Odatix : Une solution libre pour l'automatisation de l'implémentation matérielle FPGA et ASIC

Jonathan SAUSSEREAU Christophe JEGO Camille LEROUX Jean-Baptiste BEGUERET
Universite de Bordeaux, Bordeaux INP, Laboratoire IMS, UMR CNRS 5218, France

Résumé – Dans la conception matérielle numérique, concilier performances, consommation énergétique et utilisation des ressources représente un défi de taille. La nécessité d'accroître les performances tout en limitant les coûts et la consommation, rend le compromis complexe. Pour pallier ces difficultés, les outils d'automatisation de conception se révèlent essentiels.
Odatix est une solution open-source destinée à automatiser l'implémentation et la validation d'architectures numériques paramétrables. Il permet de mettre en oeuvre en parallèle, pour chaque configuration architecturale et chaque cible technologique, la synthèse, le placement-routage, la simulation et la validation pour divers outils FPGA et ASIC. L'extraction automatique des résultats et leur mise en forme dans une interface d'exploration interactive permettent de choisir sans effort les solutions architecturales et technologiques les plus adaptées à un cadre applicatif donné.

Abstract – In digital hardware design, reconciling performance, power consumption, and resource utilization is a formidable challenge. The need to boost performance while limiting costs and energy usage makes achieving the right balance complex. To overcome these difficulties, design automation tools are essential.
Odatix is an open-source solution designed to automate the implementation and validation of parameterizable digital architectures. It enables the parallel execution—across each architectural configuration and technological target—of synthesis, place-and-route, simulation, and validation for various FPGA and ASIC tools. Automatic extraction of results and their presentation via an interactive exploration interface allow for the effortless selection of the most suitable architectural and technological solutions for a given application scenario.

1 Introduction

Pour implémenter une fonctionnalité numérique, le concepteur peut opter pour une solution logicielle ou une solution matérielle. Dans divers contextes, la solution matérielle peut offrir de nombreux avantages par rapport à une approche logicielle[1]. Cependant, elle implique divers défis et nécessite l'exécution d'une série d'étapes systématiques. Heureusement, ces étapes peuvent être automatisées. L'approche matérielle peut viser aussi bien un circuit reconfigurable comme un FPGA ou un circuit intégré (ASIC). Dans ces deux cas, la démarche démarre par la description RTL en HDL, suivie de la synthèse logique, puis du placement et routage. Pour les FPGA, le bitstream est généré, tandis que la chaîne ASIC inclut des étapes complémentaires comme le floorplanning et le DRC.

Les architectures paramétrables sont particulièrement intéressantes dans un contexte où les exigences en performances, consommation et utilisation des ressources se durcissent. L'évolution des HDL classiques (p.ex. SystemVerilog[2]) et l'apparition d'outils comme Chisel[3] permettent aujourd'hui d'adapter facilement une architecture aux besoins spécifiques d'une application, notamment pour les processeurs. L'émergence de jeux d'instructions open-source comme RISC-V a en outre favorisé le développement de processeurs flexibles, comme le Rocket Chip[4], BRISC-V[5], HL5[6] ou Aste-RISC[7]. Même des cœurs simples, tels que le PicoRV32[8], offrent certaines formes de configurabilité. Ceci illustrent la tendance vers plus de flexibilité.

Le défi réside dans l'exploration efficace de ces nombreuses configurations pour identifier la solution optimale. Si des outils d'automatisation existent déjà, leur utilisation peut se révéler complexe pour des architectures paramétrables. Des outils comme OpenTuner[9], Heracles[10] ou Aladdin[11] aident à réduire l'espace des paramètres, mais il manque une solution globale intégrant automatisation, simulation et validation sur FPGA et ASIC.

C'est dans ce contexte qu'Odatix a été conçu. Il propose d'automatiser la synthèse, le placement-routage et la simulation de nombreuses configurations pour cibles technologiques et divers outils FPGA et ASIC (Vivado, OpenLane, Design Compiler). Cette approche permet aux chercheurs et ingénieurs d'explorer rapidement différentes configurations architecturales et d'optimiser la chaîne de conception selon leurs contraintes spécifiques. Par exemple, Odatix peut être utilisé de pair avec des plateformes d'exploration telles que LiteX[12] ou Boom Explorer[13].

2 Fonctionnalités principales

Odatix est une boîte à outils conçue pour simplifier la synthèse, le placement et le routage, la simulation et la validation d'architectures numériques configurables. Il peut être utilisé en parallèle avec divers outils FPGA et ASIC, listés dans le tableau ci-dessous. Il est à noter que le support d'outils supplémentaires peut être étendu aisément.

Outil EDA	Type de licence
AMD Vivado	Commerciale
Synopsys Design Compiler	Commerciale
OpenLane[14]	Libre et Open Source

Table 1. Outils supportés pour la synthèse logique

Odatix propose les fonctionnalités suivantes :

- **Exploration architecturale**
  Définition et génération de multiples configurations architecturales, quel que soit le langage HDL utilisé (VHDL, Verilog, SystemVerilog et même Chisel ou HLS).
- **Synthèses**
  Synthèses logiques automatisées sur des outils FPGA et/ou ASIC pour chaque configuration et cible technologique. Il est possible de faire des synthèses à des fréquences personnalisées ou de déterminer automatiquement la fréquence maximale de fonctionnement (Fmax).
- **Simulations**
  Simulations automatisées pour chaque configuration. Utile pour les validations et le benchmarking.
- **Suivi des Tâches**
  Suivi en temps réel la progression des tâches qui s'exécutent en parallèles. Il est possible de démarrer, mettre en pause, reprendre ou annuler les tâches à tout moment.
- **Exploration Interactive des Résultats**
  Export automatique et analyse visuelle des résultats grâce à une interface interactive. Support intégré pour les graphiques linéaires, en colonnes, radar, nuage de points 2D et 3D. Export des graphiques en formats vectoriel (SVG) ou matriciel (PNG, JPEG, WEBP).
- **Démarrage Rapide**
  Installation facile via Pypi avec `pip`. Exemples intégrés et de tutoriels pas à pas disponibles en ligne.

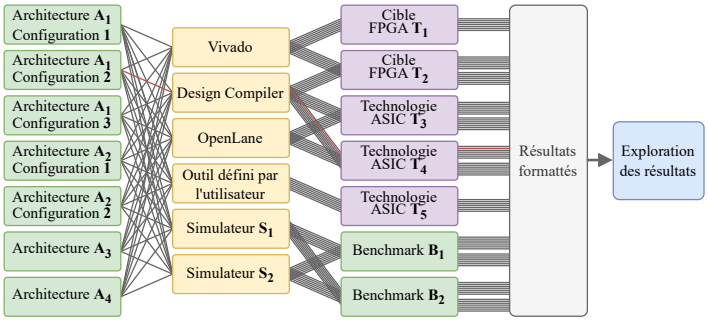


FIGURE 1 : Schéma fonctionnel d'Odatix

La Figure 1 illustre visuellement les capacités d'Odatix. À gauche, différentes configurations architecturales sont utilisées en entrée pour divers outils de synthèse ou de simulation. Chaque outil de synthèse peut avoir plusieurs cibles, incluant à la fois des technologies ASIC et des circuits FPGA. Tous les résultats sont automatiquement formatés dans une structure commune, qui sert d'entrée pour un outil d'exploration des résultats.

Le nombre de combinaisons possibles entre configurations architecturales, outils EDA et cibles technologiques est immense. En général, un concepteur est familier avec un seul outil et implémente manuellement un ensemble limité d'architectures sur quelques cibles technologiques, devant ensuite formater manuellement les résultats, ce qui représente un processus fastidieux. La combinaison correspondant à l'implémentation d'une configuration architecturale sur une cible technologique spécifique, en utilisant un outil EDA particulier, est mise en évidence en rouge. Cela souligne l'énorme effort requis pour effectuer l'ensemble du travail manuellement, alors que le nombre d'architectures et de cibles technologiques illustré ici reste raisonnable. Odatix automatise l'ensemble de ce processus. Il permet aux utilisateurs de trouver la meilleure combinaison de configuration architecturale, d'outil EDA et de cible technologique pour une application donnée. De plus, grâce à ses capacités de traitement des données, Odatix facilite la connexion entre validation, benchmarking et implémentation. En comparaison, l'utilisation de Dovado, bien qu'offrant des capacités uniques en termes d'exploration de conception, ne fournit des résultats que pour Vivado sur FPGA et ne propose pas les mêmes fonctionnalités de comparaison de résultats. De même, l'utilisation directe des capacités d'exploration architecturale d'OpenLane ne permet pas d'obtenir des résultats d'implémentation avec d'autres outils ou cibles FPGA, et il n'existe pas d'outil d'exploration permettant de comparer facilement les résultats obtenus. Les deux outils manquent d'interaction avec la simulation.

3 Architecture logicielle

L'architecture d'Odatix est présentée par la Figure 2. Les différents composants sont décrits ci-dessous.

Dans cette figure, le vert représente les éléments fournis par l'utilisateur, tels que les fichiers sources et la configuration. Le bleu représente les outils constituant Odatix. Le jaune correspond aux outils externes, incluant les outils de synthèse/placement et routage, les simulateurs, les générateurs RTL, les convertisseurs et autres.

3.1 Configuration utilisateur

Pour utiliser Odatix, l'utilisateur commence par définir les détails de l'architecture matérielle (entité de niveau supérieur, signaux d'horloge et de réinitialisation) dans un fichier de configuration. Ce fichier précise également les délimiteurs dans le code où se trouvent les paramètres à substituer. L'outil remplace alors automatiquement le contenu entre ces balises par les valeurs fournies, et permet, au besoin, de fixer des limites de fréquence pour chaque cible technologique.

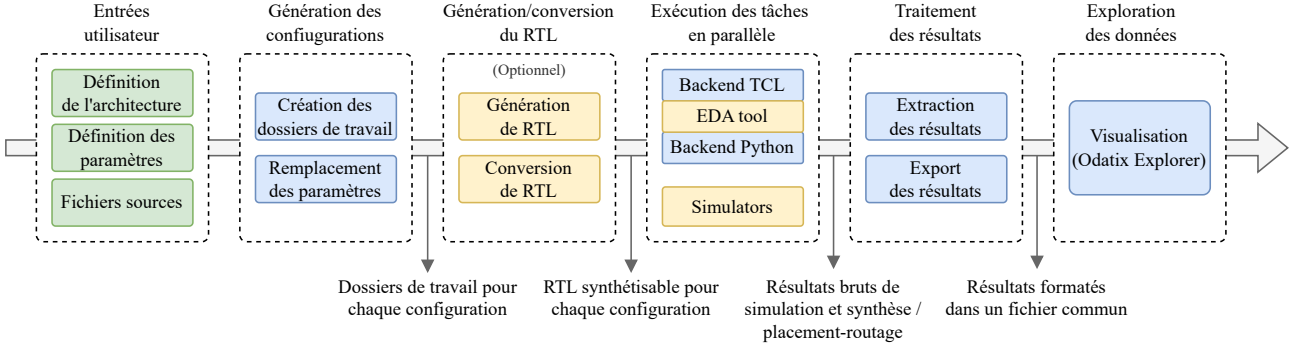


FIGURE 2 : Schéma de la chaîne de traitement d'Odatix

Le Listing 1 montre un exemple de fichier de réglages pour un counter défini en systemverilog. Les Listings 2 et 3 illustrent deux exemples de fichiers de paramètres.

```yaml
rtl_path: "examples/alu_sv"

top_level_file: "alu_top.sv"
top_level_module: "alu_top"

clock_signal: "i_clk"
reset_signal: "i_rst"

# delimiters for parameter files
use_parameters: Yes
start_delimiter: "#("
stop_delimiter: ")("

# target-specific options
xc7a100t-csg324-1:
   fmax_synthesis:
      lower_bound: 50
      upper_bound: 800
   custom_freq_synthesis:
      lower_bound: 200
      upper_bound: 1800
      step: 200
```

Listing 1 : Exemple de réglages d'architecture

```
  parameter BITS = 16
```

Listing 2 : Exemple de fichier de paramètres

```
  parameter BITS = 32
```

Listing 3 : Autre exemple de fichier de paramètres

Odatix permet également de définir des domaines de paramètres afin de faciliter les combinaisons de paramètres et l'exploration des résultats. Les paramètres peuvent être générés automatiquement à l'aide de nombreuses méthodes de définitions d'ensembles : intervalles, listes, puissances de deux, multiples, fonctions, unions, intersections, et autres.

3.2 Génération RTL

Odatix peut exécuter une commande de génération RTL avant le remplacement des paramètres. Celle-ci peut être une génération par HLS ou via Chisel[3] par exemple. Mais il peut aussi s'agir d'une conversion, par exemple, de VHDL vers Verilog.

3.3 Synthèse logique

Les outils EDA convertissent le code RTL en un netlist via plusieurs étapes (analyse, synthèse générique, synthèse spécifique et optimisations). sPour les ASIC, une bibliothèque de cellules est nécessaire, tandis que pour les FPGA, la logique est mappée sur des LUTs et parfois sur des blocs DSP. Odatix supporte les deux approches, en permettant soit à l'outil de charger automatiquement les données technologiques, soit en exécutant des scripts dédiés à la cible.

3.4 Synthèses logiques

Odatix permet de lancer des synthèses à des fréquences précises, afin de comparer les caractéristiques des configurations architecturales synthétisées, à diverses fréquences. Celles-ci peuvent être définies par listes ou par intervalle et pas.

3.5 Détermination de la fréquence maximale

La fréquence maximale d'un circuit est déterminée par le chemin critique entre registres. Pour l'estimer, il est possible de synthétiser le design et d'effectuer une analyse statique de temporisation (STA). Comme le résultat dépend de la cible et des choix de l'outil, une recherche dichotomique (binary search) est mise en œuvre pour converger vers la fréquence réelle. La synthèse à la fréquence la plus haute, respectant des contraintes de timing, est sauvegardée.

3.6 Simulation

Odatix offre une grande flexibilité totale pour la simulation. L'utilisateur a juste à définir une commande de lancement de simulation, qui est exécutée pour chaque configuration dans un répertoire de travail dédié. Ceci permet l'utilisation d'outils de simulation variés sans contrainte.

3.7 Planification parallèle des tâches

Pour optimiser le temps d'exécution sur les processeurs multi-cœurs, Odatix exécute chaque tâche de synthèse ou de simulation en parallèle. L'utilisateur peut limiter le nombre de tâches simultanées pour éviter toute surcharge, tandis que les tâches en attente sont automatiquement planifiées dès qu'un créneau se libère. Une interface interactive dans le terminal permet le suivi de la progression et des logs de chaque tâche.

3.8 Traitement des résultats

Chaque tâche génère des résultats stockés dans des répertoires hiérarchisés par type de tâche, outil, cible et architecture. Les données (issues de synthèse et de simulation) sont extraites et formatées de manière uniforme, grâce à des définitions flexibles (expressions régulières, lecture de fichiers CSV ou YAML, et opérations sur d'autres métriques). Cette approche permet une personnalisation aisée des métriques et assure la compatibilité entre outils.

3.9 Interface d'exploration des résultats

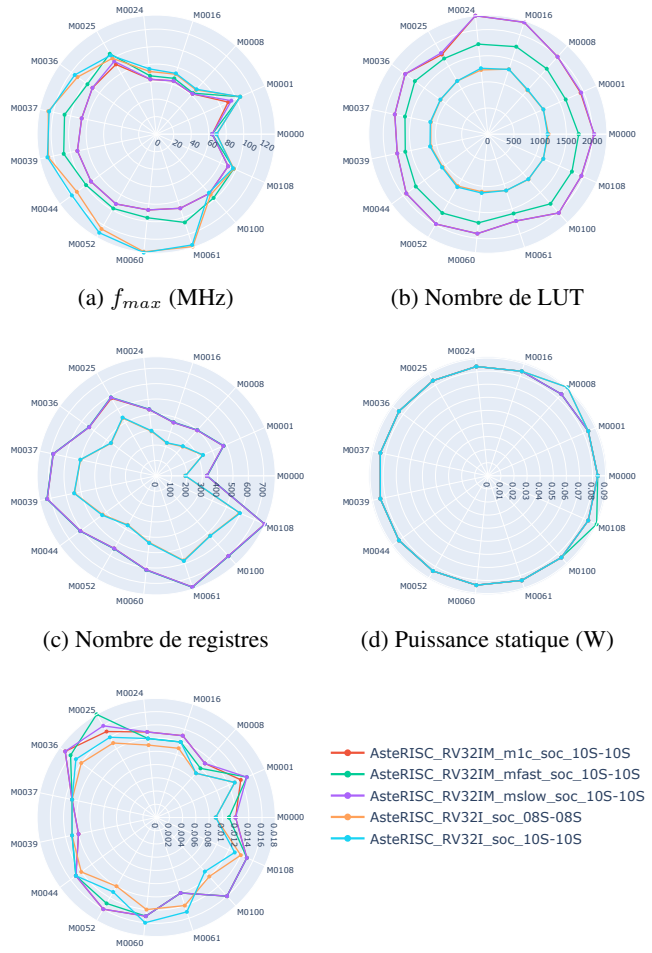
Odatix propose une interface web interactive (basée sur `plotly`) qui permet de comparer visuellement les configurations sur une large gamme de métriques. Différents modes d'affichage (lignes, barres, radar, nuages de points 2D et 4D) sont disponibles et les graphiques peuvent être exportés en vectoriel (SVG, TeX/pgfplots) ou en formats matriciels (PNG, JPEG, WEBP).

4 Exemple illustratif

L'environnement du processeur AsteRISC[7] utilise Odatix pour automatiser à la fois la simulation et l'implémentation technologique[15]. Ce processeur se caractérise par sa flexibilité, offrant une large gamme de configurations architecturales.

La Figure 2 représente des courbes générées par Odatix, après des synthèses f_{max} sur de nombreuses configurations d'AsteRISC (`MXXX`, visibles sur l'axe angulaire). Plusieurs

implémentations de la multiplication et configurations de mémoires sont représentés par des courbes aux couleurs différentes. Ces résultats mettent en évidences les compromis qui peuvent être faits entre différentes métriques.



(a) f_{max} (MHz)

(b) Nombre de LUT

(c) Nombre de registres

(d) Puissance statique (W)

(e) Puissance dynamique (W)

FIGURE 2 : Résultats d'AsteRISC sur FPGA

5 Conclusion

Odatix offre une solution puissante pour l'implémentation et l'optimisation d'architectures numériques. Ses capacités d'automatisation de la simulation et de l'implémentation, l'évaluation de multiples configurations et la visualisation des différentes métriques, en font un outil utile pour les chercheurs et ingénieurs. Sa nature flexible et open-source[16], associée à sa capacité d'intégrer plusieurs outils et de cibler aussi bien des implémentations FPGA qu'ASIC, permet aux concepteurs d'optimiser leurs architectures selon divers indicateurs, incluant les performances, l'utilisation des ressources et la consommation d'énergie.

La conception configurable d'Odatix garantit qu'il pourra être adapté à de nouveaux outils qui pourraient émerger dans le futur. À l'avenir, Odatix devrait justement étendre son support à d'autres outils existants, tels Intel Quartus Prime et F4PGA[17]. Il est également prévu d'intégrer des algorithmes d'exploration de l'espace de conception dans Odatix, similaires à ceux de Dovado[18], afin d'étendre ses capacités pour des architectures à nombre élevé de paramètres, sans avoir recours à une exploration exhaustive.

Références

[1] Dirk KOCH et al. "FPGA Versus Software Programming : Why, When, and How ?" In : *FPGAs for Software Programmers*. Sous la dir. de Dirk KOCH et al. Cham : Springer International Publishing, 2016.

[2] Design Automation Standards COMMITTEE et al. "IEEE Standard for SystemVerilog Unified Hardware Design, Specification, and Verification Language Standard IEEE 1800". In : (2005).

[3] Jonathan BACHRACH et al. "Chisel : Constructing Hardware in a Scala Embedded Language". In : DAC '12. New York, NY, USA, 3 juin 2012.

[4] Krste ASANOVIĆ et al. *The Rocket Chip Generator*. EECS Department, University of California, Berkeley, avr. 2016.

[5] Sahan BANDARA et al. "BRISC-V : An Open-Source Architecture Design Space Exploration Toolbox". In : FPGA '19. New York, NY, USA, 20 fév. 2019.

[6] Paolo MANTOVANI et al. "HL5 : A 32-Bit RISC-V Processor Designed with High-Level Synthesis". In : *CICC 2020*. Mars 2020.

[7] Jonathan SAUSSEREAU et al. "AsteRISC : A Size-Optimized RISC-V Core for Design Space Exploration". In : *ISCAS 2023*. Mai 2023.

[8] C. WOLF. *PicoRV32—A Size-Optimized RISC-V CPU*. URL : `https://github.com/YosysHQ/picorv32`.

[9] Jason ANSEL et al. "OpenTuner : An Extensible Framework for Program Autotuning". In : *PACT 2014*. Août 2014.

[10] Michel A. KINSY et al. "Heracles : A Tool for Fast RTL-based Design Space Exploration of Multicore Processors". In : FPGA '13. 2013.

[11] Yakun Sophia SHAO et al. "Aladdin : A Pre-RTL, Power-Performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures". In : *ISCA 2014*. Juin 2014.

[12] Florent KERMARREC et al. *LiteX : An Open-Source SoC Builder and Library Based on Migen Python DSL*. 5 mai 2020. Prépubl.

[13] Chen BAI et al. "BOOM-Explorer : RISC-V BOOM Microarchitecture Design Space Exploration Framework". In : *ICCAD 2021*. Nov. 2021, p. 1-9.

[14] Mohamed SHALAN et al. "Building OpenLANE : A 130nm OpenROAD-based Tapeout- Proven Flow : Invited Paper". In : *ICCAD 2020*. 2020.

[15] Jonathan SAUSSEREAU. *AsteRISC*. URL : `https://github.com/jsaussereau/AsteRISC`.

[16] Jonathan SAUSSEREAU. *Odatix*. URL : `https://github.com/jsaussereau/Odatix`.

[17] *F4PGA - FOSS Flow For FPGA*. CHIPS Alliance. URL : `https://github.com/chipsalliance/f4pga`.

[18] Daniele PALETTI et al. "Dovado : An Open-Source Design Space Exploration Framework". In : *IPDPSW 2021*. Juin 2021.