Efficient Software and Hardware Implementations of a QCSP Communication System

Camille Monière[1, 2] Bertrand Le Gal[2] Emmanuel Boutillon[1]

[1]:Lab-STICC, Université de Bretagne Sud, 56100 Lorient, France, Email:
firstname.lastname@univ-ubs.fr

[2]:IMS, Bordeaux-INP, 33400 Talence, France, Email: firstname.lastname@ims-bordeaux.fr

In proceedings of DASIP, Budapest, Hungary
20th of June, 2022

Outline

Context

System Implementation

- Principle
- Inaccuracy mitigations
- Variations

Performances

- Software Implementation
- Hardware Implementation

Conclusion

Internet of Things (IoT)

- Exponential growth during last decades,
- over 50 billions connected devices expected soon,

and yet…

- detection/synchronization metadata can represent more than 50% of the consumed resources.

Spatial Technologies

- Require stability and certainty

and…

- Cyclic-Code Shift Keying (CCSK) is used by *Quasi-Zenith Satellite System*

(Japanese satellite navigation enhancement system)

- Non-Binary Error Correcting Codes are used in BeiDou

(Chinese satellite navigation system)

Internet of Things (IoT)

- Exponential growth during last decades,
- over 50 billions connected devices expected soon,

and yet…

- detection/synchronization metadata can represent more than 50% of the consumed resources.

Spatial Technologies

- Require stability and certainty

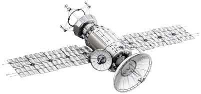
and…

- Cyclic-Code Shift Keying (CCSK) is used by *Quasi-Zenith Satellite System*

(Japanese satellite navigation enhancement system)

- Non-Binary Error Correcting Codes are used in BeiDou

(Chinese satellite navigation system)



Internet of Things (IoT) VS Spatial Technologies

- Exponential growth during last decades,
- over 50 billions connected devices expected soon,

and yet…

- detection/synchronization metadata can represent more than 50% of the consumed resources.

- Require stability and certainty

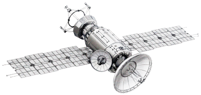
and…

- Cyclic-Code Shift Keying (CCSK) is used by *Quasi-Zenith Satellite System*

(Japanese satellite navigation enhancement system)

- Non-Binary Error Correcting Codes are used in BeiDou

(Chinese satellite navigation system)



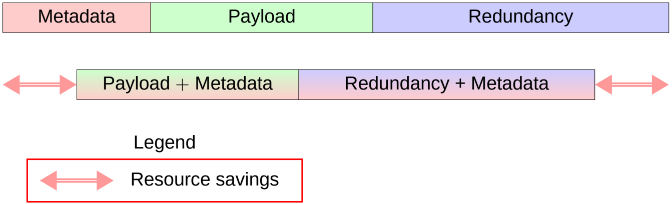
Quasi-Cyclic Small Packet (QCSP) Project

`https://qcsp.univ-ubs.fr/`

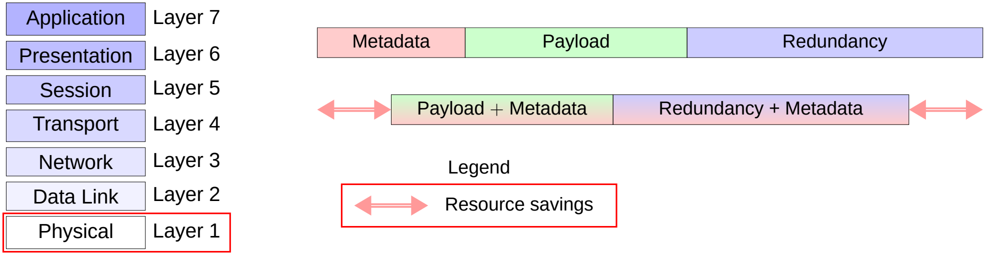
- Project funded by the ANR, grant ANR-19-CE25-0013-01.
- Thesis directed by E. BOUTILLON, supervised by B. LE GAL.

"The aim of the QCSP project is to contribute to the evolution of IoT networks by defining, implementing and testing a new coded modulation scheme dedicated to IoT networks."

Application	Layer 7
Presentation	Layer 6
Session	Layer 5
Transport	Layer 4
Network	Layer 3
Data Link	Layer 2
Physical	Layer 1



Metadata	Payload	Redundancy
Allows detection/synchronization of the frame	Contains the information to transmit	Provides error-tolerance to the payload

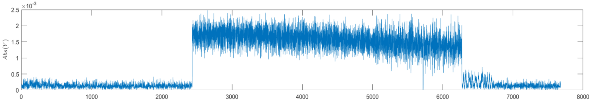


Metadata	Payload	Redundancy
Is not actually used to communicate informations so **related resources are wasted from this point of view**	Contains the information to transmit	Provides error-tolerance to the payload

[1]: Y. Polyanskiy. "Asynchronous Communication". In: *IEEE Trans. Inform. Theory* 59.3 (2013), pp. 1256–1270

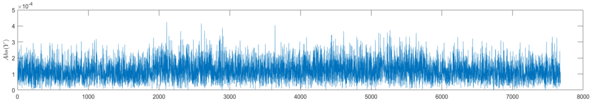
A model exists, how can it reach real-time ?

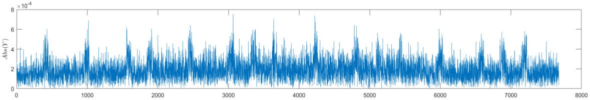
throughput > 4 MChip/s | Low-power transmitter | completely blind transmission



→ High SNR

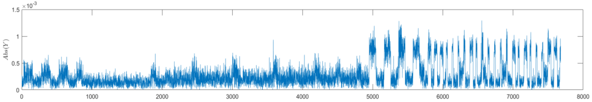
Low SNR (-10 dB) ←





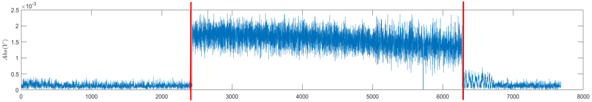
→ Low SNR and sparse interferences

Low SNR and strong interferences ←



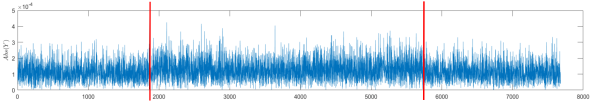
A model exists, how can it reach real-time ?

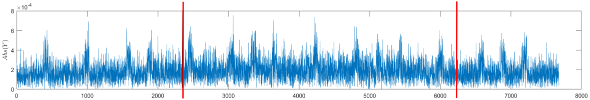
throughput > 4 MChip/s | Low-power transmitter | completely blind transmission



→ High SNR

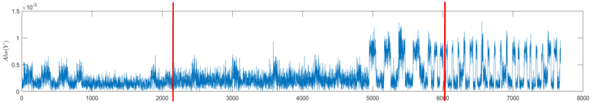
Low SNR (-10 dB) ←



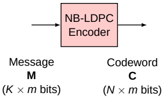


→ Low SNR and sparse interferences

Low SNR and strong interferences ←



QCSP Communication Chain



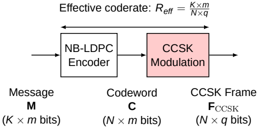
NB-LDPC Encoder

Message
M
($K \times m$ bits)

Codeword
C
($N \times m$ bits)

Message **M** "1 2 1" $\Rightarrow$ Codeword **C** "1 2 1 3 2 3"

QCSP Communication Chain


Effective coderate: $R_{eff} = \frac{K\times m}{N\times q}$

NB-LDPC Encoder

CCSK Modulation

Message
$\mathbf{M}$
($K \times m$ bits)

Codeword
$\mathbf{C}$
($N \times m$ bits)

CCSK Frame
$\mathbf{F}_{\text{CCSK}}$
($N \times q$ bits)

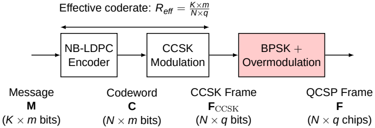
Codeword **C**
"1 2 1 3 2 3"

$\Rightarrow$

CCSK Frame $\mathbf{F}_{CCSK}$
"🟥🟦🟦🟦 🟦🟥🟦🟦 🟥🟦🟦🟦 🟦🟦🟥🟦 🟦🟥🟦🟦 🟦🟦🟥🟦"

🟥: 1 — 🟦: 0
CCSK Sequence: 🟥🟦🟦🟦

QCSP Communication Chain



Effective coderate: $R_{eff} = \frac{K \times m}{N \times q}$

NB-LDPC Encoder

CCSK Modulation

BPSK + Overmodulation

Message **M** ($K \times m$ bits)

Codeword **C** ($N \times m$ bits)

CCSK Frame $\mathbf{F}_{\text{CCSK}}$ ($N \times q$ bits)

QCSP Frame **F** ($N \times q$ chips)

CCSK Frame $\mathbf{F}_{CCSK}$

"🟥🟦🟦🟦 🟦🟥🟦🟦 🟥🟦🟦🟦 🟦🟦🟥🟦 🟦🟥🟦🟦 🟦🟦🟥🟦"

🟥 : 1 — 🟦 : 0

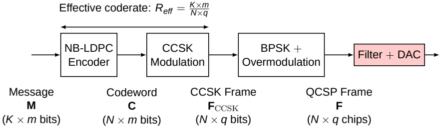
$\Rightarrow$

QCSP Frame **F**

"🟦🟥🟥🟥 🟦🟥🟦🟦 🟥🟦🟦🟦 🟥🟥🟦🟥 🟦🟥🟦🟦 🟥🟥🟦🟥"

🟥 : +1 — 🟦 : -1

QCSP Communication Chain



Effective coderate: $R_{eff} = \frac{K \times m}{N \times q}$

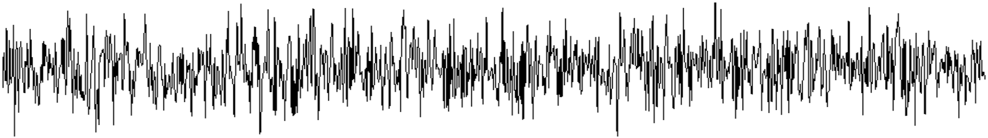
NB-LDPC Encoder → CCSK Modulation → BPSK + Overmodulation → Filter + DAC

Message
M
($K \times m$ bits)

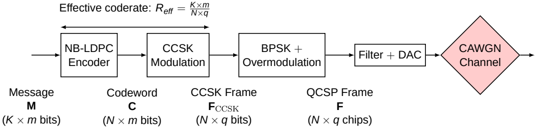
Codeword
C
($N \times m$ bits)

CCSK Frame
$\mathbf{F}_{\mathrm{CCSK}}$
($N \times q$ bits)

QCSP Frame
F
($N \times q$ chips)



QCSP Communication Chain



Effective coderate: $R_{eff} = \frac{K \times m}{N \times q}$

NB-LDPC Encoder

CCSK Modulation

BPSK + Overmodulation

Filter + DAC

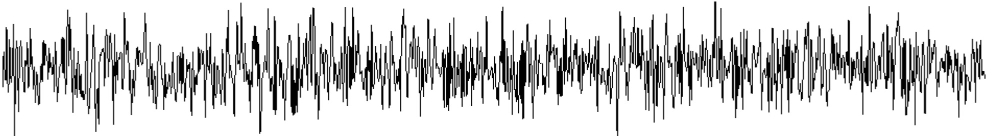
CAWGN Channel

Message
M
($K \times m$ bits)

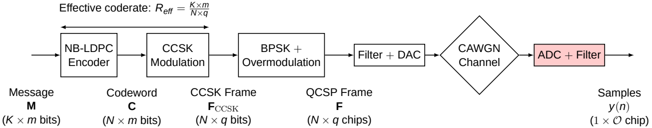
Codeword
C
($N \times m$ bits)

CCSK Frame
$\mathbf{F}_{\text{CCSK}}$
($N \times q$ bits)

QCSP Frame
F
($N \times q$ chips)



QCSP Communication Chain



Effective coderate: $R_{eff} = \frac{K \times m}{N \times q}$

NB-LDPC Encoder → CCSK Modulation → BPSK + Overmodulation → Filter + DAC → CAWGN Channel → ADC + Filter →

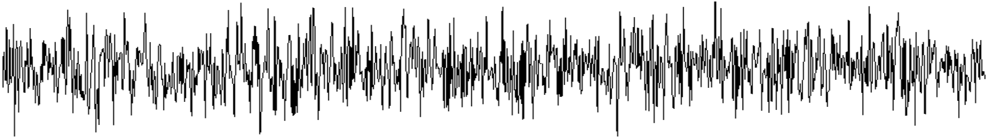
Message
M
($K \times m$ bits)

Codeword
C
($N \times m$ bits)

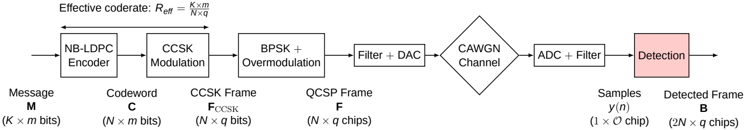
CCSK Frame
$\mathbf{F}_{\text{CCSK}}$
($N \times q$ bits)

QCSP Frame
F
($N \times q$ chips)

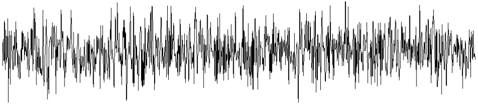
Samples
$y(n)$
($1 \times \mathcal{O}$ chip)



QCSP Communication Chain



Samples ($\mathcal{O}\times$)



⇒

Detected Frame **B**

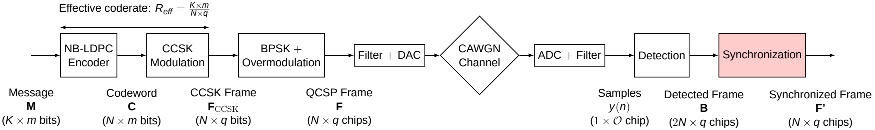
"ZZZZZZZZ

🟦🟥🟥🟦 🟦🟥?🟦 🟥??🟦 🟥🟥🟦🟥 🟦🟥?🟦 🟦🟥🟦🟥

ZZZZZZZZZZZ"

🟥: +1 — 🟦: -1 — Z: Noise — ?: Noisy Chip

QCSP Communication Chain


Effective coderate: $R_{eff} = \frac{K \times m}{N \times q}$

NB-LDPC Encoder

CCSK Modulation

BPSK + Overmodulation

Filter + DAC

CAWGN Channel

ADC + Filter

Detection

Synchronization

Message **M** ($K \times m$ bits)

Codeword **C** ($N \times m$ bits)

CCSK Frame $\mathbf{F}_{\text{CCSK}}$ ($N \times q$ bits)

QCSP Frame **F** ($N \times q$ chips)

Samples $y(n)$ ($1 \times \mathcal{O}$ chip)

Detected Frame **B** ($2N \times q$ chips)

Synchronized Frame **F'** ($N \times q$ chips)

Detected Frames **B**

"ZZZZZZZZ 🟦🟥🟥🟦 🟦🟥?🟦 🟥??🟦 🟥🟥🟦🟥 🟦🟥?🟦 🟦🟥🟦🟥 ZZZZZZZZZZZ"

🟥: +1 — 🟦: -1 — Z: Noise — ?: Noisy Chip

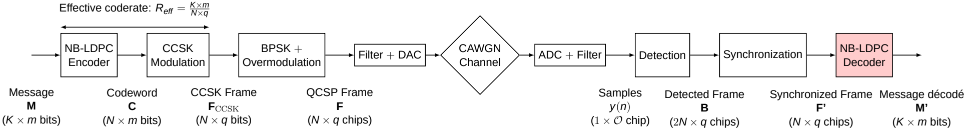
⇒

Synchronized Frame **F'**

"🟦🟥🟥🟦 🟦🟥?🟦 🟥??🟦 🟥🟥🟦🟥 🟦🟥?🟦 🟦🟥🟦🟥"

🟥: +1 — 🟦: -1 — ?: Noisy Chip

QCSP Communication Chain



Synchronized Frame **F'**

”🟦🟥🟥🟦 🟦🟥?🟦 🟥??🟦 🟥🟥🟦🟥 🟦🟥?🟦 🟦🟥🟦🟥”

$\Rightarrow$

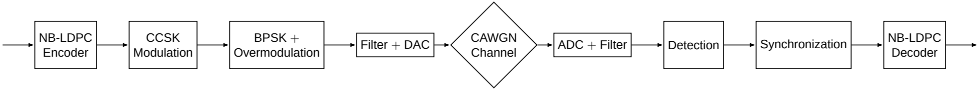
Decoded Message **M'**

”1 2 1”

🟥: +1 — 🟦: -1 — ?: Noisy Chip

Sidenote: QCSP frame demodulation product is directly usable by the NB-LDPC decoder.

Need efficient implementations.


NB-LDPC Encoder → CCSK Modulation → BPSK + Overmodulation → Filter + DAC → CAWGN Channel → ADC + Filter → Detection → Synchronization → NB-LDPC Decoder

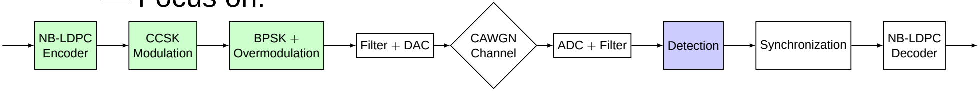
- The legacy CCSK demodulation method is not efficient for CCSK based detection,
- The new Time Sliding window method is promising.

[2]: O. Abassi et al. “Non-Binary Low-Density Parity-Check Coded Cyclic Code-Shift Keying”. In: *proceedings of WCNC*. IEEE, 2013

[3]: C. Monière et al. “Time Sliding Window for the Detection of CCSK Frames”. In: *proceedings of SiPS*. IEEE, 2021

Need efficient implementations.

— Focus on:



Transmission

Detection

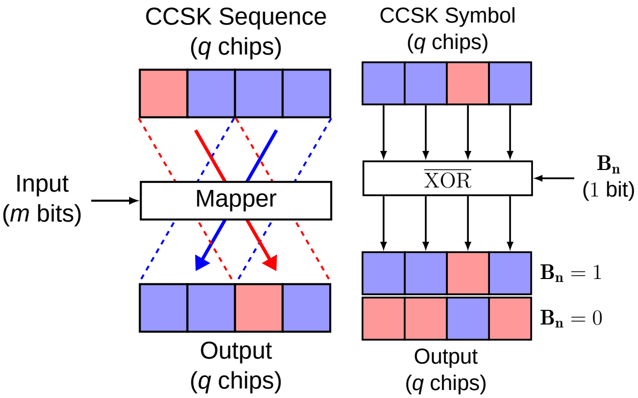
Transmission must be simple enough for low-end sensor nodes, and detection must satisfy standard defined throughput

[4]: “IEEE Std 802.15.4-2020, IEEE Standard for Low-Rate Wireless Networks”. In: (May 2020), p. 799

Transmitter — Low cost

CCSK Modulation

Overmodulation



SIMD and memory swap on CPU

BRAM or LUTRAM direct read, or shift register on FPGA.

Transmitter — *"Bottleneck"*

NB-LDPC — GF(q), $q = 2^m$

Nested loops (redundancy calculus)

FIR Filter — 21 coefficients

Cumulative sum of products through time

just some few CPU instructions,

— or —

pipelines and duplicated operators on FPGA.

Transmitter — *"Bottleneck"*

NB-LDPC — GF(q), $q = 2^m$

Nested loops (redundancy calculus)

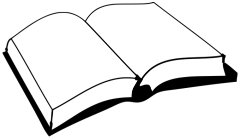
FIR Filter — 21 coefficients

Cumulative sum of products through time

just some few CPU instructions,

— or —

pipelines and duplicated operators on FPGA.



Both processes are already explored and optimized.

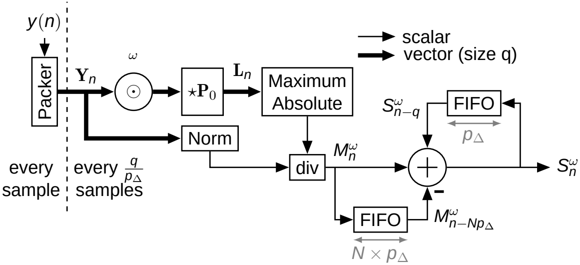
Detection

The harder part

Principle :

Compare a detection score against a threshold [5].

Score $\approx$ cumulative sum of the maxima of the N last correlation with the CCSK sequence $\mathbf{P}_0$, thus representing the *likelihood* of the last frame-long buffer to be a frame.



Time granularity p_Δ: score values calculated every q chips.

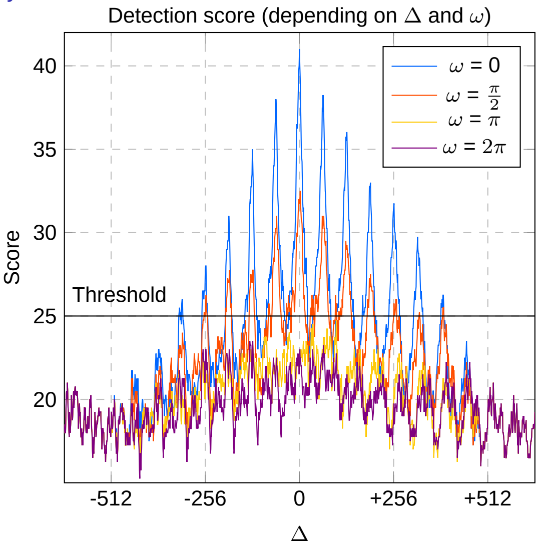
Detection

Time/frequency errors impact on reliability

Δ	—	time shift (chips)
ω	—	frequency shift (radians/symbol)

A parasitic rotation results from frequency errors (clock inaccuracies, doppler effect)

Note: rotation for q chips (size of the correlation and of a symbol) is considered



Detection

Time/frequency errors impact on reliability

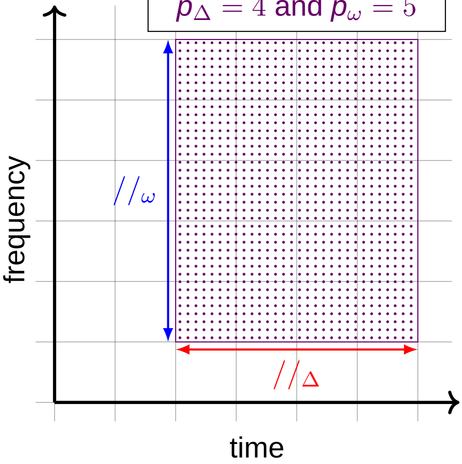
Δ	—	time shift (chips)
ω	—	frequency shift (radians/symbol)

Time window: $[0, p_\Delta - 1]$,
with $p_\Delta \in [1, q-1]$

Rotation window: $[-\pi, \pi]$ divided in p_ω equal part,
with $p_\omega = 1, 2, \ldots, 8$

reliability/performance trade off possible, by adjusting p_Δ and p_ω values.

Detection window with : $p_\Delta = 4$ and $p_\omega = 5$



Detection

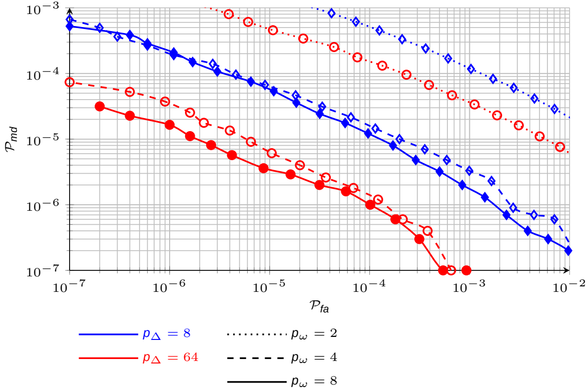
Time/frequency errors impact on reliability

Δ	—	time shift (chips)
ω	—	frequency shift (radians/symbol)

Time window: $[0, p_\Delta - 1]$,
with $p_\Delta \in [1, q - 1]$

Rotation window: $[-\pi, \pi]$ divided in p_ω equal part,
with $p_\omega = 1, 2, \ldots, 8$

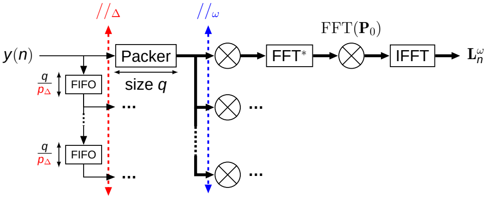
reliability/performance trade off possible, by adjusting p_Δ and p_ω values.



Correlation Methods

Legacy

Legacy method, inherited from the literature[2].



Using FFT/IFFT

Pros:

- flexibility,
- independent processing along $//_{\triangle}$ and $//_{\omega}$,
- FFTs are already optimized,
- FIFO memory can be shared or distributed.

Cons:

- batch processing,
- not well suited for dataflow tasks,
- consumption.

Correlation Methods

New

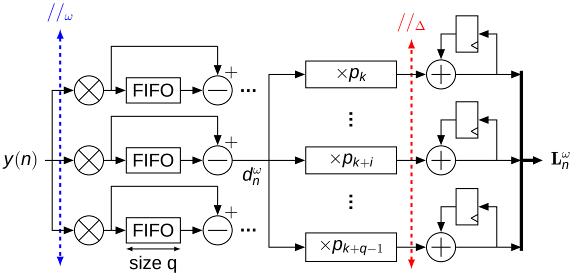
Recently introduced method [3].

Pros:

- independent processing along $//_\omega$,
- quite lighter, complexity speaking (for equivalent p_Δ, p_ω),
- dataflow by design.

Cons:

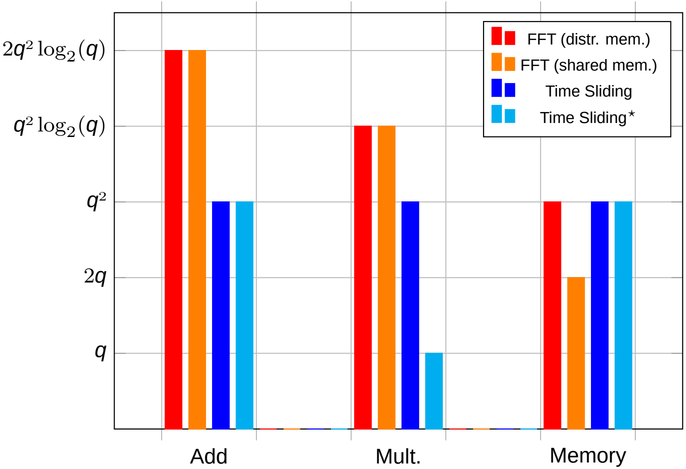
- requires p_Δ set at q,
- memory sharing is harder.



Using Time Sliding Windows

Correlation Methods

Algorithmic Complexities comparison



when $p_\triangle = q$ (which also results in better reliability), TS has a clear advantage.

Performances

Settings

From now on:

$$\begin{aligned} K &= 20 \\ m &= 6 \\ q &= 64 \\ N &= 60 \\ \mathcal{O} &= 1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} R_{eff} &= \frac{1}{32} \\ Payload &= 120 \text{ bits} \\ Symbol &= 64 \text{ chips} \\ Frame &= 3840 \text{ chips} \end{aligned}$$

Performances

Settings

From now on:

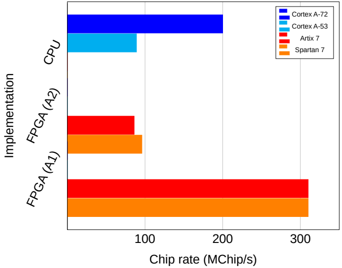
$$\begin{aligned} K &= 20 \\ m &= 6 \\ q &= 64 \\ N &= 60 \\ \mathcal{O} &= 1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} R_{eff} &= \frac{1}{32} \\ Payload &= 120 \text{ bits} \\ Symbol &= 64 \text{ chips} \\ Frame &= 3840 \text{ chips} \end{aligned}$$

Note: Oversampling is never set to 1 in real systems, rather to 8. However, each sampling frequency is process independently

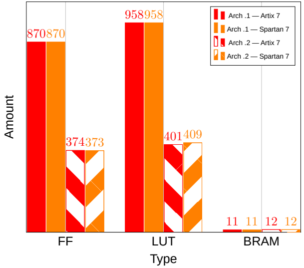
Transmitter

Implementation results

Throughput for three different implementation on different targets.



Resource consumption for FPGA implementations.



Software using C/C++ on ARM CPUs:

- A-53 (1.4 GHz, 32-bit, RAM 1 GB),
- A-72 (1.5 GHz, 64-bit, RAM 4 GB).

Hardware using C/C++ for HLS on Xilinx targets:

- Artix 7,
- Spartan 7,

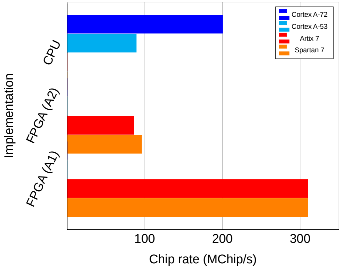
clocked at 100 MHz, and using two architectures (using #pragma and directives):

- Arch. 1 — throughput optimized,
- Arch. 2 — resource optimized.

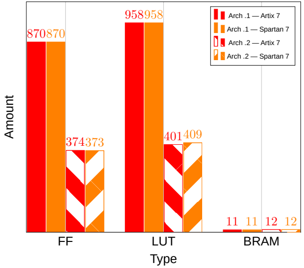
Transmitter

Implementation results

Throughput for three different implementation on different targets.



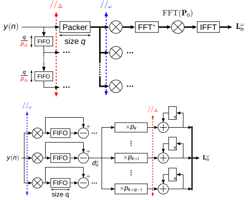
Resource consumption for FPGA implementations.



Way above targeted results, emphasizing the low complexity of the transmitter. Plus, who can do more can do less.

Detector

Software Implementation



Benchmarked on a Linux server, equipped with an Intel Xeon-6148 Gold
dual socket, 20 cores/socket, 256 GB of RAM, clocked to 3.5 GHz in average.

FFT Method

- FFT implemented thanks to FFTW[6]

FFTW

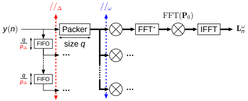
- Implemented monothreaded and multithreaded along $//_\Delta$ using OpenMP

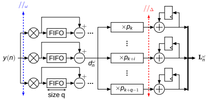
Time Sliding Method

- Written from scratch in C++11
- Implemented to make use of GCC vectorization feature (SIMD, loop unrolling, …)

Detector

Results



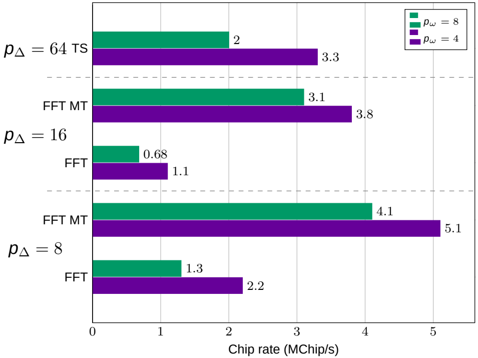


Slowest ⇒ FFT with $p_\Delta = 16$ and $p_\omega = 8$

Fastest ⇒ FFT MT with $p_\Delta = 8$ and $p_\omega = 4$
→ but the lowest detection performances

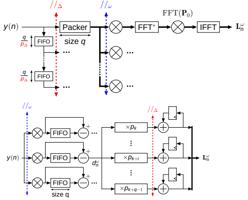
— However —

Time Sliding ($p_\omega = 4$) achieve better throughput than FFT MT ($p_\Delta = 16$, $p_\omega = 8$), for comparable detection performances and $16\times$ less CPU power



Detector

Hardware Implementation



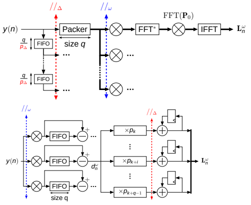
Results given by the Xilinx HLS tool (after place and route stage) for a Kintex 7 clocked at 100MHz

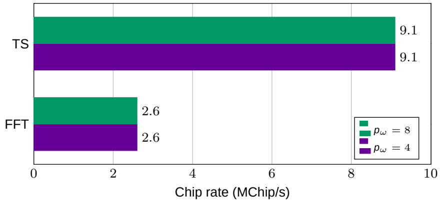
- FFT data are synthetic, extrapolated from one optimized core which processes 16-bits fixed-point data
- Time Sliding data correspond to a full system processing floating-point data

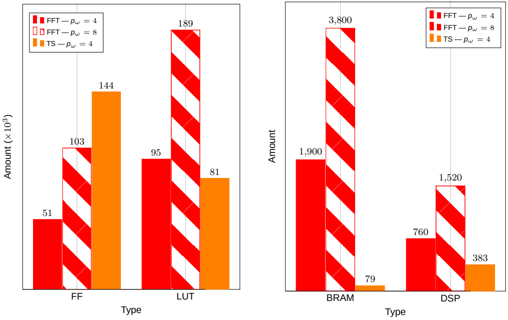
Both written in C/C++ for HLS, optimized for throughput at all costs (to explore the limits)

Detector

Results







Fastest ⇒ Time Sliding with the highest detection performances ($p_\omega = 4$)

Time sliding with $p_\omega = 8$ cannot be implemented (neither FFT because of DSP, lowering their number would affect throughput ...)

Suited for Wireless Sensor Networks

The QCSP transmitter is low-cost and low-complexity.

An implementation of the receiver have achieved throughput allowing real-time frame detection, for an acceptable complexity for a high end base station

The new time sliding method is undoubtedly the best for dataflow processing

Suited for Wireless Sensor Networks

But still work to do

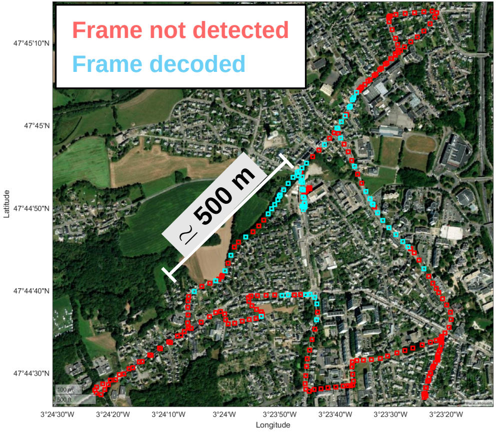
A fixed-point model of the receiver has been defined and is at last stage of implementation on FPGA

A way to process small batch of data using the time sliding method has been imagined, and may reduce memory usage

The remains of the communication system must be optimized

Multi-user scenarios are currently explored

Last achievement



- GPS location of a moving device sent using QCSP modulation
- Software detector running on the roof of a building
- Achieved a range of 500 m with low-quality antennas
- For a consumption lower than 1 μJ per information bit

Thank you for your attention,
have you any question ?

Bibliography

[1] Y. Polyanskiy. “Asynchronous Communication”. In: *IEEE Trans. Inform. Theory* 59.3 (2013), pp. 1256–1270.

[2] O. Abassi et al. “Non-Binary Low-Density Parity-Check Coded Cyclic Code-Shift Keying”. In: *proceedings of WCNC*. IEEE, 2013.

[3] C. Monière et al. “Time Sliding Window for the Detection of CCSK Frames”. In: *proceedings of SiPS*. IEEE, 2021.

[4] “IEEE Std 802.15.4-2020, IEEE Standard for Low-Rate Wireless Networks”. In: (May 2020), p. 799.

[5] K. Saied. “Quasi-Cyclic Short Packet (QCSP) Transmission for IoT”. Theses. Université Bretagne Sud, Mar. 2022.

[6] M. Frigo and S.G. Johnson. “The Design and Implementation of FFTW3”. In: *Proceedings of the IEEE* 93.2 (Feb. 2005), pp. 216–231. issn: 1558-2256. doi: `10.1109/JPROC.2004.840301`.