

06/01/2021

# ROB307 : Multiprocessor System On Chip

Drone agriculture  
nov-dec 2020

For internship exchanges purpose only  
do not disseminate

# Objectif projet

- Concevoir un SOC pour UAV (drone application agriculture) avec outil CAO [Vivado](#) 2019.1 implémentation sur circuit [Xilinx Zynq XC7020](#) et test/vérification sur carte [zedboard](#)
1. **Objectifs :** maximisation performance (min temps d'exécution)
  2. **Contraintes:** surface/énergie
  3. **Temps de développement:** nov-dec 2020 ( 6 séances de 3H30 cours/TP) + travail personnel
  4. **Mode de travail:** distanciel et accès à distance réseau informatique ENSTA et outil CAO



# plan

1. Conception NOC AXI 3x3 , 5x5
2. Intégration processeur ARM 9 + 2 processeurs Microblaze
3. Intégration processeur ARM 9 + 4 processeurs Microblaze
4. Programmation et MPSoC
5. Automatisation exploration options IPs TCL
6. Automatisation exploration options IPs pilotée par algorithme optimisation multicritère
7. conclusion

# Conception NoC

Conception NOC AXI 3x3 , 5x5

Intégration processeur ARM 9 + 2 processeurs Microblaze

Intégration processeur ARM 9 + 4 processeurs Microblaze

# Conception du NOC

## Objectif :

- Construire un design de base qui permet d'effectuer des traitements parallèles sur un ou plusieurs processeurs.

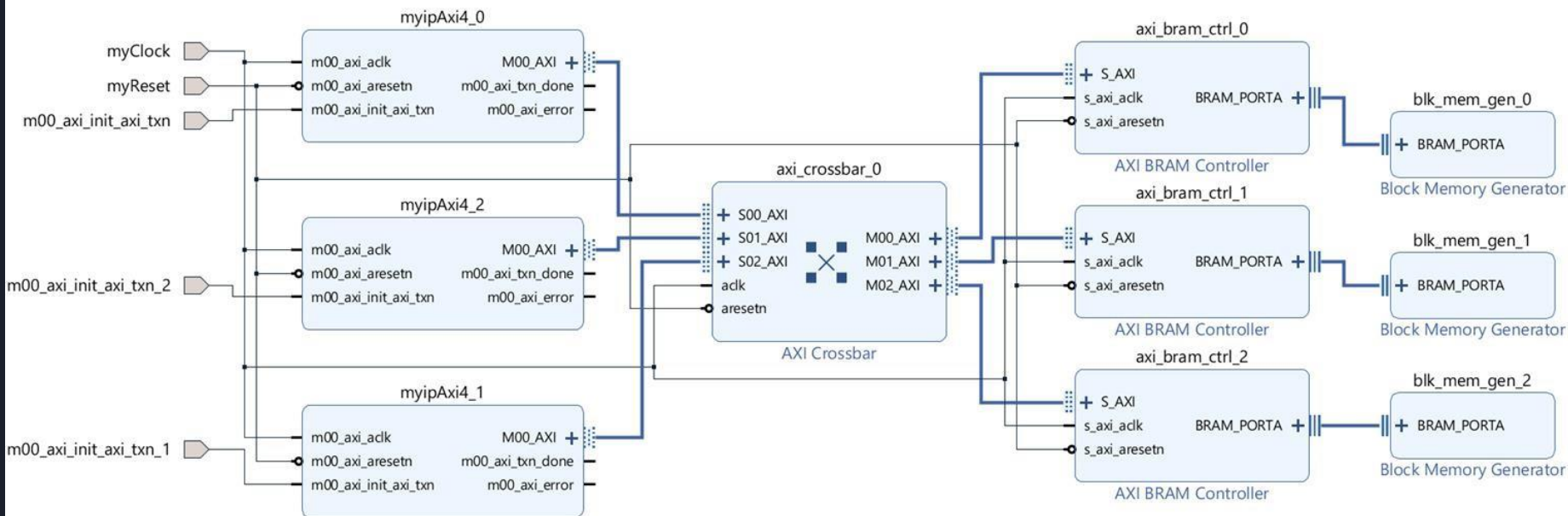
**Principe :** Utiliser le AXI Crossbar Interconnect.

**Test :** Tester sur des IPs simples : des machines d'état

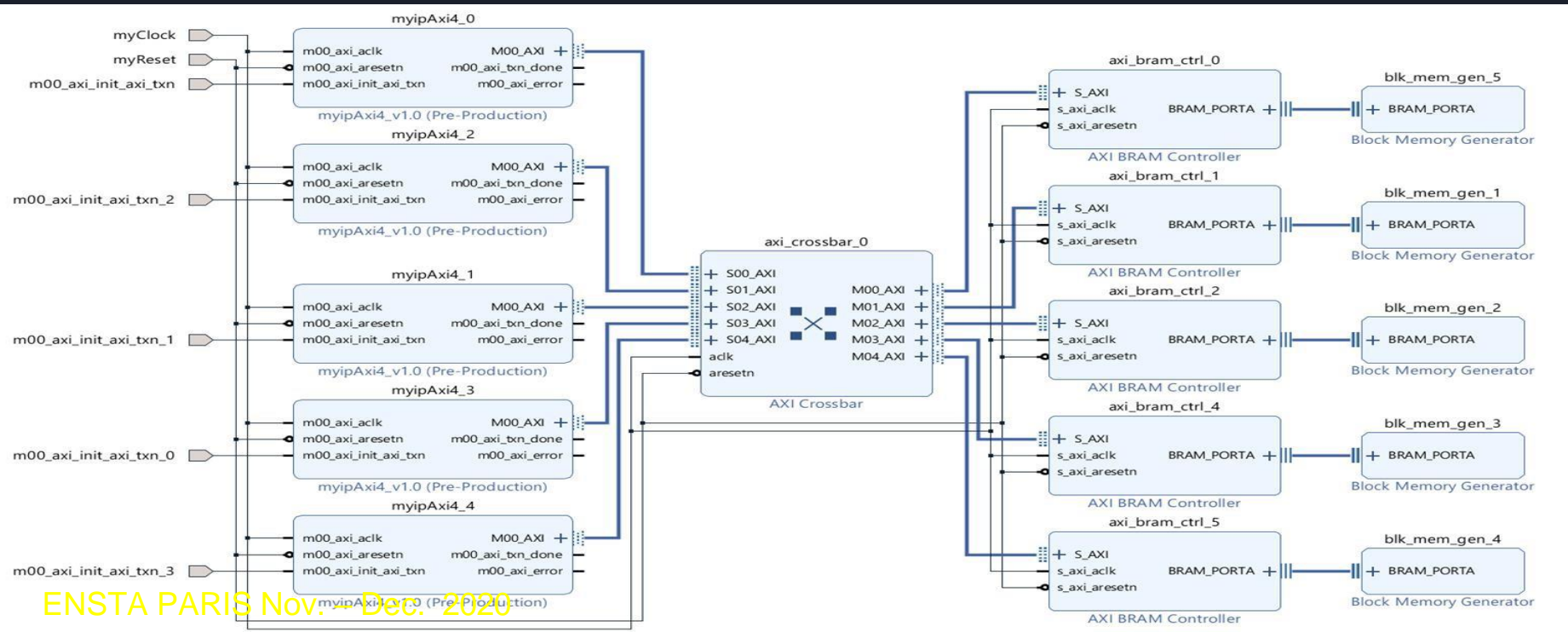
## References IP

1. [AXI Interconnect v2.1](#)
2. [AXI4-Stream interconnect](#)

# Conception du NOC 3x3



# Conception du NOC 5x5





## **Deux modes de connectivité :**

- Full Crossbar mode
- Shared Access Mode

## **Le protocole choisi pour la communication :**

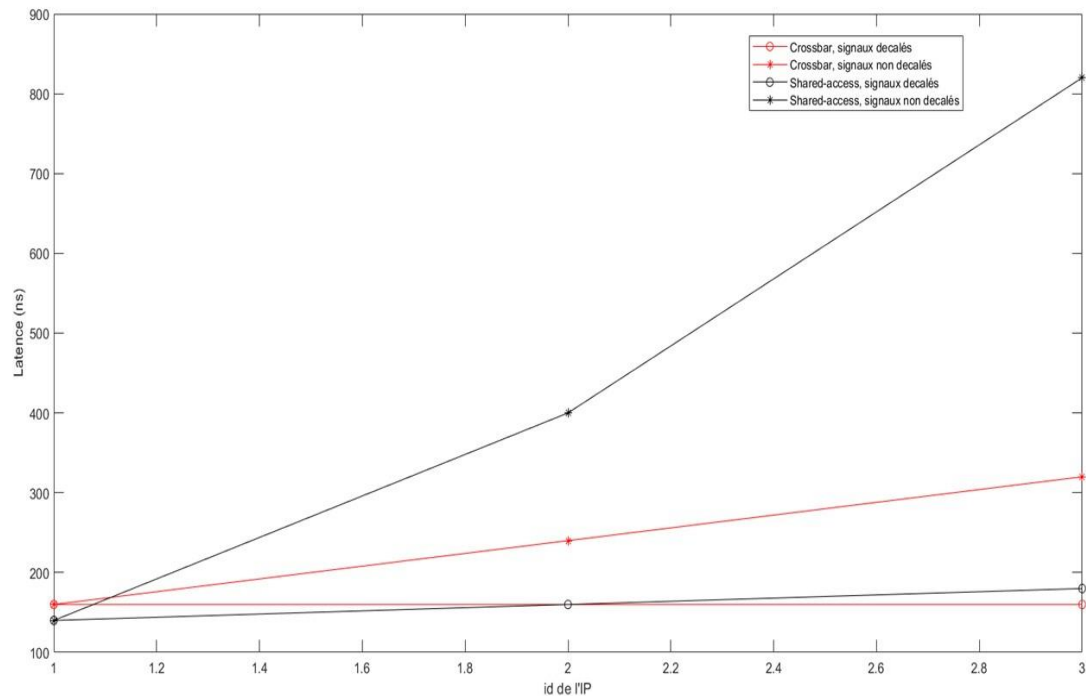
- AXI4

## **La génération des signaux :**

- Signaux décalés
- Signaux générés aux même instant

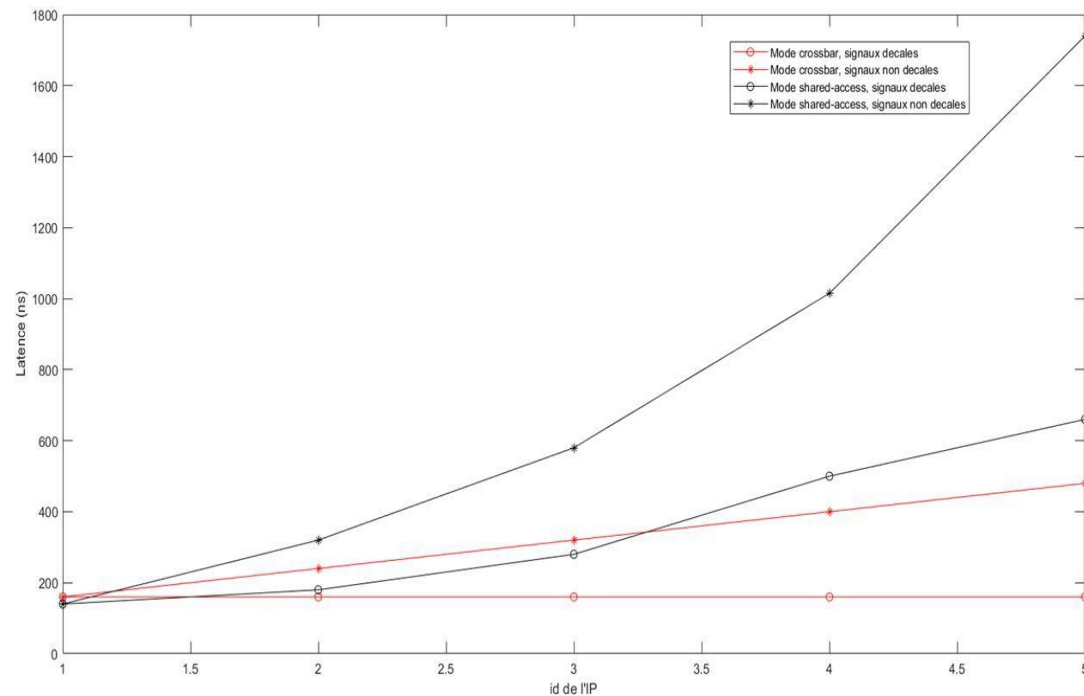
# Simulation du NOC 3x3

## Latence

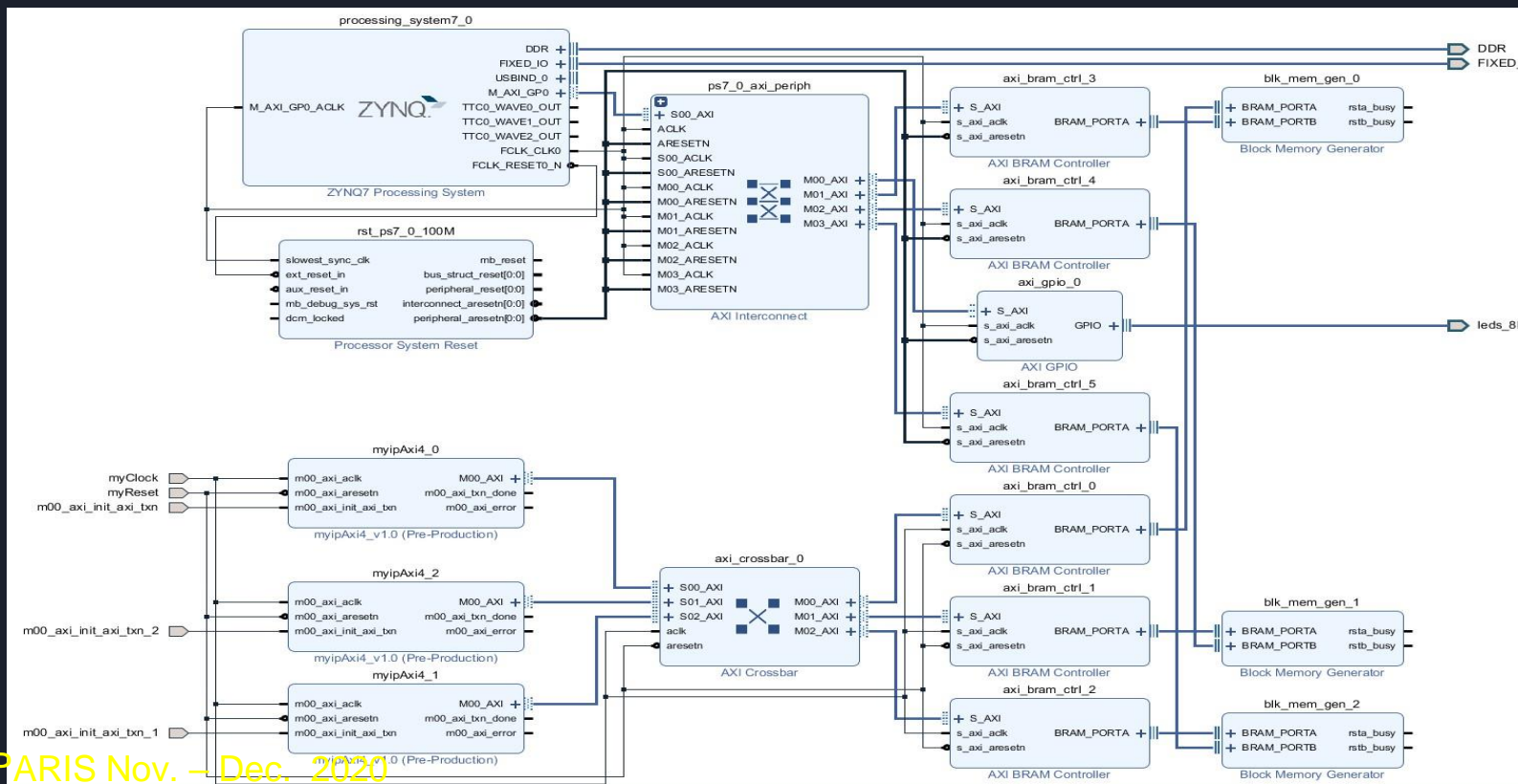


# Simulation du NOC 5X5

## Latence



# Implémentation sur Zynq - NOC 3x3



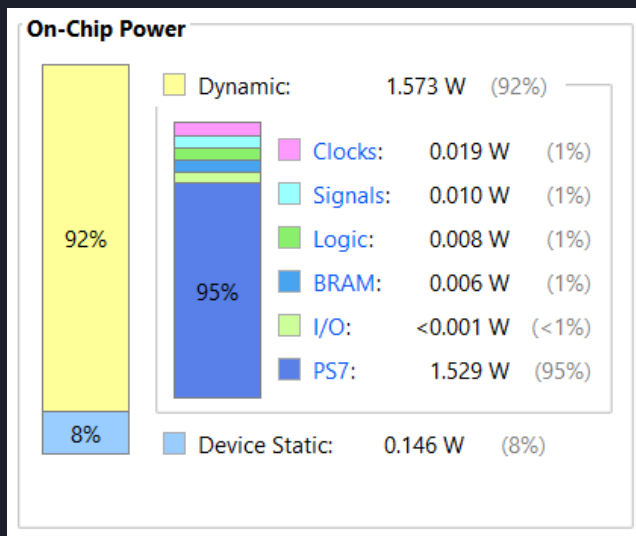


# Informations de placement routage : NOC 3X3

## 1. Timing

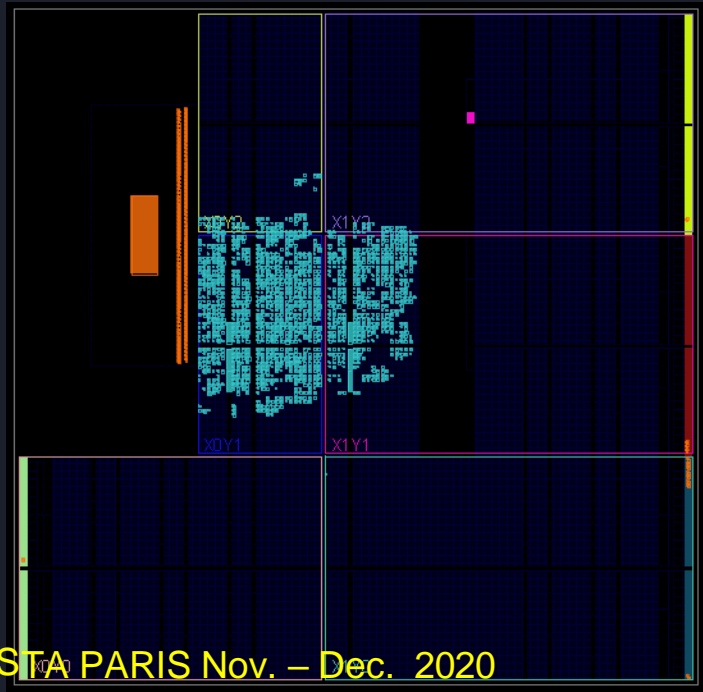
Worst Negative Slack (WNS): 0,237 ns  
 Total Negative Slack (TNS): 0,000 ns  
 Number of Failing Endpoints: 0  
 Total Number of Endpoints: 8572

## 2. Consommation d'énergie

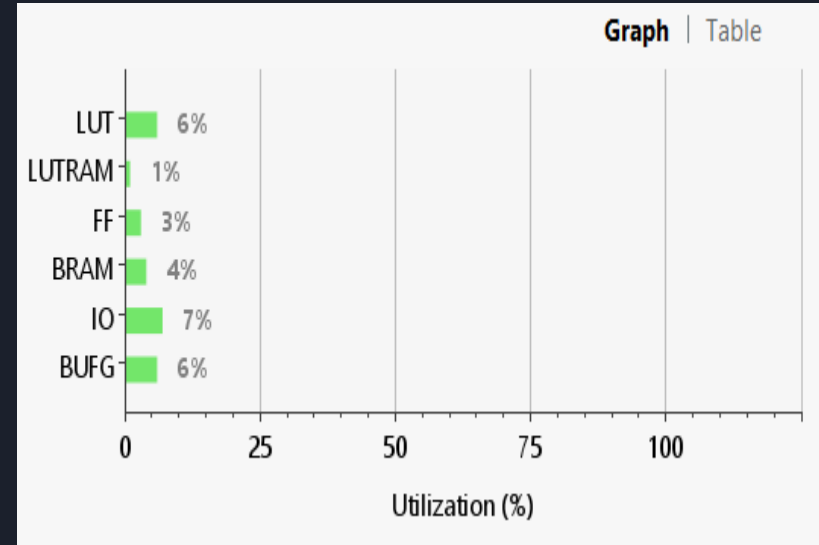


# Informations de placement routage : NOC 3X3

## 3. Surface occupée



## 4. Utilisation matérielle

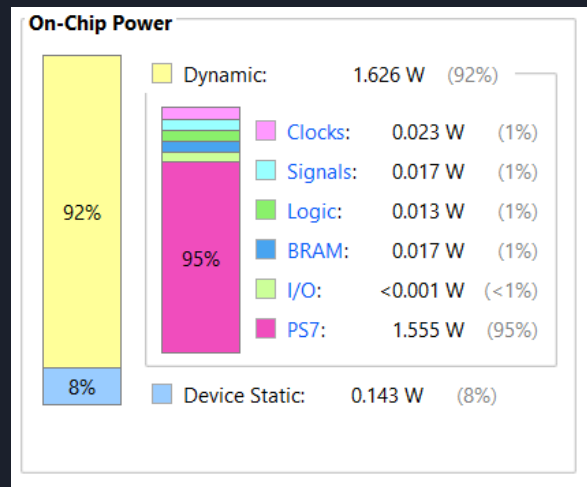


# Informations de placement routage : NOC 5X5

## 1. Timing

Setup	
Worst Negative Slack (WNS):	-1,187 ns
Total Negative Slack (TNS):	-43,062 ns
Number of Failing Endpoints:	61
Total Number of Endpoints:	13902
Timing constraints are not met.	

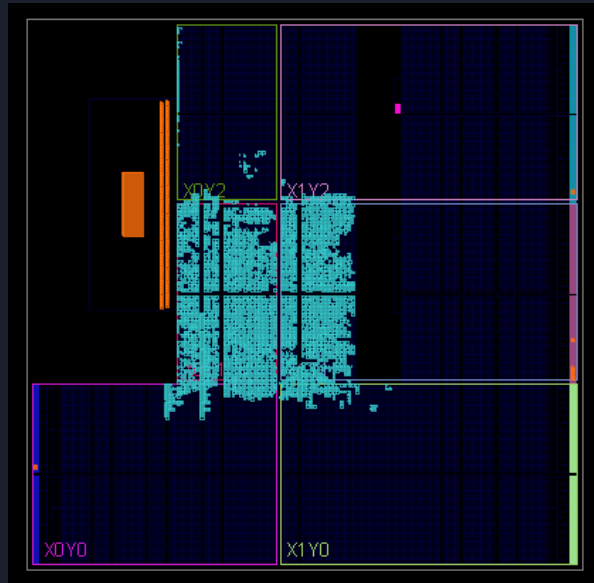
## 2. Consommation d'énergie



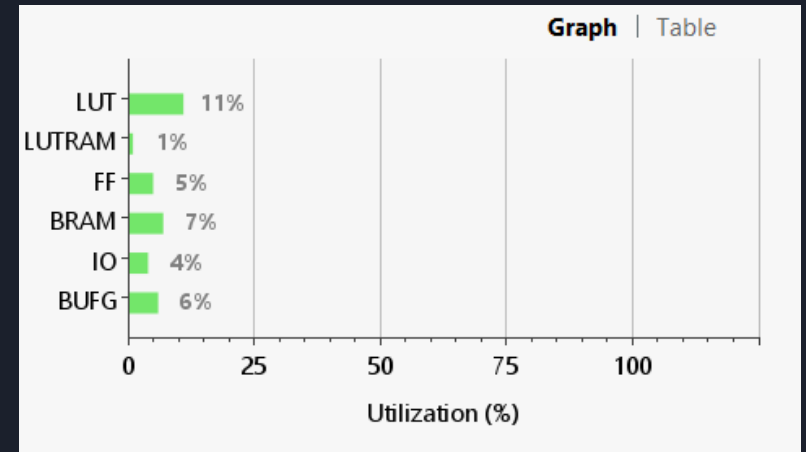


# Informations de placement routage : NOC 5X5

## 3. Surface occupée



## 4. Utilisation matérielle



# Test et validation sur ZedBoard - NOC 3X3

ROB307, DRONE AGRICULTEUR  
Signaux décalés

Saved data 1 is : 0, saved in 0x40000000  
Saved data 2 is : 0, saved in 0x42000000  
Saved data 3 is : 0, saved in 0x44000000

Next set of data

\*\*\*\*\*

ROB307, DRONE AGRICULTEUR  
Signaux décalés

Saved data 1 is : 1, saved in 0x40000000  
Saved data 2 is : 0, saved in 0x42000000  
Saved data 3 is : 0, saved in 0x44000000

Next set of data

\*\*\*\*\*

ROB307, DRONE AGRICULTEUR  
Signaux décalés

Saved data 1 is : 1, saved in 0x40000000  
Saved data 2 is : 0, saved in 0x42000000  
Saved data 3 is : 1, saved in 0x44000000

Next set of data [

\*\*\*\*\*

ROB307, DRONE AGRICULTEUR  
Signaux décalés

Saved data 1 is : 1, saved in 0x40000000  
Saved data 2 is : 1, saved in 0x42000000  
Saved data 3 is : 1, saved in 0x44000000

Next set of data

2020

\*\*\*\*\*

ROB307, DRONE AGRICULTEUR  
Signaux décalés

Saved data 1:

1 2 3 4 5 6 7 8 9, saved in 0x40000000

Saved data 2:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17, saved in 0x42000000

Saved data 3:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33, saved in 0x44000000

\*\*\*\*\*

Next set of data

\*\*\*\*\*

# Test et validation sur ZedBoard - NOC 5X5

Saved data 1 is : 0, saved in 0x40000000  
 Saved data 2 is : 0, saved in 0x42000000  
 Saved data 3 is : 0, saved in 0x44000000

Saved data 4 is : 0, saved in 0x46000000  
 Saved data 5 is : 0, saved in 0x48000000

Next set of data

Saved data 1 is : 0, saved in 0x40000000  
 Saved data 2 is : 1, saved in 0x42000000  
 Saved data 3 is : 0, saved in 0x44000000

Saved data 4 is : 0, saved in 0x46000000  
 Saved data 5 is : 0, saved in 0x48000000

Next set of data

Saved data 1 is : 1, saved in 0x40000000  
 Saved data 2 is : 1, saved in 0x42000000  
 Saved data 3 is : 0, saved in 0x44000000

Saved data 4 is : 0, saved in 0x46000000  
 Saved data 5 is : 0, saved in 0x48000000

Next set of data

Saved data 1 is : 1, saved in 0x40000000  
 Saved data 2 is : 1, saved in 0x42000000  
 Saved data 3 is : 0, saved in 0x44000000

Saved data 4 is : 1, saved in 0x46000000  
 Saved data 5 is : 0, saved in 0x48000000

Next set of data

Saved data 1 is : 1, saved in 0x40000000  
 Saved data 2 is : 1, saved in 0x42000000  
 Saved data 3 is : 1, saved in 0x44000000

Saved data 4 is : 1, saved in 0x46000000  
 Saved data 5 is : 0, saved in 0x48000000

Next set of data

Saved data 1 is : 1, saved in 0x40000000  
 Saved data 2 is : 1, saved in 0x42000000  
 Saved data 3 is : 1, saved in 0x44000000

Saved data 4 is : 1, saved in 0x46000000  
 Saved data 5 is : 1, saved in 0x48000000

Next set of data

Saved data 1 :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 saved in 0x40000000

Saved data 2 :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 saved in 0x42000000

Saved data 3 :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 saved in 0x44000000

Saved data 4 :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 saved in 0x46000000

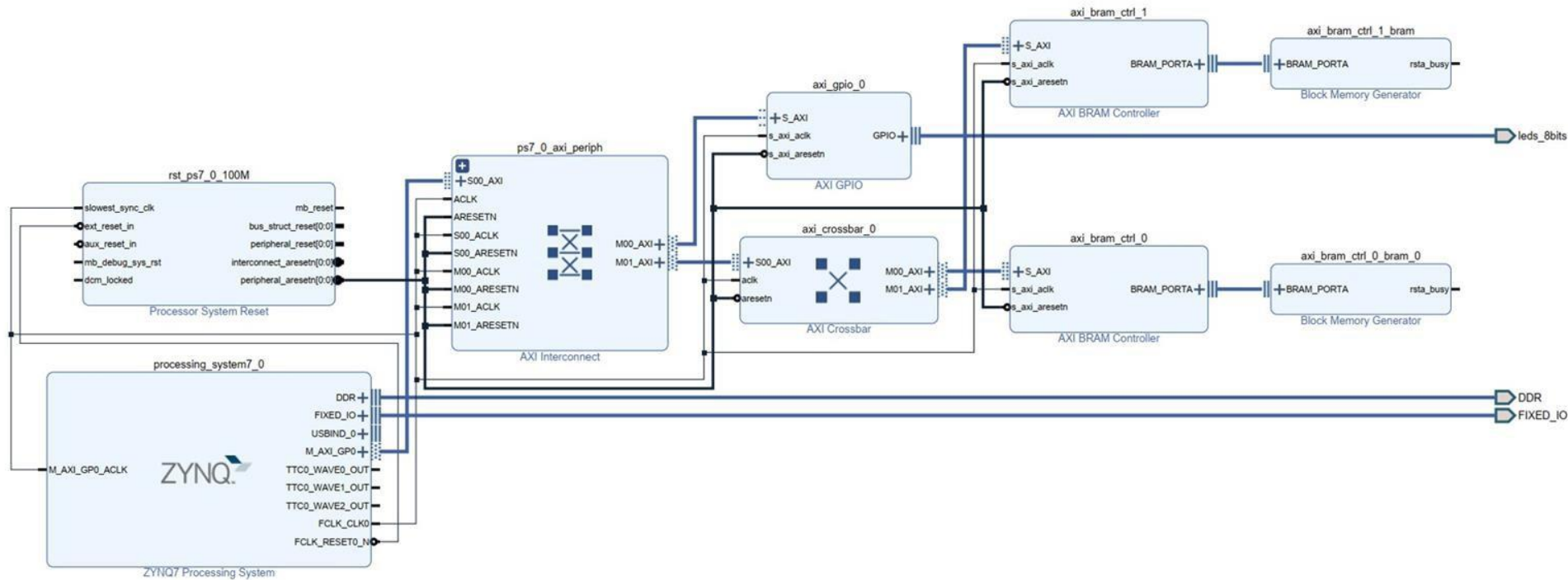
Saved data 5 :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 saved in 0x48000000

Next set of data

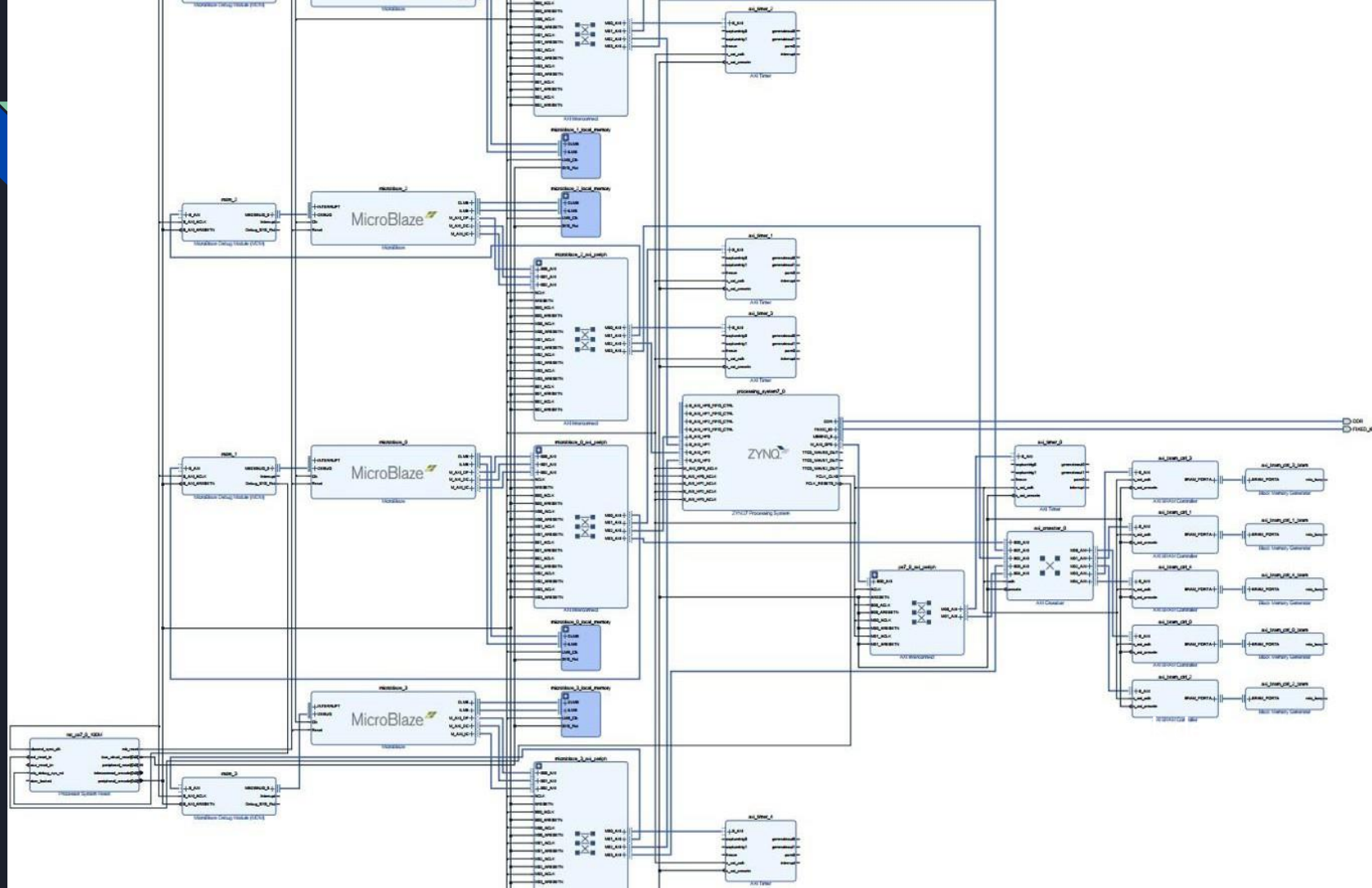
# Intégration Processeurs ARM9 et Microblaze

# Conception des block designs







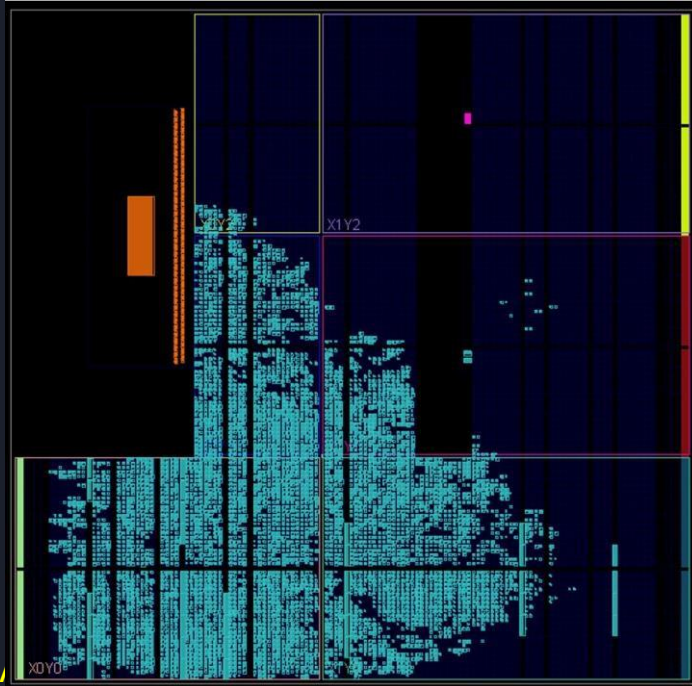


5:

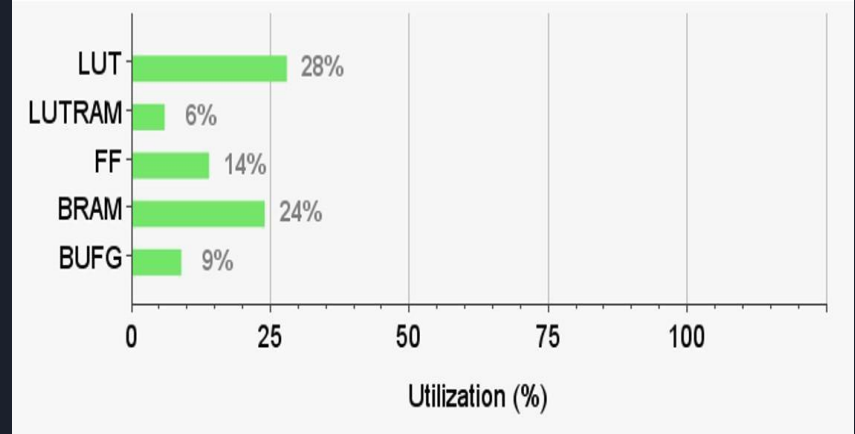


# Synthèse des block designs : 1ARM + 2 MB

- Occupation de la surface



- Utilisation matérielle



# Synthèse des block designs : 1ARM + 2 MB

- Rapport du timing

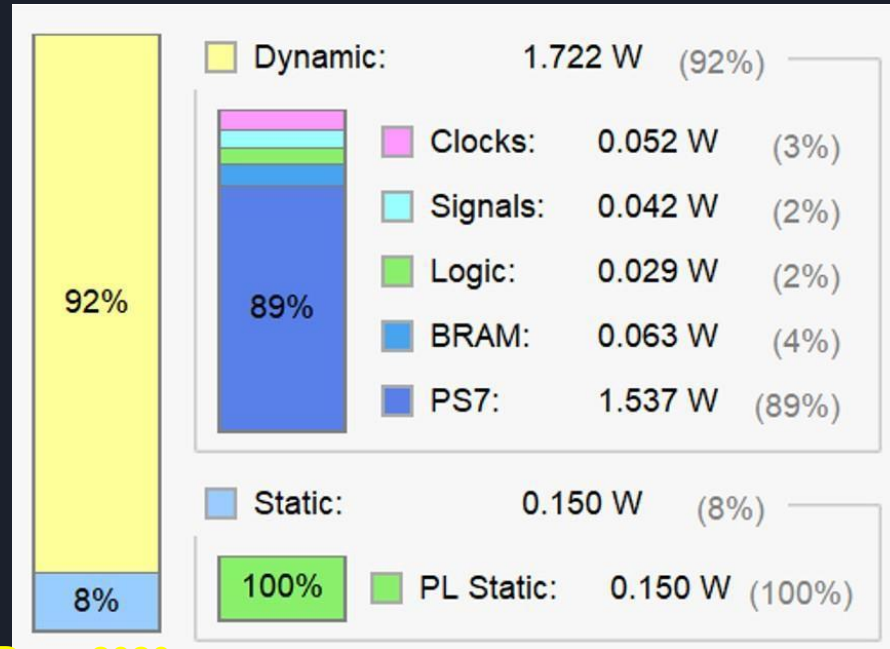
## Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 0.482 ns	Worst Hold Slack (WHS): 0.012 ns	Worst Pulse Width Slack (WPWS): 3.750 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 43848	Total Number of Endpoints: 43848	Total Number of Endpoints: 16553

All user specified timing constraints are met.

# Synthèse des block designs : 1ARM + 2 MB

- Consommation d'énergie



# Synthèse des block designs : 1ARM + 4 MB

## Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): -0.076 ns	Worst Hold Slack (WHS): 0.016 ns	Worst Pulse Width Slack (WPWS):
Total Negative Slack (TNS): -1.081 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS):
Number of Failing Endpoints: 20	Number of Failing Endpoints: 0	Number of Failing Endpoints:
Total Number of Endpoints: 81901	Total Number of Endpoints: 81901	Total Number of Endpoints:

Timing constraints are not met.

→ Directive de l'optimisation sous Vivado : **Performance ExtraTiming Opt**

# Synthèse des block designs : 1ARM + 4 MB

- Rapport du timing

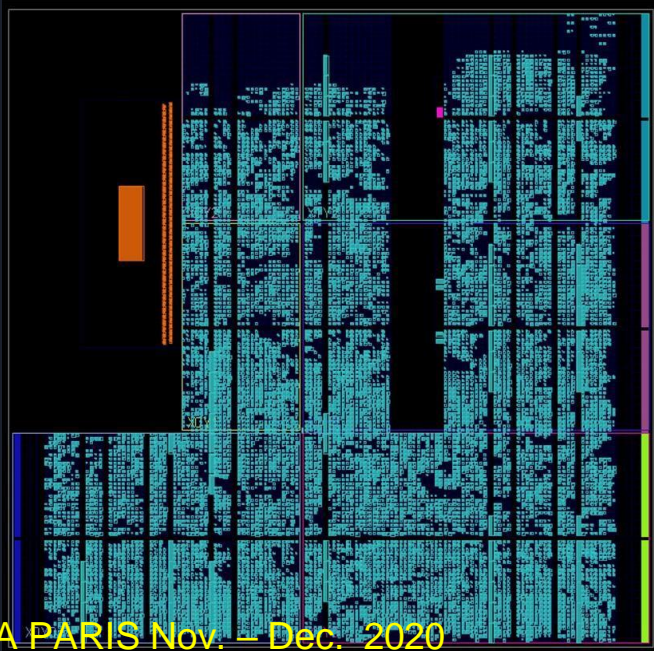
## Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 0.008 ns	Worst Hold Slack (WHS): 0.054 ns	Worst Pulse Width Slack (WPWS): 3.750 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 81901	Total Number of Endpoints: 81901	Total Number of Endpoints: 30753

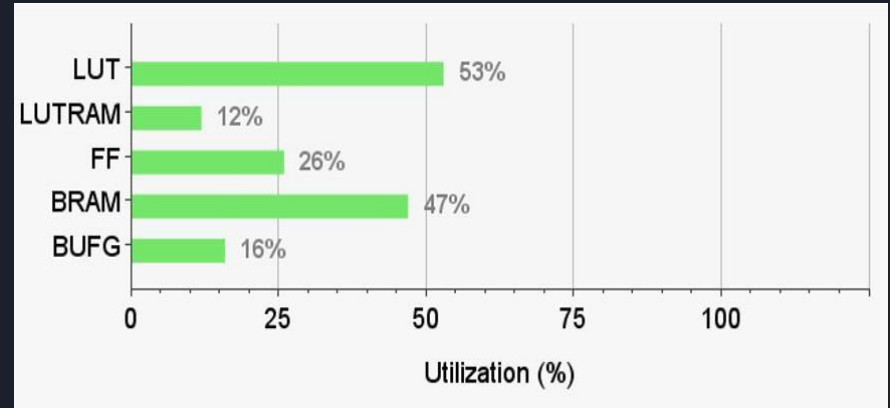
All user specified timing constraints are met.

# Synthèse des block designs : 1ARM + 4 MB (5 cœurs de processeurs)

- Occupation de la surface

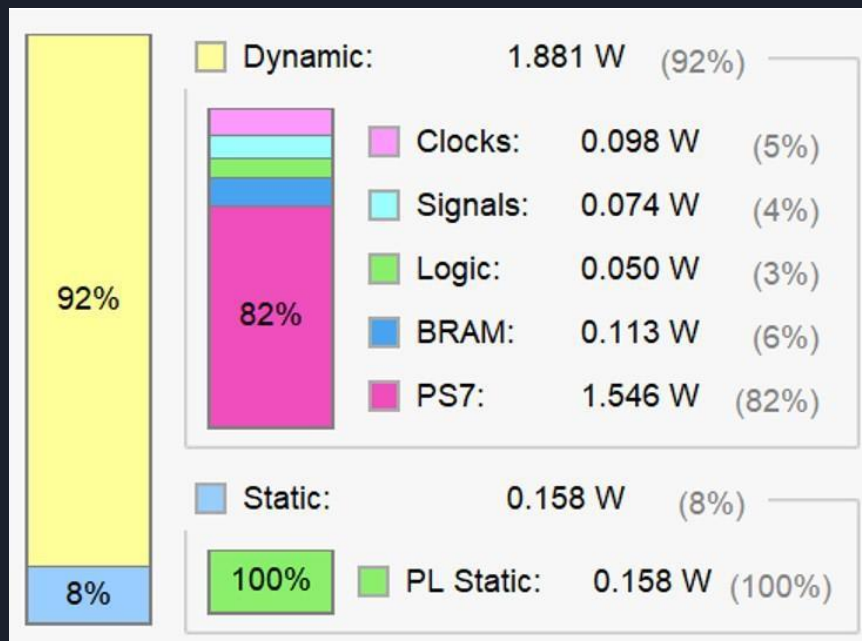


- Utilisation matérielle



# Synthèse des block designs

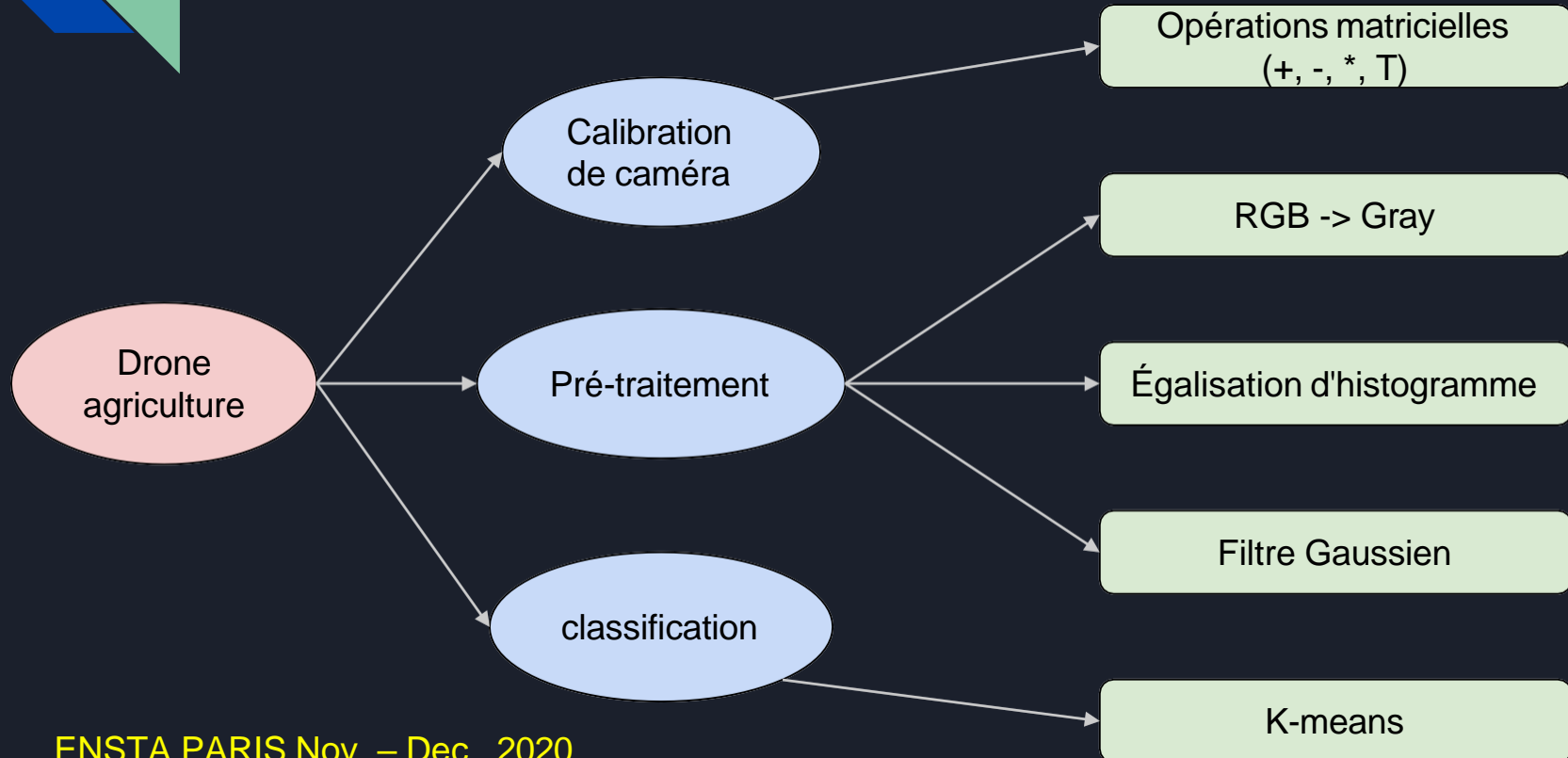
- Consommation d'énergie



# Programmation MPSOC



# Programmation SDK



# Programmation

## ➤ Calibration de caméra

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & \nu \end{bmatrix} \begin{bmatrix} l_1 \\ l_2 \\ 0 \end{bmatrix} = \begin{bmatrix} l_1 a_{11} + l_2 a_{12} \\ l_1 a_{21} + l_2 a_{22} \\ l_1 v_1 + l_2 v_2 \end{bmatrix}$$

## Transformation projective



## Homographie

# Programmation

## ➤ Opérations matricielles

```
void matrix_operation(mat_a a[IN_A_ROWS][IN_A_COLS],
                    mat_b b[IN_B_ROWS][IN_B_COLS],
                    mat_result c[IN_A_ROWS][IN_B_COLS], char OPERATION)
{
    switch(OPERATION)
    {
        case 'A':
            matrix_add(a, b, c);
            break;
        case 'S':
            matrix_sub(a, b, c);
            break;
        case 'M':
            matrix_multi(a, b, c);
            break;
        case 'T':
            matrix_multi(a, c);
            break;
    }
}
```

```
void matrix_multi(mat_a a[IN_A_ROWS][IN_A_COLS],
                mat_b b[IN_B_ROWS][IN_B_COLS],
                mat_result c[IN_A_ROWS][IN_B_COLS])
{
    for(int i = 0; i < IN_A_ROWS; i++){
        for(int j = 0; j < IN_B_COLS; j++){
            // Iterate over the columns of the B matrix
            int sum = 0;
            // Do the inner product of a row of A and col of B
            for(int k = 0; k < IN_B_ROWS; k++){
                sum += a[i][k] * b[k][j];
            }
            c[i][j] = sum;
        }
    }
}

void matrix_add(mat_a a[IN_A_ROWS][IN_A_COLS],
                mat_b b[IN_B_ROWS][IN_B_COLS],
                mat_result c[IN_A_ROWS][IN_B_COLS])
{
    for(int i = 0; i < IN_A_ROWS; i++){
        for(int j = 0; j < IN_A_COLS; j++){
            c[i][j] = a[i][j] + b[i][j];
        }
    }
}
```

# Programmation

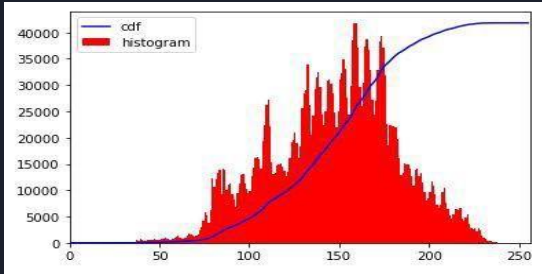
➤ RGB -> Gray



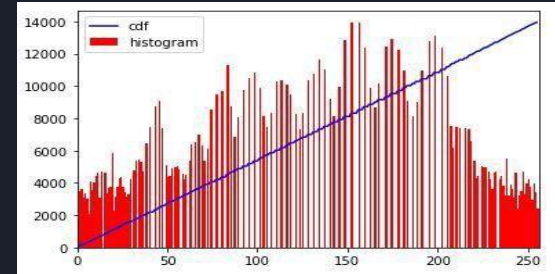
$$gray = 0.299r + 0.587g + 0.114b$$

# Programmation

## ➤ Égalisation d'histogramme



## Augmenter le contraste



# Programmation

## ➤ Égalisation d'histogramme

```
//Calculate initial histogram
for (i = 0; i < IN_ROWS; i++){
    for (j = 0; j < IN_COLS; j++){
        ini_hist[img[i][j]]++;
    }
}

//Calculate Cumulative Distribution Function
for (i = 0; i < 256; i++){
    for (j = 0; j <= i; j++){
        Fx[i] += ini_hist[j];
    }
    Fx[i] /= IN_ROWS*IN_COLS;
}

//Gray scale mapping
for (i = 0; i < 256; i++){
    hist_mapping[i] = (int)255 * Fx[i] + 0.5;
}
// New image after histogram equalization
for (i = 0; i < IN_ROWS; i++){
    for (j = 0; j < IN_COLS; j++){
        img[i][j] = hist_mapping[img[i][j]];
    }
}
```

# Programmation

## ➤ Filtre Gaussien



- Réduire du bruit
- Extraire des caractéristiques dans une plus grande échelle

# Programmation

## ➤ Filtre Gaussien

```
int x, y;
int m = size / 2;
kernel_type sum = 0;

//get kernel
for (y = 0; y < size; y++)
{
    for (x = 0; x < size; x++)
    {
        kernel[y][x] = (1 / (2 * 3.14 * sigma * sigma)) * exp(-((x - m) * (x - m) + (y - m) * (y - m)) / (2 * sigma * sigma));
        sum += kernel[y][x];
    }
}

//normal
for (y = 0; y < size; y++)
{
    for (x = 0; x < size; x++)
    {
        kernel[y][x] /= sum;
    }
}
```

calcul du noyau

```
for (y = m; y < ROW - m; y++)
{
    for (x = m; x < COL - m; x++)
    {
        value = 0;
        k = 0;
        for (j = -m; j < m; j++)
        {
            for (i = -m; i < m; i++)
            {
                uchar temp = src[y+j][x+i];
                kernel_type temp1 = kernel_vec[k++];
                value += temp * temp1;
            }
        }
        dst[y][x] = (uchar)(value);
    }
}
```

glissement sur l'image

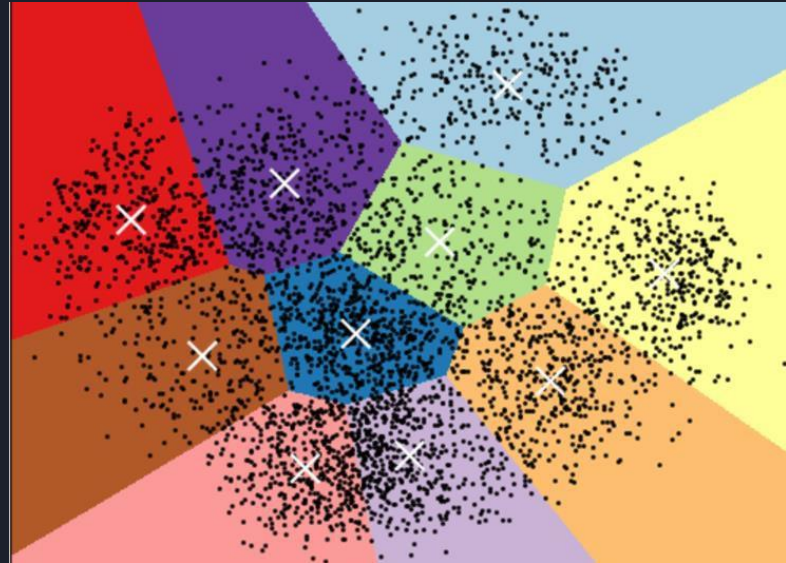


# Programmation

## ➤ K-Means



terrains divisés



ROB311 TP6

# Programmation

## ➤ K-Means

```
void cluster(Point *point, Point *mean, int *center, int n, int k)
{
    int i, j, q;
    float min;
    float distance[n][k];
    for(i = 0; i < n; ++i)
    {
        min = 999999.0;
        for(j = 0; j < k; ++j)
        {
            distance[i][j] = getDistance(point[i], mean[j]);
        }
        for(q = 0; q < k; ++q)
        {
            if(distance[i][q] < min)
            {
                min = distance[i][q];
                center[i] = q;
            }
        }
        printf("( %.0f, %.0f )\t in cluster-%d\n", point[i].x, point[i].y, center[i] + 1);
    }
    printf("-----\n");
}
```

algorithme principal

```
void getMean(Point *point, Point *mean, int *center, int n, int k)
{
    Point tep;
    int i, j, count = 0;
    for(i = 0; i < k; ++i)
    {
        count = 0;
        tep.x = 0.0;
        tep.y = 0.0;
        for(j = 0; j < n; ++j)
        {
            if(i == center[j])
            {
                count++;
                tep.x += point[j].x;
                tep.y += point[j].y;
            }
        }
        tep.x /= count;
        tep.y /= count;
        mean[i] = tep;
    }
    for(i = 0; i < k; ++i)
    {
        printf("The new center point of %d is : \t( %f, %f )\n", i+1, mean[i].x, mean[i].y);
    }
}
```

Calculer des centroids

# Étape d'intégration HW et SW

## ❖ **La synchronisation des cœurs :**

### ➤ **Le fonctionnement synchrone**

- Démarrer 5 algorithmes différents en même temps, avec chacun sur un cœur
- Comparer le résultat de chaque algorithme de SDK et celui obtenu en PC

### ➤ **Le partage de la mémoire**

- L'implémentation de sémaphore

# Fonctionnement synchrone des multi-cœurs

## ➤ Séparation de mémoire en DDR

Available Memory Regions

Name	Base Address	Size
axi_bram_ctrl_0_Mem0	0x40000000	0x2000
axi_bram_ctrl_1_Mem0	0x42000000	0x2000
axi_bram_ctrl_2_Mem0	0x44000000	0x2000
axi_bram_ctrl_3_Mem0	0x46000000	0x2000
axi_bram_ctrl_4_Mem0	0x48000000	0x2000
ps7_dds_0	0x1000000	0x3000000
ps7_qspi_linear_0	0xFC000000	0x1000000
ps7_ram_0	0x0	0x30000
ps7_ram_1	0xFFFF0000	0xFE00

L'adresse de DDR pour ARM0

Available Memory Regions

Name	Base Address	Size
microblaze_0_local_memory_ilmb_bram_if_cntlr_Mem_mic...	0x50	0x7FB0
axi_bram_ctrl_0_Mem0	0x40000000	0x2000
axi_bram_ctrl_1_Mem0	0x42000000	0x2000
axi_bram_ctrl_2_Mem0	0x44000000	0x2000
axi_bram_ctrl_3_Mem0	0x46000000	0x2000
axi_bram_ctrl_4_Mem0	0x48000000	0x2000
ps7_dds_0_HP0_AXI_BASENAME	0x4000000	0x1000000

L'adresse de DDR pour MicroBlaze0

# Fonctionnement synchrone des multi-cœurs

## ➤ Modification de Run Configuration

Name: System Debugger using Debug\_ARM0.elf on Local

Target Setup Application Arguments Environment Symbol Files Source Path Map Common

☒ Stop at 'main'

Summary

Download	Processor	Project	Application	Details
<input checked="" type="checkbox"/>	microblaze_0	MicroBlaze0	Debug/MicroBlaze0.elf	stop at entry = false, relocate elf ...
<input checked="" type="checkbox"/>	microblaze_1	MicroBlaze1	Debug/MicroBlaze1.elf	stop at entry = false, relocate elf ...
<input checked="" type="checkbox"/>	microblaze_2	MicroBlaze2	Debug/MicroBlaze2.elf	stop at entry = false, relocate elf ...
<input checked="" type="checkbox"/>	microblaze_3	MicroBlaze3	Debug/MicroBlaze3.elf	stop at entry = false, relocate elf ...
<input checked="" type="checkbox"/>	ps7_cortexa9_0	ARM0	Debug/ARM0.elf	reset = true, stop at entry = false, ...
<input type="checkbox"/>	ps7_cortexa9_1			reset = true, stop at entry = false, ...

# Fonctionnement synchrone des multi-cœurs

- Modification de stack/heap size

**Stack and Heap Sizes**

Stack Size

Heap Size

ARM

**Stack and Heap Sizes**

Stack Size

Heap Size

MicroBlaze (local memory : 32 MB)

# Fonctionnement synchrone des multi-cœurs

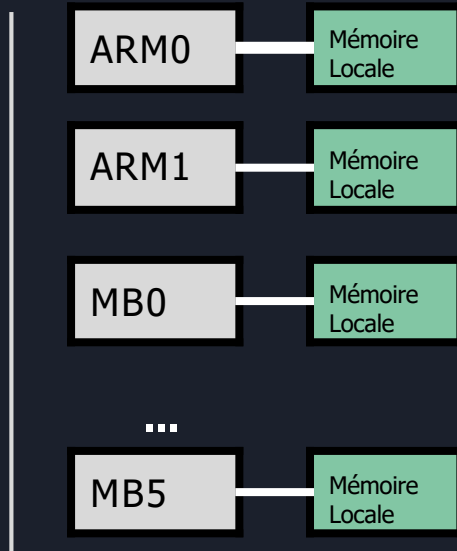
- Mesure du temps pour MB fréquence = 50 MHz, PS fréquence = 666.67MHz

Algorithme	Nombre de cycles	Temps(s)
K-means (sur PS)	12953787	0.248
Opération Matricielle	1302	2.6e-5
Filtre Gaussien	22211	4.4e-4
RGB2GRAY	114796	2.3e-3
Histogramme Égalisation	20905952	0.42

# Partage de la mémoire

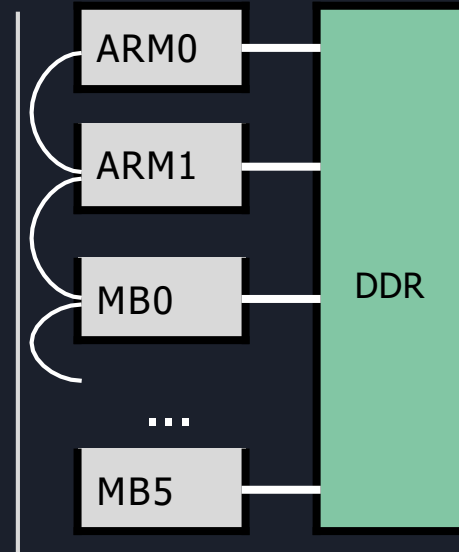
Applications  
Asynchrones

Execution Précédente



Synchronisation de la mémoire

Synchronisation  
des Applications

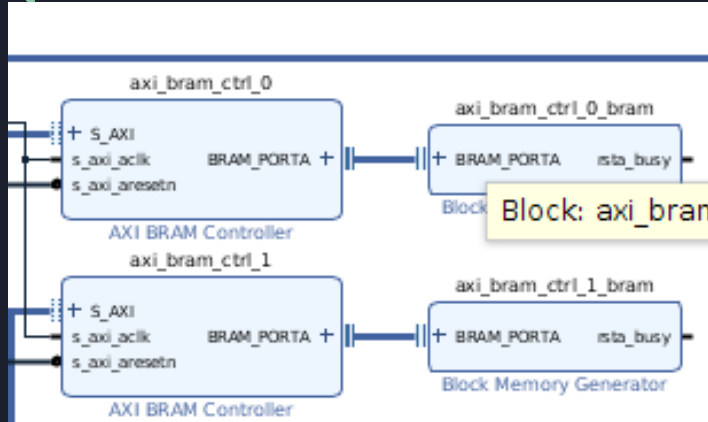


- Restriction Mémoire et Puissance de chaque cœur
- Synchronisation et Parallélisme de la carte ?





# Partage de la mémoire



BRAM non vues par le Linker et communes à tous les processeurs  
 => Mise en dur de données possibles : accès PS/PL par les blocs mémoire pour traiter les données

Adressage et zones disponibles

axi_bram_ctrl_0	S_AXI	Mem0	0x4000_0000	8K	▼	0x4000_1FFF
axi_bram_ctrl_1	S_AXI_1	Mem0	0x4200_0000	8K	▼	0x4200_1FFF

# Partage de la mémoire : BRAM

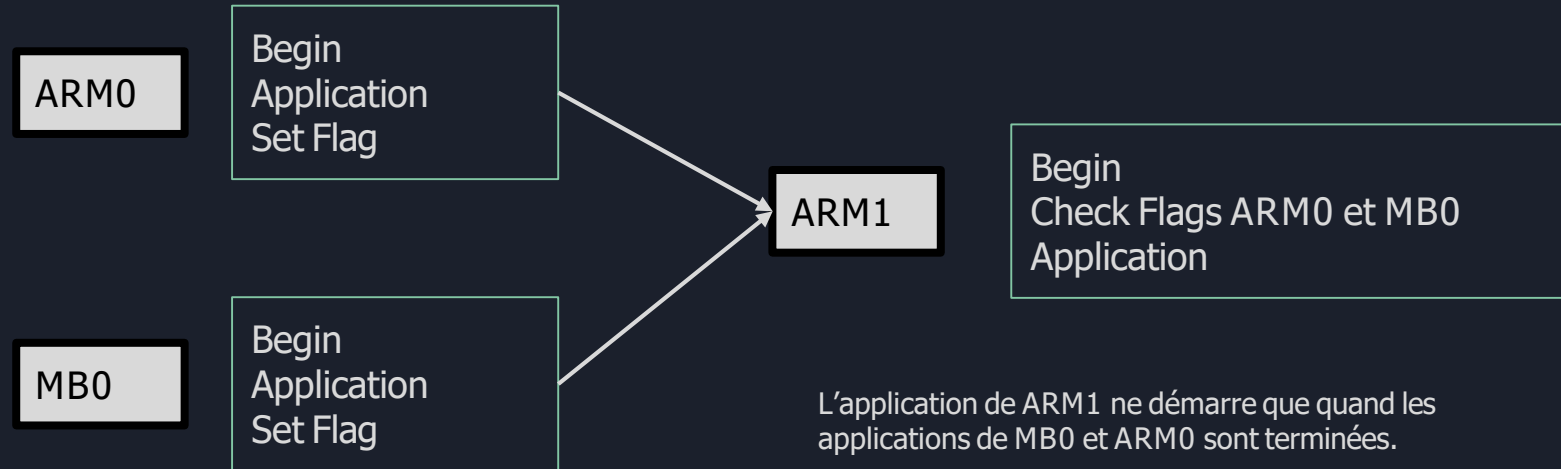
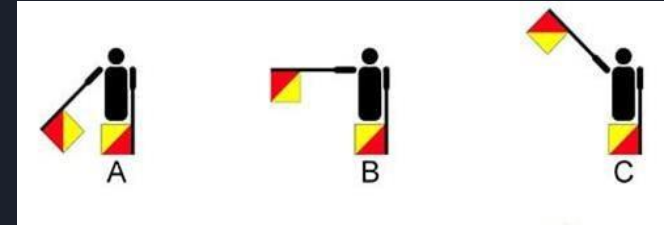
```
//Test écriture en mémoire BRAM
xil_printf("simple value test %d \r\n", XBram_ReadReg(XPAR_BRAM_0_BASEADDR + 12, LOC_BRAM_SEMAPHORE));
```

```
//Test écriture en mémoire BRAM
int a = 56789 ;
XBram_WriteReg(XPAR_BRAM_0_BASEADDR + 12, LOC_BRAM_SEMAPHORE, a);
xil_printf("(ARM0) writing a simple value test %d \r\n", a);
```

```
(ARM0) Hello World
(ARM0) writing a simple value test 56789
ARM FLAG 12345678
ARM up stop 0
ARM0 : fin
(ARM1) Hello World
(ARM1) starting to synchronize
(ARM1) Do something
simple value test 56789
ARM1 : Fin Synchro
```

# Synchronisation par les sémaphores

**Sémaphore** : Flag que tous les processeurs vont voir - stocké dans les BRAM



L'application de ARM1 ne démarre que quand les applications de MB0 et ARM0 sont terminées.

# Synchronisation : Code et Exemple

## ARM0

```
//signals the end of ARM app
dBram = XBram_ReadReg(XPAR_BRAM_0_BASEADDR, LOC_BRAM_SEMAPHORE);
XGpio_DiscreteWrite(&Gpio, LED_CHANNEL, led); //turn off the LED for ARM0
printf("ARM FLAG %8x\r\n", dBram);
if (dBram == FLAG_SEMAPHORE_START){
    XBram_WriteReg(XPAR_BRAM_0_BASEADDR, LOC_BRAM_SEMAPHORE, FLAG_SEMAPHORE_STOP_0);
    printf("ARM up stop 0 \r\n");
}
else if (dBram == FLAG_SEMAPHORE_STOP_0){
    XBram_WriteReg(XPAR_BRAM_0_BASEADDR, LOC_BRAM_SEMAPHORE, FLAG_SEMAPHORE_STOP);
    printf("ARM up stop\r\n");
}

printf("ARM0 : fin \r\n");
```

## MB

```
//signals the end of MB app
dBram = XBram_ReadReg(XPAR_BRAM_0_BASEADDR, LOC_BRAM_SEMAPHORE);
printf("MB FLAG %8x\r\n", dBram);
if (dBram == FLAG_SEMAPHORE_START){
    XBram_WriteReg(XPAR_BRAM_0_BASEADDR, LOC_BRAM_SEMAPHORE, FLAG_SEMAPHORE_STOP_0);
    printf("MB up stop 0 \r\n");
}
else if (dBram == FLAG_SEMAPHORE_STOP_0){
    XBram_WriteReg(XPAR_BRAM_0_BASEADDR, LOC_BRAM_SEMAPHORE, FLAG_SEMAPHORE_STOP);
    printf("MB up stop\r\n");
}
}
```

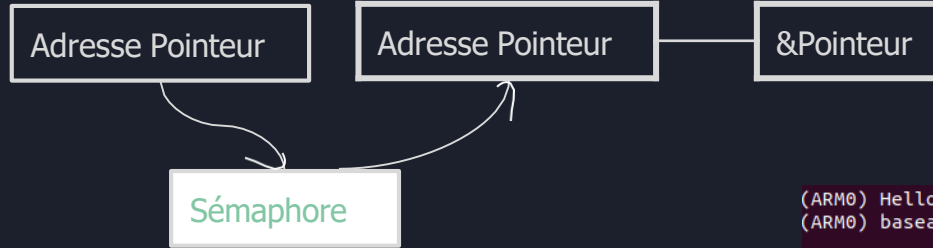
## ARM1

```
printf("(ARM1) starting to synchronize\n");
dBram = XBram_ReadReg(XPAR_BRAM_0_BASEADDR, LOC_BRAM_SEMAPHORE);
do {
    //printf("Loop From ARM1\n\r");
    dBram = XBram_ReadReg(XPAR_BRAM_0_BASEADDR, LOC_BRAM_SEMAPHORE);
    XGpio_DiscreteWrite(&Gpio, LED_CHANNEL, led);
} while(dBram != FLAG_SEMAPHORE_STOP);
// wait for ARM0 and MB to finish
```

Lien Vidéo  
mySleep(30000000);

# Partage de la mémoire : Accès DDR

Adresse	Valeur
1599	---
1600	18
1601	
1602	
1603	
1604	
1604	---



```

//MISE EN DDR MATRICE 2*2
//config outside synchro//
int* baseaddr_p = NULL ;
baseaddr_p = malloc(4 * sizeof(int));

int A[2][2] = {4,3,7,9};
int i=0;
int j=0;
int k=0;
for (i=0; i<2; i++)
{
    for (j=0; j<2; j++)
    {
        //write to the address
        *(baseaddr_p+k)=(int)(A[i][j]);
        xil_printf("(ARM0) baseaddr %d \n", *(baseaddr_p+k));
        xil_printf("(ARM0 addr) %8x \r\n", (baseaddr_p+k));
        k++;
    }
}

///write ptr in the BRAM
XBrAm_WriteReg(XPAR_BRAM_0_BASEADDR + 36, LOC_BRAM_SEMAPHORE, baseaddr_p);

free(baseaddr_p);
  
```

```

//LECTURE MATRICE 2*2
int* baseaddr_p ;
baseaddr_p = (int*) XBrAm_ReadReg(XPAR_BRAM_0_BASEADDR + 36, LOC_BRAM_SEMAPHORE);

int k;
k=0;
for (k=0; k<4; k++)
{
    xil_printf("(ARM1) baseaddr %d \n", *(baseaddr_p+k));
    xil_printf("(ARM1 addr) %8x \r\n", (baseaddr_p+k));
}
  
```

```

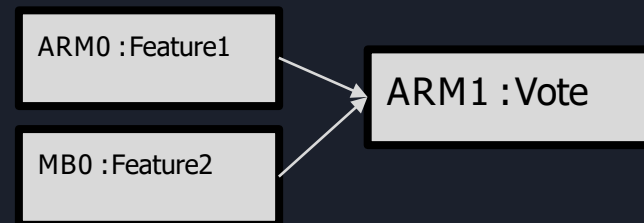
(ARM0) Hello World
(ARM0) baseaddr 4
(ARM0) baseaddr 3 (ARM0 addr) 114398
(ARM0) baseaddr 7 (ARM0 addr) 11439C
(ARM0) baseaddr 9 (ARM0 addr) 1143A0
(ARM0) baseaddr 9 (ARM0 addr) 1143A4
ARM FLAG 12345678
ARM up stop 0
ARM0 : fin
(ARM1) Hello World
(ARM1) starting to synchronize
(ARM1) Do something
(ARM1) baseaddr 4 (ARM1 addr) 114398
(ARM1) baseaddr 3 (ARM1 addr) 11439C
(ARM1) baseaddr 7 (ARM1 addr) 1143A0
(ARM1) baseaddr 9 (ARM1 addr) 1143A4
ARM1 : Fin Synchro
  
```

# Applications :

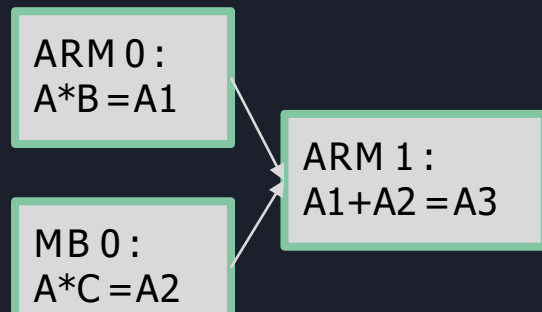
**Parallélisation** : laisser ARM0/MB0 traiter une partie de la matrice et donner le résultat à ARM1

**Calcul** : traiter une image à différentes échelles ( ex : SIFT... )

**Flot d'Applications** : traiter feature 'forme' par ARM0 ;  
traiter feature 'couleur' par ARM1 ; vote par ARM1



## Exemple d'application



$A = [4, 3, 7, 9]$   
 $B = [1, 1, 1, 2]$   
 $C = [2, 1, 1, 2]$

(ARM1) A3 18  
 (ARM1) A3 20  
 (ARM1) A3 39  
 (ARM1) A3 50

# Améliorations à apporter

- Code des applications à peaufiner dans les flots
- Utilisation de tous les Coeurs trouvés par
- Optimisation des différentes Clock
- Mettre en place des meilleurs accès mémoire
- Boucle While pour des traitements de vidéo



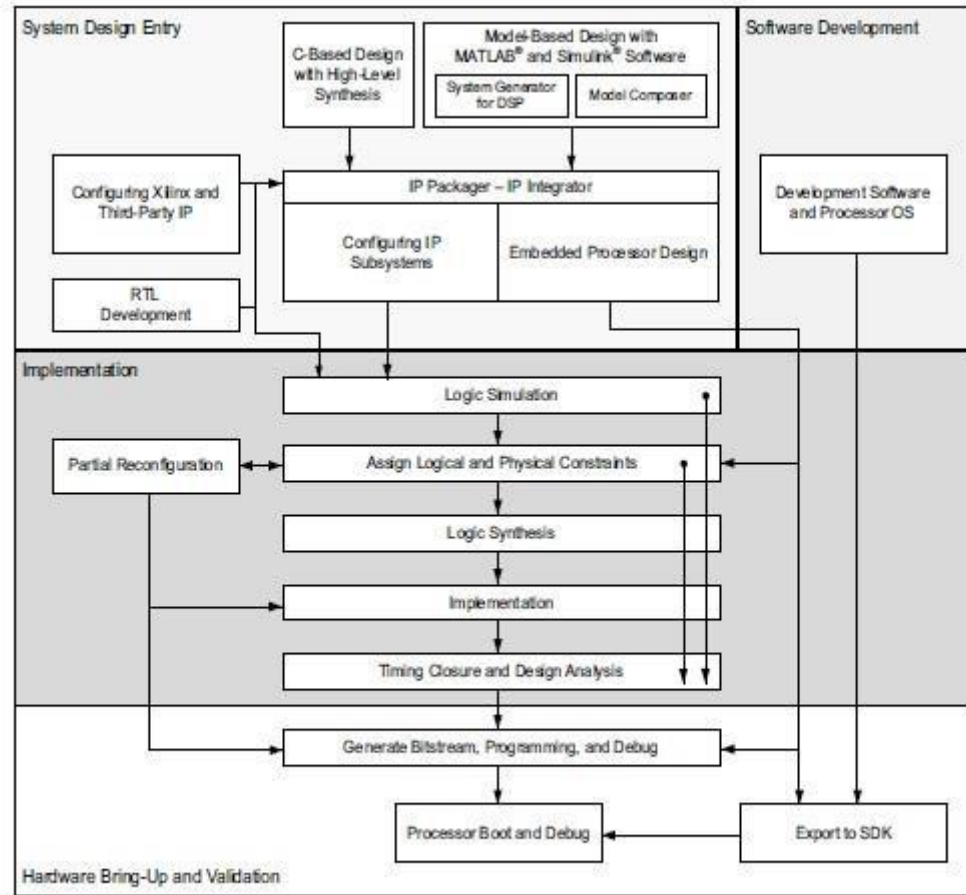


Automatisation exploration options IPs TCL

Automatisation exploration options IPs pilotée par algorithme  
optimisation multicritère

# Automatisation et optimisation

- MPSOC : Soft IPs configurables avec de très nombreuses options
- Combinatoire des configurations de MPSOC très importante
- **Objectif:** Obtention de résultats surface/énergie/temps d'exécution réels (*non estimés*)
- Nécessité de synthétiser placer router et exécuter sur carte pour toutes les configurations
- Temps de SPR important
- Solutions :
- L'objectif dans cette partie est d'optimiser les paramètres des Microblazes :
  - Utilisation faible de ressources
  - Consommation d'énergie faible



# Langage TCL – configuration Soft IP

```
# Create instance: microblaze_1, and set properties
set microblaze_1 [ create_bd_cell -type ip -vlnv xilinx.com:ip:microblaze:11.0 microblaze_1 ]
set_property -dict [ list \
    CONFIG.C_ADDR_TAG_BITS {15} \
    CONFIG.C_AREA_OPTIMIZED {1} \
    CONFIG.C_CACHE_BYTE_SIZE {64} \
    CONFIG.C_DCACHE_ADDR_TAG {15} \
    CONFIG.C_DCACHE_BASEADDR {0x0000000010000000} \
    CONFIG.C_DCACHE_BYTE_SIZE {128} \
    CONFIG.C_DCACHE_HIGHADDR {0x000000001FFFFFFF} \
    CONFIG.C_DEBUG_ENABLED {1} \
    CONFIG.C_D_AXI {1} \
    CONFIG.C_D_LMB {1} \
    CONFIG.C_ICACHE_BASEADDR {0x0000000010000000} \
    CONFIG.C_ICACHE_HIGHADDR {0x000000001FFFFFFF} \
    CONFIG.C_I_LMB {1} \
    CONFIG.C_USE_DCACHE {1} \
    CONFIG.C_USE_ICACHE {1} \
] $microblaze_1
```

# Langage TCL

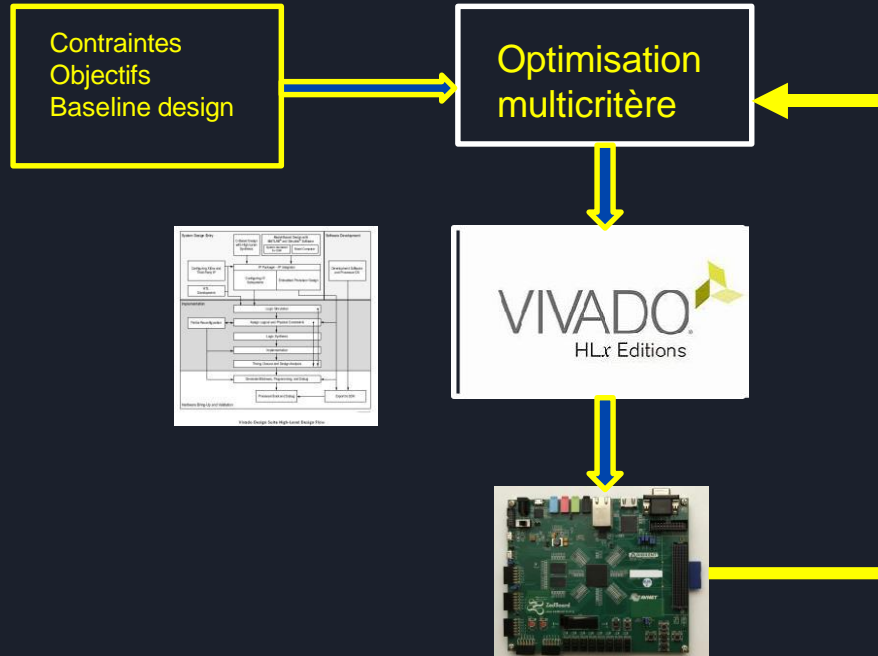
```
generate_target all [get_files /auto/a/ahammami/Bureau/vivado/myproj/project_1.srcs/sources_1/bd/design_1/design_1.bd]
make_wrapper -files [get_files /auto/a/ahammami/Bureau/vivado/myproj/project_1.srcs/sources_1/bd/design_1/design_1.bd] -top
add_files -norecurse /auto/a/ahammami/Bureau/vivado/myproj/project_1.srcs/sources_1/bd/design_1/hdl/design_1_wrapper.v
synth_ip [get_files *.xci]
synth_design -top design_1_wrapper

report_utilization -file ./reports/uti_512_128_64_128.txt
report_power -file ./reports/power_512_128_64_128.txt
```

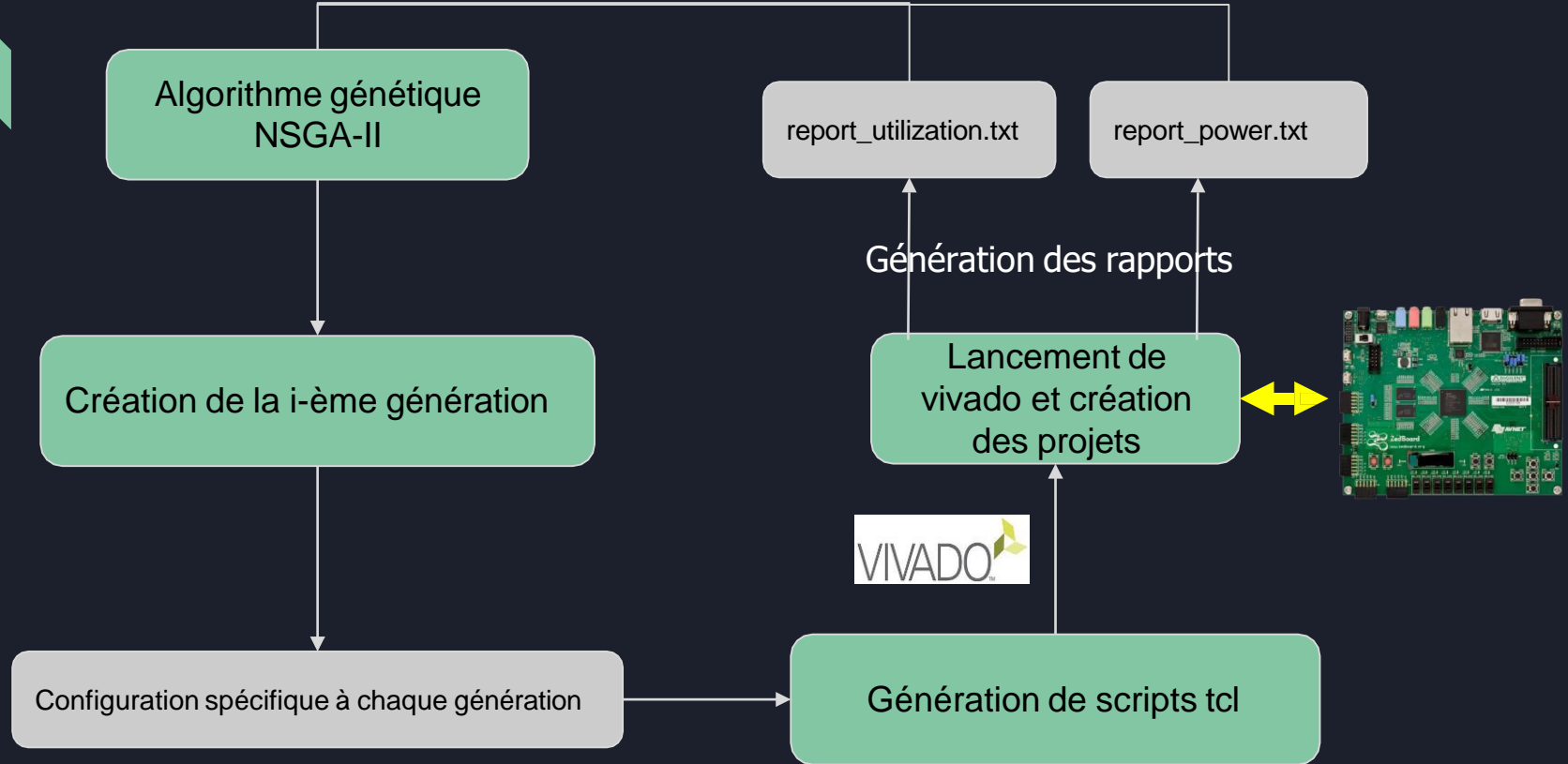
## References

1. [Vivado Design Suite User Guide Using Tcl Scripting](#)
2. [Vivado Design Suite Tcl Command Reference Guide](#)

# Fully Automated design flow v.1 (POC)



# Principe général



# Création d'autres block design à partir d'un modèle

```
# Create instance: microblaze_0, and set properties
set microblaze_0 [ create_bd_cell -type ip -vlnv xilinx.com:ip:microblaze:11.0 microblaze_0 ]
set_property -dict [ list \
  CONFIG.C_ADDR_TAG_BITS {15} \
  CONFIG.C_AREA_OPTIMIZED {1} \
  CONFIG.C_CACHE_BYTE_SIZE {rob307ic1} \
  CONFIG.C_DCACHE_ADDR_TAG {15} \
  CONFIG.C_DCACHE_BASEADDR {0x0000000010000000} \
  CONFIG.C_DCACHE_BYTE_SIZE {rob307dc1} \
  CONFIG.C_DCACHE_HIGHADDR {0x000000001FFFFFFFF} \
  CONFIG.C_DEBUG_ENABLED {1} \
  CONFIG.C_D_AXI {1} \
  CONFIG.C_D_LMB {1} \
  CONFIG.C_ICACHE_BASEADDR {0x0000000010000000} \
  CONFIG.C_ICACHE_HIGHADDR {0x000000001FFFFFFFF} \
  CONFIG.C_I_LMB {1} \
  CONFIG.C_USE_DCACHE {1} \
  CONFIG.C_USE_ICACHE {1} \
] $microblaze_0
```



# DSE-TCL.c

```
while ((fgets(buffer, BUFFER_SIZE, fPtr)) != NULL)
{
    // Replace all occurrence of word from current line

    replaceAll(buffer, ref_ic1, ic1[j]);
    replaceAll(buffer, ref_ic2, ic2[j]);
    replaceAll(buffer, ref_dc1, dc1[j]);
    replaceAll(buffer, ref_dc2, dc2[j]);
    replaceAll(buffer, ref_r, d_file_name); //for me
    replaceAll(buffer, ref_d, d_file_name);
    fputs(buffer, fTemp);
}
```

```
printf("run%d", j);
sprintf(command, "ssh ahammami@%s.ensta.fr '/auto/appy/ensta/pack/xilinx/vivado/2019.1/Vivado/2019.1/bin/vivado
    -mode batch -source CONFIG-TCL_%s_%d.tcl' &", available_computers[j], report, j);





















system(command);
```

# Fichiers TCL et Projets vivado créés pour 10 individus

- myproj\_\_2048\_4096\_16384\_32768\_4
- myproj\_\_2048\_65536\_65536\_16384\_6
- myproj\_\_32768\_2048\_512\_8192\_7
- myproj\_\_32768\_32768\_16384\_32768\_9
- myproj\_\_32768\_4096\_65536\_65536\_5
- myproj\_\_4096\_1024\_1024\_2048\_8
- myproj\_\_512\_2048\_1024\_4096\_0
- myproj\_\_512\_2048\_65536\_4096\_2
- myproj\_\_65536\_16384\_32768\_2048\_1
- myproj\_\_8192\_1024\_1024\_32768\_3
- Public
- reports

CONFIG-TCL_2048_4096_16384_32768_4.tcl	53
CONFIG-TCL_2048_65536_65536_16384_6.tcl	53
CONFIG-TCL_32768_2048_512_8192_7.tcl	53
CONFIG-TCL_32768_32768_16384_32768_9.tcl	53
CONFIG-TCL_32768_4096_65536_65536_5.tcl	53
CONFIG-TCL_4096_1024_1024_2048_8.tcl	53
CONFIG-TCL_512_2048_1024_4096_0.tcl	53
CONFIG-TCL_512_2048_65536_4096_2.tcl	53
CONFIG-TCL_65536_16384_32768_2048_1.tcl	53
CONFIG-TCL_8192_1024_1024_32768_3.tcl	53

# Rapports créés pour une génération de 10 individus

 power_2048_4096_16384_32768_4.txt	11
 power_2048_65536_65536_16384_6.txt	11
 power_32768_2048_512_8192_7.txt	11
 power_32768_32768_16384_32768_9.txt	11
 power_32768_4096_65536_65536_5.txt	11
 power_4096_1024_1024_2048_8.txt	11
 power_512_2048_1024_4096_0.txt	11
 power_512_2048_65536_4096_2.txt	11
 power_65536_16384_32768_2048_1.txt	11
 power_8192_1024_1024_32768_3.txt	11
 uti_2048_4096_16384_32768_4.txt	9
 uti_2048_65536_65536_16384_6.txt	9
 uti_32768_2048_512_8192_7.txt	9
 uti_32768_32768_16384_32768_9.txt	9
 uti_32768_4096_65536_65536_5.txt	9
 uti_4096_1024_1024_2048_8.txt	9
 uti_512_2048_1024_4096_0.txt	9
 uti_512_2048_65536_4096_2.txt	9
 uti_65536_16384_32768_2048_1.txt	9
 uti_8192_1024_1024_32768_3.txt	9

# Parallélisation de l'exécution des projets sous plusieurs machines de l'ENSTA

- Création d'une liste d'ordinateur disponible à l'ENSTA
- Modification du droit d'accès sécurisé pour ne pas demander à l'algorithme un mot de passe pour chaque commande.

Ces machines sont disponibles dans la salle *spock* :

```
char available_computers[11][256] = { "spock01", "spock03", "spock04", "spock06",  
"spock07", "spock10", "spock13", "spock15", "spock16", "spock18", "spock20"};
```

Commande ssh pour lancer la génération des projets vivado sous plusieurs machine :

```
printf("run%d", j);  
sprintf(command, "ssh ahammami@%s.ensta.fr '/auto/appy/ensta/pack/xilinx/vivado/2019.1/Vivado/2019.1/bin/vivado  
-mode batch -source CONFIG-TCL_%s_%d.tcl' &", available_computers[j], report, j);  
  
system(command);
```

# Résultat obtenu

Enregistrement des index de chaque individu qui présente un résultat optimal.

Parmi les configurations optimales : IC1 = 512 , DC1 = 2048, IC2 = 512, DC2 = 4096

```
-----
Population at generation no. -->5
-----
Generation No.      ->5
-----
variables (real 0 binary 4)  fitness (2)  constraint (
-----
0.000000 2.000000 0.000000 3.000000    17.9200    1.8210
0.000000 3.000000    17.920000    1.821000 1    1.00e+01
-----
0.000000 0.000000 0.000000 3.000000    18.1900    1.8170
0.000000 3.000000    18.190001    1.817000 1    1.00e+01
-----
```

# Points forts de notre solution

- 100 % automatisé
- Exécution sur réseau de PCs à distance pour augmenter le parallélisme du flot
- Réduction importante dans le temps d'exécution de recherche de solutions optimales (à titre d'exemple on doit attendre 2 ans si on parcourt toutes les configurations possibles) .
- Flexibilité : on peut ajouter d'autres configurations à optimiser : synthesis, place and route options, power options



# Questions ?