Joël Cathébras

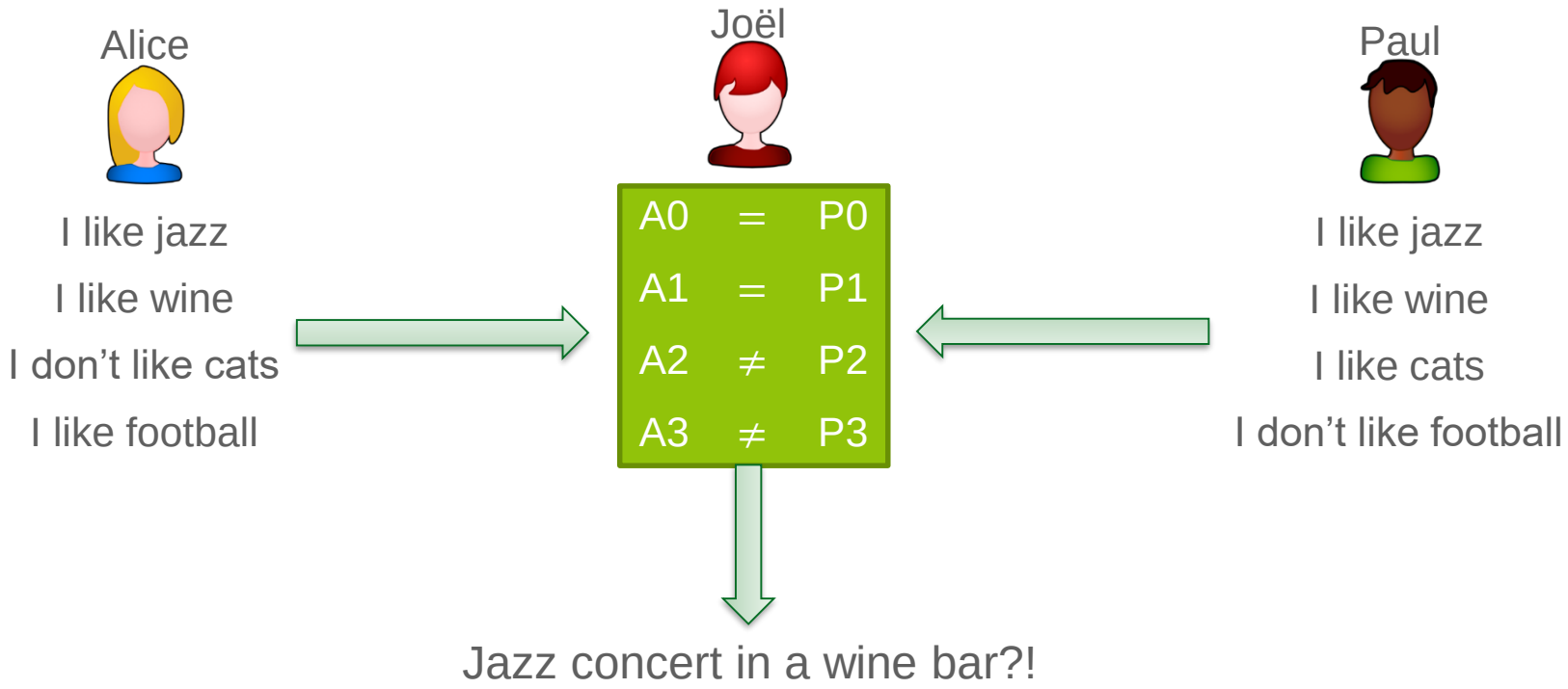
Vendredi 14 décembre 2018

Séminaire sécurité des systèmes électroniques embarqués

Campus de Beaulieu, 263 avenue du Général Leclerc, Rennes

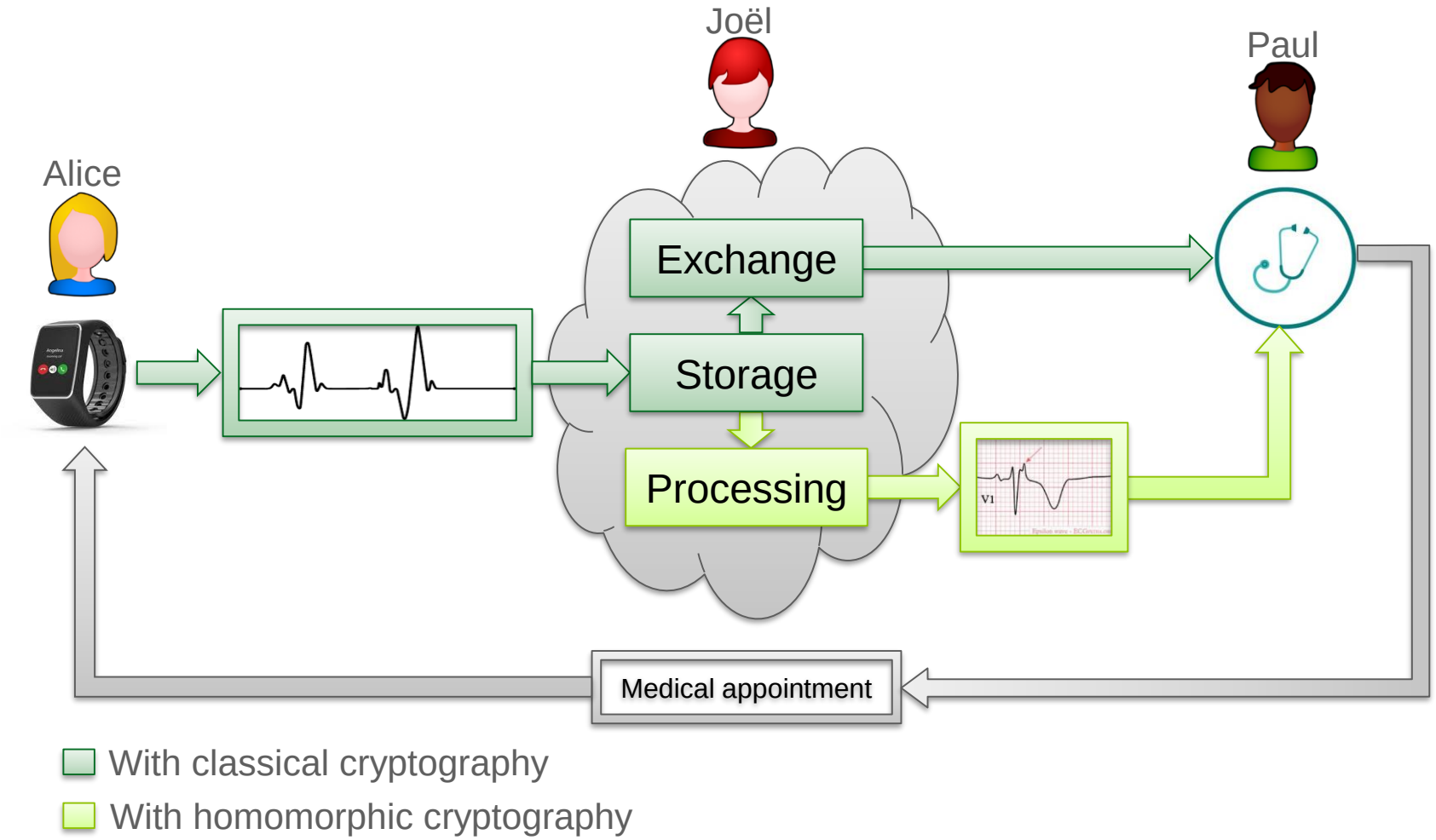
AN APPROACH FOR THE HARDWARE ACCELERATION OF HOMOMORPHIC CRYPTOGRAPHY

THE PERFECT NEUTRAL MATCHMAKER



Homomorphic encryption: processing encrypted data without decrypting them!

A MORE SERIOUS EXAMPLE: THIRD-PARTY MEDICAL MONITORING

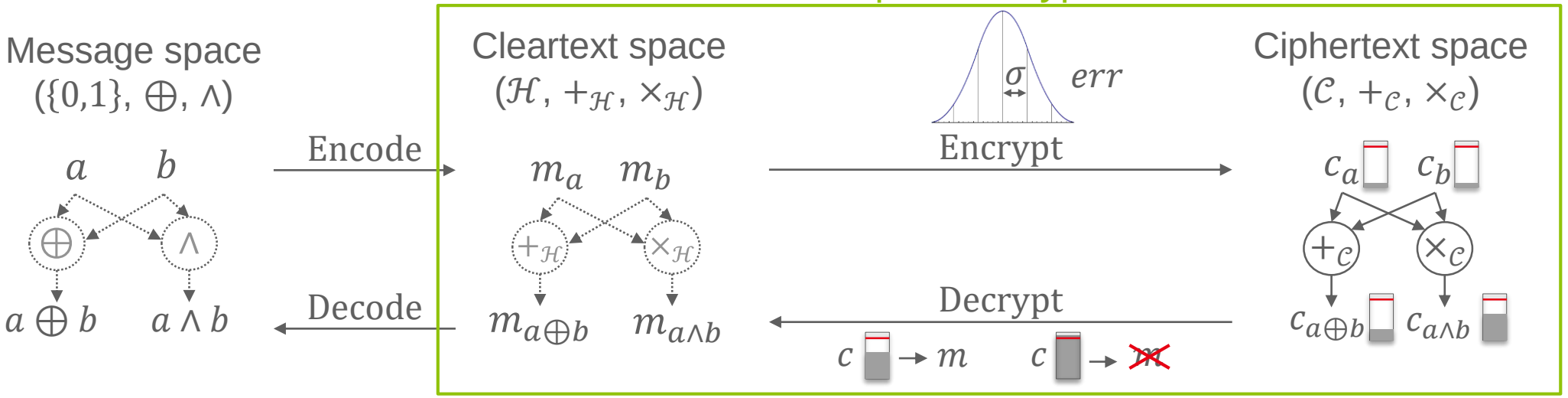


MAIN PROBLEMATICS OF HOMOMORPHIC CRYPTOGRAPHY

Decryption is an homomorphism: for all ciphertexts ct_a and ct_b

$$\mathrm{Dec}(\mathrm{ct}_a \circledast \mathrm{ct}_b) = \mathrm{Dec}(\mathrm{ct}_a) * \mathrm{Dec}(\mathrm{ct}_b) = m_a * m_b$$

Homomorphic encryption scheme



Theoretical problematics

Find decryption homomorphism

Noise management for correctness

Practical problematics

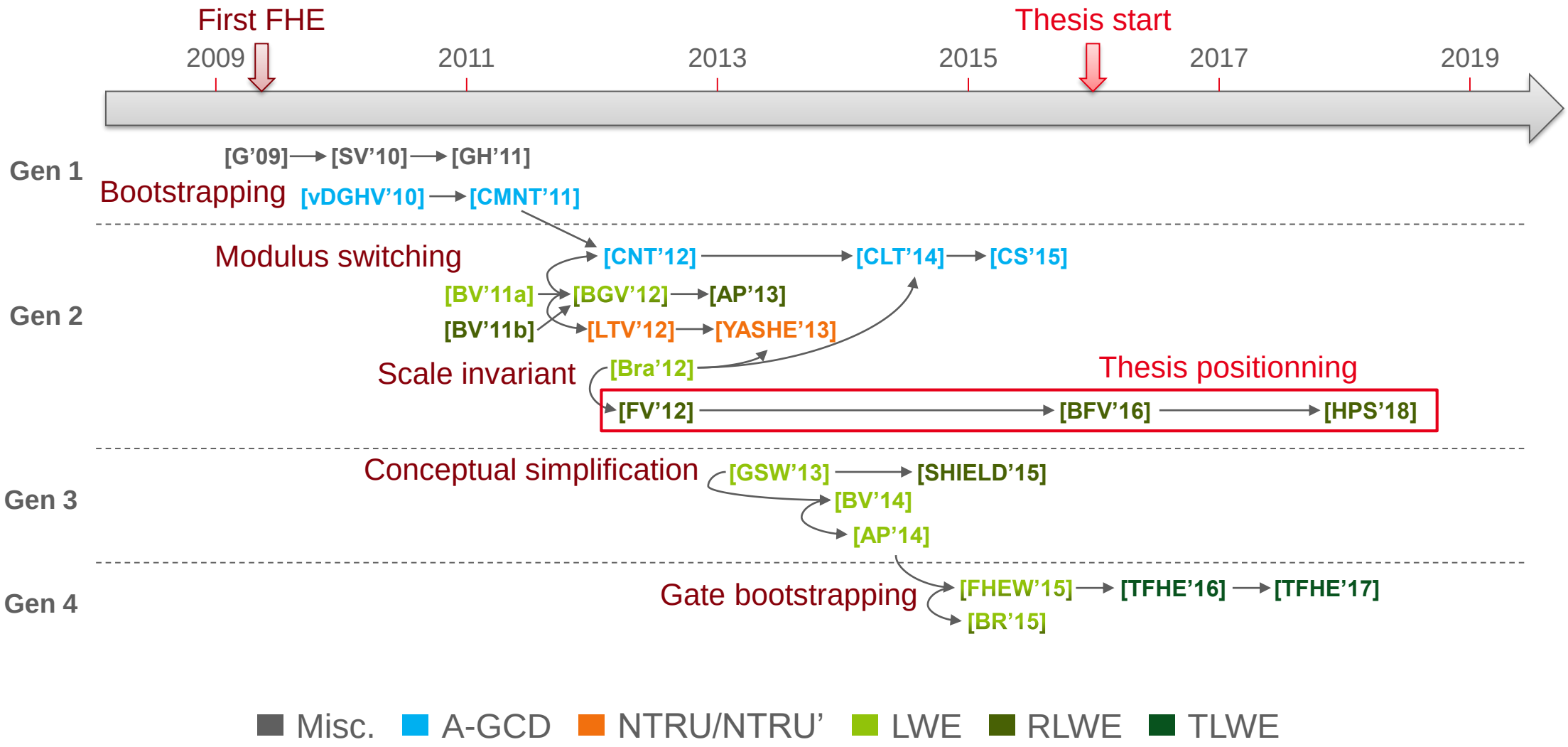
Data size expansion (1 bit ⟶ ~ 10 kbit)

Computational complexity (1 AND ⟶ ~ ms)

Different generation of HE schemes

Need hardware acceleration

STATE OF THE ART OF HOMOMORPHIC SCHEMES



MAIN PROBLEMATIC OF RLWE-BASED L-FHE SCHEMES

Ring Learning With Errors: handling polynomial ring elements

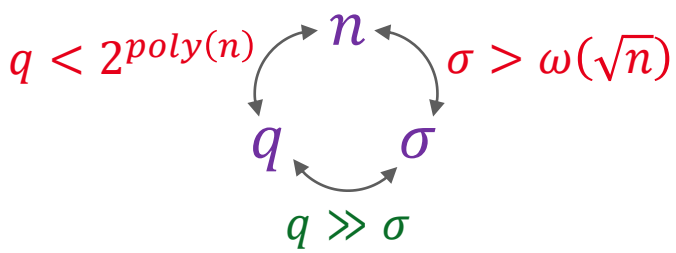
$$\left.\begin{array}{l} \boldsymbol{R}_q = \mathbb{Z}_q[X]/(F(X)) \\ \boldsymbol{S} \in \boldsymbol{R}_q \\ \boldsymbol{A} \leftarrow \boldsymbol{R}_q \text{ and } \boldsymbol{E} \leftarrow \chi(\sigma) \end{array}\right\} \xrightarrow{\text{Construct pairs}} (\boldsymbol{A}, \boldsymbol{B}) \in \boldsymbol{R}_q^2$$

($\mathbb{Z} \bmod q$, $\deg(F) = n$)

$$(\boldsymbol{A}, \boldsymbol{B}) = (\boldsymbol{A}\boldsymbol{S} + \boldsymbol{E}) \bmod q + \textit{message}$$

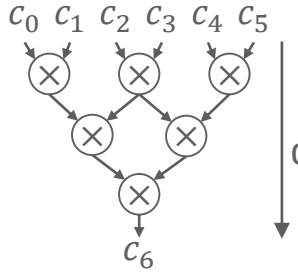
The problem is … → **Decision**: Distinguish $(\boldsymbol{A}, \boldsymbol{B})$ from a random pair in $\boldsymbol{R}_q^2$

Security and correctness depend on parameters:



n increases security & q increases correctness

Leveled-FHE parameters depend on application:



Mult. depth ⇒ q ~ several hundred of bit; n ~ several thousand

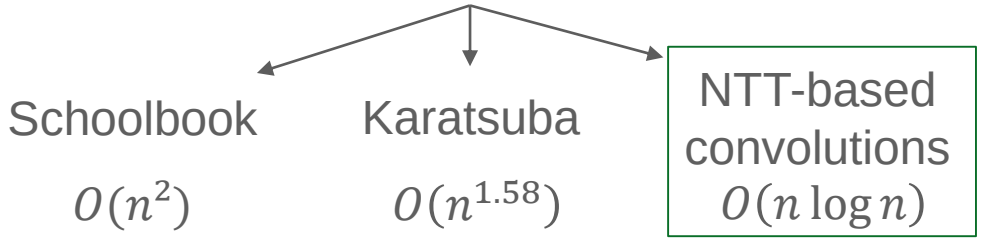
Grow with the complexity of encrypted evaluation

Flexible acceleration strategy for { Polynomial arithmetic over $\boldsymbol{R}_q$; Integer arithemetic over $\mathbb{Z}_q$ }

STATE OF THE ART OF HARDWARE ACCELERATION FOR RLWE-BASED L-FHE SCHEMES

Handling large n

polynomial multiplications

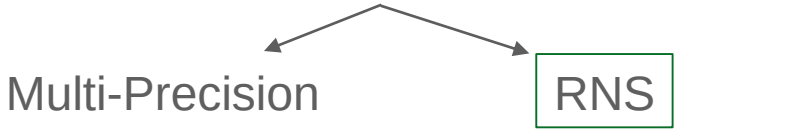


Thesis positionning

Handling large q

large arithmetic

divisions & modular reductions



Thesis positionning

Among related works on coupling RNS and NTT strategies

Öztürk et al. 2015

Memory-access iterative NTT

External computation of twiddle factors

Doubling communication bandwidth

Cousins et al. 2017

Dataflow oriented NTT

Local storage of twiddle factors

High storage cost

Sinha Roy et al. 2015

Memory-access iterative NTT

On-the-fly computation of twiddle factors.

Generation insert bubbles

impacting NTT throughput

Dataflow oriented NTT-based convolutions with on-the-fly computation of twiddle factors

OUTLINE

Analysis of the FV scheme towards its hardware acceleration.

Proposal of a data flow oriented Residue Polynomial Multiplier (RPM).

In-depth on our design of fully-streaming multi-field NTT circuits.

Analysis of the scalability of our NTT-based RPM.

Other contributions, conclusion and perspectives.

ANALYSIS OF FV TOWARD ITS HARDWARE ACCELERATION (1)

PROFILING OF FV HOMOMORPHIC EVALUATION

Performance bottlenecks of FV homomorphic evaluation?

Profiling of an homomorphic evaluation of Trivium

Trivium-12	#steps	#AND	#XOR
WarmUp	1,152	3,456	12,672
KSGen	57	171	798

Trivium stream-cipher
Generation of 57 keystream elements

HE Primitives	Estimated cycles	Estimated %
Total Trivium	$\mathbf{26.84 \times 10^{12}}$	**100 %**
Mult&Relin	$\mathbf{26.68 \times 10^{12}}$	**99.36 %**
- Relin	14.25×10^{12}	53.07 %
- Mult	11.40×10^{12}	42.45 %
- Misc.	1.03×10^{12}	3.84 %
CtxtAdd	$\mathbf{0.14 \times 10^{12}}$	**0.54 %**
Misc.	$\mathbf{0.02 \times 10^{12}}$	**0.1 %**

FV implementation from the Cingulata compiler
$\lambda < 80,\ L = 19,\ n = 8192,\ \log_2 q = 913,\ \log_2 \sigma = 383$
Valgrind 3.10 on Intel Core i7-3770.

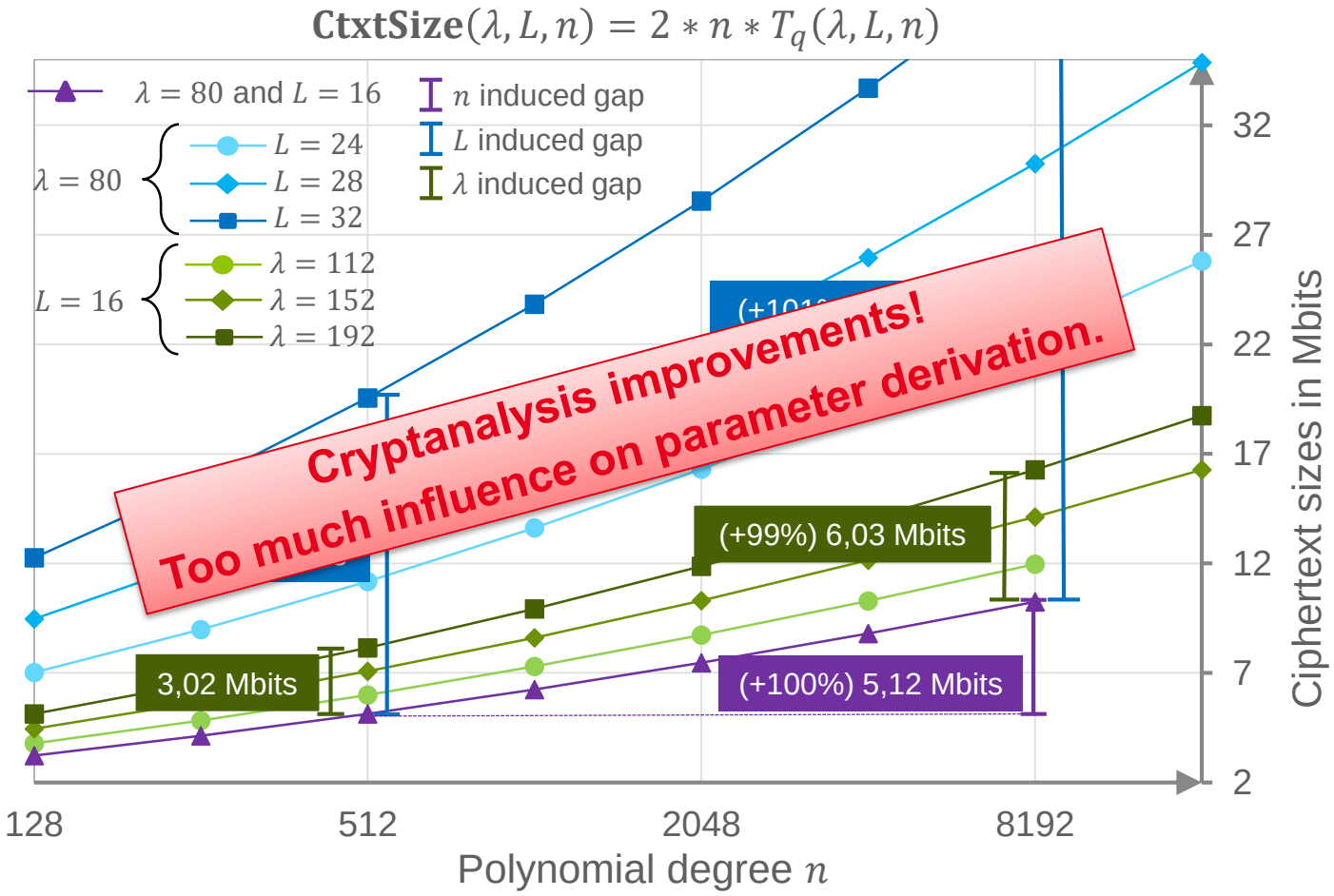
Hardware acceleration should target ciphertext multiplications

ANALYSIS OF FV TOWARD ITS HARDWARE ACCELERATION (2)

FV PARAMETERS ANALYSIS

Is there parameter choices more suitable for hardware acceleration?

Parameter analysis w.r.t. security and multiplicative depth requirements



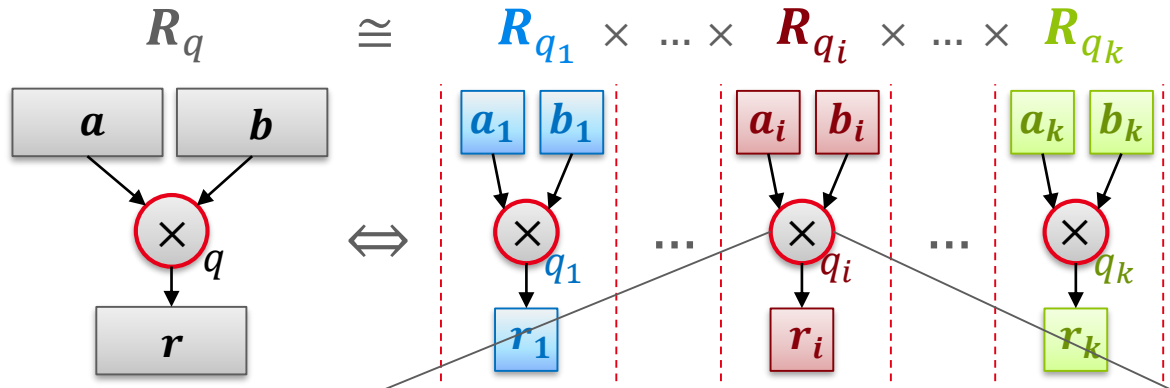
Change of methodology

Define hardware acceleration strategy suitable for the wide range of FV parameters

ANALYSIS OF FV TOWARD ITS HARDWARE ACCELERATION (3)

PRESENTATION OF THE RNS/NTT STRATEGY

Residue Number System (RNS):



RNS basis

$$\mathcal{B} = \{q_1, \ldots, q_k\}$$

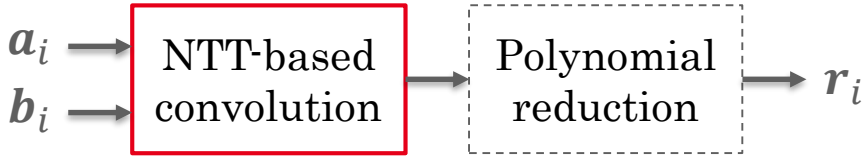
$$\mathcal{B}' = \{q'_1, \ldots, q'_{k'}\}$$

$$q = \prod_{\in \mathcal{B}} q_i$$

$$2nq^2 < \prod_{\in \mathcal{B}} q_i \prod_{\in \mathcal{B}'} q'_i$$

Number Theoretical Transform for R_{q_i} multiplications:

$$(a_i, b_i) \in R_{q_i}$$



Existence if $2n \mid q_i - 1$ & $O(n)$ twiddle factors

Which RNS basis elements for efficient implementation of the RNS/NTT strategy?

ANALYSIS OF FV TOWARD ITS HARDWARE ACCELERATION (4)

VALIDATION OF THE RNS/NTT STRATEGY

Is RNS/NTT strategy possible up to very large FV parameter sets?

Look for suitable RNS basis element respecting the strategy constraints

Constraints on RNS basis elements:

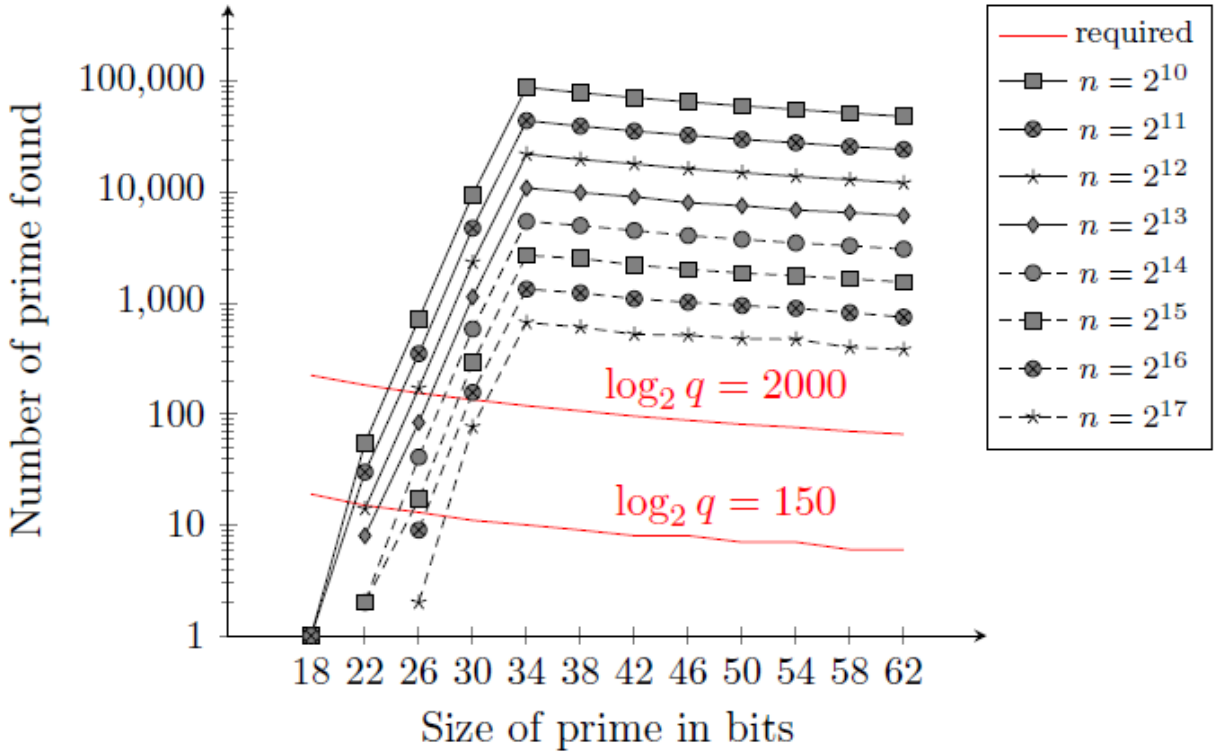
Must be co-primes

Must guarantee NTT existence

Should allow efficient modular arithmetic

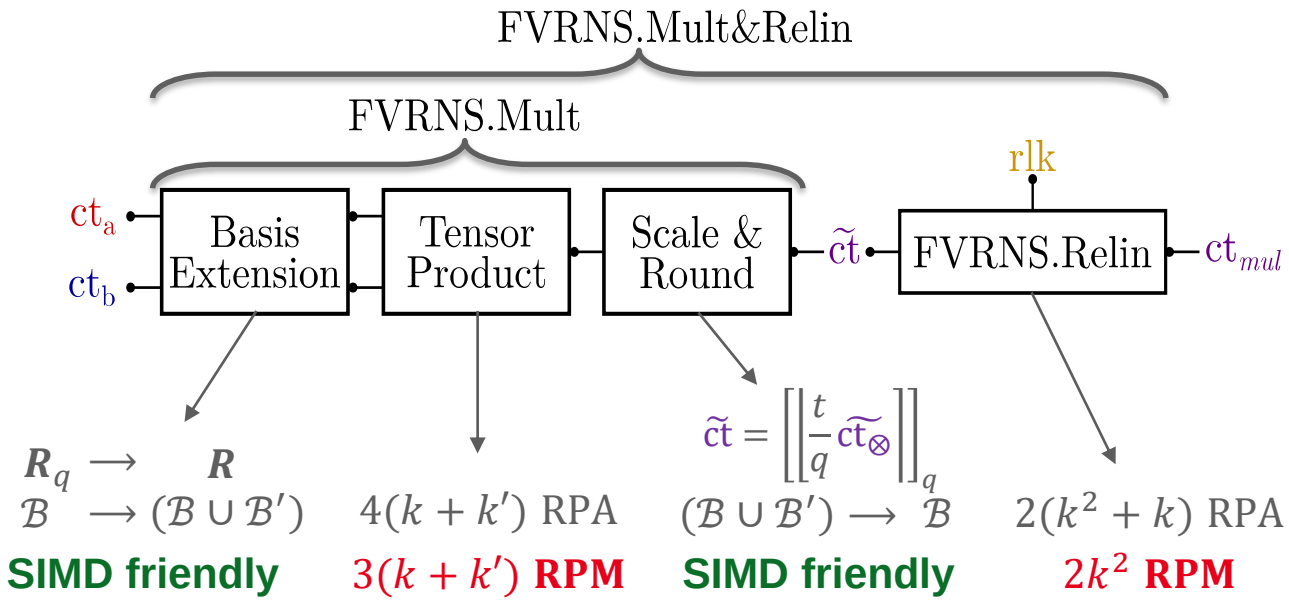
Should have tunable size

NFLlib prime selection



Enough primes > 32-bit for very large parameters ($\lambda > 128, L{\sim}100$)

THE FULL RNS VARIANT OF FV



Complexity partition w. r. t. the Mult&Relin operation

RNS specific functions	RPM
$\sim [20 - 35]\%$	$\sim [60 - 75]\%$

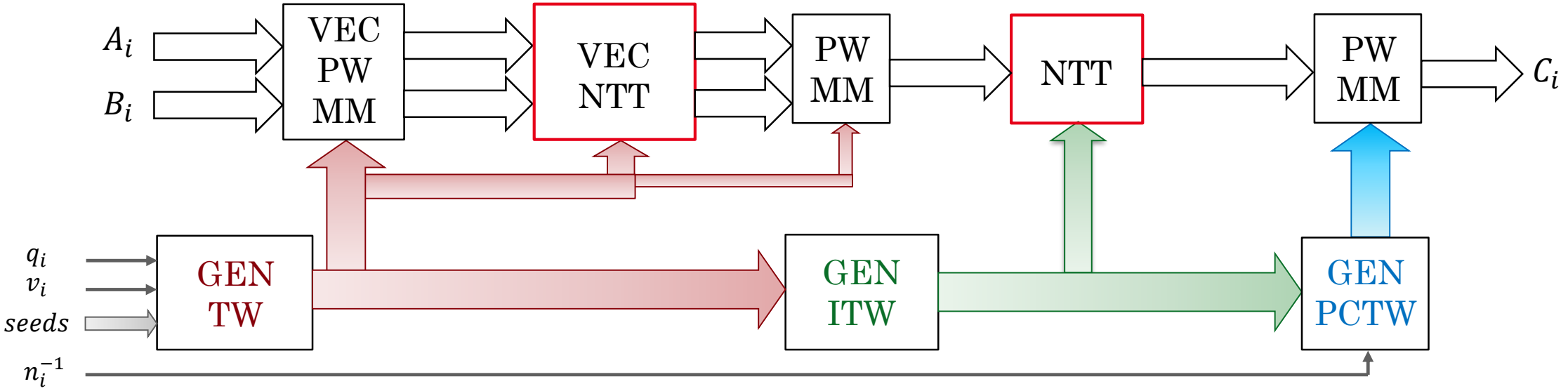
Dedicated hardware for RPM → **Several tenth of different finite-fields** →

Need high throughput RPM designs — /!\ SotA issues! → **On-the-fly handling of twiddle factors**

DATA FLOW ORIENTED RPM THROUGH NEGATIVE WRAPPED CONVOLUTION

ARCHITECTURE PRINCIPLE

- **Negative Wrapped Convolution over $\mathbb{Z}_{q_i}^n$ $\Leftrightarrow$ RPM over $\mathbb{Z}_{q_i}[X]/(X^n+1)$:**
  - Let ψ_i be a n-th primitive root of -1 over $\mathbb{Z}_{q_i}^*$, exists if $2n$ divides $q_i - 1$.



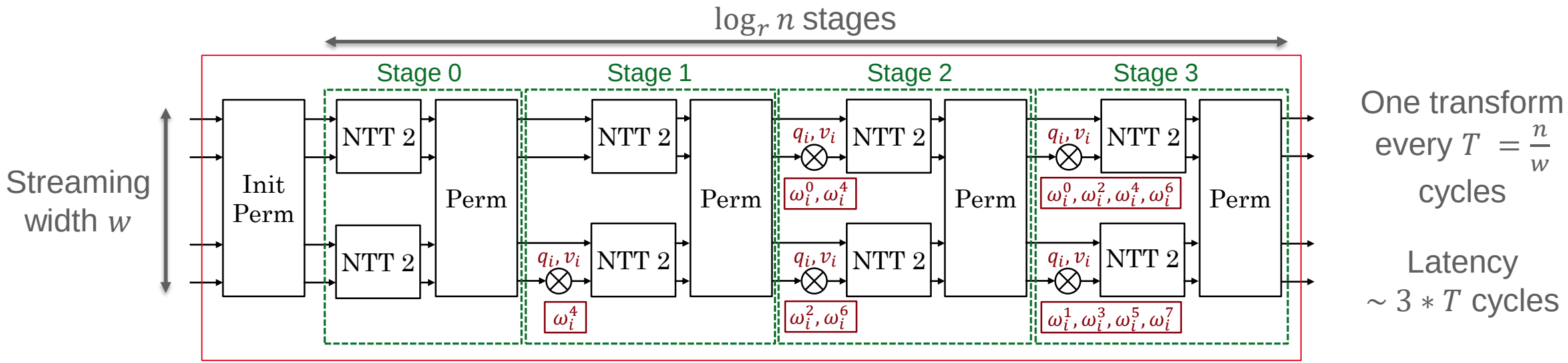
High throughput NTT circuits with on-the-fly change of twiddle factor sets

TOWARDS AUTOMATIC GENERATION OF MULTI-FIELD NTT DESIGN (1)

PRINCIPLE OF A TWIDDLE BANK

SPIRAL DFT hardware generator ⇒ Multi-field NTT hardware generator

Example of NTT ($r = 2, n = 16, w = 4$)



One transform every $T = \frac{n}{w}$ cycles

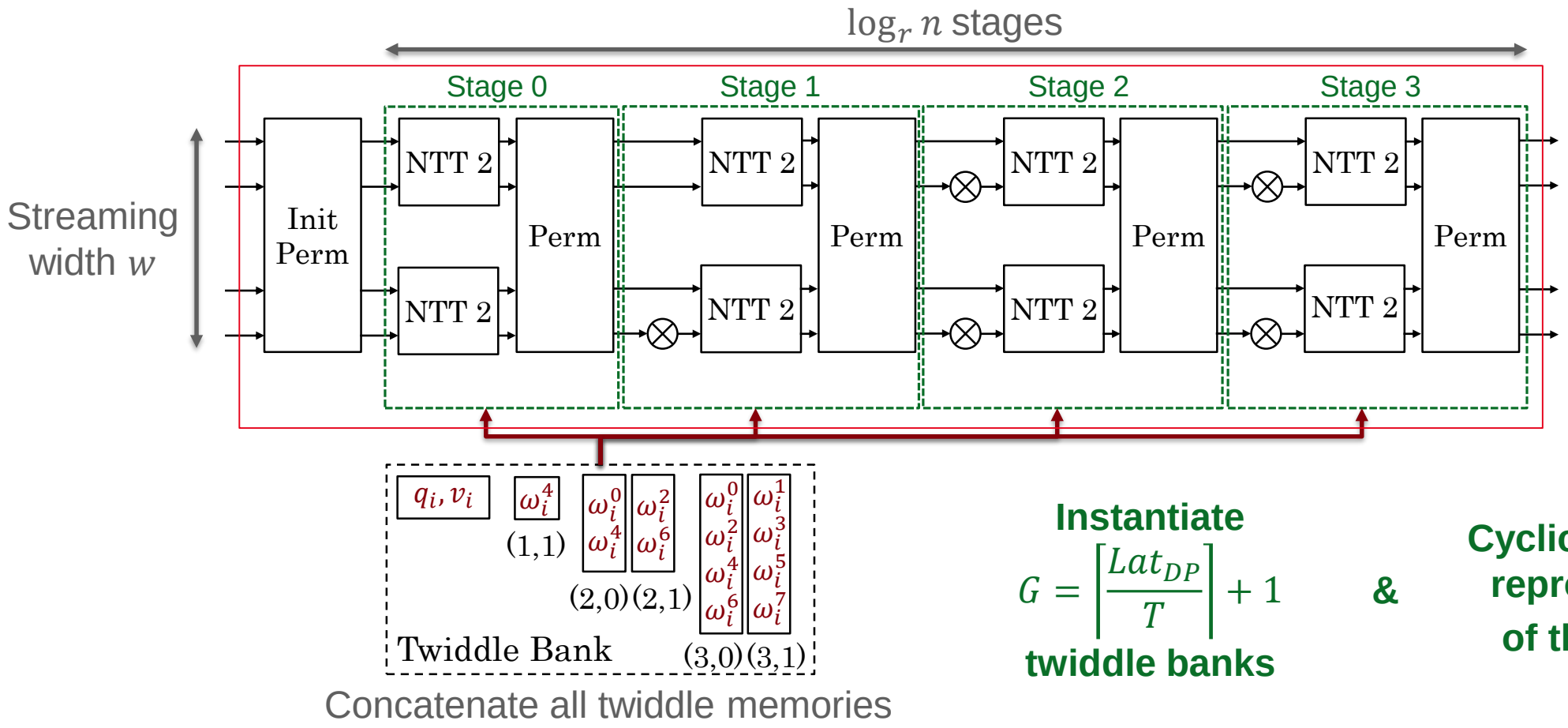
Latency $\sim 3 * T$ cycles

TOWARDS AUTOMATIC GENERATION OF MULTI-FIELD NTT DESIGN (1)

PRINCIPLE OF A TWIDDLE BANK

SPIRAL DFT hardware generator ⇒ Multi-field NTT hardware generator

Example of NTT ($r = 2, n = 16, w = 4$)



Instantiate

$$G = \left\lceil \frac{Lat_{DP}}{T} \right\rceil + 1$$

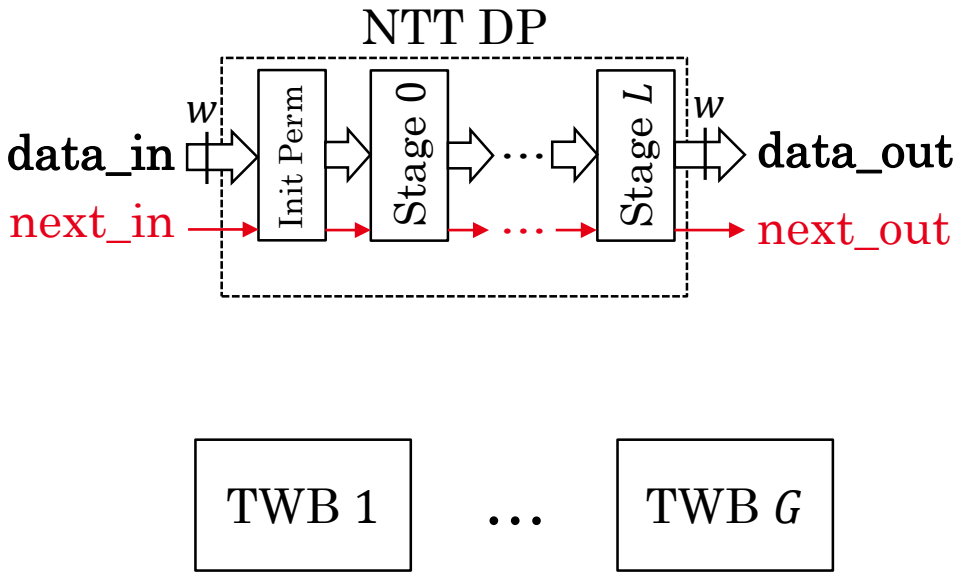
twiddle banks

&

Cyclic access and reprogramming of the G TWBs

TOWARDS AUTOMATIC GENERATION OF MULTI-FIELD NTT DESIGN (2)

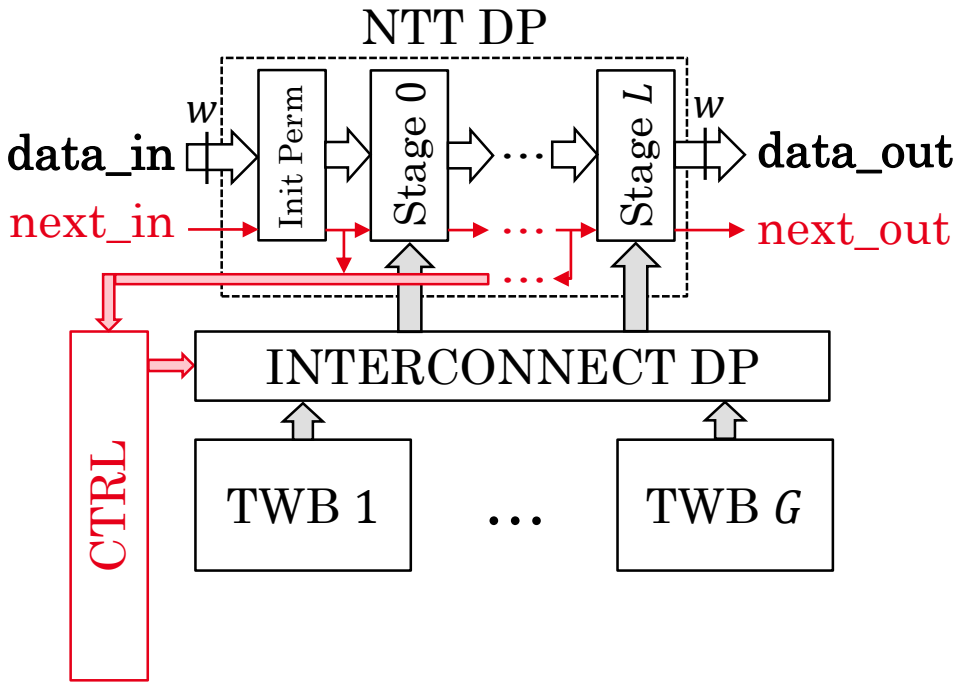
DESIGN OF THE TWIDDLE PATH



⇨ data flow

TOWARDS AUTOMATIC GENERATION OF MULTI-FIELD NTT DESIGN (2)

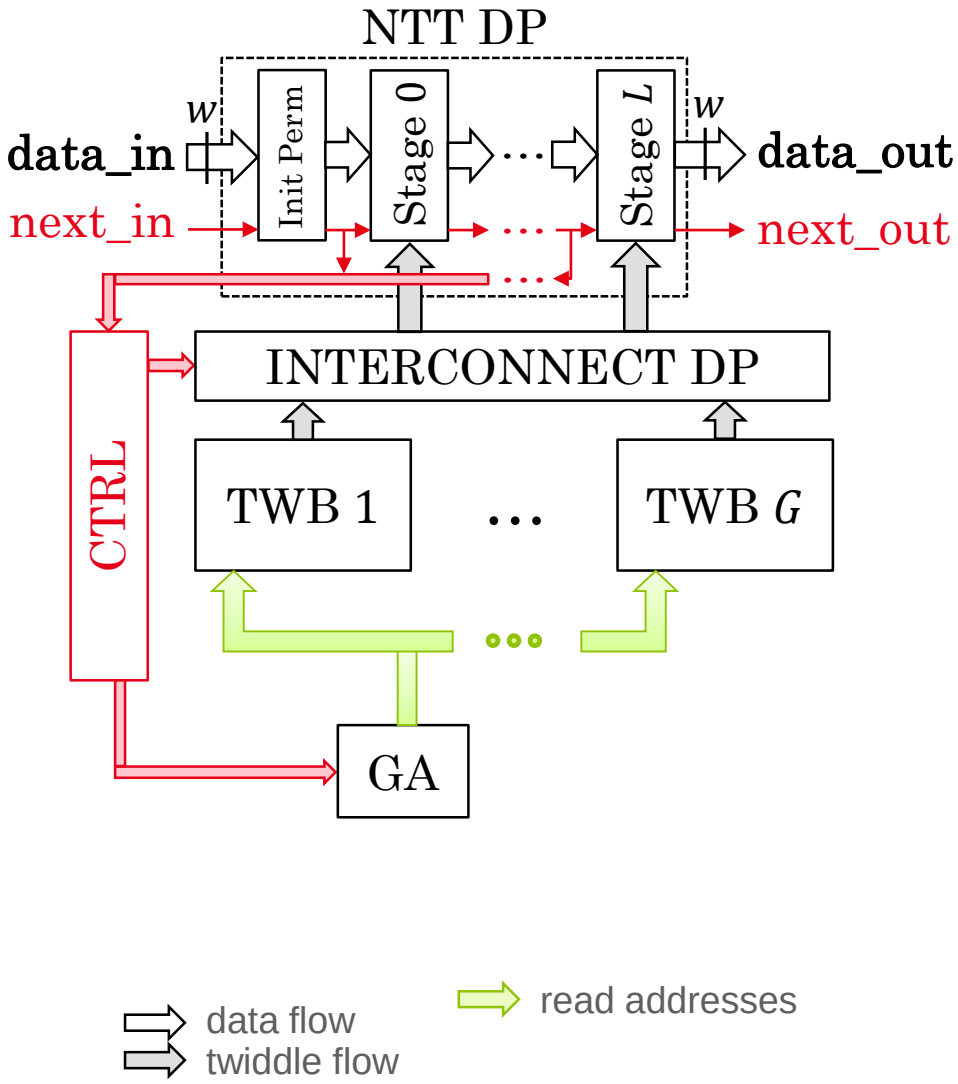
DESIGN OF THE TWIDDLE PATH



data flow
twiddle flow

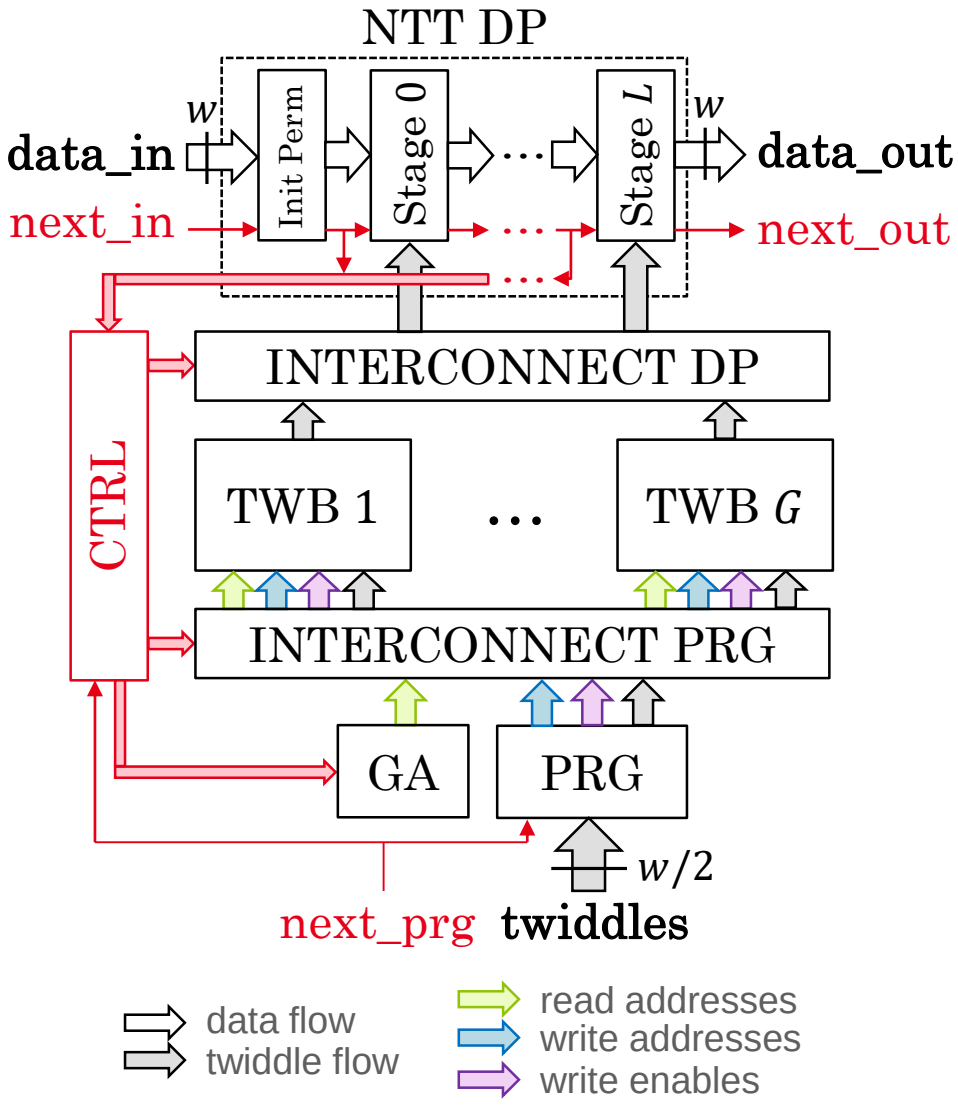
TOWARDS AUTOMATIC GENERATION OF MULTI-FIELD NTT DESIGN (2)

DESIGN OF THE TWIDDLE PATH

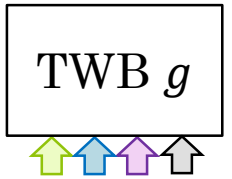


TOWARDS AUTOMATIC GENERATION OF MULTI-FIELD NTT DESIGN (2)

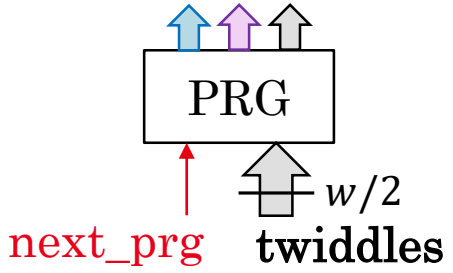
DESIGN OF THE TWIDDLE PATH



How to make a TWB reprogrammable?



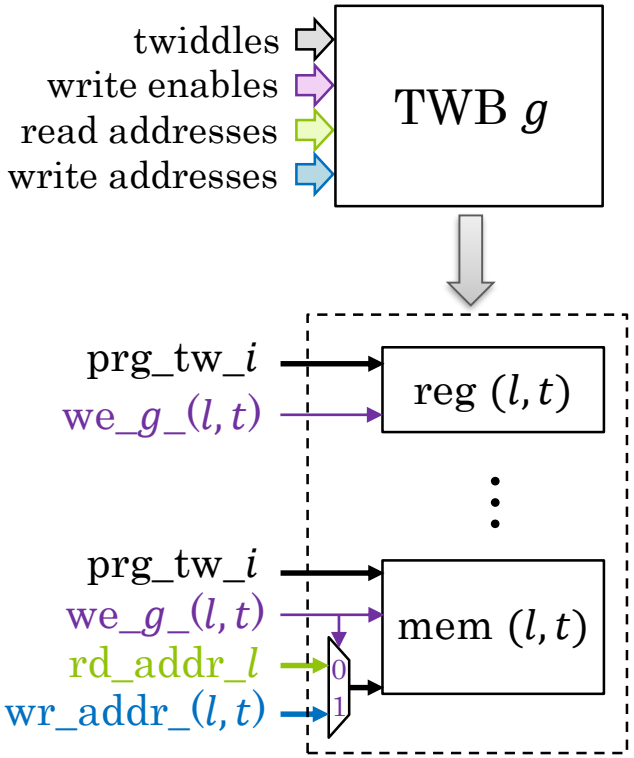
How to dispatch the twiddle factors in the TWB memory elements?



AUTOMATIC GENERATION OF MULTI-FIELD NTT DESIGN (3)

REPROGRAMMING OF TWIDDLE BANKS

Making a TWB reprogrammable



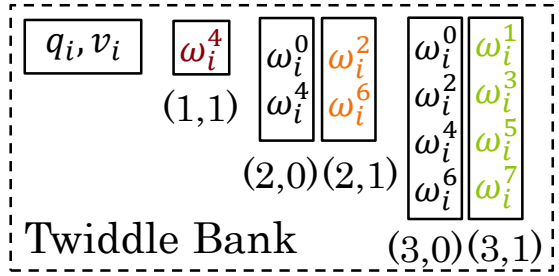
l : stage index

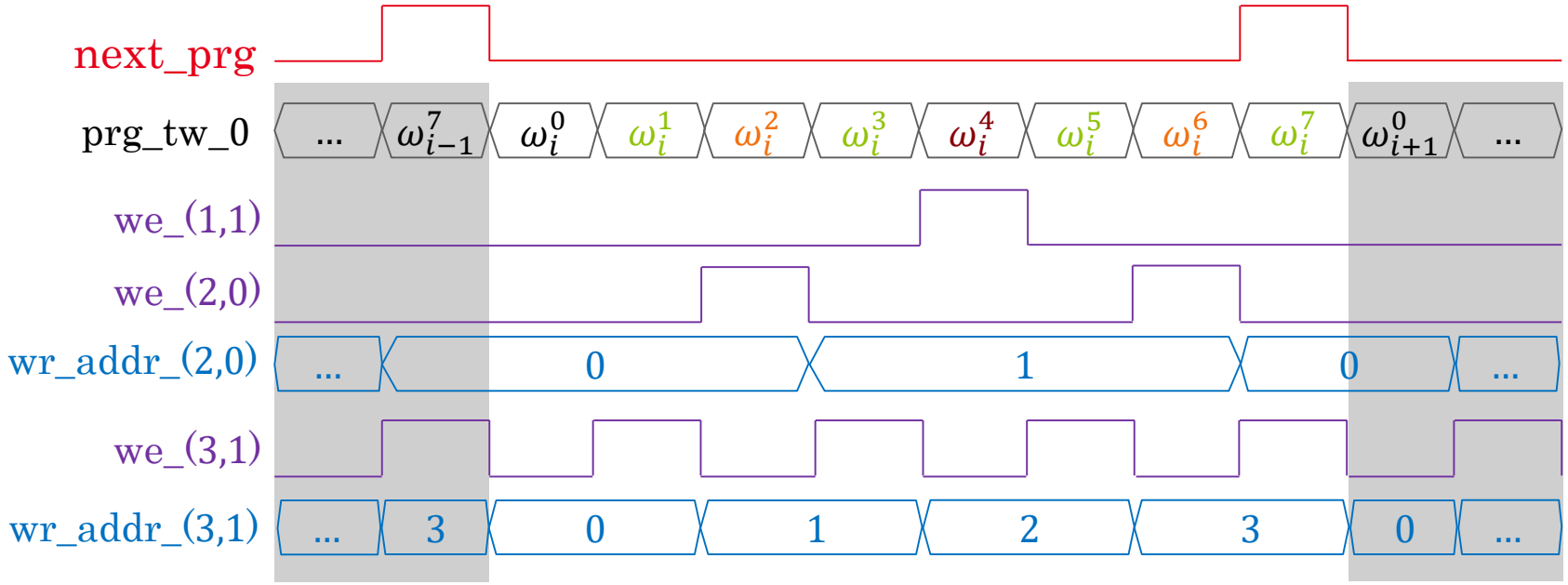
t : way index (in $0, \dots, \frac{w}{2} - 1$)

i : programming flow index

Programming module: in flow generation of signals

Example





Example of programming signals

AUTOMATIC GENERATION OF MULTI-FIELD NTT DESIGN (4)

TWIDDLE PATH SYNTHESIS RESULTS

L: Multiplicative depth
n: deg(F)
S_q: $\log_2 q$
s: $\log_2 q_i$
k, k': RNS basis sizes
w: streaming width

(n, w, s)	G	Res.	Our Twiddle Path %	Our Twiddle Path Total	STWB	Misc.	TWB
(2^{12}, 2, 30)	4	LUTs	**0.59**	**2,567**	2,043	523	**562**
		FFs	**0.45**	**3,912**	3,512	400	**944**
		BRAMs	**1.9**	**28**	28	–	**7**
(2^{14}, 2, 30)	4	LUTs	**0.71**	**3,055**	2,350	705	**658**
		FFs	**0.52**	**4,495**	3,968	[illegible]	**1,083**
		BRAMs	**5.17**	**76**	76	[illegible]	**19**
(2^{14}, 8, 30)	4	LUTs	**1.83**	**7,950**	6,117	1,830	**1,581**
		FFs	**1.7**	**14,658**	13,568	994	**3,458**
		BRAMs	**7.62**	**112**	112	–	**28**
(2^{14}, 8, 46)	4	LUTs	**2.5**	**10,826**	8,997	1,826	**2,301**
		FFs	**2.47**	**21,350**	20,064	1,174	**5,082**
		BRAMs	**16.3**	**240**	240	–	**60**

$\times G$

Synthesis result of our twiddle path

Vivado 2018.1 targeting Xilinx Virtex 7 XC7VX690T-2-FFG1157C

Comparison with local storage of all twiddle factors

Parameters

L	n	S_q	s	$k+k'$	w	Local Storage MB	Local Storage BRAM	Our Twiddle Path MB	Our Twiddle Path BRAM(%)
1	2^{11}	54		5		0.31	25	0.25	**20 (-20 %)**
5	2^{12}	108		8		0.98	56	0.49	**28 (-56 %)**
10	2^{13}	216	30	16	2	3.93	176	0.98	**44 (-75 %)**
20	2^{14}	432		30		14.75	570	1.97	**76 (-87 %)**
30	2^{15}	594		41		40.30	1,394	3.93	**136 (-90 %)**

G times more costly than for a single-field NTT

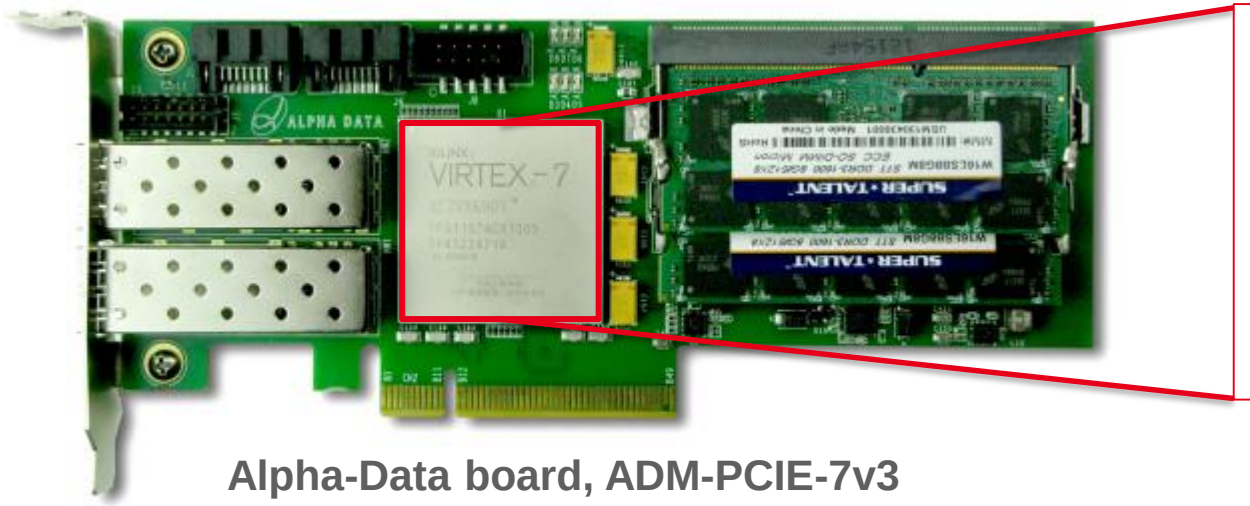
Empirically $4 \leq G \leq 6$

Up to -90 % of memory utilization compared to some state-of-art approaches

No impact on NTT throughput

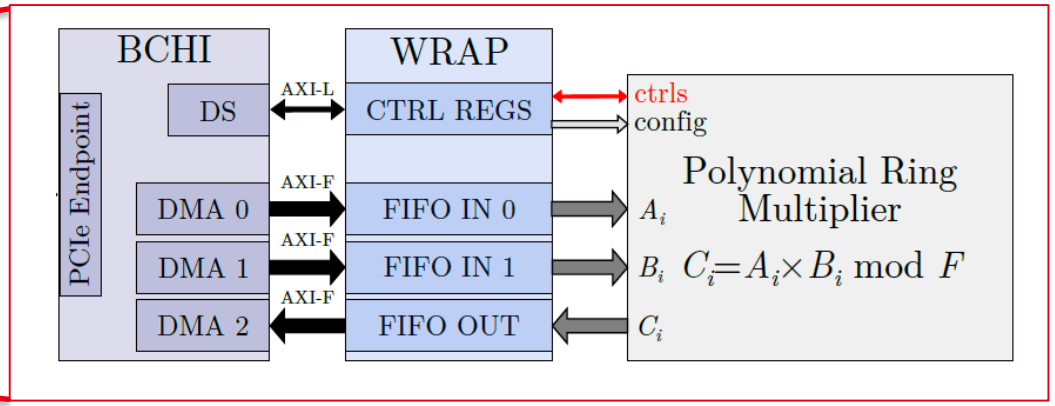
RPM CHARACTERIZATION (1)

PROOF-OF-CONCEPT INTEGRATION



Alpha-Data board, ADM-PCIE-7v3

Xilinx Virtex 7 xc7vx690t-ffg1157c-2 & PCIe Gen3 ×8



$n = 2^{12}, \log q_i = 30, w = 2$

Vivado 2016.3: simulated, placed and routed.

LUT	12.5%
LUTRAM	8.3%
FF	7.7%
BRAM	**14.1%**
DSP	**14.4%**

Ressources		RPM					BCHI
type	available	**total**	NTT	MM	GTW	Others	& WRAP
LUT	432,368	**54,188**	41,964	5,198	5,906	1,120	27,775
LUTRAM	173,992	**14,402**	10,710	2,056	1,550	86	5,425
FF	864,736	**66,444**	50,961	6,755	7,761	967	39,614
BRAM	1,470	**208**	147	0	21	40	153
DSP	3,600	**517**	363	88	66	0	48
IO	600	**0**	0	0	0	0	59
Pcie	3	**0**	0	0	0	0	1

Post-implementation resources utilization

Running frequency

$f_{RPM} = 200$ MHz

How does RPM scale in FV context?

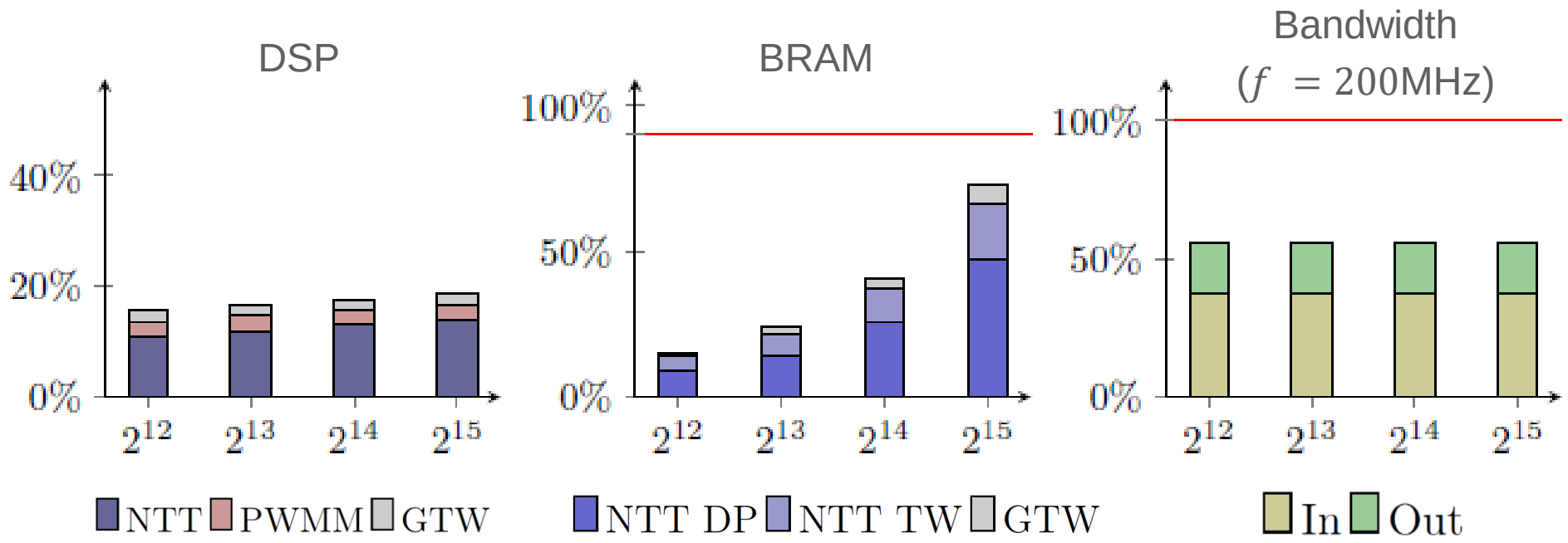
RPM CHARACTERIZATION (2)

PROJECTION: INFLUENCE OF DEGREE n

Impact of the polynomial degree n ($w = 2$ and $\log_2 q_i = 30$):

Xilinx Virtex 7: XC7VX690T-2-FFG1157C

— Resource limitation (FPGA / PCIe Gen3 x8)



Slight increase in DSP utilization.

BRAM is restrictive for $n > 2^{15}$ ([58-65]% for NTT permutations)

Required bandwidth is acheivable

RPM CHARACTERIZATION (3)

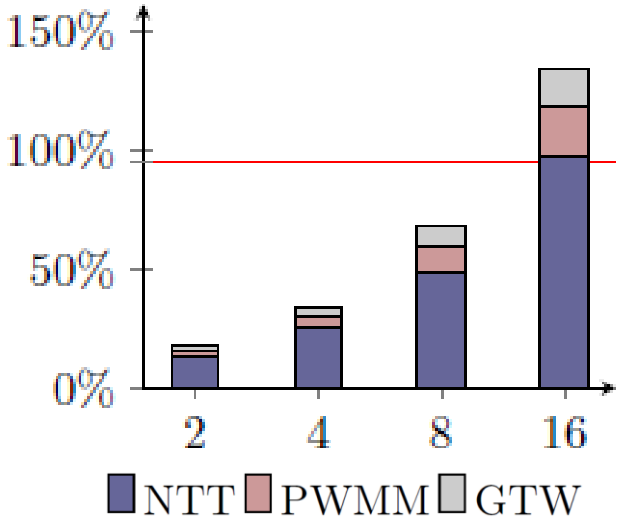
PROJECTION: INFLUENCE OF STREAMING WIDTH w

Impact of the polynomial degree w ($n = 2^{14}$ and $\log_2 q_i = 30$):

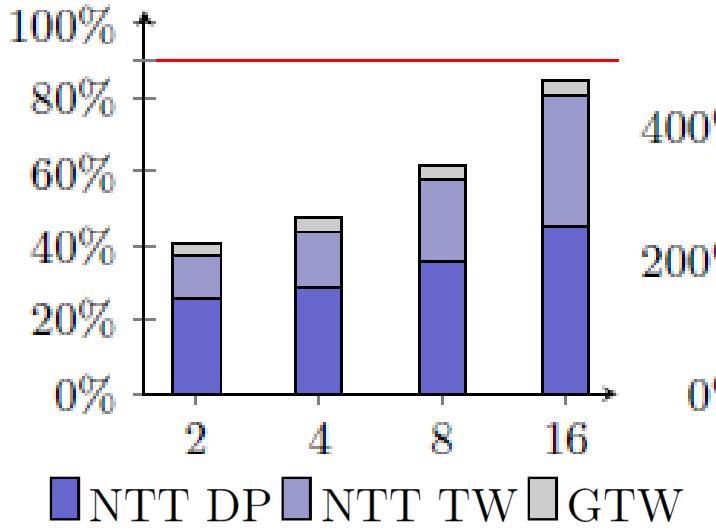
Xilinx Virtex 7: XC7VX690T-2-FFG1157C

— Resource limitation (FPGA / PCIe Gen3 x8)

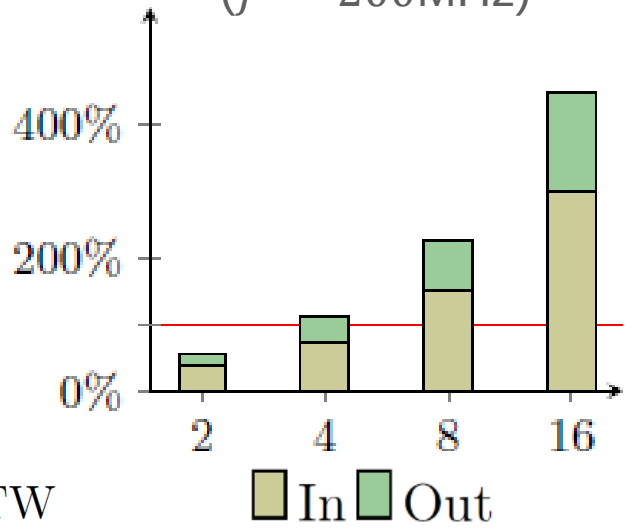
DSP



BRAM



Bandwidth ($f = 200$MHz)



Great increase in DSP utilization.

Increase of BRAM utilization.

Required bandwidth is prohibitive

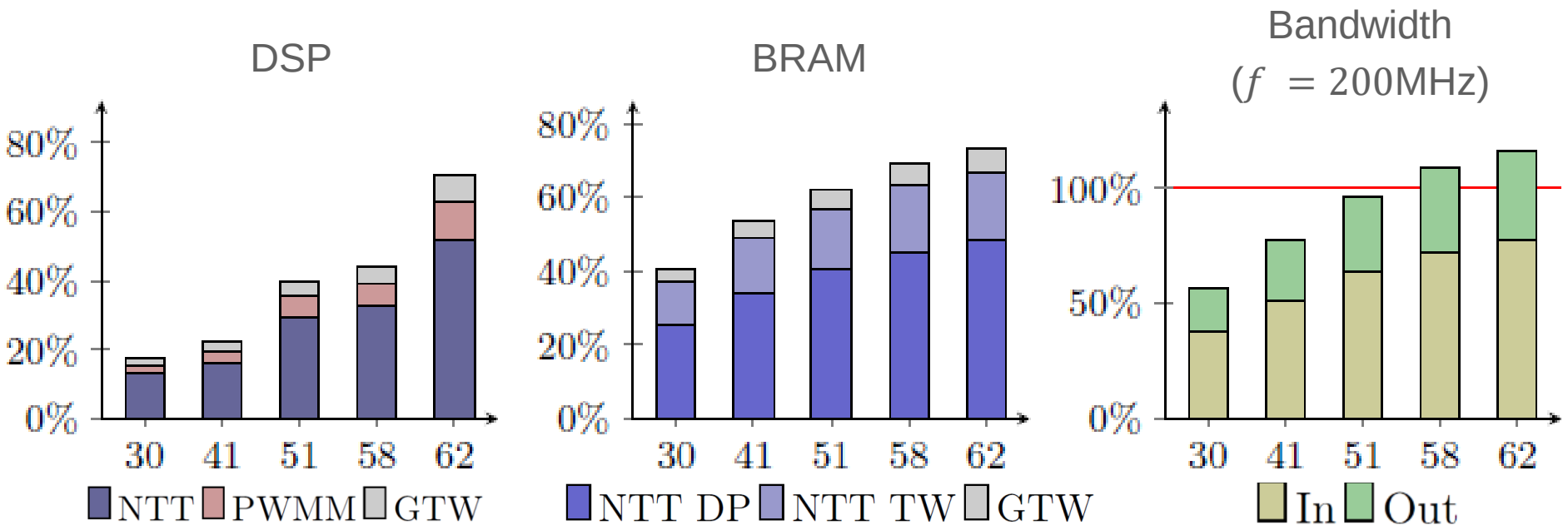
RPM CHARACTERIZATION (4)

PROJECTION: INFLUENCE OF PRIME SIZE $\mathrm{LOG}_2\, q_i$

Impact of the polynomial degree $\log_2 q_i$ ($n = 2^{14}$ and $w = 2$)**:**

Xilinx Virtex 7: XC7VX690T-2-FFG1157C

— Resource limitation (FPGA / PCIe Gen3 x8)



Balanced impact on DSP and BRAM utilization.

Required bandwidth may become restrictive

PERFORMANCE PROJECTIONS: FV-RNS APPLICATION

Performance projection @200MHz:

With respect to timing from [HPS18] ($\lambda > 128$) (su = speedup)

Parameters							RPM	Mul.RPM		Relin.RPM	
L	n	S_q	s	k	k'	w	1/ms	#	ms(**su**)	#	ms(**su**)
1	2^{12}	54		2	3		195.3	15	0.08(×**23.4**)	8	0.04(×**9.5**)
5	2^{13}	108		4	5		97.7	27	0.28(×**22.2**)	32	0.33(×**7.5**)
10	2^{13}	216	30	8	8	2	48.8	48	0.98(×**24.4**)	128	2.62(×**6.8**)
20	2^{14}	432		15	15		24.4	93	3.69(×**27.4**)	450	18.4(×**4.0**)
30	2^{15}	594		20	21		12.2	126	10.1(×**31.3**)	882	65.5(×**4.8**)
20	2^{14}	432	30	15	15	2	24.4	93	3.69(×**27.4**)	450	18.4(×**4**)
						4	48.8		1.84(×**54.8**)		9.22(×**8.1**)
						8	97.7		0.92(×**109.5**)		4.61(×**16.1**)
						16	195.3		0.46(×**219**)		2.30(×**32.3**)
20	2^{14}	432	30	15	15	2	24.4	93	3.69(×**27.4**)	450	18.4(×**4**)
			41	11	11			69	2.70(×**37.3**)	242	9.91(×**7.5**)
			51	9	9			54	2.21(×**45.6**)	162	6.64(×**11.2**)
			58	8	8			48	1.97(×**51.3**)	128	5.24(×**14.2**)
			62	7	8			45	1.84(×**54.8**)	98	4.01(×**18.5**)

Scalable speedup w.r.t. multiplicative depth (FV parameter growth)

Parallelism improves speedup but is costly

Prime size reduces number of operations at reasonable cost

OTHER CONTRIBUTIONS

- **Design of a generator of twiddle factor sets**

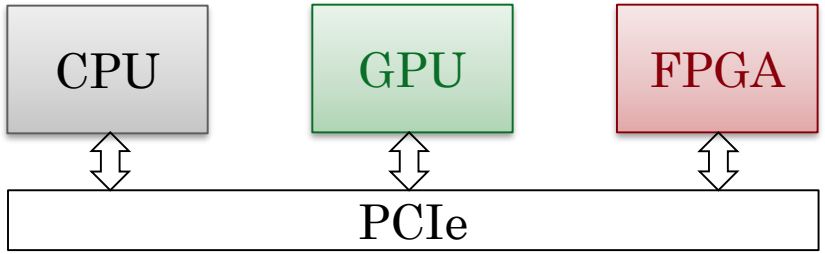
Adapted to multi-field NTT throughput **&** **Intermediate storage independent of n**

- **Prototype of GPU acceleration for RNS specific functions**

Up to 30 times faster than single threaded implementation

Up to 60 times faster than single threaded implementation

- **Exploration of potential acceleration with a hybrid computing system for FV**



One order of magnitude faster for ciphertext multiplication

Communication intensive application

CONCLUSION

Positioning: Choice of accelerating the FV scheme

Problematic: Definition of an hardware acceleration strategy that cope with the wide range of parameters.

Contributions:

Theoretical validation of the RNS/NTT coupled strategy

GPU acceleration of RNS specific functions + Hardware design of NTT-based RPM

Exploration of hybrid computing system for FV acceleration

Expecting a ciphertext multiplication in 61 ms for mult. depth 30

Hardware design of NTT-based RPM → twiddle factor management issues

- Fully-streaming multi-field NTT
  Up to −90 % of memory utilization
- Twiddle factor set generator
  Bandwidth requirement from GB/s to kB/s for twiddle factors

PERSPECTIVES

- **Improving bandwidth for communication intensive system**
  - High bandwidth & low latency interfaces (IBM Capi, Intel UltraPath, …)
  - 3D memory
- **Leveled homomorphic cryptography with the FV scheme**
  - Target application: low-depth and batching intensive applications
  - Batching compatible NTT-based Residue Polynomial Multiplication
- **Hardware acceleration for TFHE**
  - Polynomial multiplications with real coefficients modulo 1
  - Exploration of SPIRAL opportunities for FFT-based convolutions
- **Hardware acceleration for Post-Quantum cryptography**
  - NIST competition: 1/3 lattice-based with Polynomial Multiplication required
  - Explore single-field NTT generation with SPIRAL

Thanks!

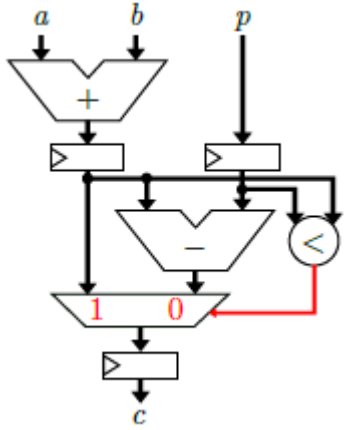
Questions?

Centre de Saclay
Nano-Innov PC 172 - 91191 Gif sur Yvette Cedex

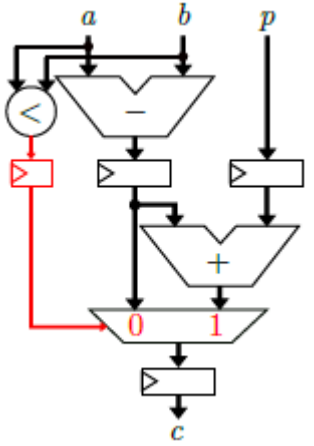
INSTITUT CARNOT CEA LIST

MODULAR ARITHMETIC

- **Modular Addition:**



- **Modular Subtraction:**



- **Modular Multiplication (NFLlib):**