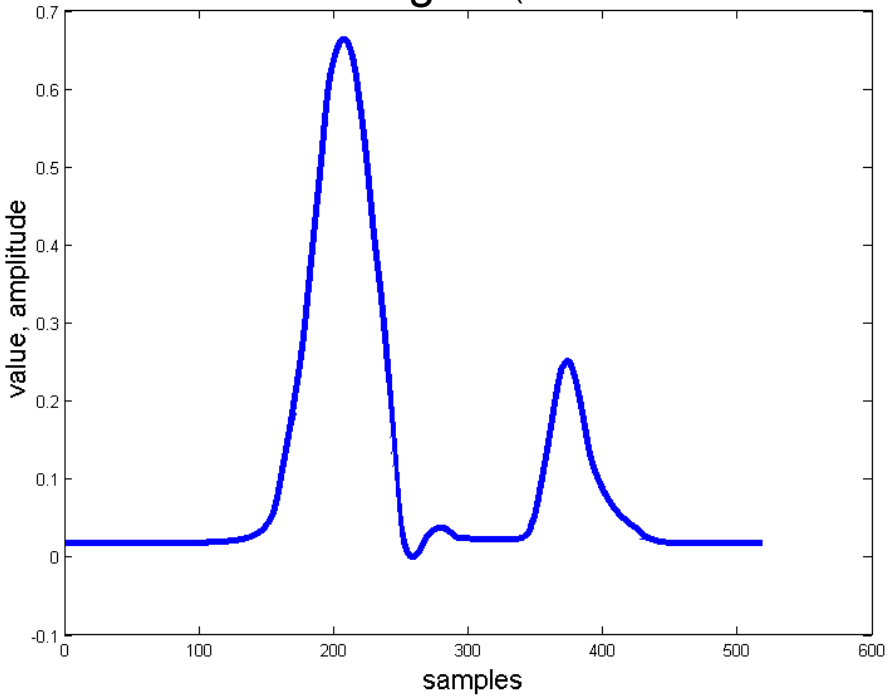
Example

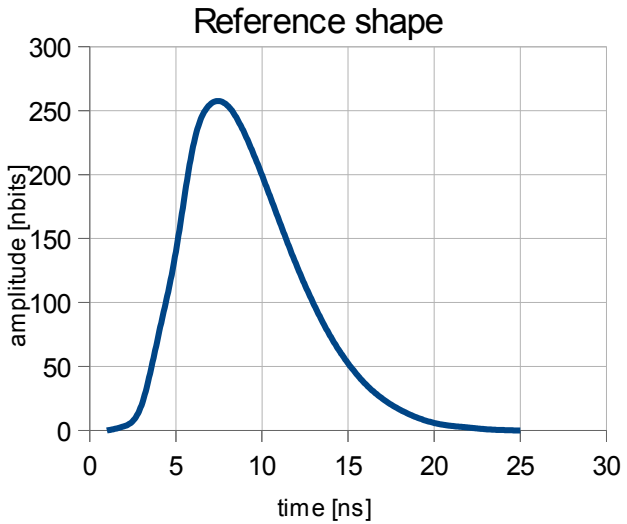




SIMPAS - Succesive Identify Max Position And Subtract

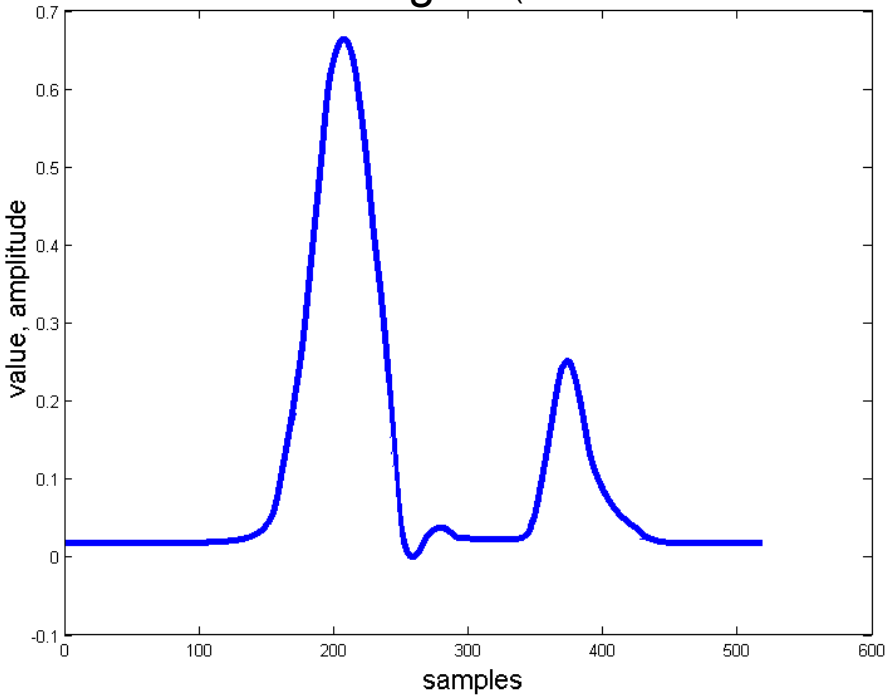
4 MIP and 1 MIP signal (from Dominik's simulations)

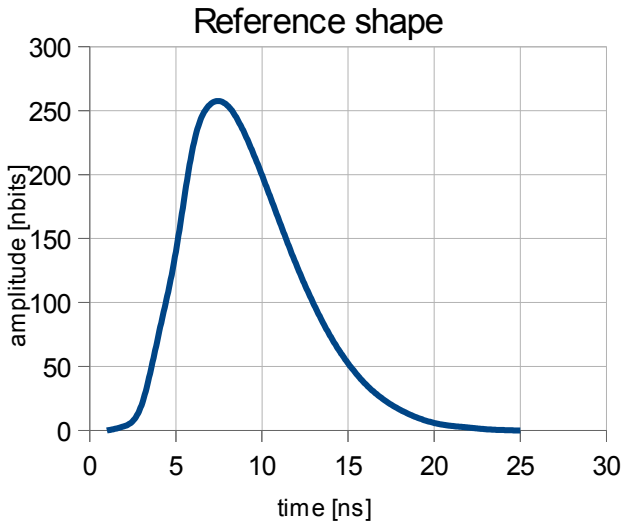


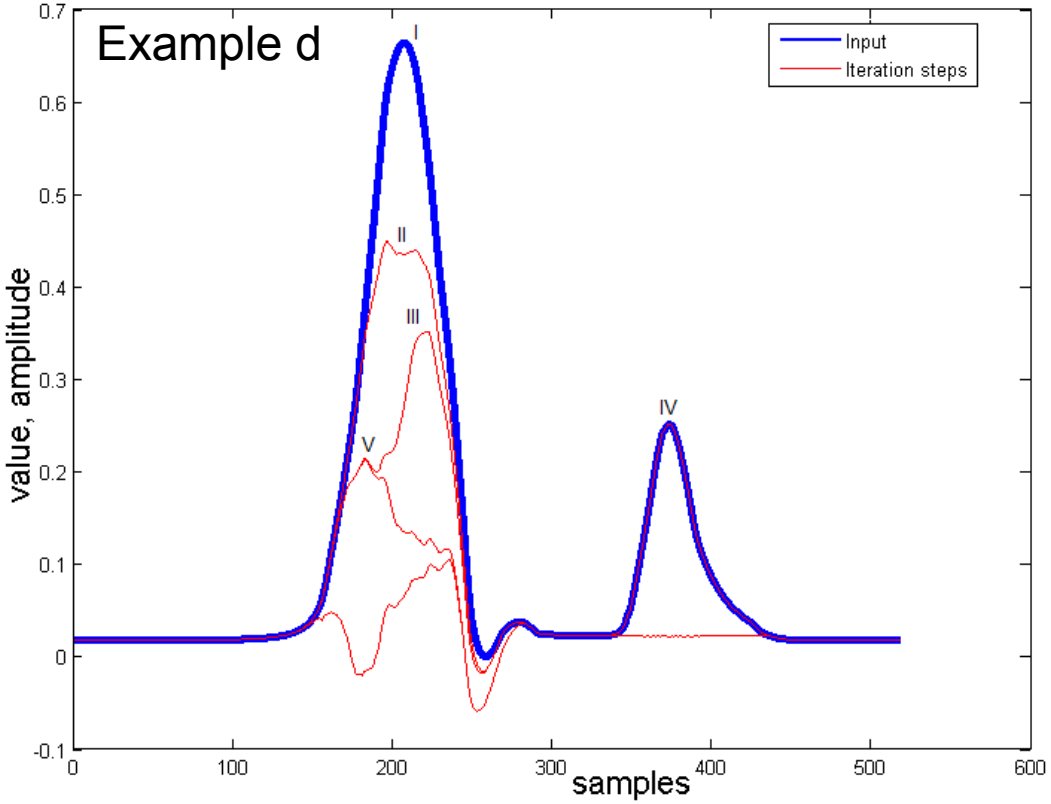


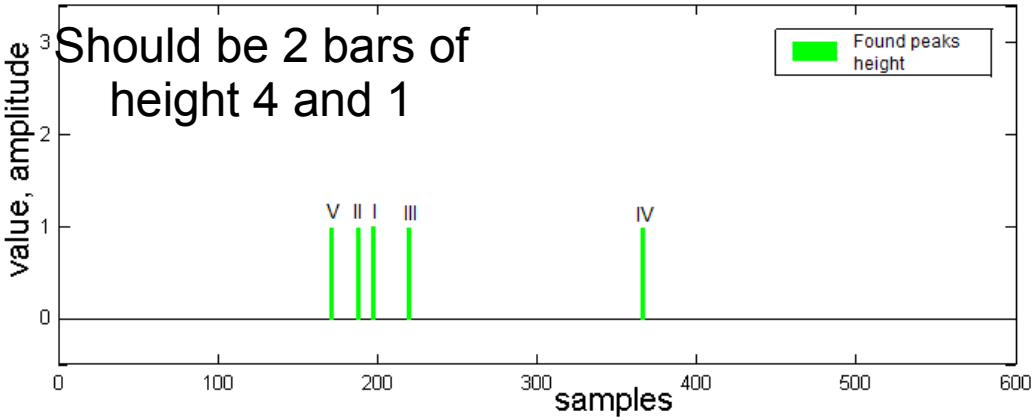
SIMPAS - Succesive Identify Max Position And Subtract

4 MIP and 1 MIP signal (from Dominik's simulations)

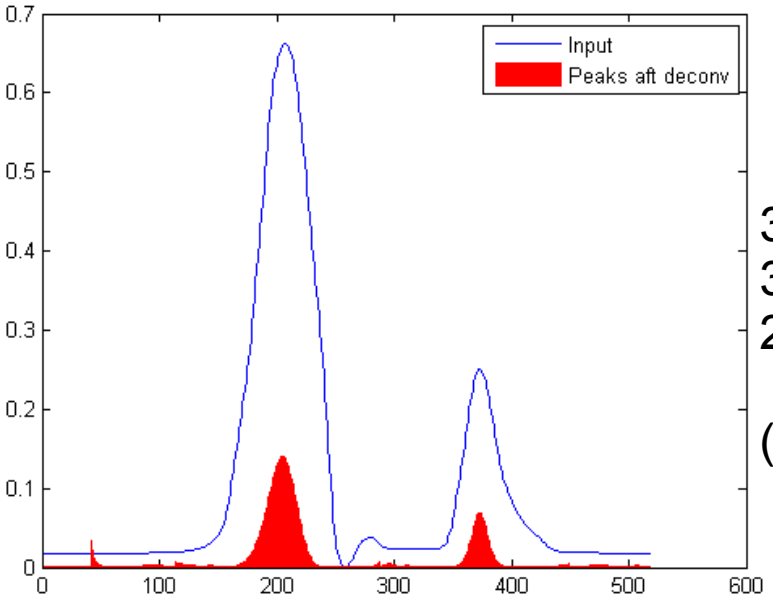






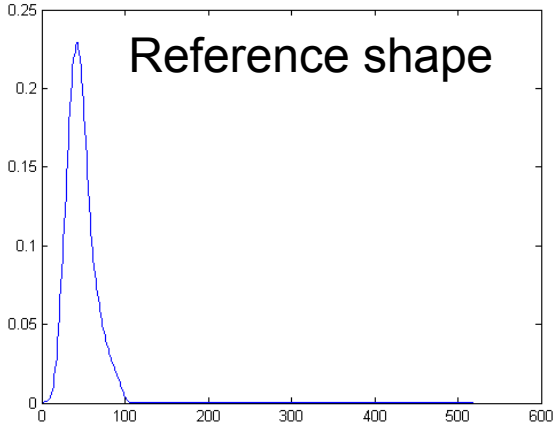


Deconvolution vs. SIMPAS

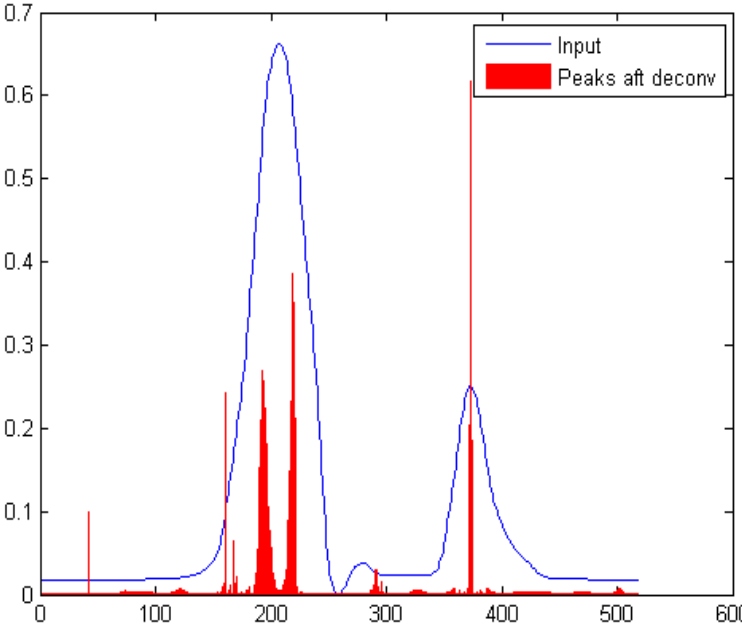


Deconvolution:
3 iterations
3 repetitions
2 boost

(total 9 rep.)

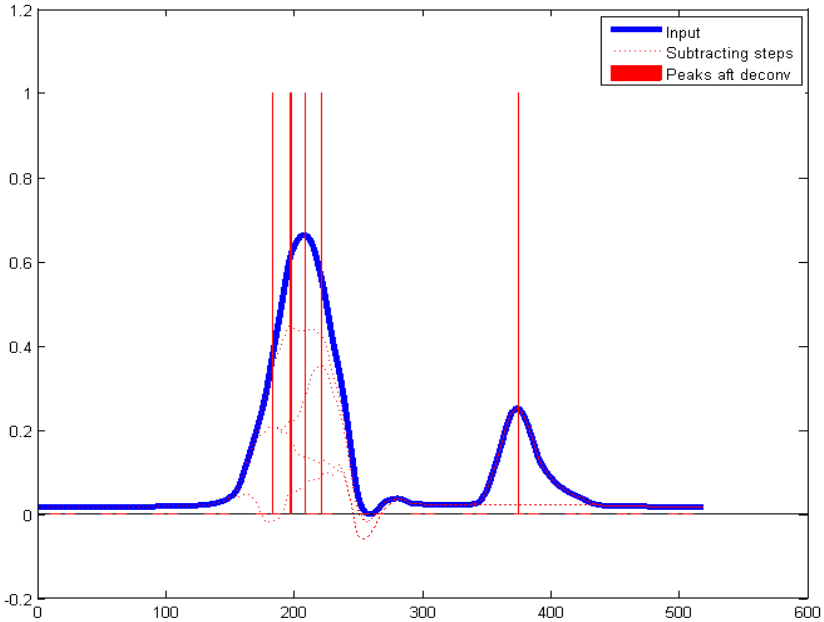


SIMPAS: 5 subtraction



Deconvolution:
70 iterations
5 repetitions
3 boost

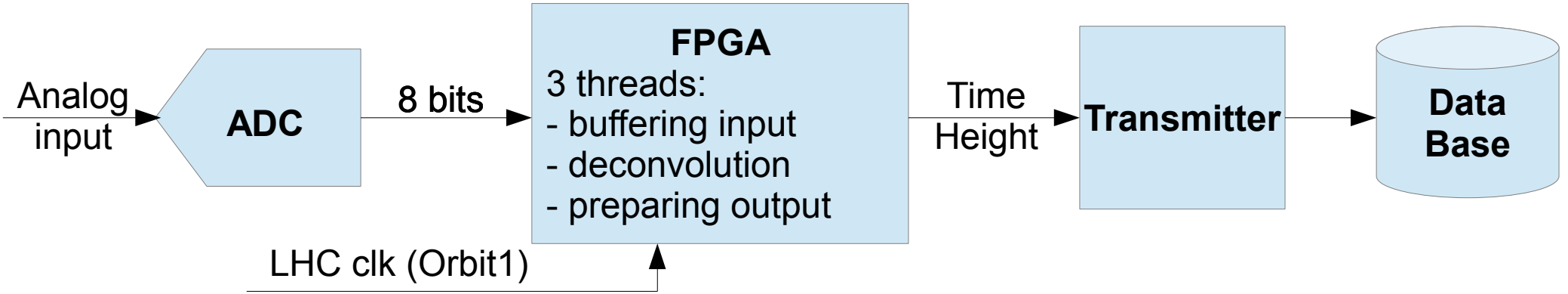
(total 350 rep.)



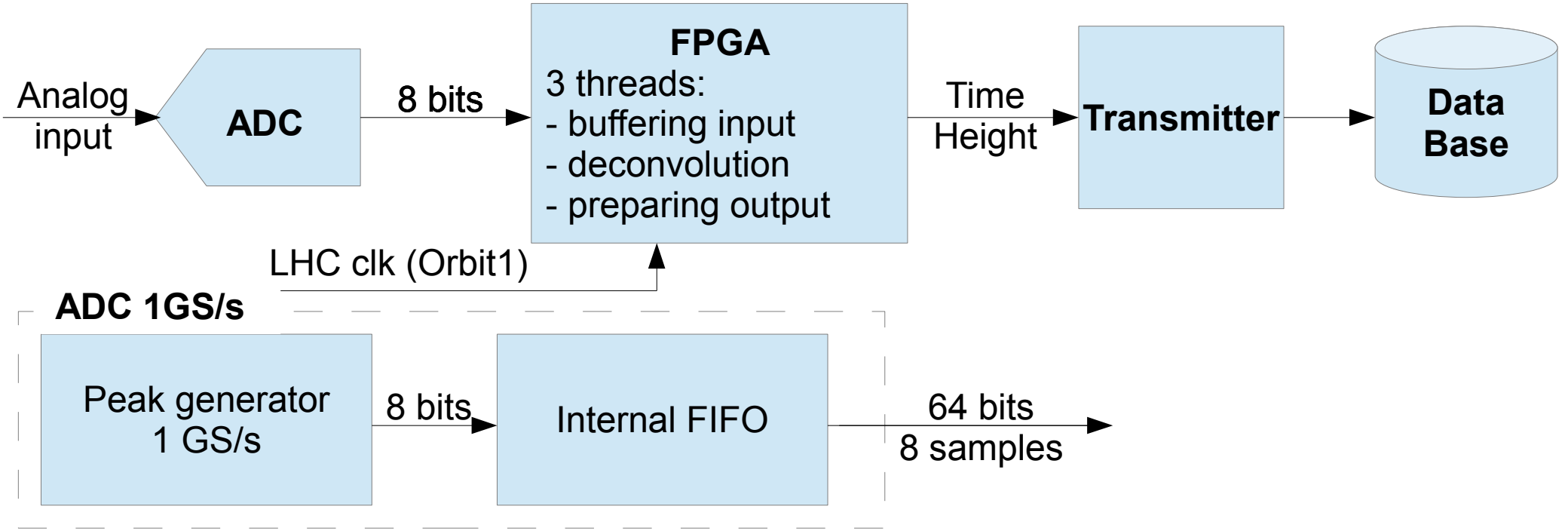
Agenda

1. Motivation
2. Deconvolution
   - Function in Root
   - SIMPAS - Succesive Identify Max Position And Subtract
   - Deconvolution vs. SIMPAS
3. **FPGA**
   - Data flow – block diagram
   - Peak generator
     - Non-saturated peaks
     - Saturated peaks, moving averages
   - Other blocks
   - Speed of calculations
4. Summary – what is done
5. Next steps

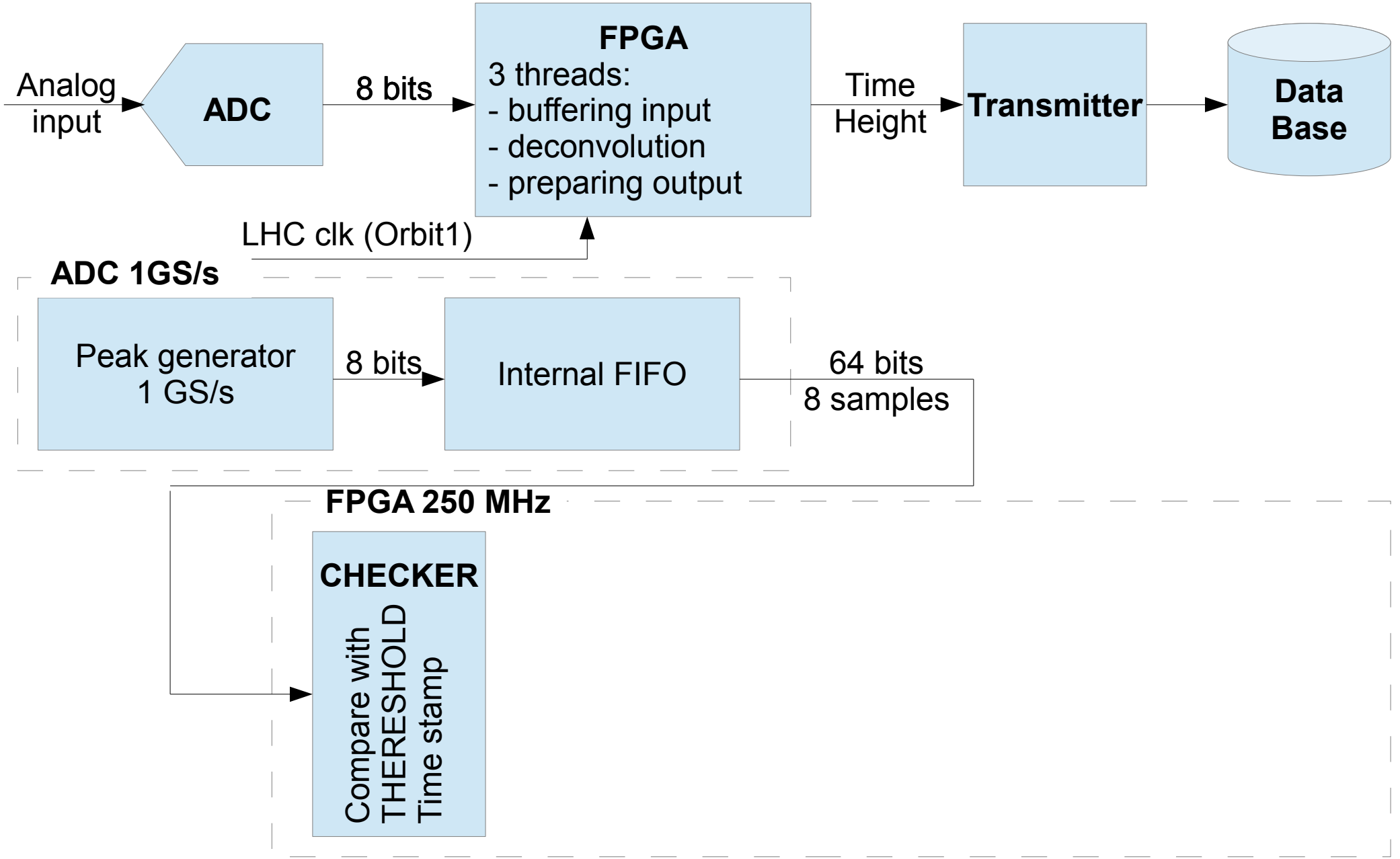
FPGA - data flow



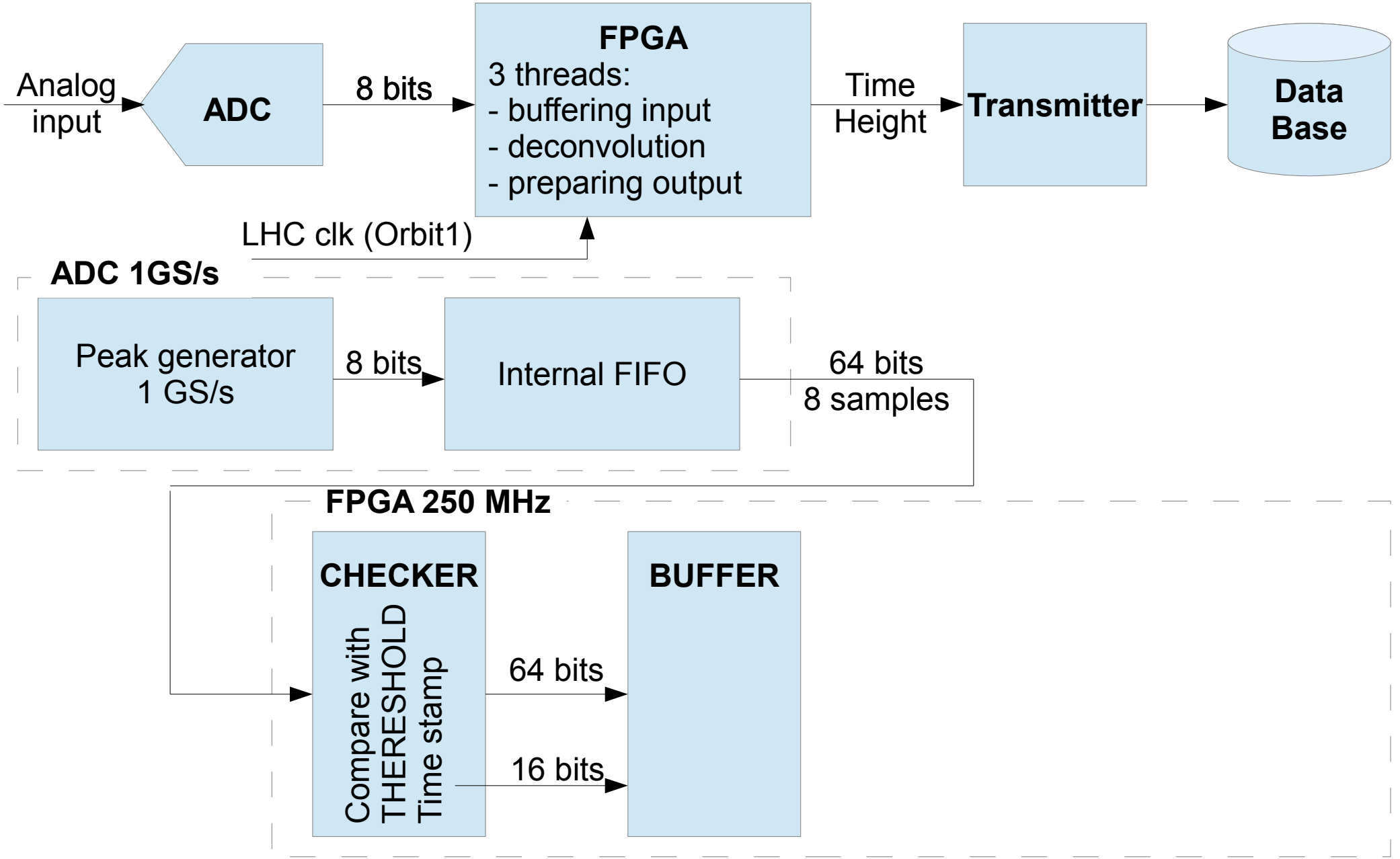
FPGA - data flow



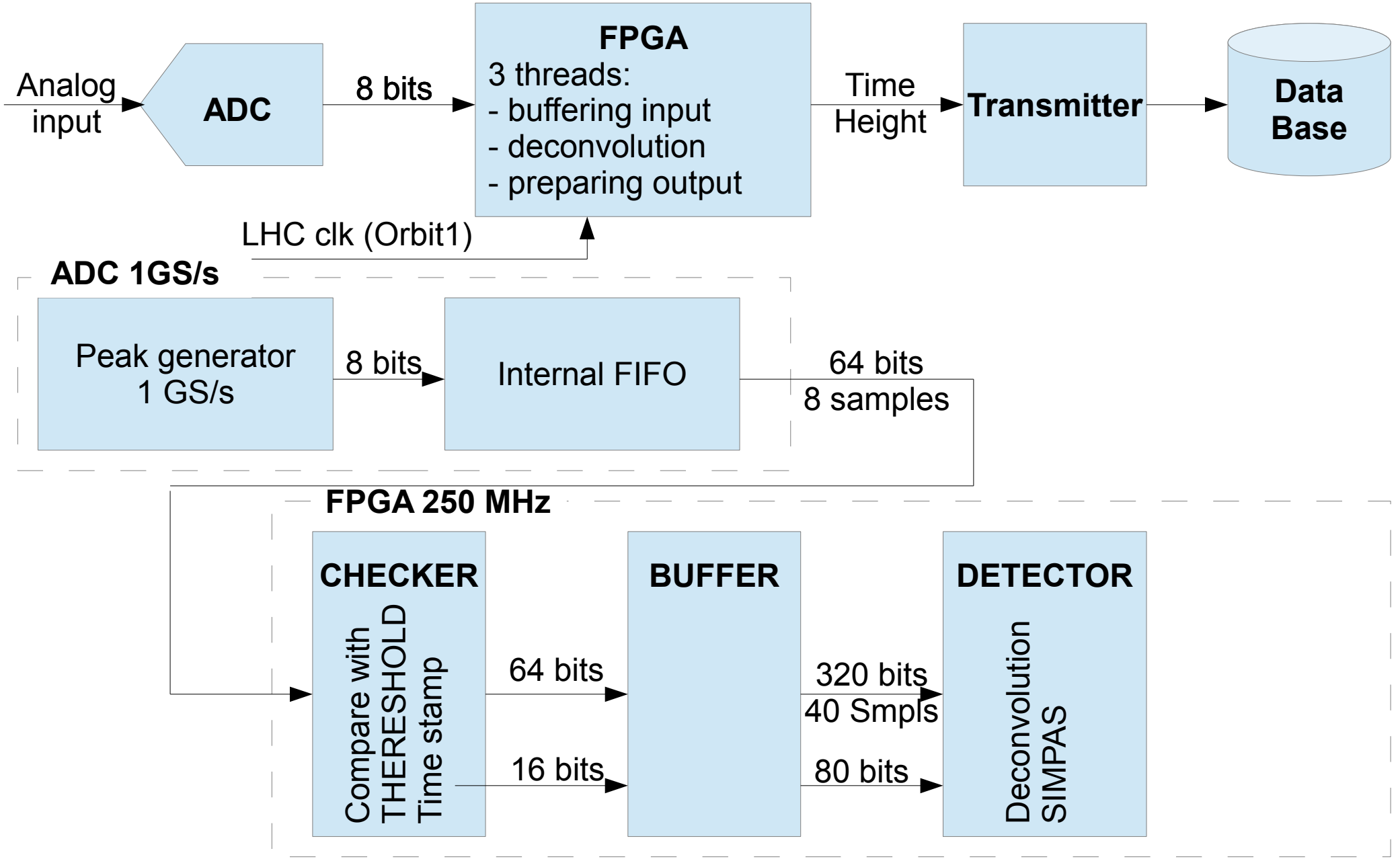
FPGA - data flow



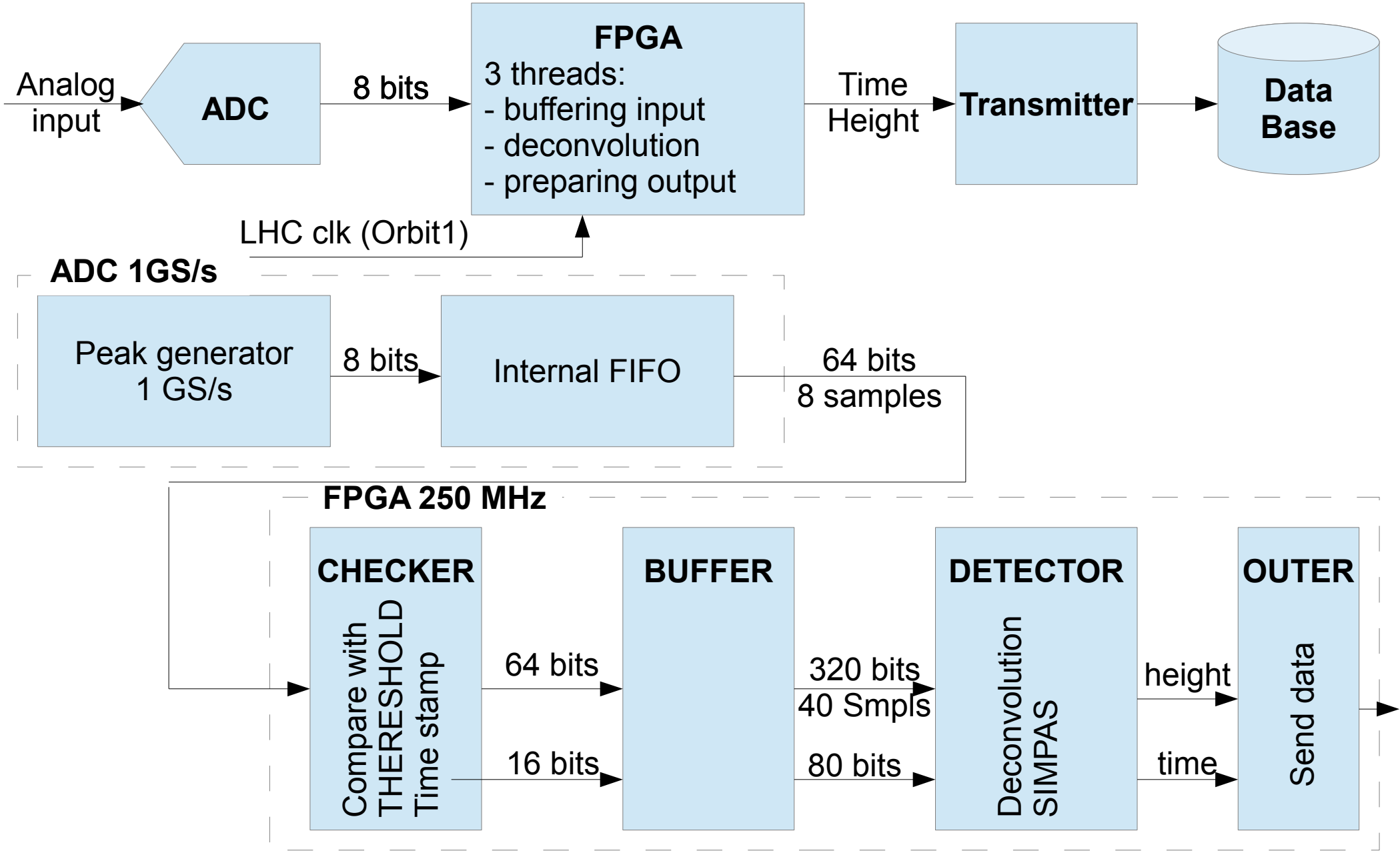
FPGA - data flow



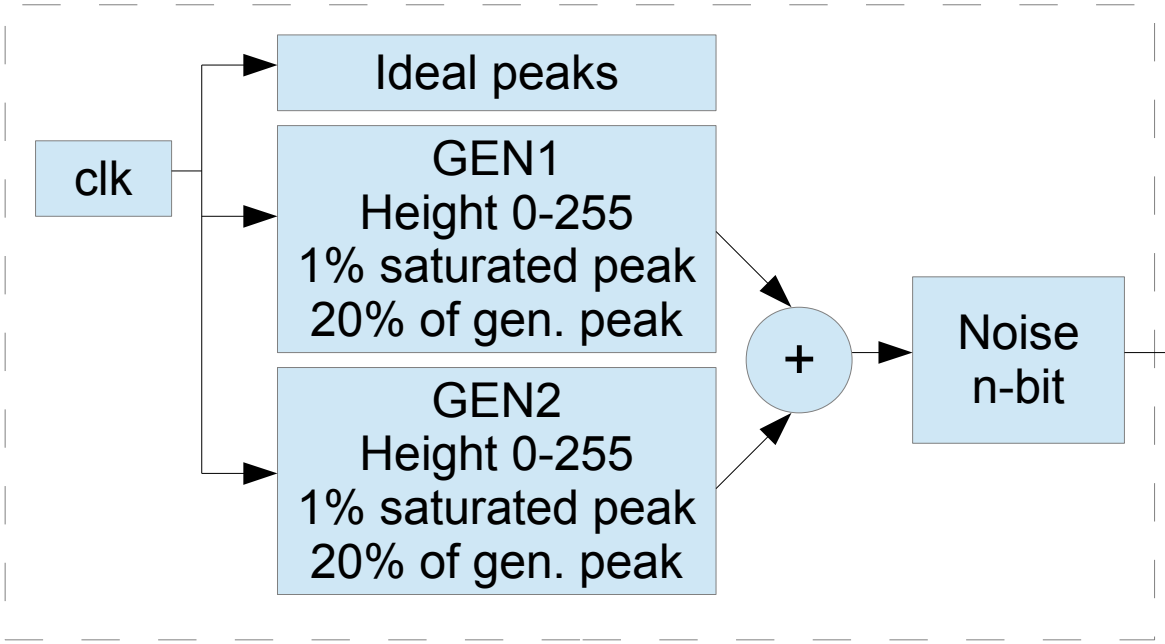
FPGA - data flow

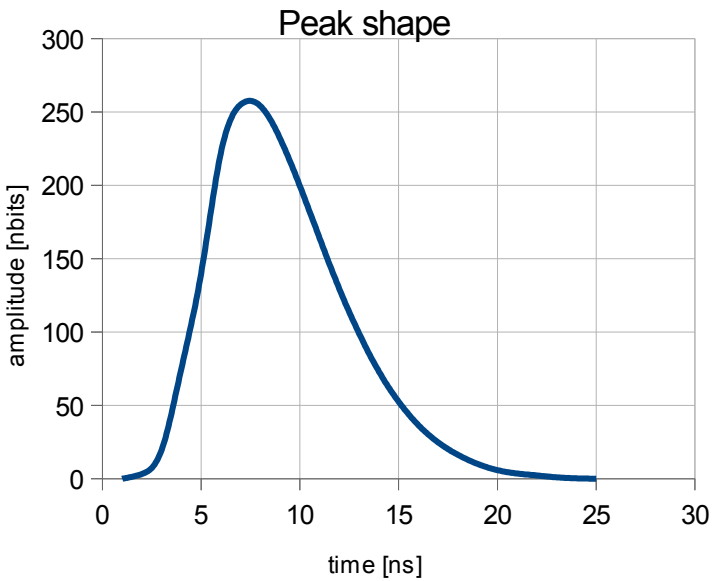


FPGA - data flow

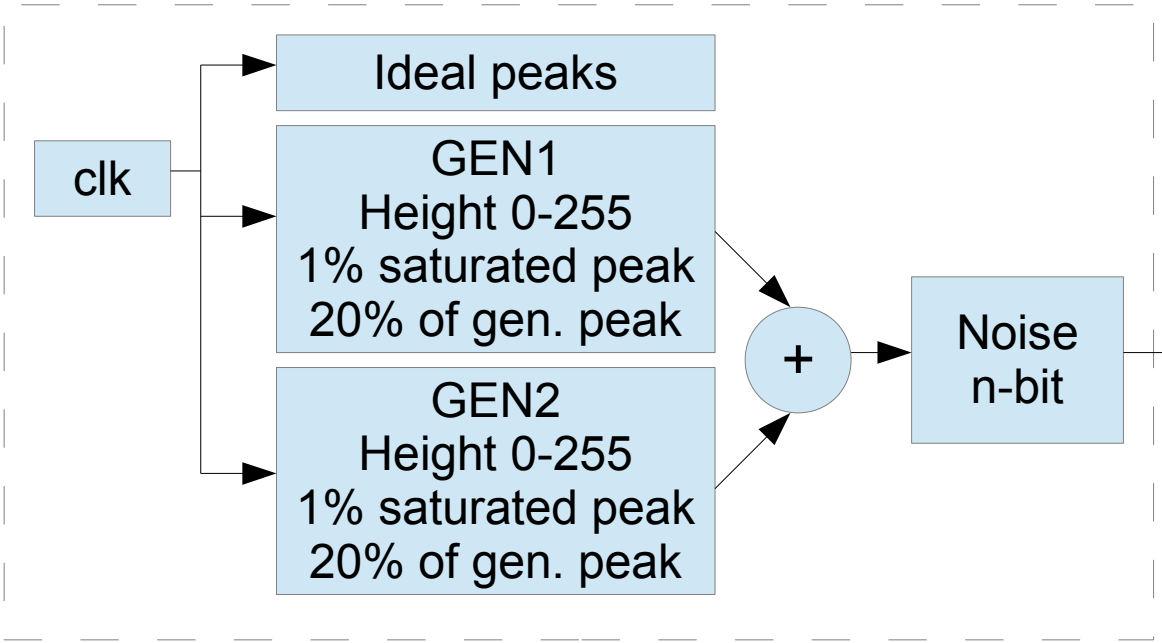


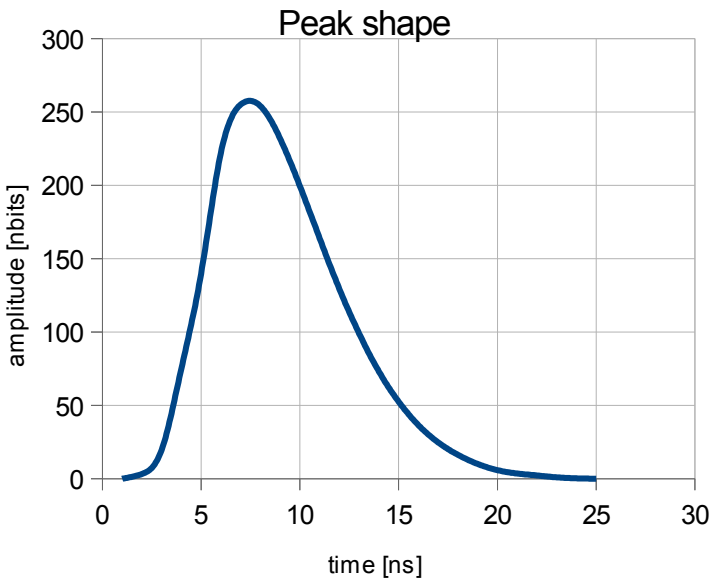
FPGA – peak generator

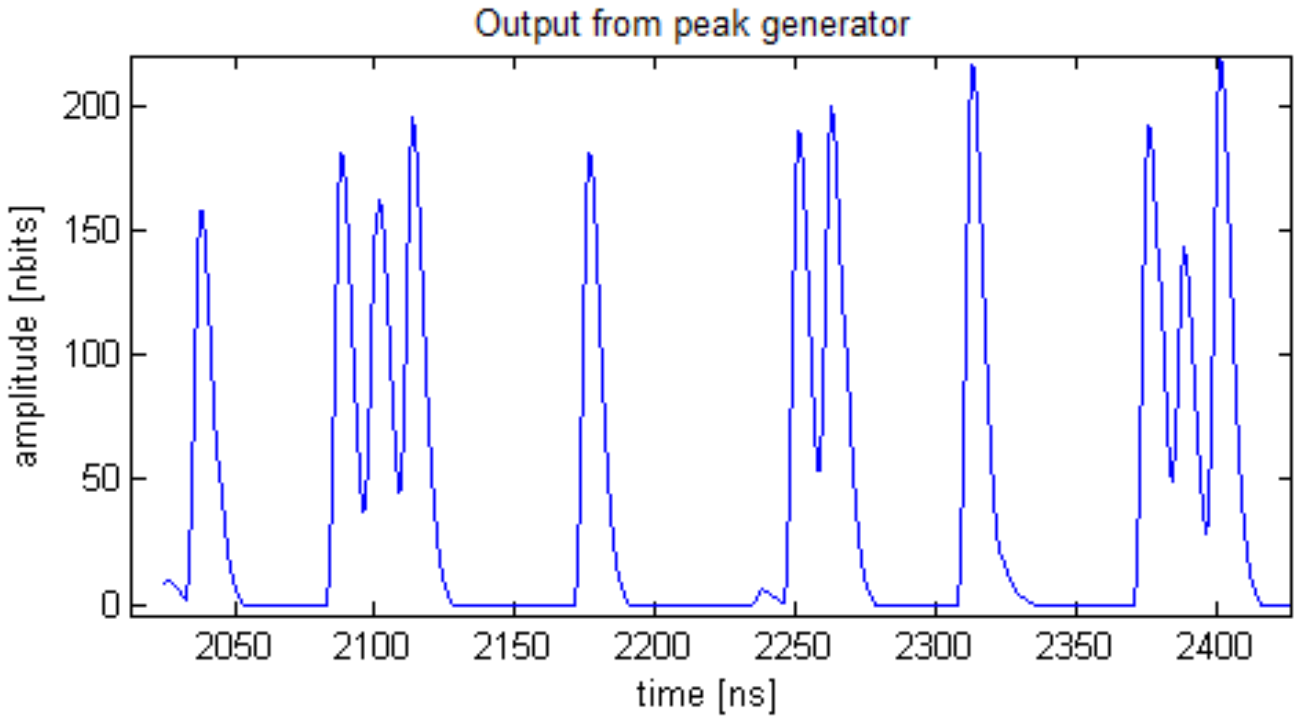




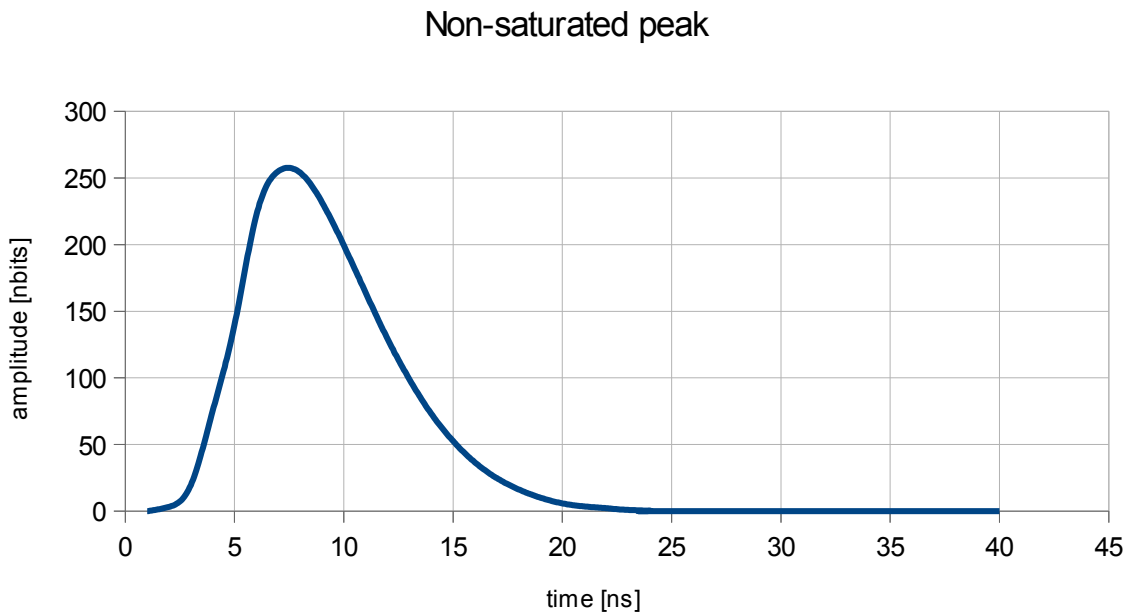
FPGA – peak generator







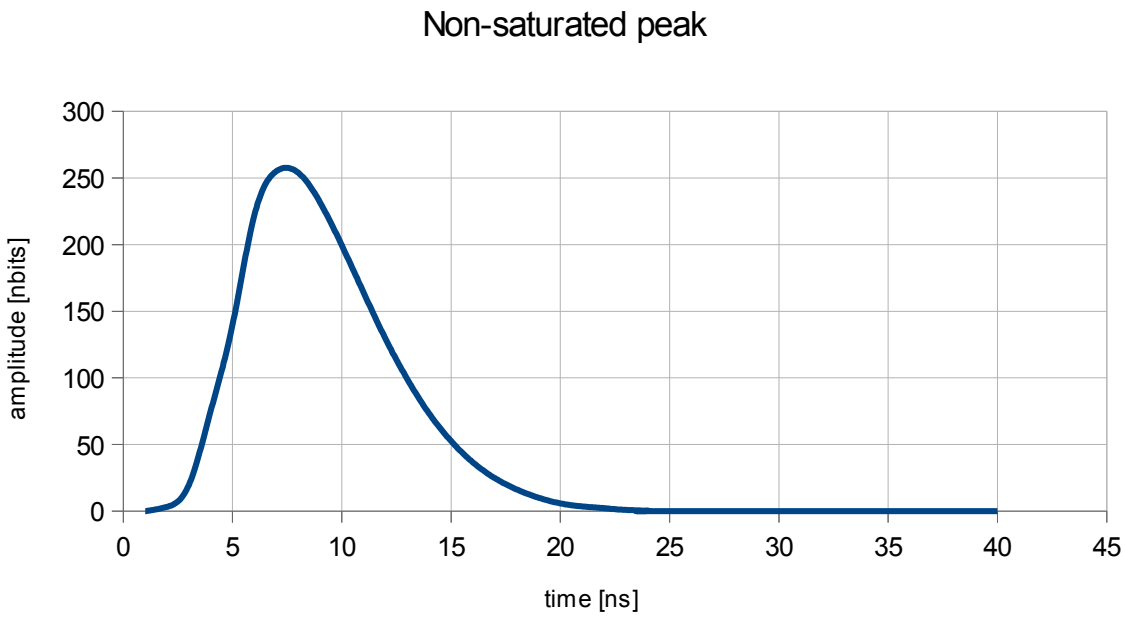
FPGA – peak generator – non saturated peak



Non-Saturated peak:

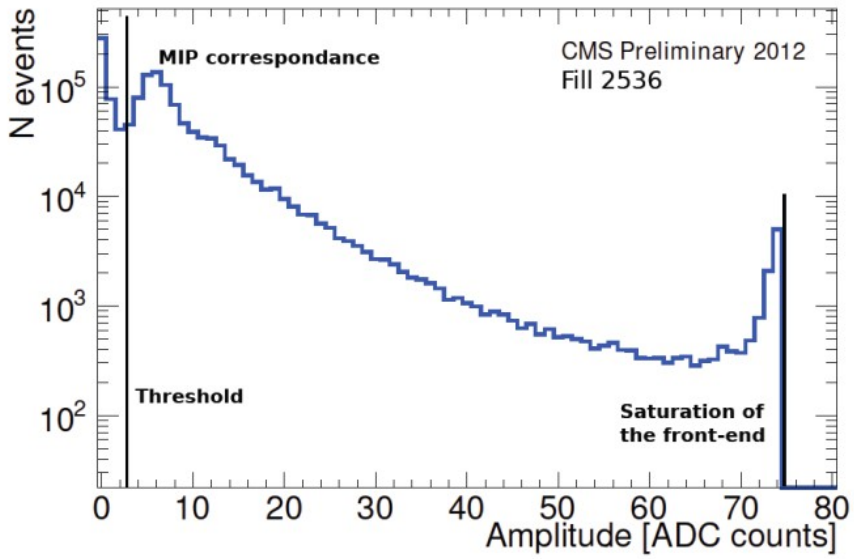
1. Shape similar to Landau distibution
2. FWHM = 10 ns
3. Peaking time (rising edge) = 9ns
4. Peak time = max 25 ns
5. Height in range 0 – 255 (8 bits)
6. Height probability based on peak density distribution

FPGA – peak generator – non saturated peak



Non-Saturated peak:

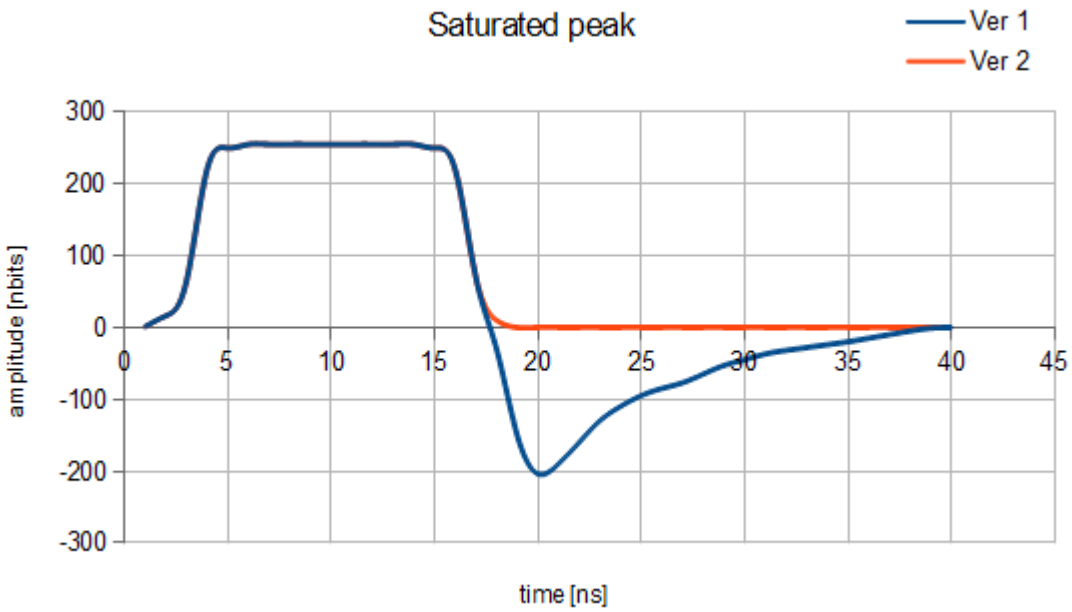
1. Shape similar to Landau distibution
2. FWHM = 10 ns
3. Peaking time (rising edge) = 9ns
4. Peak time = max 25 ns
5. Height in range 0 – 255 (8 bits)
6. Height probability based on peak density distribution



Frontend 2007-2013

height	probability	height	probability
>11 MIP	3.91%	5 MIP	0.55%
10 MIP	0.06%	4 MIP	1.01%
9 MIP	0.07%	3 MIP	3.67%
8 MIP	0.11%	2 MIP	7.34%
7 MIP	0.18%	1 MIP	27.54%
6 MIP	0.37%	noise	55.09%

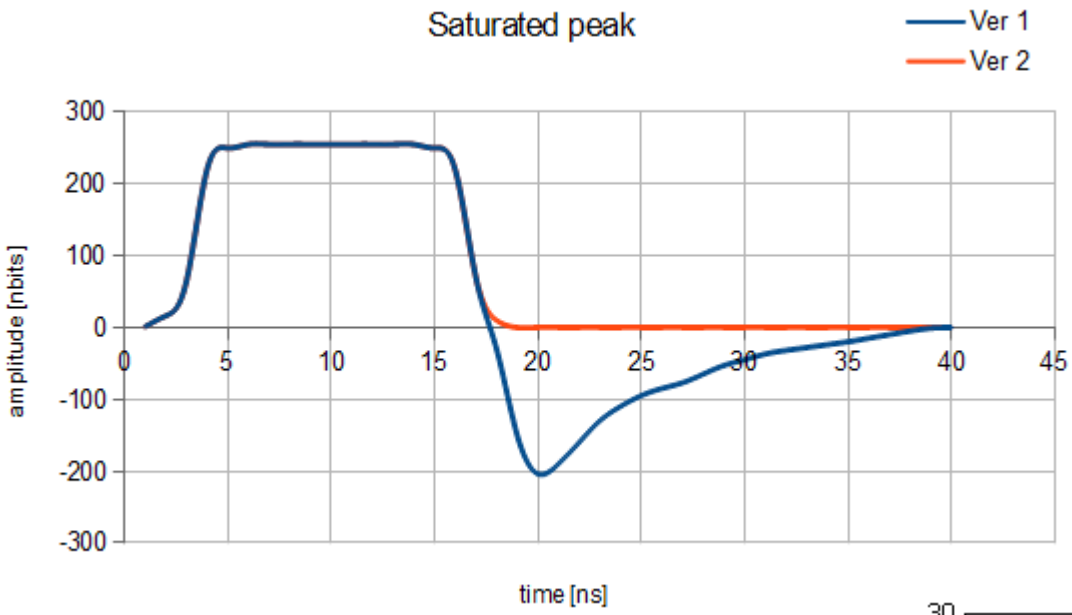
FPGA – peak generator – saturated peak



Saturated peak:

1. causes sensor inactive for 40 ns
2. When peak is saturated do pararell:
   - store data for ~100ns
   - try to calculate on-line

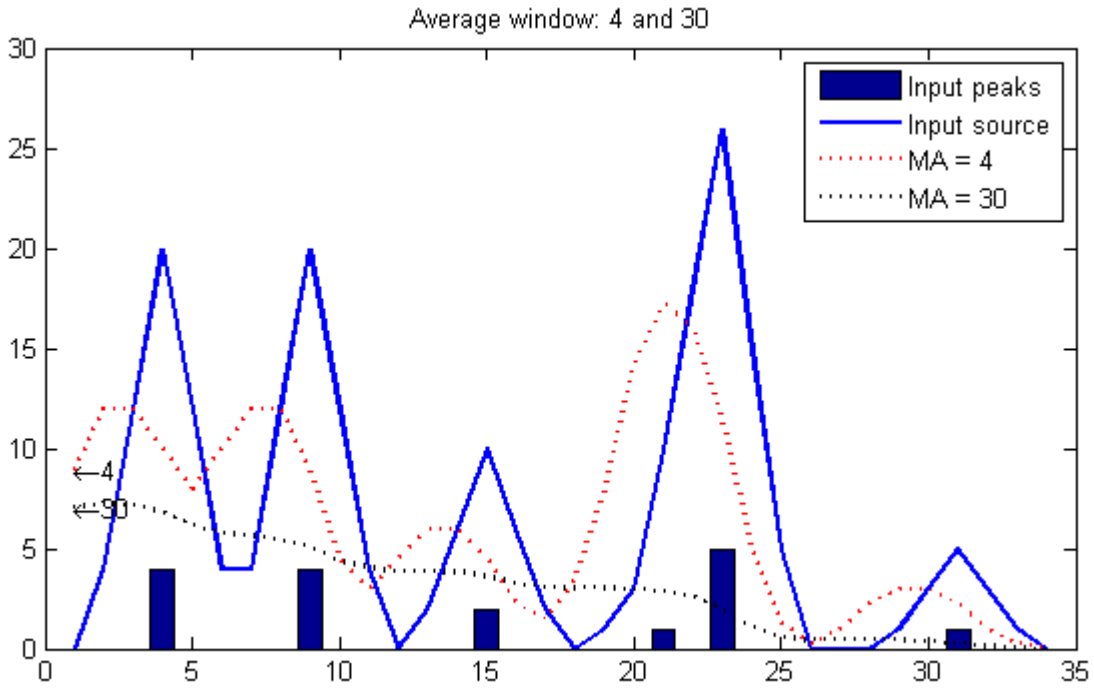
FPGA – peak generator – saturated peak



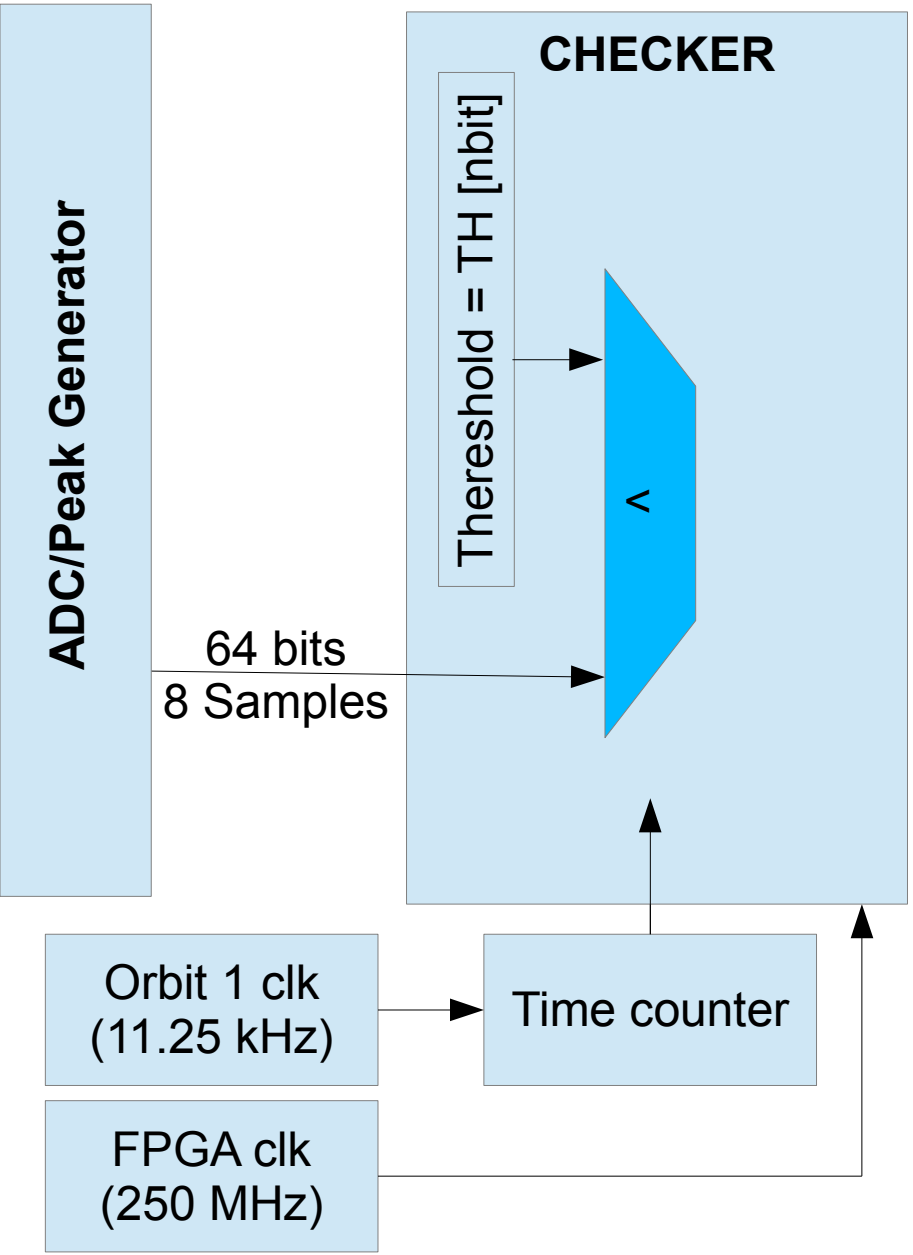
Saturated peak:

1. causes sensor inactive for 40 ns
2. When peak is saturated do pararell:
   - store data for ~100ns
   - try to calculate on-line

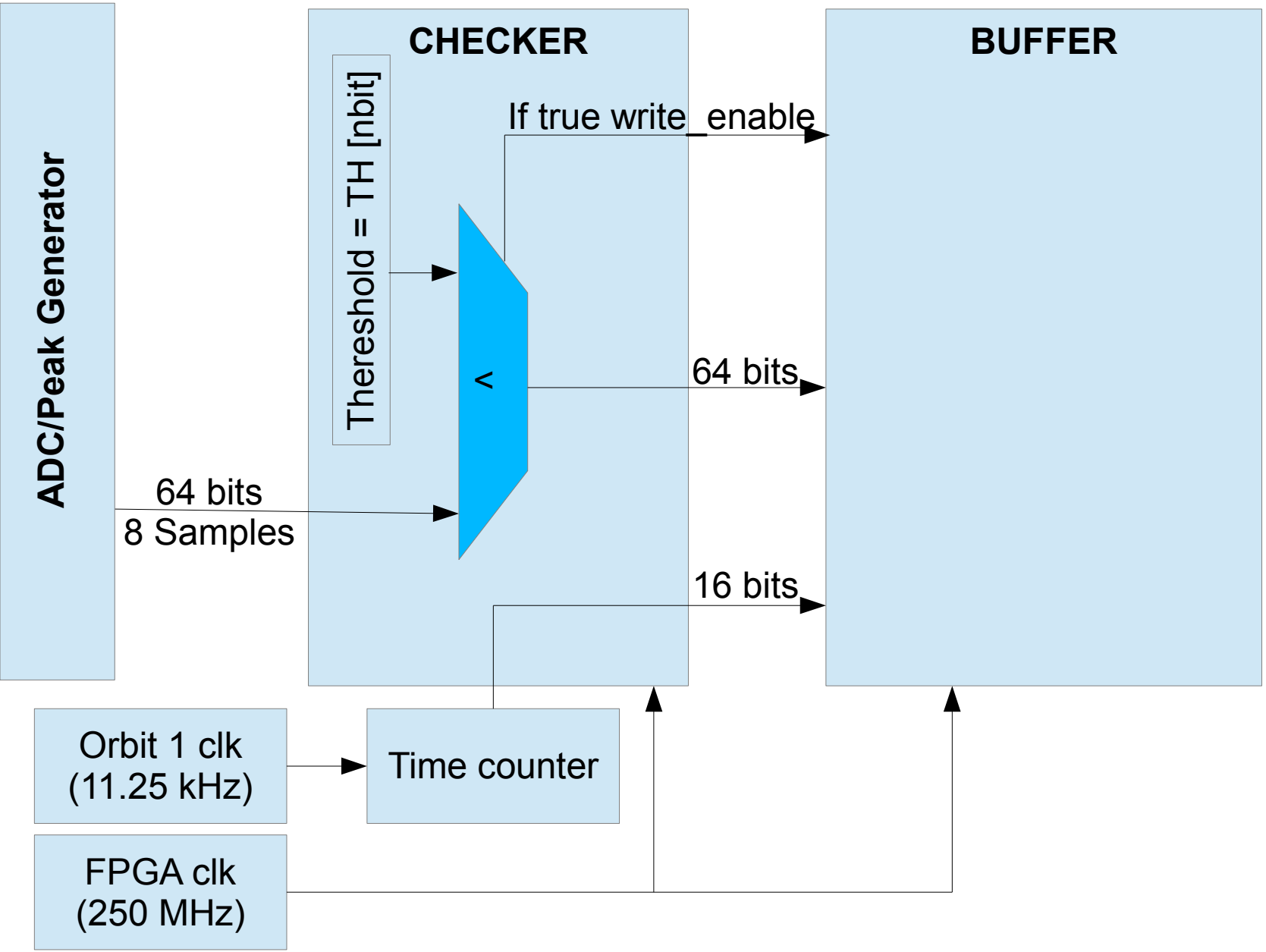
Using moving average (with large averaging window ~30 samples) may be used for follow „dynamic baseline” and do calculations on-line



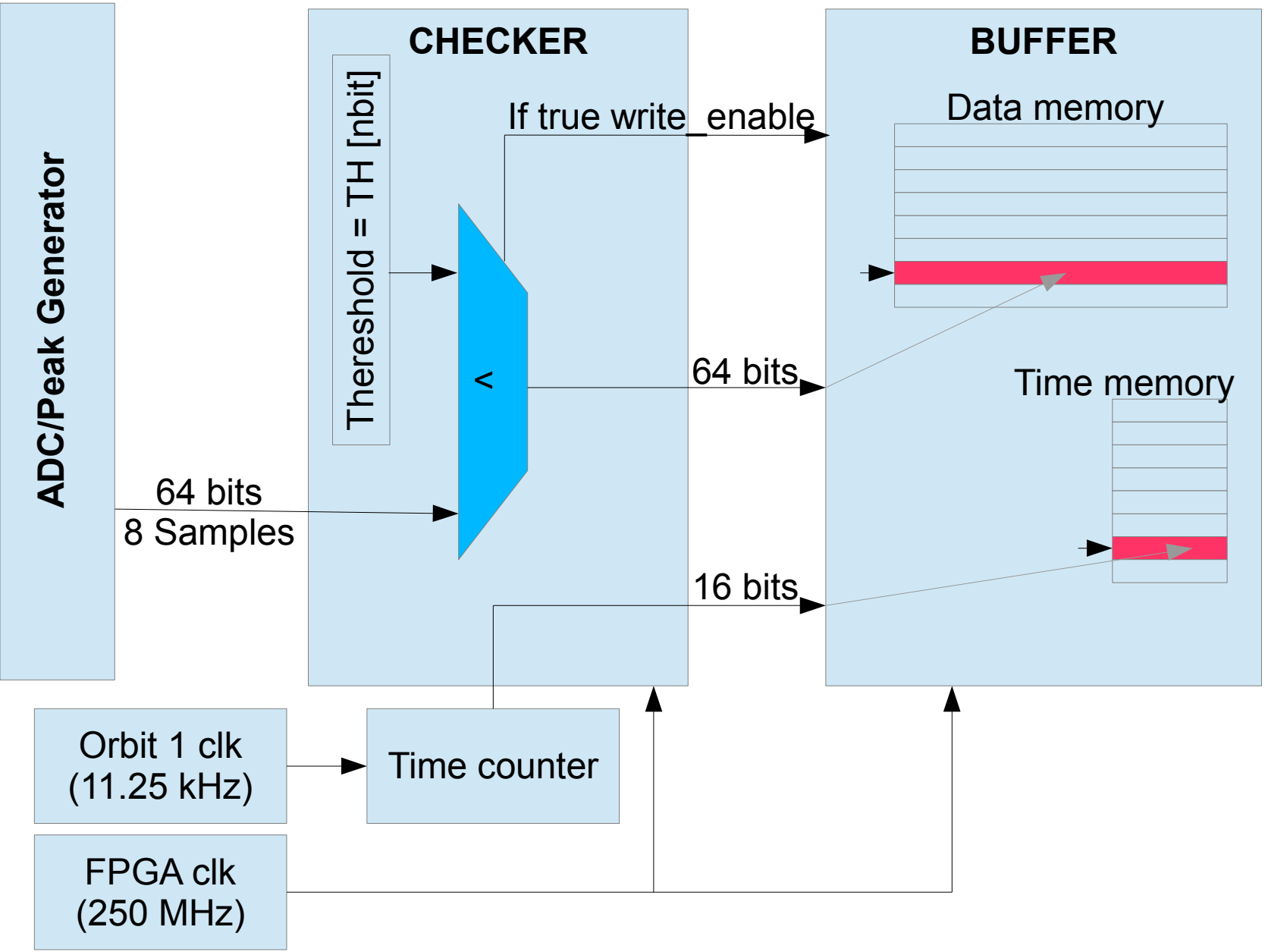
FPGA – other blocks – Checker and Buffer



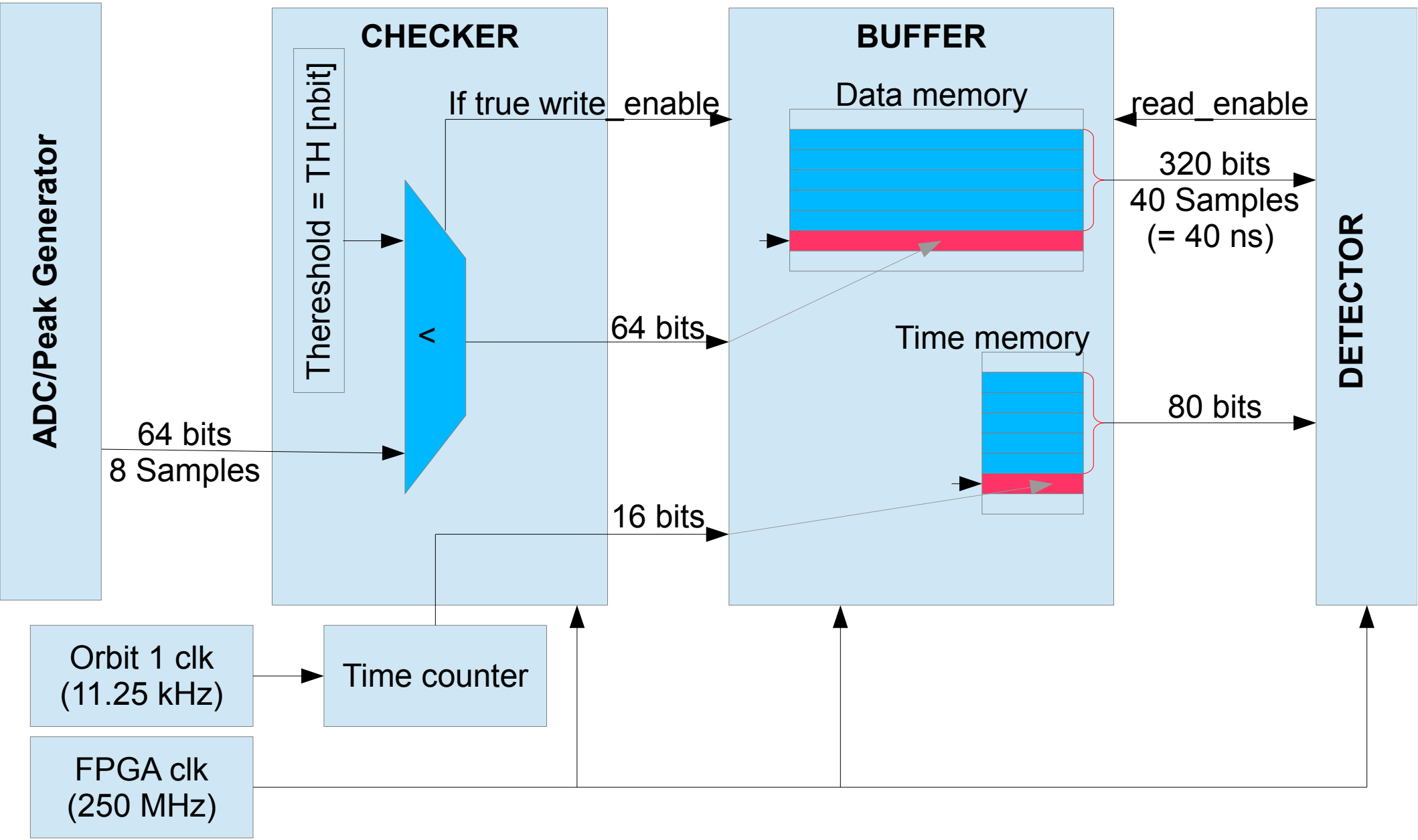
FPGA – other blocks – Checker and Buffer



FPGA – other blocks – Checker and Buffer

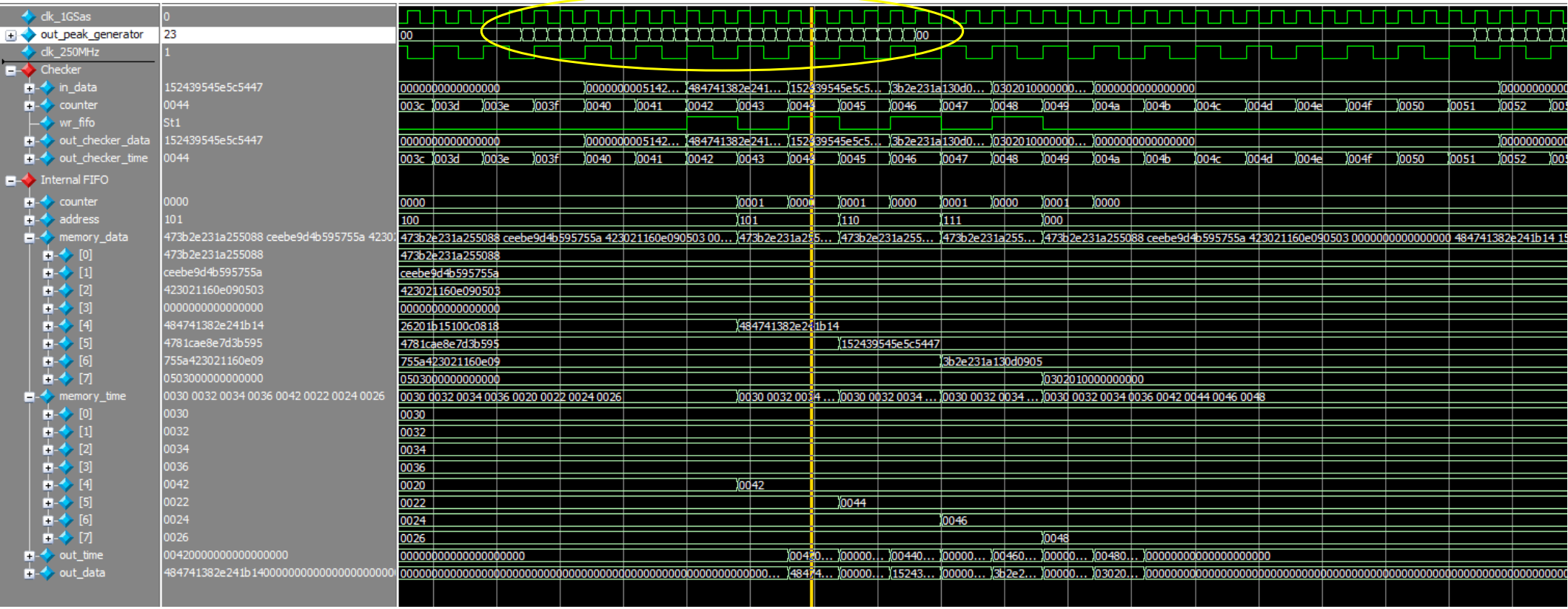


FPGA – other blocks – Checker and Buffer



FPGA – other blocks – Checker and Buffer

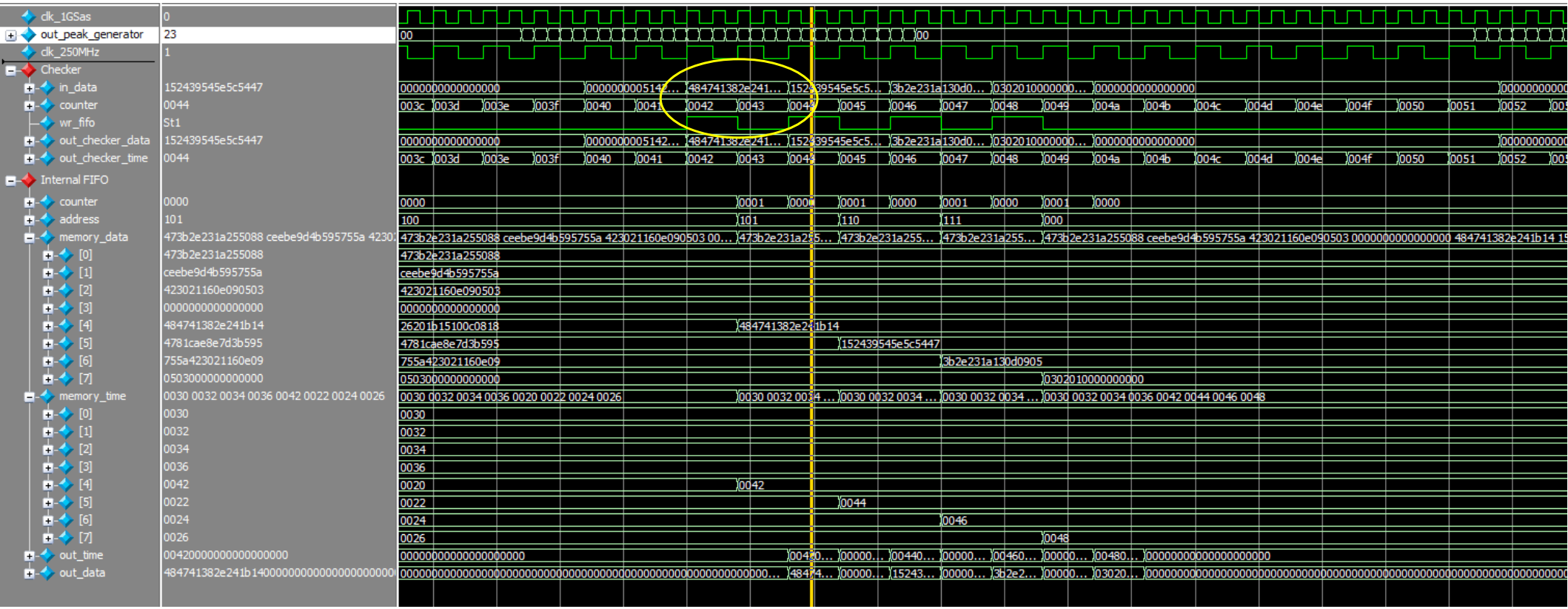
Peak generation



MultiSim

FPGA – other blocks – Checker and Buffer

8 Samples go to Checker...



MultiSim

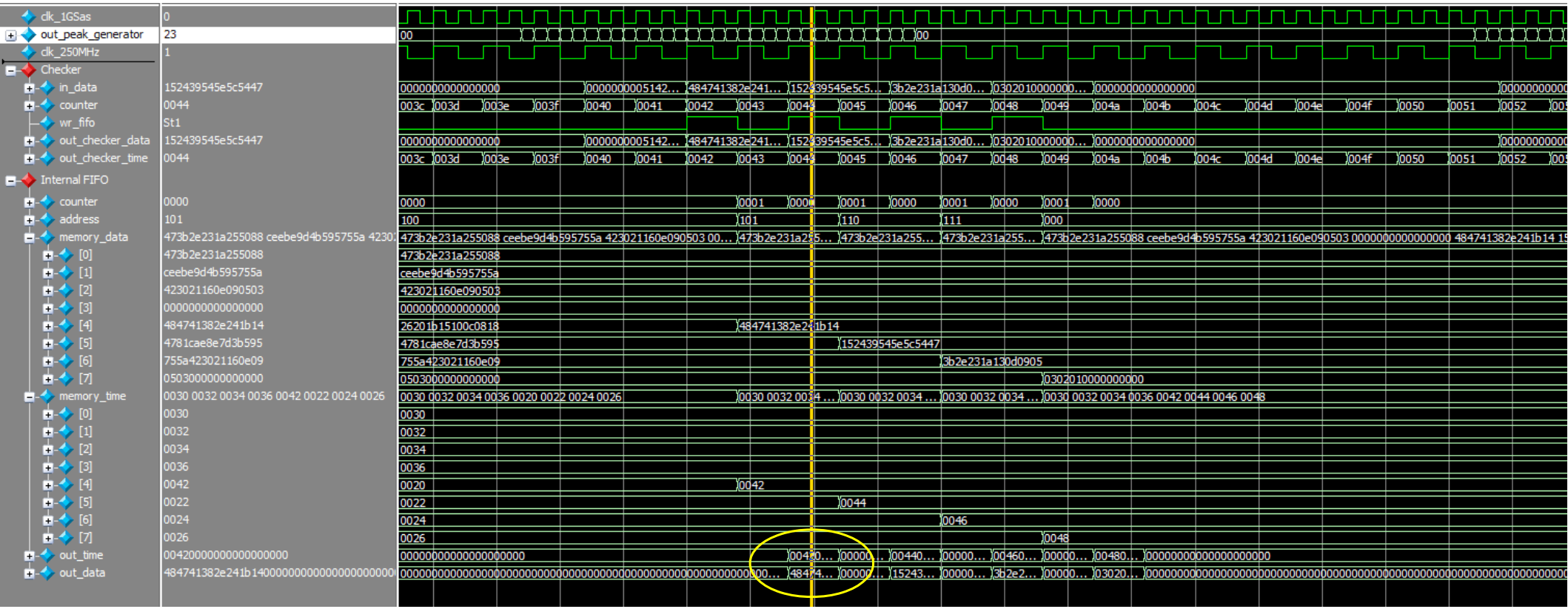
FPGA – other blocks – Checker and Buffer

… and is written to Buffer (data and time)

MultiSim

FPGA – other blocks – Checker and Buffer

Sample of output



MultiSim

FPGA – speed of calculations

Assumptions: FPGA clock = 250 MHz
ADC sampl. = 1 GSa/s
Max peak = 10 MIP Refer peak = 0.5 MIP

FPGA – speed of calculations

Assumptions: FPGA clock = 250 MHz
ADC sampl. = 1 GSa/s
Max peak = 10 MIP Refer peak = 0.5 MIP

The worst case: Buffer get values all the time (peaks on input)

Out Checker: **every** 8 ns write 8 samples to Buffer (2 clocks)

Out Buffer: **every** 40 ns read data by Detector (10 clocks)

FPGA – speed of calculations

Assumptions: FPGA clock = 250 MHz
ADC sampl. = 1 GSa/s
Max peak = 10 MIP Refer peak = 0.5 MIP

The worst case: Buffer get values all the time (peaks on input)
Out Checker: **every** 8 ns write 8 samples to Buffer (2 clocks)
Out Buffer: **every** 40 ns read data by Detector (10 clocks)

Better case: 20% of getting peak to deconvolute
Out Checker: ~40 ns write 8 samples to Buffer
Out Buffer: ~200ns need data to be calculated (50 clocks)

FPGA – speed of calculations

Assumptions: FPGA clock = 250 MHz
ADC sampl. = 1 GSa/s
Max peak = 10 MIP Refer peak = 0.5 MIP

The worst case: Buffer get values all the time (peaks on input)
Out Checker: **every** 8 ns write 8 samples to Buffer (2 clocks)
Out Buffer: **every** 40 ns read data by Detector (10 clocks)

Better case: 20% of getting peak to deconvolute
Out Checker: ~40 ns write 8 samples to Buffer
Out Buffer: ~200ns need data to be calculated (50 clocks)

SIMPAS: Find max, subtract → 2 clk cycles
Number of subtraction (10MIP/0.5MIP =) 20
Operation: 40 clk cycles < 50 clk cycles (should work!*)

FPGA – speed of calculations

Assumptions: FPGA clock = 250 MHz
ADC sampl. = 1 GSa/s
Max peak = 10 MIP Refer peak = 0.5 MIP

The worst case: Buffer get values all the time (peaks on input)
Out Checker: **every** 8 ns write 8 samples to Buffer (2 clocks)
Out Buffer: **every** 40 ns read data by Detector (10 clocks)

Better case: 20% of getting peak to deconvolute
Out Checker: ~40 ns write 8 samples to Buffer
Out Buffer: ~200ns need data to be calculated (50 clocks)

SIMPAS: Find max, subtract → 2 clk cycles
Number of subtraction (10MIP/0.5MIP =) 20
Operation: 40 clk cycles < 50 clk cycles (should work!*)

Deconvolution: At least 50 iterations <= 50 clk cycles (difficult...)

Agenda

1. Motivation
2. Deconvolution
   - Function in Root
   - SIMPAS - Succesive Identify Max Position And Subtract
   - Deconvolution vs. SIMPAS
3. FPGA
   - Data flow – block diagram
   - Peak generator
     - Non-saturated peaks
     - Saturated peaks, moving averages
   - Other blocks
   - Speed of calculations
4. **Summary – what is done**
5. Next steps

Summary – what is done

Done:

1. Checked 2 deconvolution algorithms: Deconvolution and SIMPAS in Root and Matlab
2. Calculated time consumption of these algorithms
3. Wrote Verilog codes:
   - peak generator with upgrade parameters
   - Input FIFO, Checker, Buffer
   - testbenches for that codes

Agenda

1. Motivation
2. Deconvolution
   - Function in Root
   - SIMPAS - Succesive Identify Max Position And Subtract
   - Deconvolution vs. SIMPAS
3. FPGA
   - Data flow – block diagram
   - Peak generator
     - Non-saturated peaks
     - Saturated peaks, moving averages
   - Other blocks
   - Speed of calculations
4. Summary – what is done
5. **Next steps**

Next steps

1. Found algorithm with FFT – need to be checked
2. Study Moving Average algorithm to use it in peaks identifying
3. Change peak generator with peak shape before upgrade
4. Finish “Detector” module in Verilog (with 3 deconvolution options: deconvolution, SIMPAS, with FFT?)
5. Implement pipepline processing in Verilog

6. Estimating memory on FPGA, algorithm time consumption, speed of sampling is necessary to decide on BCM1f choose of backend electronics boards