



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



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



- 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
- 3. Algorithm should be:
 - Reliable
 - Stable

@ PB 2013



64000

65000

66000



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

Source signal amplitude [nbits] i \mathcal{X} time [ns]

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

Deconvolution - Root



Thanks R. Walsh for initial example codes!!



@ PB 2013

Upgrading BCM1f: Deconvolution, FPGA



Upgrading BCM1f: Deconvolution, FPGA

Deconvolution - Root

 $\sum H'_{im} x_m^{(k)}$

m = 0

Algorithm:

0. set x⁰, nRepetitions, nIterations, boost

- 1. repeat nRepetitions times:
 - **1a.** $x_i^{(k)} := [x_i^{(k)}]^{boost}$

1b. Repeat nlterations times: $x^{(k+1)} := x^{(k)} \frac{y}{N-1}$



Deconvolution - Root

Algorithm:

0. set x⁰, nRepetitions, nIterations, boost

- 1. repeat nRepetitions times:
 - **1a.** $x_i^{(k)} := [x_i^{(k)}]^{boost}$

1b. Repeat niterations times: $x^{(k+1)} := x^{(k)} \frac{\mathcal{Y}}{\sum_{m=0}^{N-1} H'_{im} x_m^{(k)}}$

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

	nlterations	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!

- 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))



- 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 = $\begin{bmatrix} 2 & 3 & 1 \end{bmatrix}$ Source = $\begin{bmatrix} 0 & 6 & 9 & 5 & 3 & 1 & 2 & 3 \\ Peaks = \begin{bmatrix} 0 & 0 & 3 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$

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))

Reference = [2	3	1]		So	ource	= [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]	// m	naxX =	= 3 \				

maxX	N times
3	1

- 0. Get Reference and Source signal, thereshold //= expected noise
 1. max(Source) // seeking from right side of Source
 2. Source Source Source
- 2. Source = Source Reference

// Xpos(max(Reference)) = Xpos(max(Source))

Reference = [2	3	1]	S	ource	e = [0	6	9	5	3	1	2	3]	
				Р	eaks	= [0	0	3	0	1	0	0	1]	
1. Source = [0	6	9	5	3	1	2	3]	// r	maxX	= 3				
- Reference	2	3	1												
2. Source = [0	4	6	4	3	1	2	3	1	// r	maxX	= 3			maxx	N times
	$ce = [0 4 \mathbf{b}]$				-		5	1	,,					3	a 2

- 0. Get Reference and Source signal, thereshold //= expected noise
 1. max(Source) // seeking from right side of Source
 2. Source Deference
- 2. Source = Source Reference

// Xpos(max(Reference)) = Xpos(max(Source))

Reference = [2	3	1]	S	ource	e = [0	6	9	5	3	1	2	3]	
					Р	eaks	= [0	0	3	0	1	0	0	1]	
1. Source = [0	6	9	5	3	1	2	3]	// r	maxX	(= 3					
- Reference		2	2 3 1									/	Netime				
2. Source = [0	4	4 6 4 3 1 2 3] // maxX = 3 -								max		in times				
- Reference		2	3	1					-	// maxX = 3					3		2
3. Source = [0	2	3	3	3	1	2	3]	// r	maxX	. = 8 -		_	8		▶ 1

- 0. Get Reference and Source signal, thereshold //= expected noise
 1. max(Source) // seeking from right side of Source
 2. Source Source
- 2. Source = Source Reference

// Xpos(max(Reference)) = Xpos(max(Source))

Reference = [2	2	3	1]	1]		ource	= [0	6	9	5	3	1	2	3]		
					Pe	eaks	= [0	0	3	0	1	0	0	1]		
1. Source = [(C	6	9	5	3	1	2	3]	// r	naxX	= 3						
- Reference		2	3	1											may	\mathbf{v}	N timos	
2. Source = [(C	4	6	4	3	1	2	3]	// maxX = 3					Παλ			
- Reference		2	3	1						// maxx = 3 —					3		2	
3. Source = [(C	2	3	3	3	1	2	3]	// r	naxX			5		— 1		
- Reference							2	3	(1)						8		▶ 1	
4. Source = [(C	2	3	3	3	1	0	0]	// r	naxX	= 5						

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))

Reference = [2	3	1]		Sc	ource	= [0	6	9	5	3	1	2	3]	
					Pe	eaks	= [0	0	3	0	1	0	0	1]	
1. Source = [0	6	9	5	3	1	2	3]	// r	naxX	= 3					
- Reference		2	3	1						// maxX = 3					max	X	N times
2. Source = [0	4	6	4	3	1	2	3]	// r	naxX	= 3 -			тал		
- Reference		2	3	1											3) 3
3. Source = [0	2	3	3	3	1	2	3]	// r	naxX	= 8			5		– 1
- Reference							2	3	(1)						8		▶ 1
4. Source = [0	2	3	3	3	1	0	0]	// r	naxX	= 5					
- Reference				2	3	1											
5. Source = [0	2	3	1	0	0	0	0]	// r	naxX	=3					

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))

Reference = [2	3	1]]		urce	= [0	6	9	5	3	1	2	3]			
					Ре	aks	= [0	0	3	0	1	0	0	1]			
1. Source = [0	6	9	5	3	1	2	3]	// r	naxX	= 3 -							
- Reference		2	3	1											may	~	NI +	imo	
2. Source = [0	4	6	4	3	1	2	3]	// r	naxX	= 3			Παλ			iiiie:	>
- Reference		2	3	1											3			3	
3. Source = [0	2	3	3	3	1	2	3]	// r	naxX	= 8 _			5			1	
- Reference							2	3	(1)						8			1	
4. Source = [0	2	3	3	3	1	0	0]	// r	naxX	= 5 -			Ŭ			•	
- Reference				2	3	1													
5. Source = [0	2	3	1	0	0	0	0]	// r	naxX	=3	~						
- Reference		2	3	1															
6. Source = [0	0	0	0	0	0	0	0]	// f	inish								



convolution, FPGA











Deconvolution vs. SIMPAS





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 – peak generator



FPGA – peak generator



FPGA – peak generator – non saturated peak



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

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



time [ns]

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









43



Peak generation

4	dk_1GSas	0								ปากก			பட	UU										
+ 	out_peak_generator	23	00																					
4	clk_250MHz	1																						
	Checker																							
	🕂	152439545e5c5447	0000000000	000000		00000	0005142	4847413	82e241 (1	52 <mark>39545e5c</mark> 5	<u>3</u> b2e2	31a130d0.	. 03020	1000000))000000	0000000000	0						0000	0000000
	🕂	0044	003c 003d)003e	<u>)003f</u>	0040	0041	0042	0043 0	044 (0045	0046	0047	0048	0049)004a)004b	004c)004d	004e	004f	0050	0051	0052	00
		St1																						
	🕂	152439545e5c5447	0000000000	000000		00000	0005142	4847413	82e241 (1	52 <mark>4</mark> 39545e5c5)3b2e2	31a130d0.	.)03020	1000000))000000	0000000000	0						00000	0000000
	+	0044	003c 003d)003e	<u>)003f</u>	0040	0041	0042)0043)0	044 0045	0046	0047	0048	0049	004a	<u>)004b</u>	004c	004d	004e)004f	0050	0051	0052	00
	Internal FIFO																							
	+	0000	0000)0001)0	000	0000	0001)0000	0001	0000									
		101	100						(101	(110		1111		000										
	🛓 🥎 memory_data	473b2e231a255088 ceebe9d4b595755a 42303	473b2e231a	255088 c	eebe9d4b5	95755a 423	021160e09	0503 00)473b2e231	a2 <mark>5 (473b</mark>)	2e231a255.	(473b2 e	231a255	(473b	2e231a2550	88 ceebe9d	14b59575	5a 42302	1160e09	0503 00000	000000000000000000000000000000000000000	00 48474	1382e24	41b14 1
	🛓 🚽 [0]	473b2e231a255088	473b2e231a	255088						l –														
	🛓 🕂 🙀 [1]	ceebe9d4b595755a	ceebe9d4b5	95755a						ļ														
	🛓 🕂 [2]	423021160e090503	423021160e	090503																				
	🛓 🕂 🔁 [3]	00000000000000	00000000000	000000						ļ														
	4]	484741382e241b14	2620161510	0c0818					484741382	24 <mark>1b14</mark>														
	🛓 🕂 [5]	4781cae8e7d3b595	4781cae8e7	d3b595						(1524)	39545e5c54	47												
	🛓 🕂 [6]	755a423021160e09	755a423021	160e09						ļ		3b2e23	1a130d0	905										
	🛉 📥 - 🧇 [7]	05030000000000	0503000000	000000						ļ				0302	0100000000	00								
		0030 0032 0034 0036 0042 0022 0024 0026	0030 0032 0	034 0036	5 0020 002	2 0024 0026	i		0030 0032	0034)0030	0032 0034	0030 0	032 0034)0030	0032 0034	0036 0042 0	0044 004	5 0048						
	🛓 🕂 [0]	0030	0030							_ <u> </u>														
	🛓 - 🥎 [1]	0032	0032																					
	🛓 🕂 [2]	0034	0034							1														
	i 🛓 - 🔷 [3]	0036	0036																					
	+	0042	0020						0042	1														
	🛓 🕂 🔁 [5]	0022	0022							0044														
	🛓 🔶 [6]	0024	0024							1		0046												
	🛓 📥 - 🧇 [7]	0026	0026											0048										
	🛓 🧇 out_time	0042000000000000000	0000000000	000000000	000)0	04 <mark>.0 (0000</mark>))00440.	00000.)00460)0000	0)00480.)0000000	0000000	000000						
	🛨 - 🔷 out_data	484741382e241b140000000000000000000000	0000000000	000000000	000000000	0000000000	0000000000	0000000	000000)4	84 <mark>7</mark> 4 (0000)) (15243.	00000.)3b2e2)0000	0)03020.)0000000	0000000	00000000	0000000	0000000000	000000000	0000000	0000000	0000000

8 Samples go to Checker...

🔶 dk_1GSas	0																							
🕀 🔶 out_peak_generator	23	00																						
dk_250MHz	1																							
						— –																		
🕂 🕂 in_data	152439545e5c5447	000000	000000000	00)000	0000005142	484741	1382e241	152	39545e5c5	. 3b2e231	a130d0	030201	1000000	0)000000	000000000000000000000000000000000000000	0						20000	0000000
🛓 🧄 counter	0044	003c 🕽	003d)(003e)00)3f)004) (0041	0042	0043	004	0045	0046	0047	0048	0049)004a)004b	004c)004d)004e	004f	0050	0051	0052	00
→ wr_fifo	St1																							
🛓 🧄 out_checker_data	152439545e5c5447	000000	000000000	οφ	000	0000005142	484741	1382e241	(1524	39545e5c5	.)3b2e231	a130d0	030201	1000000	0)000000	000000000000000000000000000000000000000	0						0000	0000000
🛓 🧇 out_checker_time	0044	003c	003d)(00 <u>3e)</u> 00)3f)004	0041	0042	0043	0044	0045	0046	0047	0048	0049)004a	<u>)004b</u>	004c)004d)004e	004f	0050	0051	0052	00
🖃 🔶 Internal FIFO																								
🛓 - 🔶 counter	0000	0000						0001	0000	0001	0000	0001	0000	0001	0000									
🛓 🔶 address	101	100						(101	j	(<u>110</u>		1111		000										
🛓 🧇 memory_data	473b2e231a255088 ceebe9d4b595755a 4230	473b2e	231a2550	88 ceebe9d	4b595755a ·	423021160e	090503 00.	(473b2	e231a2	5 (473b2e	231a255	473b2e2	31a255.	(473b	2e231a2550	88 ceebe9d	4 b 59575	5a 42302	21160e09	0503 00000	000000000	00 48474	1382e24	41b141
🛓 📥 - 🤣 [0]	473b2e231a255088	473b2e	231a2550	88					j															
🔄 🛓 - 🥎 [1]	ceebe9d4b595755a	ceebe9	d4b59575	5a					j															
🛓 📥 - 🤣 [2]	423021160e090503	423021	160e0905	03					j															
🛓 🛓 🔶 [3]	00000000000000	000000	000000000	00					j															
🔄 🛓 🔶 [4]	484741382e241b14	26201b	15100c08	18				(48474	1382e24	1b14														
🔄 🛓 🔶 [5]	4781cae8e7d3b595	4781ca	e8e7d3b5	95					j	152439	545e5c544	7												
🛓 🕂 🔶 [6]	755a423021160e09	755a42	23021160e	09					j			3b2e231	a130d09	05										
🛛 📥 - 🥎 [7]	05030000000000	050300	000000000	00					j					0302	0100000000	00								
🖕 🔶 memory_time	0030 0032 0034 0036 0042 0022 0024 0026	0030 0	032 0034 0	0036 0020 0	0022 0024 00)26)0030 (0032 003	4)0030 00	032 0034	0030 003	32 0034 .)0030	0032 0034	0036 0042 (0044 0046	5 0048						
🗼 - 🥎 [0]	0030	0030							į															
📋 🖕 🎝 [1]	0032	0032																						
🛓 🕂 🔁 [2]	0034	0034							į															
🛓 🕂 🔶 [3]	0036	0036																						
🕴 🕂 🕂 🕂 🕂	0042	0020						0042																
🛓 🕂 🤣 [5]	0022	0022								0044														
🛓 🕂 🔶 [6]	0024	0024										0046												
🔄 🛨 🔶 [7]	0026	0026												0048										
🛓 - 🔷 out_time	0042000000000000000	000000	000000000	00000					0042	0 (00000.	.)00440	00000	00460.	(0000	0)00480.)0000000	0000000	000000						
🛓 - 🧇 out_data	484741382e241b140000000000000000000000	000000	000000000	000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	0000000000	00000000)484	4 (00000.	. (15243	00000	3b2e2.	0000	0 03020.		0000000	0000000	0000000	000000000000000000000000000000000000000	000000000	00000000	000000	0000000

... and is written to Buffer (data and time)

A																							_
dk_1GS	Sas	0		\Box	ЦЦ						ЦUL					ו⊔ו				$\sqcup \sqcup \sqcup$	4 LJ L		
🖃 🔶 out_pe	ak_generator	23	00																				
🔶 dk_250	OMHz	1																					
🖃 🔶 Checke	er																						
🔄 🛓 🥠 in_o	data	152439545e5c5447	00000000000	000000		000000	0005142	484741382	e241 (15	2439545e5c5		130d0)0	3020100000	00 00000	000000000000000000000000000000000000000	D						000000	00000
🛓 🤙 cou	unter	0044	003c 003d)003e)003f	0040	0041	0042 0	043 00	14 (0045	0046	0047 0	048 004	19 004a)004b	004c)004d)004e	004f	0050	0051	0052	00
—🔶 wr_	_fifo	St1																					
🛓 🤙 out	t_checker_data	152439545e5c5447	00000000000	000000		000000	0005142	484741382	e241 (15	2439545e5c5	. 3b2e231	130d0)0	3020100000	00 000000	000000000000000000000000000000000000000	D						1000000	000000
🛓 🤙 out	t_checker_time	0044	003c 003d)003e)003f	0040	0041	0042 0	043 00	1 (0045	0046	0047 0	048 004	19 (004a)004b	004c)004d)004e	004f	0050	0051	0052	00
💶 🔶 Interna																							
		0000	0000						Van Van	Vacat	Vacana	0001	Van Van	10000							<u> </u>		
	droce	101	100					/L		10001 V110	10000	<u>,0001 ,0</u>											
	meru data	101 472h2a221a2EE000 caaba0d4bE0E7EEa 4220'	100	255000	-b-odab 5	05755- 400	001100-000		1725-2-221-1		- 221-255	4725-221	-255 /47) 	000		- 40000	1100-00	0502 00000	0000000000	0 40 4741	00-041	
	rol	473b2e231a255000 Ceebe904b595755a 4250.	47302e231a	255088 ce	ebegaab	957558 423	0211606090	1503 00 <u>j</u> e	1/302e231a	(15 <u>1473D</u> 2	ez31a255	47302e231	a255 <u>1</u> 47.	02e231a255	USS CEEDE90	4059575	5a 42302	1160609	0503 00000	000000000000000000000000000000000000000	0 484/41	82e241	0141:
_ ±- <u>×</u>	[0]	47002E201d20000	4/302e231a	255088																			
1 ± 🭸	[1]	ceebe9d4b595755a	ceebe9d4b5	95755a																			
+ -?∕	[2]	423021160e090503	423021160e	090503																			
- I 🕂 🔶	[3]	00000000000000	00000000000	000000																			
📃 😐 🔶	[4]	484741382e241b14	2620161510	0c0818					84741382e	241b14													
📃 🛓 🔶	[5]	4781cae8e7d3b595	4781cae8e7	d3b595						15243	9545e5c5447	/											
🛛 🗎 🛓 🔶	[6]	755a423021160e09	755a423021	.160e09								3b2e231a1	30d0905										
🛛 🗎 📥 🔶	[7]	05030000000000	0503000000	000000									03	2010000000	000								
📥 🔶 mer	mory_time	0030 0032 0034 0036 0042 0022 0024 0026	0030 0032 0	034 0036 0	0020 0022	0024 0026)(030 0032 00) <mark>4 (0030 (</mark>	032 0034	0030 0032	0034)00:	30 0032 0034	0036 0042 0	044 0046	0048						
🛛 🗎 🛓 🔶	[0]	0030	0030																				
🛛 🕴 📥	[1]	0032	0032																				
🛛 🕴 📥	[2]	0034	0034																				
🔹 🖬 🍝	[3]	0036	0036																				
🛛 🛛 🐺 🛶	[4]	0042	0020					<u>ار</u>	042														
- I II-	[5]	0022	0022							10044													
1 1 🗸	[6]	0024	0024									0046											
1 1 🗸	[7]	0026	0025										Yoo	19									
	time	0042000000000000000	00000000000	000000000000000000000000000000000000000	00				lao		100440	100000 Yo	00	00 100490	10000000	0000000	00000						
- out	t data	484741382=2415140000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000	0000 149	100000	115242	00000			10000000			000000	0000000000	0000000000	0000000	000000	00000
Out			00000000000	0000000000		000000000000000000000000000000000000000		000000000000000000000000000000000000000	0000	1, 1 100000	10210	100000	0202	00 103020		0000000	0000000	0000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	00000000	000000	100000

Sample of output

💠 dk_1GSas	0									ſſ									цц				ک ک	
+ + out_peak_generator	23	00			1000				xiaax))00											± 0		
🔷 dk_250MHz	1									ιLΓ														
🖃 🔶 Checker																								
🛓 🥠 in_data	152439545e5c5447	000000	000000000000000000000000000000000000000)¢	00000	00005142	. 484741	382e241)	152 <mark>4</mark> 39545e5	c5)3	3b2e231a	130d0	030201	000000)000000	00000000000	00						00000	000000
🛓 - 🔶 counter	0044	003c	003d)0	03e (003	f <u>)</u> 0040	0041	0042)00 <u>43</u>))04 <mark>4)</mark> 004	5)0	0046	0047	0048	0049	004a	004b	004c	004d)004e	004f	0050	0051	0052	00
🛶 wr_fifo	St1																							
🛓 🥠 out_checker_data	152439545e5c5447	000000	000000000000000000000000000000000000000)¢	00000	00005142	. 484741	382e241)	152 <mark>4</mark> 39545e5	c5)3	3b2e231a	130d0	030201	000000)000000	000000000	00						00000	000000
🛓 🥠 out_checker_time	0044	003c	003d)0	03e (003	f <u>)</u> 0040	0041	0042)00 <u>43</u>)	004 004	5)0	0046	0047	0048	0049	004a)004b	004c	<u>004d</u>)004e	004f	0050	0051	0052	00
= Internal FIFO																								
🛓 - 🔶 counter	0000	0000						<u>)0001</u>)	000 000	1)0	0000	0001	0000	0001	0000									
🛓 - 🔶 address	101	100						(101	(110			111		000										
🛓 🤣 memory_data	473b2e231a255088 ceebe9d4b595755a 4230	473b2e	231a25508	38 ceebe9d4	595755a 42	3021160e09	90503 00	. (473b2e23)	la2.5 (473	b2e231	1a255	473b2e2	231a255	. (473b2	e231a2550	88 ceebe90	d4b59575	5a 42302	1160e090	0503 00000	00000000	0 48474	1382e24	1b14 1
[0] <u>↓</u> -◆	473b2e231a255088	473b2e	231a25508	38					j														هه	
🛓 - 🔶 [1]	ceebe9d4b595755a	ceebe9	d4b595755	ia																			ه ک	
🛓 📥 - 🥎 [2]	423021160e090503	423021	1160e09050)3																				
🛓 📥 🔶 [3]	00000000000000	000000	000000000000000000000000000000000000000)¢					i														 _	
🛛 🛓 🔶 [4]	484741382e241b14	26201	15100c081	8				48474138	e241b14														83 3	
🛓 🛓 🔶 [5]	4781cae8e7d3b595	4781ca	e8e7d3b59	5					152	439545	5e5c5447												هه	
🛓 📥 - 🤣 [6]	755a423021160e09	755a42	23021160e0	9					j			3b2e231	la 130d09	05									هه	
📥 - 🥎 [7]	05030000000000	050300	000000000000000000000000000000000000000)¢					j					03020	100000000	00							هه	
🛓 🥠 memory_time	0030 0032 0034 0036 0042 0022 0024 0026	0030 0	032 0034 0	036 0020 00	22 0024 002	5		0030 0032	0014)003	0 0032	0034)	0030 00	32 0034 .		0032 0034 (036 0042	0044004	5 0048					هه	
[0] <u>↓</u> -◆	0030	0030																						
📋 🖕 🄶 [1]	0032	0032																					د ک	
🛓 📥 - 🔶 [2]	0034	0034																					د ک	
🛓 📥	0036	0036																					د ک	
4	0042	0020						0042															= =	
🛓 📥 - 🥎 [5]	0022	0022							004	4													 _	
🛓 📥 - 🔶 [6]	0024	0024							i i i			0046											 _	
i i i i i i i i i i i i i i i i i i i	0026	0026						-						0048									د ک	
🛓 - 🧇 out_time	0042000000000000000	000000	000000000000000000000000000000000000000	00000					00420 000	00)0	00440	00000	. 00460	. 100000	00480.		0000000	000000					د ک	
🛓 - 🔶 out_data	484741382e241b140000000000000000000000	000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000	0000 <mark>000)</mark>	184 <mark>7</mark> 4 (000	00 <mark>)</mark> 1	15243	00000	.)3b2e2	.)00000	03020.		0000000	000000000	0000000	000000000	000000000	0000000	0000000	000000

FPGA – speed of calculations

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

Assumptions:FPGA clock = 250 MHz
ADC sampl. = 1 GSa/s
Max peak = 10 MIPThe 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)

<u>Assumptions</u> :	FPGA clock	= 250 MHz		
	ADC sampl.	= 1 GSa/s		
	Max peak	= 10 MIP	Refer peak	= 0.5 MIP
The worst case:	Buffer get va	lues all the tir	ne (peaks on	input)
Out Checker:	every 8 ns w	rite 8 samples	to Buffer (2 d	clocks)
Out Buffer:	every 40 ns r	ead data by D	etector (10 c	locks)
Better case:	20% of gettir	ng peak to dec	onvolute	
Out Checker:	~40 ns write	8 samples to	Buffer	
Out Buffer:	~200ns need	data to be ca	lculated (50 c	locks)

Assumptions:	FPGA clock	= 250 MHz		
	ADC sampl.	= 1 GSa/s		
	Max peak	= 10 MIP	Refer peak	= 0.5 MIP
The worst case:	Buffer get va	lues all the tin	ne (peaks on	input)
Out Checker:	every 8 ns w	rite 8 samples	to Buffer (2 o	clocks)
Out Buffer:	every 40 ns r	read data by D	etector (10 c	locks)
Better case:	20% of gettir	ng peak to dec	onvolute	
Out Checker:	~40 ns write	8 samples to	Buffer	
Out Buffer:	~200ns need	data to be ca	Iculated (50 c	locks)

SIMPAS:Find max, subtract \rightarrow 2 clk cyclesNumber of subtraction (10MIP/0.5MIP =) 20Operation: 40 clk cycles < 50 clk cycles (should work!*)</td>

Assumptions:	FPGA clock	= 250 MHz		
	ADC sampl.	= 1 GSa/s		
	Max peak	= 10 MIP	Refer peak	= 0.5 MIP
The worst case:	Buffer get va	lues all the tir	ne (peaks on	input)
Out Checker:	every 8 ns w	rite 8 samples	s to Buffer (2 o	clocks)
Out Buffer:	every 40 ns r	ead data by D	etector (10 c	locks)
Better case:	20% of gettir	ng peak to dec	convolute	
Out Checker:	~40 ns write	8 samples to	Buffer	
Out Buffer:	~200ns need	data to be ca	Iculated (50 c	clocks)
<u>SIMPAS</u> :	Find max, sul	btract \rightarrow)MIP/0.5MIP :	2 clk cycles

Operation: 40 clk cycles < 50 clk cycles (should work!*)

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

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

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

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