I2C Master Core with Interface to GTB Bus

The task: To design an *I2C master* core with Interface to *GTB Bus*.

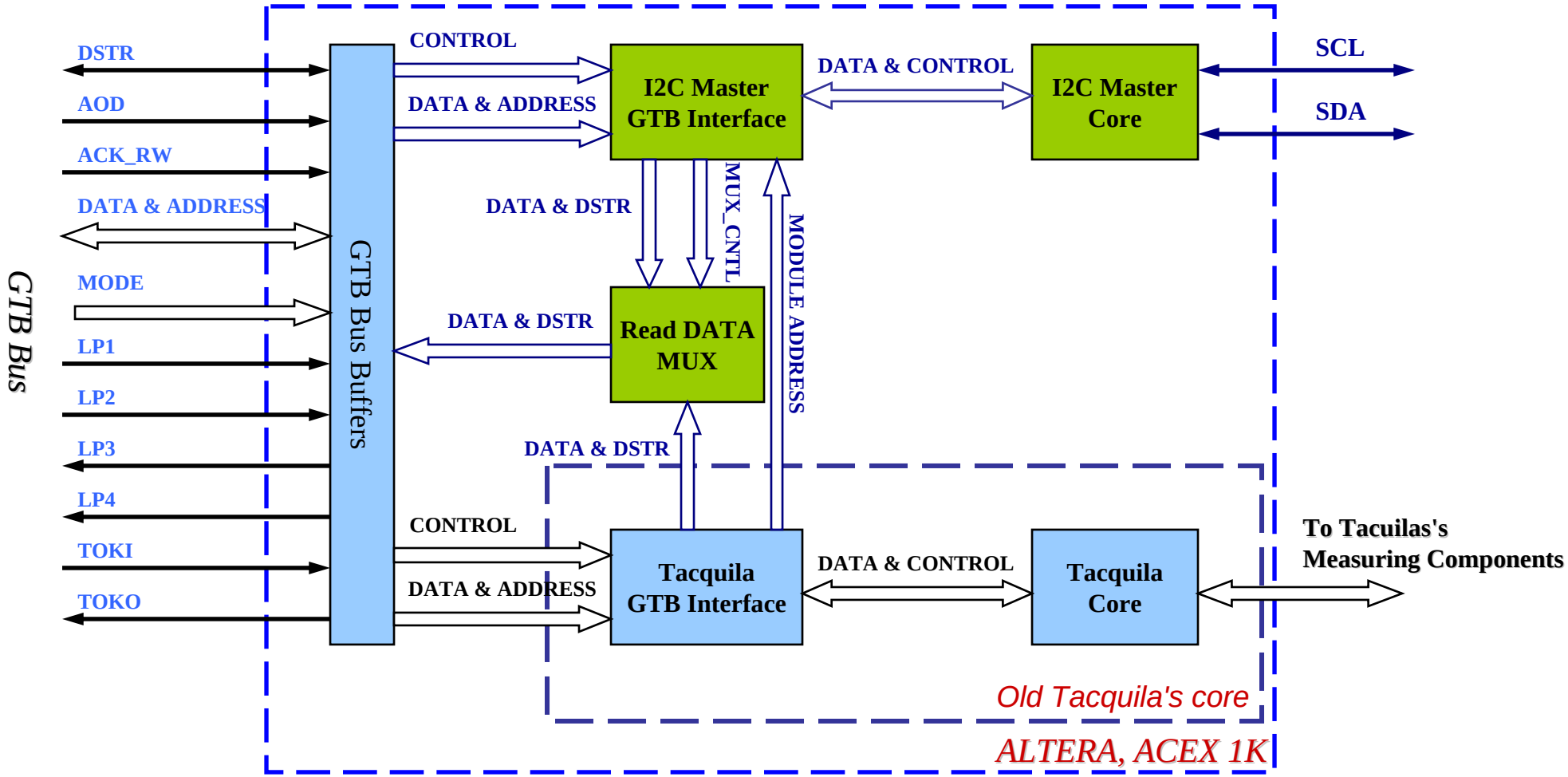
The Start: Jun, 2008.

The End: January, 2009. The *I2C core* is implemented in *Tacquila's FPGA* (Altera, ACEX 1K) and tested.

Dr. Ivan Rusanov
Experiment-Electronics
GSI, Darmstadt

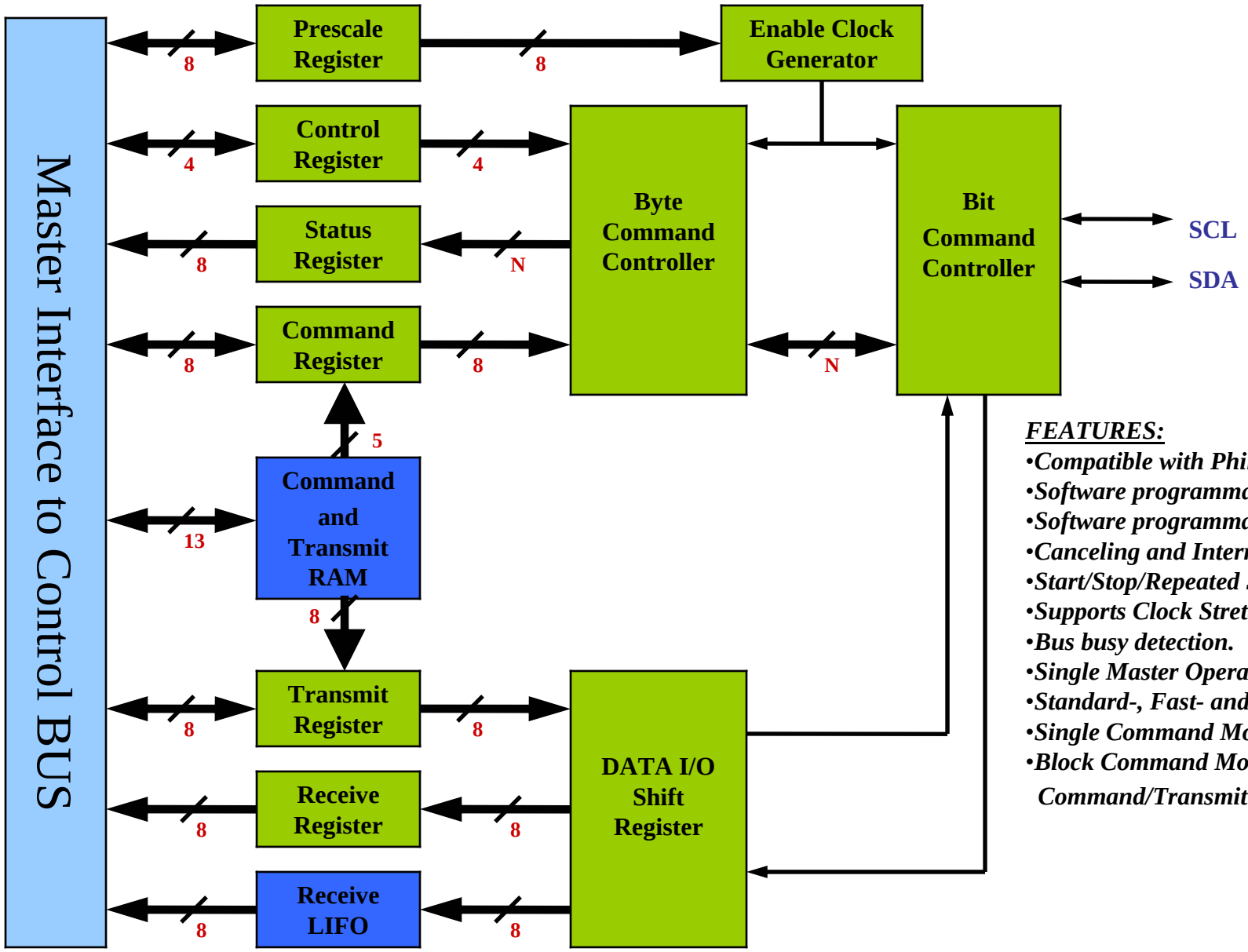
I2C Master Core with Interface to GTB Bus

(Block Diagram)



Dr. Ivan Rusanov
Experiment-Electronics
GSI, Darmstadt

I2C Master: Block Diagram



FEATURES:

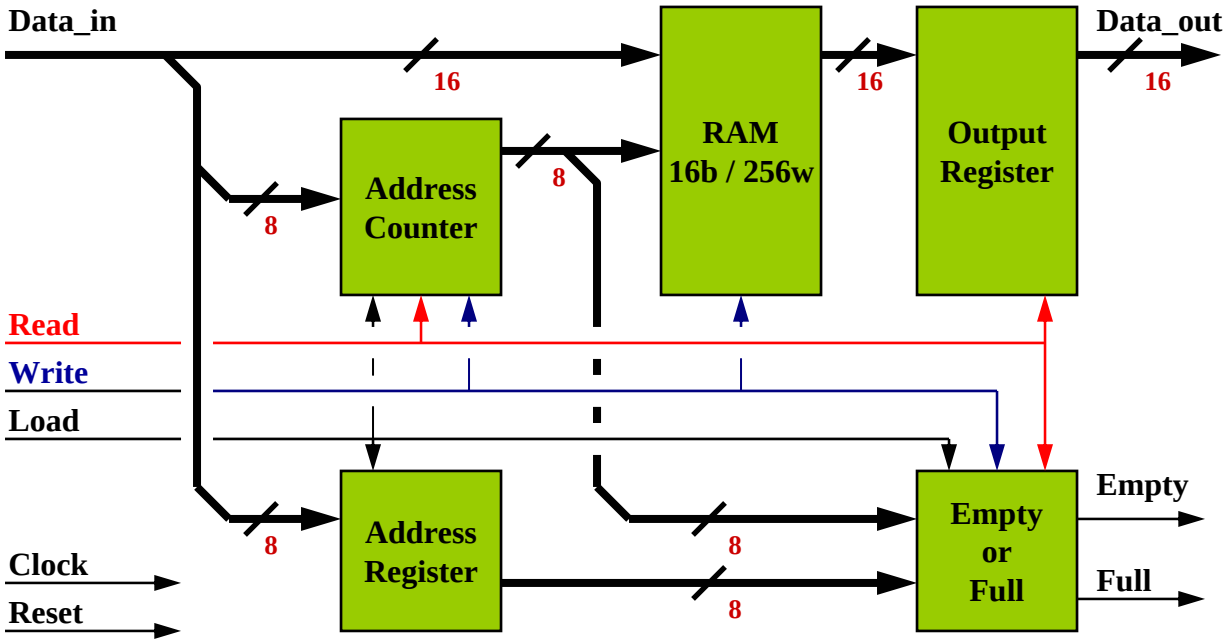
- *Compatible with Philips I2C standard.*
- *Software programmable clock frequency.*
- *Software programmable acknowledge bit.*
- *Canceling and Interrupting of data-transfers.*
- *Start/Stop/Repeated Start/Acknowledge generation.*
- *Supports Clock Stretching/Wait state generation.*
- *Bus busy detection.*
- *Single Master Operation.*
- *Standard-, Fast- and High-Speed Modes*
- *Single Command Mode.*
- *Block Command Mode, using Command/Transmit RAM and Receive LIFO.*

Dr. Ivan Rusanov
Experiment-Electronics
GSI, Darmstadt

I2C Master: Command / Transmit RAM

1. The ***Receive LIFO*** is standard ***LIFO memory*** (8b x 512w).

2. The ***Command / Transmit RAM***: This is a ***RAM***, working as a ***FIFO*** (16b x 256w).
   (Each ***Block of Commands*** could be executed a lot of times, no limit.)



Command/Transmit RAM: Block Diagram.

Mapping of "Block of Commands"
(Command / Transmit RAM)

Address *255*	← Address Register	ST0
Block 3	Count: From: Value of Counter To: Value of Register	
Address *N +1*	← Address Counter	STA
Address *N*	← Address Register	ST0
Block 2	Count: From: Value of Counter To: Value of Register	
Address *M +1*	← Address Counter	STA
Address *M*	← Address Register	ST0
Block 1	Count: From: Value of Counter To: Value of Register	
Address *0*	← Address Counter	STA

***Important*: When is executed command *"Load Address Counter and Register"* both flags "*Empty* and *Full*" are cleared.**

Dr. Ivan Rusanov
Experiment-Electronics
GSI, Darmstadt

I2C Master: Registers

Register	Action	Mapped to internal address
Combined Command/Transmit Registers	*Read-Write*	*0x0401*
Command/Transmit RAM	*Read-Write*	*0x0402*
Combined Status /Receive Registers	*Read*	*0x0404*
Receive LIFO	*Read*	*0x0450*
Combined Control/Status Registers	*Read-Write/Read*	*0x0410*
Prescale Register	*Read-Write*	*0x0420*
Combined Address Register/Counter	*Read-Write*	*0x0440*

The full GTB register (or memory) address is sum of slave (with left shifting on 11 bits) and internal addresses:

Example 1: Full GTB Address = 0x0001*0x0800 + 0x0401 = 0x0C01 (1st GTB slave)
Example 2: Full GTB Address = 0x001F*0x0800 + 0x0401 = 0xFC01 (31st GTB slave)

Where:

0x0800 – Shifting coefficient.
0x0401 – Internal address of Combined Command/Transmit Registers.

Dr. Ivan Rusanov
Experiment-Electronics
GSI, Darmstadt

I2C Master: Register DATA on GTB Bus

The I2C register bits are mapped on GTB-bus in following way:

I2C registers	GTB-bus: Data [15 : 0]															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Command/Transmit Register	*	*	*	ACK	STA	STO	RD	WD	D7	D6	D5	D4	D3	D2	D1	D0
or	-or-			**Command Register/RAM**					**Transmit Register/RAM**							
Command/Transmit RAM	*	*	*	ACK	STA	STO	RD	WD	A6	A5	A4	A3	A2	A1	A0	RW
Status/Receive Register	S7	S6	S5	S4	S3	S2	S1	S0	R7	R6	R5	R4	R3	R2	R1	R0
Receive LIFO	*	*	*	*	*	*	*	*	R7	R6	R5	R4	R3	R2	R1	R0
Control Register (write)	*	*	*	*	*	*	*	*	C7	C6	C5	C4	C3	*	*	*
Control/Status Register (read)	C7	C6	C5	C4	C3	*	*	*	S7	S6	S5	S4	S3	S2	S1	S0
Prescale Register	*	*	*	*	*	*	*	*	P7	P6	P5	P4	P3	P2	P1	P0
Address Register/Counter	M7	M6	M5	M4	M3	M2	M1	M1	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
Tacquila – FIFO	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0

Where:

ACK signal transmitted from master to salve. STA – start, STO – stop, RD – read, WD – write (commands).
R [7 : 0] – data, received by the master (Receive Register/LIFO).
C [7 : 3] – data of Control Register.
S [7 : 0] – data of Status Register.
P [7 : 0] – data of Prescale Register.
M [7 : 0] – data of Address Register.
Q [7 : 0] – data of Address Counter.
T [15 : 0] – Tacquila's data.

Dr. Ivan Rusanov
Experiment-Electronics
GSI, Darmstadt

I2C Master Core: The operating via GTB Bus

1. For the operating with I2C core via GTB bus is only foreseen **'GTB16 address mode Read/Write'.**

2. The initializing of I2C Core for mode **'Block of commands'** and executing of it:

• Clear all bits of **Control Register**: GTB writing command onto address 0x0410 (DATA = **0x0000**).
• Clear all bits of **Command/Transmit Register**: GTB writing command onto address 0x0401 (DATA = **0x0000**).
• Set the speed of I2C Bus: GTB writing command onto address 0x0420 (DATA = **0x0018**).
• Set RAM's area for a **'Block of commands'** writing start address and end address (**Address Register/Counter**): GTB writing command onto address 0x0440.
• Execute ***N***-times GTB writing command onto address 0x0402 (RAM). The amount of cycles is defined by the start values of **Address Counter** and **Address Register**:

N = [Address Register] – [Address Counter] + 1, N_{max} = *256 words (commands)*.

Important: Do not do more that N writing cycles. The GTB master will hang the GTB-bus.

• Set RAM's area for a **'Block of commands':** GTB writing command onto address 0x0440.
• Set the proper values of all **Control Register**'s bits: GTB writing command onto address 0x0410 (DATA = **0x0090**).

The I2C bus is activated and the communication between I2C core and slaves is started

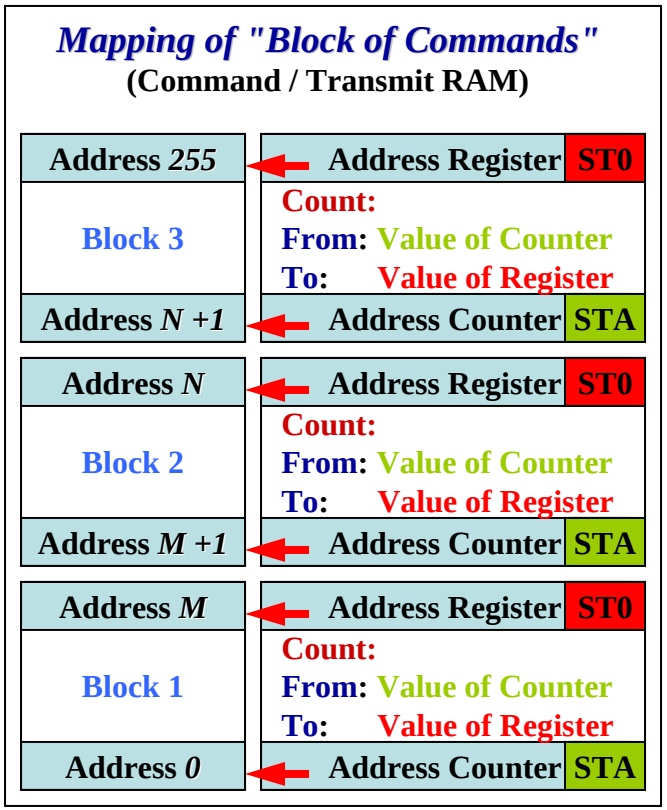
• Read data from **Control/Status Register**: GTB reading command from address 0x0410 (DATA = **0x9022**).
• If there is data in **Receive LIFO**, read it: GTB reading command from address 0x0450 (DATA = **0x00xx**).
• The last read 16 bit word from **Receive LIFO**: DATA = **0x22xx**. When this word is received the reading has to stop**.**

Important: Do not do more reading cycles. The GTB master will hang the GTB-bus.

• Clear all bits of **Control Register**: GTB writing command onto address 0x0410 (DATA = **0x0000**).

Dr. Ivan Rusanov
Experiment-Electronics
GSI, Darmstadt

I2C Master Core: The executing of more Blocks



• A lot of times without limit each **'Block of commands'** can be executed**:**

1. If the I2C core is active (**Control Register**: DATA = **0x0090**), then:
Set RAM's area for a **'Block of commands'**, and the executing is started.

2. If the I2C core is not active (**Control Register**: DATA = **0x0000**), then:
Set RAM's area for a **'Block of commands'**.
Set the proper values of all **Control Register**'s bits.
(**Control Register**: DATA = **0x0090**) , and the executing is started.

• Partially can be executed each **'Block of commands' (Block 1):**

1. If the I2C core is active (**Control Register**: DATA = **0x0090**), then:
Set RAM's area for a **'Half Block of commands'**.
(**Address Register/Counter** : DATA = **0x'L';'0'**) , and the executing is started.
Set RAM's area for a rest of **'Block of commands'**.
(**Address Register/Counter** : DATA = **0x'M';'L+1'**) , and the executing is started.

2. If the I2C core is not active (**Control Register**: DATA = **0x0000**), then:
Set RAM's area for a **'Half Block of commands'**.
(**Address Register/Counter** : DATA = **0x'L';'0'**)
Set the proper values of all **Control Register**'s bits.
(**Control Register**: DATA = **0x0090**) , and the executing is started.
Set RAM's area for a rest of **'Block of commands'**.
(**Address Register/Counter** : DATA = **0x'M';'L+1'**) , and the executing is started.

Dr. Ivan Rusanov
Experiment-Electronics
GSI, Darmstadt

That is all!

Thank you!

Dr. Ivan Rusanov
Experiment-Electronics
GSI, Darmstadt