PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

| | |
|---|---|
| **Titre:** Title: | Convertisseurs temps-numérique distribués et à mesures multiples pour FPGA |
| **Auteur:** Author: | Safa Berrima |
| **Date:** | 2021 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Berrima, S. (2021). Convertisseurs temps-numérique distribués et à mesures multiples pour FPGA [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/6641/ |

**Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

| | |
|---|---|
| **URL de PolyPublie:** PolyPublie URL: | https://publications.polymtl.ca/6641/ |
| **Directeurs de recherche:** Advisors: | Yvon Savaria, & Yves Blaquière |
| **Programme:** Program: | Génie électrique |

**Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal**
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

**POLYTECHNIQUE MONTRÉAL**
affiliée à l'Université de Montréal

# Convertisseurs temps-numérique distribués et à mesures multiples pour FPGA

**SAFA BERRIMA**
Département de génie électrique

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie électrique

Juin 2021

© Safa Berrima, 2021.

**POLYTECHNIQUE MONTRÉAL**

affiliée à l'Université de Montréal

Cette thèse intitulée :

**Convertisseurs temps-numérique distribués et à mesures multiples pour FPGA**

Présentée par **Safa BERRIMA**

en vue de l'obtention du diplôme de *Philosophiæ Doctor*

a été dûment acceptée par le jury d'examen constitué de :

**Jean-Pierre DAVID**, président

**Yvon SAVARIA**, membre et directeur de recherche

**Yves BLAQUIÈRE**, membre et codirecteur de recherche

**Yves AUDET**, membre

**Réjean FONTAINE**, membre externe

# DÉDICACE

*À ma fille,*

*mon chéri,*

*maman et papa*

# REMERCIEMENTS

J’aimerais remercier toute personne avec qui j’ai travaillé durant mes études doctorales. Particulièrement, j’aimerais exprimer toute ma gratitude à mon directeur de thèse Yvon Savaria et à mon codirecteur Yves Blaquière pour leur appui scientifique, leurs innombrables conseils, leur support académique et financier. Ils ont beaucoup œuvré dans la mise en valeur de mon travail. C’est grâce à leur soutien que j’ai pu mener cette thèse à terme. C’était un plaisir de travailler avec eux pendant toutes ces années.

Je remercie également les membres du jury, messieurs Jean-Pierre David, Yves Audet et Réjean Fontaine pour avoir accepté d’étudier et évaluer mon travail.

Je remercie spécialement mon mari Selim pour son support constant et infaillible. Il a toujours cru en mes compétences et a partagé avec moi les moments de joie comme les moments durs de la thèse. Je ne pourrai mesurer son apport dans l’accomplissement de ce travail.

Je remercie mes parents qui m’ont appuyée durant tout mon cursus scolaire et m’ont aidée à surmonter les défis de la vie. Merci de m’avoir donné toutes les chances pour réussir.

Mes plus profonds remerciements vont aussi à mes formidables amis qui par leur amour, présence et encouragements m’ont aidée à mener ce projet à terme. Je tiens à m’excuser pour toutes les fois où j’ai raté l’occasion de partager de bons moments avec eux pour travailler sur la thèse.

Une pensée spéciale pour mes collègues du Groupe de Recherche en Microélectronique et Microsystèmes (GR2M) avec qui j’ai partagé le bureau pendant plusieurs années. J’en garde de bons souvenirs et je leur souhaite une pleine réussite professionnelle.

# RÉSUMÉ

Les convertisseurs temps-numérique (TDC), outils de mesure de la durée de l'intervalle temps entre deux événements, connaissent un intérêt croissant dans la communauté scientifique et dans l'industrie, et ce depuis des décennies. Ils sont largement utilisés dans des applications variées, que ce soit en imagerie médicale, dans les automobiles ou en instrumentation. Cependant, dans les applications contemporaines et futures, les microsystèmes électroniques incluent plusieurs puces électroniques, dont certaines dans un même boîtier pour réduire leur volume, l'énergie consommée et le coût. Ces microsystèmes miniaturisés peuvent contenir des centaines de milliers de nœuds portant des signaux entre les modules d'un système et un des défis qui leur est associé concerne le diagnostic d'éventuels mal-fonctionnements. Par conséquent, une faible complexité ainsi que le support de multiples points de mesures deviennent des critères tout aussi importants, pour un TDC, que ses performances temporelles telles que la résolution, la linéarité et la précision.

Cette thèse vise à faciliter le prototypage, le déverminage et le test des microsystèmes actuels par l'élaboration de circuits de support in-situ, en particulier pour mesurer précisément les délais entre les transitions sur de multiples nœuds portant des signaux numériques. Nous proposons d'élaborer des architectures distribuées de convertisseurs temps-numérique, qui sont à la fois compacts et précis, tout en permettant la mesure d'intervalles de temps sur de nombreux points de mesure. Des solutions sont proposées grâce à des circuits intégrés programmables (*Field Programmable Gate Array* : FPGA) tirant avantage de l'évolution de leur technologie de fabrication, leur flexibilité et leur faible coût de développement.

La première contribution de cette thèse a été inspirée d'une architecture de TDC implémentée sur un circuit dédié, où les ressources sont disposées en boucles pour être réutilisées, limitant ainsi leur complexité. Une preuve de concept d'une telle architecture implémentée sur un circuit programmable est présentée. L'architecture proposée réduit la complexité de la mise en œuvre en tirant avantage des mémoires distribuées implémentées dans des tables de stockage (*Look-Up Tables*) disponibles dans les FPGA de Xilinx. Une étude théorique comparative a démontré que la solution proposée est plus compacte que les TDC bouclés classiques.

La deuxième contribution de cette thèse est la preuve de concept d'un outil d'ajustement fin des délais dans un FPGA. Cette contribution est venue comme une solution pour améliorer la linéarité

du TDC proposé dans la première contribution. Au meilleur de nos connaissances, cet outil est le premier qui propose un contrôle fin de délai automatisé sans apporter de modifications au circuit déjà placé et routé, sans nécessiter une reconfiguration dynamique du FPGA ni une application logicielle dédiée. Il s'agit d'ajouter uniquement des jonctions de fils au niveau des ports des matrices de commutation augmentant ainsi leur charge capacitive et par conséquent le délai de propagation à travers les traces passant par ces ports. Seulement des commandes TCL prédéfinies par Xilinx sont utilisées.

Une troisième contribution de cette thèse consiste à mettre en place une méthodologie d'automatisation de la technique d'ajustement des délais à travers des scripts TCL exécutés par l'outil de développement Vivado pour les FPGA de Xilinx. Une stratégie d'application de ces scripts sur le TDC proposé a été développée et il a été démontré que cette technique de contrôle fin des délais permet d'en améliorer considérablement la linéarité.

Une quatrième contribution consiste à étudier l'impact de la parallélisation du TDC proposé sur ses performances temporelles. Ceci consiste à proposer une stratégie de mesure à canaux multiples. Une méthode de calibration de TDC est proposée. Cette méthode exploite les algorithmes présentés dans la troisième contribution pour augmenter l'exactitude des mesures. Il a été démontré que le TDC parallélisé et balancé permet d'atteindre une précision et une exactitude des mesures compétitives à ce qu'on trouve dans la littérature, tout en complexité moindre.

Nous pensons que les travaux élaborés dans le cadre de cette thèse non seulement préparent le terrain au déverminage des circuits à haute densité d'encapsulation, mais aussi fournissent des outils d'ajustement des délais qui peuvent être utilisés dans toutes les architectures de TDC implémentées sur des FPGA de Xilinx, ou plus généralement, dans n'importe quelle application où le contrôle fin des délais dans un FPGA de Xilinx est nécessaire.

# ABSTRACT

Time-to-Digital Converters (TDC), tools for measuring the time between two events, have an increased interest by the scientific community and the industry for decades. They are widely used in various applications whether in medical imaging, automotive or instrumentation. However, in contemporary and future applications, microsystems include multiple microchips, some in a single package, to reduce bulk, power consumption and cost. These miniaturized microsystems can contain hundreds of thousands of signals between the chips and one of the challenges is diagnosing a malfunction. Therefore, a small form factor as well as the support of multiple measurements points become just as important criteria for a TDC as the temporal performances such as resolution, linearity and precision.

This thesis aims to facilitate the prototyping, debugging and testing of current microsystems by the development of in-situ support circuits, in particular to precisely measure the delays between transitions on multiple digital signals. We propose to develop distributed architectures of time-to-digital converters, which are both compact and precise while supporting a large number of measurement points. Solutions are offered for programmable integrated circuits (FPGAs) taking advantage of their evolving manufacturing technology, flexibility and low development cost.

The first contribution was inspired by a TDC architecture implemented on a dedicated circuit, where resources are arranged in loops for reuse, thus minimizing the form factor. A proof of concept of such an architecture implemented on a programmable circuit was presented. The proposed architecture takes advantage of the distributed memories implemented with simple Look-Up Tables available in Xilinx FPGAs to save resources. A comparative theoretical study has shown that the proposed solution is more compact than that of a conventional looped TDC.

The second contribution is a proof of concept of a fine resolution delay adjustment tool in FPGA. This contribution came as a solution to improve the linearity of the TDC proposed in the first contribution. To the best of our knowledge, this tool is the first to offer fine automated delay control without modifying the design already placed and routed and without necessitating the dynamic reconfiguration or a dedicated software library. It is only a matter of adding wire junctions at the level of the ports of the switching matrices thus increasing their capacitive load and consequently

the propagation delay through the traces passing through these ports. Only Xilinx-predefined TCL commands are used.

The third contribution is to set up a methodology for automating the delay adjustment technique through TCL scripts executed by the Vivado FPGA development tool from Xilinx. A strategy for applying these scripts to the proposed TDC has been developed and this fine delay control technique has been shown to significantly improve linearity.

The fourth contribution consists in studying the impact of the parallelization of the proposed TDC on its temporal performances. This consists of proposing a multichannel measurement strategy. A proposed TDC balancing method using the algorithms presented in the third contribution in order to increase the accuracy of the measurements has been developed. Parallelized and balanced TDC has been shown to achieve measurement precision and accuracy competitive with that found in the literature while having the smallest form factor.

We believe that the work carried out in the framework of this thesis not only prepares the ground for the debugging of high-density encapsulation circuits but also provides delay adjustment tools that can be used in all TDC architectures implemented on Xilinx FPGAs or more generally in any application where fine control of delays in an Xilinx FPGA is required.

# TABLE DES MATIÈRES

# LISTE DES TABLEAUX

# LISTE DES FIGURES

# LISTE DES SIGLES ET ABRÉVIATIONS

| | |
|---|---|
| ASIC | Application-Specific Integrated Circuit |
| AV | Average |
| BRAM | Block Random Access Memory |
| CLB | Configurable Logic Block |
| DNL | Differential Non-Linearity |
| DSP | Digital Signal Processor |
| FF | Flip Flop |
| FLIM | Fluorescence-Lifetime Imaging Microscopy |
| FPGA | Field Programmable Gate Array |
| FTD | Fine-Tuning Delay |
| ILA | Integrated Logic Analyzer |
| INL | Integral Non-Linearity |
| LiDAR | Light Detection and Ranging |
| LUT | Look-Up Table |
| LUTRAM | Look-Up Table Random Access Memory |
| MMRO | Multi-Measurements Ring Oscillator |
| PET | Positron Emission Tomography |
| PIP | Programmable Interconnect Point |
| RO | Ring Oscillator |
| RMS | Root Mean Square |
| RMSD | Root Mean Square Deviation |
| SSP | Single-Shot Precision |

| | |
|---|---|
| STA | Static Timing Analyzer |
| STD | Standard Deviation |
| TCL | Tool Command Language |
| TDC | Time-to-Digital Converter |
| TDL | Tapped Delay Line |
| VDL | Vernier Delay Line |

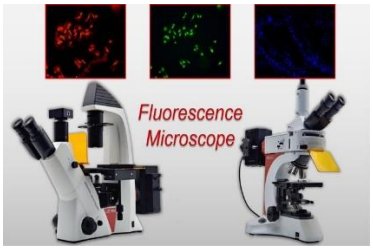
# CHAPITRE 1 INTRODUCTION

Ce chapitre introduit la recherche effectuée en la plaçant d'abord dans son contexte. L'énoncé du problème est souligné par la suite. Ce chapitre présente également les objectifs ciblés par les travaux de recherche réalisés au sein de ces études doctorales ainsi que leurs contributions. Finalement, un sommaire de la structure du reste de la thèse est présenté.

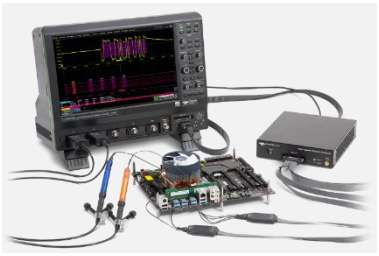
## 1.1 Contexte de la recherche et motivation

La mesure précise du délai entre deux événements est utilisée depuis des décennies en physique des particules et des hautes énergies. On la trouve également dans divers domaines tels qu'illustré par la Figure 1.1. Dans la médecine nucléaire, notamment en imagerie médicale, elle est requise dans la tomographie par émission de positrons (PET) (Figure 1.1(a)) basée sur le temps de vol (*Time of Flight* ToF) [1]. La tomographie par émission de positrons se base sur l'injection d'un traceur radioactif dans le corps humain, qui émet des positrons dont l'annihilation produit deux photons. Le concept de temps de vol permet de déterminer la différence entre les temps de détection des deux photons, ce qui permet de localiser l'endroit où l'annihilation a eu lieu et par conséquent d'améliorer le rapport contraste à bruit des images produites. Le temps de vol est déterminé par un outil de mesure de délais. Une autre application en biomédical où la mesure de délais est largement utilisée, est la microscopie à fluorescence (FLIM) (Figure 1.1(b)) [2]. Il s'agit d'une technique d'imagerie où un échantillon fluorescent est excité par une source de lumière (laser) afin d'extraire ses propriétés moléculaires. La technique se base sur la mesure de la durée de vie de l'objet fluorescent, définie par le temps durant lequel la molécule reste excitée avant de retourner à son état normal en émettant un photon. On trouve les circuits de mesure de délais temporels également en instrumentation, plus précisément dans les analyseurs logiques (Figure 1.1(c)) ou en systèmes électroniques comme dans les boucles à verrouillage de phase (PLL) numériques (ADPLL) [3]. Dans les ADPLL, de tels circuits sont utilisés pour détecter la fréquence/phase remplaçant ainsi le détecteur de phase et la pompe de charge dans une PLL conventionnelle. La mesure de délai a fait également son entrée dans le monde de l'automobile avec la technologie LiDAR (Figure 1.1(d)) qui permet de détecter les distances des obstacles. Le principe est d'émettre des faisceaux et mesurer le temps de retour après réflexion sur des objets à proximité.

(a)




(b)



(c)

(d)

Figure 1.1 Exemples d'applications où la mesure temporelle est utilisée (a) PET (b) FLIM (c) instruments de mesure (d) LiDAR

Une telle technologie ouvre la voie aux véhicules autonomes sans conducteur. Pour toutes ces applications, une panoplie de circuits de mesure de délais a été proposée, où chaque circuit répond aux exigences de l'application à laquelle il est dédié.

Les exigences des systèmes électroniques actuels sont guidées par une explosion d'applications de pointe comme les centres de données dans l'ère des zettaoctets de capacité de stockage, l'internet des objets (IoT), la diffusion des vidéos à très haute définition ou la communication sans fil 5G. Ces applications nécessitent une large bande passante, une faible consommation d'énergie, une grande densité ainsi qu'une fonctionnalité et une flexibilité élevées. Pour répondre à ces besoins, l'industrie des semi-conducteurs a introduit des techniques d'intégration innovantes au fil des années. On trouve comme exemple les microsystèmes avec encapsulation/intégration tridimensionnelle (3D) [4], ou l'intégration 2.5 D [5] ou encore l'intégration EMIS (*Embedded Multi-Die Interconnect Bridge*) [6].

## 1.2 Énoncé du problème

Les microsystèmes contiennent des milliers, voire des millions, d'interconnexions sur lesquelles se propagent des signaux numériques caractérisés par des transitions entre les potentiels qui expriment des niveaux binaires 0 et 1. Dans les microsystèmes actuels, la miniaturisation, le manque d'observabilité ainsi que le très grand nombre de signaux constituent des défis majeurs pour la mesure des délais sur les signaux.

Il existe une panoplie d'applications où la mesure des délais entre les transitions serait nécessaire telles que pour :

- détecter les différences (*mismatching*) en temps de propagation des signaux sur un bus parallèle reliant deux puces afin de fournir l'information nécessaire à un module de calibration pour l'équilibrage;
- réduire la consommation d'énergie : dans un système DVFS (*Dynamic Voltage and Frequency Scaling*), un outil de mesure de délais fournit le temps de propagation à travers le chemin critique à un contrôleur de fréquence/tension. Le contrôleur ajustera par la suite la fréquence d'opération et la tension d'alimentation du système de façon à améliorer ses performances et à réduire sa consommation d'énergie;
- capter/mesurer la température : une implémentation possible d'un capteur de température serait de mesurer la fréquence de deux oscillateurs en anneau, l'un alimenté par une source de courant indépendante de la température tandis que l'autre est alimenté par une source de courant proportionnelle à la température absolue (PTAT). Un comparateur de fréquence produit une impulsion correspondant à la différence des fréquences des deux oscillateurs. La largeur de cette impulsion représentant un délai, doit par la suite être mesurée pour extraire la température;
- détecter les erreurs temporelles (*Timing errors*) et les pannes de délai (*Timing faults*) pouvant être associées aux interconnexions seulement ou à la logique (ou des deux). Par exemple, un temps de montée ou de descente excessivement long ou rapide peut engendrer une panne de délai. Un circuit basé sur un inverseur Schmidt peut transformer le temps de montée (ou descente) en une impulsion qui est par la suite mesurée pour être comparée à un seuil;
- comparer des phases de plusieurs signaux;
- réduire le biais entre plusieurs horloges (*Clock deskewing*);
- construire des diagrammes de l'œil.

Par conséquent, élaborer des circuits de support in situ distribués et compacts, offrant de multiples points de mesure pour la mesure fine et précise des délais entre un grand nombre de signaux numériques est une contribution qui aiderait au déverminage des microsystèmes. De plus, comme le marché concurrentiel actuel ajoute la nécessité d'un temps de développement relativement court et d'un prototypage rapide, de tels circuits de mesure de délais implémentés sur un circuit intégré reprogrammable (FPGA) seraient une solution répondant à la fois aux exigences des systèmes électroniques actuels et futurs et aux exigences du marché. En effet, la technologie de fabrication des FPGA est en constante évolution, atteignant les 16 nm par exemple pour les FPGA Virtex Ultrascale et Kintex Ultrascale de Xilinx. Cette évolution de processus de fabrication a fait que les FPGA sont de plus en plus miniaturisés et les performances temporelles obtenues sont compétitives à celles atteintes avec des circuits intégrés dédiés. De plus, les outils de conception des circuits reprogrammables se sont améliorés facilitant ainsi le flux de développement. Les FPGA ont commencé à être intégrés dans les produits finaux et ne sont plus limités à la phase de prototypage.

## 1.3 Les objectifs de la recherche

Les objectifs de cette thèse peuvent être décrits comme suit :

Élaborer une architecture de circuit de mesure de délais, sur un circuit programmable, pouvant être déployée comme un circuit de support in situ facilitant le prototypage et le déverminage des microsystèmes en répondant à la fois aux exigences telles que le nombre élevé de points de mesure, faible consommation de ressources et des performances temporelles compétitives. Par conséquent, l'objectif principal de ce travail consiste à:

*Élaborer des circuits de support compacts pour la mesure précise des délais entre de nombreux signaux.*

Pour cela, des architectures distribuées de convertisseur temps-numérique (TDC) seront étudiées et élaborées pour un circuit programmable. De nouvelles connaissances seront générées quant à la caractérisation et la calibration de tels convertisseurs. Une évaluation et une comparaison des ressources et des performances temporelles avec ce qui est récemment rapporté dans la littérature seront effectuées. Dans ce sens, les objectifs spécifiques suivants découlent de l'objectif principal :

1) Proposer et concevoir des convertisseurs temps-numérique dans un circuit programmable compacts et distribués qui se veulent des circuits de support in situ pour la mesure de délai;

2) Mettre en place des stratégies de calibration et d'amélioration des performances temporelles
   a) proposer et générer des outils pour contrôler finement les délais;
   b) élaborer une méthodologie d'implémentation de ces outils dans des circuits programmables et donnant des résultats reproductibles.

## 1.4 Contributions et impact

Au mieux de nos connaissances, les contributions de cette thèse, décrites dans le corps de ce manuscrit, sont significatives et originales. Elles ouvrent la voie à des avancées dans le domaine de mesure de temps dans les circuits reconfigurables. La première contribution consiste à proposer une nouvelle architecture de TDC répondant aux exigences des microsystèmes modernes et futurs à savoir la miniaturisation et les multiples mesures. Cette contribution a été publiée à la conférence ISCAS en 2017 :

> S. Berrima, Y. Blaquière and Y. Savaria, "A multi-measurements RO-TDC implemented in a Xilinx field programmable gate array," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-4, 2017, doi: 10.1109/ISCAS.2017.8050436.

La deuxième contribution consiste en une preuve de concept d'un nouvel outil d'ajustement fin des délais dans un FPGA. Ce travail a été publié à la conférence MWSCAS en 2017 :

> S. Berrima, Y. Blaquière and Y. Savaria, "Sub-ps resolution programmable delays implemented in a Xilinx FPGA," *IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 918-921, 2017, doi: 10.1109/MWSCAS.2017.8053074.

La troisième contribution consiste à démontrer l'impact que la méthode d'ajustement des délais proposée a sur les performances du TDC proposé dans la première contribution. Il a été démontré que la linéarité de ce TDC peut être nettement améliorée. Les algorithmes développés en TCL (*Tool Command Language*) peuvent parfaitement être appliqués sur tout TDC implémenté sur un FPGA de Xilinx pour réduire la non-linéarité. Ce travail a été publié dans la revue IET Circuits, Devices and Systems en 2020 :

> S. Berrima, Y. Blaquière and Y. Savaria, "A fine Resolution Delay Tuning Method to improve the Linearity of an Unbalanced Time-to-Digital Converter on a Xilinx FPGA", *IET*

*Circuits, devices & systems*, vol 14, issue 8, pp. 1243-1252, 2020, doi: 10.1049/iet-cds.2020.0026

La dernière contribution propose une version améliorée du TDC à multiples canaux de la première contribution et démontre l'impact de l'utilisation de l'outil d'ajustement des délais sur l'amélioration de l'exactitude des mesures. Ce travail a été soumis à IEEE Transactions on Instrumentation and Measurement :

S. Berrima, Y. Blaquière and Y. Savaria, "Ring-Oscillator Based High Accuracy Low Complexity Multichannel Time-to-Digital Converter Architecture for Field-Programmable Gate Arrays", *IEEE Transactions on Instrumentation and Measurement*, submitted on May 08, 2021.

## 1.5 Aperçu de la thèse

Cette thèse est rédigée selon le modèle de thèse par articles où les articles publiés et soumis sont présentés dans le corps du manuscrit. Ce manuscrit couvre 2 articles de conférence et un article de revue publiés ainsi qu'un article de revue soumis.

Ce document contient les chapitres suivants :

Chapitre 1 : est une introduction mettant la recherche dans son contexte et présentant les objectifs ciblés par ce travail;

Chapitre 2 : met la lumière certains concepts fondamentaux liés à la mesure temporelle et utilisés dans les chapitres subséquents. On y trouve aussi une revue critique de la littérature;

Chapitre 3 : présente la démarche de l'ensemble du travail de recherche et l'organisation générale du document indiquant la cohérence des articles par rapport aux objectifs énumérés au chapitre 1;

Chapitre 4 : présente la première partie de la contribution doctorale : une proposition d'une architecture compacte et distribuée d'un convertisseur temps-numérique implémentée sur un FPGA;

Chapitre 5 : présente la deuxième partie de la contribution doctorale : une preuve de concept d'une méthode d'ajustement fin des délais dans un FPGA;

Chapitre 6 : présente la troisième partie de la contribution doctorale : élaboration d'algorithmes d'ajustement des délais par des scripts en TCL et l'étude de l'impact de ces algorithmes sur la réduction de la non-linéarité du convertisseur présenté au chapitre 4;

Chapitre 7 : présente la quatrième partie de la contribution doctorale : élaboration d'une version du TDC proposé supportant multiples canaux et l'étude de l'impact des algorithmes d'ajustement des délais sur l'exactitude des mesures;

Chapitre 8 : discute les contributions décrites dans les chapitres précédents;

Chapitre 9 : clôt le document en résumant les principales contributions et donne des pistes et des recommandations pour des travaux futurs.

# CHAPITRE 2 CONCEPTS DE BASE ET TRAVAUX ANTÉRIEURS SUR LES CONVERTISSEURS TEMPS-NUMÉRIQUE

Ce chapitre présente d'abord quelques concepts associés aux convertisseurs temps-numérique largement utilisés dans les chapitres suivants. De plus il comprend une synthèse critique des travaux publiés dans la littérature et en lien avec les travaux réalisés dans le cadre de cette thèse. Ce chapitre se divise en trois parties : la première définit les caractéristiques temporelles basiques de TDC telles que la résolution, l'erreur de quantification, la précision, la linéarité et l'exactitude. La deuxième partie présente les principales architectures de TDC implémentées sur FPGA ainsi que leurs avantages et inconvénients. La troisième partie survole quelques méthodes d'ajustement des délais dans les FPGA.

## 2.1 Caractéristiques fondamentales d'un convertisseur temps-numérique

Afin de mesurer un intervalle de temps entre deux événements, les TDC associent en général une architecture à fine granularité avec une architecture à une granularité grossière basée sur des compteurs [7]. Pour mesurer l'intervalle temporel ($T_{Départ\text{-}Arrêt}$) séparant l'arrivée d'une transition montante sur un signal « Départ » et l'arrivée d'une transition montante sur un signal « Arrêt », l'interpolation Nutt [8] est communément utilisée comme illustré à la Figure 2.1.

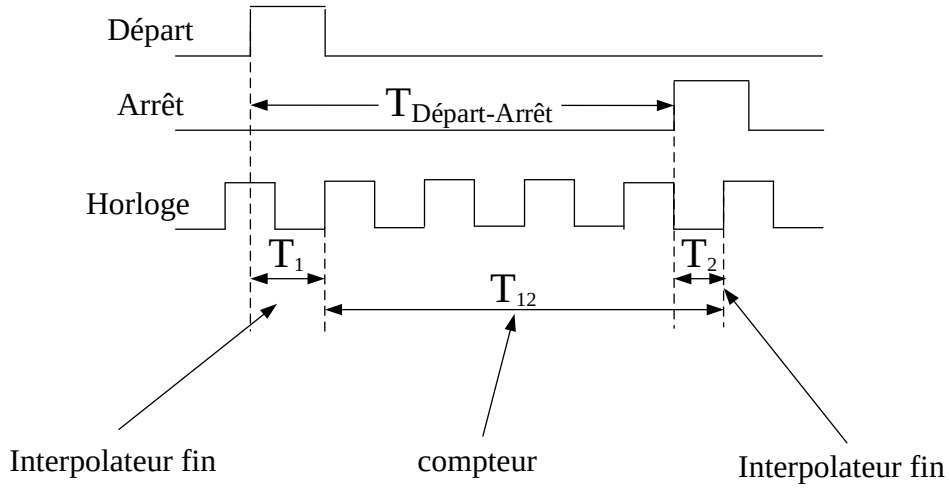



Figure 2.1 Interpolation de Nutt [8]

Un premier interpolateur fin mesure le temps ($T_1$) entre la transition montante de « Départ » et la prochaine transition montante de l'horloge, un compteur mesure le temps ($T_{12}$) séparant la transition montante de l'horloge suivant le signal « Départ » et la transition montante de l'horloge suivant le signal « Arrêt » et un deuxième interpolateur fin mesure le temps ($T_2$) séparant la transition montante du signal « Arrêt » et la prochaine transition montante de l'horloge. L'intervalle de temps séparant « Départ » et « Arrêt » est donné par l'équation (2.1) :

$$T_{\text{Départ-Arrêt}} = T_1+T_{12}-T_2 \tag{2.1}$$

Comme tout instrument de mesure, le TDC possède des caractéristiques fondamentales. Dans cette section, nous revenons sur les principales à savoir la résolution, l'erreur de quantification, la précision, la linéarité et l'exactitude.

À noter que dans le reste de la section, le terme TDC désigne l'interpolateur fin du TDC.

### 2.1.1 Résolution et erreur de quantification

La caractéristique d'entrée-sortie d'un TDC idéal est donnée par la Figure 2.2 où l'axe des abscisses représente l'entrée du TDC (intervalle continu de temps) et l'axe des ordonnées représente la sortie du TDC (une représentation binaire de l'entrée). L'intervalle de temps en entrée est une valeur continue traduite en une valeur discrète. Par conséquent, il existe une plage d'intervalles de temps qui seront traduits par le même code en sortie. Ces plages sont séparées par la résolution $R_{LSB}$. Le code numérique sortant du TDC augmente de 1 à chaque incrémentation de $1\times R_{LSB}$ de l'intervalle de temps en entrée. La résolution peut donc être définie comme le pas de quantification ou encore comme le plus petit intervalle de temps pouvant être mesuré par le TDC.

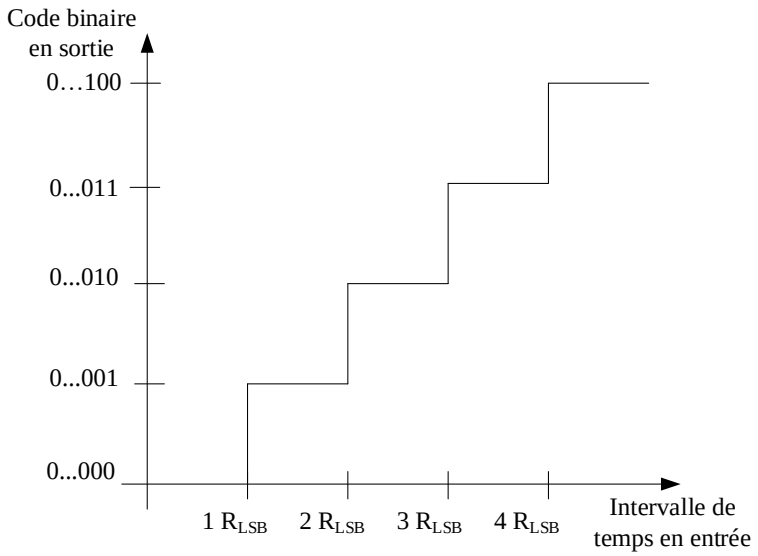



Figure 2.2 La caractéristique entrée-sortie d'un TDC idéal

La sortie du TDC est donnée par l'équation (2.2) :

$$TDC_{sortie} = R_{LSB} \times (\text{code binaire en sortie}) + \varepsilon \qquad (2.2)$$

où ε est l'erreur de quantification (ou bruit de quantification), définie comme la distorsion entre la sortie numérique et l'entrée temporelle du TDC. L'erreur de quantification doit se situer dans l'intervalle [0, $R_{LSB}$[ et elle est définie, comme mentionné dans [9][10], par l'équation (2.3) :

$$\varepsilon = \sqrt{\frac{R_{LSB\ départ}^2}{12} + \frac{R_{LSB\ arrêt}^2}{12}} = \frac{R_{LSB}}{\sqrt{6}} \qquad (2.3)$$

où $R_{LSB\ départ}$ est la résolution de l'interpolateur du signal de départ, $R_{LSB\ arrêt}$ est la résolution de l'interpolateur du signal d'arrêt et $R_{LSB}$ est la résolution du TDC qui comprend les deux interpolateurs. Il est clair d'après (2.3) que pour réduire l'erreur de quantification il faut améliorer la résolution.

### 2.1.2 Précision

Pour mesurer la précision d'un TDC, un intervalle temporel fixe est appliqué d'une manière répétitive à l'entrée du TDC. Dans le cas idéal où le bruit est considéré comme nul, chaque mesure devrait donner le même résultat. Or, le bruit cause une variation dans les résultats de mesure même

si l'intervalle temporel mesuré demeure inchangé. En général, ces mesures suivent une distribution gaussienne dont l'écart type (*standard deviation*) est appelé *Single Shot precision* (SSP). La SSP décrit la capacité du TDC à reproduire fidèlement la même mesure en présence du bruit, en d'autres termes, décrit la variabilité des mesures. Cette métrique dépend de l'intervalle temporel mesuré. En effet, chaque élément à délai introduit une certaine variation de délai et contribue donc à une incertitude temporelle. Plus l'intervalle temporel est long, plus le nombre d'éléments à délai parcourus est grand et par conséquent l'incertitude totale est grande. Si les variations des éléments à délais sont non corrélées alors l'incertitude temporelle croit avec un facteur de $\sqrt{T}$ [11] ($T$ étant l'intervalle temporel à mesurer).

Selon [9][10], la précision d'un TDC ($\sigma_{TDC}$) s'exprime par l'équation (2.4) :

$$\sigma_{TDC} = \sqrt{\sigma_q^2+\sigma_{INL}^2+\sigma_{CLK}^2+\sigma_{extra}^2} \tag{2.4}$$

où $\sigma_q$ est l'écart type causé par l'erreur de quantification, $\sigma_{INL}$ est l'écart type causé par la non-linéarité intégrale (INL), $\sigma_{CLK}$ est l'écart type causé par la gigue (*jitter*) de l'horloge du système et $\sigma_{extra}$ représente l'écart type causé par la gigue des autres signaux présents dans le TDC.

### 2.1.3 Exactitude

L'exactitude (*accuracy*) s'exprime par l'écart temporel entre la mesure retournée par l'instrument et la mesure réelle. Elle exprime la capacité de l'instrument de mesure à produire une valeur aussi proche que possible de la valeur réelle. Lorsque la précision est mauvaise, on ne peut pas se fier à une seule mesure pour estimer correctement l'exactitude. Un nombre élevé de mesures doit être effectué dans ce cas et la moyenne est retenue. Un instrument plus précis peut afficher une exactitude moins bonne qu'un instrument moins précis et vice-versa. Un exemple communément utilisé pour faciliter la différenciation entre la précision et l'exactitude est présenté à la Figure 2.3 où chaque point foncé correspond à une mesure et le point central correspond à la mesure cible (réelle). Dans les cercles à gauche, les mesures étant groupées et variant très peu, l'instrument de mesure est donc précis, par contre la moyenne des mesures est loin de la valeur réelle, il n'est donc pas exact. Dans les cercles à droite, les mesures sont espacées et varient relativement beaucoup, par contre en moyenne elles restent proches de la valeur réelle. L'instrument est donc exact, mais pas précis.

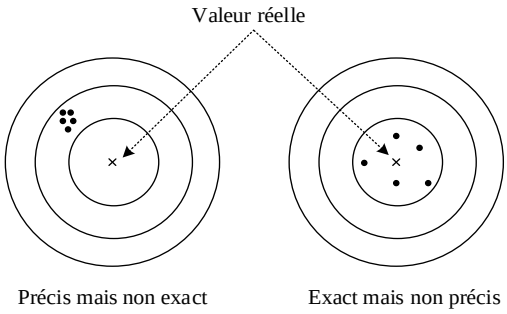



Figure 2.3 Illustration communément utilisée pour différencier la précision et l'exactitude

La moyenne quadratique (*Root Mean Square Deviation* RMSD) aussi appelée l'erreur de la moyenne quadratique (*Root Mean Square Error* RMSE) peut être utilisée pour décrire l'exactitude du TDC. C'est une mesure de la différence entre les valeurs données par le TDC et la valeur prédite telle que présentée par l'équation (2.5) :

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - x)^2}{N}} \tag{2.5}$$

où $N$ est le nombre de mesures, $x_i$ sont les sorties du TDC et $x$ est la valeur prédite.

### 2.1.4 Non-linéarité

La non-linéarité provoque une déviation de la fonction de transfert (caractéristique de quantification) par rapport à l'idéale. Plusieurs sources de non-linéarité existent, on cite par exemple les temps de propagation non uniformes à travers les étages du TDC, la fluctuation de la température et de la tension d'alimentation ainsi que la variation des procédés de fabrication. La non-linéarité se définie toujours par : la non-linéarité différentielle (DNL) et la non-linéarité intégrale (INL), tous deux exprimées en LSB.

- DNL

Comme communément défini, la DNL est ‹‹ La déviation d'un pas de quantification par rapport au LSB ››. Pour calculer la DNL, la méthode de densité des codes est souvent utilisée. Il s'agit d'envoyer un nombre $N$ très élevé d'intervalles temporels aléatoires à l'entrée du TDC et de construire un histogramme de chaque code numérique à sa sortie. Ceci permet ainsi de déterminer

la largeur de chaque pas de quantification. La DNL d'un pas de quantification *i* est alors donné par l'équation (2.6) :

$$DNL_i\,(LSB) = \frac{W_i - R_{LSB}}{R_{LSB}} \tag{2.6}$$

Avec : $W_i$ et $R_{LSB}$ étant respectivement la largeur du pas de quantification *i* et la résolution.

- INL

L'INL d'un pas de quantification *i* est la somme des DNL qui la précèdent et s'exprime par l'équation (2.7) :

$$INL_i(LSB) = \sum_{n=1}^{i} DNL_n\,(LSB) \tag{2.7}$$

Dans la littérature, on trouve également le temps mort et la plage dynamique parmi les caractéristiques principales d'un TDC. Le temps mort est le temps pendant lequel le TDC ne répond pas à d'autres entrées. Autrement dit, c'est le temps durant lequel le TDC est occupé à faire une mesure. Il comprend le temps de conversion, l'enregistrement des données dans une mémoire et la remise à zéro à la fin de la conversion. La plage dynamique est l'intervalle temporel maximal que le TDC est capable de convertir en une représentation numérique.

## 2.2 Survol des principales architectures de TDC

Plusieurs architectures de TDC ont été implémentées sur FPGA. Dans cette section, nous nous limiterons aux principales, à savoir les TDC à base de ligne à délai et les TDC bouclés. Une synthèse des TDC réalisés sur FPGA se trouve dans [12].

### 2.2.1 TDC à base de ligne à délai

Deux principales architectures à base de ligne à délai peuvent être distinguées : le TDC à base de ligne à délai simple et le TDC à base de ligne à délai Vernier.

#### 2.2.1.1 TDC à base de ligne à délai simple

C'est l'architecture la plus simple aussi appelée TDL (*Tapped Delay Line*). Tel que décrit dans la Figure 2.4, il s'agit de faire propager le signal « Départ » tout au long d'une ligne constituée de *n* étages à délai. La sortie de chaque étage à délai se connecte à une bascule D pour échantillonner

son état à l'arrivée du signal « Arrêt ». Le nombre d'étages parcourus par le signal « Départ » avant l'arrivée de « Arrêt » donne un code thermométrique qui sera traduit en représentation numérique par un encodeur. La position de la transition 1-0 dans le code thermométrique indique le nombre d'étages parcourus par « Départ » avant l'arrivée de « Arrêt », en d'autres termes indique l'intervalle de temps séparant « Départ » et « Arrêt ».

Cette architecture est la plus utilisée et la plus adaptée pour les FPGA. En effet, dans les FPGA il existe des chaînes de retenue dédiées pour accélérer les calculs et les comparaisons. Ces chaînes se caractérisent par un routage relativement régulier et par des délais de propagation d'un étage à un autre assez rapide ce qui permet d'atteindre une mesure à haute résolution.

La résolution dans ce cas correspond au temps de propagation à travers un élément à délai. La résolution dépend donc du nœud technologique utilisé pour concevoir l'élément à délai, de sa charge de sortie, mais aussi des fluctuations au niveau de la température et de la tension d'alimentation. Des TDC avec ligne à délai simple comportant des résolutions de 8.6 ps et 5.8 ps ont été implémentés dans un Kintex 7 [13] et Virtex 5 [14] respectivement.

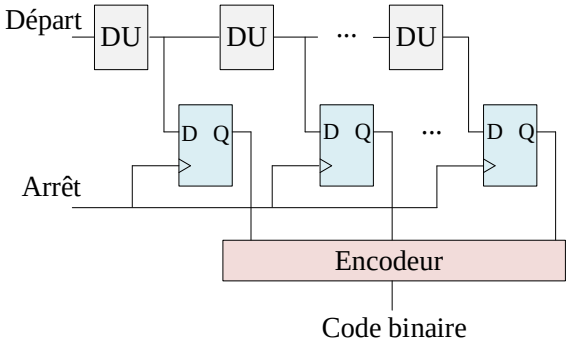



Figure 2.4 Ligne à délai simple

Selon [15][16], l'écart type causé par l'erreur de quantification $\sigma_q$ dans un TDL-TDC, est donné par l'équation (2.8) :

$$\sigma_q^2 = \sum_i \left( \frac{W_i^2}{12} \times \frac{W_i}{W} \right) \tag{2.8}$$

où $W_i$ est la largeur de la division $i$ et $W$ est la somme des largeurs de toutes les divisions ($W=\sum_i W_i$). À noter qu'un pas de quantification est aussi désigné dans la littérature par le terme division (*bin*). Ce terme sera aussi utilisé dans le reste de ce chapitre.

D'après l'équation (2.8), la contribution des divisions larges dans la détérioration de la précision est plus importante que celle des divisions moins larges. Dans l'équation (2.8) le terme $(\frac{W_i^2}{12})$ désigne l'erreur de quantification et le terme $(\frac{W_i}{W})$ désigne la probabilité que la mesure tombe dans la division *i*.

Une méthode répandue dans la littérature pour améliorer la linéarité des TDL-TDC implémentés sur FPGA est la calibration division par division [17]: une fois le TDC caractérisé, c'est-à-dire que les largeurs $W_k$ des divisions sont connues, chaque division *i* est calibrée à son centre pour avoir un délai égal à celui de l'équation (2.9) :

$$t_i = \frac{W_i}{2} + \sum_{k=0}^{i-1} W_k \tag{2.9}$$

Il a été démontré dans [17] que la précision est maximale lorsque les divisions sont calibrées à leurs centres. En général, une table de référence (*Look-Up table* LUT) est construite donnant pour chaque largeur de division mesurée par caractérisation son équivalent en délai après calibration. La LUT est mise à jour régulièrement pour tenir compte des variations de la température et de la tension d'alimentation.

La résolution de ce TDC se limite au délai de propagation à travers un étage à délai. Afin d'améliorer davantage la résolution, l'architecture Vernier a été proposée.

### 2.2.1.2 TDC à base de ligne à délai Vernier

Dans ce type de TDC, les deux signaux « Départ » et « Arrêt » se propagent à travers deux lignes à délai comme le montre la Figure 2.5. Le temps de propagation à travers un élément à délai de la première ligne est $\tau_1$ tandis que le temps de propagation à travers un élément à délai de la deuxième ligne est $\tau_2$ tel que $\tau_2 < \tau_1$. La résolution est donc de $\tau_1$-$\tau_2$. Plus $\tau_1$ et $\tau_2$ sont proches, meilleure est la résolution. Le signal « Départ » se propage à travers la ligne à délai lente et le signal « Arrêt » à travers la ligne à délai rapide. Le moment où « Arrêt » rattrape « Départ » correspond au temps recherché.

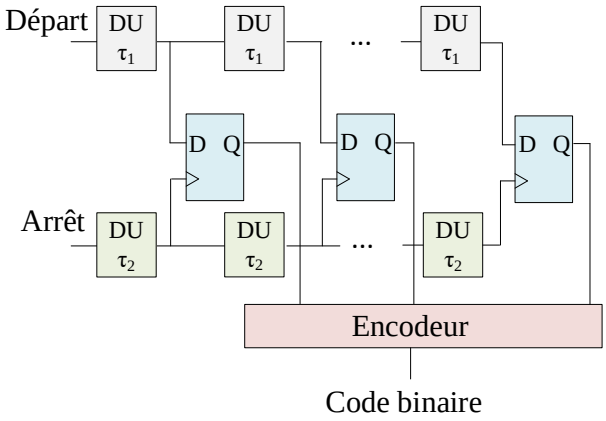



Figure 2.5 Ligne à délai Vernier

Ce type d'interpolateur améliore considérablement la résolution, qui dépasse le délai de propagation à travers une porte logique. En effet, la résolution ne dépend plus du délai à travers une porte, mais plutôt de la différence des délais des portes des deux lignes. Cette amélioration de la résolution vient au détriment des ressources et de la puissance consommées. En effet, tel que démontré dans [11], la surface occupée et la puissance augmente d'une manière linéaire avec l'intervalle de temps maximal à mesurer.

### 2.2.1.3 Limitations des TDC à base de ligne à délai

Une première limitation des TDC à base de ligne à délai est les ressources consommées. En effet, pour atteindre des plages dynamiques élevées, les lignes à délais doivent être longues donc consomment plus de cellules logiques non seulement dans la structure intrinsèque de la ligne, mais aussi au niveau de l'implémentation de l'encodeur et des circuits de post-traitement. D'autre part, une ligne à délai qui s'étale sur plusieurs CLB (*Configurable Logic Block*) dans un FPGA engendre une non-linéarité élevée. En effet, les divisions se trouvant à la frontière d'un CLB sont beaucoup plus larges que les autres divisions. Dans [17], on mentionne qu'une division typique dans une chaîne de retenue est de l'ordre de 60 ps tandis que les divisions localisées à la frontière du CLB atteignent 160 ps (FPGA Altera Cyclone II EP2C8T144C6). Afin d'améliorer la précision de mesure, une pratique courante consiste à utiliser des TDC à multiples canaux, c'est-à-dire mesurer le même intervalle de temps avec plusieurs canaux de TDC et la mesure finale retenue est la moyenne des mesures fines retournées par tous les canaux. Hors, les architectures à multiples canaux viennent avec ces deux principales limitations :

- Implémenter de multiples canaux de TDC à proximité dans un même FPGA pourrait dégrader leurs performances. Dans [18], Menninga et *al* ont réalisé deux implémentations. Dans la première la ligne à délai est protégée par des *slices* vides de part et d'autre, et dans la deuxième la ligne à délai est placée entre deux autres lignes sans laisser d'espace vide. Trois essais de mesures de DNL et INL ont été effectués pour chacune des deux implémentations. Dans la deuxième implémentation, les mesures sont extraites à partir de la ligne à délai se trouvant au milieu. Dans la première implémentation, il n'y a pas eu de changement considérable d'une mesure à une autre tandis que dans la deuxième, les mesures diffèrent aléatoirement d'un essai à un autre. Ceci est dû à l'effet imprévisible causé par la logique entourant la ligne du milieu. Espacer les trois lignes à délai en insérant des *slices* vides entre elles, contribue donc à diminuer la fluctuation du DNL et INL.

- Des TDC placés dans différentes régions du FPGA présentent des performances différentes même s'ils sont conçus exactement de la même manière. Ceci a été démontré dans [19]. Un total de 161 canaux de TDC a été implémenté sur un Virtex-6 et les mesures de DNL et INL ont été extraites pour chaque canal. Le DNL est plus large à y=80 et y=160 (Le FPGA est vu comme une matrice de blocs logiques ayant chacune des coordonnées x et y). Le INL fluctue entre 8 et 4 LSB selon la position. Ces variations du INL et DNL sont causées par les différentes régions d'horloges constituant le FPGA tel que présenté dans [20]. En effet, les temps de propagation du signal d'horloge depuis le tampon global (*global buffer*) situé au milieu de l'FPGA vers les différentes régions ne sont pas identiques signifiant que les bascules (l'échantillonneur du TDC) ne sont pas déclenchées en même temps. Le Virtex-6 (ML605) de Xilinx par exemple, contient 12 régions d'horloges avec 40 CLB chacune. Le biais d'horloge (*clock skew*) entre deux CLB adjacents situés dans la même région d'horloge est estimé à 2 ps alors que le biais d'horloge entre deux CLB situés dans deux régions d'horloges différentes serait de 100 ps.

Pour pallier les problèmes rencontrés avec les lignes à délai, les TDC à base de boucle ont été introduits.

### 2.2.2 TDC à base de boucle

Afin de réduire la non-linéarité et l'espace occupé par un TDC à base de ligne à délai tout en maintenant une résolution et une plage dynamique élevées, le RO-TDC (*Ring Oscillator* TDC) a été introduit. Il s'agit d'utiliser les mêmes éléments à délai de manière itérative au lieu de rallonger

la chaîne. La ligne à délai est donc configurée en oscillateur en anneau avec un compteur enregistrant le nombre d'oscillations effectué.

L'architecture de base d'un RO-TDC est présentée à la Figure 2.6. À noter que cette architecture a été implémentée sur un ASIC [21].

La ligne de DU (*Delay Unit*) est configurée en anneau en connectant la sortie du dernier DU à l'entrée du premier DU. La sortie de n'importe quel DU est connectée à l'entrée horloge d'un compteur enregistrant le nombre de tours effectués à travers l'oscillateur. Des bascules D sont connectées à la sortie des DU et au compteur pour échantillonner l'état de la ligne et du compteur à l'arrivée du signal « Départ » puis à l'arrivée du signal « Arrêt ». L'oscillateur est en libre fonctionnement dans cette architecture. Il y a donc deux niveaux d'interpolations : une interpolation grossière de résolution associée à la période d'oscillation (donnée par le compteur) et une interpolation fine de résolution égale au délai de propagation à travers un DU (donnée par l'oscillateur). Le temps ($T_{Départ\text{-}Arrêt}$) séparant le signal « Départ » du signal « Arrêt » est donné par l'équation (2.10) :

$$T_{Départ\text{-}Arrêt} = (SO_{Arrêt} - SO_{Départ}) \times \tau + ((SC_{Arrêt} - SC_{Départ})/f_{RO}) \quad (2.10)$$

Avec :

- $SO_{Départ}$ et $SO_{Arrêt}$ : les sorties des unités de délai constituant l'oscillateur à l'arrivée de « Départ » et « Arrêt » respectivement.
- $\tau$ : le délai de propagation à travers une unité à délai de l'oscillateur.
- $SC_{Départ}$ et $SC_{Arrêt}$ : les sorties du compteur à l'arrivée de « Départ » et « Arrêt » respectivement.
- $f_{RO}$ : la fréquence d'oscillation.

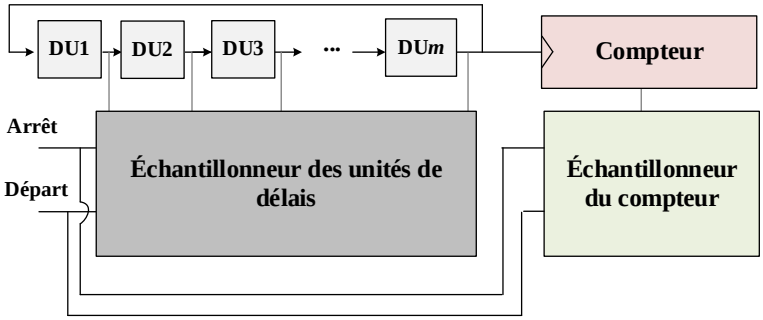



Figure 2.6 Circuit d'un RO-TDC basique

Un RO-TDC comportant deux oscillateurs (Osci1 et Osci2) a été implémenté sur FPGA (Figure 2.7) [12]. Dans cette architecture, Osci1 se déclenche à l'arrivée de « Départ » tandis que Osci2 se déclenche à l'arrivée de « Arrêt ». Osci2 a une période plus courte que Osci1. Les compteurs 1 et 2 s'incrémentent respectivement à chaque période d'oscillation de Osci1 et Osci2. Les valeurs des compteurs sont capturées au moment où les deux oscillateurs deviennent en phase. La sortie du TDC est donnée par l'équation (2.11) :

$$T_{\text{Départ-Arrêt}} = (n_1 - 1) \times T_1 - (n_2 - 1) \times T_2 \qquad (2.11)$$

Avec :

- $n_1$ et $n_2$ les valeurs des compteurs 1 et 2 respectivement,
- $T_1$ et $T_2$ les périodes d'oscillations de Osci1 et Osci2 respectivement.

À noter qu'un seul compteur peut également être utilisé qui s'incrémente à chaque période d'oscillation de Osci1 (oscillateur lent) et s'arrête lorsque Osci2 rattrape Osci1.

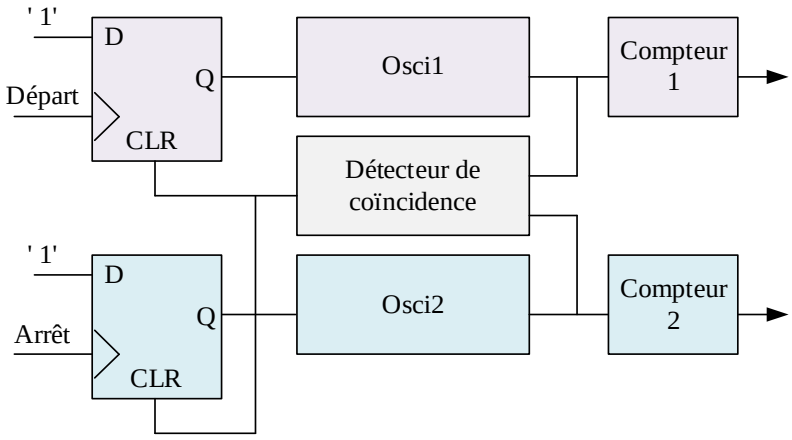



Figure 2.7 RO-TDC implémenté sur FPGA

Une telle implémentation a été réalisée sur un Stratix III atteignant une résolution de 31 ps et une non-linéarité intégrale maximale de 0.091 LSB [22].

À noter qu'un TDC compact à base de désérialiseurs inclus dans les blocs d'entrée/sortie (I/O) du FPGA a été implémenté par Arpin et *al* [80]. Bien que ce TDC consomme peu de ressources, la résolution atteinte demeure faible (321.5 ps) et les ressources I/O sont très limitées dans un FPGA.

## 2.3 Ajustement fin des délais dans un FPGA de Xilinx

Dans [23], une méthode pour contrôler finement le délai d'une trace dans un FPGA entre un point source et un point destination est proposée. Il s'agit d'utiliser les matrices de commutation pour changer le chemin emprunté par la trace. En effet, certains ports d'entrée d'une matrice de commutation permettent de rebondir le signal avant de le diriger vers un port de sortie. Il existe donc plusieurs traces possibles à l'intérieur de la matrice pouvant connecter un port d'entrée à un port de sortie. Plus le nombre de traces possibles est élevé plus leurs délais différentiels sont rapprochés. Un ensemble de 546 traces a permis d'avoir un délai différentiel de 18 ps sur une plage de 947 ps. Dans [24], le nombre de traces possibles a été augmenté à 40 000 et le délai différentiel entre les traces a atteint la picoseconde sur une plage de 2,5 ns. Comme chaque trace est une nouvelle configuration qui demande un nouveau fichier (.bit) pour programmer les cellules SRAM du FPGA, la reconfiguration dynamique a été utilisée. De plus, le temps requis pour extraire les délais de propagation de toutes les traces est relativement long (estimé à quelques heures dans [24]). Cette méthode est basée sur la programmation des points d'interconnexions, rendue possible grâce à une bibliothèque développée en C (nommée DXCL). Dans [25][26], on trouve une autre méthode pour contrôler finement le délai d'une trace en utilisant l'architecture interne d'une LUT, constituée en réalité d'un ensemble de multiplexeurs. Une LUT à six entrées est configurée en un inverseur et seule une entrée est considérée tandis que les cinq autres sont des '*don't cares'* mais affectent la propagation du signal à l'intérieur de la LUT. Cette méthode permet de produire des traces avec des délais différentiels inférieurs à la picoseconde, mais sur une courte plage de seulement 10 ps [25]. Ces deux méthodes trouvées dans la littérature se basent sur le changement du chemin emprunté par la trace pour contrôler son délai de propagation. Dans [27], une technique d'ajustement des délais basée sur la régulation de charge en connectant la ligne à ajuster à des tampons *tri-state* a été présentée. Cette technique, bien qu'elle permet un contrôle fin (entre 2,4 ps et 5,5 ps par tampon ajouté), nécessite un routage manuel. Aucun mécanisme d'automatisation n'a été présenté.

Dans ce chapitre, des concepts de base reliés à la conversion numérique de temps ont été présentés. Ces concepts seront utilisés dans les chapitres qui suivent. Vu le nombre élevé de publications sur les convertisseurs temps-numérique avec des architectures différentes dédiées chacune pour une

application spécifique, faire un inventaire exhaustif de tous ces TDC serait trop long. Nous nous sommes limités aux TDC à base de ligne à délai et aux TDC bouclés. Un survol de ces deux classes de TDC, de leurs avantages ainsi que leurs limitations a été fait dans ce chapitre. Finalement, les principaux travaux qui rapportent un ajustement à grains fins des délais dans un circuit programmable ont été présentés.

# CHAPITRE 3 DÉMARCHE DE L'ENSEMBLE DU TRAVAIL DE RECHERCHE

Les articles publiés et soumis au sein de ce travail de recherche sont intégrés dans ce document dans les chapitres 4, 5, 6 et 7. Ce chapitre indique les liens entre ces articles ainsi que leur cohérence par rapport aux objectifs de la recherche mentionnés dans le chapitre 1.

Le premier objectif est l'élaboration d'une nouvelle architecture d'un outil de mesure de temps sur FPGA compacte et supportant de multiples mesures. Une preuve de concept d'un tel outil a été présentée dans l'article de conférence intitulé - A multi-measurements RO-TDC implemented in a Xilinx Field-Programmable Gate Array (ISCAS 2017) - et intégré dans le chapitre 4 de ce document. Dans ce travail, une nouvelle architecture de TDC basée sur une topologie bouclée permettant la réutilisation des ressources, est présentée. Cette architecture est également distribuée supportant plusieurs points de mesure avec un oscillateur partagé entre tous les points de mesures. L'originalité de cette architecture consiste à utiliser les mémoires implémentées par de simples tables de commutation (*Look-Up Table*) pour stocker la mesure fine de temps. Une étude théorique comparant les ressources estimées avec les ressources utilisées par un TDC bouclé classique a permis de conclure que le TDC proposé est compact. Cet article satisfait l'objectif 1 « Proposer et concevoir des convertisseurs temps-numérique dans un circuit programmable compacts et distribués qui se veulent des circuits de support in situ pour la mesure de délai ».

Lors de l'extraction des mesures temporelles par simulation du TDC proposé dans l'article 1, on a remarqué qu'il souffre d'une non-linéarité élevée causée par les délais arbitraires des traces créées lors de la phase de routage par l'outil de développement de Xilinx (Vivado). Par conséquent, une élaboration d'outil de contrôle fin du routage dans un circuit programmable est indispensable pour améliorer la linéarité du circuit proposé de mesure des délais. Une preuve de concept d'un tel outil a été publiée dans l'article de conférence intitulé - Sub-ps resolution programmable delays implemented in a Xilinx FPGA (MWSCAS 2017) - et intégré dans le chapitre 5 de ce document. Ce travail démontre qu'il est possible de rallonger le délai de propagation à travers une trace à la picoseconde près sans apporter de modification au design déjà placé et routé. La technique consiste

à rajouter de la charge capacitive au niveau d'un port d'entrée d'une matrice de commutation par lequel passe la trace en question. La charge est augmentée en ajoutant des interconnexions non utilisées. Il a été démontré par des résultats expérimentaux réalisés sur un FPGA de la famille 7-series de Xilinx (ZYNQxc7z010-3clg400) qu'une combinaison de 15 interconnexions peut fournir un délai supplémentaire avec une résolution à la sous-ps dans une plage de 2,17 ps à 48,6 ps. Cet article satisfait l'objectif 2.a « Proposer et générer des outils pour contrôler finement les délais dans un FPGA ».

Comme le but de l'outil de contrôle des délais proposé dans le chapitre 5 était d'améliorer la linéarité trouvée insatisfaisante du convertisseur présenté au chapitre 4, il fallait donc trouver une méthodologie pour appliquer cet outil à ce convertisseur. Une telle méthodologie est élaborée dans l'article de revue intitulé – A fine resolution delay tuning method to improve the linearity of an unbalanced time-to-digital converter on a Xilinx FPGA (IET Circuits, devices and systems 2020) - et intégré dans le chapitre 6 de ce document. Dans cet article, des algorithmes en TCL sont développés pour automatiser la création des interconnexions augmentant la charge capacitive des ports des matrices de commutation. Des algorithmes de parcours en largeur (*Breadth First search*) y sont notamment présentés. Une stratégie pour rendre ces algorithmes applicables à tout FPGA de Xilinx supporté par l'outil de développement Vivado (Virtex-7, Kintex-7, Artix-7, Zynq7000 et Ultrascale) est présentée. Il a été démontré par des résultats expérimentaux conduits sur un FPGA 7-series de Xilinx (ZYNQxc7z010-3clg400) que les algorithmes ont permis de réduire la non-linéarité différentielle de 0,51 LSB à 0,05 LSB et la non-linéarité intégrale de -0,54 LSB à 0,06 LSB. Cet article de revue satisfait l'objectif 2.b « Élaborer une méthodologie d'implémentation de ces outils dans des circuits programmables et donnant des résultats reproductibles ».

Finalement, dans le but d'obtenir une exactitude et une précision des mesures compétitives avec ce qui est récemment publié dans la littérature, une stratégie de parallélisation a été mise en place pour ce convertisseur. Ce travail est présenté dans l'article de revue intitulé - Ring-Oscillator Based High Accuracy Low Complexity Multichannel Time-to-Digital Converter Architecture for Field-Programmable Gate Arrays (soumis à IEEE Transactions on Instrumentation and measurement) - et intégré dans le chapitre 7 de ce document. Dans ce travail, une implémentation à multiples canaux a été proposée pour ce TDC afin d'améliorer la précision de mesure. D'autre part, les algorithmes d'ajustement de délai ont été appliqués avec une stratégie visant à augmenter

l’exactitude des mesures, une métrique à nos connaissances, rarement rapportée dans la littérature. Il a été démontré que cette stratégie a permis d’atteindre une exactitude de 92,9 ps. Le TDC final consomme 1,49% de LUT et 1,31% des bascules disponibles dans le FPGA (ZYNQxc7z010-3clg400) pour 9 canaux. Ceci est nettement inférieur aux ressources consommées par les TDC publiés. Cet article satisfait l’objectif 2 « Mettre en place des stratégies de calibration et d’amélioration des performances temporelles ».

# CHAPITRE 4 ARTICLE 1 - A MULTI-MEASUREMENTS RO-TDC IMPLEMENTED IN A XILINX FIELD-PROGRAMMABLE GATE ARRAY

**Abstract**

In this paper, an area efficient time to digital converter (TDC) performing measurements between multiple hit signals is proposed. Our TDC is based on a delay line configured as a ring oscillator and a round tracker to count the number of iterations through the oscillator. Lookup tables configured as distributed RAMs and shift registers are used to sample the oscillator and the round tracker states whenever a transition on a signal occurs. A theoretical study is elaborated to estimate FPGA resources required to implement the proposed TDC in comparison with a multichannel basic RO-TDC. It is shown that the gain in the number of Flip-Flops and Lookup tables can reach factors of 85 and 2.1 respectively in an architecture made of a six-stage oscillator, a 32-state round tracker and 20 input hit signals. Temporal characteristics extracted from our TDC implemented in a Xilinx ZYNQ family FPGA are reported.

Keywords: Time-to-digital converter (TDC); Multi-measurement; area saving.

**Authors:** Safa Berrima, Yves Blaquière and Yvon Savaria
**Published in:** IEEE International Symposium on Circuits and Systems (ISCAS) 2017 [28]

## 4.1 Introduction

Time to digital converters (TDCs) are widely used as high precision instruments for time measurements in many science and engineering applications. For instance, TDCs are used in biomedical imaging applications for Time-of-flight (TOF) measurements in positron emission tomography (PET) [1] and fluorescence lifetime imaging microscopy (FLIM) [29]. TDCs are also used in many electrical engineering applications, such as digital oscilloscopes and all digital phase-locked-loops (ADPLLs) [30]. Fully digital TDCs have been implemented in Field Programmable Gate Arrays (FPGAs) taking advantage of their flexibility, low development costs and fast development cycles. Several FPGA-based TDC architectures have been reported in the literature. A basic TDC is based on a Tapped Delay Line (TDL) [15][31]. The main idea is to propagate a

start signal along a chain of delay units, generally a carry chain. The state of that chain is sampled when a stop signal arrives. Usually, the time resolution of the TDL architecture is determined by the propagation delay of the delay unit. In order to improve their resolution, some TDC circuits use the VDL (Vernier Delay Line) architecture [32], where both start and stop signals propagate in two chains composed of slightly different delay units. The VDL architecture allows achieving a resolution smaller than one gate delay, but suffers from a limited dynamic range. Several other approaches to implement FPGA-based TDCs also exist, such as the pulse shrinking method [33], the multi-phase clock sampling [34] and the multiple interpolation [35]. Some approaches combine fine measurement (sub-clock cycle) circuits with coarse counters driven by a reference clock to measure time intervals with the resolution of a clock period. Another type of TDC architecture that improves the logic fill-factor is called the ring oscillator TDC (RO-TDC) [10]. In a basic RO-TDC, the delay line is configured in a ring format such that transitions propagate iteratively through the ring. Thus instead of using a longer delay chain, the same delay elements can be re-used until the conversion completes. A counter is used to track the number of rounds. In this work, we propose a multi-measurement, i.e. multi-hit, RO-TDC, called MMRO-TDC, implemented in a Xilinx FPGA. Our proposed TDC measures simultaneously relative delays between transitions on multiple signals. In order to measure relative delays between transitions on multiple signals, one can replicate the same TDC architecture, i.e. using a multichannel TDC. In many applications where area saving is critical, such as in 3D integration, replication can be cumbersome. In our work, we propose an RO-TDC capable of carrying out measurements between multiple hits without resorting to replication. Our proposed TDC saves resources compared to replication by using distributed RAMs and shift registers available in Xilinx FPGAs to sample the content of the oscillator and to track the number of rounds for each measurement.

The next section analyzes the resources required in a multichannel basic RO-TDC architecture. In section 4.3, our proposed MMRO-TDC is presented as well as its required resources. Section 4.4 shows area savings of the MMRO-TDC in comparison with a multichannel RO-TDC. It also highlights selected temporal results obtained from an implementation using a ZYNQxc7z010 Xilinx FPGA. The main conclusions of this work are summarized in section 4.5.

## 4.2 Required resources in a multichannel RO-TDC

A multi-measurement RO-TDC can be obtained by simply replicating channels. This section evaluates the resources needed if such a multichannel basic RO-TDC is implemented in an FPGA. Resources are expressed in terms of number of Lookup tables (LUTs), Flip-Flops (FFs) and delay units (DUs) making up the oscillator. Delay units can be implemented with LUTs or with dedicated carry logic for a better resolution. In order to measure the relative delays between transitions on $k$ signals under test ($SUT_1$, $SUT_2$, … $SUT_k$), each channel can measure the time interval between $SUT_i$ and $SUT_{i+1\ (1\leq i\leq k)}$ for a total of ($k$-1) channels. Thus, the time interval between any $SUT_{i\ (1\leq i\leq k)}$ and $SUT_{j\ (1\leq j\leq k,\ j\neq i)}$ can be deduced. Consider a one channel basic RO-TDC as referred to in [10][21] and shown in Figure 4.1. The oscillator is working in a free running mode. A round tracker counts the number of oscillator rotations. DUs outputs and the round tracker states are sampled at the rising edge of $SUT_1$ and sampled again at the rising edge of $SUT_2$. The change of state during the interval between the rising edges on $SUT_1$ and $SUT_2$ represents the quantized time difference. In the basic RO-TDC (Figure 4.1), the oscillator is composed of $m$ DUs and the round tracker is a $n$-state counter. A n-state binary counter is made of $\log_2(n)$ FFs and log2($n$) LUTs. The counter sampler and the oscillator sampler contain $2\log_2(n)$ FFs and $2m$ FFs respectively. A ($k$-1) channel RO-TDC replicates this architecture ($k$-1) times.

Tableau **4.1** summarizes the required resources per RO-TDC.

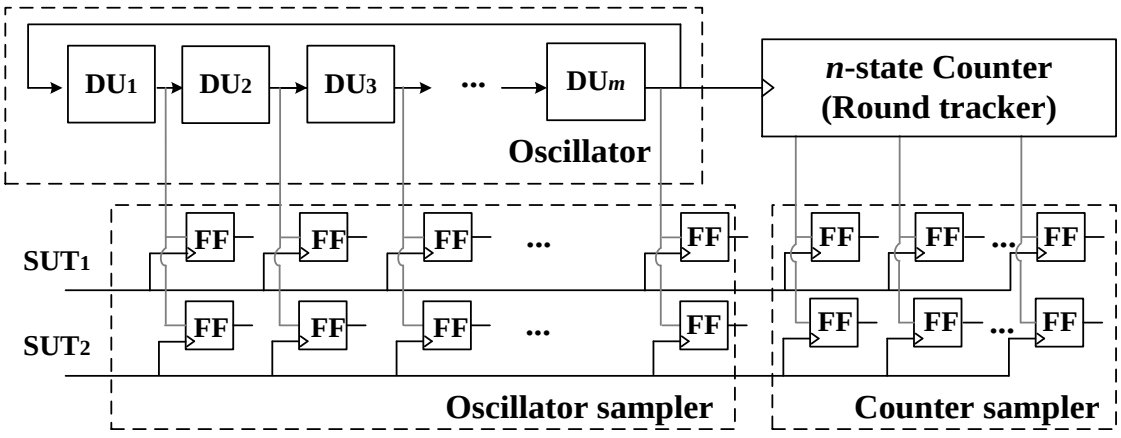



Figure 4.1 A basic RO-TDC circuit as refered to in [10][21]

Tableau 4.1 Resources used in a RO-TDC (Figure 4.1) where *m* and *n* are the number of DUs and states in the binary counter respectively

| Block | Number of required resources | | |
|---|---|---|---|
| | FFs | LUTs | DUs |
| *n*-state counter | $\log_2(n)$ | $\log_2(n)$ | 0 |
| counter samplers | $2\log_2(n)$ | 0 | 0 |
| oscillator | 0 | 0 | *m* |
| oscillator sampler | 2 *m* | 0 | 0 |
| Total | $3\log_2(n) + 2m$ | $\log_2(n)$ | *m* |

## 4.3 Proposed Multi-Measurements RO-TDC

### 4.3.1 Overview of the proposed architecture

The target MMRO-TDC consists of four blocks (Figure 4.2): the oscillator trigger, the oscillator, the sampler and the round tracker, described in the following.

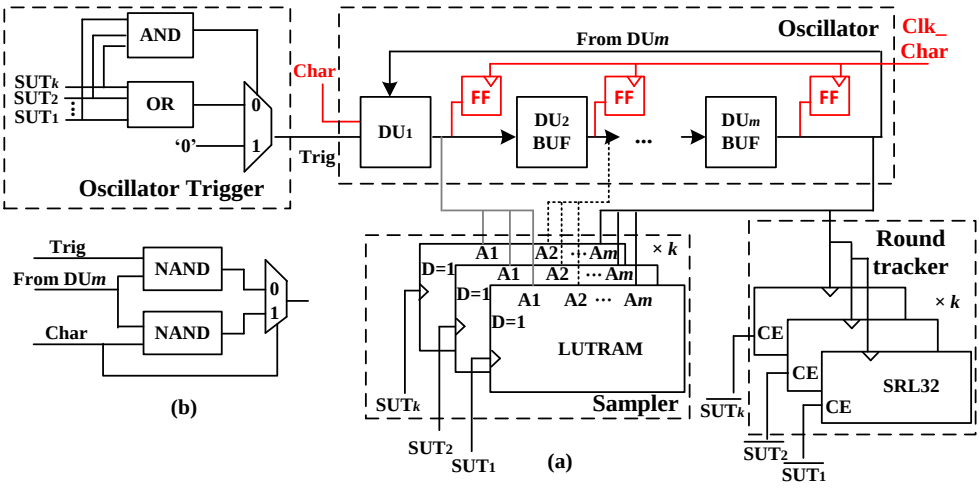



Figure 4.2 The MMRO-TDC (a) Detailed architecture (the red part is for characterization purpose) (b) First Delay Unit ($DU_1$) circuitry

#### 4.3.1.1 Oscillator trigger

This block activates the oscillator only during measurement to reduce power consumption [36]. It receives, as inputs, *k* transition signals under test ($SUT_1$, $SUT_2$, … $SUT_k$) initially at '0'. On the first $SUT_{i\ (1\leq i\leq k)}$ rising edge occurrence, Trig is asserted and the oscillation begins. On the last $SUT_{j\ (1\leq j\leq k,\ j\neq i)}$ rising edge occurrence, the output of the AND gate toggles to one and the oscillation stops.

#### 4.3.1.2 Oscillator

The ring oscillator is made of *m* cascaded delay units ($DU_1$, $DU_2$, ... $DU_m$). $DU_2$ to $DU_m$ are implemented as buffers while the $DU_1$, depicted in Figure 4.2 (b), controls two operation modes: in the self-characterization mode [37] (char='1'), the oscillator is enabled regardless of the state of Trig; in the normal mode (char='0'), the circuit oscillates when Trig='1'. Notice that in the self-characterization mode, aiming to estimate the real bins' width of the oscillator, there is no need to use an external oscillator asynchronous to the system clock (Clk_Char) to generate hits for histogram building. The oscillator made by the *m* DUs can be used for this purpose.

#### 4.3.1.3 Sampler

The sampler includes one Xilinx Distributed RAM (LUTRAM) per $SUT_i$ that samples the state of the oscillator on transition occurrence on $SUT_i$. The *m* DUs outputs define the LUTRAM address, while $SUT_i$ its sampling clock. The arrival of the first transition on any $SUT_{i\ (1\leq i\leq k)}$ triggers the oscillation then, a logic one is written in $LUTRAM_{j\ (1\leq j\leq k)}$ to the address given by the DUs outputs on the rising edge of $SUT_j$.

#### 4.3.1.4 Round tracker

The basic dynamic range of the oscillator is defined by its delay line and its oscillation period. To extend it, a *n*-state round tracker counts the number of rounds until the rising edge on $SUT_{i\ (1\leq i\leq k)}$. Each $SUT_{i\ (1\leq i\leq k)}$ is thus connected to the enable pin of a binary counter or a shift register logic (SRL). A SRL is more compact than a binary counter when *n* is small.

### 4.3.2 Required resources for the proposed MMRO-TDC

This section estimates the resources required to implement the MMRO-TDC (Figure 4.2) in Xilinx FPGAs as a function of *k*, *m*, and *n*. Resources needed for each of the four blocks are summarized in Tableau 4.2 and are compared to those needed for the multichannel basic RO-TDC in section 4.4.

**Oscillator trigger:** This block is mainly based on two *k*-input logic gates (AND/OR) and one 2×1 multiplexer. One *k*-input logic gate mapped to a tree of 6-input LUT can be built using (floor((*k*-2)/5)+1) LUTs where the floor( ) function returns the nearest integer below the operand value. One extra LUT is needed for the 2×1 Mux, for a total of (2×(floor((*k*-2)/5)+1))+1 LUTs if no logic optimization is considered.

**Oscillator:** Only one oscillator composed of *m* DUs is shared among the *k* samplers, where each DU can be made with a LUT or carry4 block for smaller delay unit. Notice that *m* FFs are needed to extract bins' width of the oscillator in a self-characterization mode.

**Sampler:** This block is made of *k* LUTRAMs. According to [38], a 64×1-bit LUTRAM occupies one LUT and multiple LUTs are combined to make larger RAMs. In general, $2^{m-6}$ LUTs are needed to build an *m*-bit address LUTRAM.

**Round tracker:** Shift registers are used to count the number of rounds. In fact, some Xilinx LUTs can be configured as 32-bit shift registers (SRL32s) without using the Flip-Flops available in the slice [38]. Each SRL32 can delay data from one to 32 clock cycles. Adjacent SRLs can then be cascaded to make larger shift registers. In general, a *n*-bit SRL can be built with (*n*/32) LUTs. The *k* SRLs in the MMRO architecture are then mapped into (*k*×*n*)/32 LUTs. Thus a shift register based on SRL32 units uses less resources than a binary counter for $n \leq 256$ states.

Tableau 4.2 Summary of FPGA resources for the MMRO-TDC where *m* is the number of DUs, *n* is the states number of the round tracker and *k* is the number of signals under test

| Block | Number of FPGA resources | | |
|---|---|---|---|
| | FF | LUT | DU |
| Oscillator trigger | 0 | $(2\times(\text{floor}((k\text{-}2)/5)+1))+1$ | 0 |
| Oscillator | $m$ | 0 | $m$ |
| Oscillator sampler | 0 | $k\times2^{m-6}$ if $m \geq 6$; $k$ if $m \leq 6$ | 0 |
| Round tracker | 0 | $(k\times n)/32$ | 0 |
| Total | $m$ | $(2\times(\text{floor}((k\text{-}2)/5)+1))+1+(k\times n)/32+k\times2^{m-6}$ } if $m \geq 6$; $(2\times(\text{floor}((k\text{-}2)/5)+1))+1+(k\times n)/32+k$ } if $m \leq 6$ | $m$ |

## 4.4 Performance evaluation results

### 4.4.1 Estimation of FPGA resources

The main objective of the MMRO-TDC is to minimize the FPGA resources to measure simultaneously relative delays between transitions on multiple signals under test. To prove its area savings feature, its resources are compared to those needed when a basic multichannel RO-TDC is replicated for each pair of signals. The ratio of the number of LUTs, FFs and DUs, named $G_{LUT}$, $G_{FF}$ and $G_{DU}$ respectively for the multi-channel RO-TDC over MMRO-TDC are estimated in equations (4.1), (4.2) and (4.3). $G_{LUT}$ and $G_{FF}$ are plotted in Figure 4.3 and Figure 4.4 for different ranges of *k*, *n* and *m*. In Figure 4.3, $G_{LUT}$ is plotted for $n \in [1,32]$, $k \in \{5,10,15,20,36\}$ and $m \leq 6$. $G_{LUT}$ is smaller than one for $n \in \{1,2\}$ (gray zones in Figure 4.3). $G_{LUTmax}$ identified on the graph is 1.5, 1.8, 1.9, 2 and 2.1 for $k$=5, 10,15, 20 and 36 respectively.

$$G_{DU} = k\text{-}1 \tag{4.1}$$

$$G_{FF} = \frac{(k\text{-}1)(3\log_2(n)+2m)}{m} \tag{4.2}$$

$$G_{LUT} = \frac{(k\text{-}1)(\log_2(n))}{(2\times(\text{floor}(\frac{k\text{-}2}{5})+1))+1+k\times2^{m-6}+\frac{kn}{32}} \quad m \geq 6$$

$$G_{LUT} = \frac{(k\text{-}1)(\log_2(n))}{(2\times(\text{floor}(\frac{k\text{-}2}{5})+1))+1+k+\frac{kn}{32}} \quad m \leq 6 \tag{4.3}$$

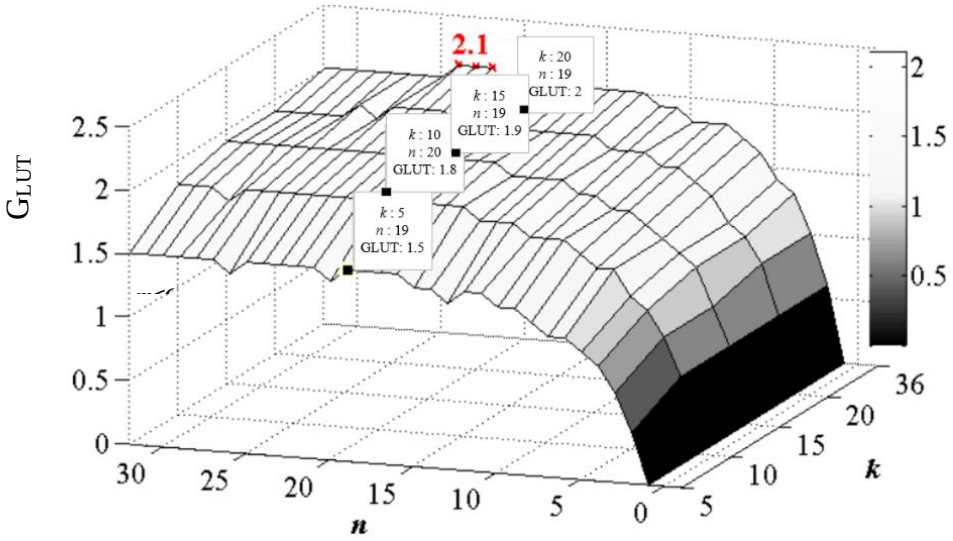



Figure 4.3 Ratio of the number of LUTs, $G_{LUT}$, depending on $k$ and $n$

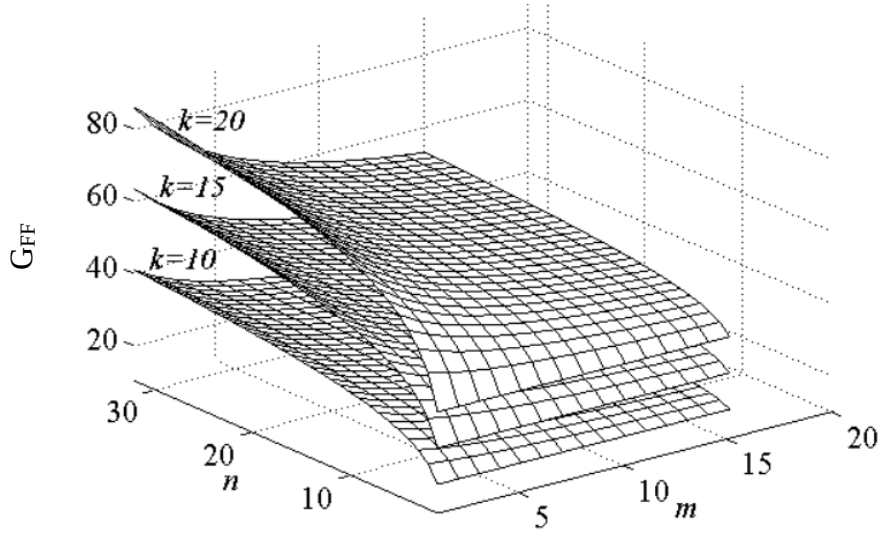



Figure 4.4 Ratio of the number of Flip-Flops, $G_{FF}$, depending on $k$, $n$ and $m$

$G_{LUTmax}$ remains at 2.1 for $k \geq 36$ (not shown in Figure 4.3). It has also been found that for $m \geq 8$, $G_{LUT}$ is smaller than one, regardless of $k$ and $n$ due to the dominance of $k \times 2^{m-6}$, the number of LUTRAMs, in Eq. (4.3). According to Figure 4.4, $G_{FF}$ decreases as $m$ increases and reaches its maximum when $n$ is maximum. The maximum of $n$ is equal to 32 in Figure 4.4 but higher values are possible. The gain $G_{FF}$ reaches 40.5 when $k$=10, $n$=32 and it increases as $k$ increases (reaches 85.5 when $k$=20). Three implementations (Impl1{$n$=32, $m$=6, $k$=10}, Impl2{$n$=32, $m$= 6, $k$=15} and Impl3{$n$=32, $m$=6, $k$=20}) were tested in a Xilinx ZYNQxc7z010-3clg400 FPGA, with a LUT-based oscillator. It has been found that the total numbers of LUTs are equal respectively to 28, 40

and 51 for Impl1, Impl2 and Impl3. These numbers are slightly less than the estimated ones (31, 43, 55) due to the optimization/compaction performed by the synthesis tool. Notice that this resource estimation is applicable to any Xilinx FPGA with 6-input LUTs and SRL32s, from Spartan 6 to the UltraScale+ families.

### 4.4.2 Temporal performances of the MMRO-TDC

This section presents temporal results in terms of bin width, DNL and INL, obtained with our proposed MMRO-TDC. These results were deduced from the static timing analyzer report generated after place and route in both fast and slow processes. The design was implemented in a Xilinx ZYNQxc7z010-3clg400 FPGA with $n$=32, $m$=6, $k$=10 and a LUT-based oscillator. The bin widths showed in Figure 4.5 correspond to those of the oscillator and those seen at LUTRAMs level. From Figure 4.5, bin widths are on average 296 ps and 147 ps in the slow and fast process corners respectively. Bin widths are affected by delays of oscillator's DUs and interconnects delays up to LUTRAMs address pins.

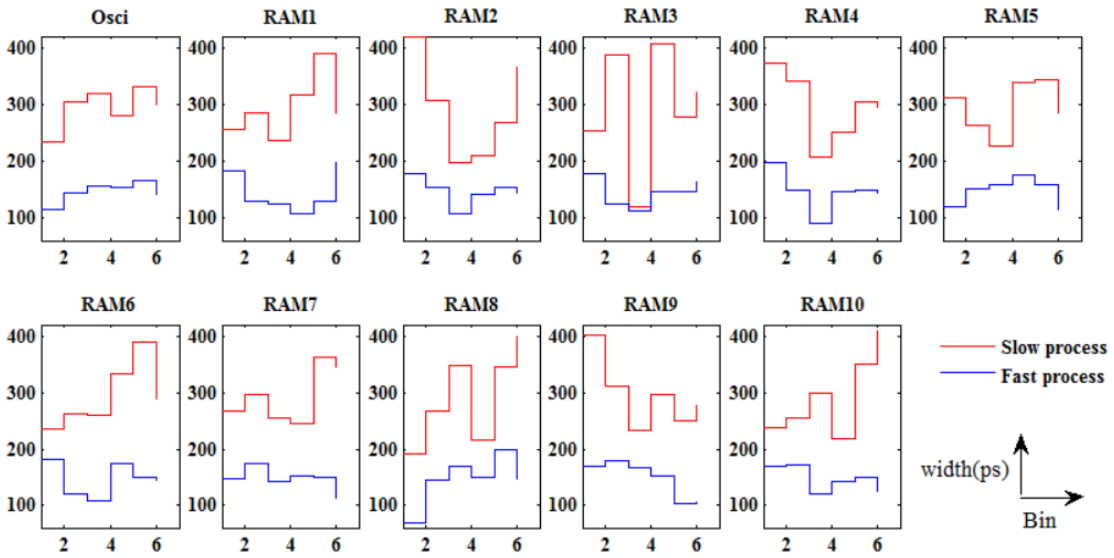



Figure 4.5 Bins widths seen by the oscillator (Osci) and LUTRAMs (RAM*i*) in the slow and fast processes after interconnect remapping and bin_by_bin calibration

In fact, arbitrary propagation delays of interconnects from the oscillator to the LUTRAMs make bin sizes somewhat large and non-linear. Non-linearity was reduced by remapping interconnects

between the DUs outputs and the LUTRAMs address pins. Bin-by-Bin calibration [16] further reduced non-linearity. Tableau 4.3 shows the worst INL and DNL.

Tableau 4.3 Worst DNL and INL corresponding to the oscillator and each LUTRAMs measured in slow and fast processes after interconnect remapping and bin_by_bin calibration

| | Slow process | | Fast process | |
|---|---|---|---|---|
| | DNL (LSB) | INL (LSB) | DNL (LSB) | INL (LSB) |
| Osci | -0.21 | -0.21 | -0.21 | -0.21 |
| LUTRAM1 | +0.32 | -0.35 | +0.37 | -0.38 |
| LUTRAM2 | +0.42 | +0.47 | -0.27 | +0.27 |
| LUTRAM3 | +0.15 | -0.18 | -0.23 | +0.22 |
| LUTRAM4 | -0.29 | +0.42 | -0.37 | +0.38 |
| LUTRAM5 | -0.22 | -0.27 | -0.23 | +0.24 |
| LUTRAM6 | +0.32 | -0.43 | -0.26 | +0.24 |
| LUTRAM7 | +0.22 | -0.39 | -0.24 | +0.24 |
| LUTRAM8 | +0.36 | -0.53 | -0.52 | -0.54 |
| LUTRAM9 | +0.36 | +0.42 | -0.26 | +0.57 |
| LUTRAM10 | +0.4 | -0.59 | -0.18 | +0.34 |

Compared to recently proposed FPGA-based TDCs, our bin widths are larger than those in [33] that got DNL=0.55 LSB, INL=0.72 LSB for LSB=63.3 ps. Notice that large and different interconnect delays causing large bins can be addressed by better calibrating interconnects delay, using delay tuning techniques as proposed in [25] or [23] but these investigations are beyond the scope of the present paper.

## 4.5 Conclusion

This paper proposed a new multi-measurement TDC that supports a large number of signal inputs while using minimal FPGA resources. This compact architecture shares ring oscillator delay line among the input signals under test and uses a round tracker to count the number of iterations. Experimental results for a six-stage oscillator, 32-state round tracker and 20 input signals TDC implemented in a Xilinx FPGA showed that our MMRO-TDC occupies 85 and 2.1 times fewer Flip-Flops and lookup tables respectively compared to a multichannel basic RO-TDC. These resource savings were partly obtained at the cost of degraded delay resolutions. Several investigations are ongoing to improve the temporal performances of the MMRO-TDC.

## 4.6 Acknowledgment

We thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Microsystems Strategic Alliance of Quebec (ReSMIQ) for their financial support.

# CHAPITRE 5 ARTICLE 2 - SUB-PS RESOLUTION PROGRAMMABLE DELAYS IMPLEMENTED IN A XILINX FPGA

**Abstract**

In this paper, a novel way to finely tune a net delay on Xilinx Field Programmable Gate arrays (FPGAs) is proposed. It consists of adding floating interconnects (nodes) to the net on which the delay is to be tuned, connected to any input pin of a switch matrix along the net. Adding nodes is made with a TCL script applied to an already placed and routed design. However, such nodes, also called antennas, typically cause fatal errors during the design flow and normally prevent the tools from generating the bit stream. To overcome this issue, a breadth-first search algorithm connecting each node to a load is proposed in this work. Experimental results conducted on a ZYNQxc7z010-3clg400 Xilinx FPGA using the Vivado Design suite showed that it is possible to add small delay steps to a net with a resolution under a pico-second and a covered range proportional to the number of added nodes reaching 48.6 ps for a net with 15 added nodes.

Keywords: Delay tuning; Programmable delay line; sub-ps temporal resolution, FPGA.

**Authors :** Safa Berrima, Yves Blaquière and Yvon Savaria

**Published in :** IEEE International Midwest Symposium on Circuits and Systems (MWSCAS) 2017 [39]

## 5.1 Introduction

Delay tuning is required in a variety of FPGA applications, such as clock tuning [40], Time to Digital Converters (TDC) [41][28], Physical Unclonable Functions (PUF) [25] and True Random Number Generators (TRNG) [42]. Such applications require the ability to control the delay at the finest granularity between two given points in the circuit. Even though the placement of circuit elements can be fully controlled, the routing controllability is limited. In fact, Xilinx Vivado router can help to restrain net delays between upper and lower limits by using appropriate timing constraints, but this still gives the router a great deal of flexibility.

Although FPGAs have become the platform of choice for many applications due to their flexibility, low development costs and fast development cycles, the scope of previous work on fine delay tuning with a pico-second resolution is limited and based on changing the path propagation length. In [23], the Switch Matrix (SM) associated to a Configurable Logic Block (CLB) was used to adjust the propagation path length between two Lookup tables (LUTs). Indeed, a switch matrix is made of several input pins that direct a signal to one or several output pins. Some input pins allow the signal to be bounced inside the switch matrix and the number of possible routes connecting the two LUTs is large. Experiments on Virtex-II Pro FPGA have showed that with a total of 546 routes, a differential delay with a resolution of 18 ps in a range of 947 ps can be reached. This technique was further enhanced [24] by using more routes (~40 000) to achieve a differential delay with a resolution of 1 ps in a range of 2.5 ns. However, as each route requires a new configuration, dynamic reconfiguration must be used. Furthermore, the construction of the complete database containing the delay associated to each of the 40 000 routes takes a long time. Another delay tuning method was proposed [42][25] using a single LUT implemented as an inverter and where only one input is used, while the others that are “don’t cares” affect the signal propagation path through the tree-like structure of the multiplexers making up the LUT. Experiments performed with Xilinx Virtex-5 FPGA showed that a delay with sub-ps resolution can be achieved without dynamic reconfiguration. However, this technique suffers from a low dynamic range estimated to 10 ps. Notice that in the Ultrascale Xilinx FPGAs there are programmable input/output delay primitives that can delay any non-clock signal using a 512-tap delay line with a resolution of 2.5 to 15 ps [43].

In this paper we propose a novel way to tune/control net delays in a Xilinx FPGA with a sub-ps resolution without changing the net routing and by only increasing capacitive loading on input pins of any switch matrices on the net. Indeed, an input pin of a switch matrix is a set of programmable interconnect points that can be turned on/off to enable/disable connections between one input node and one or several output nodes. The net delay can be finely adjusted by altering the number of activated output nodes per input pin using a TCL script applied to an already placed and routed design.

The next section presents an overview of the routing resources and switch matrices in Xilinx FPGAs. In section 5.3, the proposed way to tune delays is described. Section 5.4 shows experimental results obtained from an implementation using a ZYNQxc7z010 Xilinx FPGA. The main conclusions of this work are summarized in section 5.5.

## 5.2 Overview of the routing resources inside a Xilinx FPGA

An FPGA is a grid-like architecture made of tiles. The different types of tiles mentioned below are for the 7-Series to Ultrascale+ families. Older FPGAs may contain minor differences.

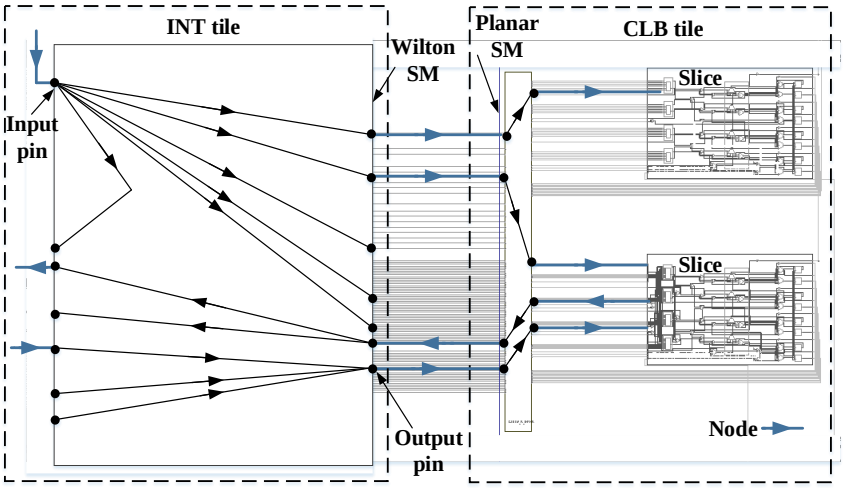



Figure 5.1 Topology of CLB and INT tiles in a 7-Series FPGA (The two slices are merged into one in Ultrascale/Ultrascale+ devices)

- A Configurable Logic Block tile (CLB tile) of 7-Series FPGAs is made of two slices (Figure 5.1), each composed mainly of four six-input LUTs, eight Latches/Flip-Flops and a 4-bit carry chain [38]. In Ultrascale and Ultrascale+ devices, the two slices are combined into one cohesive slice. A CLB implements most of the logic in an FPGA. It also includes a planar switch matrix (SM) [44].

- An Interconnect tile (INT tile) associated to each CLB tile (Figure 5.1) is made of one Wilton switch matrix (SM) [44] that directs a signal from one slice to another inside the same CLB or to the routing fabric, i.e. to other CLBs. Notice that before Virtex-4 FPGAs, CLB and INT tiles were merged into a single tile [45]. The Wilton or planar SM are made of input and output pins as depicted in Figure 5.1. Input pins have data coming into the SM, while output pins have data outgoing from the SM. In a Wilton SM, an input pin can connect a signal to multiple destinations simultaneously and an output pin can accept one signal from one of the multiple pins. In a planar SM, an input pin connects a signal to a unique destination and an output pin accepts a signal from a unique source.

A digital signal can be expressed by a nested list of nodes (Figure 5.1). A node can be an input or an output to a switch matrix and is unidirectional. An input node directs data to several nodes called

*downhill nodes,* while an output node receives data, separately, from several nodes called *uphill nodes*. Each connection between an input and an output node of the SM is controlled by a programmable interconnect point (PIP). If a PIP is turned on then the signal passes from the corresponding input node to the corresponding output node, otherwise the connection is open. In the Xilinx Vivado Tool, the pair of connected nodes per PIP can be extracted with the *get_nodes –of_objects [get_pips]* command line.

- Other tiles exist in an FPGA such as clock manager, block ram, and digital signal processing tiles. The complete list of tiles making up the device can be extracted with the *get_tiles* command in Xilinx Vivado tool.

A digital signal connecting a source component to a target component in different CLBs starts with a planar SM, passes through a series of Wilton SMs and ends up at a planar SM ($Net_A$ in Figure 5.2(a)). It is possible to control routing with Vivado by specifying the name of nodes that the net should pass through. This can be done with the Xilinx Design Constraints file (XDC) or with a TCL script using the *fixed_route* property. For instance, to enforce $Net_A$, connecting the source (Src) to the destination (Dest) in Figure 5.2(a), to follow the precise shown route, one can use the following command: *set_property fixed_route {*$N_1$ $N_2$ $N_3$ $N_4$ $N_5$*} [get_nets* $Net_A$*]*.

## 5.3 Adding floating nodes to finely increase a net delay

The delay of a net made of several nodes can be finely increased by adding floating nodes, which increase capacitive loading on one or several Wilton SM input pins on the net. Figure 5.2 illustrates an example of the proposed idea. The net $Net_A$ shown in Figure 5.2(a) links a source point (Src) to a destination point (Dest) through five nodes {$N_1$,$N_2$,$N_3$,$N_4$,$N_5$}. $Net_A$ passes through two Wilton SMs and two planar SMs. As a Wilton SM input pin can direct a node to several output nodes simultaneously, node $N_2$ is directed to $N_3$, but also to two other extra nodes (Figure 5.2(b)). Therefore, the fan-out of the SM input pin receiving $N_2$ is increased, which increases the delay of $Net_A$.

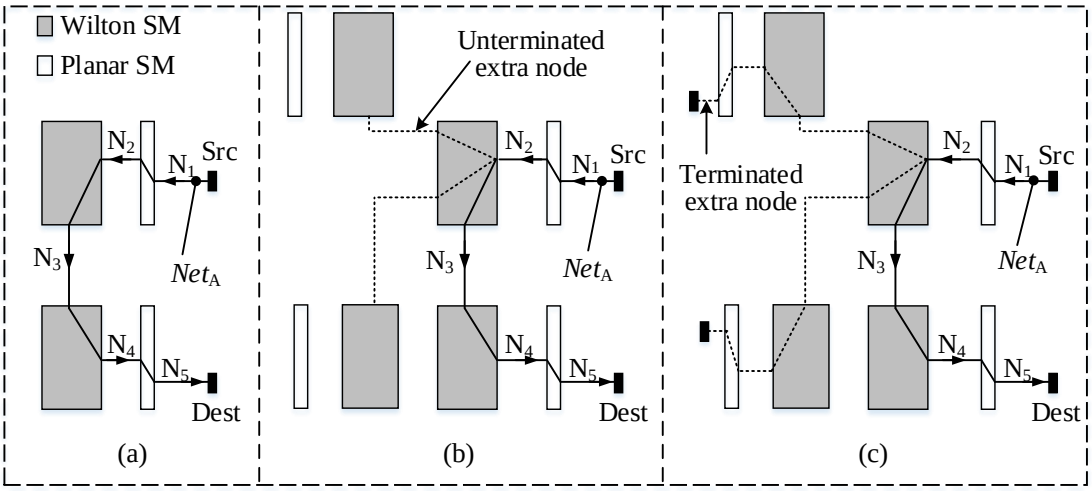



Figure 5.2 (a) $Net_A$ made of five nodes (b) $Net_A$ with two extra floating nodes (c) $Net_A$ with two extra floating nodes terminated to loads

Notice that a higher number of floating nodes can be added using any Wilton SM input pin along the net. These additions are easily made with a TCL script applied to any already placed and routed design, such that no changes other than the added nodes are made to the initial design.

Extra nodes can effectively increase the delay at a fine resolution, but create antennas in the design that prevent the tool from generating the bit stream. These antennas must be terminated to a pin to avoid their removal from Vivado as depicted in Figure 5.2(c). This is performed with a TCL script described in the pseudo-code of Figure 5.3. This script performs a breadth-first search algorithm to find a route from the floating node up to a pin. The benefit of the breadth-first search algorithm is finding the shortest path, which alleviates the routing congestion in the design. A pin could be an input to a CLB's component, a DSP, a Block RAM or any other FPGA component. Let *N* be the floating node on $Net_A$ (as in Figure 5.2(b)), *pin* be the searched pin and *load* be the component having *pin* as input. The Vivado command *get_property cost_code_name* determines whether a given node enters a planar SM and returns the PINFEED value if true:

1) If *N* is an input to a planar SM: it is connected to an input pin through its unique downhill node called $DN_{unique}$ (line 7 in Figure 5.3). The coordinates of the *load*'s site as well as its input *pin* name to which $DN_{unique}$ has to be connected are extracted (line 8). Then, a component (a cell) of the same type and the same location as *load* is created and placed (lines 9 and 10). Finally, *N* and $DN_{unique}$ are added to $Net_A$ (line 12) and the route linking $Net_A$ to *pin* is created (line 13). Notice that the

*find_routing_path* command (line 11) returns the shortest list of nodes between a start node and an end node inclusively.

```
1.  Create empty queue Q and list T
2.  Set Node = N
3.  Q.enqueue (N)
4.  Terminate_Node (N)
5.  Begin
6.  If (get_property cost_code_name (N) = PINFEED)
7.    DN_unique=get_nodes -downhill of N
8.    get_site_pins of DN_unique            //Extract load and pin locations
9.    create_cell reference load
10.   place_cell  at load location
11.   T= find_routing_path from Node to DN_unique
12.   set_property fixed_route {N1 N2{T1 T2...T_length T} N3 N4 N5} of Net_A
13.   connect_net Net_A to pin
14.   return
15. Else
16.   Q.dequeue (N)
17.   DN = get_nodes -downhill (N)
18.        For each node DN_i of DN
19.             Q.enqueue (DN_i)
20.        End for
21.   Terminate_Node (Q.front)
22. End if
23. End
```

Figure 5.3 Pseudo-code to terminate a floating node *N* on $Net_A$ (Figure 5.2) to a load

2) If *N* is an input to a Wilton SM: there is no direct downhill node connected to a pin and further nodes must be traversed before ending up to a pin. In this case a search for a node connected to a planar SM is performed among the downhill nodes $\{DN_1, DN_2,\ldots\}$ of *N* stored in DN (line 17). If no node from DN is connected to a planar SM, the search continues on the downhill nodes of each element of DN in a breadth-first search manner. Once a node entering a planar SM is found, its unique downhill node $DN_{unique}$ is extracted and the same steps described in lines 7-13 are applied to connect it to the corresponding pin. In this case, all nodes connecting *N* up to $DN_{unique}$, stored in T list (line 11) are added to $Net_A$ (line 12). Finally, the route linking $Net_A$ to *pin* is created.

## 5.4 Implementation results

To prove the effectiveness of our proposed sub-ps delay control way, floating nodes were added to a net along a ring oscillator, which increase the oscillation period. The oscillator is made of two

LUTs (LUT1 and LUT2), placed in two CLBs: one configured as an inverter while the other as a buffer (Figure 5.4). The net joining LUT1's output to LUT2's input (zoomed in Figure 5.4) passes through nodes named CLBLM_L_B, CLBLM_LOGIC_OUTS9, SR1BEG2, IMUX38 and CLBLM_M_D3. Results reported in this section were obtained by adding floating nodes to the Wilton SM input pin receiving SR1BEG2 node. Fifteen floating nodes $\{N_1 \ldots N_{15}\}$ were added and connected to their respective pins to avoid the antenna effect (Figure 5.4). Experiments were conducted on a ZYNQxc7z010-3clg400 Xilinx FPGA using the Vivado 2016.1 design suite.

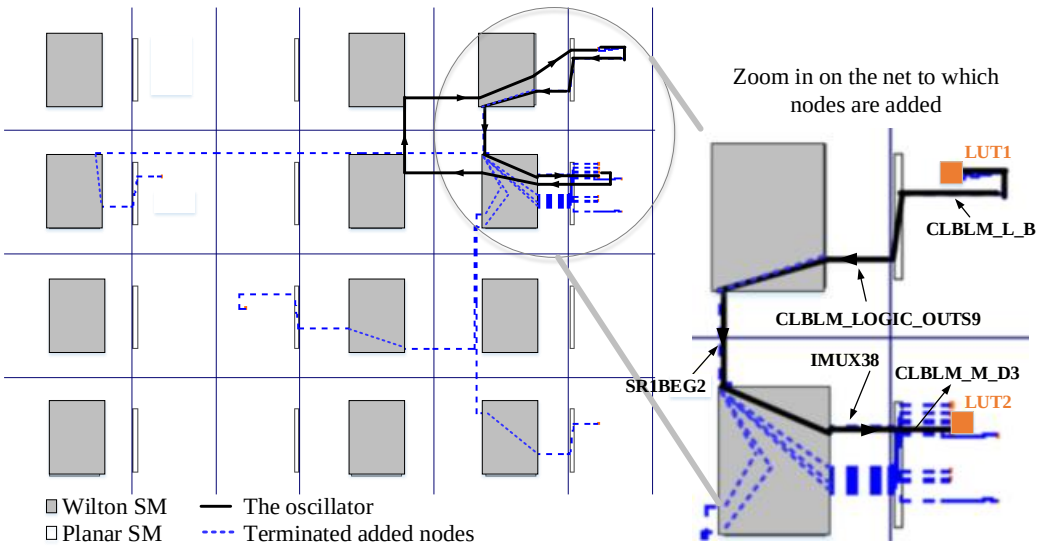



Figure 5.4 Fifteen added nodes outgoing an input pin of a Wilton SM extracted from Vivado tool

The effect of the added nodes on the oscillation period $T_{osci}$ is measured with a 23-bit binary counter driven by the ring oscillator. The value of the counter is read after a reference period set to 9 ms and obtained from a 125 MHz crystal.

Figure 5.5 shows the experimental $T_{osci}$ increases observed when 15 extra nodes are independently connected to the ring oscillator. Each experiment was repeated thirty times to average the fluctuation due to clock jitter, power supply and signal noise. For each node, the average of the 30 measurements, the minimum and maximum values as well as the standard deviations are shown in Figure 5.5. The oscillation period increases vary between 4.35 ps (node $N_1$) and 7.06 ps (node $N_{15}$). The worst standard deviation is equal to 0.37 ps, observed on node $N_{14}$. Notice that the added

delay to the net linking LUT1 to LUT2 (zoomed in Figure 5.4) can be deduced by dividing by two the increase observed on the oscillation period.

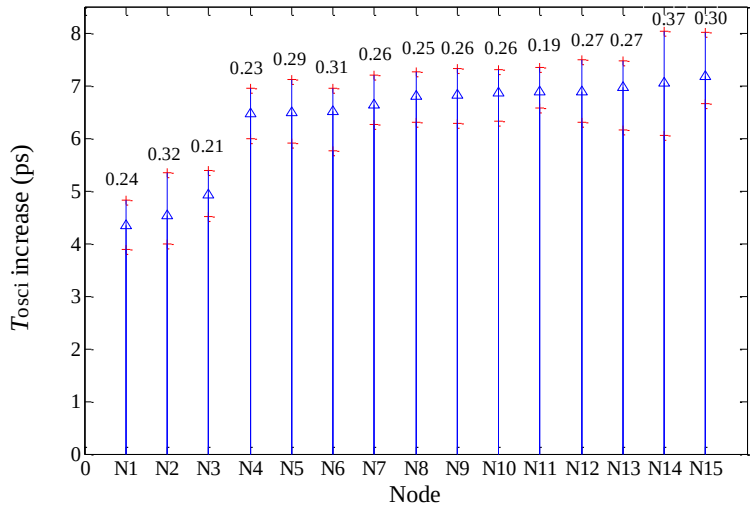



Figure 5.5 Oscillation period increase corresponding to each of the 15 added nodes (Averages shown with triangles, min/max values shown with red markers and the standard deviations in ps depicted above each column)

Knowing the $T_{osci}$ increase introduced by each extra node, it is possible to combine multiple extra nodes connected to the ring oscillator to reach any target oscillation. For this purpose, 58 different sets of nodes were experimented, where each set is a combination of extra nodes connected simultaneously. For each set, thirty $T_{osci}$ increases measurements were extracted and the averages are shown in the plotted red curve of Figure 5.6. Expected values are obtained by summing $T_{osci}$ increases introduced by each node (shown in Figure 5.5) and are depicted in the plotted blue curve of Figure 5.6. The two curves match with a maximum differential delay of 0.72 ps, which is smaller than one picosecond.

Figure 5.6 also depicts the sub-ps obtained resolution. In fact, the $T_{osci}$ increase starts from 4.35 ps and reaches 31.96 ps (i.e. the added delay to the net linking LUT1 to LUT2 is between 2.17 ps and 15.98 ps) with a sub-ps step between two successive points on the red curve. The lower and upper limits of 4.35 ps and 31.96 ps were obtained with one and a set of five added nodes respectively. When the 15 nodes are added simultaneously, the experimental upper limit reaches 97.3 ps (not shown in Figure 5.6). Thus, the added delay to the net linking LUT1 to LUT2 (zoomed in Figure 5.4)

is equal to 48.6 ps. Notice that a higher upper limit could be reached by involving a larger number of extra nodes and a larger number of Wilton SMs. The input pin receiving SR1BEG2 supports 26 extra nodes while some other input pins support up to 34 extra nodes.

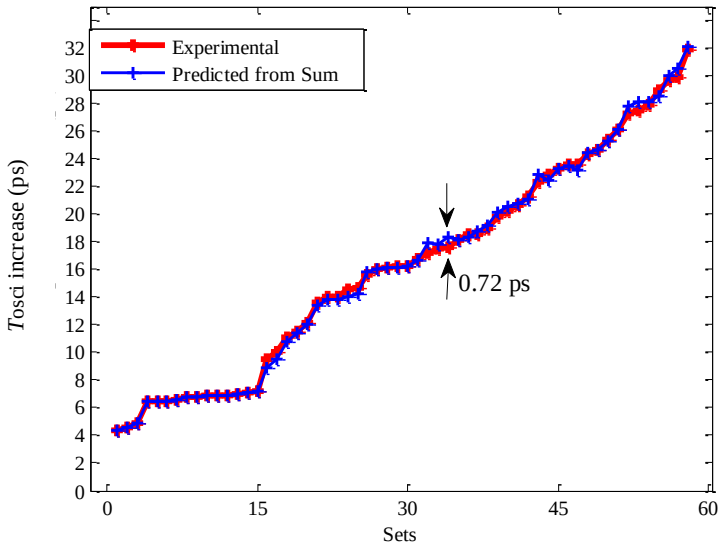
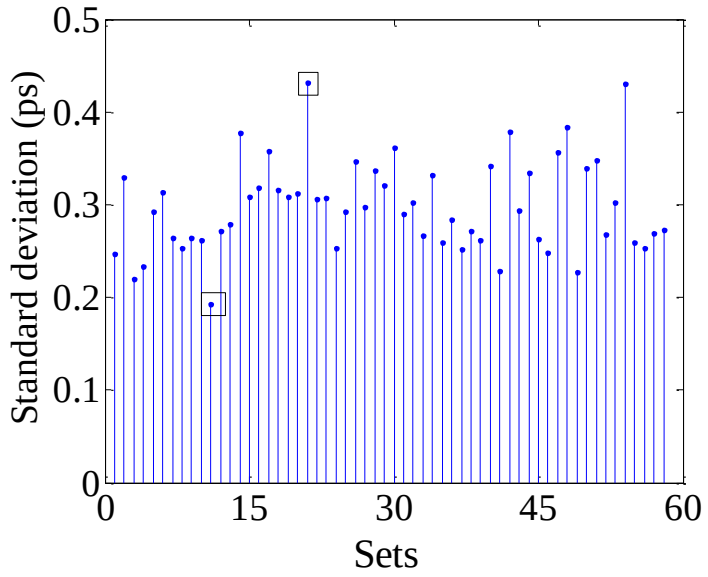



Figure 5.6 Oscillation period increases for 58 different combinations of extra-nodes with a sub-picosecond resolution

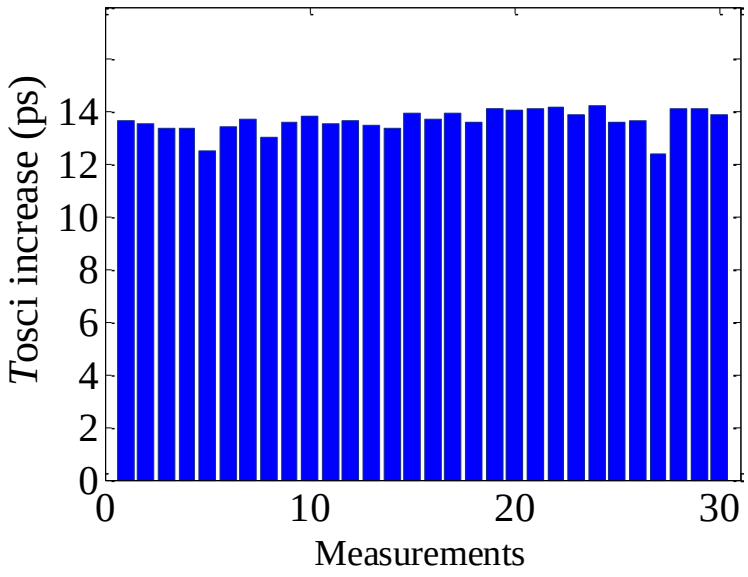
To better show the stability of the experimental results obtained in Figure 5.6, and the dispersion of the 30 measurements around the average, the standard deviations corresponding to each set are reported as an histogram in Figure 5.7(a). They are between a maximum of 0.43 ps (set nb.21) and a minimum of 0.19 ps (set nb.11). The 30 measurements corresponding to the set with the worst standard deviation (set nb.21) are shown in Figure 5.7(b). These experimental values are relatively stable as the maximum and minimum $T_{osci}$ increases are equal respectively to 14.2 ps and 12.3 ps (marked on Figure 5.7(b)). The remaining 57 sets have smaller deviations (Figure 5.7(a)) than those shown in Figure 5.7(b).

Notice that results shown in this section were extracted by using only 58 combinations of extra nodes among $2^{15}$. Fifty eight sets were sufficient to show the sub-ps delay increase. However, involving more combinations would show a better resolution by further decreasing the step between two successive points of the curves in Figure 5.6. This proposed method requires a prior phase to characterize the delay increase introduced by each node before choosing the appropriate

combination to achieve the target delay. Several investigations are ongoing to propose a temporal-optimized characterization phase. Future work will also include the study of a net's added delay introduced by nodes in different parts of the FPGA and in function of the node length.




(a)




(b)

Figure 5.7 (a) Standard deviations of each of the 58 sets (min and max values are marked with squares) (b) Raw *T*osci increases obtained in each of the 30 measurements of the set having the worst standard deviation of 0.43 ps

## 5.5 Conclusion

This paper proposed a sub-ps delay tuning method in Xilinx FPGAs. This tuning is performed by increasing the capacitive loading of any input pin of switch matrices along the net. Loading is increased by adding floating or unused interconnects in the FPGA design. A TCL script is proposed to automatically add these floating interconnects without design alteration using the Xilinx Vivado tool. Experimental results conducted on ZYNQxc7z010-3clg400 Xilinx FPGA using the Vivado 2016.1 design suite showed that combinations of 15 floating interconnects can easily provide an added delay to a net with sub-ps resolution in a range of 2.17 to 48.6 ps.

## 5.6 Acknowledgment

We thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support.

# CHAPITRE 6 ARTICLE 3 - A FINE RESOLUTION DELAY TUNING METHOD TO IMPROVE THE LINEARITY OF AN UNBALANCED TIME-TO-DIGITAL CONVERTER ON A XILINX FPGA

**Abstract**

In this paper, a method for fine adjustment of Xilinx FPGA routing delays is proposed and applied to improve the linearity of an unbalanced multi-measurement time-to-digital converter (TDC). The delay control method increases load capacitances of interconnect points of switch matrices by small amounts using additional connections to unused interconnects in the FPGA fabric. The novel delay control method uses the TCL scripting feature available in the Xilinx Vivado tool to automatically add wires into a fully placed and routed design. A total of 61 additional wires were successfully and automatically added to reduce the differential and integral non-linearities of the target TDC from 0.51 LSB and -0.54 LSB to 0.05 LSB and 0.06 LSB respectively (reduction factors of 10.2 and 9) for an LSB equal to 333 ps.

**Authors:** Safa Berrima, Yves Blaquière and Yvon Savaria
**Published in**: IET Circuits, devices & systems (2020) [46]
Received on 22nd January 2020
Revised 3rd June 2020
Accepted on 6th July 2020
E-First on 12th November 2020

## 6.1 Introduction

Time-to-digital converters (TDCs) are used in various engineering applications for fine time intervals measurements. For instance, TDCs are used in Positron Emission Tomography [1], Fluorescence Lifetime Imaging Microscopy [29] or to build all digital phase-locked loops [47]. TDCs are also used in some measurement instruments such as oscilloscopes and logic analyzers.

Several architectures have been proposed for implementing TDCs in the literature. Application Specific Integrated Circuit (ASIC)-based TDCs achieved few picoseconds resolutions, however they remain expensive and require long development times. A survey of CMOS TDCs is given

in [48]. From the last decade, fully digital TDCs started to be implemented in Field Programmable Gate Arrays (FPGAs) taking advantage of their flexibility, low development costs and fast development cycles.

The simplest TDC architecture is a delay line usually made of inverters (delay units, DUs) whose outputs are connected to flip-flops [49]. In order to measure the time interval between two events (START and STOP), the START signal is propagated along the delay line and the output status of the delay line is recorded by the flip-flops triggered on the arrival of the STOP signal. Knowing the status of the flip-flops and the propagation delay through one delay unit, the time interval between START and STOP can be computed. The TDC's output vector given by the flip-flops is generally expressed as a thermometer code that is further translated to a digital representation with an encoder. Several derived TDC architectures were proposed such as the one exploiting the Vernier concept [50], one based on multiple interpolations [51] and the pulse shrinking method [52] all aiming to improve the temporal performances.

A main concern of a TDC, when implemented in an FPGA, is its linearity defined by the DNL (Differential Non-Linearity) and INL (Integral Non-Linearity) parameters. The DNL is the difference between the actual and the ideal steps in the input-output curve, while the INL is the integral of DNL values along the DUs delay chain up to the current position of calculation. In fact, when implemented in an FPGA, a high nonlinearity is observed due to intrinsic delay differences present in the FPGA fabric. Some bins (the delays between two subsequent DUs) are wider due to boundary crossings between logic array blocks. In addition, changes in local temperature, process and voltage cause bins to be uneven.

Several published works presented solutions to improve FPGA-based TDC's linearity. The most popular technique is bin-by-bin calibration [17], where a Look-Up Table (LUT) semi-continuously stores the updated bins at the ambient temperature. The updated bins are measured using a well-known statistical method called the code density test [53] and calibrated to the center. In [20], a two-step pipelined on-the-fly calibration architecture was proposed to compensate the nonlinearity. It updates bins in a LUT using the code density test with a prefix adder to compute the new bins. The authors in [15] propose to improve the linearity by connecting the input signal to multiple parallel chains simultaneously and using the average of their respective outputs as the final output. However, this method is expensive as it consumes considerable FPGA resources to achieve good

timing linearity. In [54], a better linearity was achieved with a method based on on-line bin calibration derived from the bin-by-bin calibration using addends and bit-shifting operations. However, this method suffers from a high FPGA resource consumption.

Another concern when implementing a TDC is its size. In order to reduce the area occupied by TDCs based on delay lines, ring oscillator-based TDCs (RO-TDCs) were proposed [10][21]. The same delay units are used in an iterative way instead of extending the delay line and a counter tracks the number of cycles. Apart from saving logic resources, the RO-TDC architecture helps to improve the linearity of the TDC. Recently we proposed a multi-measurements RO-TDC (MMRO-TDC) implemented in a Xilinx FPGA [28]. This architecture was supported by a compact implementation suitable for multiple simultaneous measurements. It uses LUTs configured as distributed memories and shift registers. This paper does not associate our MMRO-TDC to a specific target application. It could be used in applications where relative time intervals between multiple hit signals are measured and where area savings are critical.

While the MMRO-TDC reduced significantly the FPGA resource requirements compared to the basic RO-TDC architecture [21], it suffered from poor linearity due to the different propagation delays of routes in the FPGA fabric. Techniques to fine control delays of routes in an FPGA were therefore needed to improve its linearity. To the best of our knowledge, only three previous works presented fine delay control (tuning) of routes in FPGAs. In [23][24], a signal from a LUT is bounced inside a switch matrix (SM) before reaching another LUT. As the number of routes inside the switch matrix is high, the sorted propagation delays of all possible routes vary by fine amounts. Experiments on a Virtex II-Pro FPGA showed that propagation delays between two LUTs could vary by 1 ps in a range of 2.5 ns, depending on the adopted route (for ~40 000 extracted routes). This method required re-configuring the FPGA for each created route, therefore, to use this approach, a form of dynamic FPGA configuration [55] must be available. In [25], the tree-like switch structure making up a LUT was used to generate multiple parallel routes between two points. The LUT is configured as an inverter, where only one input is used. Experiments on a Virtex-5 FPGA showed a sub-picosecond resolution but in a very narrow range of 10 ps. In [39], reported experiments show that sub-ps delay tuning was possible by adding branches on nets in a Xilinx Zynq-7 FPGA.

The main contributions of this paper are as follows:

- A novel method that automatically fine-tunes the propagation delays of Xilinx FPGA routes using extra load capacitances introduced by unused interconnects is proposed;

- A tool that implements this fine-tuning method, which can be applied to increase delays on any route in an FPGA design. It uses only free FPGA resources, not used by the design;

- This tool is successfully applied to improve the linearity of a compact unbalanced MMRO-TDC with unsatisfactory DNL (0.51 LSB) and INL (-0.54 LSB) for an LSB of 333 ps. These DNL and INL were improved by factors of 10.2 and 9 respectively;

- The effectiveness of the proposed method is experimentally validated by comparing the measured MMRO-TDC's linearity before and after applying the delay control method on a Xilinx Zynq-7 FPGA.

The rest of this paper is organized as follows: section 6.2 describes the proposed delay tuning method and its implementation with TCL scripts. In section 6.3, the MMRO-TDC is briefly introduced and the technique used to improve its linearity with the proposed delay tuning method is detailed. Experimental results are shown in section 6.4. Section 6.5 discusses some important aspects of the paper and proposes promising future directions. Section 6.6 concludes the paper.

## 6.2 Proposed Delay Tuning Method

The main principle to fine-tune the delay in this paper consists of adding extra load capacitances from unused FPGA interconnects on a route [39]. Some definitions based on Xilinx's terminology and 7-series FPGA topology are first provided in section 6.2.1. They are then used to describe our proposed fine-tuning approach in section 6.2.2. Notice that, although the terms and topology reported in this section are related to the 7-series FPGAs, the proposed approaches are still applicable on any FPGA families supported by the Xilinx Vivado design suite with minor changes.

### 6.2.1 Review of Xilinx FPGAs resources

Xilinx FPGA devices are made with an array of tiles, including logical components (LUTs, flip-flops, Block RAMs, etc.), switch matrices (SMs), I/O blocks and clock management modules. The following terminology is used in the remaining sections. Terms are specific to FPGA device or post-synthesis user design netlist.

FPGA device terms:

- **Tile:** three types of tile are considered in this paper:
  - DSP tiles: stacked in Digital Signal Processing (DSP) columns, each tile is made of two DSP sites and one dedicated switch matrix [56];
  - CLB tiles: each tile contains two Configurable Logic Block (CLB) sites and one dedicated switch matrix [38];
  - BRAM tiles: stacked in Block RAM columns. Each BRAM tile contains two 18-Kb BRAM sites, one 36 Kb BRAM site, one multiplexer and one dedicated switch matrix [57];
- **Bel:** a basic element is an FPGA device object. Bels are grouped together on the device in sites. Figure 6.1 depicts the Bels in a CLB, a BRAM and a DSP tile;
- **Site**: represents logic resources available in a tile. A CLB tile includes two sites, called Slice, as depicted in Figure 6.1(a). A slice contains four LUT6 Bels (labeled D6LUT, C6LUT, B6LUT and A6LUT), one CARRY4 Bel and eight flip-flop Bels (labeled D5FF, DFF, C5FF, CFF, B5FF, BFF, A5FF and AFF). Apart from these Bels, a slice contains also multiplexer Bels used only for routing purposes (not shown in Figure 6.1). Each site is identified with a suffix X*x*Y*y*, where (*x*,*y*) corresponds to its coordinates. Sites and Bels of the BRAM and DSP tiles are also shown in Figure 6.1(b) and Figure 6.1(c) respectively;
- **Node:** a wire that interconnects sites and/or switch matrices. In the example shown in Figure 6.2, $N_1$, $N_2$, $N_3$, $N_4$ and $N_5$ are nodes;
- **Downhill node**: an input node to a switch matrix SM is connected using PIPs (Programmable Interconnect Points) to one or several output nodes called downhill nodes. In Figure 6.2, node $N_2$ has four downhill nodes;
- **Bel_Input node**: node intended to be connected to a **Bel pin** such as a LUT, DSP, BRAM, flip-flop pin. In other words, it is the last node of a net, such as $N_5$ in Figure 6.2.

Post-synthesis user design netlist terms:

- **Net:** a net connects a start point to an end point, i.e. an output and input pin of logical components (such as LUTs, DSPs, flip-flops, BRAMs, etc.) and is made of several nodes. For example, the net shown in Figure 6.2 is composed of five nodes ($N_1$-$N_5$). A net is connected to Bels through Bel pins and to sites through **Site pins** (Figure 6.2);

• **Cell**: is an instance of a design netlist primitive with no further logic details like flip-flops, LUTs, carry logic, DSP or memory block. Cells are placed in Bels.

### 6.2.2 Adding unused downhill nodes to a net for fine delay tuning

For a given net routed on several nodes, adding unused downhill nodes at the input pins of one or several SM increases the capacitive load at those pins, and therefore the total net propagation delay. The unused downhill nodes cannot simply be added to a net, because they would create floating interconnects, called antennas according to Xilinx terminology. Such antennas must be resolved to let the Vivado tool generate the FPGA configuration file. As there is no way to automatically resolve antennas in a design with Vivado, a script made of TCL commands was developed to resolve this issue. The antennas resolution process consists of routing the floating node to a Bel pin termination.

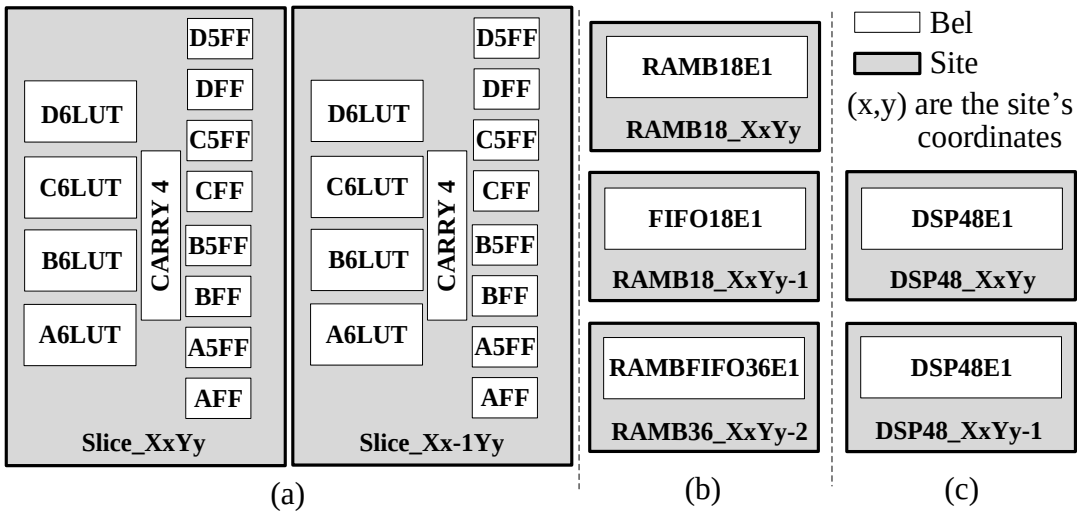



Figure 6.1 Bels and sites in (a) a CLB (b) a BRAM and (c) a DSP tile

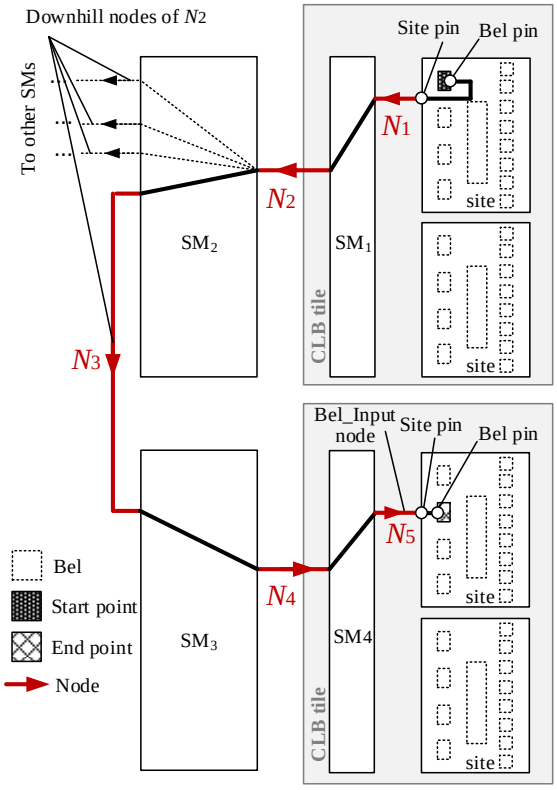



Figure 6.2 Example of a net that starts and ends in different CLB tiles, composed of nodes $N_1$-$N_5$ and passing through switch matrices $SM_1$-$SM_4$

Let us assume that a node $N_i$ is part of a given *net*, whose downhill nodes are used to increase the overall propagation delay of that *net*. The first step in the delay increase process is to search and add a branch to the *net* with a path that connects an added downhill node of $N_i$ to a Bel_Input node. This step finds the branch that uses a minimum set of nodes to minimize routing congestion. The search script for a path leading to a Bel_Input node is described in section 6.2.2.1. The second step tries to resolve the antenna issue by terminating all paths, i.e. connecting each floating Bel_Input node to a Bel pin. It is described in section 6.2.2.2.

### 6.2.2.1 Script to search for a Bel_Input node

The overall propagation delay through a *net* is increased by adding one or more unused Bel_Input nodes, from any node $N_i$ on the *net*. Several Bel_Input nodes can be available with different route lengths. One possible strategy consists of selecting the shortest route to minimize routing congestion, as proposed by the breadth-first search approach in Figure 6.3. It first searches for a

Bel_Input node $N_{INPUT}$ from $N_i$ (lines 1-9 in Figure 6.3) and then adds a branch to the *net* by creating a *path* that connects $N_i$ and $N_{INPUT}$ (lines 10-11).

The script begins by storing the downhill nodes $\{N_1, \ldots, N_{k1}\}$ of $N_i$ in a list $Q$ (line 1). If the first element $FN$ of $Q$ is not an INPUT (line 4) (extracted from its property COST_CODE_NAME [58] (line 3)), then the search continues with all the downhill nodes $\{N_{1\text{-}1},\ldots, N_{1\text{-}k2}\}$ of $FN$ appended to $Q$, which then contains $\{N_1,\ldots, N_{k1}, N_{1\text{-}1},\ldots, N_{1\text{-}k2}\}$ (line 5). The first element of $Q$ is then removed (line 6) and $FN$ is set to the following one. The While loop iterates until the COST_CODE_NAME property of $FN$ returns INPUT, which makes $FN$ the $N_{INPUT}$ of $N_i$. Finally, the shortest path is made between $N_i$ and $N_{INPUT}$ to connect them with the find_routing_path TCL command [58] (line 10). The *path* found is then added to *net* using the FIXED_ROUTE property.

A simplified example is shown in Figure 6.4 where $N_i$ is represented by $N_2$ and *net* by $\{N_1\text{-}N_5\}$. First, the Bel_Input node is searched among the downhill nodes ($N_{2\text{-}1}$, $N_{2\text{-}2}$, $N_{2\text{-}3}$) of $N_2$. As no Bel_Input node is found, the search is then performed on the downhill nodes ($N_{2\text{-}1\text{-}1}$, $N_{2\text{-}1\text{-}2}$, $N_{2\text{-}1\text{-}3}$, $N_{2\text{-}1\text{-}4}$) of $N_{2\text{-}1}$. Again, as none of these nodes is a Bel_Input node, the search continues on the downhill node ($N_{2\text{-}1\text{-}1\text{-}1}$) of $N_{2\text{-}1\text{-}1}$. As $N_{2\text{-}1\text{-}1\text{-}1}$ is a Bel_Input node, it is labeled $N_{INPUT}$. The nodes ($N_{2\text{-}1}$, $N_{2\text{-}1\text{-}1}$, $N_{2\text{-}1\text{-}1\text{-}1}$) are then added to *net*. The *net* finally changes from nodes $\{N_1\text{-}N_5\}$ to $\{N_1, N_2\ \{N_{2\text{-}1}, N_{2\text{-}1\text{-}1}, N_{2\text{-}1\text{-}1\text{-}1}\}, N_3, N_4, N_5\}$.

**Search_Bel_Input node** ($N_i$, *net*)

```
1.  set Q [get_nodes -downhill -of_objects (Ni)]
2.  set FN [lindex Q 0]] # Get first node of Q
3.  set p [get_property COST_CODE_NAME (FN)]
4.  While (p != INPUT)
5.    set Q [concat Q [get_nodes -downhill -of_objects (FN)]]
6.    set Q [lreplace Q 0 0] # Remove the first node of Q
7.    set FN [lindex Q 0]]   # Get first node of Q
8.    set p [get_property COST_CODE_NAME (FN)]
9.  End While
10.   set path [find_routing_path -from Ni –to FN]
11.   set_property FIXED_ROUTE {path} (net)
```

Figure 6.3 TCL pseudo code of the Search_Bel_Input node script

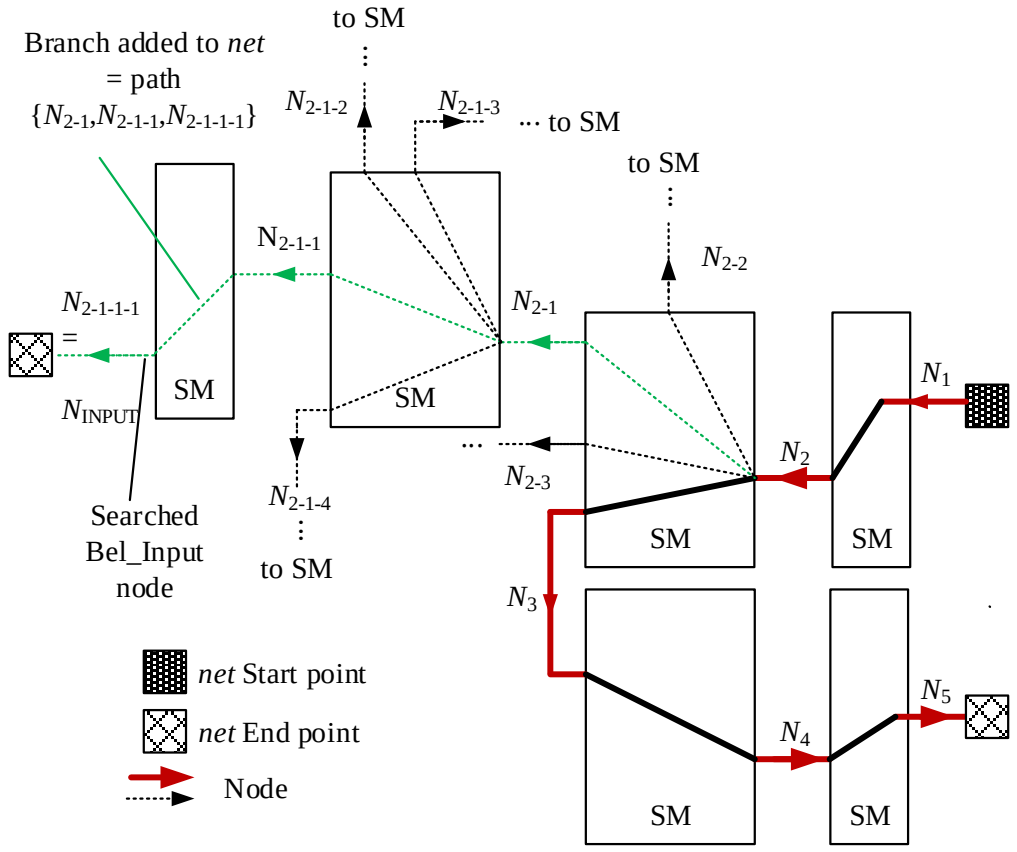



Figure 6.4 An example showing the breadth-first search methodology used to find a Bel_Input node ($N_{INPUT}$) from node $N_2$ (applicable to any Xilinx 7-series FPGA)

### 6.2.2.2 Script to search for a Bel termination

After finding a path connecting $N_i$ to a Bel_Input node $N_{INPUT}$ and adding it to *net*, the route status of *net* becomes partially routed and the Vivado router complains about antennas in the design. Thus, the next step completes the path by connecting the floating Bel_Input node to a Bel termination. In this paper, a Bel termination could be a LUT, flip-flop, CARRY4 (located in a CLB tile), Digital Signal Processor (in a DSP tile) or block RAM (in a BRAM tile).

As mentioned in section 6.2.1, the Bel termination depends on the tile type to which it belongs. The $N_{INPUT}$'s corresponding tile is extracted with the get_tiles –of_objects ($N_{INPUT}$) TCL command [58]. Only the pseudo code of the Bel termination for a CLB tile is given in Figure 6.5. The codes for the DSP and BRAM tile types are similar.

In a CLB tile, the node $N_{INPUT}$ could be connected to a LUT (line 2 in Figure 6.5), flip-flop (lines 6 and 10) or CARRY Bel (line 14), located in the same site (Figure 6.1). As there is no TCL command

that directly returns the Bel from a given node, the site is first extracted with the get_site_pins –of_objects ($N_{INPUT}$) TCL command [58]. The result returns a couple (*site*/*site pin*), where *site* follows this pattern: Slice_X*x*Y*y* (*x* and *y* being its coordinates). The Bel is then deducted from the *site pin'*s name as shown in lines 2, 6, 10 and 14 of Figure 6.5. A cell is then created and placed in the location of that Bel with the commands Create_cell and Place_cell [58] respectively. Finally, the net is connected to the created and placed cell using the connect_net command [58].

**Search_Bel Termination** ($N_{INPUT}$, *net*) (for CLB)

```
1.  get_site_pins –of_objects(N_INPUT) # returns Slice_XxYy/site pin
2.  If (site pin = Di || Ci || Bi || Ai) then # i ∈ [1, 6]
3.    Create_cell –reference LUT6 cell
4.    Place_cell cell in site Slice_XxYy
        and in Bel D6LUT || C6LUT ||B6LUT || A6LUT
5.    Connect_net net to pin cell/Ii  # i ∈ [1, 6]
6.  Elseif (site pin = CE || CLK || SR) then
7.    Create_cell –reference FDRE cell
8.    Place_cell cell in site Slice_XxYy  and
        in any Bel D5FF, DFF, C5FF, CFF, B5FF, BFF, A5FF or AFF
9.    Connect_net net to pin cell /CE || cell /CLK|| cell /SR
10. Elseif (site pin = Dx || Cx|| Bx || Ax) then
11.   Create_cell –reference FDRE cell
12.   Place_cell cell in site Slice_XxYy
        and in Bel DFF || CFF || BFF || AFF
13.   Connect_net net to pin cell /D
14. Elseif (site pin = CIN) then
15.   Create_cell –reference CARRY4 cell
16.   Place_cell cell in site Slice_XxYy and in Bel CARRY4
17.   Connect_net net to pin cell /CIN
18. End if
```

Figure 6.5 TCL pseudo code of the Search_Bel Termination script for CLB sites

Notice that, although the script uses five types of terminations (LUTs, FFs, CARRY4s, DSPs and BRAMs), it can be constrained to use only some of them depending on the available FPGA resources.

### 6.2.3 Delay tuning script

The **Search_Bel_Input node** and **Search_Bel Termination** scripts are used in the global delay tuning script (**Delay_tuning**) described in Figure 6.6. The aim of this global script is to finely increase the propagation delay through *net* until reaching the desired delay denoted by *Delay*. First, the script stores the downhill nodes of the first node $N_i$ of *net* in *nlist* (line 1 in Figure 6.6). The **Search_Bel_Input node** script is then executed on the first element $N_{i1}$ of *nlist* to find a *path* that connects it to a searched Bel_Input node $N_{\text{INPUT}}$ (lines 5-6). A branch is added to *net* (line 7) with the found *path* and the **Search_Bel Termination** script connects $N_{\text{INPUT}}$ to a created and placed cell (line 8). Thus a complete properly terminated branch is added to *net* from node $N_{i1}$.

When a *path* is constructed from a given node of *net* to a termination Bel, the **Delay_tuning** script checks the new status of *net* with two possible scenarios:

- If all the elements used (nodes and logic) are free, *net* is then fully routed. The script continues adding further nodes;
- If one or several nodes or Bels were previously used by other routes of the design, *net* status is then identified as partially routed. In this case, the last performed steps of cell's creation, placement and connection to *net* are deleted by the undo TCL command [58] (lines 9-11). Notice that the undo operation is necessary as no TCL command exists to check in advance the occupation of a node.

After adding a branch to *net*, the script checks whether the desired delay *Delay* is reached (lines 12-16): if the *net* delay is larger or equal to *Delay* then the script stops, otherwise it continues to add further branches from another downhill node of $N_i$. If all the downhill nodes of $N_i$ (stored in *nlist*) were used and *Delay* is not reached yet, the script is then executed on the next node $N_{i+1}$ of *net.* Notice that the net delay in an FPGA could be extracted from a static timing analyzer tool or for better accuracy, by experimentally comparing the frequencies of a ring oscillator that includes or not the added branch, as presented in [59]. As shown in section 6.4.2, our scripts are applied on nets with initial propagation delays of 219 ps, 315 ps, and 203 ps to reach a desired *Delay* of 333 ps.

```
Delay_tuning (Ni, net, Delay)
1.  Set nlist [get_nodes -of_objects (Ni)]
2.  Set j 1
3.  While (j < [llength nlist])
4.    Set Nij [lindex nlist j]
5.    Set path Search_Bel_Input node (Nij)
6.    Set NINPUT [lindex path end]
7.    Set_property FIXED_ROUTE N1 N2 … Ni {path} Ni+1 … Nn of net
8.    Search_Bel Termination (NINPUT)
9.    While [get_property ROUTE_STATUS (net)! = Routed]
10.     Undo
11.   End While
12.   If (propagation delay of net < Delay) then
13.   Incr j                    #stay in the while loop (add another branch)
14.   Else
15.     Set j [llength nlist]   #exit the while loop (stop adding branches)
16.   End if
17. End While
```

Figure 6.6 TCL pseudo code of the Delay_tuning script

## 6.3 Proposed Delay Tuning Method Applied on an unbalanced Multi-Measurements Time-to-digital converter

The delay tuning method presented in the previous section is used in this paper to improve the unsatisfactory linearity of the multi-measurements RO-TDC (MMRO-TDC) presented in [28]. This compact TDC is suitable for delay measurement on multiple inputs. Section 6.3.1 gives a brief overview of the MMRO-TDC architecture. Section 6.3.2 describes in details the application of the delay tuning method on the MMRO-TDC and experimental extraction of the MMRO-TDC linearity before and after delay tuning.

### 6.3.1 Review of the MMRO-TDC

The architecture of the MMRO-TDC proposed in [28] measures relative time intervals between multiple signals under test. In the simplified version shown in Figure 6.7, the MMRO-TDC measures a time interval between only two signals under test ($SUT_i$ and $SUT_{i+1}$). It is mainly made of three blocks. The gated ring *oscillator* block is activated upon receiving the start signal under test ($SUT_i$). It is made of chained delay units (DUs). The first DU is used to activate the oscillation

whenever a measurement is needed and is based on NAND gates while the remaining DUs are buffers. Such a gated ring oscillator allows power consumption savings compared to a free running oscillator.

In the architecture presented in [28], the DUs are implemented using LUTs. Notice that the dedicated carry lines present in the FPGA fabric could be used for better measurement resolutions. The second block is the *sampler* made of one LUTRAM. In fact, some LUTs of the Xilinx FPGAs present in M-type slices [38] can be configured as synchronous RAM resources, called distributed RAMs or LUTRAMs. One LUT can be configured as a single-port 64×1-bit RAM and the four LUTs of the same slice M can be cascaded to make a larger RAM (a single-port 256×1-bit RAM is made of four LUTs).

The output of each DU is connected to an address pin of the LUTRAM. The data input of the LUTRAM is always asserted high. The idea behind this structure is to write a logic '1' at the address of the LUTRAM defined by the outputs of the DUs when the stop signal ($SUT_{i+1}$) arrives. For instance, if the DU outputs are defined by the sequence "111000" at the arrival of $SUT_{i+1}$, then a '1' logic is written in the LUTRAM at the address 0x38.

The third block is the *round tracker* used to track the number of oscillations done before the arrival of the stop signal ($SUT_{i+1}$). It can be made of a simple binary counter or a shift register. The *round tracker* is used to measure the time interval between $SUT_i$ and $SUT_{i+1}$ with a coarse resolution equal to the oscillation period, while the *oscillator* is used for fine measurements (sub-oscillation period resolution). Notice that *k* LUTRAMs and *k round trackers* are needed to measure relative time intervals between *k* signals under test [28].

The chained LUTs making up the oscillator DUs were placed such that the output of each LUT is connected to the input of its adjacent LUT. The output of the last LUT is connected back to the input of the first LUT. Even if a rigorous routing is performed between each adjacent stages of the ring oscillator to make the bins as uniform as possible for a better linearity, bins observed at the LUTRAM side (at the address inputs) can show significant differences (Figure 6.8). In fact, bins observed at the LUTRAM side depend entirely on the propagation delays from the DU outputs to the LUTRAM address inputs, which depend on each routed interconnect. Let $Bin_i$ be the bin between DU*i* and DU*i*+1, $Bin'_i$ be the bin between the LUTRAM addresses A*i* and A*i*+1, and $d_{DUi\text{-}Ai}$ be the propagation delay from DU*i* to LUTRAM address A*i*.

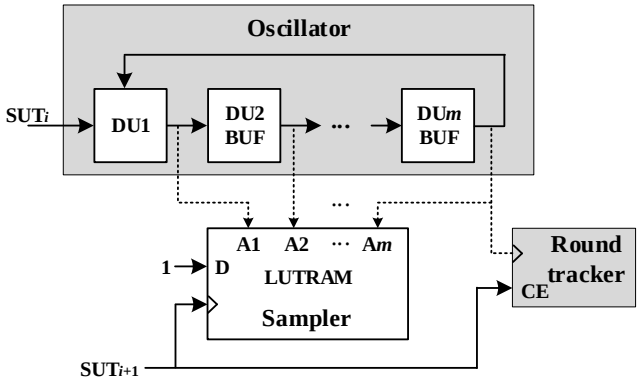



Figure 6.7 A simplified version of the proposed MMRO-TDC architecture where only two hit signals ($SUT_i$ and $SUT_{i+1}$) are considered

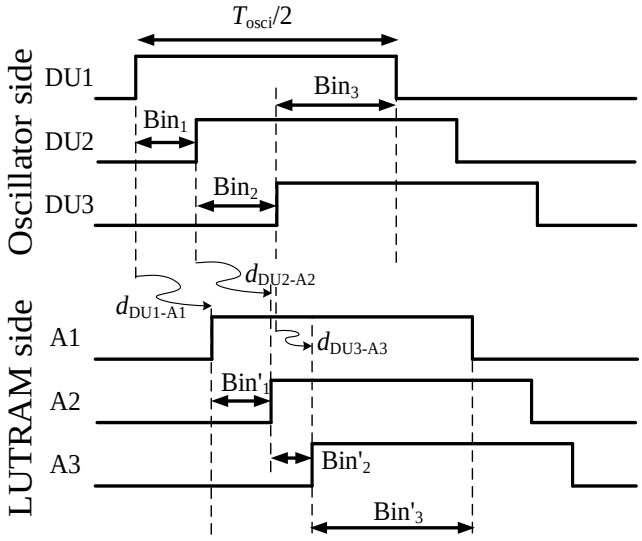



Figure 6.8 Example of bins at the oscillator and LUTRAM sides for a 3-stage oscillator

The relationship between $Bin_i$, $Bin'_i$ and $d_{DUi\text{-}Ai}$ is defined by equations (6.1) and (6.2), where $m$ is the number of oscillator stages and $T_{osci}$ is the oscillation period:

$$\begin{aligned} Bin'_1 &= Bin_1 + d_{DU2\text{-}A2} - d_{DU1\text{-}A1} \\ Bin'_2 &= Bin_2 + d_{DU3\text{-}A3} - d_{DU2\text{-}A2} \\ &\cdots \\ Bin'_m &= Bin_m + d_{DU1\text{-}A1} - d_{DUm\text{-}Am} \end{aligned} \tag{6.1}$$

$$\sum_{i=1}^{m} Bin'_i = \sum_{i=1}^{m} Bin_i = T_{osci}/2 \tag{6.2}$$

The delay tuning method presented in section 6.2 is therefore used to get more similar bins at the LUTRAM side and to compensate for the mismatch between bins in the oscillator as well as for the arbitrary routes delays between DU*i* and the LUTRAM. Its purpose is to make bins at the LUTRAM side even and equal to an average $Bin'_{mean}$ defined by $(Bin'_1+\ldots+ Bin'_m)/m$, where $m$ is the number of oscillator stages.

Notice that a bin can be translated to a statistical entity called bin width using the code density test where the TDC is fed by random time intervals and its output codes (thermometer codes) are stored in a histogram. This histogram represents the number of times each output code has occurred. By dividing the number of occurrences of a corresponding output code by the total number of random time intervals, the bin width can be deducted.

Bins balancing starts by building a histogram of the initial (raw) bins measured by an experimental method described in section 6.3.2. Intuitively, bins found smaller than $Bin'_{mean}$ could be simultaneously enlarged by amounts equal to their differences with $Bin'_{mean}$. However, according to equations (6.1), when a LUTRAM bin $Bin'_i$ is enlarged by $d_{DUi+1\text{-}Ai+1}$, its following bin ($Bin'_{i+1}$) in the histogram is narrowed by $d_{DUi+1\text{-}Ai+1}$. In addition, $Bin'_{i+1}$ is affected by the oscillator $Bin_{i+1}$ that is slightly enlarged when a capacitive load is added on net DU*i*+1-A*i*+1 (as depicted in Figure 6.9(a)). For these reasons, when one LUTRAM bin is enlarged, a new histogram is generated with the updated LUTRAM bins before enlarging a second bin.

### 6.3.2 High-resolution Delay tuning method applied to MMRO-TDC

The proposed delay tuning method is applied to the MMRO-TDC to improve its linearity and get even bins. In fact, the delay control is applied on nets DU*i*-A*i* to compensate for the delay mismatches between the DUs of the oscillator as well as between the routes connecting DUs to the LUTRAM. As shown in Figure 6.9(a) and according to equations (6.1), additional nodes are added to nets DU*i*-A*i* to get even bins $Bin'_{i-1}$ at the LUTRAM level. In the following sections, the bins on the oscillator side and on the LUTRAM side (before and after balancing) are characterized. Even though the oscillator bins characterization is useless for the LUTRAM bins balancing process, it quantifies how much the FPGA bins were unbalanced before applying our proposed delay tuning method.

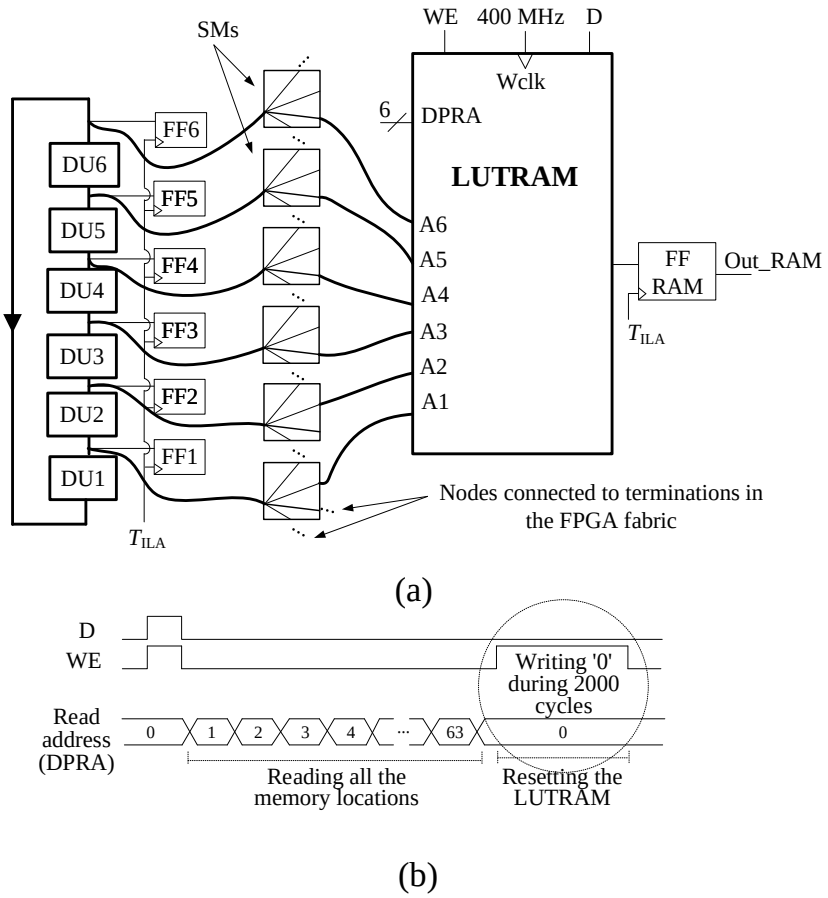



Figure 6.9 LUTRAM bins characterization setup design (a) and waveforms (b) for an MMRO-TDC with a six-stage ring oscillator

In fact, even if the DUs are placed in adjacent LUTs located in the same CLB, the propagation delays of routes between two DUs are most likely different. Furthermore, as the last DU is connected back to the first DU, the last bin is likely to be larger than the other bins.

### 6.3.2.1 Method for oscillator bins characterization

The purpose of the oscillator bins characterization is to experimentally determine the propagation time from one stage to another in the delay line making up the oscillator. This characterization is performed with the code density test [53] in which the ring oscillator stages outputs defined by DUs are sampled at a constant time interval. In an ideal ring oscillator where all the bins are even, all the thermometer codes would have the same probability to appear. However, in the real case and due to the non-uniformities in the routes between two consecutive stages as well the

propagation time through LUTs, different bin sizes are present in the delay line making up the oscillator. The Vivado integrated logic analyzer (ILA) is used for the oscillator bins characterization. The ILA's clock period, denoted $T_{\text{ILA}}$, is generated from the clock system using a clock manager module (MMCM) integrated into the FPGA fabric. It samples the DUs outputs while the ring oscillator is in a free running mode. The DUs outputs are captured by the flip-flops FF1, FF2… FF*n* as shown in Figure 6.9.

A $\text{Bin}_i$ is given by equation (6.3):

$$\text{Bin}_i = \frac{N_i}{N_{\text{total}}} \times \frac{T_{\text{osci}}}{2} \tag{6.3}$$

where $T_{\text{osci}}$ is the oscillator's period, $N_i$ is the number of occurrences of the code associated to $\text{bin}_i$ and $N_{\text{total}}$ is the total number of samples. The thermometer codes for each bin and for each oscillator's falling or rising transition, sampled by the flip-flops attached to DUs, are given in Tableau 6.1 for an *m*-stage oscillator. For instance, the thermometer codes associated to $\text{bin}_1$ are equal to 1000…0 and 0111...11 (for the falling and rising edges respectively).

Tableau 6.1 Thermometer codes captured by the flip-flops (associated to DUs) corresponding to each bin of an *m*-stage ring oscillator

| $FF_1$ | $FF_2$ | … | $FF_{m-1}$ | $FF_m$ | | |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | $\text{Code}_{f1}$ | $\text{Bin}_1$ |
| 0 | 1 | 1 | 1 | 1 | $\text{Code}_{r1}$ | |
| 1 | 1 | 0 | 0 | 0 | $\text{Code}_{f2}$ | $\text{Bin}_2$ |
| 0 | 0 | 1 | 1 | 1 | $\text{Code}_{r2}$ | |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 1 | 1 | 1 | 1 | 0 | $\text{Code}_{rm-1}$ | $\text{Bin}_{m-1}$ |
| 0 | 0 | 0 | 0 | 1 | $\text{Code}_{fm-1}$ | |
| 1 | 1 | 1 | 1 | 1 | $\text{Code}_{rm}$ | $\text{Bin}_m$ |
| 0 | 0 | 0 | 0 | 0 | $\text{Code}_{fm}$ | |

### 6.3.2.2 Method for LUTRAM bins characterization

According to equations (6.1), in order to balance the bins $\text{Bin}'_{i-1}$ at the LUTRAM side, the sets $(\text{Bin}_{i-1} + d_{\text{DU}i\text{-A}i} - d_{\text{DU}i\text{-1-A}i\text{-1}})$ should be balanced for $i \in [1, m]$, where *m* is the number of oscillator

stages. As the oscillator bins are fixed and delays cannot be subtracted, balancing is done by increasing the propagation delays $d_{DUi\text{-}Ai}$. The purpose of the characterization process is to determine the amount of delay to be added to nets DU*i*-A*i* in order to have balanced bins at the LUTRAM side. The added delay should compensate not only for the mismatches occurring between the bins at the ring oscillator side but also for the mismatches occurring between the net delays connecting DU*i* outputs to the LUTRAM. Note that the circuit of Figure 6.9 is not modified to perform bins characterization and during the balancing process as only branches are added to nets DU*i*-A*i*. These branches have an almost negligible effect on the oscillation period, as mentioned in section 6.4.2.

The characterization is experimentally performed without design modification and reroute, where the LUTRAM is configured as a dual-port LUTRAM rather than a simple-port LUTRAM in normal operation. One port is for synchronous writes (writing address being defined by the DUs of the oscillator) and one port is for asynchronous reads defined by a counter. In the example shown in Figure 6.9(a), a 64×1-bit LUTRAM is used and the reading port address is defined by an external counter named DPRA. Each time a new address is read at port DPRA, the memory value located at that address is available at the output port after the time delay to access the LUTRAM. A synchronous read is implemented by connecting the LUTRAM's output to a data port of a flip-flop (FF RAM in Figure 6.9(a)) placed in the same slice.

The characterization flow of the bins at the LUTRAM side follows three phases shown in Figure 6.9(b):

1- during one clock cycle, the LUTRAM's write enable (WE) and data (D) ports are set to one, thus enabling writing a logic one at the address defined by the DUs outputs;
2- the write enable (WE) is deasserted for 64 cycles. During these cycles, the read address (DPRA) is incremented to read the 64 memory locations of the LUTRAM. The output of the LUTRAM is expected to be asserted to '1' once during these 64 cycles (at only one address in the LUTRAM);
3- the LUTRAM must be reset. As the LUTRAM cannot be reset directly, it is forced to all zeros content by activating the write enable (WE) and the data (D) to zero for a number of cycles. Fifty tests with 500 cycles and another fifty tests with 2000 cycles were conducted and in all cases the LUTRAM was completely reset. In our calibration experiments, 2000 cycles were

used to secure a complete reset, which takes 5 µs with a 400 MHz clock. No investigation was performed to find or characterize the optimal minimum number of cycles needed to ensure that the LUTRAMs are reset with high certainty.

### 6.3.2.3 Delay balancing of the MMRO-TDC

The linearity of the MMRO-TDC (Figure 6.9(a)) is improved by applying the **Delay_tuning** script (Figure 6.6) on nets DU*i*-A*i*, which allows better balanced bins at the LUTRAM side. For a given net DU*i*-A*i*, branches are sequentially added to its nodes until $\text{Bin}'_{i-1}$ (equations (6.1)) reaches the target delay. In this section, a statistical approach to automatically verify whether the target delay is reached or not is described and applied.

First, a test is done to extract the raw bins at the LUTRAM side and a histogram with the number of occurrences of each code is constructed. The codes associated with the LUTRAM side bins ($\text{Bin}'_i$) are determined from the reading address DPRA and the output of the LUTRAM (Figure 6.9(a)). For example, when the $\text{Bin}'_1$ occurs, a '1' is written at the LUTRAM addresses defined by $\text{code}_{f1}$ "100000" (32 in decimal, falling edge data) or $\text{code}_{r1}$ "011111" (31 in decimal, rising edge data), assuming a six-stage oscillator. Then during the reading phase (Figure 6.9(b)), the LUTRAM's output (Out_RAM in Figure 6.9(a)) should be asserted only when DPRA is equal to 31 or 32. As the oscillation period seen at the LUTRAM side is the same as the oscillation period of the ring oscillator (see Figure 6.8), a LUTRAM bin width (in picoseconds) is given by equation (6.4):

$$\text{Bin}'_i = \frac{Nb_{\text{occ}}}{Nb_{\text{samples}}} \times \frac{T_{\text{osci}}}{2} \tag{6.4}$$

where $T_{\text{osci}}$ is the period of the ring oscillator, $Nb_{\text{samples}}$ is the number of samples taken to construct the histogram and $Nb_{\text{occ}}$ is the number of occurrences of the two codes (see Tableau 6.1) associated to the bin.

In order to get balanced bins at the LUTRAM side, the number of occurrences $Nb_{\text{occ}}$ of all $\text{Bin}'_i$ should be equal to the mean number of occurrences extracted from the histogram, denoted by $Nb_{\text{mean}}$. Notice that increasing a bin would decrease the other bins as shown by equations (6.1). In fact, bins balancing is a combinatorial problem whose optimal solution is out of the scope of this paper. However, in this work, bin durations are increased in a heuristic manner to balance the

MMRO-TDC bins. Therefore, the purpose of the delay tuning method applied to our TDC is to increase $Nb_{occ}$ of each bin until reaching $Nb_{mean}$.

When sequentially adding extra nodes to net DU*i*-A*i*, the $Nb_{occ}$ associated to Bin'$_{i-1}$ finely increases, according to equations (6.1). Therefore, each time a branch to a Bel_Input node and its associated Bel termination is added to a net, the **Test** script (Figure 6.10) experimentally verifies whether $Nb_{occ}$ reached $Nb_{mean}$. It first generates the bitstream file of the new design and opens the hardware manager tool, which is a feature of the Vivado design suite allowing interaction with the FPGA device (lines 1 and 2 in Figure 6.10). Then it opens a connection to a hardware server and to a hardware target on it (lines 3 and 4). The generated bitstream and probing files are set to be used by the **Test** script (lines 5 and 6). The FPGA is then programmed (line 7), the triggers armed and the debug cores run (lines 8 and 9). The ILA data (thermometer codes of the DUs outputs) are exported in a .csv file for external analysis. The external analysis consists of counting the number of occurrences $Nb_{occ}$ of the two codes (denoted by Code$_{fi}$ and Code$_{ri}$ in lines 13 and 14) associated to a given bin (see Tableau 6.1): if $Nb_{occ}$ is smaller than $Nb_{mean}$, then the script returns the CONTINUE value. In this case, further external nodes must be added to the net to achieve $Nb_{mean}$. If $Nb_{occ}$ is equal or exceeds $Nb_{mean}$ then the **Test** script returns the STOP value to end the adding nodes process.

The **Delay_tuning** ($N_i$, DU*i*-A*i*, $Nb_{mean}$) script of Figure 6.6 is applied on net DU*i*-A*i* in order to finely adjust the LUTRAM bin'$_{i-1}$ (equations (6.1)). The script aims to equalize the occurrence rate of Code$_{fi}$ and Code$_{ri}$ (the two codes associated to bin'$_{i-1}$ (Tableau 6.1 for $m$=6)) equal to $Nb_{mean}$. It uses the **Test** script to check whether the bin occurrence reached $Nb_{mean}$.

**Test** ($Nb_{mean}$)

```
1.  Write_bitsream file.bit
2.  open_hw
3.  connect_hw_server
4.  open_hw_target
5.  Set_property PROGRAM.FILE file.bit
6.  Set_property PROBES.FILE  file.ltx
7.  Program_hw_device
8.  Run_hw_ila
9.  Wait_on_hw_ila
10. Write_hw_ila_data –csv_file file.csv # Code_fi and Code_ri
11. Set fp [open " file.csv " r]
12. Set file_data [read fp]
13. Set Nb_occ_fi [llength [regexp -inline -all -nocase Code_fi file_data]]
14. Set Nb_occ_ri [llength [regexp -inline -all -nocase Code_ri file_data]]
15. Set Nb_occ [Nb_occ_fi + Nb_occ_ri]
16. Close_hw
17. If (Nb_occ < Nb_mean )
18.    Return CONTINUE
19. Else
20.    Return STOP
21. End If
```

Figure 6.10 TCL pseudo code of the Test script

## 6.4 Implementation and Results

A ring oscillator made of six LUTs was implemented in a Zynq7Z010 Xilinx FPGA using the Vivado 2015.4 tool. As demonstrated in [28], a six-stage oscillator makes the MMRO-TDC compact compared to the state-of-the-art ring oscillator-based TDCs. The first DU is configured as a NAND gate while the five others as buffers. The six DUs were placed in two adjacent slices as shown in Figure 6.11. The resulting oscillation period ($T_{osci}$) was measured with the B&K Precision 1823A Frequency counter and found equal to 4114 ps (243 MHz). The clock ($T_{ILA}$ in Figure 6.9(a)) is equal to 400 MHz and is generated from the quartz on the Zybo prototyping board using the Mixed-Mode Clock Manager in the FPGA fabric (pk-to-pk jitter equal to 95.93 ps). The integrated logic analyzer of Vivado was used to extract the results.

### 6.4.1 Oscillator bins characterization

The outputs of the six flip-flops (FF1,…, FF6) in Figure 6.9 (and Figure 6.11) are connected to ILA probes to sample the DUs contents. A total of 32 768 samples were taken. In fact, the maximum sample data depth supported by our FPGA (xc7z010clg400), without exhausting the available internal memory resources, is 32 768. Figure 6.12 shows the raw bins seen at the oscillator side and calculated using equation (6.3). As expected, these bins are between the worst and best cases estimated by the Vivado static timing analyzer.

According to Figure 6.12, the oscillator bins are uneven. In fact, bins 3 and 6 are wider because DU3 and DU4 are located in two different slices as well as DU6 and DU1 (Figure 6.11). This has a negative impact on the oscillator's DNL and INL.

### 6.4.2 Delay balancing of the MMRO-TDC

In order to measure the raw linearity of the MMRO-TDC, the TDC was excited with a large number of random time intervals. In our implementation, the ring oscillator is kept in a free running mode and the writing clock of the LUTRAM is fixed to 400 MHz (Figure 6.9(a)). Recall that the write address at which a '1' is written is defined by a thermometer code in the ring oscillator. As the 400 MHz clock and the ring oscillator frequency are independent, the location where that '1' is written should be uniformly distributed if all bins widths were the same. As bin widths are not exactly the same, different number of occurrences will be observed at the LUTRAM memory locations. Thus the final content of the LUTRAM characterizes the bins uneven duration. A total of 32 768 write cycles were generated and the number of code occurrences corresponding to each $Bin'_i$ was stored in the histogram shown in Figure 6.13(a). Each measurement was repeated 100 times and the average of the 100 measurements was computed.

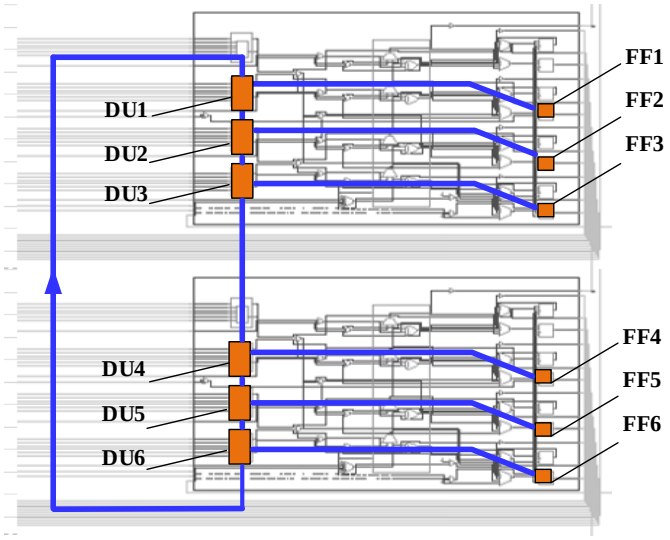



Figure 6.11 The implemented six-stage ring oscillator (for the sake of simplicity, interconnects passing through switch matrices and connecting two subsequent DUs are not shown)

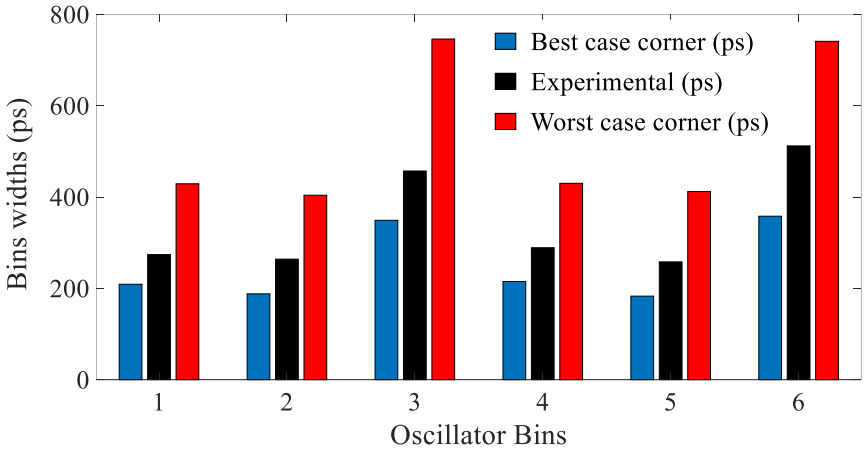



Figure 6.12 Oscillator bins in picoseconds (experimental vs estimated in both worst and best cases by Vivado static timing analyzer tool)

According to Figure 6.13(a), the mean number of occurrences $Nb_{mean}$ is equal to 5310. The occurrences of codes associated to Bin'$_1$, Bin'$_3$ and Bin'$_5$ are below $Nb_{mean}$, therefore these bins have to be enlarged to reduce the standard deviation (Std) initially equal to 1840. Thus, and according to equations (6.1), the delay balancing scripts must be applied to nets DU2-A2, DU4-A4 and DU6-A6 respectively to make the $Nb_{occ}$ associated to Bin'$_1$, Bin'$_3$ and Bin'$_5$ closer to $Nb_{mean}$.

The delay tuning script was first applied on net DU6-A6 in order to enlarge Bin'$_5$. First, the downhill nodes of its first node ($N_{1\text{-}DU6}$) were used for this purpose. Although $N_{1\text{-}DU6}$ has 32 downhill nodes,

only 16 were free, not causing any routing conflicts and usable. Therefore, only 16 extra branches were created from $N_{1\text{-DU6}}$. The $Nb_{occ}$ of Bin'$_5$ increases from 3246 (before balancing i.e., Nb.of added branches = 0) to 4591 (Tableau 6.2). As no other branch could be added from $N_{1\text{-DU6}}$, the script selects the DU6-A6's second node ($N_{2\text{-DU6}}$) as the source to create branches. Although $N_{2\text{-DU6}}$ has 24 downhill nodes, only 6 could be used as the rest were already used by other nets in the design. After creating 6 extra branches, the $Nb_{occ}$ increased to 4948 (Tableau 6.2) but still below $Nb_{mean}$. The third node ($N_{3\text{-DU6}}$) is selected to be used. Ten extra branches were then added from $N_{3\text{-DU6}}$, increasing $Nb_{occ}$ to 5311. These created branches are between DU*i* and A*i* and they have an almost negligible effect on $T_{osci}$. Indeed, in our design, the net DU*i*-A*i* and the net between DU*i* and DU*i*+1 (part of the ring oscillator) share at most one switch matrix input node. Our experiments show that after adding 32 branches from $N_{1\text{-DU6}}$, $N_{2\text{-DU6}}$ and $N_{3\text{-DU6}}$, an increase of 0.1% was observed on $T_{osci}$.

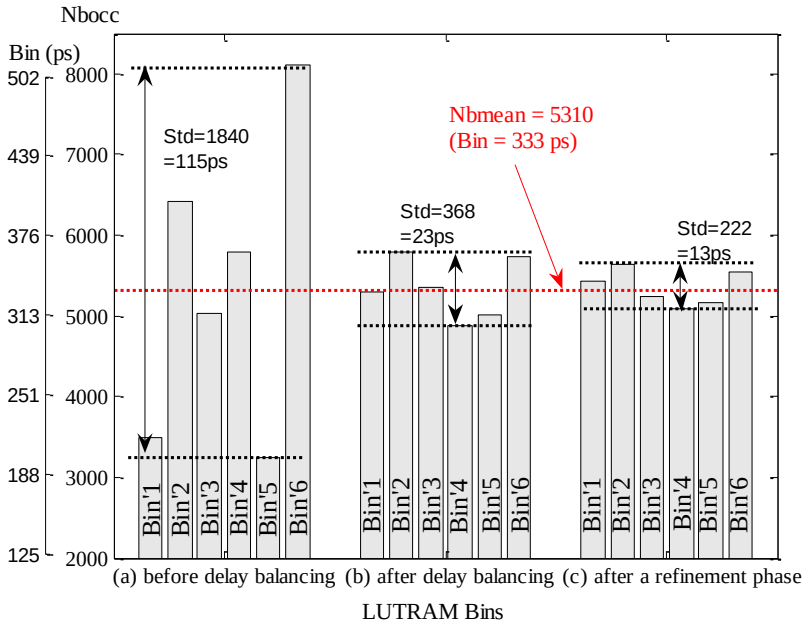



Figure 6.13 Measurements of LUTRAM Bins

Similarly, the delay tuning script was then applied on net DU2-A2 as the updated (after increasing Bin'$_5$) $Nb_{occ}$ associated to Bin'$_1$ is smaller than $Nb_{mean}$. A total of 13 and 8 branches were respectively added from the first and second nodes ($N_{1\text{-DU2}}$ and $N_{2\text{-DU2}}$) of DU2-A2, increasing $Nb_{occ}$ from 3491 to 4795 then 5346 respectively (Tableau 6.2). As $Nb_{occ}$ reached or exceeded $Nb_{mean}$, the

script stops creating branches from DU2-A2. Figure 6.14 shows the resulting created branches from the net DU2-A2. A total of 1 DSP, 1 BRAM and 19 LUTs cells were created and placed in order to terminate the added branches. Finally, eight added branches to DU4-A4 net were sufficient to increase $Nb_{occ}$ of Bin'$_3$ from 5030 to 5361 (Tableau 6.2). Figure 6.13(b) shows the number of code occurrences associated to each bin at the LUTRAM side after delay balancing. It is shown that the standard deviation dropped from 1840 (corresponding to 115 ps) (Figure 6.13(a)) to 368 (23 ps) (Figure 6.13(b)). A drop of 92 ps was achieved by only increasing Bin'$_1$, Bin'$_3$ and Bin'$_5$. To further drop the standard deviation between the bins, a refinement phase was performed to enlarge bins of Figure 6.13(b) that are below $Nb_{mean}$. Five, two and three branches were added to bins Bin'$_1$, Bin'$_5$ and Bin'$_4$ respectively to drop the standard deviation from 23 ps to 13 ps (Figure 6.13(c)).

Tableau 6.2 The $Nb_{occ}$'s evolution of Bin'$_1$, Bin'$_5$ and Bin'$_3$ ($Nb_{mean}$ = 5310)

| | After delay balancing | | | |
|---|---|---|---|---|
| | Nb.of added branches | Source node | $Nb_{occ}$ | Added cells |
| Bin'$_1$ | 0 | – | 3491 | – |
| Bin'$_1$ | 13 | $N_{1\text{-DU2}}$ | 4795 | 1DSP/19LUTs/ 1BRAM |
| Bin'$_1$ | 8 | $N_{2\text{-DU2}}$ | 5346 | |
| Bin'$_5$ | 0 | – | 3246 | – |
| Bin'$_5$ | 16 | $N_{1\text{-DU6}}$ | 4591 | 4DSPs/1FF/ 1BRAM 24LUTs |
| Bin'$_5$ | 6 | $N_{2\text{-DU6}}$ | 4948 | |
| Bin'$_5$ | 10 | $N_{3\text{-DU6}}$ | 5311 | |
| Bin'$_3$ | 0 | – | 5030 | – |
| Bin'$_3$ | 8 | $N_{1\text{-DU4}}$ | 5361 | 7LUTs |

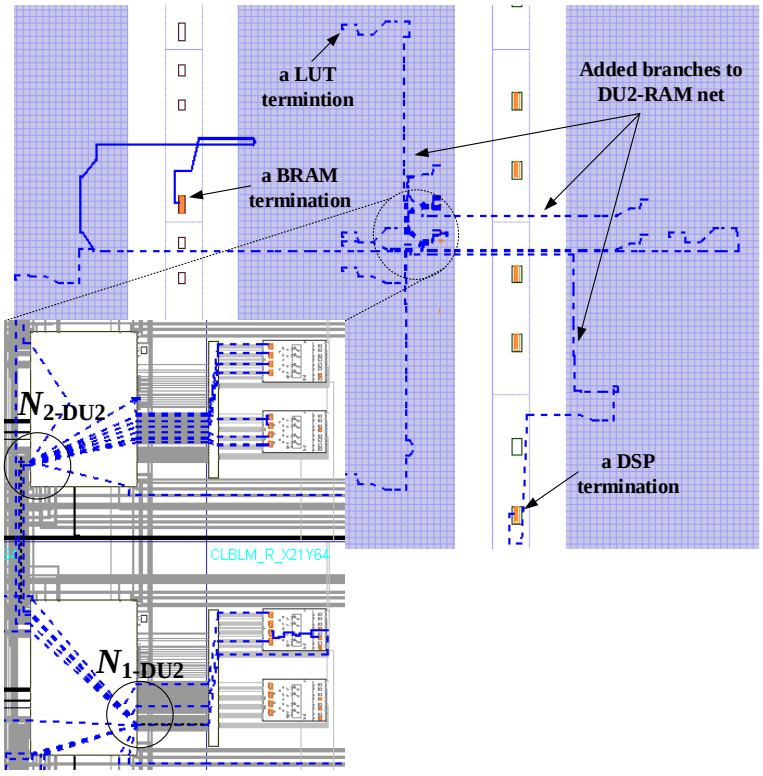



Figure 6.14 Netlist after the delay increase step of Bin'$_1$ (net DU2-A2) with 21 added and terminated branches using $N_{1\text{-DU2}}$ and $N_{2\text{-DU2}}$ as source nodes

Figure 6.15 shows an example with a random set of 100 measurements of $Nb_{occ}$ associated to Bin'$_1$ where the standard deviation is equal to STD=50.5 (equivalent to 3.17 ps according to equation (6.4)).

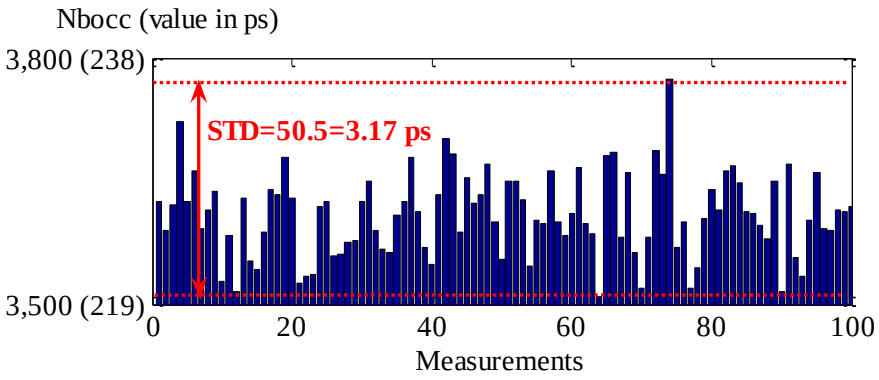



Figure 6.15 Example showing the raw $Nb_{occ}$ associated to Bin'$_1$ obtained in a set of 100 measurements

A total of 61 paths and their respective terminations were added to the fully placed and routed design in order to widen the three smallest bins (Bin'$_1$, Bin'$_3$ and Bin'$_5$) at the LUTRAM side. Figure 6.16 depicts the evolution of $Nb_{occ}$ over the number of added branches.

The differential and integral non-linearities (DNL and INL) of the oscillator as well as for the LUTRAM bins before and after balancing are shown in Figure 6.17. The worst DNL dropped from 0.51 LSB to 0.05 LSB (blue and red plots) while the worst INL dropped from -0.54 LSB to 0.06 LSB after delay balancing (LSB=333 ps). They have both been improved by factors of 10.2 and 9 respectively. Future work will focus on the effect of the temperature on the delay introduced by the added branches and explore better online calibration approaches.

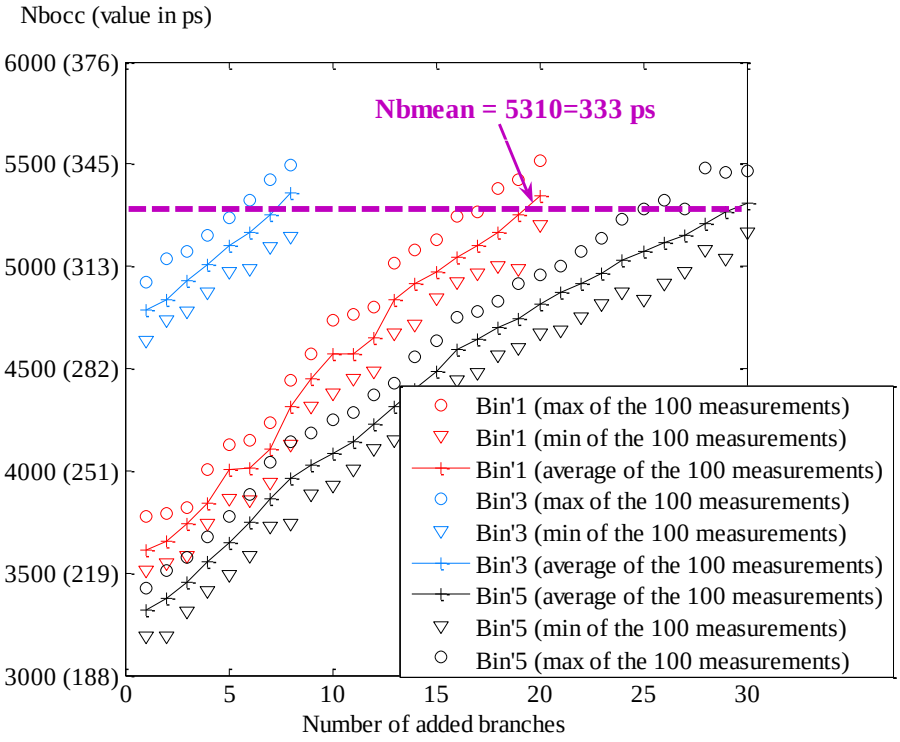
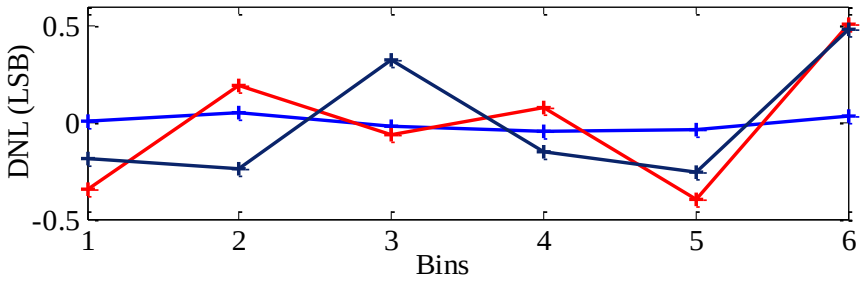
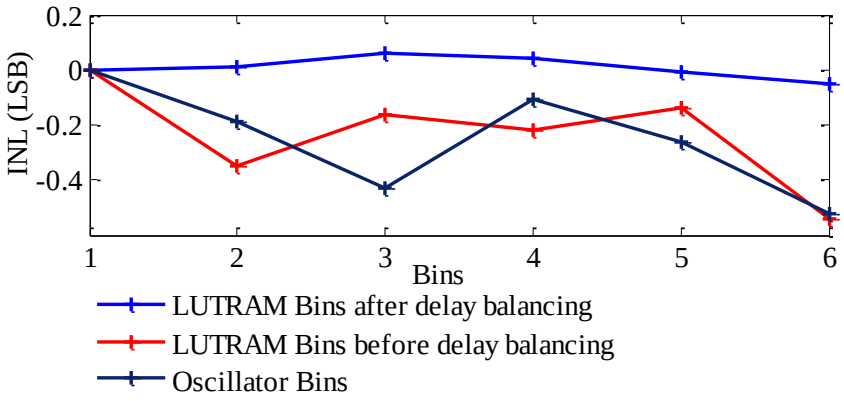



Figure 6.16 Evolution of $Nb_{occ}$ for Bin'$_1$, Bin'$_3$ and Bin'$_5$




(a)




(b)

Figure 6.17 DNL (a) and INL (b) of the oscillator bins and LUTRAM bins before and after delay balancing

Notice that, the execution time of the proposed scripts is dominated by the Bel_Input node and Bel termination search processes, described in Figure 6.3 and Figure 6.5 respectively. Tableau 6.3 summarizes the execution time of each process in our tests._Our scripts were run on a computer composed of eight Intel ® Core ™ i7-6700HQ CPUs running at 2.60 GHz and 8192 MB of memory.

We suspected that the Vivado static timing analyzer tool (STA) could help to estimate the delay introduced by a given subset of nodes. However, the delays extracted by STA are too large to be used. For example, when sequentially adding nodes (that span over 1 or 2 tiles) to DU2-A2 net, $Bin'_1$ extracted from the STA tool is always increased by 10 ps (Figure 6.18). However, the increase observed by the experimental method varies between 6.7 ps and 13.6 ps (Figure 6.18). The experimental method used in this paper goes beyond the limitations of the STA tool.

Tableau 6.3 Estimated time associated to each test process

| Process | Estimated execution time |
|---|---|
| Search_Bel_Input + Search_Bel Termination + bitstream generation | More than 2 minutes |
| Programming the FPGA + launching ILA | 6 seconds |
| Building 100 histograms | 100 seconds |

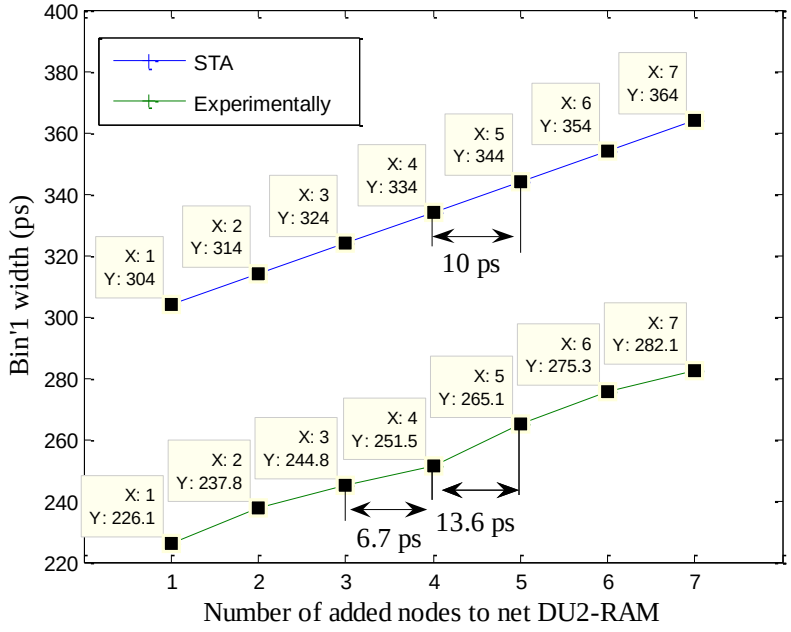



Figure 6.18 Bin'$_1$ vs the number of added nodes to net DU2-RAM (extracted experimentally and from the Vivado STA tool)

Our results were compared to some recently reported works in the literature (Tableau 6.4) and showed that with our proposed delay tuning method, the obtained linearity is competitive with state-of-the-art methods. In fact, in [60], a TDC was implemented in a Cyclone-IV FPGA based on carry LUTs present in logic elements (LEs). With timing adjustment and bin realignment as well as an online bin-by-bin calibration, the achieved DNL and INL are still larger than those obtained in this work. The two-step pipelined TDC presented in [20], continuously compensates the nonlinearity by updating bins in a LUT using the code density test. Despite the additional required circuitry (duty cycle estimator, bin estimator, a prefix adder and a calibrated fine time calculator among others), the worst DNL and INL achieved are equal to 1.91 LSB and 3.93 LSB respectively

(for an LSB equal to 10 ps). TDCs reported in [15] and [61], based respectively on averaging multiple tapped delay lines and a run-time calibration scheme, also achieved a DNL and INL below the proposed ones in this paper. The TDC in [62] shows the best linearity performances as it integrates several innovative methods such as the delay line averaging methodology, a compensation architecture and a mixed calibration method. It was also implemented in a newer process-technology FPGA. However, in order to achieve these performances, 96 TDC channels were implemented using more than 25% of the available slices in the FPGA. When compared to TDCs implemented in a similar FPGA ([63], [13]), our results show a better DNL with less resources.

Tableau 6.4 TDCs linearity comparison table

*Worst values, NS: Not Specified, [1]CARRY, [2]LUTs, [3]FFs, [4]BRAMs, [5]I/O ports, [6]clocking, [7]DSP

| [Ref] | FPGA | LSB (ps) | DNL* (ps) | INL* (ps) | Temp (°C) | Method | Resources |
|---|---|---|---|---|---|---|---|
| [60] | Cyclone-IV | 45 | -22.5 | -21.6 | NS | TDL, calibration by bin realignment, bin-by-bin, double registration | NS |
| [20] | Virtex-6 | 10 | +19.1 | +39.3 | 20 | Dual phase TDL, calibration LUT | NS |
| [15] | Virtex-6 | 24 | +19.2 | -24 | 20 | TDL, multichain averaging | NS |
| [61] | Virtex-5 | 46.8 | +48.6 | -199.8 | 27 | TDL-TDC semi-continuous calibration LUT | 2% of the slices,2%[2],1%[3] |
| [62] | Ultrascale | 5 | -0.60 | +2.40 | NS | Sub-TDL averaging Histogram compensation | 25.34%[1],28.2%[2],23.6%[3],24%[4] |
| [63] | Kintex-7 | 280 | +168 | NS | NS | Multi-sampling with quad phase clocks | 2.1%[2],1.5%[3],10.8%[4],10.3%[5],43.8%[6] |
| [13] | Kintex-7 | NS | +40 | NS | NS | TDL, bin realignment, Wu A [17] calibration LUT | 85% of the slices,80%[4] |
| This work | Zynq-7 | 333 | +16.65 | +19.98 | 37 | Ring-oscillator based TDC Calibration by fine delay tuning | 7%[2],5%[3],5%[4],50%[6],5%[7] (includes an 8192-deep ILA) |

Only a portion of the MMRO-TDC was used in this work to present the characterization and balancing processes using the proposed delay tuning scripts. For a full operating mode, a LUTRAM should be associated to each hit signal as depicted in Figure 2 of [28] as well as a counter to measure

the coarse time interval between the signals under test. Future work will perform precision measurements on the complete MMRO-TDC to report time intervals and single shot precision with our balancing approach.

## 6.5 Discussion

In this work, branches are blindly added to the route, i.e., without considering their specifications in terms of directions, the number of tiles they span or their location in the FPGA. Considering these parameters, each branch would introduce a specific amount of delay when added to the route. Future work will perform a prior detailed characterization step of the delay introduced by each node. Therefore, the delay tuning script could be based on a selective approach to choose the suitable set of nodes to achieve the target delay faster and with better accuracy. Furthermore, improvements could be applied to the presented TCL scripts such as adding a constraint to restrict the search space for terminations on some specific regions in the FPGA to minimize routing congestion.

The delay balancing problem for one bin was defined in this paper as finding a subset of nodes from a set of nodes where the absolute difference between the sum of node delays and the target delay is minimal. This problem requires testing all the possible combinations of nodes. Investigations will be conducted in our planned future work in order to minimize the complexity of this combinational problem and the convergence time of the proposed delay tuning scripts.

The proposed balancing approach could be applied to any TDC implemented in a Xilinx FPGA supported by the Vivado design tool (Virtex-7, Kintex-7, Artix-7, Zynq7000 and Ultrascale) to improve its linearity. For example, in a classical tapped delay line (TDL) TDCs [49] implemented in one of these FPGAs, our delay tuning scripts could be applied to balance delays between the TDL cells or to reduce the clock skew between the sampling registers. Notice that the proposed delay tuning method could not be applied at this point on Intel and Microsemi FPGAs as we do not have access to the internal routing structures of the switch matrices via TCL commands.

Although the delay tuning script proposed in this paper was only applied on a time-to-digital converter, there are many other applications, requiring fine delay control between two given points. For instance, delay tuning can be useful to implement Physically Unclonable Functions (PUFs)

[25] or True Random Number Generators (TRNGs) [42]. Other applications will be addressed in the future work.

## 6.6 Conclusion

This paper presented a method to finely tune the propagation delay of FPGA routes. For a given route, it increases the fanout of its switch matrices input pins by adding further capacitive loads contributed by unused interconnects. A set of TCL scripts were presented in this paper aiming to create routes starting from the added nodes and ending to searched FPGA device resources such as LUTs, FFs, CARRY4s, DSPs or BRAMs. A multi-hits TDC with an initial poor linearity (MMRO-TDC) was used as an application, where the presented TCL scripts for delay tuning were applied to drastically improve its linearity by reducing the DNL and INL by factors of 10.2 and 9 respectively.

## 6.7 Acknowledgment

We thank the Natural Sciences and Engineering Research Council of Canada for their financial support.

# CHAPITRE 7 ARTICLE 4 - RING-OSCILLATOR BASED HIGH ACCURACY LOW COMPLEXITY MULTICHANNEL TIME-TO-DIGITAL CONVERTER ARCHITECTURE FOR FIELD-PROGRAMMABLE GATE ARRAYS

**Abstract**

This paper proposes and validates a low complexity multichannel ring-oscillator based Time-to-Digital Converter (TDC) architecture for field programmable gate arrays. Channels of that TDC are mainly composed of Look-Up Tables (LUTs) configured as embedded memories. The channels share a same ring oscillator. A previously proposed delay tuning method was used to increase the overall accuracy by balancing the TDC channels. Compared to previously reported TDCs, the proposed architecture consumes less resources without degrading performances. A nine-channel TDC was implemented in a Zynq 7 Xilinx FPGA. Single-shot precision of 92.7 ps and accuracy of 92.9 ps were achieved, while consuming 1.49% and 1.31% of the available LUTs and FFs of a ZYNQxc7z010-3clg400 Xilinx FPGA respectively.Keywords: Accuracy, Delay tuning, FPGA resources, Multichannel, Multihit, Time-to-digital converter.

**Authors:** Safa Berrima, Yves Blaquière and Yvon Savaria
**Submitted in**: IEEE Transactions on Instrumentation and Measurement (May 08th 2021)

## 7.1 Introduction

Field Programmable Gate Array (FPGA)-based Time-to-Digital Converters (TDCs) are attractive for many applications due to their low development cost, flexibility and short time to market [64][41][49]. Many such TDCs were reported in the literature. The simplest is a counter triggered by the arrival of a start signal and stopped by the arrival of a stop signal. The resolution of such TDC is equal to the period of the counter's clock. In modern FPGAs, that period cannot be less than 1250 ps [65].

To achieve a higher resolution and a large dynamic range, the Nutt architecture [8] is frequently used. It consists of one fine TDC to measure the time between a start signal and the subsequent system clock edge ($T_{start\text{-}clk1}$) and a second fine TDC to measure the time between a stop signal and

the subsequent clock edge ($T_{stop-clk2}$). A coarse TDC (synchronous counter) measures the delay between the two system clock edges ($T_{clk1-clk2}$). The time interval between start and stop is given by: $T_{clk1-clk2} + T_{start-clk1} - T_{stop-clk2.}$

The Nutt architecture requires two fine TDCs per channel to measure the time interval between start and stop. A fine TDC could be a Tapped Delay Line (TDL) [20], where the start signal is propagated through delay elements such as inverters. The stop signal samples the delay line. The time resolution of this architecture is lower-bounded by the propagation delay through one delay element. To achieve a finer resolution, carry chains are usually exploited as delay lines. In [66][67][68], resolutions of 3.29 ps, 10 ps and 20.5 ps were achieved in Kintex-7 Ultrascale, Virtex-4 and Lattice FPGAs. Other fine TDCs architectures derived from the TDL were introduced such as the Vernier Delay Line (VDL) [69].

Delay-line based TDCs suffer from the fact that after a signal has propagated through a number of gates, the errors introduced by random deviations of the gate delays accumulate along the line. In other words, as non-linearities accumulate through the delay line, it is desirable to have a delay line as short as possible. On the other hand, long delay lines are needed to obtain large dynamic ranges, which introduces an important area overhead.

To address these issues, TDCs based on looped delay lines (typically ring-oscillators) were introduced. These TDCs are called RO-TDCs. In [70][71], a RO-TDC is based on two rings oscillating at slightly different frequencies and the TDC's resolution is given by their frequency difference. Two RO-TDCs variants have been proposed. One uses two counters, each incremented by one of the oscillators, and a phase detector [72] while the other employs only one counter that is clocked by the slow oscillator and that stops counting once the fast oscillator gets ahead of the slow one [73].

In the FPGA-based TDCs presented in the literature, the main source of non-linearity that was considered is the uneven bin sizes. In fact, in an ideal fine resolution TDC, all delay elements that are used for time quantization should have equal bin widths. However, due to internal routing delay uncertainties and process-voltage-temperature (PVT) variations, the TDC has different bin widths. Unlike ASIC-TDCs in which voltage-controlled delay cells and voltage-controlled oscillators are well matched and can be used directly without calibration, in FPGA-TDCs, the observed non-

linearities are large and complex calibration is indispensable. To mitigate this issue, multichannel averaging architectures were explored by several authors [62][18][19].

By using multiple TDC channels to measure the same signal, different thermometer codes can be obtained. Averaging these measured values reduces the system's non-linearity. Moreover, the average cell propagation delay is also reduced, which allows improving the time resolution. The results can be further improved by increasing the number of channels used to obtain distinct time measurements of a single signal.

However, when implementing multichannel TDCs, the routing of the signal to be measured must minimize the offset between channels. Thus, for each channel, the routing delays from the start signal to the channel ($d_{start \rightarrow channel}$) and the routing delay from the stop signal to the channel ($d_{stop \rightarrow channel}$) must be equalized, otherwise the measured time would include a systematic error $\varepsilon$ defined by: $|\varepsilon| = |d_{stop \rightarrow channel} - d_{start \rightarrow channel}|$. Even with rigorous manual routing, the paths delays $d_{stop \rightarrow channel}$ and $d_{start \rightarrow channel}$ can differ by thousands of picoseconds, which considerably degrades the accuracy. To the best of our knowledge, the majority of previous works do not report the resulting accuracy. TDCs found in the literature essentially report the repeatability of the measurements.

Usually, the multichannel averaging method is costly in FPGA resources as TDC channel resources are replicated, especially with the Nutt architecture, where two fine TDCs are deployed per TDC channel. However, as the MMRO-TDC presented in [28] has a small area overhead, the resulting multichannel TDC would consume fewer resources than recently reported multichannel TDCs [63][74][75].

The multichannel MMRO-TDC adopted in the present work saves resources by implementing channels with Look-Up Tables (LUTs) configured as memories [38]. All the TDC channels share the same ring oscillator. Moreover, this TDC supports the multihit feature by measuring the arrival times of *m* hit signals. Furthermore, this work considers the propagation delays between the start and stop signals to the MMRO-TDC channels when reporting accuracy. The Fine-Tuning Delay (FTD) introduced in [39][46] were used to improve measurement accuracy.

The main contributions of this paper are as follows:

- A low complexity multihit and multichannel RO-TDC is proposed;

- The impact on accuracy of the difference between routing delays of the start and stop signals to the TDC channels is studied;
- The accuracy of the multichannel MMRO-TDC is improved by using the FTD method introduced in [46][39]. Proper mitigation of delay differences between the start/stop signals and the TDC relaxes TDC placement constraints and enables moving TDCs away from FPGA pins at more convenient locations.

The paper is structured as follows: section 7.2 describes the design of the multichannel averaging MMRO-TDC and section 7.3 presents its implementation in a Zynq-7 FPGA. Sections 7.4 and 7.5 describe the time performances and the FPGA resource requirements of the proposed TDC. Section 7.6 discusses some important aspects of the paper and proposes promising future directions. Section 7.7 concludes by summarizing the main findings and results from this work.

## 7.2 Multichannel averaging measurement MMRO-TDC Circuit

The proposed multichannel averaging MMRO-TDC design is depicted in Figure 7.1. This circuit was implemented on a 7-series Xilinx FPGA. It could be implemented on any Xilinx FPGA family supporting the distributed RAM feature, i.e., LUTs implemented as synchronous RAM resources, with minor changes. The proposed TDC is composed of $n$ channels, each containing $m$ fine interpolators. The incoming hit signals SUT1 to SUT$m$ initiate multiple interpolations. Each hit signal SUT$i$, $i \in [1;m]$, is fed into the fine interpolator (FI$ik$, $k \in [1;n]$) of each channel $k$. A channel $k$ generates the timestamp TS$ik$ corresponding to SUT$i$'s transition arrival time. The difference between the arrival times of SUT$i$ and SUT$j$ recorded by a channel $k$ is given by the difference between TS$ik$ and TS$jk$, $i,j \in [1;m]$, $k \in [1;n]$. A total of $n$ differences are produced and their average is taken as the final measurement result.

The proposed TDC has coarse and fine measurement granularities, described in sections 7.2.1 and 7.2.2 respectively.

### 7.2.1 Coarse measurement granularity

A shared free running ring oscillator (RO) is implemented with a series of delay units (DUs), made from LUTs or carry-chain primitives [38] for better resolution. One DU could implement an inverter, while the others implement buffers, or all the DUs implement inverters with an odd number of DUs. This oscillator produces the RoClk clock that drives the CoarseCounter that counts

the number of cycles until a rising edge on SUT*i* happens. Its value is sampled at the arrival of SUT*i* and stored in respective CoarseMeas registers (see Figure 7.1).

As the period of the RO needs to be precisely measured to translate the TDC discrete output to a temporal value, the RO's frequency should not be faster than the FPGA and IOs' ratings. Let $T_{RoClk\_max}$ be the maximum measurable RO frequency and $d_{DU}$ be the propagation delay through one DU. The minimum number $N_{DU}$ of DUs in the oscillator is defined by equation (7.1):

$$N_{DU} = \frac{T_{RoClk\_max}}{2 \times d_{DU}} \tag{7.1}$$

Indeed, $N_{DU}$ increases as $d_{DU}$ decreases. Therefore, a carry chain-based RO uses more DUs than a LUT-based RO. Also, the resources used by one fine interpolator increase as $N_{DU}$ increases. In fact, each fine interpolator implements an $N_{DU}$-bit address LUTRAM and $2^{N_{DU}-6}$ LUTs are needed to build an $N_{DU}$-bit address LUTRAM [28]. To get a small area overhead, the RO is implemented with six LUTs. The dynamic range of the proposed TDC is only restricted by the number of bits $N_D$ in the coarse counter. Note than in this work, a Gray counter is used rather than a binary one to reduce the counter sampling errors as only one bit toggles every cycle.

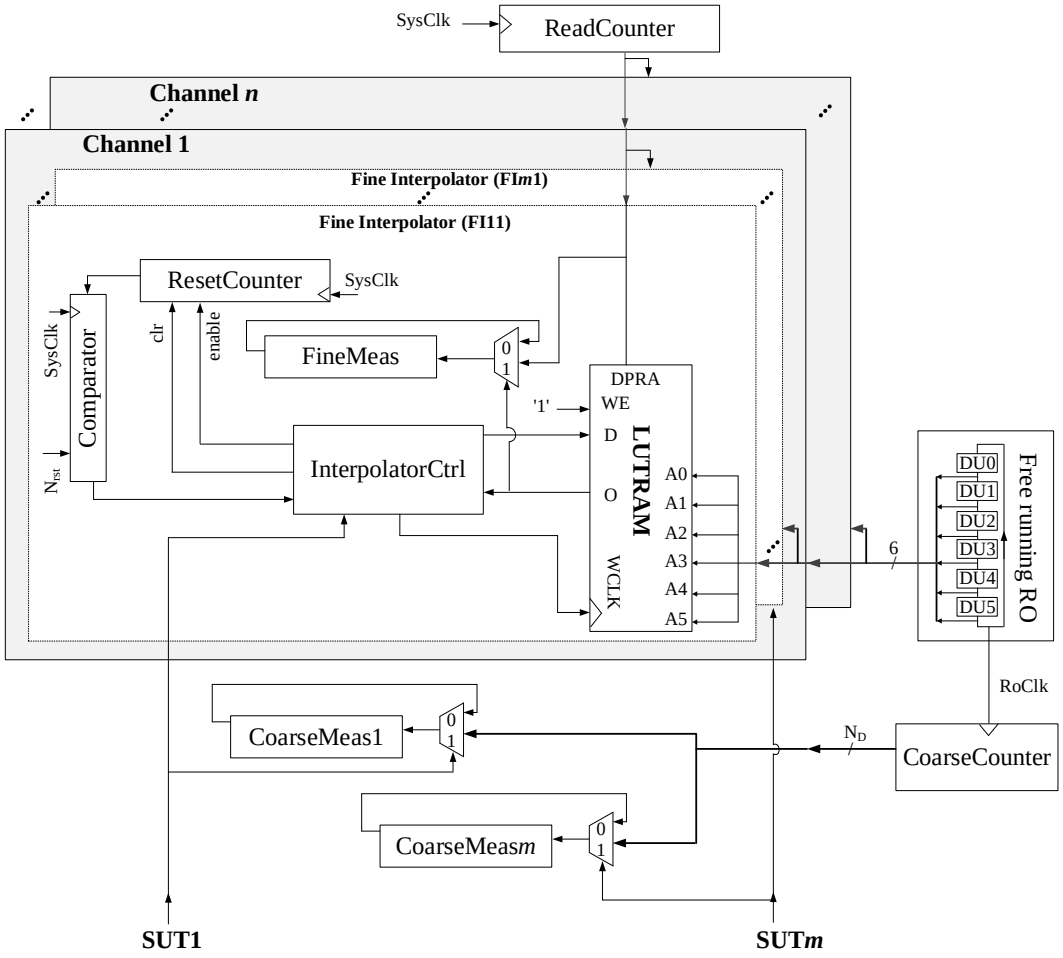



Figure 7.1 Circuit diagram of the Multichannel averaging measurement MMRO-TDC on a Zynq7 Xilinx FPGA

### 7.2.2 Fine measurement granularity

The fine measurement granularity is implemented by *n* identical channels as depicted in Figure 7.1. Each channel *k* contains *m* fine interpolatorsFI*ik*, each of which containing one Look-Up Table (LUTRAM) configured as a memory. Indeed, each 7-series FPGA SLICEM slices can configure their LUTs as distributed Random Access Memories (RAMs). The LUTRAM modules are configured as dual-port 64×1-bit RAMs with two independent write (A) and read (DPRA) ports address inputs.

In each channel of the MMRO-TDC, the free running ReadCounter continuously scans the data in the LUTRAM, which is completely read after 64 system clock periods. Each channel supports a capture and a reset mode, described in sections 7.2.2.1 and 7.2.2.2 respectively.

### 7.2.2.1 Capture mode

The purpose of this mode is to record the arrival time of a transition on the hit signal. The recorded time has a coarse granularity expressed by the number of RO cycles completed before the transition arrival and a fine granularity expressed by the progression through the RO. In this mode, the LUTRAM data input D is asserted high and the write clock WCLK is connected to the hit signal SUT*i*. Upon a rising edge on SUT*i*, a '1' logic is written in the LUTRAM at the address defined by the thermometer code in the free-running RO. The DPRA port driven by the ReadCounter continuously reads the LUTRAM. When the LUTRAM output O is asserted high, the ReadCounter value is sampled and stored in the FineMeas register. This data corresponds to the fine measurement associated to the arrival time of SUT*i*.

The timestamp TS*ik* produced by a channel *k* corresponding to the arrival time of SUT*i* is given by equation (7.2):

$$TS_{ik} = FineMeas + CoarseMeas_i \tag{7.2}$$

and the TDC output corresponding to the time delay between SUT*i* and SUT*j* is given by equation (7.3):

$$TDC_{ij} = \frac{1}{n} \sum_{k=1}^{k=n} (TS_{ik} - TS_{jk}) \tag{7.3}$$

In order to translate the thermometer codes stored in FineMeas register into temporal values (in ps), the statistical code density test [53] is first used to extract the bin sizes at the LUTRAM levels. Indeed, the bins' sizes at each LUTRAM are different from the bins' sizes at the oscillator level due to the different routing delays from the oscillator DUs up to the LUTRAMs write address pins. Thus, the TDC is fed by a periodic sequence comprising a large number of measured time intervals and the number of occurrences of each thermometer code is calculated. The bin size is proportional to the occurrences of its associated thermometer code. A detailed description of the method used to extract the LUTRAMs bin sizes can be found in [46].

### 7.2.2.2 Reset mode

After each measurement, the LUTRAM must be cleared. The TDC channel enters in the reset mode where the LUTRAM data input D is asserted low and the write clock WCLK is connected to the

system clock SysClk. The LUTRAM is flashed with a series of zeros during $N_{rst}$ system clock cycles found enough to completely reset the LUTRAM. Note that to make the LUTRAM clearable, one could use a set of multiplexers to connect the LUTRAM write address pins to a counter (to sweep the write addresses while writing zeros) during the reset mode and to the DUs during the capture mode. However, as the counter is asynchronous to WCLK (section 7.2.3), some LUTRAM addresses may not be reached and therefore the LUTRAM may not be totally cleared.

#### 7.2.2.3 Switching between capture and reset modes

When the TDC is in the capture mode, i.e., ready for a new measurement, it must be set to write a logic one into the LUTRAM at the rising edge of SUT*i*. After taking the measurement, the TDC must flash zeros into the LUTRAM at the rising edges of SysClk. Therefore, the data input D is switched from '1' to '0' and the clock input WCLK from SUT*i* to SysClk when the TDC mode switches from capture to reset. The InterpolatorCtrl (IC) module ensures this transition. The TDC timing diagram is described in Figure 7.2. Initially, at power up or system reset, the TDC is in the capture mode where D is asserted high and SUT*i* drives WCLK (phase A in Figure 7.2). Upon the O output assertion (i.e., a '1' logic is read at the address defined by DPRA), the LUTRAM switches to the reset mode where D is asserted low, SysClk is driving WCLK and the ResetCounter is enabled (phase B in Figure 7.2). The ResetCounter increments then each SysClk period. A synchronous comparator is used to compare, at each SysClk period, the output of the ResetCounter with the number $N_{rst}$ of clock cycles used for resetting the LUTRAM. When ResetCounter reaches $N_{rst}$, it is cleared and SUT*i* drives WCLK. The D input is also set to '1' (phase C in Figure 7.2). Therefore, the LUTRAM is ready for a new fine measurement.

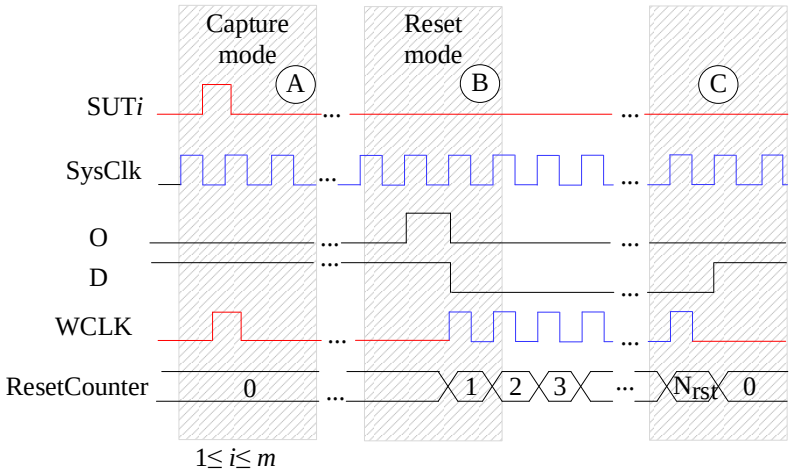



Figure 7.2 Temporal diagram of the MMRO-TDC circuit

### 7.2.3 Timing model and analysis of MMRO-TDC

To make this TDC usable, the difference of the pulse widths of hit signals SUT*i* must be smaller than one RoClk period, as shown in Figure 7.3. In fact, while SUT*i* is high, the CoarseCounter value is latched in the CoarseMeas register at each rising edge of RoClk (Figure 7.1). In the example depicted in Figure 7.3, the CoarseMeas associated to SUT*i* latches the value of *c*+1 at t0 and *c*+2 at t1. When SUT*j* pulse is shorter than SUT*i* (by more than one $T_{RoClk}$) (Figure 7.3), the CoarseMeas associated to SUT*j* latches the value of *c*+2 at t1, making the difference between the two CoarseMeas registers equal to zero instead of one. However, when the difference of pulse widths of SUT*i* and SUT*j* is smaller than one RoClk period, CoarseMeas associated to SUT*j*, latches the value of *c*+3 at t2, making the difference between the CoarseMeas registers equal to one. Note that a pulse reshaping circuit as the one presented in [71] could be used to get pulses of desired length. In addition, the hit pulses should be greater than the period of the RO oscillator.

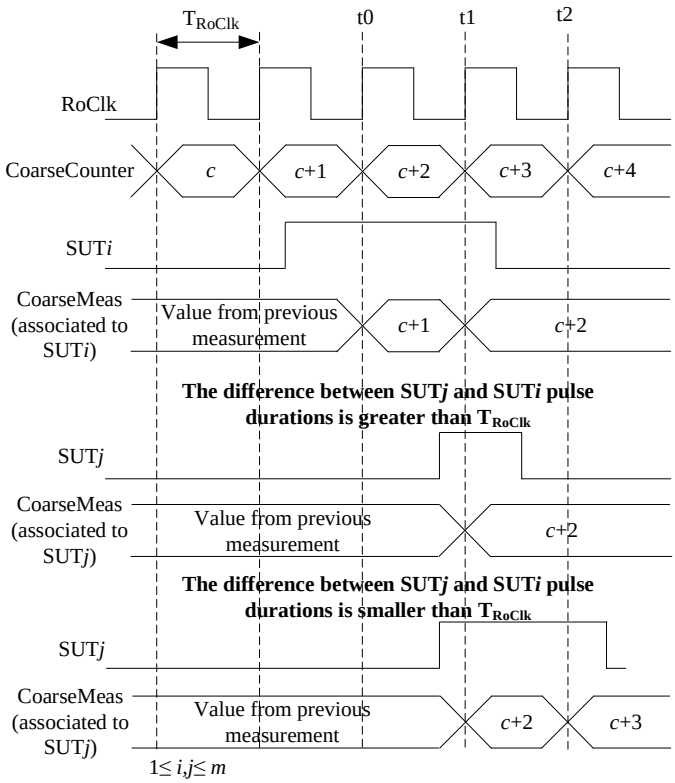



Figure 7.3 Effect of different SUT*i* and SUT*j* pulse durations on the coarse measurement accuracy (error rate)

As each LUTRAM timestamps the arrival time of the hit signal with a thermometer code, one has to ensure that the '1' logic is written in an address forming a thermometer code. For so, the routing delay ($d_{DUi\text{-}Ai}$) from DU*i* to address pin A*i* should be shorter than the delay ($d_{DUi+1\text{-}Ai+1}$) from DU*i*+1 to A*i*+1 plus the delay ($d_{DUi\text{-}DUi+1}$) between DU*i* and DU*i*+1 as shown in Figure 7.4. Notice that *i* goes from 0 to $N_{DU}$-1, with $N_{DU}$ defining the number of delay units in the oscillator. This condition must be satisfied for each oscillator stage and for each LUTRAM. For so, the *set_max_delay* and *set_min_delay* routing constraints [58] could be used. The FTD (presented in section 7.2.4) could also be applied on DU*i*-A*i* paths that could not satisfy these min-max routing constraints with the Xilinx Vivado router.

The precision and accuracy of the proposed TDC are mainly affected by two routing delay skews:

*DUi-LUTRAM routes skew*

The condition depicted in Figure 7.4 must be satisfied to ensure the proper TDC behavior. However the mismatch between the bin widths at the LUTRAMs levels causes a high non-linearity as

demonstrated in [46]. Indeed, the bin widths at the LUTRAMs levels are non-uniform due to arbitrary routing delays between the oscillator and the LUTRAMs. In [46], one LUTRAM bins were balanced by applying the FTD method (presented in section 7.2.4) on nets between the oscillator and the LUTRAM. It has been shown that the standard deviation of the LUTRAM bin widths dropped from 115 ps to 13 ps which makes the integral non-linearity drop from -0.54 LSB to 0.05 LSB. In this multichannel scheme, balancing the $N_{DU} \times m \times n$ nets between the oscillator and the LUTRAMs shall significantly reduce the non-linearity, *m* and *n* being the number of hit signals and channels respectively. However, non-linearity reduction is out of scope of this present paper.

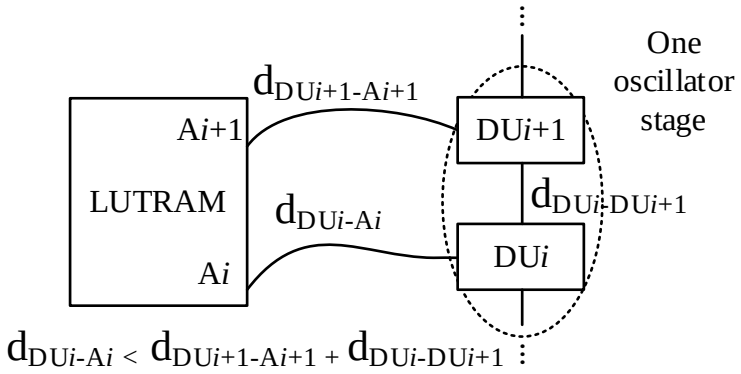



Figure 7.4 Routing delays in the ring oscillator and between the oscillator stage to the LUTRAM

*SUTi-LUTRAM routes skew*

SUT*i* is routed to WCLK through InterpolatorCtrl (IC) (Figure 7.1). The measured time interval has an uncertainty caused by routing delays between SUT*i*, IC and LUTRAM shown in Figure 7.5. IC is implemented in a LUT and it selects either the SysClk (reset mode) or SUT*i* (capture mode) to be routed to LUTRAM. Note that, as SysClk is routed to a LUT, it leaves the clock dedicated path and is connected to LUTRAM through standard interconnection network.

The routing delay skews ε defined by the difference between $d_{SUTi\text{-}LUTRAM}$ and $d_{SUTj\text{-}LUTRAM}$ must be controlled to reduce measurement error rate of our TDC (ε is one source of measurement errors among others such as the ones enumerated in section 7.6). Indeed, it exists a target skew defined by the difference between $d_{SUTi\text{-}LUTRAM}$ and $d_{SUTj\text{-}LUTRAM}$ ensuring a maximal closeness of measurements to the real value. For so, a fixed known time interval is applied on the TDC. If the average of the TDC outputs corresponding to the time delay between SUT*i* and SUT*j* defined by

TS*ik* – TS*jk* is smaller than the applied time interval, the delay $d_{SUTj\text{-}LUTRAM}$ is tuned otherwise the delay $d_{SUTi\text{-}LUTRAM}$ is tuned.

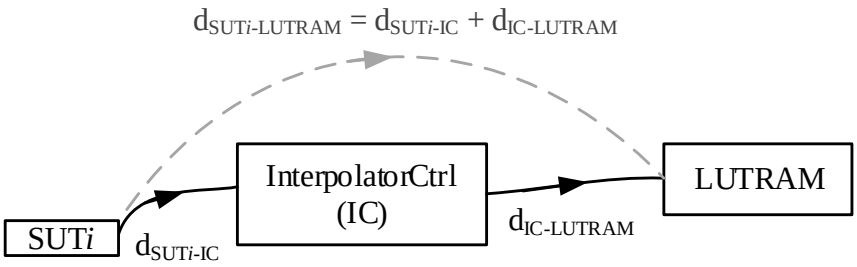



Figure 7.5 Routing delay skews between SUT*i* signal and the LUTRAM in the circuit of Figure 7.1

### 7.2.4 Fine-Tuning Delay method (FTD)

The Fine-Tuning Delay method principle is presented in Figure 7.6. It is based on increasing the capacitive loads on the switch matrices input pins by adding extra branches. In fact, the FPGA has a grid-like architecture made of tiles. An interconnect tile is made of one Wilton switch matrix (SM) that directs the signal to the routing fabric. These Wilton SMs are made of input and output pins. The input pins can direct the signal to multiple destinations at the same time. By creating new branches from an input pin being part of a given net $Net_A$, the net delay is increased. The created branches must be terminated by loads to avoid antennas in the design. In [46], TCL scripting algorithms were elaborated to automatically create branches and terminations.

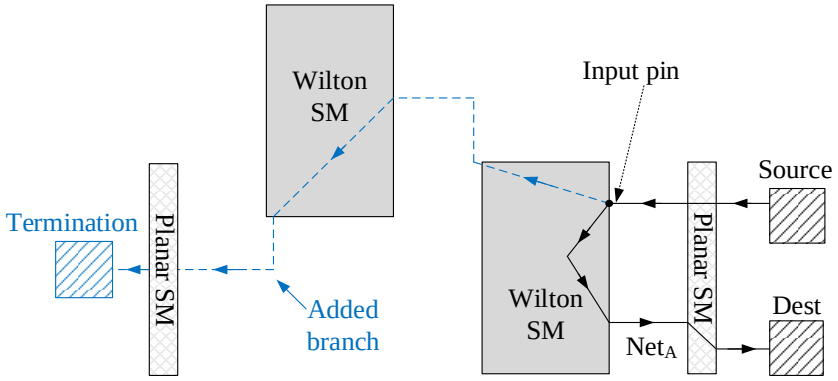



Figure 7.6 The FTD method principle [46][39] applied on the route $Net_A$ between Source and Dest: the delay of $Net_A$ is finely increased by adding extra branches from a Wilton SM input pin (only one extra branch is shown in this example)

## 7.3 Implementation of the MMRO-TDC

The Multichannel averaging measurements MMRO-TDC was implemented in a Zynq7-Z010 Xilinx FPGA using the Vivado 2015.4 tool. The ring-oscillator is made of six LUTs to minimize the FPGA resources as demonstrated in [28]. The delay unit DU0 performs a NAND operation while DU1-DU5 are buffers (Figure 7.1). The six DUs were placed in two adjacent slices using the LOC and BEL constraints to make the oscillator stages as uniform as possible [58]. The resulting oscillation period ($T_{RoClk}$) was measured using a 23-bit counter clocked at $T_{RoClk}$. The counter was enabled during 8192 cycles of a 6.25 MHz clock, generated from the 125 MHz quartz of the Zybo prototyping board using a MMCM (Mixed-Mode Clock Manager). The ring oscillator period $T_{RoClk}$ was measured to 3986 ps.

To conduct the experiments presented in this paper, two input signals SUT1 and SUT2 were used. The circuit shown in Figure 7.7 was used to generate SUT1 and SUT2 internally in the FPGA.

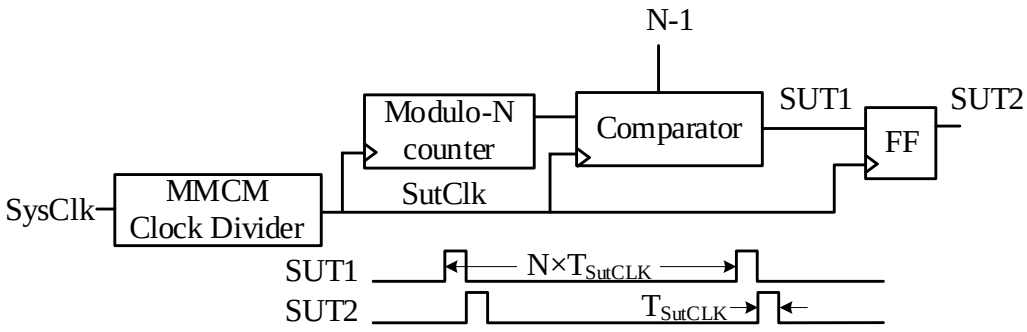



Figure 7.7 Circuit used to generate SUT1 and SUT2 pulses internally in the FPGA

Note that an external equipment could also be used for this purpose. The circuit of Figure 7.7 was configured to get precise 5 ns delay between rising edges of SUT1 and SUT2 by deriving a 200 MHz clock from the MMCM (SutClk).

A total of nine ($n$=9) MMRO-TDC channels were implemented in the FPGA, sharing the same ring oscillator. The CoarseCounter comprises 6 bits in this work ($N_D$=6) but could be enlarged for higher dynamic ranges.

## 7.4 Experimental results

This section presents the experiments conducted to show the impact of (1) the propagation delay

difference between the routes SUT1 and SUT2 on the TDC's accuracy, (2) the FTD on the TDC's accuracy and (3) the multichannel feature on the TDC's accuracy and precision.

The routing delays from SUT1 and SUT2 to their respective LUTRAMs (Figure 7.5) are extracted from STA while the TDC output is measured experimentally by reading the FineMeas and CoarseMeas registers using the Vivado Integrated Logic Analyzer (ILA) tool [58]. Equations (7.2) and (7.3) are then used to compute the TDC output.

The impact of the propagation delay difference between the routes SUT1 and SUT2 to their respective LUTRAMs is shown for one TDC channel in Figure 7.8. The propagation delays $d_{SUT1\text{-}LUTRAM}$ and $d_{SUT2\text{-}LUTRAM}$ are extracted from STA at the slow process corner. Note that the difference ($d_{SUT2\text{-}LUTRAM}$ - $d_{SUT1\text{-}LUTRAM}$) remains identical on slow and fast corners in STA.

The results shown in Figure 7.8 are extracted from 11 experiments E$i$ $\{i \in [1,11]\}$ where each time interval data corresponds to the average of 8192 repetitive TDC output measurements. Without delay balancing, the measured time interval is 4708 ps. To match the expected 5 ns time interval, the delay $d_{SUT2\text{-}LUTRAM}$ is increased with FTD by loading route nets on SUT2-LUTRAM route while maintaining the delay $d_{SUT1\text{-}LUTRAM}$ fixed. Indeed, the delay from SUT2 to the InterpolatorCtrl (IC) was finely increased by adding extra branches from switch boxes pins and terminations in each experiment E$i$ $\{i \in [1,11]\}$, as listed in Tableau 7.1. A total of 53 extra branches were added to increase the measured TDC time interval from 4708 ps to 5001 ps. The 5001 ps is obtained when the difference between $d_{SUT2\text{-}LUTRAM}$ and $d_{SUT1\text{-}LUTRAM}$ is set to 738 ps with FTD. The black curve in Figure 7.8 depicts the standard deviation (STD) of the 8192 measurements in each experiment E$i$ $\{i \in [1, 11]\}$. The STD is a measure of precision, used to describe how reproducible the TDC is in presence of noise. Note that STD is almost constant for each experiment, which varies between 246 ps and 258 ps. In fact, as shown in section 7.2.3, the precision of the proposed TDC (represented by the standard deviation of the measurements) is improved by balancing the LUTRAMs bins. However, as this is out of scope of this work, no improvement is expected on the STD and it remains constant for a one-channel TDC. The STD decreases with a multichannel topology as shown in the following experiments. The results reported in Figure 7.8 show that FTD can be used to minimize TDC measurement errors by matching its output according to the expected 5 ns SUT1-SUT2 delay. More importantly, this target compensates not only for the mismatch

between $d_{SUT1\text{-}LUTRAM}$ and $d_{SUT2\text{-}LUTRAM}$ but also for the mismatch between the propagation delays of the RO outputs to the two LUTRAMs.

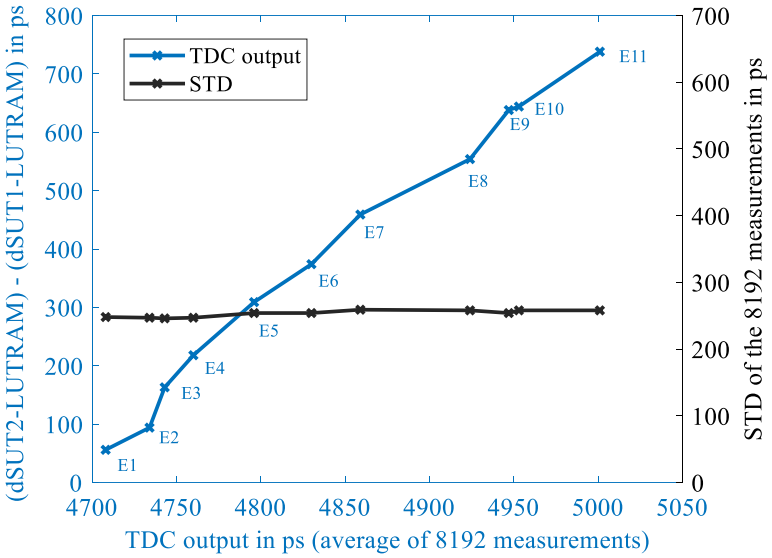



Figure 7.8 Variation of the TDC output and the standard deviation of the time interval measurements in function of ($d_{SUT2\text{-}LUTRAM}$ - $d_{SUT1\text{-}LUTRAM}$) for one channel

Tableau 7.1 Number of created branches and terminations used in the 11 experiments shown in Figure 7.8

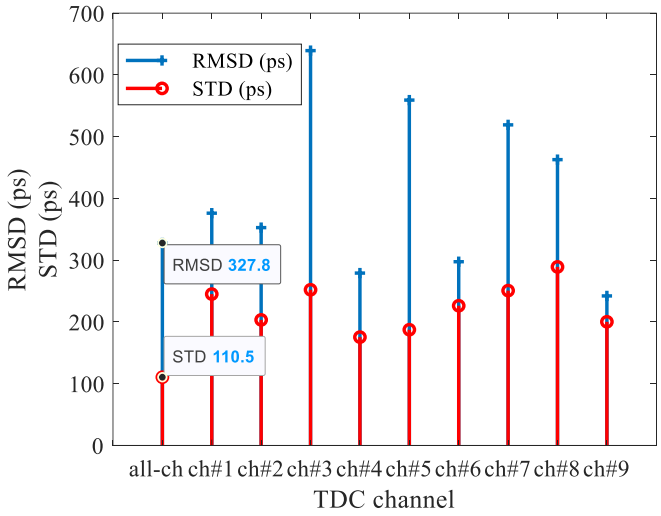
| Experiment | Nb. of created branches | Terminations |
|---|---|---|
| E1 | 0 | None |
| E2 | 3 | 3 LUTs |
| E3 | 8 | 4LUTs+3DSPs+1FF |
| E4 | 15 | 10LUTs+3DSPs+2FFs |
| E5 | 23 | 17LUTs+3DSPs+3FFs |
| E6 | 28 | 21LUTs+4DSPs+3FFs |
| E7 | 34 | 26LUTs+4DSPs+4FFs |
| E8 | 41 | 31LUTs+5DSPs+5FFs |
| E9 | 48 | 35LUTs+7DSPs+6FFs |
| E10 | 49 | 35LUTs+7DSPs+7FFs |
| E11 | 53 | 39LUTs+7DSPs+7FFs |

The Root-mean-square deviation (RMSD) is used in this work as a metric of the TDC's accuracy. It quantifies how data (TDC outputs) is concentrated around the expected time interval between SUT1 and SUT2 edges. Nine identical channels ($n$ = 9 in Figure 7.1) were implemented in the FPGA and the accuracy as well as the precision are shown in Figure 7.9 for each channel and for

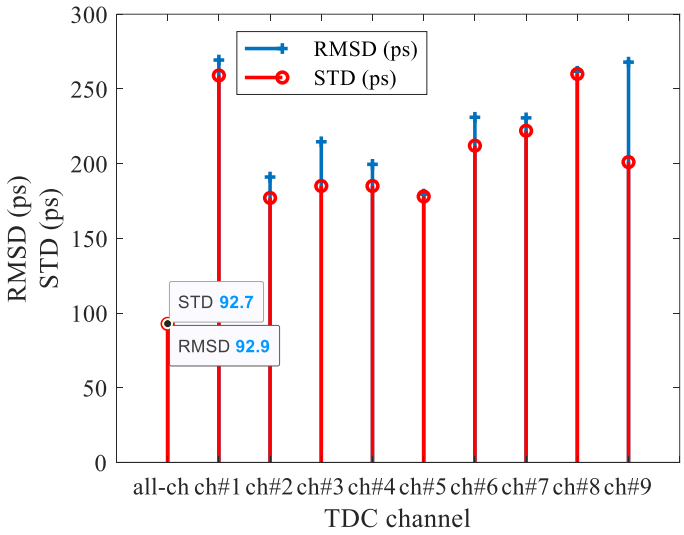
the 9 averaged channels (all-ch in Figure 7.9). Note that in the 9 averaged channels scheme, 9 timestamps are obtained, which after averaging, give the final timestamp. The averaging computation is externally post-processed with scripts. In each point depicted in Figure 7.9, a total of 8192 input time intervals of 5 ns were measured by the TDC. Figure 7.9 presents the RMSD and the standard deviation (STD) of the 8192 measurements. In Figure 7.9(a) the channels are unbalanced while in Figure 7.9(b) each channel is balanced.

The RMSD decreased by applying the balancing algorithms FTD and also when the averaging technique is used on the 9 channels. The RMSD and STD of the unbalanced 9-channel average (all-ch.) are 327.8 ps and 110.5 ps respectively, and reduce to 92.9 ps and 92.7 ps respectively when balanced.

Figure 7.10 shows the evolution of the RMSD and the STD compute on the 8192 measurements in function of the number of channels in the (a) unbalanced and (b) balanced multichannel measurement averaging TDC. As expected, the STD decreases as the number of channels increase; it is equal to 259.7 ps (and 245 ps) for a 1-channel balanced (and unbalanced) TDC and decreases to 92.9 ps (and 110.5 ps) for the 9-channel balanced (and unbalanced) TDC. The RMSD decreases as the number of used channels increases when balanced (Figure 7.10(a)). By definition, for data that has normal distribution, 68% of the data lies within one STD, i.e., for a set of measurements with an average AV and a standard deviation STD, 68% of the measurements lie within [AV-STD;AV+STD] (Tableau 7.2). As with balanced channels, the TDC outputs AV are close to each other (and close to the 5 ns target) and STD decreases with the number of channels increase, the ranges presented in column 4 are narrowed around the target 5 ns as the number of channels increases. Therefore, the RMSD decreases when $n$ increases.




(a)



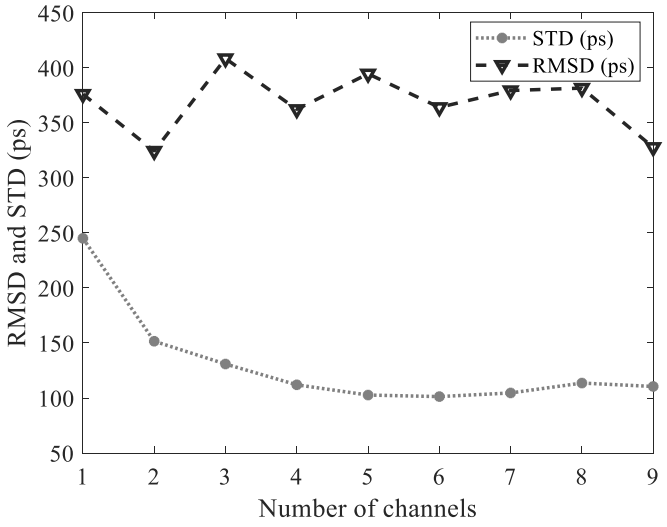

(b)

Figure 7.9 RMSD and STD of TDC outputs for each channel and their average over all channels (all-ch) (a) before route balancing (b) after route balancing

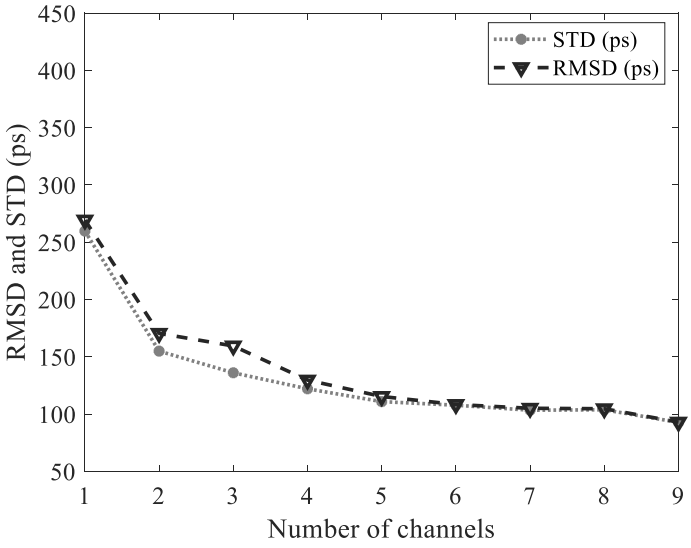
The random fluctuation of the RMSD for the unbalanced TDC (Figure 7.10(a)) is due to the AV that are arbitrary (depend on Vivado router) and not necessarily close to each other. Therefore, even if the STD decreases as the number of channel increases, the ranges are arbitrary.

When comparing the STD of the balanced and unbalanced TDC, they both decrease as the number of channels grow (i.e. the STD decreases if the number of channels increases). In fact, by having *n*

MMRO-TDC sample the same input signal, the system's non-linearities are reduced. The STD of the balanced and unbalanced TDC are almost equal for an equal number of channels. This is explained by the fact that the LUTRAMs bins are not balanced in this work, Nevertheless, the accuracy was improved from 327.8 ps to 92.9 ps by balancing nets SUT1-LUTRAM and SUT2-LUTRAM.




(a)




(b)

Figure 7.10 RMSD and STD in function of the number of channels *n* (a) unbalanced TDC (b) balanced TDC

Tableau 7.2 Average, standard deviation and the TDC output ranges in a balanced TDC (each line is associated to one point of Figure 7.10(b))

| Number of used channels | AV (ps) | STD (ps) | Output delay range for 68% of the measurements |
|---|---|---|---|
| 1 | 4928.5 | 259.7 | ∈ [4668.8 ps;5188.2 ps] |
| 2 | 4929.0 | 154.9 | ∈ [4774.1 ps;5083.9 ps] |
| 3 | 4916.8 | 136 | ∈ [4780.8 ps;5052.8 ps] |
| 4 | 4956.0 | 122 | ∈ [4834.0 ps;5078.0 ps] |
| 5 | 4968.0 | 110.8 | ∈ [4857.2 ps;5078.8 ps] |
| 6 | 4988.4 | 107.7 | ∈ [4880.7 ps;5096.1 ps] |
| 7 | 4981.3 | 103.3 | ∈ [4878.0 ps;5084.6 ps] |
| 8 | 4986.2 | 103.7 | ∈ [4882.5 ps;5089.9 ps] |
| 9 | 5007.3 | 92.7 | ∈ [4914.6 ps;5100.0 ps] |

There is a noticeable increase of STD for 7, 8 and 9 channels (Figure 7.10(a)). It is believed that this could be due to noise altering the measurements or to the STD reaching its limit. Considering a greater number of channels (> 9) could mitigate this observed increase.

## 7.5 FPGA area overhead

Figure 7.11 shows the 9-channel balanced TDC that was implemented. A total of 71 extra routes were created to balance the nine channels. Tableau 7.3 shows the FPGA logic utilization for the 9 channels, extracted from Xilinx Vivado with and without delay balancing. Notice that the resources used to internally generate SUT1 and SUT2 are also included in Tableau 7.3. Note that unused resources are exploited as part of the delay balancing process for the sole purpose of serving as node termination, thus their logical functionality remain unused.

Tableau 7.4 summarizes the proposed TDC's performance along with prior art contributions that use the tapped delay line (TDL) method rather than ring oscillator in this work. The proposed TDC is the method that uses the least resources per channel, while offering a competitive precision even with previously reported solutions requiring much more resources. In [63][74][75], the reported precisions are lower than the one achieved in this work while consuming more resources. The TDCs in [76][77] achieve a precision of less than 15 ps at the cost of high FPGA resources usage. One of the most attractive features of the proposed TDC is its low FPGA resource utilization. The present work achieves a high accuracy of 92.9 ps (RMSD). The obtained accuracy could not be compared

to the ones of these TDCs as their accuracies are not reported.

The resources needed to balance nine channels reported in Tableau 7.3 and Figure 7.11 (71 extra routes) are less than nine times those reported in Tableau 7.1 (53 extra routes) to balance one channel. This is explained by the fact that while balancing 9 channels, *set_min_delay* and *set_max_delay* constraints were used in order to direct the router to achieve delays $d_{SUT1\text{-}LUTRAM}$ and $d_{SUT2\text{-}LUTRAM}$ as close as possible to the targets that make the TDC output as close as possible to 5 ns. The FTD was then applied. This helped to reduce the resources consumption. Moreover, when branches are created from SUT*i*-LUTRAM to increase the channel *k* output, some other channels' outputs are also increased as nets from SUT*i* to the different channels may use common nodes. Therefore, the same resources are used to balance more than one channel. In this work, the reported accuracy (92.9 ps) was achieved by balancing channels 1, 4 and 7.

Note that the dynamic range of the proposed TDC is limited by the time required to reset the LUTRAMs as well as the time required to read the 64 addresses. In other words, the dynamic range of the proposed TDC is equal to $(N_{rst}+64) \times SysClk$.

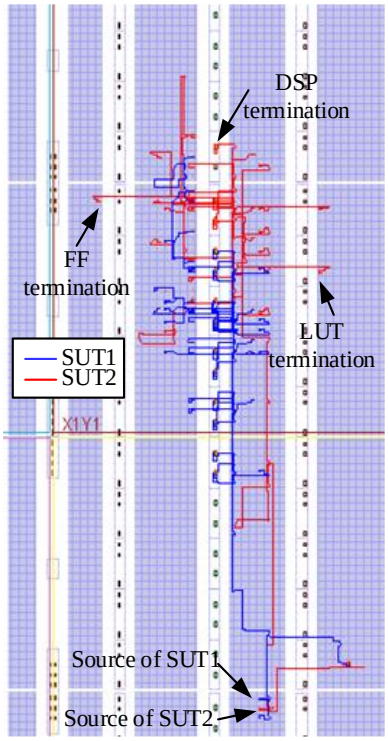



Figure 7.11 The layout of the 9-channel balanced TDC (43 LUTs, 13 DSPs and 15 FFs)

Tableau 7.3 FPGA resources utilization of the reported TDC for 9-channel unbalanced and balanced TDC

| Resources | Utilization (after balancing) | Utilization (before balancing) |
|---|---|---|
| Logic LUT | 264 | 221 |
| FFs | 479 | 464 |
| DSPs | 13 | 0 |

Tableau 7.4 Comparison with previous works on FPGA-based TDCs

| Ref | Method | FPGA | FFs* | LUTs* | Precision (ps) | Nb. of channels |
|---|---|---|---|---|---|---|
| [76] | TDL | Kintex-7 | 1641 | 577 | 8.13 | 16 |
| | | Virtex 6 | 1641 | 577 | 9.82 | 16 |
| | | Spartan 6 | 787 | 261 | 12.75 | 6 |
| [77] | TDL | Virtex 7 | 992 | 1578 | 4.8 | 4 |
| [63] | TDL | Kintex 7 | 742 | 545 | 280 | 8 |
| [74] | TDL | Virtex 5 | 274 | 68 | 255 | 32 |
| [75] | TDL | Artix 7 | 449 | 452 | 165 | 96 |
| This work | RO | Zynq-7 | 53 | 29 | 92.6 | 9 |

*per channel

## 7.6 Discussion

In the present paper, the FTD was applied on routes between the hit signals and the sampling elements (SUT*i*-LUTRAM) in order to minimize the TDC measurement error. The experiments show that the TDC output was successfully made close to the ideal output with an RMS error of 92.9 ps. Note that the accuracy could be further improved by equalizing the route delays between SUT*i*-LUTRAM and SUT*i*-CoarseMeas (from SUT*i* to the coarse counter sampler). In fact, at the arrival of SUT*i*, the coarse (represented by CoarseMeas) and fine (represented by LUTRAM) parts of the measurement must be taken with a minimal skew in order to minimize the measurement error.

As previously shown in [46], by making uniform the LUTRAMs bins widths, the differential and integral non-linearity decrease and therefore the measurement precision is improved. The LUTRAMs bins widths uniformization was achieved by applying the FTD algorithms on routes between the shared oscillator and the LUTRAMs (DU*i*-LUTRAM). Investigations will be

conducted in our planned future work to balance the DU*i*-LUTRAM as well as SUT*i*-LUTRAM with low routing complexity. Therefore, the precision as well as the accuracy could be improved.

The TDC performance metrics reported in this paper would vary due to process, voltage and temperature (PVT) conditions. The ring oscillator jitter would also make the oscillation period vary during the TDC operation. In our future work, the variation of the TDC metrics in function of the temperature will be conducted. Also, the performances will be extracted from multiple TDCs implemented in different regions of the FPGA and on different FPGAs. The impact of the oscillation frequency change on the TDC output will be studied.

In applications where the resolution is crucial, the LUT-based delay units making up the shared oscillator could be replaced by CARRY4-based ones. However, the resources used are increased as mentioned in section 7.2.3. On other hand, in order to reduce the energy consumption, the shared free running ring oscillator could be replaced by a gated one like in [36] that is enabled at the arrival of the first hit signal. However, the oscillator goes through a transitional and unstable phase at start-up. Future work will investigate techniques to reduce the energy consumption of the proposed MMRO-TDC.

## 7.7 Conclusion

This paper proposed a multichannel and multihit time-to-digital converter. Its proof of concept was implemented in a Zynq-7 Xilinx FPGA. The channels share a LUT-based ring oscillator and sample the arrival times of the hit signals using LUTs configured as distributed memories. A fine-tuning delay (FTD) method based on increasing the capacitive loading of switch matrices input pins was successfully applied to improve the accuracy of the proposed TDC. The FTD was used to balance the routes between the hit signals and the channels. Results show that the TDC root mean square error is equal to 327.8 ps and 92.9 ps for unbalanced and balanced 9-channel TDC respectively. The main feature of the proposed TDC is its low FPGA resource consumption compared to other works. It consumes only 53 FFs and 29 LUTs per channel. Possible future research work was also presented including studying the impact of PVT on the TDC performances and the evolution of the precision and accuracy when balancing the LUTRAM bins as well as SUT*i*-LUTRAM routes.

## 7.8 Acknowledgment

We thank the Natural Sciences and Engineering Research Council of Canada for their financial support.

# CHAPITRE 8 DISCUSSION GÉNÉRALE

Ce chapitre discute le travail de recherche présenté dans cette thèse. Il se divise en deux sections : discussion sur la méthode d'ajustement fin des délais dans les FPGA de Xilinx développée au sein de cette recherche et discussion sur le TDC proposé.

## 8.1 Ajustement fin des délais dans un FPGA de Xilinx

La méthode de contrôle des délais proposée dans cette thèse permet d'ajuster finement le délai de propagation à travers une trace en appliquant une charge capacitive au niveau d'un point d'interconnexion programmable d'une matrice de commutation faisant partie de la trace en question. L'application d'une charge capacitive revient à créer une branche de trace qui commence du point d'interconnexion programmable et se termine par une cellule de l'FPGA telle qu'une LUT, une bascule (FF), un processeur de signal numérique (DSP) ou un bloc mémoire (BRAM). L'approche a été décrite dans le chapitre 5 et les algorithmes développés en TCL ont été présentés dans le chapitre 6. Sa particularité consiste à n'utiliser que les ressources du FPGA non occupées. De plus, il est possible de contraindre les algorithmes à chercher des ressources dans une zone précise du FPGA ou à chercher un type de ressources spécifique (LUT, DSP, FF, BRAM) pour plus de flexibilité. Les algorithmes continuent de créer des branches tant que le délai désiré n'est pas atteint. Il fut montré qu'il est théoriquement possible d'atteindre n'importe quel délai tant que des ressources sont disponibles et que les branches créées ne causent pas de congestion dans le FPGA.

Afin de démontrer l'efficacité de la méthode de contrôle fin des délais proposée, elle a été appliquée sur un TDC proposé dans cette thèse (nommé MMRO-TDC et discuté dans la section 8.2) dans le but d'améliorer ses piètres performances temporelles. En effet, les divisions (*bins*) du TDC proposé (et de tous les TDC) ont besoin d'être le plus uniforme possible pour minimiser la non-linéarité. Les algorithmes d'ajustement des délais ont donc été appliqués pour ce fait. Dans le chapitre 6, il a été démontré qu'il est possible de réduire l'écart type de six divisions d'une valeur initiale de 115 ps à une valeur finale de 13 ps. À noter que l'écart type pourrait encore diminuer mais au coût d'itérations supplémentaires. En effet, comme mentionné dans le chapitre 6, les divisions au niveau d'une LUTRAM sont inter-reliées, c'est-à-dire en élargissant des divisions, d'autres divisions sont

affectées. En partant de divisions complétement débalancées (écart type de 115 ps), on procède d'abord par élargir les plus petites divisions en créant des branches de traces arbitraires. Ceci permet de diminuer l'écart type jusqu'à ce que les divisions soient rendues rapprochées (écart type de 23 ps tel que montré dans Figure 6.13). À ce stade, l'ajout de branches arbitrairement pourrait détériorer l'écart type. Il faudrait donc choisir les branches adéquates en essayant plusieurs configurations et procéder itérativement: si une branche détériore l'écart type, elle est remplacée par une autre branche. Autrement dit, rendu à ce stade, il faut procéder d'une manière heuristique. Le balancement des divisions est donc un problème combinatoire dont la solution optimale est hors de portée de cette thèse. Dans nos travaux, pour éviter un nombre élevé d'itérations heuristiques, on s'est arrêté à un écart type de 13 ps. Un écart type de l'ordre de la picoseconde, bien que théoriquement possible, serait entravé par la gigue produite par l'oscillateur en anneau, la variation de sa fréquence dans le temps ainsi que les variations des délais en fonction de la température et des procédés de fabrication.

La méthode d'ajustement des délais proposée dans cette thèse se distingue par rapport à celle proposée dans [23] par le fait qu'aucune librairie logicielle n'ait besoin d'être créée, simplement des commandes TCL prédéfinies par Xilinx sont utilisées. En effet, dans la méthode d'ajustement fin des délais présentée dans [23] (et améliorée dans [24]), une trace qui entre par un PIP peut rebondir sur plusieurs points d'interconnexions avant de sortir par un autre PIP donnant ainsi une multitude de délais possibles. Une telle approche nécessite la connaissance de la structure interne de la matrice de routage et des interconnexions entre les différents PIP. Cette approche nécessite aussi d'accéder à la mémoire de configuration de l'FPGA et programmer (activer/désactiver) les PIP. Ceci a été rendu possible grâce à une librairie développée en langage C. Un processeur embarqué a été utilisé pour cette fin. Un autre point fort de la méthode d'ajustement des délais proposée dans cette thèse est qu'elle ne requiert pas la reconfiguration dynamique contrairement à [23][24], où elle a été utilisée pour atteindre les performances rapportées. À noter que la reconfiguration dynamique, bien que supportée par l'ensemble des FPGA modernes, reste compliquée à mettre en place.

## 8.2 Le MMRO-TDC

Comme les FPGA apportent de la flexibilité, de la rapidité de développement et des coûts moins élevés comparés aux ASIC, d'innombrables TDC ont été implémentés sur des FPGA. La plupart

se basent sur des lignes à délais (TDL-TDC) [15][20] constituées de chaînes de retenues qui permettent d'atteindre des résolutions inférieures à une dizaine de picosecondes, mais souffrent d'une non-linéarité élevée due au routage non uniforme causé principalement par les divisions trop larges (*ultra wide bins*) situées entre deux CLB consécutifs. Par conséquent, plusieurs travaux se sont tournés vers l'architecture à base de boucle (RO-TDC ou oscillateur en anneau). Notre solution, le *Multi-Measurements Ring-Oscillator* (MMRO) TDC, est inspirée de la topologie bouclée et supporte de multiple signaux en entrée. Une telle solution a été rendue possible moyennant la méthode d'ajustement des délais. En effet, sans le contrôle fin des délais, le TDC proposé aurait de piètres performances temporelles (linéarité, exactitude). Le MMRO-TDC est une application parmi d'autres qui démontre que ces algorithmes de routage fin peuvent être avantageusement appliqués à un circuit reprogrammable et qu'ils peuvent être utilisés dans toute autre application où les délais des traces dans un FPGA ont besoin d'être ajustés.

Le MMRO-TDC se distingue par rapport aux TDC rapportés par quatre points principaux :

1. Faible consommation en ressources : dans le chapitre 4, il a été démontré par une étude théorique et comparative, que le MMRO-TDC consomme nettement moins de ressources FPGA que le TDC bouclé classique. Ce gain en ressources découle du fait qu'un seul oscillateur est partagé entre plusieurs éléments de capture de la mesure fine. Il faut savoir que dans le MMRO-TDC, tel qu'implémenté dans cette thèse, l'élément de capture de la mesure fine n'occupe qu'une seule LUT mémoire car l'oscillateur est fait de 6 DU. Cependant, d'une manière générale, l'élément de capture est fait de $2^{N_{DU}-6}$ LUT avec $N_{DU}$ étant le nombre de DU constituant l'oscillateur en anneau partagé [28]. À noter que dans le chapitre 7, une version du MMRO-TDC à multiples canaux a été implémentée et ses ressources ont été comparées à d'autres TDC à multiples canaux. Le Tableau 7.4 démontre que le MMRO-TDC est plus compact que ce qui est proposé dans la littérature récente.
2. Pas de contrainte de placement : grâce à l'algorithme de balancement (discuté à la section 8.1), l'oscillateur partagé ainsi que les LUT mémoire peuvent avoir des emplacements différents dans le FPGA. Contrairement au TDL-TDC où les éléments de capture (FF) doivent être placés le plus proche possible des éléments à délais et aux TDC bouclés implémentés sur FPGA (Figure 2.7) où les deux oscillateurs doivent être placés à distance égale du circuit de détection de coïncidence. De plus, grâce aux algorithmes de contrôle de délais, il n'y pas de contraintes sur les points d'entrée des signaux (les signaux en entrée peuvent provenir de n'importe quelle

broche ou nœud électrique dans le FPGA). Les algorithmes de contrôle fins des délais, présentés dans les chapitres 5 et 6 permettent d'équilibrer tout décalage dans le routage.

3. Multiples signaux en entrée : l'absence de contrainte de placement du MMRO-TDC permet non seulement de mesurer l'intervalle de temps entre deux signaux en entrée, mais aussi des intervalles de temps relatifs entre plusieurs signaux. Comme il a été démontré dans le chapitre 4, le moment d'arrivée de chaque signal en entrée est capté par une LUT mémoire avec un seul oscillateur en anneau partagé qui définit les adresses d'écriture des LUT. Donc en théorie, ce TDC supporte autant de signaux en entrées qu'il y a de LUT mémoires disponibles dans le FPGA. Néanmoins, le routage se complexifie en augmentant le nombre de signaux à balancer. En effet, il a été démontré dans les chapitres 6 et 7 que les traces entre l'oscillateur partagé et les LUT mémoires ainsi que les traces entre les signaux en entrée et les LUT mémoires doivent être balancés pour améliorer la linéarité et l'exactitude des mesures. Par conséquent, le nombre de traces à balancer augmente avec le nombre de signaux en entrée. Dans ce cas, les algorithmes de routage utilisés par la méthode d'ajustement des délais risquent de causer une congestion ou de ne pas trouver de ressources disponibles surtout si le FPGA contient des circuits autres que le TDC. Le nombre de signaux en entrée supportable par le TDC proposé dépend aussi de la taille du FPGA.
4- Exactitude sous les 100 ps : en explorant les travaux publiés dans la littérature sur les TDC implémentés sur FPGA, il a été noté que l'exactitude n'est généralement pas rapportée. Même quand le terme *accuracy* est évoqué, il réfère à la précision et non à l'exactitude. Ces deux termes sont souvent interchangés malgré qu'ils ont des définitions différentes. L'exactitude, telle que définie dans le chapitre 2, est une métrique qui montre à quel point la mesure donnée par le TDC s'approche de l'intervalle de temps réel appliqué à l'entrée du TDC. Tandis que la précision est une métrique qui montre à quel point la mesure donnée par le TDC est reproductible. L'exactitude est altérée par la différence de routage depuis l'entrée des signaux jusqu'au TDC. En effet, pour minimiser la différence entre la mesure donnée par le TDC et l'intervalle de temps réel, il faut que le délai de propagation du signal de départ vers l'entrée du TDC soit égal à celui du signal d'arrêt vers l'entrée du TDC. Or, ceci n'est pas faisable vu qu'il n'y aucun moyen de contrôler finement les délais des traces dans le FPGA.

Avec la méthode de balancement des délais proposée dans le cadre de cette thèse, l'exactitude du TDC a été contrôlée et améliorée pour réduire le RMSD de 327.8 ps à 92.9 ps, tel que rapporté

dans le chapitre 7. Cette méthode a été utilisée pour équilibrer finement les délais des traces des signaux vers l'entrée du TDC.

Néanmoins le TDC proposé offre une résolution inférieure à celles observées dans la littérature récente. Ceci est dû au fait que dans ce travail, l'oscillateur en anneau est constitué de LUT contrairement à ce qui est couramment publié où la ligne à délai est à base de chaînes de retenue. Une meilleure résolution pourrait être obtenue en implémentant l'oscillateur du MMRO-TDC avec une chaîne de retenue, par contre il aurait fallu augmenter le nombre d'étages dans l'oscillateur afin que la fréquence d'oscillation soit mesurable. Augmenter le nombre d'étages signifie augmenter le nombre de traces à balancer : un oscillateur à $m$ étages et $k$ signaux en entrée nécessite un balancement de $m \times k$ traces par canal. À noter que les travaux qui ont rapporté des résolutions au-dessous de 10 ps se basent sur la mesure différentielle (vernier) [79][70], qui est une toute autre architecture coûteuse en ressources.

Une autre limitation de ce TDC est qu'il ne peut être implémenté que dans des FPGA de Xilinx. En effet, au meilleur de nos connaissances, les outils de développement des autres fournisseurs (comme Altera ou Microsemi) n'offrent pas des commandes en TCL permettant d'accéder à la structure interne d'une matrice de commutation.

# CHAPITRE 9 CONCLUSION ET RECOMMANDATIONS

Les travaux de recherche présentés dans cette thèse avaient comme principaux objectifs de (O1) proposer et concevoir des convertisseurs temps-numérique compacts et distribués qui permettent la mesure précise de délais in-situ, (O2) élaborer une méthodologie d'implémentation de tels circuits de support dans un circuit programmable et (O3) mettre en place des stratégies d'amélioration des performances temporelles. Ces objectifs ont été rencontrés à travers les quatre articles inclus dans le corps de ce document :

- Chapitre 4 : objectifs O1 et O2;
- Chapitre 5 : objectif O3;
- Chapitre 6 : objectif O2 et O3;
- Chapitre 7 : objectifs O1 et O3.

Le chapitre 8 a discuté les éléments originaux présentés au sein de ce travail de recherche, de leur impact sur la mesure du temps dans les FPGA et de leurs limitations. Dans ce chapitre, nous concluons la thèse et présentons des pistes pour les travaux futurs.

## 9.1 Remarques finales

Les convertisseurs temps-numérique sont principalement utilisés dans l'imagerie médicale et dans les instruments de mesure de délais. Récemment, on les retrouve dans des applications automobiles comme dans les systèmes LIDAR. De plus en plus d'applications modernes basées sur des appareils portables utilisent aussi les TDC. Alimentés par une batterie, ils requièrent une faible consommation d'énergie et par conséquent une faible complexité. De plus, une pratique courante afin d'obtenir des performances temporelles compétitives consiste à utiliser plusieurs canaux de TDC, ce qui augmente considérablement la consommation des ressources, déjà une préoccupation majeure dans les dispositifs miniaturisés. On s'attend à ce que la priorité dans les recherches futures passe de « obtenir des résolutions plus élevées à tout prix » vers « obtenir une résolution et une précision suffisantes au prix d'une consommation de ressource modérée ou une consommation en énergie réduite ».

Dans ce contexte, les contributions scientifiques apportées par cette thèse se résument comme suit :

1- Proposition d'une méthodologie et génération d'outils pour ajuster finement les délais dans un circuit programmable;
2- Conception d'une nouvelle architecture de TDC compacte et distribuée mise en place grâce aux outils d'ajustement fins des délais élaborés dans cette thèse;
3- Élaboration d'une méthodologie d'implémentation d'un tel TDC sur des circuits programmables (FPGA).

## 9.2 Recommandations et améliorations

Les résultats rapportés dans ce travail ouvrent plusieurs pistes de recherche :

- Rendre le MMRO-TDC paramétrable et reconfigurable. En effet, chaque architecture de TDC est liée à une application spécifique et porter une architecture d'une application à une autre n'est pas chose facile. Par conséquent, trouver un moyen d'automatiser l'utilisation du MMRO-TDC le rendrait plus facile à utiliser dans une plus large gamme d'applications;
- Prendre en compte l'intégrité des signaux qui peut être altérée à cause de la boucle de rétroaction dans les TDC à base d'oscillateur en général et dans le MMRO-TDC en particulier;
- Adapter la méthode d'ajustement des délais proposée à d'autres architectures de TDC notamment pour améliorer leurs performances ou résoudre le problème des bulles [37] (*bubbles*);
- Caractériser et limiter les effets de la température sur les performances du TDC et sur l'ajustement fin des délais;
- Caractériser l'impact de la longueur des traces (distance entre la source et la destination) sur les performances de l'outil d'ajustement fin des délais et par conséquent sur le TDC. En effet, plus la trace est longue plus son délai est sensible aux perturbations telles que l'interférence entre les signaux (*crosstalk*), le bruit provenant des alimentations ainsi que la température et les procédés de fabrication ;
- Étudier la consommation d'énergie du MMRO-TDC proposé;
- Appliquer les algorithmes d'ajustement des délais sur toutes les traces liant l'oscillateur aux LUT mémoires dans la topologie à multiples canaux présentée dans le chapitre 7. En effet, il a été démontré dans le chapitre 6 que le fait d'équilibrer les traces entre le RO et une LUT mémoire améliore la linéarité, alors qu'au chapitre 7, il a été démontré qu'équilibrer les

traces liant l'entrée des signaux aux LUT mémoires améliore l'exactitude. Un travail futur intéressant serait d'équilibrer les traces entre le RO et toutes les LUT dans une architecture à canaux multiples (chapitre 7) afin d'avoir à la fois une meilleure linéarité et une meilleure exactitude;

- Étudier l'impact du nombre de signaux en entrée sur les performances temporelles;
- Améliorer les algorithmes et les outils d'ajustement des délais pour converger plus rapidement vers un balancement optimal. Par exemple, afin de minimiser le nombre d'itérations nécessaires pour atteindre le délai cible, une étude exhaustive faite au préalable serait utile afin d'extraire l'augmentation de délai engendrée par chaque branche possible et choisir les branches à créer permettant un temps de balancement optimal.

Ce chapitre clôt la thèse en rappelant ses principales contributions qui consistent principalement en la mise en place d'une nouvelle architecture de convertisseur temps-numérique compacte et supportant plusieurs points de mesure. Ce TDC a été rendu possible grâce à l'élaboration d'une nouvelle technique d'ajustement fin des délais dans un FPGA. Ce chapitre a proposé également une multitude de pistes de travaux futurs qui découlent directement de cette recherche, comme par exemple : étudier l'effet de la variation des procédés de fabrication et de la température ainsi que l'interférence des signaux sur les outils développés ou encore, élaborer des stratégies pour converger rapidement vers un délai cible à la picoseconde près. Ces travaux futurs et recommandations permettront de rendre l'outil de contrôle de délai ainsi que le TDC qui en découle plus robustes et aptes à être utilisés dans une multitude d'applications de pointe.

# LISTE DES RÉFÉRENCES


[1] H. T. Van Dam, G. Borghi, S. Seifert, and D. R. Schaart, "Sub-200 ps CRT in monolithic scintillator PET detectors using digital SiPM arrays and maximum likelihood interaction time estimation," *Physics in Medicine and Biology*, vol. 58, no. 10, pp. 3243–3257, 2013.

[2] D. Tyndall *et al.*, "A high-throughput time-resolved mini-silicon photomultiplier with embedded fluorescence lifetime estimation in 0.13 μm CMOS," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 6, pp. 562–570, 2012.

[3] W. Wu, R. B. Staszewski, and J. R. Long, "A 56.4-to-63.4 GHz multi-rate all-digital fractional-N PLL for FMCW radar applications in 65 nm CMOS," *IEEE Journal of solid-state circuits*, vol. 49, no. 5, pp. 1081–1096, 2014.

[4] J. H. Lau, "3D Integration," in *Fan-Out Wafer-Level Packaging*, Springer, Singapore, 2018, pp. 231–268.

[5] X. Zhang *et al.*, "Heterogeneous 2.5D integration on through silicon interposer," *Applied Physics Reviews*, vol. 2, no. 2. AIP Publishing LLC, p. 21308, 2015.

[6] R. Mahajan *et al.*, "Embedded Multi-die Interconnect Bridge (EMIB) A Localized, High Density, High Bandwidth Packaging Interconnect," *Advances in Embedded and Fan-Out Wafer-Level Packaging Technologies*, pp. 487–499, 2019.

[7] Y. Yao, Z. Wang, H. Lu, L. Chen, and G. Jin, "Design of time interval generator based on hybrid counting method," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 832, pp. 103–107, 2016.

[8] R. Nutt, "Digital time intervalometer," *Review of Scientific Instruments*, vol. 39, no. 9, pp. 1342–1345, 1968.

[9] B. Markovic, S. Tisa, F. A. Villa, A. Tosi, and F. Zappa, "A high-linearity, 17 ps precision time-to-digital converter based on a single-stage vernier delay loop fine interpolation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 3, pp. 557–569, 2013.

[10] Z. Cheng, X. Zheng, M. J. Deen, and H. Peng, "Recent developments and design challenges

of high-performance ring oscillator CMOS time-to-digital converters," *IEEE Transactions on Electron Devices*, vol. 63, no. 1, pp. 235–251, 2016.

[11] S. Henzler, "Time-to-digital converter basics," *Springer Series in Advanced Microelectronics*, vol. 29, Springer, pp. 5–18, 2010.

[12] R. Machado, J. Cabral, and F. S. Alves, "Recent developments and challenges in FPGA-based time-to-digital converters," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 11, pp. 4205–4221, 2019.

[13] C. Liu and Y. Wang, "A 128-channel, 710 M samples/second, and less than 10 ps RMS resolution time-to-digital converter implemented in a kintex-7 FPGA," *IEEE Transactions on Nuclear Science*, vol. 62, no. 3, pp. 773–783, 2015.

[14] D. Chaberski, "Time-to-digital-converter based on multiple-tapped-delay-line," *Measurement*, vol. 89, pp. 87–96, 2016.

[15] Q. Shen *et al.*, "A 1.7 ps equivalent bin size and 4.2 ps RMS FPGA TDC based on multichain measurements averaging method," *IEEE Transactions on Nuclear Science*, vol. 62, no. 3, pp. 947–954, 2015.

[16] J. Wu, "Uneven bin width digitization and a timing calibration method using cascaded PLL," *19th IEEE-NPSS Real Time Conference*, pp. 1–4, 2014.

[17] J. Wu, "Several Key Issues on Implementing Delay Line Based TDCs Using FPGAs," *IEEE Transactions on Nuclear Science*, vol. 57, no. 3, pp. 1543–1548, 2010.

[18] H. Menninga, C. Favi, M. W. Fishburn, and E. Charbon, "A multi-channel, 10 ps resolution, FPGA-based TDC with 300MS/s throughput for open-source PET applications," *IEEE Nuclear Science Symposium Conference Record*, pp. 1515–1522, 2011.

[19] M. W. Fishburn, L. H. Menninga, C. Favi, and E. Charbon, "A 19.6 ps, FPGA-based TDC with multiple channels for open source applications," *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 2203–2208, 2013.

[20] J. Y. Won, S. Il Kwon, H. S. Yoon, G. B. Ko, J. W. Son, and J. S. Lee, "Dual-Phase Tapped-Delay-Line Time-to-Digital Converter with On-the-Fly Calibration Implemented in 40 nm FPGA," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 1, pp. 231–

242, 2016.

[21] K. C. Choi, S. W. Lee, B. C. Lee, and W. Y. Choi, “A time-to-digital converter based on a multiphase reference clock and a binary counter with a novel sampling error corrector,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 3, pp. 143–147, 2012.

[22] K. Cui, Z. Ren, X. Li, Z. Liu, and R. Zhu, “A high-linearity, ring-oscillator-based, Vernier time-to-digital converter utilizing carry chains in FPGAs,” *IEEE Transactions on Nuclear Science*, vol. 64, no. 1, pp. 697–704, 2016.

[23] E. Bergeron, M. Feeley, M. A. Daigneault, and J. P. David, “Using dynamic reconfiguration to implement high-resolution programmable Delays on an FPGA,” *IEEE North-East Workshop on Circuits and Systems and TAISA Conference, NEWCAS-TAISA*, pp. 265–268, 2008.

[24] M. A. Daigneault and J. P. David, “A high-resolution time-to-digital converter on FPGA using dynamic reconfiguration,” *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 6, pp. 2070–2079, 2011.

[25] M. Majzoobi, A. Kharaya, F. Koushanfar, and S. Devadas, “Automated Design , Implementation , and Evaluation of Arbiter-based PUF on FPGA using Programmable Delay Lines,” 2014. [Online]. Available: http://eprint.iacr.org.

[26] N. N. Anandakumar, S. K. Sanadhya, and M. S. Hashmi, “FPGA-based true random number generation using programmable delays in oscillator-rings,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 3, pp. 570–574, 2019.

[27] D. Chaberski, R. Frankowski, M. Zieliński, and Ł. Zaworski, “Multiple-tapped-delay-line hardware-linearisation technique based on wire load regulation,” *Measurement*, vol. 92, pp. 103–113, 2016.

[28] S. Berrima, Y. Blaquiere, and Y. Savaria, “A multi-measurements RO-TDC implemented in a Xilinx field programmable gate array,” *Proceedings - IEEE International Symposium on Circuits and Systems*, pp. 1–4, 2017.

[29] M. Gersbach *et al.*, “A time-resolved, low-noise single-photon image sensor fabricated in deep-submicron CMOS technology,” *IEEE Journal of Solid-State Circuits*, vol. 47, no. 6,

pp. 1394–1407, 2012.

[30] R. B. Staszewski, D. Leipold, C.-M. Hung, and P. T. Balsara, “TDC-based frequency synthesizer for wireless applications,” *IEE Radio Frequency Integrated Circuits (RFIC) Systems. Digest of Papers*, pp. 215–218, 2004.

[31] C. Ugur, E. Bayer, N. Kurz, and M. Traxler, “A 16 channel high resolution (< 11 ps RMS) Time-to-Digital Converter in a Field Programmable Gate Array,” *Journal of Instrumentation*, vol. 7, no. 02, p. C02004, 2012.

[32] R. Narasimman, A. Prabhakar, and N. Chandrachoodan, “Implementation of a 30 ps resolution time to digital converter in FPGA,” *International Conference on Electronic Design, Computer Networks and Automated Verification, (EDCAV)*, pp. 12–17, 2015.

[33] J. Zhang and D. Zhou, “A new delay line loops shrinking time-to-digital converter in low-cost FPGA,” *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 771, pp. 10–16, 2015.

[34] Y. Wang, P. Kuang, and C. Liu, “A 256-channel multi-phase clock sampling-based time-to-digital converter implemented in a Kintex-7 FPGA,” *IEEE International Instrumentation and Measurement Technology Conference Proceedings*, pp. 1–5, 2016.

[35] J. Torres *et al.*, “Time-to-digital converter based on FPGA with multiple channel capability,” *IEEE Transactions on Nuclear Science*, vol. 61, no. 1, pp. 107–114, 2013.

[36] M. Z. Straayer and M. H. Perrott, “A multi-path gated ring oscillator TDC with first-order noise shaping,” *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, pp. 1089–1098, 2009.

[37] S. Ito *et al.*, “Stochastic TDC architecture with self-calibration,” *IEEE Asia Pacific Conference on Circuits and Systems*, pp. 1027–1030, 2010.

[38] Xilinx, “7 Series FPGAs Configurable Logic Block (UG474),” 2016.

[39] S. Berrima, Y. Blaquiere, and Y. Savaria, “Sub-ps resolution programmable delays implemented in a Xilinx FPGA,” *Midwest Symposium on Circuits and Systems*, pp. 918–921, 2017.

[40] T. Polzer, F. Huemer, and A. Steininger, “A programmable delay line for metastability

characterization in FPGAs," *Austrochip Workshop on Microelectronics (Austrochip)*, pp. 51–56, 2016.

[41] P. Chen, Y.-Y. Y. Hsiao, Y.-S. S. Chung, W. X. X. Tsai, and J.-M. M. Lin, "A 2.5-ps bin size and 6.7-ps resolution FPGA time-to-digital converter based on delay wrapping and averaging," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 1, pp. 114–124, 2017.

[42] M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA-based true random number generation using circuit metastability with adaptive feedback control," *Cryptographic Hardware and Embedded Systems – CHES*, vol. 6917 LNCS, pp. 17–32, 2011.

[43] Xilinx, "Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics (DS925)," 2017.

[44] B. Taj, "Single Event Upset error detection on routing tracks of Xilinx FPGAs." 2013, [Online]. Available: http://hdl.handle.net/11375/15283.

[45] C. Beckhoff, D. Koch, and J. Torresen, "The xilinx design language (xdl): Tutorial and use cases," *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, pp. 1–8, 2011.

[46] S. Berrima, Y. Blaquière, and Y. Savaria, "Fine resolution delay tuning method to improve the linearity of an unbalanced time-to-digital converter on a Xilinx FPGA," *IET Circuits, Devices and Systems*, vol. 14, no. 8, pp. 1243–1252, 2020.

[47] J. Wu, Y. Zhang, R. Zhao, K. Zhang, L. Zheng, and W. Sun, "Low-jitter DLL applied for two-segment TDC," *IET Circuits, Devices & Systems*, vol. 12, no. 1, pp. 17–24, 2018.

[48] F. Yuan, "CMOS time-to-digital converters for mixed-mode signal processing," *The Journal of Engineering*, vol. 2014, no. 4, pp. 140–154, 2014.

[49] Y. Wang, Q. Cao, and C. Liu, "A multi-chain merged tapped delay line for high precision time-to-digital converters in FPGAs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 1, pp. 96–100, 2018.

[50] H. Wang, F. F. Dai, and H. Wang, "A Reconfigurable Vernier Time-to-Digital Converter with 2-D Spiral Comparator Array and Second-Order Δ Σ Linearization," *IEEE Journal of*

*Solid-State Circuits*, vol. 53, no. 3, pp. 738–749, 2018.

[51] K. Kim, Y. H. Kim, W. Yu, and S. Cho, "A 7 bit, 3.75 ps resolution two-step time-to-digital converter in 65 nm CMOS using pulse-train time amplifier," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 4, pp. 1009–1017, 2013.

[52] M. T. Tsai, S. Y. Huang, K. H. Tsai, and W. T. Cheng, "Monitoring the delay of long interconnects via distributed TDC," *IEEE International Test Conference (ITC)*, pp. 1–9, 2015.

[53] B. K. Swann *et al.*, "A 100-ps time-resolution CMOS time-to-digital converter for positron emission tomography imaging applications," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 11, pp. 1839–1852, 2004.

[54] H. Chen, Y. Zhang, and D. D.-U. Li, "A Low Nonlinearity, Missing-Code Free Time-to-Digital Converter Based on 28-nm FPGAs With Embedded Bin-Width Calibrations," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 7, pp. 1912–1921, 2017.

[55] Xilinx, "Vivado design suite tutorial, partial reconfiguration (UG947)," 2018.

[56] Xilinx, "7 Series DSP48E1 Slice (UG479)," 2018.

[57] Xilinx, "7 Series FPGAs Memory Resources (UG473)," 2019.

[58] Xilinx, "Vivado Design Suite Tcl Command Reference (UG835)," 2018.

[59] M. Ruffoni and A. Bogliolo, "Direct measures of path delays on commercial FPGA chips," *Proceedings - 6th IEEE Workshop on Signal Propagation on Interconnects, SPI*, pp. 157–159, 2002.

[60] N. D. Guiping Cao, Haojie Xia, "An 18-ps TDC using timing adjustment and bin realignment methods in a Cyclone-IV FPGA," *Review of Scientific Instruments*, vol. 89, no. 5, p. 054707, 2018.

[61] Y. H. Chen, "Run-time calibration scheme for the implementation of a robust field-programmable gate array–based time-to-digital converter," vol. 47, no. 1, pp. 19–31, 2019.

[62] H. Chen and D. D.-U. Li, "Multichannel, Low Nonlinearity Time-to-Digital Converters Based on 20 and 28 nm FPGAs," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 3265–3274, 2019.

[63] T. U. Y. Sano, Y. Horii, M. Ikeno, O. Sasaki, M. Tomoto *et al.*, "Subnanosecond time-to-digital converter implemented in a kintex-7 FPGA," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 874, pp. 50–56, 2017.

[64] Q. Shen *et al.*, "A multi-chain measurements averaging TDC implemented in a 40 nm FPGA," *19th IEEE-NPSS Real Time Conference*, pp. 1–3, 2014.

[65] Xilinx, "UltraScale Architecture Clocking Resources (UG572)," 2018.

[66] Y. Wang and C. Liu, "A 4.2 ps Time-Interval RMS Resolution Time-to-Digital Converter Using a Bin Decimation Method in an UltraScale FPGA," *IEEE Transactions on Nuclear Science*, vol. 63, no. 5, pp. 2632–2638, 2016.

[67] C. Ugur, S. Linev, J. Michel, T. Schweitzer, and M. Traxler, "A novel approach for pulse width measurements with a high precision (8 ps RMS) TDC in an FPGA," *Journal of Instrumentation*, vol. 11, no. 1, p. C01046, 2016.

[68] R. Frankowski, M. Gurski, and P. Płóciennik, "Optical methods of the delay cells characteristics measurements and their applications," *Optical and Quantum Electronics*, vol. 48, no. 3, pp. 1–19, 2016.

[69] X. Kang *et al.*, "A simple smart time-to-digital converter based on vernier method for a high resolution LYSO MicroPET," *IEEE Nuclear Science Symposium Conference Record*, pp. 2892–2896, 2007.

[70] K. Cui, X. Li, Z. Liu, and R. Zhu, "Towards implementing multi-channels, ring-oscillator-based, vernier time-to-digital converter in FPGAS: Key design points and construction method," *arXiv*, vol. 1, no. 5. pp. 391–399, 2017.

[71] K. Cui and X. Li, "A high-linearity vernier time-to-digital converter on FPGAs with improved resolution using bidirectional-operating vernier delay lines," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 8, pp. 5941–5949, 2020.

[72] M. Maamoun *et al.*, "A 3ps resolution time-to-digital converter in low-cost FPGA for laser rangefinder," *Proceedings of the World Congress on Engineering*, pp. 259–263, 2017.

[73] K. Cui, Z. Liu, R. Zhu, and X. Li, "FPGA-based high-performance time-to-digital converters

by utilizing multi-channels looped carry chains," *International Conference on Field-Programmable Technology, ICFPT*, pp. 223–226, 2017.

[74] A. Balla *et al.*, "The characterization and application of a low resource FPGA-based time to digital converter," *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 739, pp. 75–82, 2014.

[75] M. Büchele, H. Fischer, F. Herrmann, and C. Schaffer, "The ARAGORN front-end - An FPGA based implementation of a Time-to-Digital Converter," *Journal of Instrumentation*, vol. 12, no. 02, p. C02006, 2017.

[76] J. Y. Won and J. S. Lee, "Time-to-digital converter using a tuned-delay line evaluated in 28-, 40-, and 45-nm FPGAs," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 7, pp. 1678–1689, 2016.

[77] X. Qin *et al.*, "A high resolution time-to-digital-convertor based on a carry-chain and DSP48E1 adders in a 28-nm field-programmable-gate-array," *Review of Scientific Instruments*, vol. 91, no. 2, p. 24708, 2020.

[78] M.-A. Daigneault, "Utilisation de la reconfiguration dynamique des FPGA pour le contrôle précis et exact des délais dans les convertisseurs temps à numérique," École Polytechnique de Montréal, 2009.

[79] M. A. Daigneault and J. P. David, "Towards 5 ps resolution TDC on a dynamically reconfigurable FPGA," in *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, p. 283, 2010.

[80] Arpin, Louis, *et al*. "A sub-nanosecond time interval detection system using FPGA embedded I/O resources." in *IEEE Transactions on Nuclear Science*, vol. 57, issue 2, p. 519-524, 2010.