

Explorations Architecturales de Processeurs Endochrones *KeyRing*

Mickaël Fiorentino

Directeur: Yvon Savaria, Co-directeur: Claude Thibeault

Thèse de doctorat

17 Décembre 2020



1 Introduction

2 Microarchitecture KeyRing

3 Processeurs KeyV

4 Conclusion

1 Introduction

2 Microarchitecture KeyRing

3 Processeurs KeyV

4 Conclusion

Contexte

- Performances des microprocesseurs limitées par leur puissance
- Objectif : **efficacité énergétique**

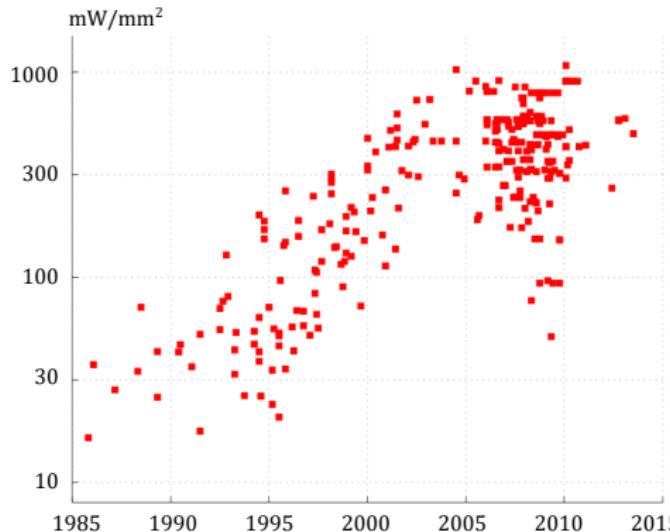


Figure – Évolution de la densité de puissance des microprocesseurs [1]

Motivations

Le processeur AnARM ([Octasic](#)) utilise une *microarchitecture efficace en énergie* basée sur une technique de conception *endochrone* originale.

→ *Cette efficacité est-elle due à la microarchitecture endochrone ?*

Motivations

Le processeur AnARM ([Octasic](#)) utilise une *microarchitecture efficace en énergie* basée sur une technique de conception *endochrone* originale.

→ *Cette efficacité est-elle due à la microarchitecture endochrone ?*

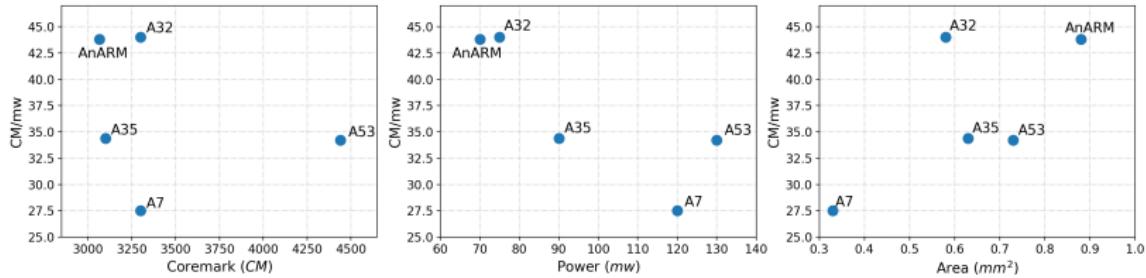


Figure – Comparaison des performances du AnARM (28 nm CMOS) [2]

Motivations

Le processeur AnARM ([Octasic](#)) utilise une *microarchitecture efficace en énergie* basée sur une technique de conception *endochrone* originale.

→ *Cette efficacité est-elle due à la microarchitecture endochrone ?*

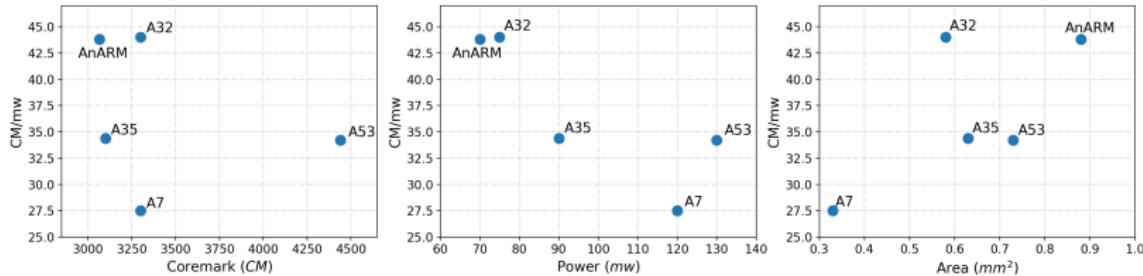
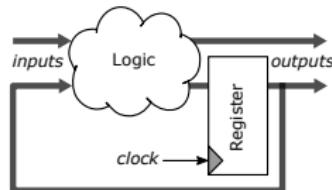


Figure – Comparaison des performances du AnARM (28 nm CMOS) [2]

Paradigmes de conception

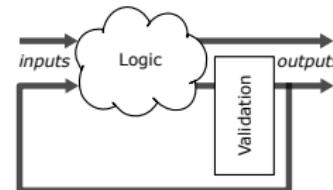
Synchrone

- Synchronisation globale, externe
- Simple, efficace, réutilisable
→ *modèle de délai borné*
- Automatisé (CAO)
→ *conception, vérifications, test*



Endochrone / Asynchrone

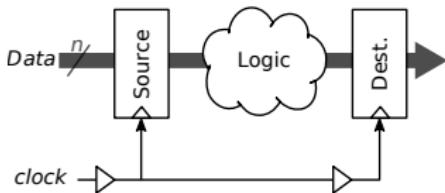
- Synchronisation locale, interne
- Flexible, modulaire, décentralisé
→ *Communication élastique*
- Peu compatible CAO
→ ↴ *productivité & fiabilité*



Paradigmes de conception

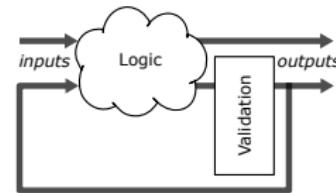
Synchrone

- Synchronisation globale, externe
- Simple, efficace, réutilisable
→ *modèle de délai borné*
- Automatisé (CAO)
→ *conception, vérifications, test*



Endochrone / Asynchrone

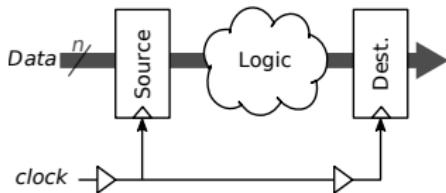
- Synchronisation locale, interne
- Flexible, modulaire, décentralisé
→ *Communication élastique*
- Peu compatible CAO
→ ↘ *productivité & fiabilité*



Paradigmes de conception

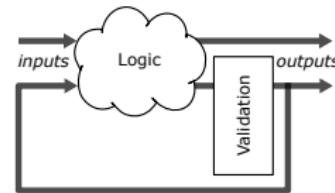
Synchrone

- Synchronisation globale, externe
- Simple, efficace, réutilisable
→ *modèle de délai borné*
- Automatisé (CAO)
→ *conception, vérifications, test*



Endochrone / Asynchrone

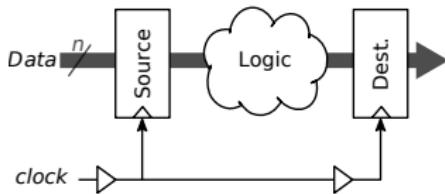
- Synchronisation locale, interne
- Flexible, modulaire, décentralisé
→ *Communication élastique*
- Peu compatible CAO
→ ↘ *productivité & fiabilité*



Paradigmes de conception

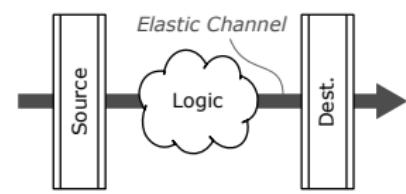
Synchrone

- Synchronisation globale, externe
- Simple, efficace, réutilisable
→ *modèle de délai borné*
- Automatisé (CAO)
→ *conception, vérifications, test*



Endochrone / Asynchrone

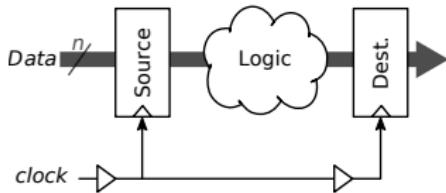
- Synchronisation locale, interne
- Flexible, modulaire, décentralisé
→ *Communication élastique*
- Peu compatible CAO
→ ↴ *productivité & fiabilité*



Paradigmes de conception

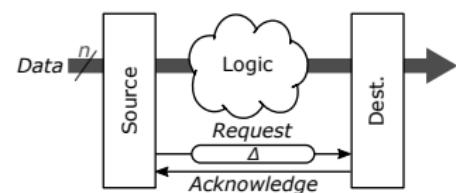
Synchrone

- Synchronisation globale, externe
- Simple, efficace, réutilisable
→ *modèle de délai borné*
- Automatisé (CAO)
→ *conception, vérifications, test*



Endochrone / Asynchrone

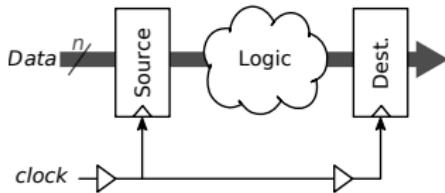
- Synchronisation locale, interne
- Flexible, modulaire, décentralisé
→ *Communication élastique*
- Peu compatible CAO
→ ↴ *productivité & fiabilité*



Paradigmes de conception

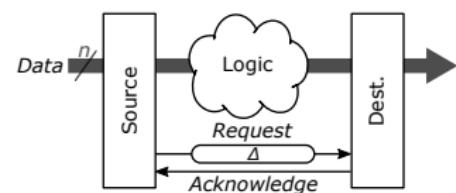
Synchrone

- Synchronisation globale, externe
- Simple, efficace, réutilisable
→ *modèle de délai borné*
- Automatisé (CAO)
→ *conception, vérifications, test*



Endochrone / Asynchrone

- Synchronisation locale, interne
- Flexible, modulaire, décentralisé
→ *Communication élastique*
- Peu compatible CAO
→ ↴ *productivité & fiabilité*



Paradigmes Asynchrones BD

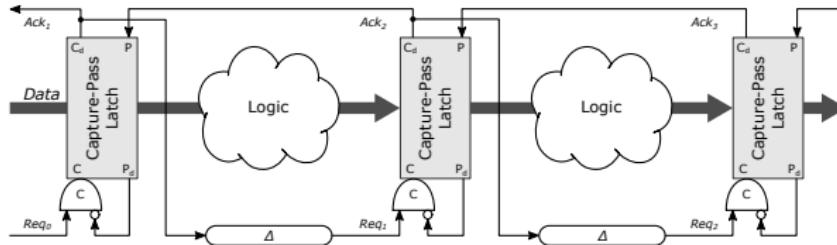


Figure – Micropipeline (1989) [3]

Paradigmes Asynchrones BD

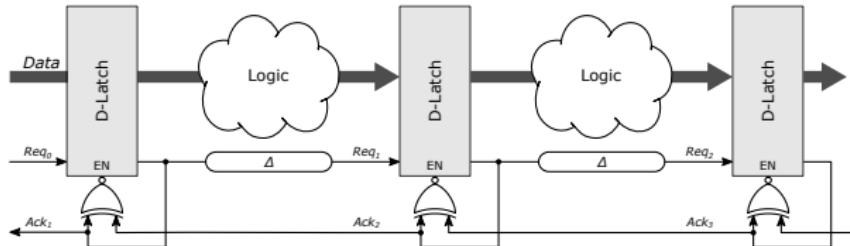


Figure – Mousetrap (2007) [4]

Paradigmes Asynchrones BD

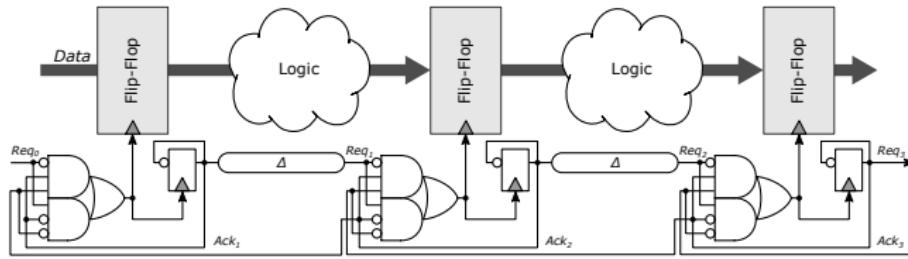


Figure – Click Elements (2010) [5]

Paradigmes Asynchrones BD

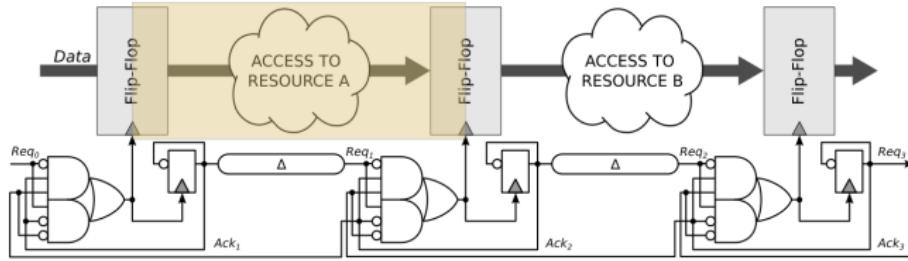


Figure – Click Elements (2010) [5] → ILP

Paradigmes Asynchrones BD

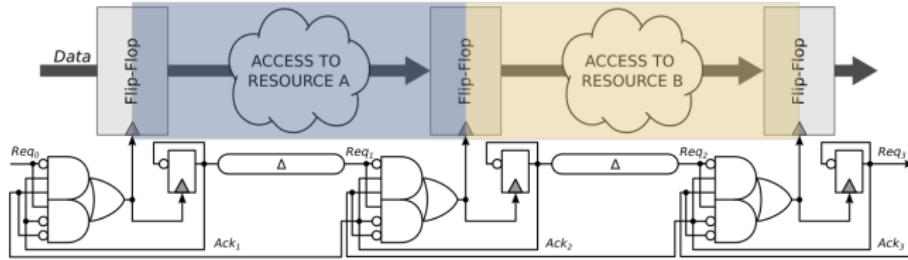


Figure – Click Elements (2010) [5] → ILP

Principes Endochrones du AnARM

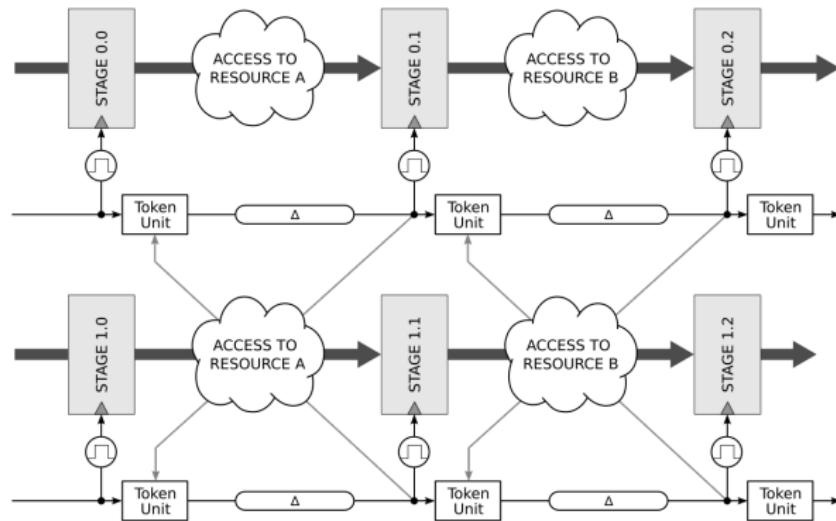


Figure – Principe de pipeline Octasen

Principes Endochrones du AnARM

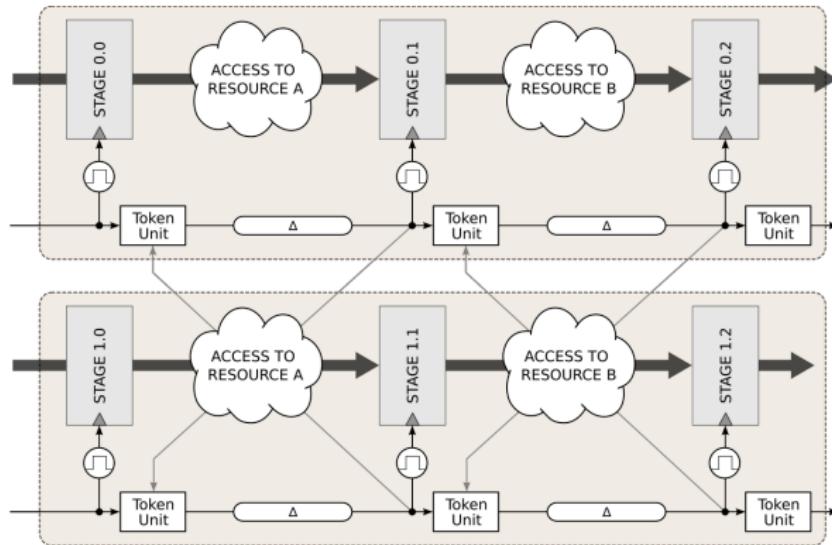


Figure – Principe de pipeline Octasien → EU

Principes Endochrones du AnARM

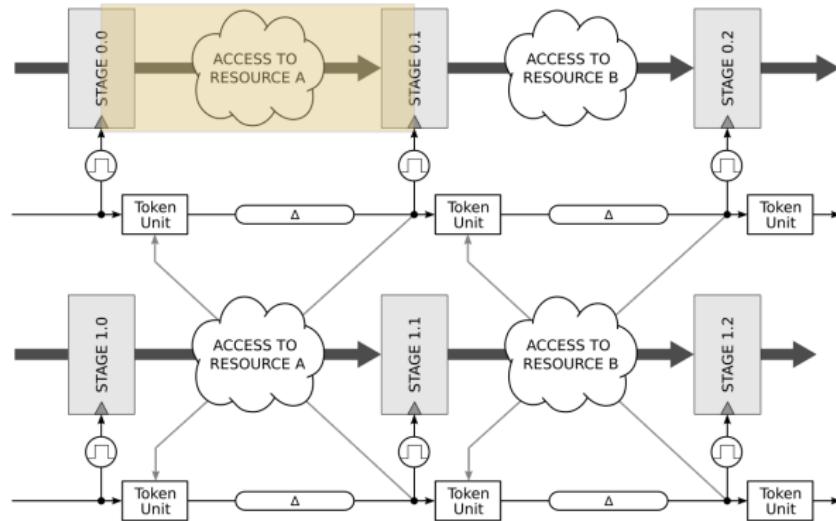


Figure – Principe de pipeline Octasien → ILP

Principes Endochrones du AnARM

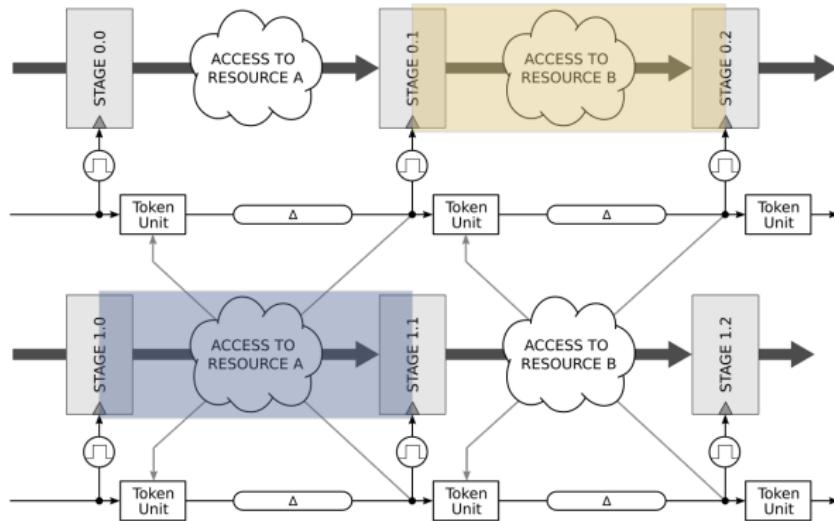


Figure – Principe de pipeline Octasien → ILP

Principes Endochrones du AnARM

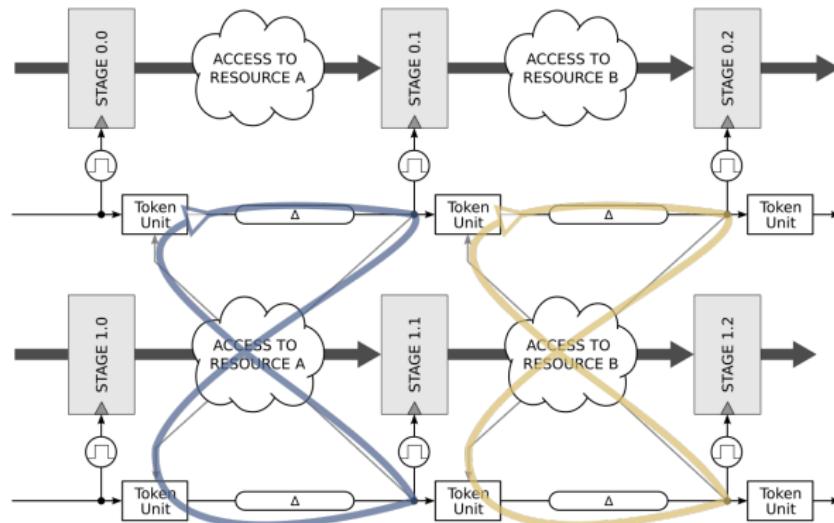


Figure – Principe de pipeline Octasien → token

Principes Endochrones du AnARM

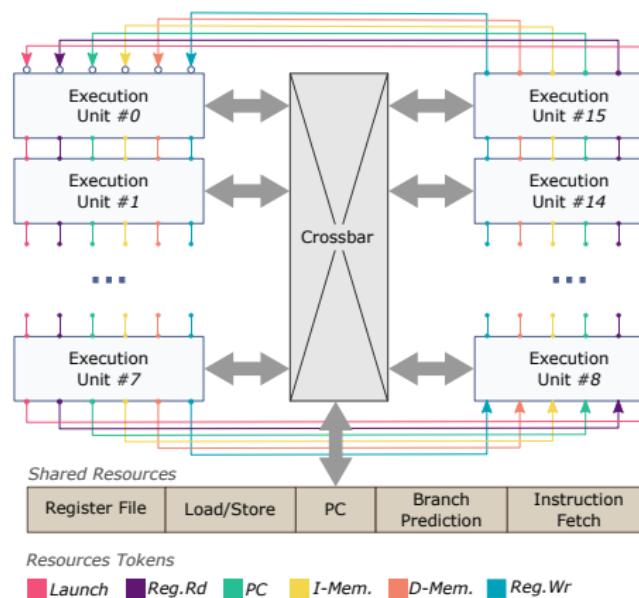


Figure – Microarchitecture du AnARM [6, 7, 8]

Objectifs & Contributions

Exploration architecturale de processeurs endochrones Octasiens

- Définir les principes de base de la microarchitecture
→ *microarchitecture KeyRing*
- Concevoir des processeurs basés sur cette microarchitecture
→ *processeurs KeyV*
- Élaborer un flot de conception adapté à la microarchitecture
→ *synthèse par contraintes temporelles, STA*
- Mettre au point des protocoles expérimentaux pertinents
→ *simulations de benchmarks, analyses de puissance*

Objectifs & Contributions

Exploration architecturale de processeurs endochrones Octasiens

- Définir les principes de base de la microarchitecture
→ *microarchitecture KeyRing*
- Concevoir des processeurs basés sur cette microarchitecture
→ *processeurs KeyV*
- Élaborer un flot de conception adapté à la microarchitecture
→ *synthèse par contraintes temporelles, STA*
- Mettre au point des protocoles expérimentaux pertinents
→ *simulations de benchmarks, analyses de puissance*

Objectifs & Contributions

Exploration architecturale de processeurs endochrones Octasiens

- Définir les principes de base de la microarchitecture
→ *microarchitecture KeyRing*
- Concevoir des processeurs basés sur cette microarchitecture
→ *processeurs KeyV*
- Élaborer un flot de conception adapté à la microarchitecture
→ *synthèse par contraintes temporelles, STA*
- Mettre au point des protocoles expérimentaux pertinents
→ *simulations de benchmarks, analyses de puissance*

Objectifs & Contributions

Exploration architecturale de processeurs endochrones Octasiens

- Définir les principes de base de la microarchitecture
→ *microarchitecture KeyRing*
- Concevoir des processeurs basés sur cette microarchitecture
→ *processeurs KeyV*
- Élaborer un flot de conception adapté à la microarchitecture
→ *synthèse par contraintes temporelles, STA*
- Mettre au point des protocoles expérimentaux pertinents
→ *simulations de benchmarks, analyses de puissance*

Objectifs & Contributions

Exploration architecturale de processeurs endochrones Octasiens

- Définir les principes de base de la microarchitecture
→ *microarchitecture KeyRing*
- Concevoir des processeurs basés sur cette microarchitecture
→ *processeurs KeyV*
- Élaborer un flot de conception adapté à la microarchitecture
→ *synthèse par contraintes temporelles, STA*
- Mettre au point des protocoles expérimentaux pertinents
→ *simulations de benchmarks, analyses de puissance*

1 Introduction

2 Microarchitecture KeyRing

3 Processeurs KeyV

4 Conclusion

1 Introduction

2 Microarchitecture KeyRing

3 Processeurs KeyV

4 Conclusion

Unité d'exécution élémentaire

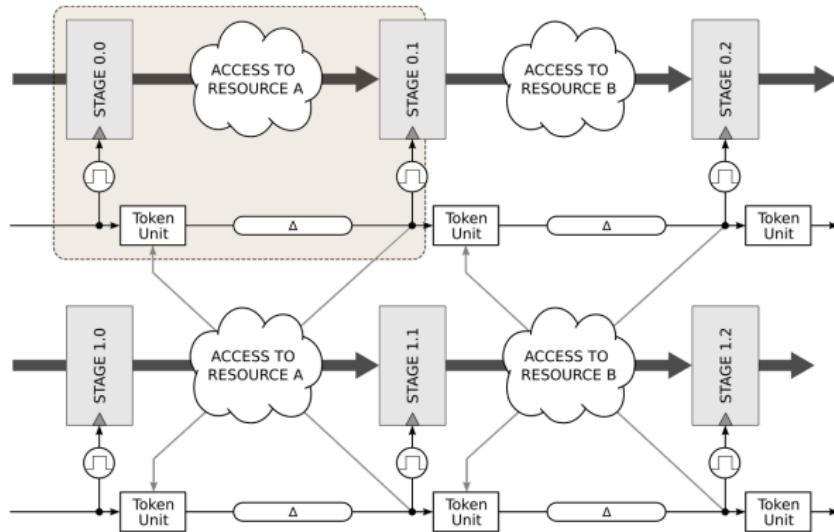


Figure – Principe de pipeline Octasen

Unité d'exécution élémentaire

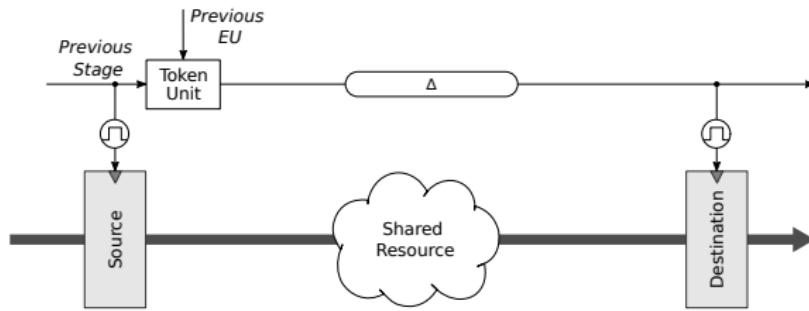


Figure – Étage d'unité d'exécution

Unité d'exécution élémentaire

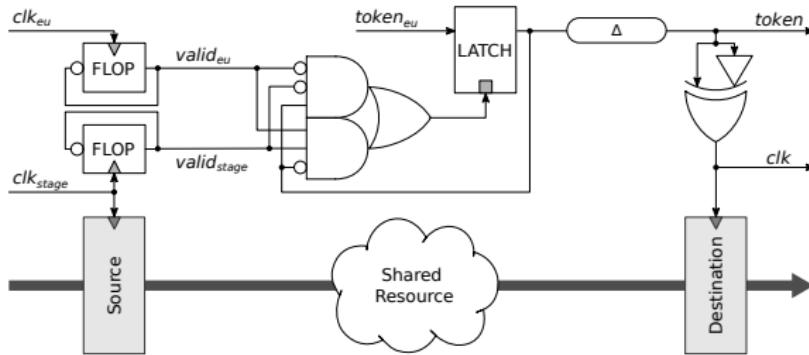


Figure – Étage d'unité d'exécution → Version 1

Unité d'exécution élémentaire

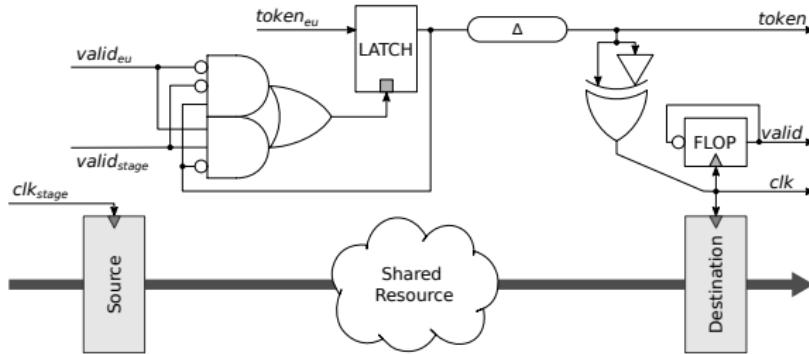


Figure – Étage d'unité d'exécution → Version 2

Unité d'exécution élémentaire

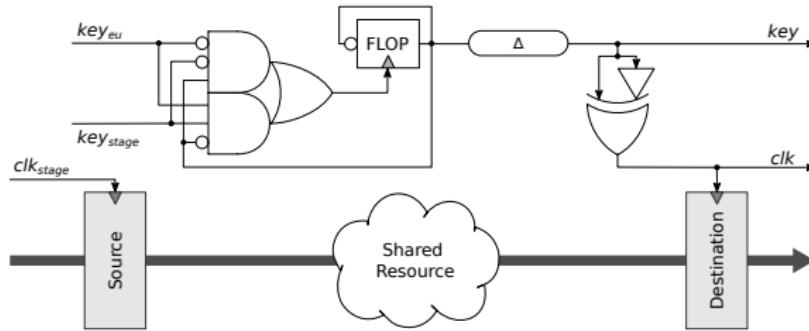


Figure – Étage d'unité d'exécution → Version 3

Unité d'exécution élémentaire

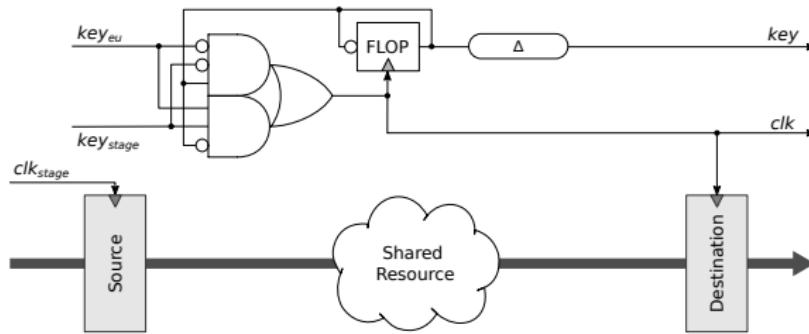


Figure – Étage d'unité d'exécution → Version 4

Unité d'exécution élémentaire

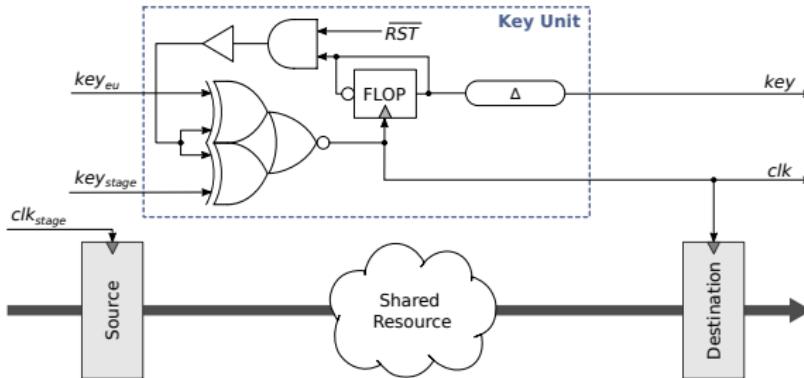


Figure – Étage d'unité d'exécution → Version Finale

Parallélisme d'instruction

$$(e, s) \leftarrow (e, \langle s - 1 \rangle_S), (\langle e - 1 \rangle_E, \langle s + (\alpha - 1) \rangle_S)$$

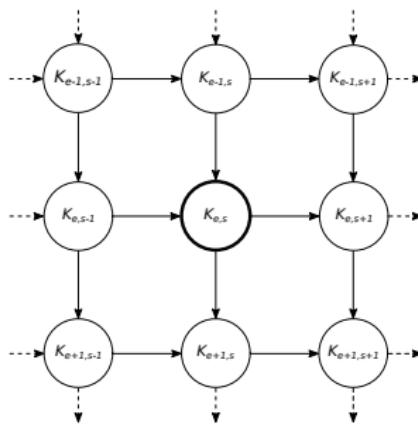


Figure – Organisation des circuits KeyRing

E : Nombre d'unité d'exécution ; S : Nombre d'étages par unité d'exécution ; α : Décalage entre les étages d'unités d'exécution voisines

Parallélisme d'instruction

$$(e, s) \leftarrow (e, \langle s - 1 \rangle_S), (\langle e - 1 \rangle_E, \langle s + (\alpha - 1) \rangle_S)$$

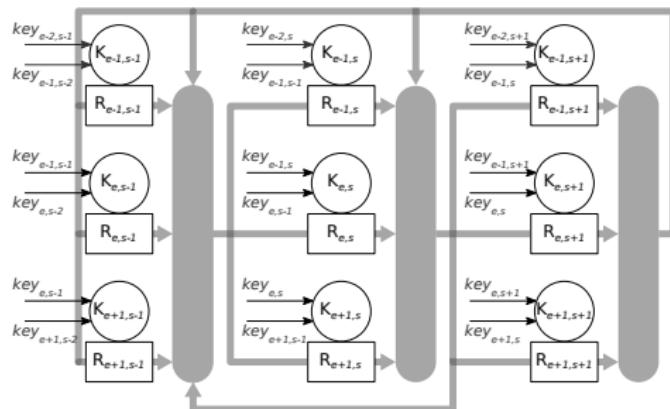


Figure – Circuit KeyRing générique

E : Nombre d'unité d'exécution ; S : Nombre d'étages par unité d'exécution ; α : Décalage entre les étages d'unités d'exécution voisines

Exemple : Tribonacci

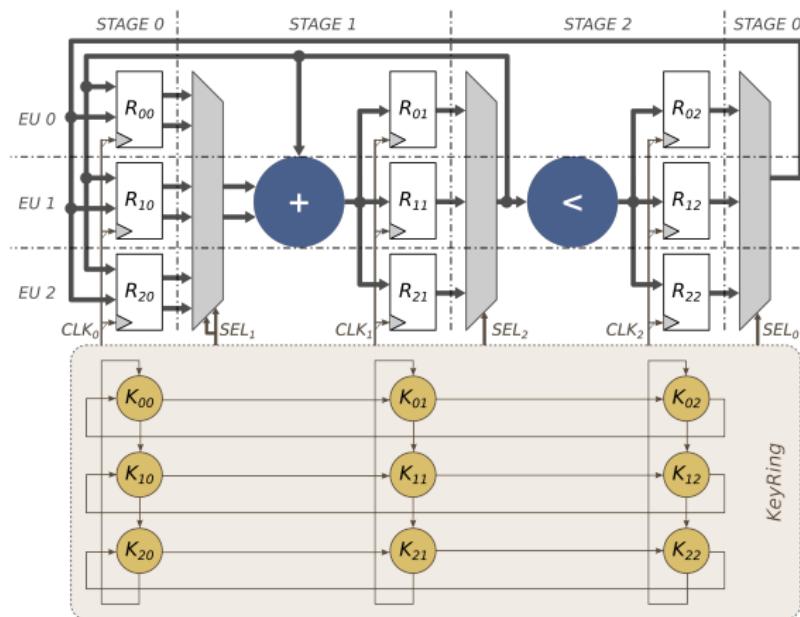


Figure – Circuit séquentiel KeyRing à 3 étages qui implémente la suite Tribonacci

Exemple : Tribonacci

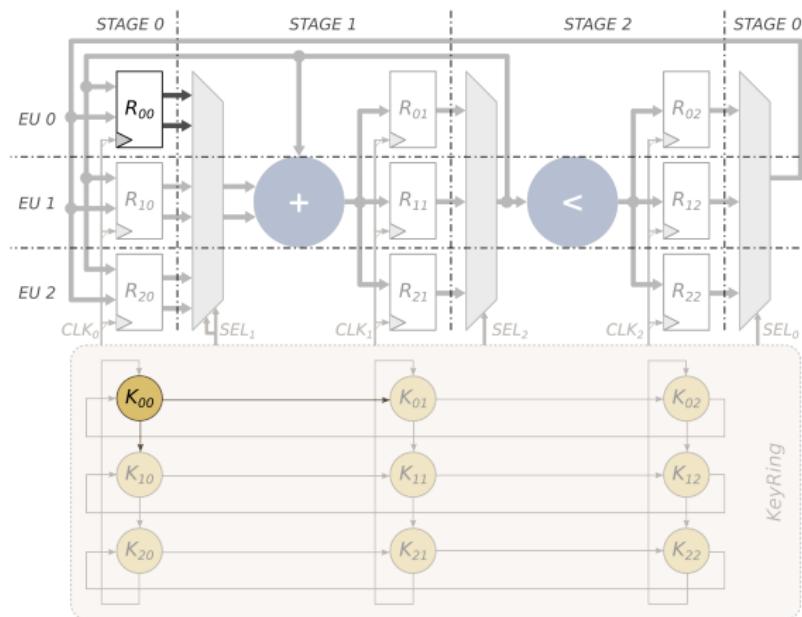


Figure – Circuit séquentiel KeyRing à 3 étages qui implémente la suite Tribonacci

Exemple : Tribonacci

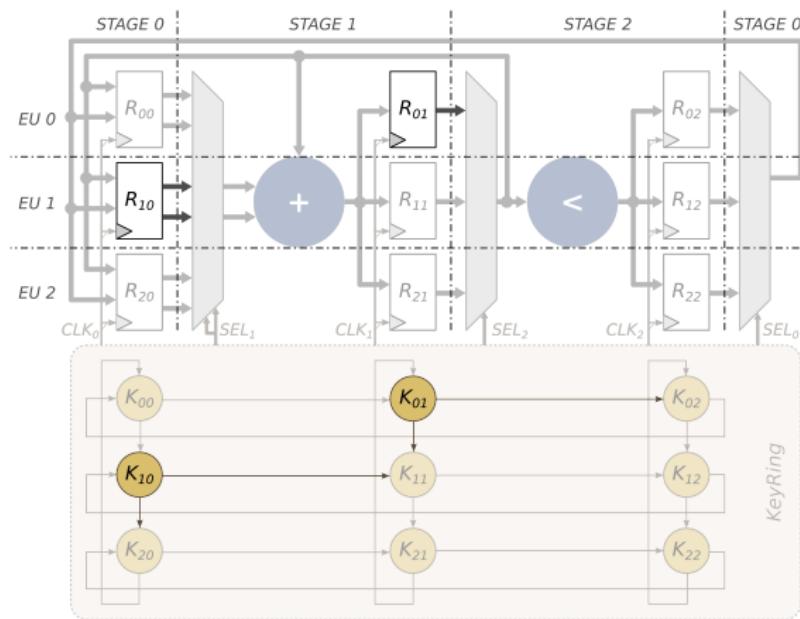


Figure – Circuit séquentiel KeyRing à 3 étages qui implémente la suite Tribonacci

Exemple : Tribonacci

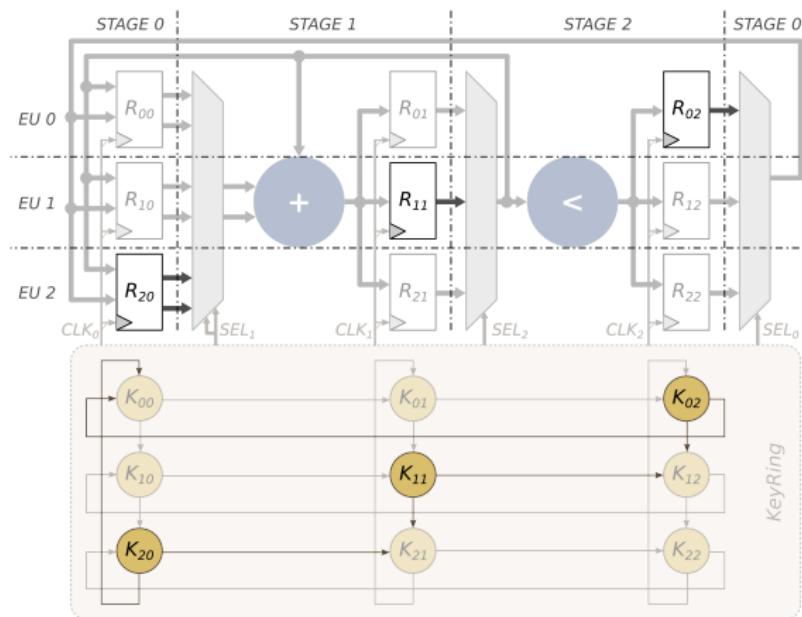


Figure – Circuit séquentiel KeyRing à 3 étages qui implémente la suite Tribonacci

Exemple : Tribonacci

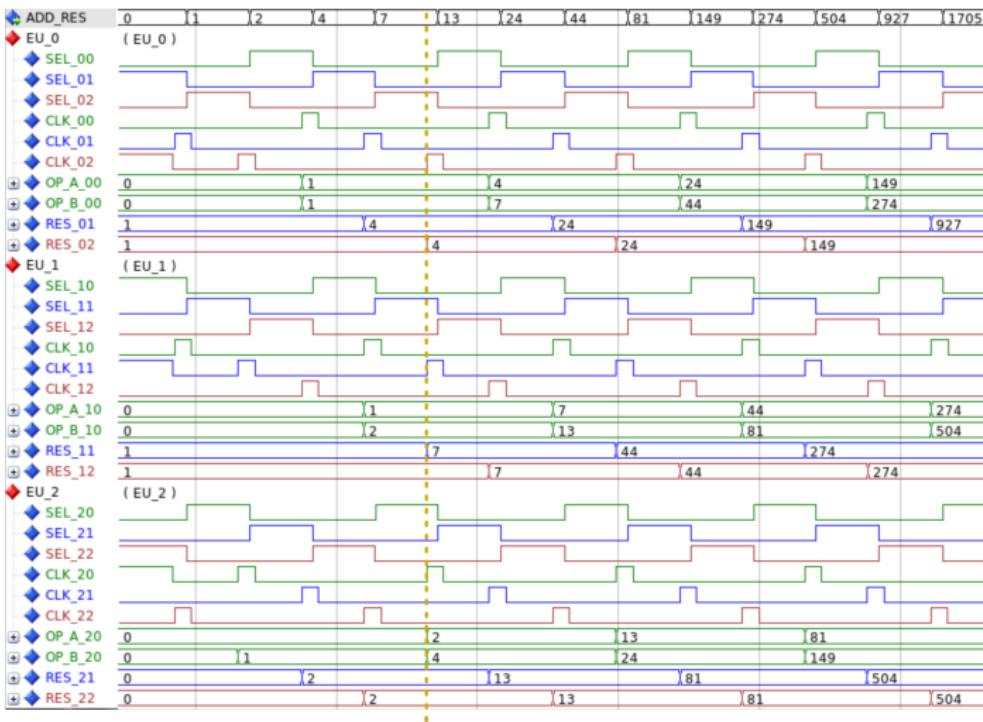


Figure – Chronogramme d'exécution du circuit KeyRing Tribonacci

Exemple : Tribonacci

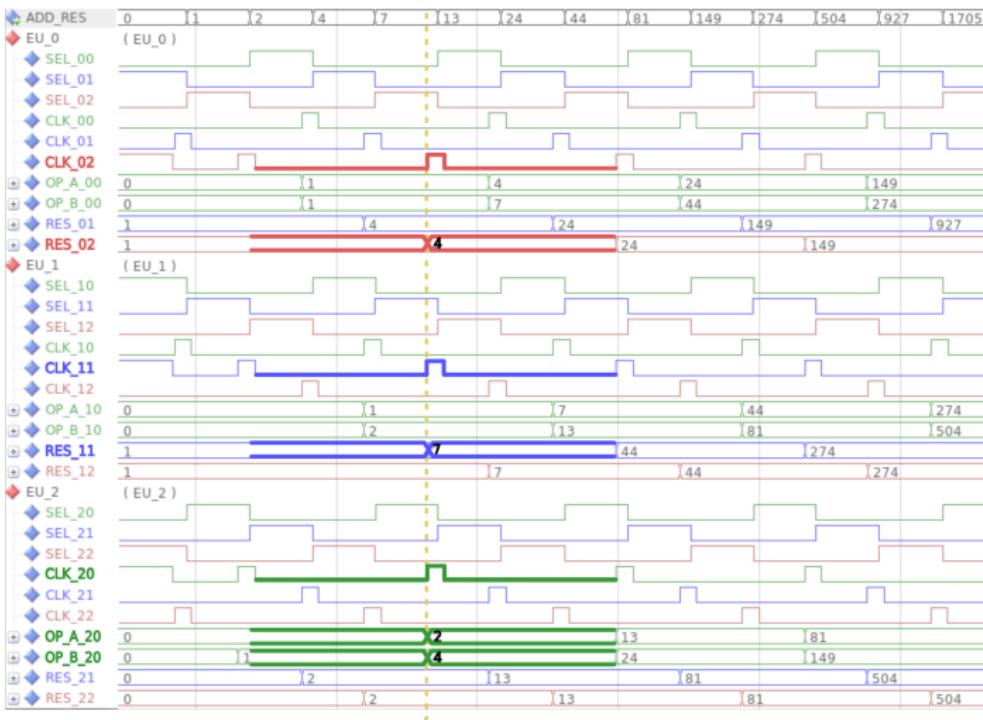


Figure – Chronogramme d'exécution du circuit KeyRing Tribonacci

Mise en évidence des conflits

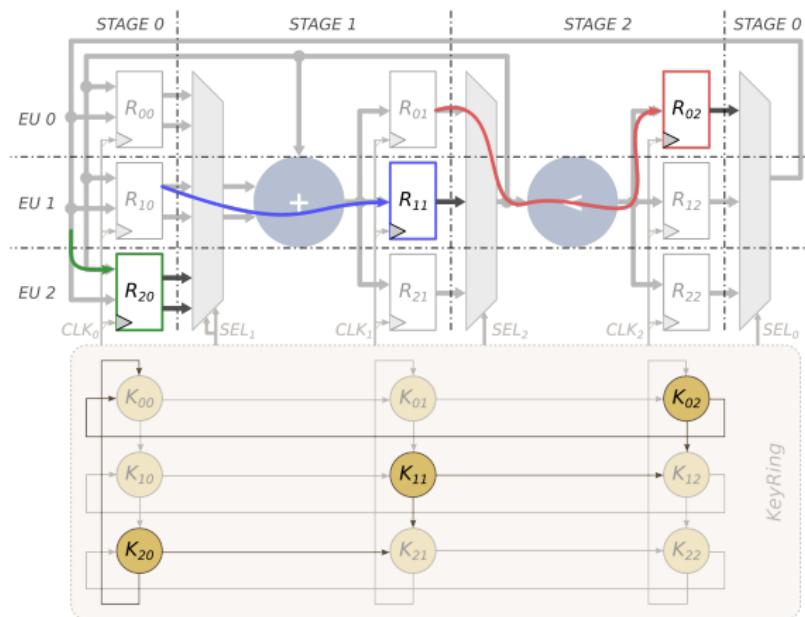


Figure – Circuit Tribonacci sans conflits → Délais identiques

Mise en évidence des conflits

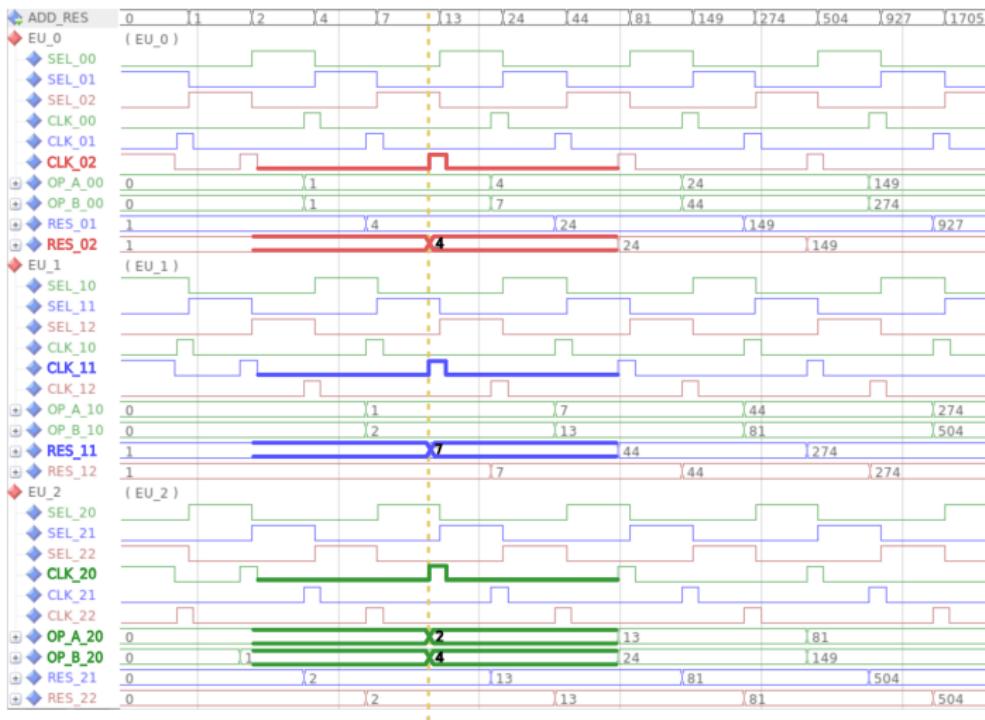


Figure – Chronogramme Tribonacci sans conflits → Délais identiques

Mise en évidence des conflits

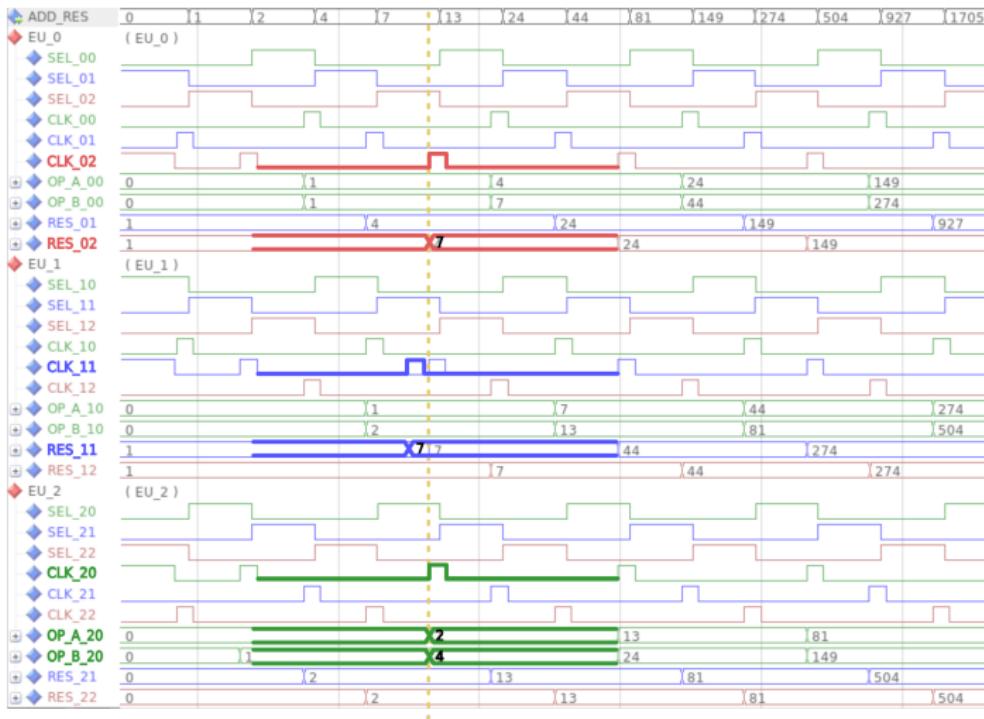


Figure – Chronogramme Tribonacci avec conflit → CLK_11 en avance

Mise en évidence des conflits

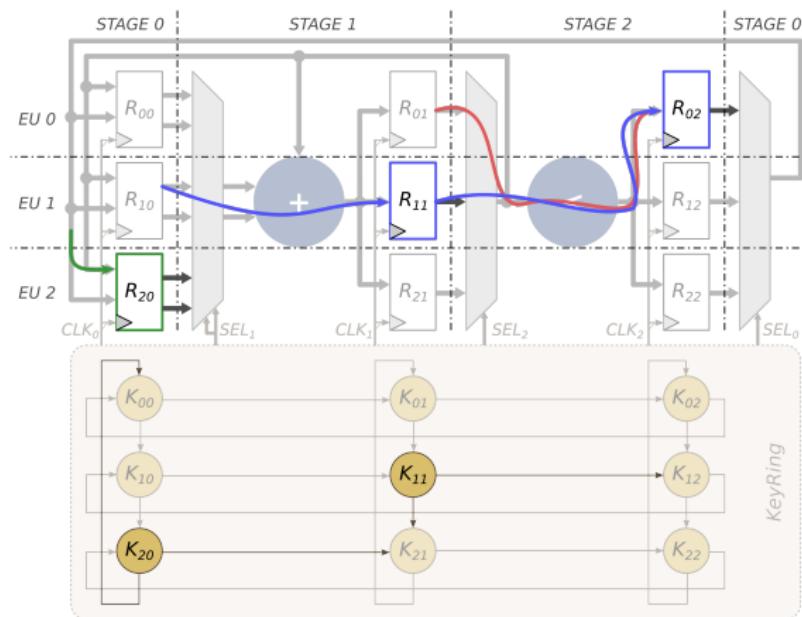


Figure – Circuit *Tribonacci* avec conflit → CLK_{11} en avance

Mise en évidence des conflits

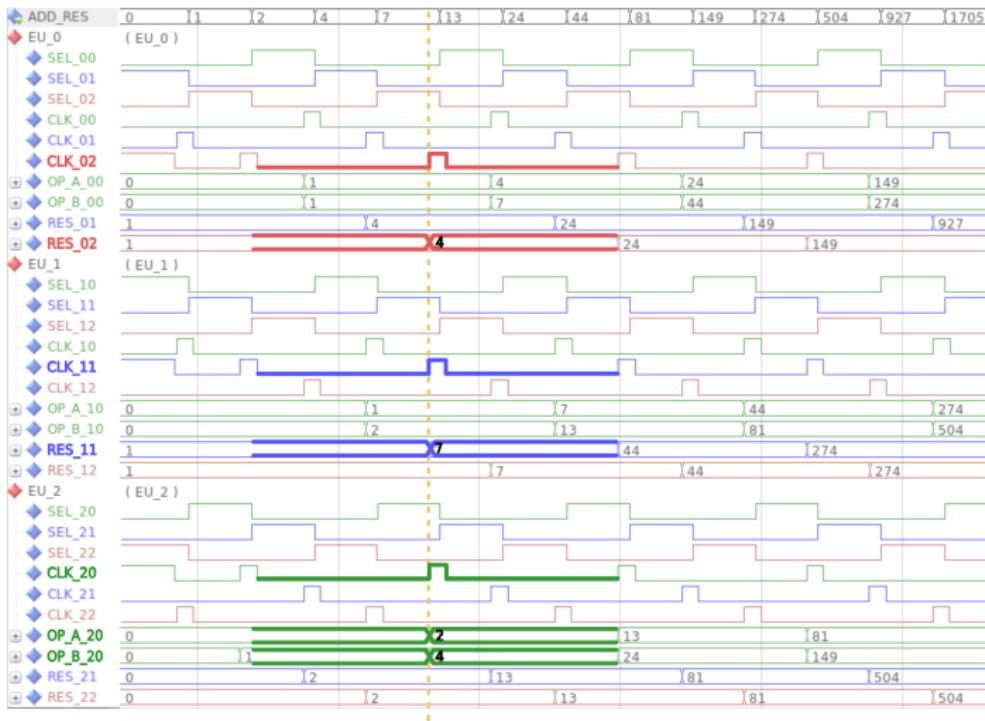


Figure – Chronogramme Tribonacci sans conflits → Délais identiques

Mise en évidence des conflits

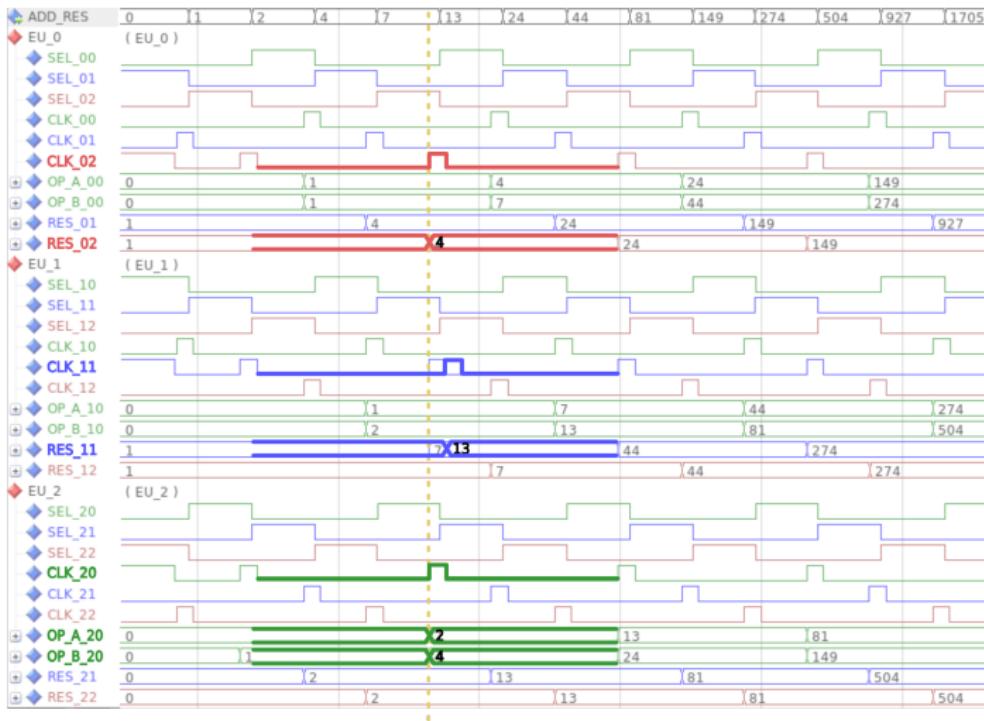


Figure – Chronogramme Tribonacci avec conflit → CLK_11 en retard

Mise en évidence des conflits

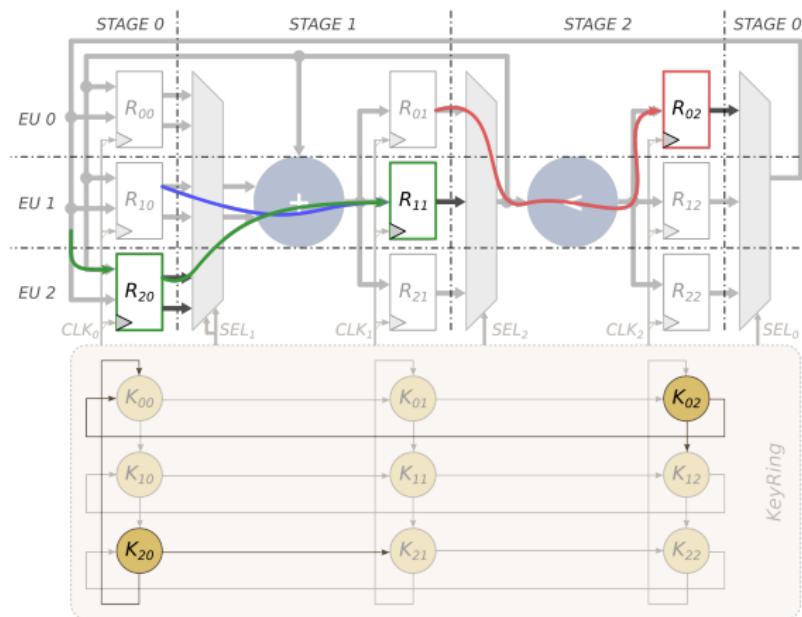


Figure – Circuit Tribonacci avec conflit → CLK_11 en retard

1) Solution architecturale

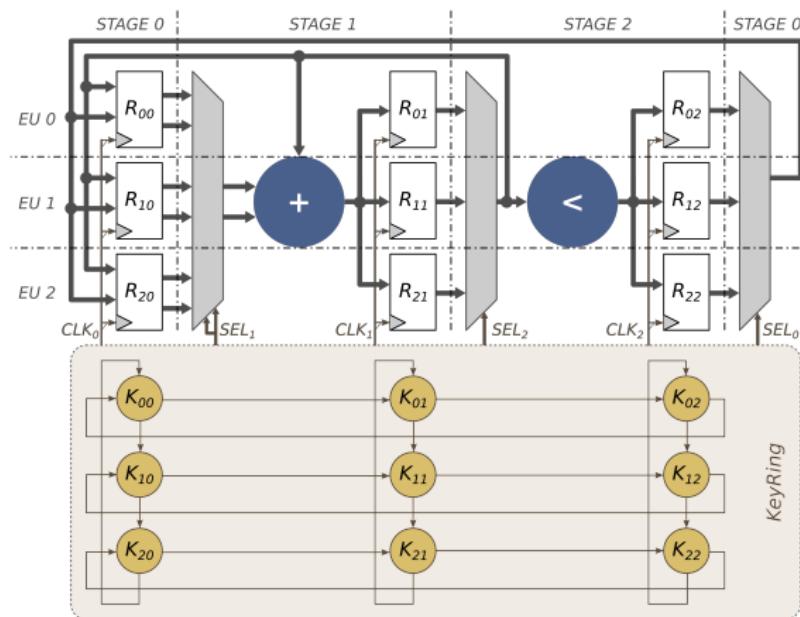


Figure – Circuit Tribonacci → ($E = 3, S = 3, \alpha = 1$)

1) Solution architecturale

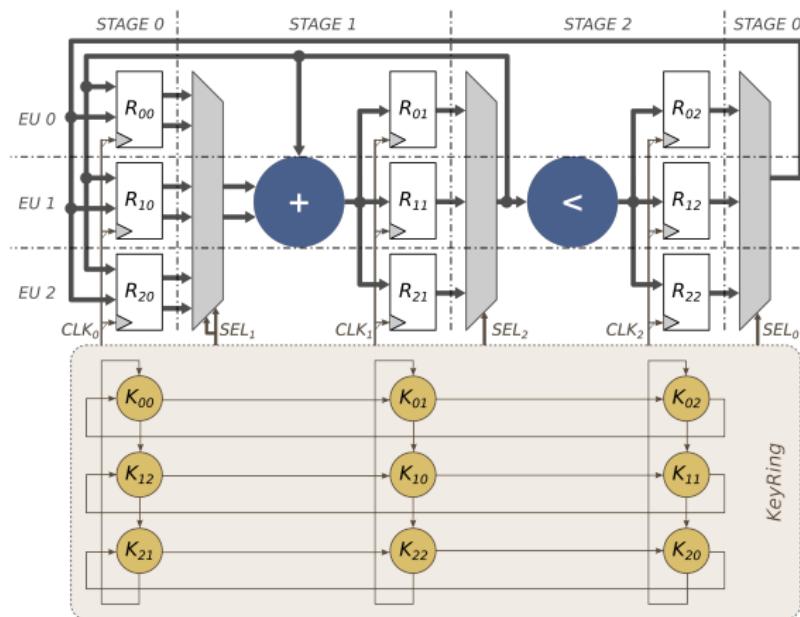


Figure – Circuit Tribonacci $\rightarrow (E = 3, S = 3, \alpha = 2)$

1) Solution architecturale

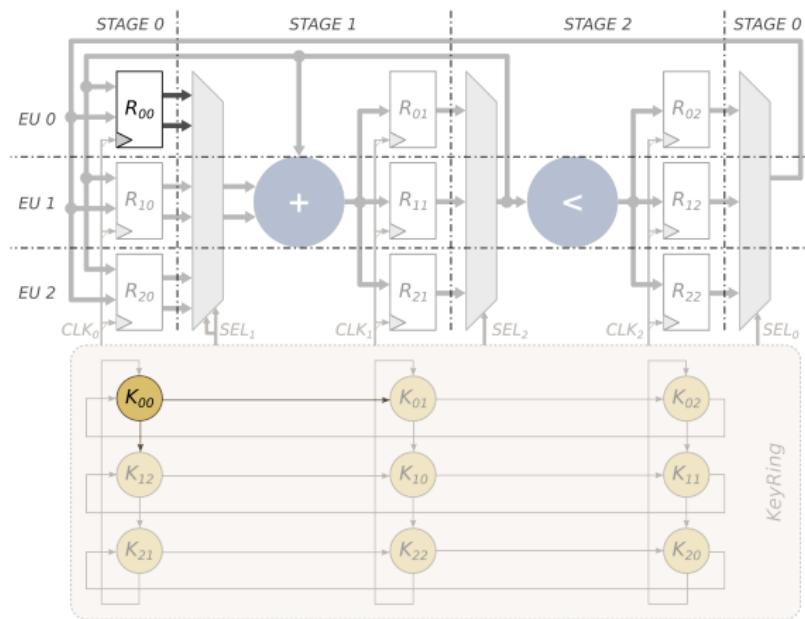


Figure – Circuit Tribonacci $\rightarrow (E = 3, S = 3, \alpha = 2)$

1) Solution architecturale

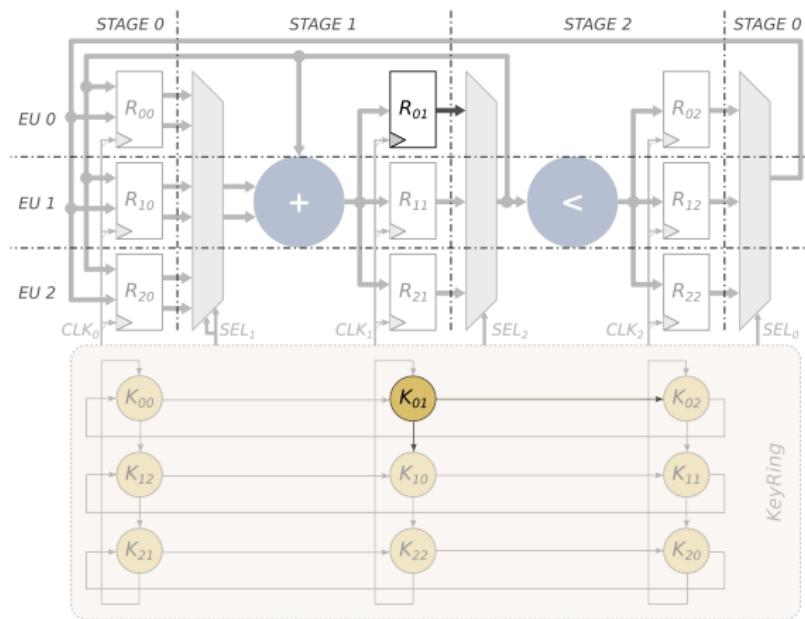


Figure – Circuit Tribonacci $\rightarrow (E = 3, S = 3, \alpha = 2)$

1) Solution architecturale

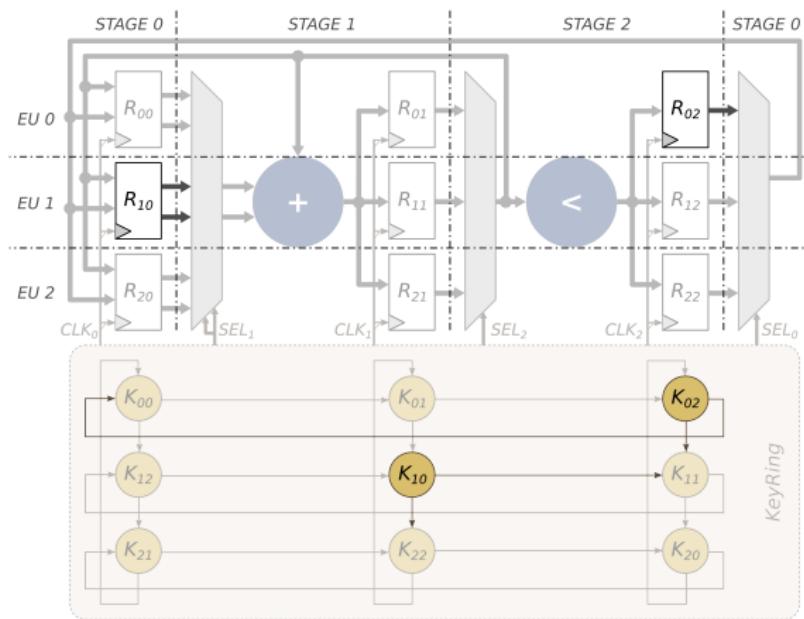


Figure – Circuit Tribonacci $\rightarrow (E = 3, S = 3, \alpha = 2)$

1) Solution architecturale

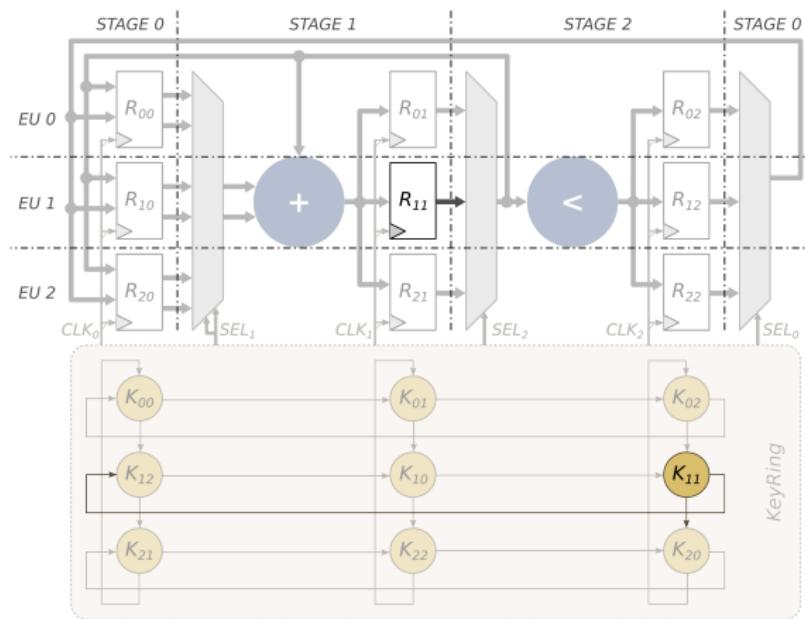


Figure – Circuit Tribonacci $\rightarrow (E = 3, S = 3, \alpha = 2)$

1) Solution architecturale

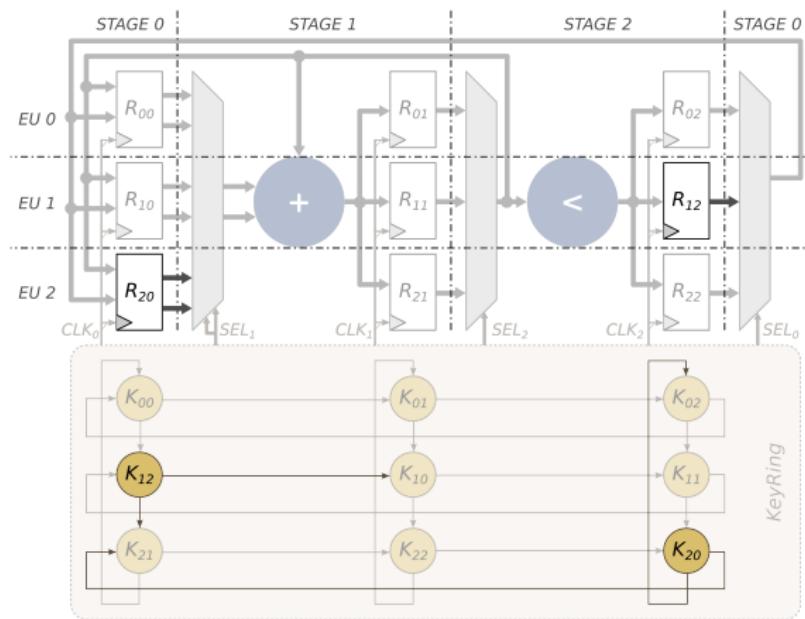


Figure – Circuit Tribonacci $\rightarrow (E = 3, S = 3, \alpha = 2)$

2) Contraintes temporelles

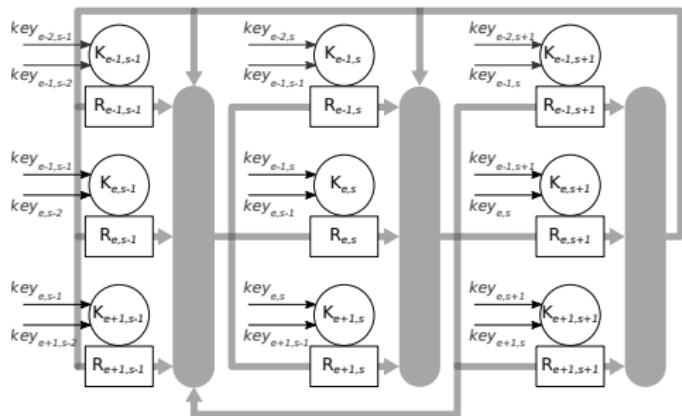


Figure – Circuit KeyRing générique

$$\begin{aligned} C_{e,s-1} \mapsto D_{e,s}^{\max} &< C_{e,s} - \varepsilon \\ C_{e-1,s} \mapsto D_{e,s}^{\max} &< C_{e,s} - \varepsilon \\ C_{e,s-1} \mapsto C_{e,s} &< \{C_{e+1,s-1}, D_{e,s}^{\min}\} - \varepsilon \\ C_{e-1,s} \mapsto C_{e,s} &< \{C_{e-1,s+1}, D_{e,s}^{\min}\} - \varepsilon \end{aligned}$$

2) Contraintes temporelles

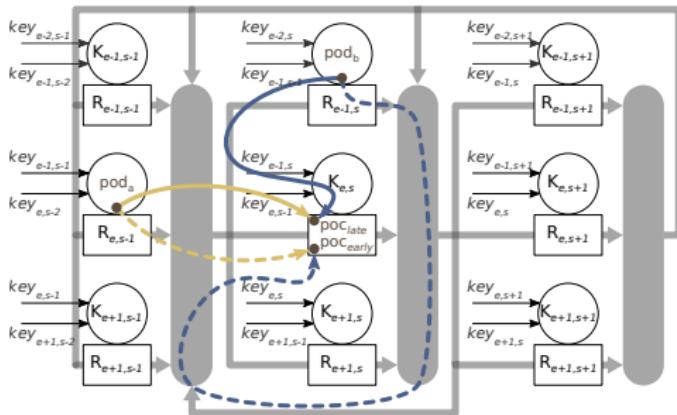


Figure – KeyRing RTC (Setup)

$$C_{e,s-1} \mapsto D_{e,s}^{\max} < C_{e,s} - \varepsilon$$

$$C_{e-1,s} \mapsto D_{e,s}^{\max} < C_{e,s} - \varepsilon$$

$$C_{e,s-1} \mapsto C_{e,s} < \{C_{e+1,s-1}, D_{e,s}^{\min}\} - \varepsilon$$

$$C_{e-1,s} \mapsto C_{e,s} < \{C_{e-1,s+1}, D_{e,s}^{\min}\} - \varepsilon$$

2) Contraintes temporelles

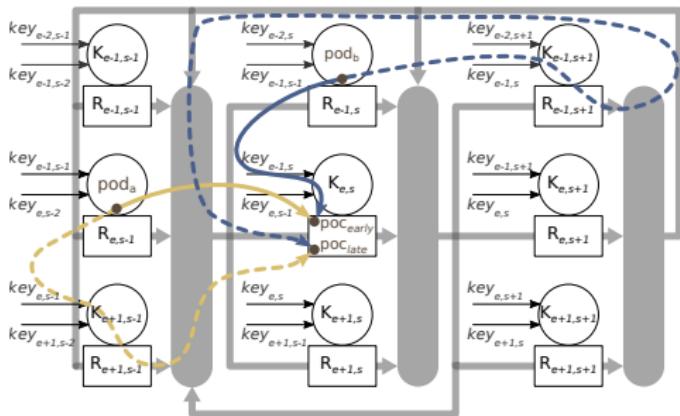


Figure – KeyRing RTC (Hold)

$$\begin{aligned} C_{e,s-1} \mapsto D_{e,s}^{\max} &< C_{e,s} - \varepsilon \\ C_{e-1,s} \mapsto D_{e,s}^{\max} &< C_{e,s} - \varepsilon \\ C_{e,s-1} \mapsto C_{e,s} &< \{C_{e+1,s-1}, D_{e,s}^{\min}\} - \varepsilon \\ C_{e-1,s} \mapsto C_{e,s} &< \{C_{e-1,s+1}, D_{e,s}^{\min}\} - \varepsilon \end{aligned}$$

1 Introduction

2 Microarchitecture KeyRing

3 Processeurs KeyV

4 Conclusion

1 Introduction

2 Microarchitecture KeyRing

3 Processeurs KeyV

4 Conclusion

Processeurs KeyV

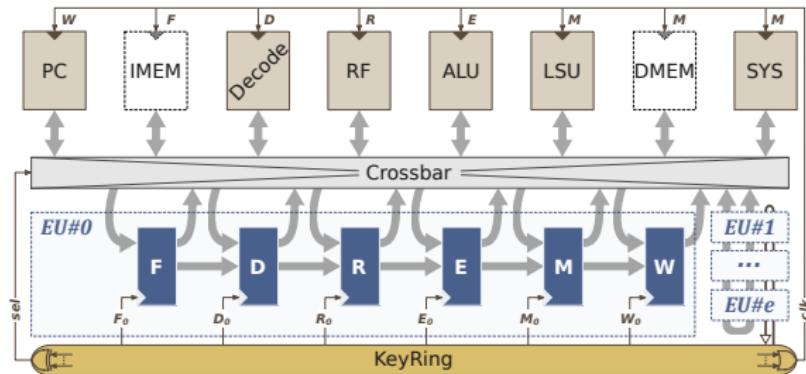


Figure – Microarchitecture des processeurs KeyV

Processeurs KeyV

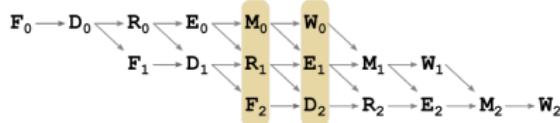
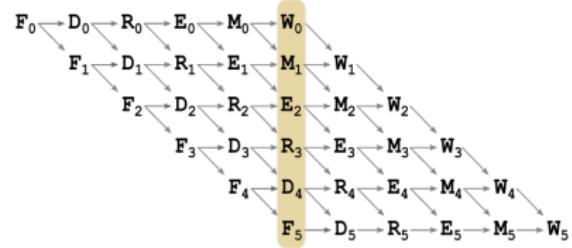
(a) KeyV₃₆₂(b) KeyV₆₆₁

Figure – Parallélisme d'instruction des processeurs KeyV

Processeurs KeyV

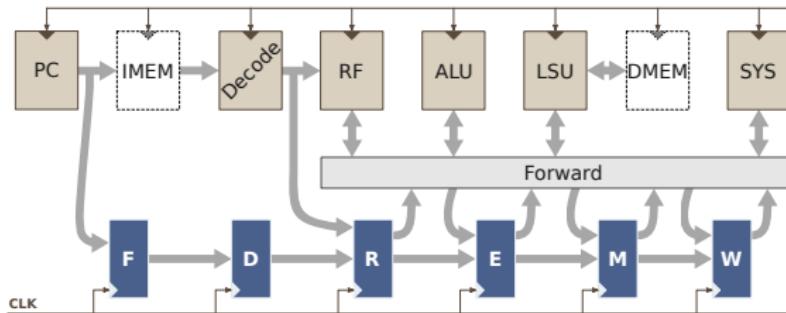


Figure – Microarchitecture du processeur SynV → référence synchrone

Protocole expérimental

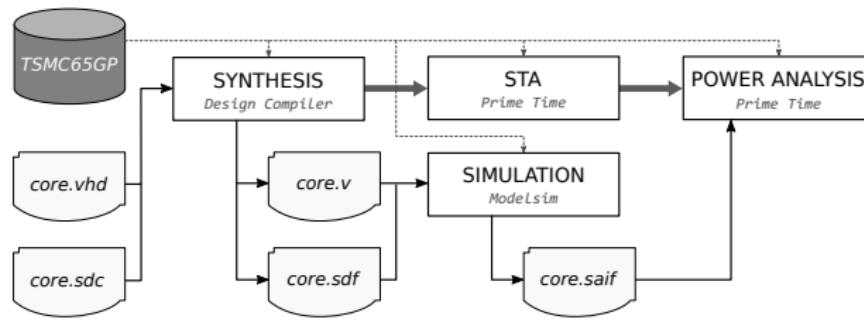


Figure – Environnement de développement

Résultats (STA)

Exemple de rapport de timing : Hold entre R_5 et R_0

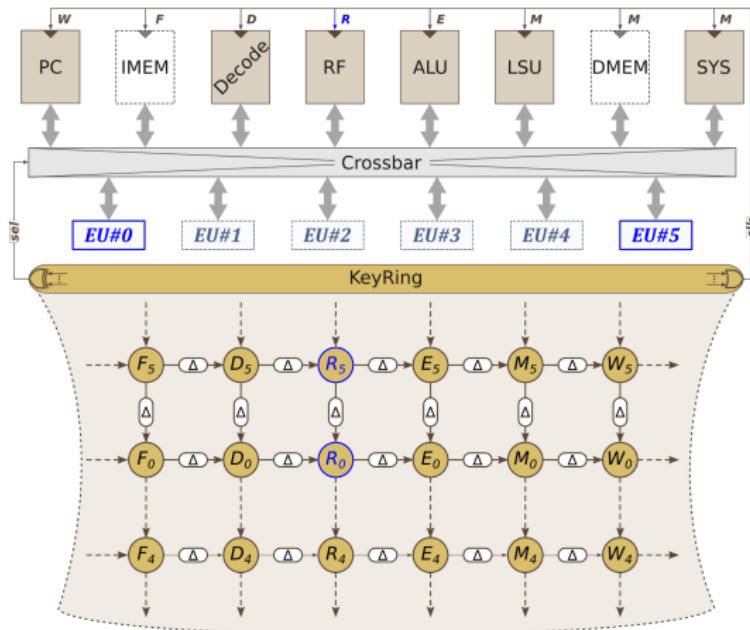


Figure – KeyV₆₆₁

Résultats (STA)

Exemple de rapport de timing : *Hold entre R₅ et R₀*

clock K_main_02_hold_right_launch (rise edge)				
keyring/ku_52/o_ku[KEY]	ku_E5_S2	0.21	0.21	r
keyring/de_52/i_de	de_30	0.00	0.21	r
keyring/de_52/o_de	de_30	2.33	2.54	r
keyring/ku_53/i_ku[KEY_E]	ku_E5_S3	0.00	2.54	r
keyring/ku_53/xor_e/Z	XOR2D0	0.08	2.61	f
keyring/ku_53/nor/ZN	NR3D0	0.05	2.66	r
keyring/ku_53/clkb/Z	CKBDO	0.08	2.74	r
keyring/ku_53/o_ku[CLK]	ku_E5_S3	0.00	2.74	r
alu/i_clk	alu	0.14	2.88	r
alu/alu_res_reg/CP	DFCNQD1	0.09	2.97	r
alu/alu_res_reg/Q	DFCNQD1	0.17	3.14	r
eu_5/i_eu[FROM_ALU]	eu_5	0.08	3.21	r
eu_5/o_eu[TO_RF][DATA_W]	eu_5	0.14	3.35	r
xbs/from_eu[5][TO_RF][DATA_W]	xbs	0.09	3.44	r
rf/i_rf[DATA_W]	rf	0.00	3.52	r
rf/rf_reg[25]/D	EDFCNQD1	0.16	3.60	r
data arrival time				3.60

Figure – STA (Launch)

Résultats (STA)

Exemple de rapport de timing : Hold entre R_5 et R_0

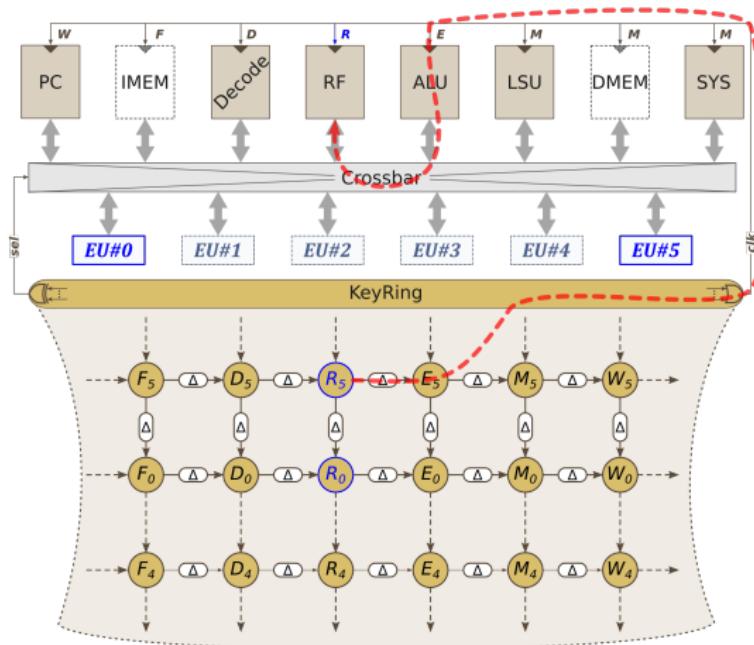


Figure – KeyV661 (Launch)

Résultats (STA)

Exemple de rapport de timing : *Hold entre R₅ et R₀*

clock K_main_02_hold_right_capture (rise edge)				
keyring/ku_52/o_ku[KEY]	ku_E5_S2	0.23	0.23	f
keyring/de_52/i_de	de_30	0.00	0.23	f
keyring/de_52/o_de	de_30	2.40	2.63	f
keyring/ku_02/i_ku[KEY_S]	ku_E0_S2	0.00	2.63	f
keyring/ku_02/xor_s/Z	XOR2D0	0.07	2.70	f
keyring/ku_02/nor/ZN	NR3D0	0.06	2.76	r
keyring/ku_02/clkb/Z	CKBDO	0.08	2.84	r
keyring/ku_02/o_ku[CLK]	ku_E0_S2	0.00	2.84	r
rf/i_clk	rf	0.51	3.35	r
rf/rf_reg[25]/CP	EDFCNQD1	0.21	3.56	r
inter-clock uncertainty		0.10	3.66	
library hold time		-0.07	3.59	
data required time			3.59	

Figure – STA (Capture)

Résultats (STA)

Exemple de rapport de timing : Hold entre R_5 et R_0

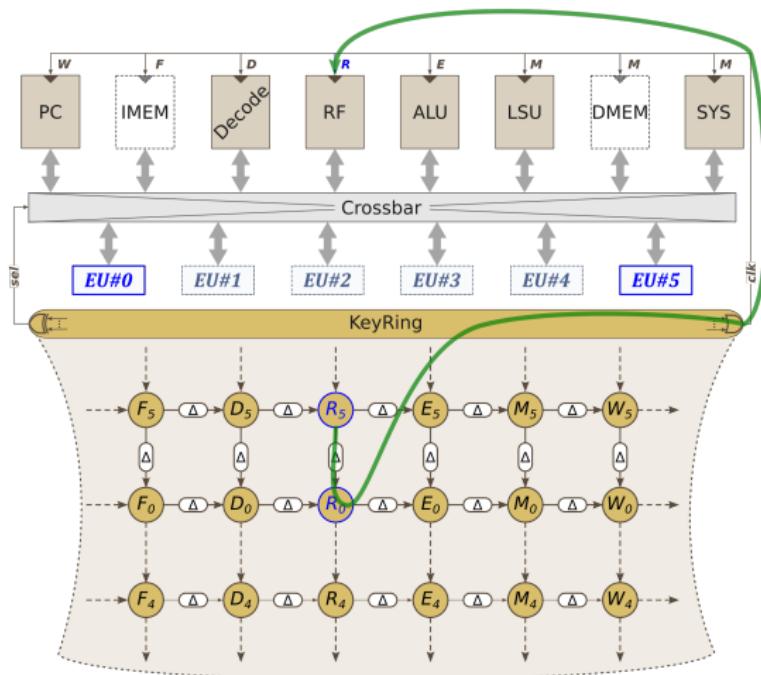


Figure – KeyV661 (Capture)

Résultats (STA)

Exemple de rapport de timing : Hold entre R_5 et R_0

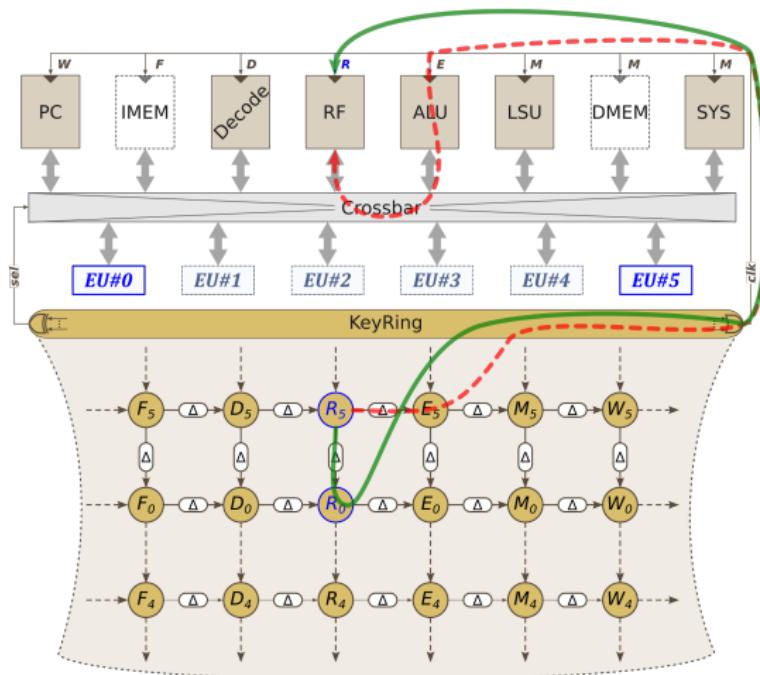
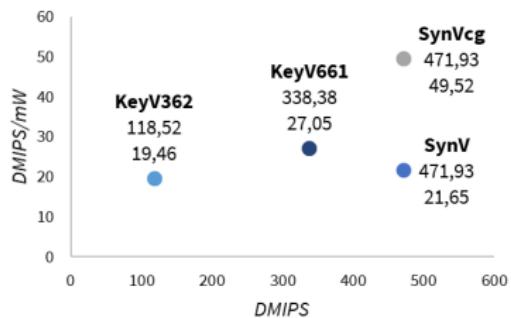
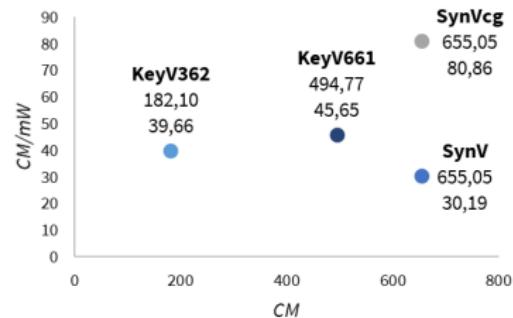


Figure – KeyV₆₆₁ (Launch & Capture)

Résultats (Performances)



(a) dhystone



(b) coremark

Figure – Comparaison des **performances** de KeyV et SynV

Résultats (Performances)

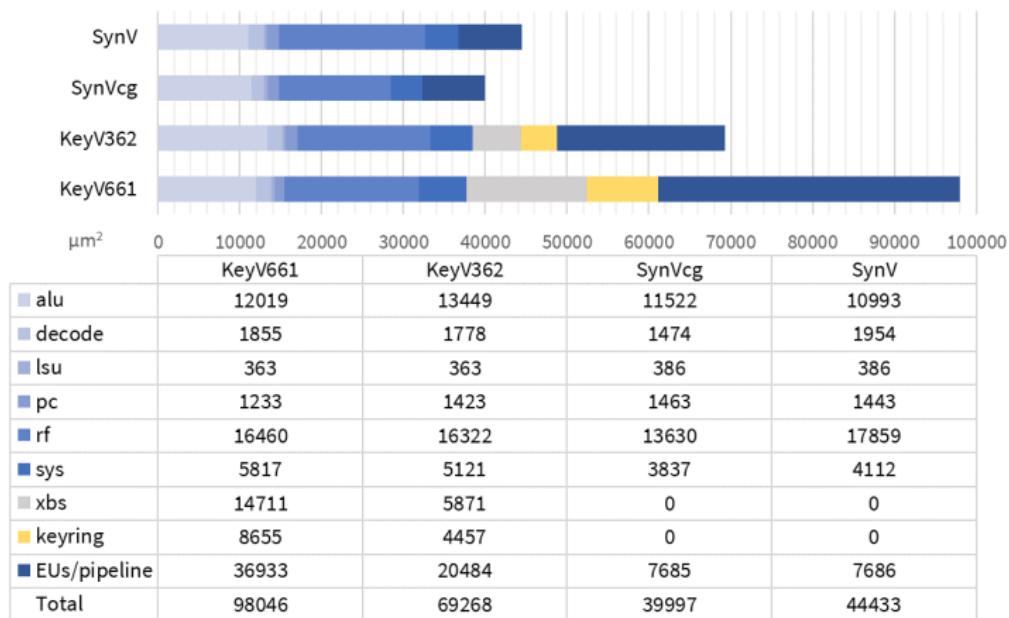


Figure – Comparaison détaillée de la **surface** occupée par KeyV et SynV

Résultats (Performances)

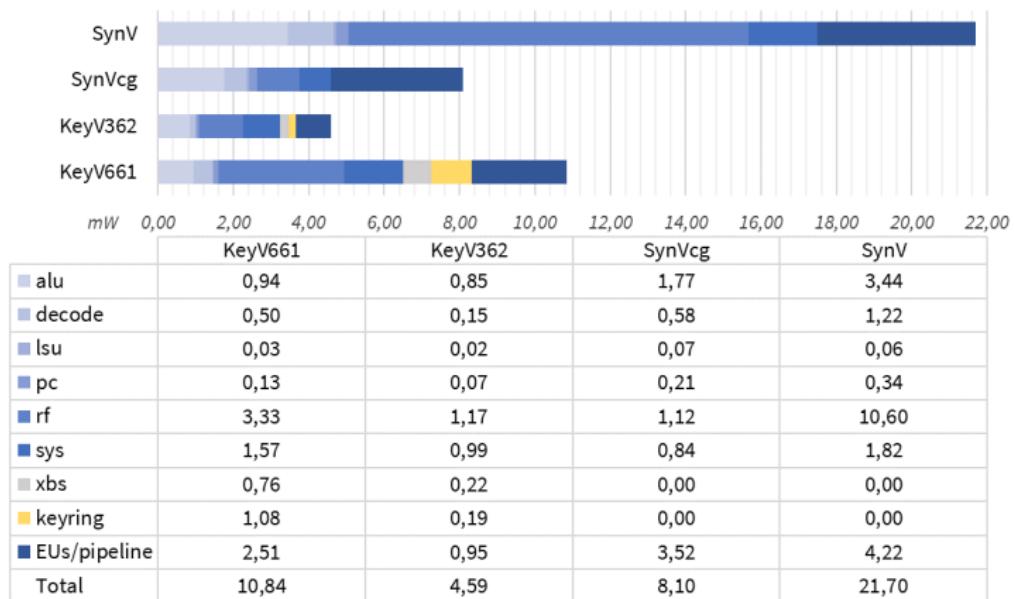


Figure – Comparaison détaillée de la **puissance** consommée par KeyV et SynV (*Coremark*)

1 Introduction

2 Microarchitecture KeyRing

3 Processeurs KeyV

4 Conclusion

1 Introduction

2 Microarchitecture KeyRing

3 Processeurs KeyV

4 Conclusion

Conclusion

En résumé

- Bases de la microarchitecture endochrone KeyRing
- Synthèse par contraintes temporelles + STA
- Étude détaillée des processeurs KeyV

Limitations et perspectives

- Microarchitecture limitée à l'exécution dans l'ordre
- Flot de conception limité à la synthèse / temps de synthèse important
- Limitations micoarchitecturales (*stalls*)
- Limitations des contraintes temporelles (*inter-clock hold*)

Conclusion

En résumé

- **Bases de la microarchitecture endochrone KeyRing**
- Synthèse par contraintes temporelles + STA
- Étude détaillée des processeurs KeyV

Limitations et perspectives

- Microarchitecture limitée à l'exécution dans l'ordre
- Flot de conception limité à la synthèse / temps de synthèse important
- Limitations micoarchitecturales (*stalls*)
- Limitations des contraintes temporelles (*inter-clock hold*)

Conclusion

En résumé

- Bases de la microarchitecture endochrone KeyRing
- Synthèse par contraintes temporelles + STA
- Étude détaillée des processeurs KeyV

Limitations et perspectives

- Microarchitecture limitée à l'exécution dans l'ordre
- Flot de conception limité à la synthèse / temps de synthèse important
- Limitations micoarchitecturales (*stalls*)
- Limitations des contraintes temporelles (*inter-clock hold*)

Conclusion

En résumé

- Bases de la microarchitecture endochrone KeyRing
- Synthèse par contraintes temporelles + STA
- Étude détaillée des processeurs KeyV

Limitations et perspectives

- Microarchitecture limitée à l'exécution dans l'ordre
- Flot de conception limité à la synthèse / temps de synthèse important
- Limitations micoarchitecturales (*stalls*)
- Limitations des contraintes temporelles (*inter-clock hold*)

Conclusion

En résumé

- Bases de la microarchitecture endochrone KeyRing
- Synthèse par contraintes temporelles + STA
- Étude détaillée des processeurs KeyV

Limitations et perspectives

- Microarchitecture limitée à l'exécution dans l'ordre
- Flot de conception limité à la synthèse / temps de synthèse important
- Limitations micoarchitecturales (stalls)
- Limitations des contraintes temporelles (*inter-clock hold*)

Conclusion

En résumé

- Bases de la microarchitecture endochrone KeyRing
- Synthèse par contraintes temporelles + STA
- Étude détaillée des processeurs KeyV

Limitations et perspectives

- Microarchitecture limitée à l'exécution dans l'ordre
- Flot de conception limité à la synthèse / temps de synthèse important
- Limitations micoarchitecturales (stalls)
- Limitations des contraintes temporelles (*inter-clock hold*)

Conclusion

En résumé

- Bases de la microarchitecture endochrone KeyRing
- Synthèse par contraintes temporelles + STA
- Étude détaillée des processeurs KeyV

Limitations et perspectives

- Microarchitecture limitée à l'exécution dans l'ordre
- Flot de conception limité à la synthèse / temps de synthèse important
- Limitations micoarchitecturales (*stalls*)
- Limitations des contraintes temporelles (*inter-clock hold*)

Conclusion

En résumé

- Bases de la microarchitecture endochrone KeyRing
- Synthèse par contraintes temporelles + STA
- Étude détaillée des processeurs KeyV

Limitations et perspectives

- Microarchitecture limitée à l'exécution dans l'ordre
- Flot de conception limité à la synthèse / temps de synthèse important
- Limitations micoarchitecturales (*stalls*)
- Limitations des contraintes temporelles (*inter-clock hold*)

Conclusion

En résumé

- Bases de la microarchitecture endochrone KeyRing
- Synthèse par contraintes temporelles + STA
- Étude détaillée des processeurs KeyV

Limitations et perspectives

- Microarchitecture limitée à l'exécution dans l'ordre
- Flot de conception limité à la synthèse / temps de synthèse important
- Limitations micoarchitecturales (*stalls*)
- Limitations des contraintes temporelles (*inter-clock hold*)

Bibliographie I

- [1] O. Shacham, O. Azizi, M. Wachs, W. Qadeer, Z. Asgar, K. Kelley, J. P. Stevenson, S. Richardson, M. Horowitz, B. Lee, A. Solomatnikov, and A. Firoozshahian, "Rethinking Digital Design : Why Design Must Change," *IEEE Micro*, vol. 30, no. 6, pp. 9–24, Nov. 2010.
- [2] M. Fiorentino, C. Thibeault, Y. Savaria, F. Gagnon, T. Awad, D. Morrissey, and M. Laurence, "AnARM : A 28nm Energy Efficient ARM Processor Based on Octasic Asynchronous Technology," in *2019 25th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, May 2019, pp. 58–59.
- [3] I. E. Sutherland, "Micropipelines," *Commun. ACM*, vol. 32, no. 6, pp. 720–738, Jun. 1989.
- [4] M. Singh and S. M. Nowick, "MOUSETRAP : High-Speed Transition-Signaling Asynchronous Pipelines," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 6, pp. 684–698, Jun. 2007.
- [5] A. Peeters, F. t Beest, M. d Wit, and W. Mallon, "Click Elements : An Implementation Style for Data-Driven Compilation," in *2010 IEEE Symposium on Asynchronous Circuits and Systems*, May 2010, pp. 3–14.
- [6] M. Laurence, "Introduction to Octasic Asynchronous Processor Technology," in *2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems*, May 2012, pp. 113–117.
- [7] T. Awad, M. Laurence, M. Filteau, P. Gervais, and D. Morrissey, "Clock signal propagation method for integrated circuits (ICs) and integrated circuit making use of same," US Patent US8 130 019B1, Mar., 2012. [Online]. Available : <https://patents.google.com/patent/US8130019B1>
- [8] T. Awad, M. Laurence, M. Filteau, P. Gervais, and D. Morrissey, "Method for sharing a resource and circuit making use of same," US Patent US8 689 218B1, Apr., 2014. [Online]. Available : <https://patents.google.com/patent/US8689218B1>

Merci de votre attention!