Paris-Saclay University

A2 : Embedded Electronic Systems

Heterogeneous architecture for Big Data algorithms
Final presentation

Presented by:
DOU Yuhan
FORCIOLI Quentin
GHAOUI Mohamed Anis
TERRACHER Audrey

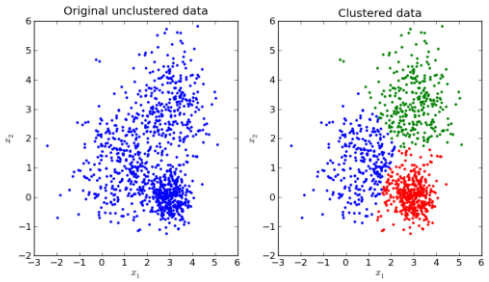
2020 February 03

TEAM

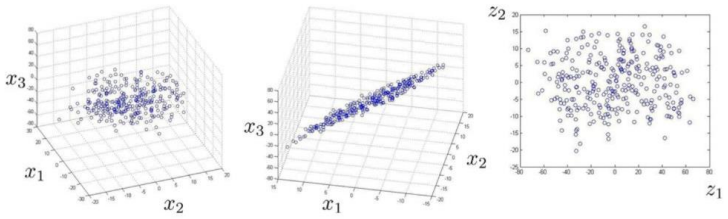
- **Software:**
  - DOU Yuhan
  - TERRACHER Audrey
- **Hardware:**
  - FORCIOLI QUENTIN
- **HLS and lead :**
  - GHAOUI Mohamed Anis

Objective : Complete a full development cycle

Our approach and algorithms chosen



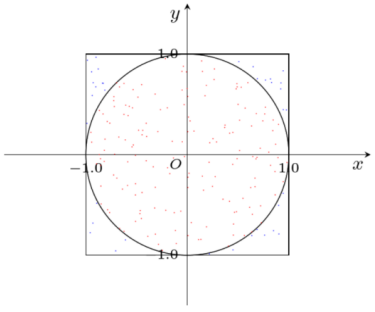
K-Means



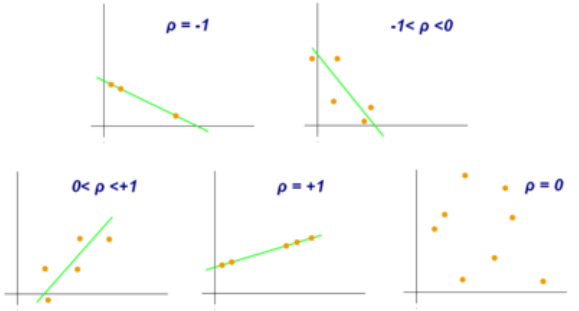
PCA/SVD

A1	A2		B1	B2	B3	A1*B1 + A2*B4	A1*B2 + A2*B5	A1*B3 + A2*B6
A3	A4	×	B4	B5	B6	A3*B1 + A4*B4	A3*B2 + A4*B5	A3*B3 + A4*B6
A5	A6					A5*B1 + A6*B4	A5*B2 + A6*B5	A5*B3 + A6*B6
A7	A8					A7*B1 + A8*B4	A7*B2 + A8*B5	A7*B3 + A8*B6

BMM



Pi Estimator



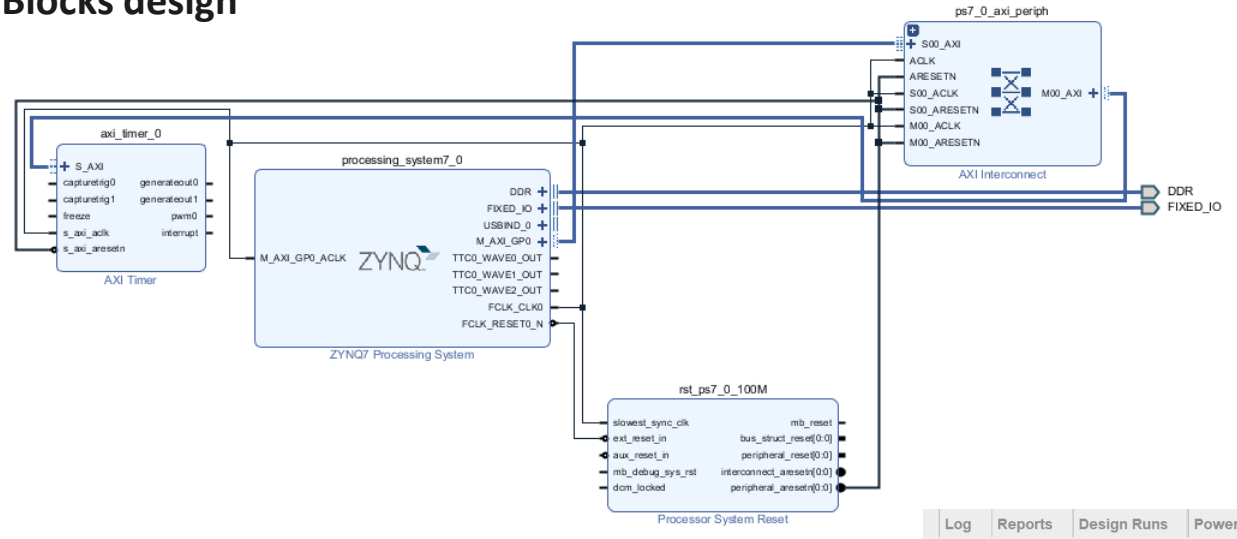
Pearson correlation

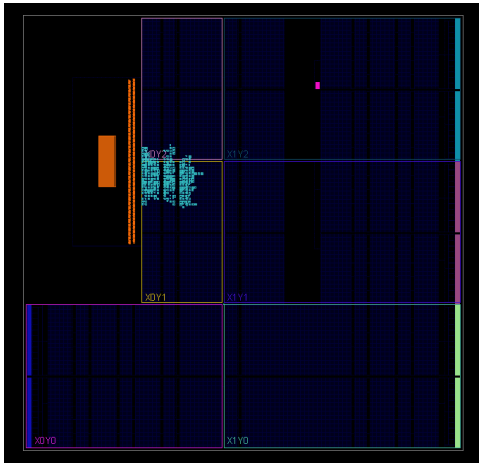
Software – PC Implementation

BMM	M	P	N	blocksize	temps (us)
	40	25	30	3	73
	250	470	316	3	97875
	250	470	316	7	70285
	1000	500	200	15	172542
Pi Estimator	**nb d'itérations**				
	1000				77
	10000				467
	100000				4773
	1000000				44501
K-Means	**K (clusters)**	**D (dimension)**	**N (nb données)**	**Itération**	
	3	4	150	20	9577
	20	2	3000	20	30049
	16	32	1024	20	92520
Pearson	**row**		**col**		
	10		2		62
PCA	**ligne**		**colonne**		
	10		2		72
SVD	**ligne**		**colonne**		
	13		7		6
	30		35		44
	100		200		978

Software – ARM Implementation

Blocks design





Report

Tcl Console | Log | Reports | Design Runs | Power | Timing

Clock Summary

- General Information
- Timer Settings
- Design Timing Summary
- Clock Summary (1)

Name	Waveform	Period (ns)	Frequency (MHz)
clk_fpga_0	{0.000 5.000}	10.000	100.000

Log | Reports | Design Runs | Power | Timing | Utilization

Hierarchy

Name	Slice LUTs (53200)	Slice Registers (106400)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	Block RAM Tile (140)	PHY_CONTROL (4)	BUFIO (16)
sign_algorithms_ARM_wrapper	1.25%	0.66%	1.92%	1.14%	0.34%	0.66%	100.00%	3.13%
design_algorithms_ARM_i (des	1.25%	0.66%	1.92%	1.14%	0.34%	0.66%	0.00%	3.13%
axi_timer_0 (design_algorith	0.55%	0.23%	0.77%	0.55%	0.00%	0.23%	0.00%	0.00%
processing_system7_0 (des	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	3.13%
ps7_0_axi_periph (design_a	0.66%	0.40%	1.16%	0.55%	0.34%	0.40%	0.00%	0.00%
rst_ps7_0_100M (design_al	0.03%	0.03%	0.09%	0.03%	<0.01%	0.03%	0.00%	0.00%

Software – ARM Implementation

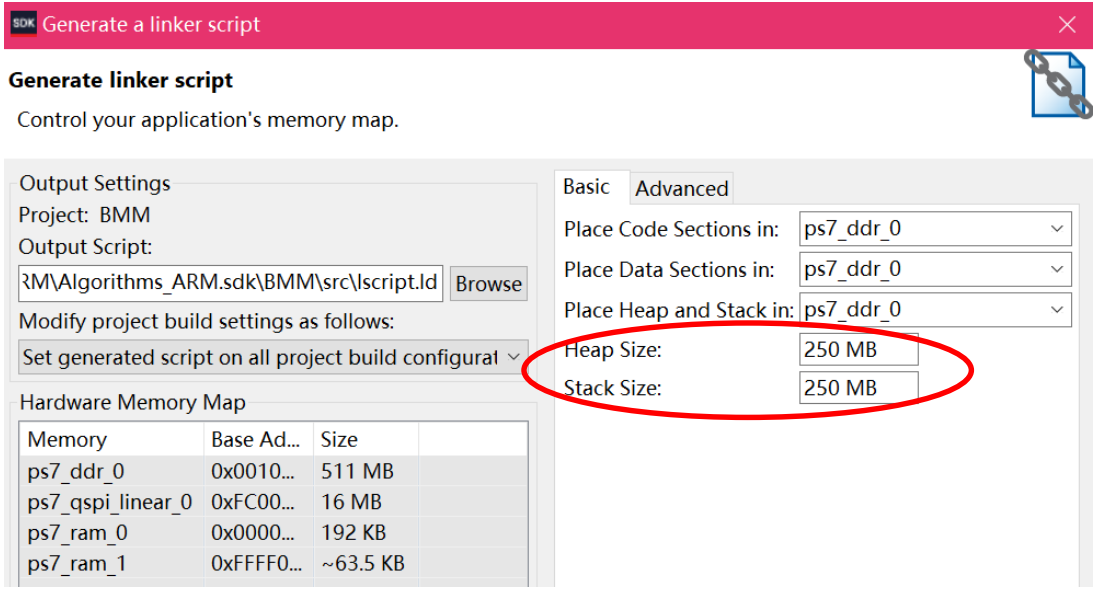
Design on Vivado SDK

1/ Tools Chain : Xilinx ARM v7 GNU Toolchain

Generator : GNU make

2/ Modification of heap and stack sizes :

1 KB (default) → 250 MB

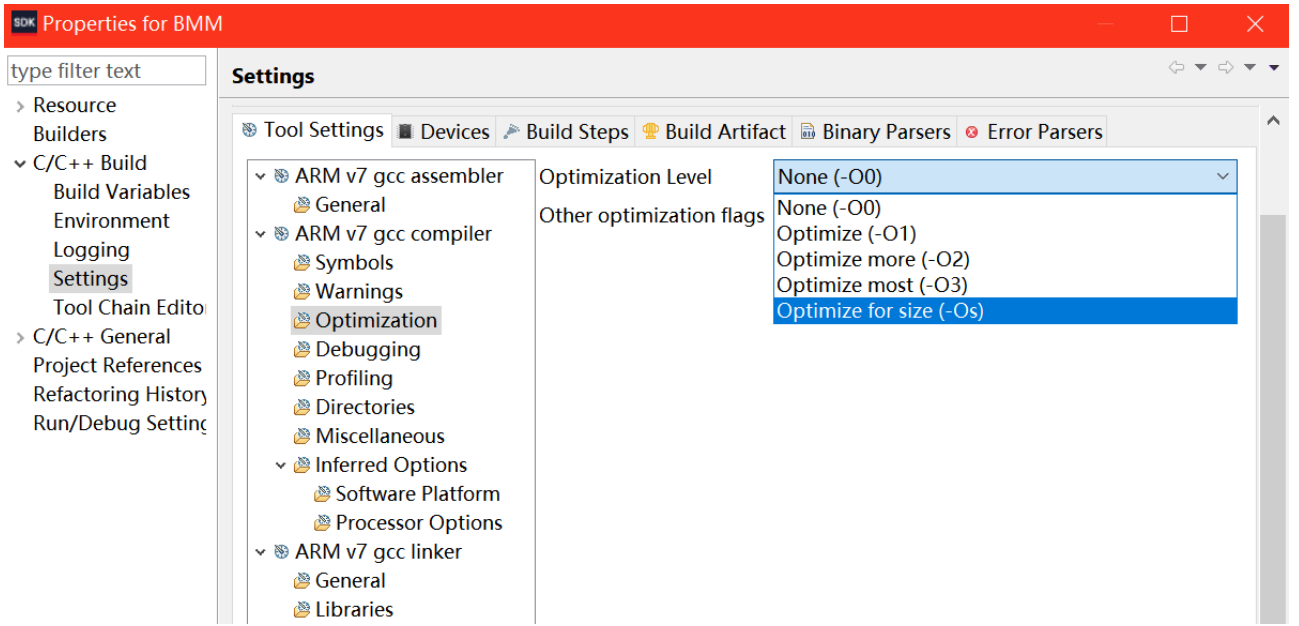


Software – ARM Implementation

Bloc-Matrix-Multipication	**M**	**P**	**N**	**blocsize**	nb de cycles	**temps (µs)**
	40	25	30	3	1694524	5088. 66066
	250	470	316	3	2045606949	6142963. 81081
	250	470	316	7	1740020211	5225285. 91892
	1000	500	200	15	4428594020	13299081. 14114
PiEstimator	**nb d'itération**				nb de cycles	**temps (µs)**
	1000				69204	207. 81982
	10000				674408	2025. 24925
	100000				6867433	20622. 92192
	1000000				68545278	205841. 67586
K-means	**K (clusters)**	**D (dimension)**	**N (nb données)**	**Itération**	nb de cycles	**temps (µs)**
	3	4	150	20	3729634	11200. 10210
	20	2	3000	20	265058761	795972. 25526
	16	32	1024	20	860685779	2584641. 97898
Pearson	**row (nb données)**		**col (dimension)**		nb de cycles	**temps (µs)**
	10		2		20869	62. 66976
	50		4		244682	734. 78078
	200		10		3627683	10893. 94294
PCA	**ligne**		**colonne**		nb de cycles	**temps (µs)**
	10		2		5680	17. 05706
	50		4		52591	157. 93093
	200		10		2992348	8986. 03003
SVD	**ligne**		**colonne**		nb de cycles	**temps (µs)**
	13		7		215159	646. 12312
	30		35		6042542	18145. 77177
	100		200		666164879	2000495. 13213

Software – ARM Implementation

Software optimization



Three main aspects :

1/ branches delays

2/ use of cache

3/ data dependencies

Optimization options for GNU :

- O0 : no optimization

- O1 : optimization on branches

- O2 : optimization at the registers and instruction level

- O3 : optimization at the highest level (eSIMD vectorization , “inline”...)

- Os : optimization on code size

Software – ARM Implementation

Code size (segmentation “text”)

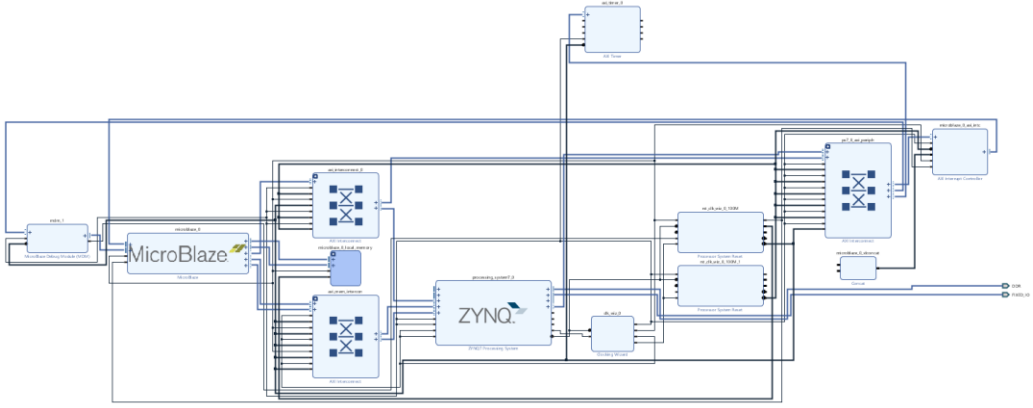
	ordre	optimisation	text
Bloc-Matrix-Multiplication	ijk	-O0	**58716**
	ikj	-O0	**58716**
	ikj	-O1	**58412**
	ikj	-O2	**58328**
	ikj	-O3	**59352**
	ikj	-Os	**58260**
		optimisation	text
K-means		-O0	**63104**
		-O1	**58792**
		-O2	**58764**
		-O3	**59324**
		-Os	**58628**
		optimisation	text
Pearson		-O0	**59244**
		-O1	**58548**
		-O2	**58560**
		-O3	**61400**
		-Os	**58436**

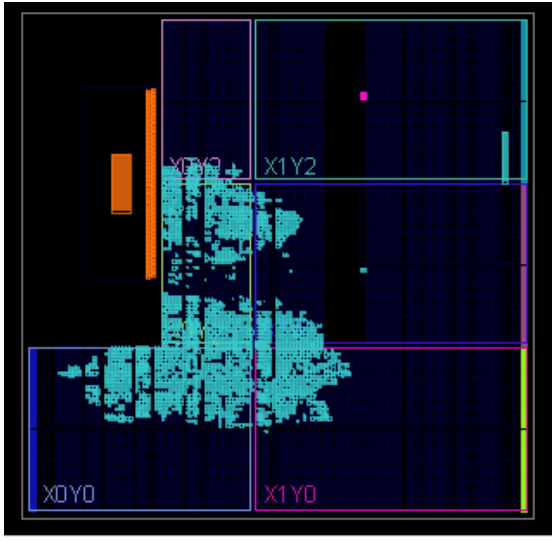
Execution times

M	N	P	blocsize	cycles	temps (µs)
1000	500	200	15	**4428594020**	13299081. 14114
1000	500	200	15	**3822785564**	**11479836.52853**
1000	500	200	15	**531048961**	**1594741.62462**
1000	500	200	15	**564442618**	**1695022.87688**
1000	500	200	15	**545292506**	**1637515.03303**
1000	500	200	15	**809730905**	**2431624.33934**
K	D	N	Itération	cycles	temps (µs)
16	32	1024	20	**860685779**	**2584641.97898**
16	32	1024	20	**123736936**	**371582.39039**
16	32	1024	20	**138956312**	**417286.22222**
16	32	1024	20	**139265209**	**418213.84084**
16	32	1024	20	**149756615**	**449719.56456**
row (nb données)		col (dimension)		cycles	temps (µs)
200		10		**3627683**	**10893.94294**
200		10		**695048**	**2087.23123**
200		10		**688131**	**2066.45946**
200		10		**617540**	**1854.47447**
200		10		**805517**	**2418.96997**

Software – Microblaze Implementation

Blocks design





Report

Log | Reports | Design Runs | Power | DRC | Methodology | **Timing** | Utilization

Clock Summary

Name	Waveform	Period (ns)	Frequency (MHz)
clk_fpga_0	{0.000 5.000}	10.000	100.000
design_mb_i/clk_wiz_0/inst/clk_in1	{0.000 5.000}	10.000	100.000
design_mb_i/mdm_1/U0/Use_E2.I	{0.000 16.667}	33.333	30.000
design_mb_i/mdm_1/U0/Use_E2.I	{0.000 16.667}	33.333	30.000

Log | Reports | Design Runs | Power | DRC | Methodology | Timing | **Utilization**

Hierarchy

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	Block RAM Tile (140)	DSPs (220)
design_mb_wrapper	5390	6314	35	2264	4685	705	6314	6
design_mb_i (design_mb)	5390	6314	35	2264	4685	705	6	3
axi_interconnect_0 (desig	86	123	0	44	86	0	0	0
axi_mem_intercon (desig	2288	3520	0	1109	1939	349	0	0

Software – Microblaze Implementation

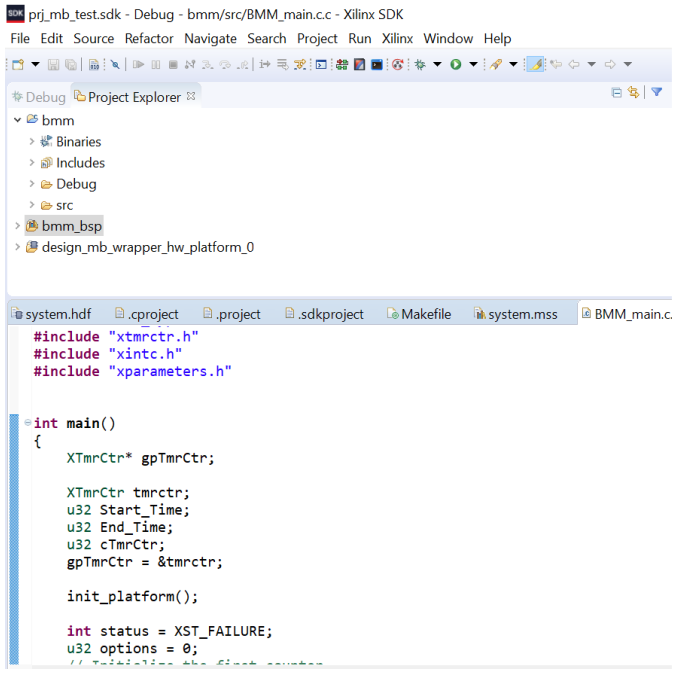
Design on Vivado SDK

1/ Changes in libraries and functions used for timing :

xtime_l.h for ARM Implementation

xtmrctr.h for Microblaze Implementation

2/ Several bugs launching FPGA programming..



High – Level Synthesis

1) Reanalyse the algorithms
2) Directive usage to infer RTL blocks
3) Implement Axi/Axilite interfaces
4) Manage and reshape memory
5) Automated material optimisations

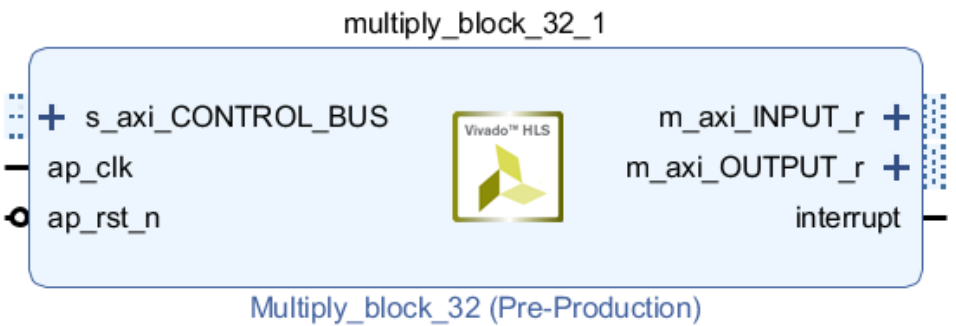
High – Level Synthesis

Control sequence conversion :

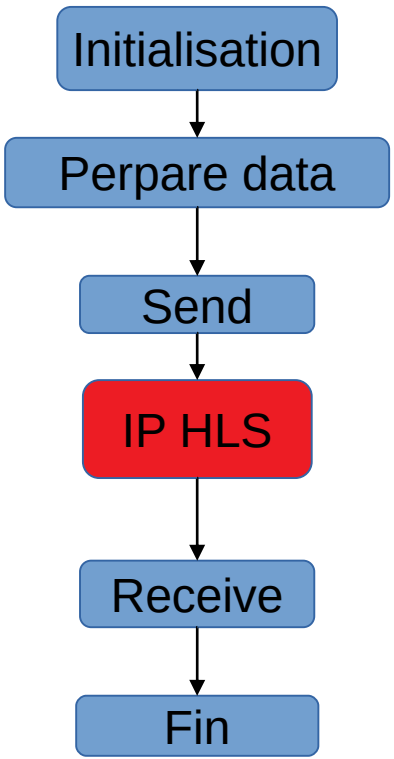
```
While (condition){
        do_stuff();
}
```

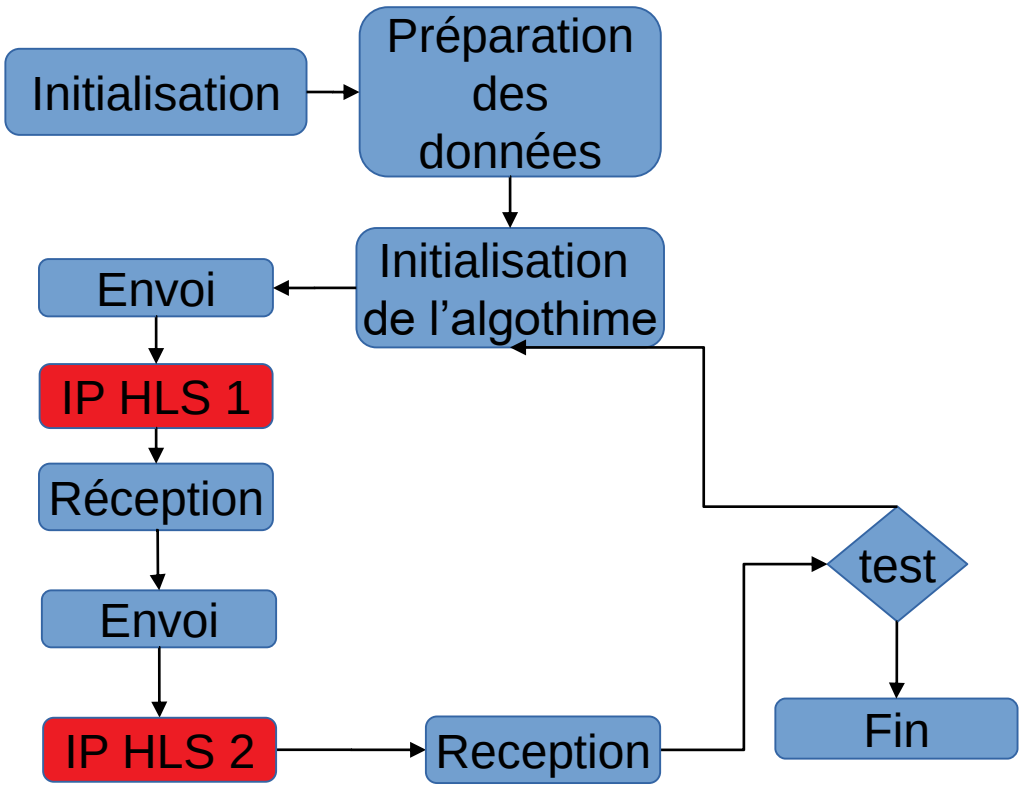
```
For (n iterations){
        do_stuff();
}
```

- **AXI/lite Interface directives :**
- Top-Level function : BUS CONTROL
- Inputs
- Outputs
- Addresses managed by the CPU



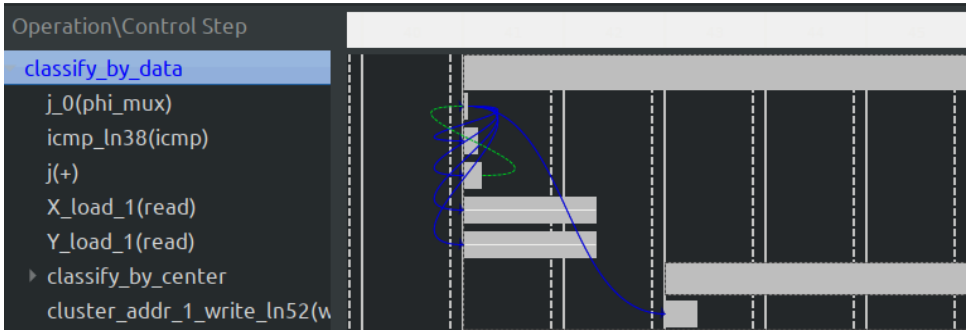
High – Level Synthesis





High – Level Synthesis : scheduler

- Analyse dependencies
- Minimise Initialisation interval
- Respect timing and FPGA resources limits
- If necessary : reshape the arrays



Resource Usage

	VHDL
SLICE	0
LUT	3570
FF	5049
DSP	7
BRAM	5
SRL	104

Final Timing

	VHDL
CP required	10.000
CP achieved post-synthesis	8.254

Timing met

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	7244	-
FIFO	-	-	-	-	-
Instance	4	10	1870	2814	-
Memory	6	-	0	0	0
Multiplexer	-	-	-	2096	-
Register	0	-	6410	224	-
Total	10	10	8280	12378	0
Available	120	80	35200	17600	0
Utilization (%)	8	12	23	70	0

High – Level Synthesis : Synthesis

- Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.750	1.25

- Latency (clock cycles)

- Summary

Latency		Interval		
min	max	min	max	Type
127427	129027	127427	129027	none

- Detail

+ Instance

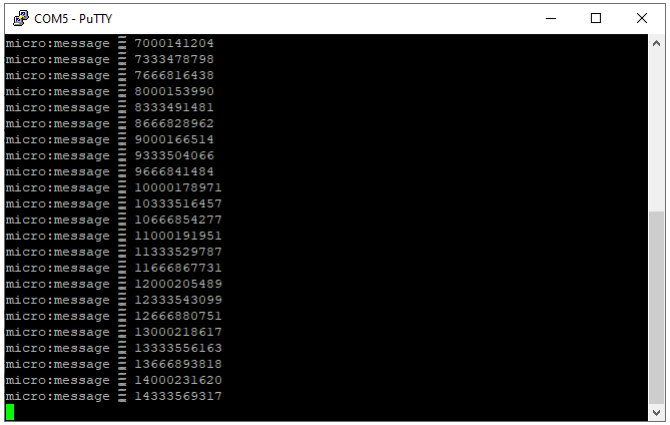
- Loop

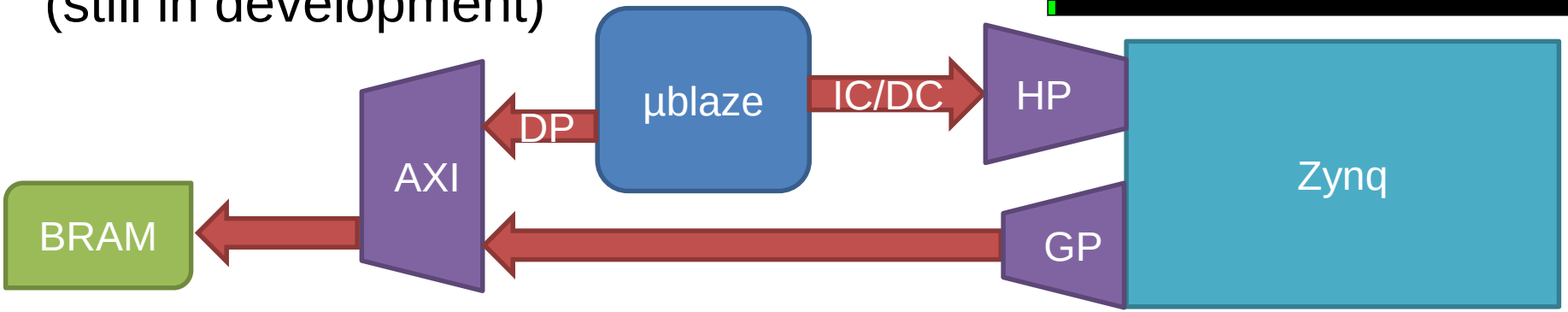
	Latency		Initiation Interval				
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- memcpy.X.in_X	129	129	3	1	1	128	yes
- memcpy.Y.in_Y	129	129	3	1	1	128	yes
- memcpy.X_prot.in_X_prot	5	5	2	1	1	4	yes
- memcpy.Y_prot.in_Y_prot	5	5	2	1	1	4	yes
- Loop 5	126990	128590	12699 ~ 12859	-	-	10	no
+ classify_by_data	9088	9088	71	-	-	128	no
++ classify_by_center	68	68	17	-	-	4	no
+ centers_loop	3608	3768	902 ~ 942	-	-	4	no
++ clustering	896	896	7	-	-	128	no
- memcpy.out_cluster.cluster.gep	129	129	3	1	1	128	yes

- Not always necessary/possible to pipeline all the loops
- HLS must respect time constraints
- Optimisation can be very time/resource costly

Architecture exploration

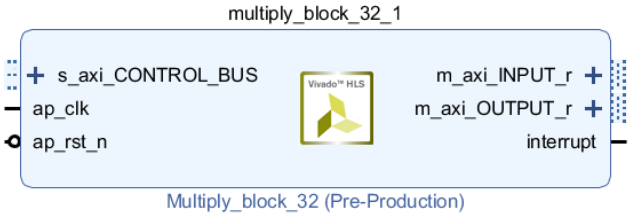
- First design used by the SW team
- 2nd design using BRAM
- 3rd design with BRAM and HLS IP (still in development)

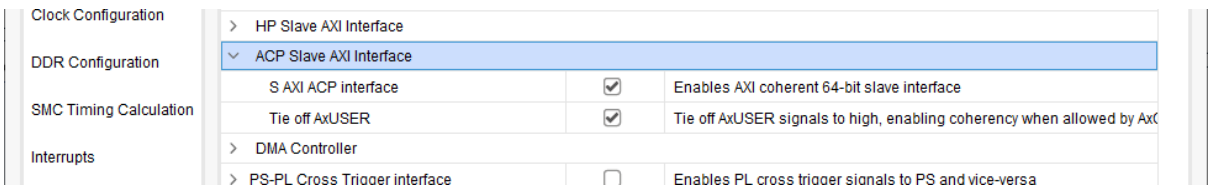


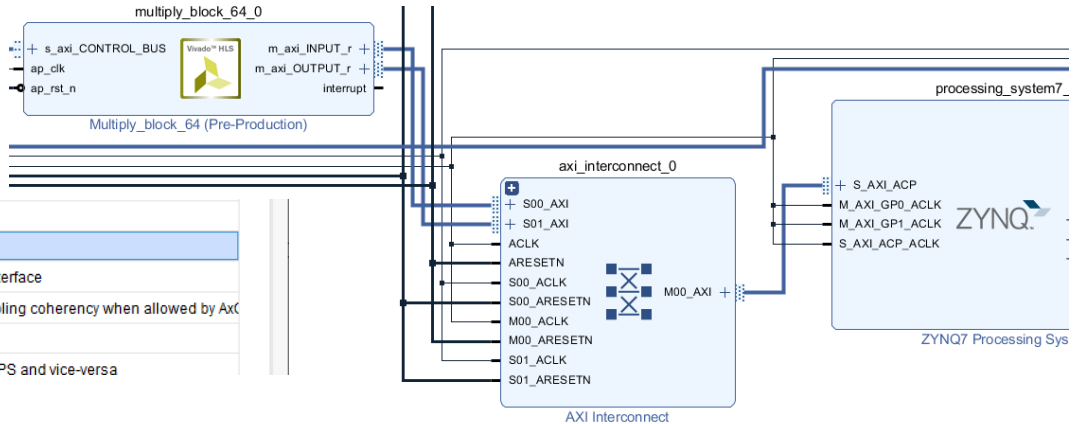


Interfacing Zynq to HLS IP

- Using ACP port => Cache coherency
- Having different clock domain

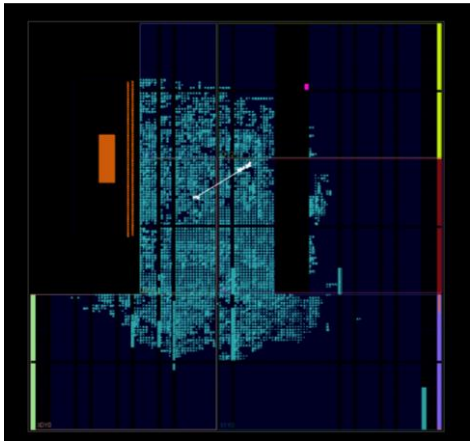


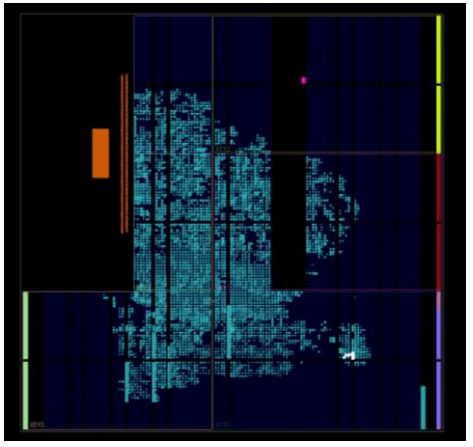




Timings-guided clock improvement

- Using Timing report to increase HLS clock frequency
- Using more performance-oriented placement and synthesis strategies





synth_1 (active)	**constrs_1**	**synth_design Complete!**								**0**	**0**	**0**	**0.0**	**0**	**0**
impl_1 (active)	**constrs_1**	**phys_opt_design (Post-Route) Complete!**	**0.007**	**0.000**	**0.018**	**0.000**	**0.000**	**2.018**	**0**						
impl_2	constrs_1	route_design Complete, Failed Timing!	-0.143	-0.733	0.013	0.000	0.000	2.014	0	8706	12816	13.5	0	10
impl_3	constrs_1	route_design Complete, Failed Timing!	-0.083	-0.240	0.052	0.000	0.000	2.022	0	8705	12816	13.5	0	10
impl_4	constrs_1	route_design Complete, Failed Timing!	-0.106	-0.164	0.051	0.000	0.000	2.014	0	8705	12816	13.5	0	10
synth_2	constrs_1	synth_design Complete!								0	0	0.0	0	0
impl_5	constrs_1	route_design Complete, Failed Timing!	-0.106	-0.164	0.051	0.000	0.000	2.014	0	8705	12816	13.5	0	10

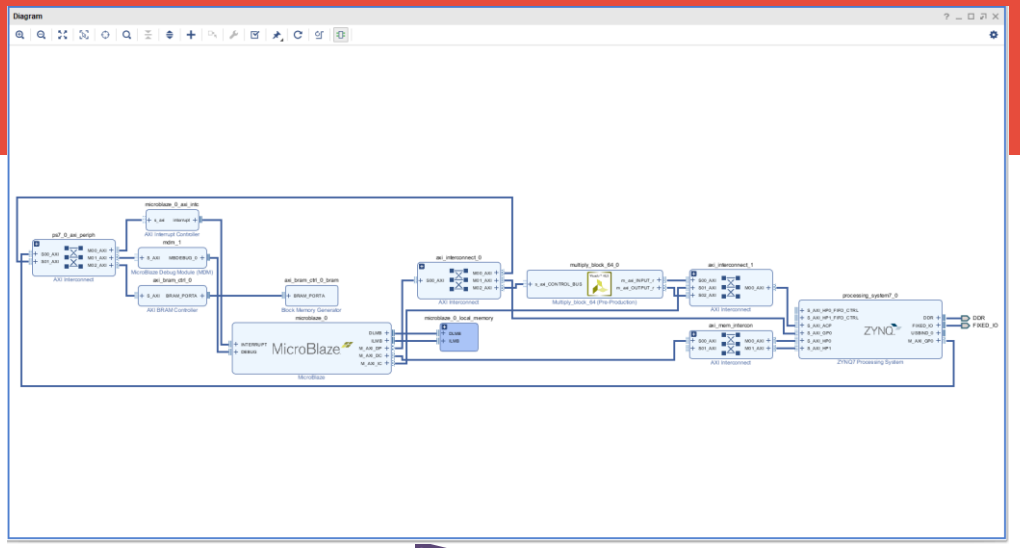
Performances

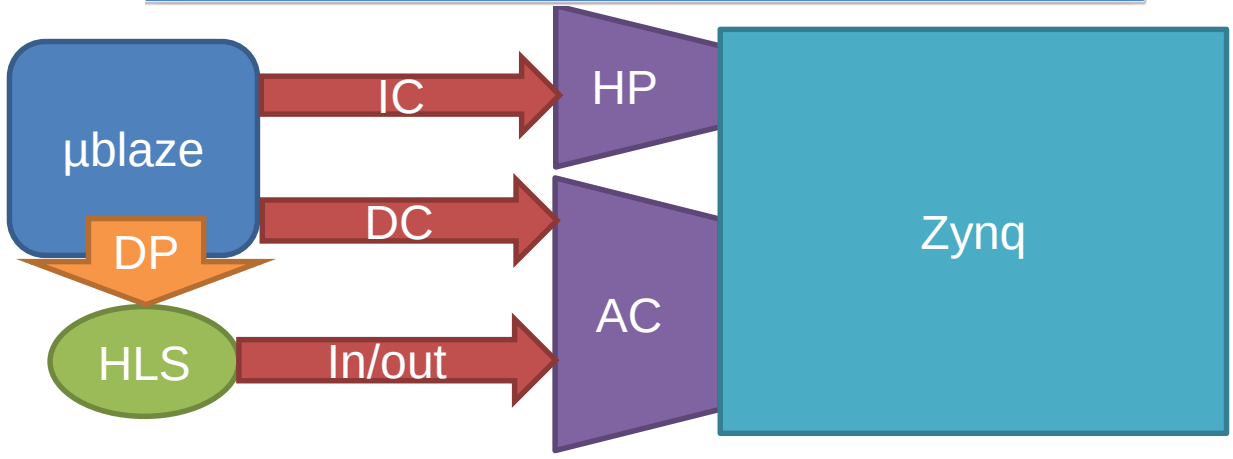
- First time @ 100MHZ
- Test on MUL64:
  - AXI@180MHZ
  - HLS@131MHZ
  - Acceleration : **320%**

IP	HW time (µs)	SW time (µs)	Improved clock	Improve HW times (µs)	Best acceleration
mul64	10280	26132	130MHZ	7042	**271%**
mul32	2412	3282	125 MHZ	1810	81%
pearson	32	5	115 MHZ	21	**-74%**
kmeans	1297	1527	120MHZ	1082	41%

Future improvements

- Using interruption instead of polling
- A microblaze handles every task involving IP
- Multi-microblazes multi-IPs.





Conclusion

- 6 Algorithms implemented and tested
- 3 IP conceived and implemented
- Multi-clock periods for a heterogeneous functionning
- Development method is satisfying with room for improuvement
- Set up the usage of automated development tools
- Develop multi-microblazes multi-Ips
- Maximise board usages

- **Completed a full development cycle : objective achieved**

Conclusion

Thank you for you attention