FPGA pour les débutants

de la programmation des microcontrôleurs à celle des FPGA

Theo Mulder (Pays-Bas)

Il existe de nombreuses cartes de développement à microcontrôleur peu coûteuses, mais les FPGA restent chers. Pour les applications qui nécessitent des performances de calcul plus élevées (par exemple, l'intelligence artificielle) ou des E/S plus rapides (par exemple, une caméra à grande vitesse, un écran à grande vitesse, un CAN ou un CNA rapide), il est préférable de trouver des cartes FPGA de petite taille et d'un prix abordable. Plongeons dans le monde passionnant des FPGA pour les débutants !

En cherchant des FPGA peu coûteux à utiliser, il est utile de comprendre les tendances actuelles du marché. Le marché des FPGA était évalué à 8,0 milliards de dollars en 2022, et devrait atteindre 15,5 milliards de dollars d'ici 2027. Le secteur est concurrentiel et a connu une consolidation importante des fournisseurs. Aujourd'hui, les marques restantes sont AMD/Xilinx, Intel/Altera, Lattice, Microchip, QuickLogic, Efinix, Flex Logix, GOWIN, Achronix, S2C et Renesas.

Grâce aux acquisitions, le paysage ne cesse de changer. Dès 1999, Lattice a acquis Vantis, puis Acree en 2001, et Microsemi a acquis Actel en 2010. En 2013, les principaux acteurs étaient Xilinx, Altera, Actel, Vantis, Lattice, Lucent, QuickLogic et Cypress. Cependant, le mouvement de fusions et d'acquisitions n'a jamais cessé ; même Xilinx et Altera, ont été rachetés par AMD et Intel, respectivement.

En 2019, Xilinx dominait le marché avec une part de 52 %, suivi par Altera avec 35 %, et des parts plus modestes étaient détenues par Microchip, Lattice et d'autres. Malgré la domination de ces grandes entreprises, les petits fournisseurs de FPGA restent actifs, une excellente situation pour soutenir différents types d'entreprises et promouvoir l'innovation.

L'utilisation accrue des FPGA dans l'industrie automobile offre des opportunités à des fournisseurs tels que Lattice, car les concepteurs industriels recherchent des alternatives rentables aux offres plus onéreuses des plus grandes entreprises. Renesas apparaît comme un concurrent important grâce à sa présence bien établie dans le secteur automobile.

La dynamique du marché des FPGA, avec ses changements concurrentiels, ses acquisitions et ses nouveaux acteurs, pourrait profiter à ceux qui recherchent des FPGA à bas prix pour leurs projets.

Cartes FPGA abordables pour les débutants

Étant donné que les chaînes de compilation des différents fabricants de FPGA sont très différentes les unes des autres et que l'apprentissage de l'utilisation d'un nouvel outil logiciel prend du temps, il est judicieux de différencier les cartes en fonction des fabricants de FPGA. Ainsi, le temps d'apprentissage aura été bien investi si, à l'avenir, vous pouvez décider de passer à un modèle plus haut de gamme du même fabricant de FPGA.

Les trois principaux fabricants à prendre en considération sont Altera/Intel, Xilinx/AMD et Lattice. Ce dernier a tendance à proposer des modèles moins puissants et plus abordables que les deux premiers.

En commençant par les cartes dotées de FPGA de Lattice Semiconductor, la carte d'évaluation Olimex iCE40HX1K-EVB, dont le prix est de 15 €, constitue un point d'entrée très abordable, bien qu'elle nécessite un programmateur externe. Cette carte dispose d'un iCE40HX1K avec 1280 éléments logiques (LE). Fait intéressant, ce FPGA possède une mémoire de configuration interne, non volatile, programmable via l'interface SPI d'une carte Arduino que vous possédez peut-être, grâce au croquis fourni par Olimex. Le programmateur officiel de Lattice utilise un FT232H de FTDI, et il est également possible d'utiliser un breakout FT232H. L'une des caractéristiques intéressantes de cette famille de FPGA est qu'elle est compatible avec l'outil graphique open source Icestudio, qui permet aux utilisateurs d'expérimenter la programmation graphique par blocs.

La famille TinyFPGA propose des produits intéressants pour ceux qui recherchent d'autres modèles très compacts, abordables. et, compatibles avec des plaques d'essai. Le TinyFPGA BX est proposé à 40 €, avec un iCE40LP8K de Lattice, également compatible avec Icestudio. Les versions AX1 et AX2 sont disponibles respectivement à 13 et 17 €, offrant des alternatives plus

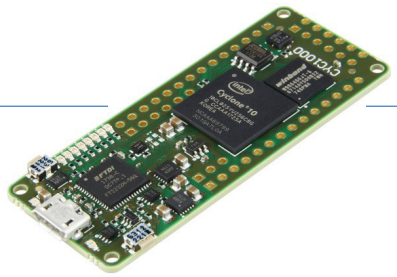


Figure 1. La carte CYC1000.

petites et moins chères, mais avec moins de fonctions. Malheureusement, ces trois modèles sont souvent en rupture de stock. L'Upduino v3.1 d'un autre fabriquant, au prix de 30 €, offre un FPGA iCE40UP5K (5.3 kLE).

Le iCEstick de Lattice est une option compacte au format d'une clé USB, intégrant un FPGA iCE40HX1K et un programmateur embarqué, au prix de 48 €. C'est l'outil officiel de Lattice, il fonctionne donc bien avec le logiciel de Lattice et il existe de nombreux tutoriels sur son utilisation.

Pour compléter les options de Lattice, mentionnons également une plateforme open-source dans un format Arduino UNO, l'Alhambra II (iCE40HX4K, 3,52 kLE, 60 €) et aussi la carte Go Board (iCE40HX1K, 70 €) qui permet de soutenir le créateur d'excellents tutoriels de Nandland.com.

Passant à Altera (maintenant propriété d'Intel), la carte MAX1000 de Trenz Electronic fournit un FPGA MAX10, disponible en 8 ou 16 kLE (34 € à 49 €), offrant une conception compatible avec les plaques d'essai. La CYC1000 [1] (**figure 1**), également de Trenz, au prix de 40 €, surenchérit avec un FPGA Cyclone 10 (25 kLE).

Pour des cartes plus classiques et riches en fonctionnalités, il y a les solutions de Terasic. Certains de leurs produits sont chers, mais d'autres sont très intéressants pour les débutants, comme le DE0-nano qui offre un FPGA Cyclone IV avec 22 kLE et une multitude de fonctionnalités embarquées pour 110 €. Le DE10-Lite (140 €) est une carte riche en fonctionnalités avec un FPGA MAX10 (50 kLE), diverses options d'E/S, des afficheurs à sept segments, des LED et des interrupteurs, ainsi qu'un support pour les shields Arduino.

Enfin, sous l'égide d'AMD (anciennement Xilinx), les Cmod S7 et Cmod A7-35T de Digilent proposent tous deux des designs compatibles avec les plaques d'essai, avec des FPGA Spartan 7 et Artix 7, respectivement de 23,4 et 33,3 kLE, au prix de 90 € chacun.

Pour une expérience d'apprentissage riche avec une carte plus grande, le Digilent Basys 3 apporte à la table un FPGA Artix 7 avec 33,3 kLE, des options d'E/S étendues, y compris une sortie VGA et un hôte USB, au prix de 155 €.

Dans cet article, je me concentrerai sur les produits Intel/Altera. Je recommande le CYC1000. Dans les sections suivantes, nous allons expérimenter avec Quartus Prime Lite d'Intel, qui est la version gratuite de l'outil de développement officiel pour les FPGA Altera/Intel.

Installation

Sur la page Intel Quartus Prime Design Software [2], vous trouverez les liens de téléchargement de la dernière version, sous Windows ou Linux. Il est nécessaire d'avoir une bonne connexion Internet, car le fichier fait environ 5,5 Go. L'installateur nécessite 15 GB d'espace libre. Certaines parties de Quartus Prime Lite utilisent du code provenant des premières versions de Quartus. Par conséquent, vous devez éviter les espaces dans les noms de fichiers, même si cela peut paraître surprenant par rapport aux normes d'aujourd'hui.

Seul l'outil de conception de bas niveau est nécessaire. Il existe des packages supplémentaires qui dépassent le cadre de cette introduction, tels que DSP Builder (pour les applications de traitement du signal), High Level Synthesis Compiler (utilisé avec C++ en entrée, pour les applications avancées) ainsi que Nios II Embedded Design Suite (utilisé pour implémenter un processeur logiciel Nios II sur un FPGA).

Créer un nouveau projet

Commencez par créer un répertoire - par exemple, *C:\NProjects\NIntelFpga\NElektor\NBeginExample*. Ensuite, lancez Quartus Prime Lite. L'écran principal est représenté sur la **figure 2**. Allez dans *File*, puis sélectionnez *New Project Wizard*.

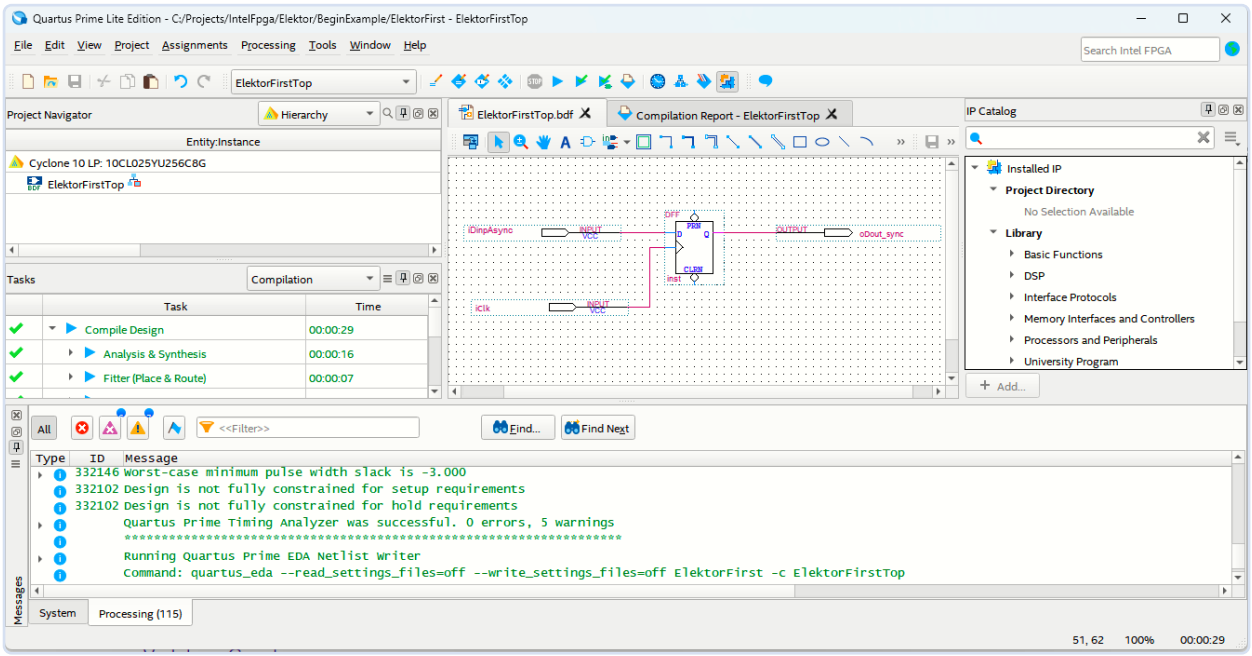
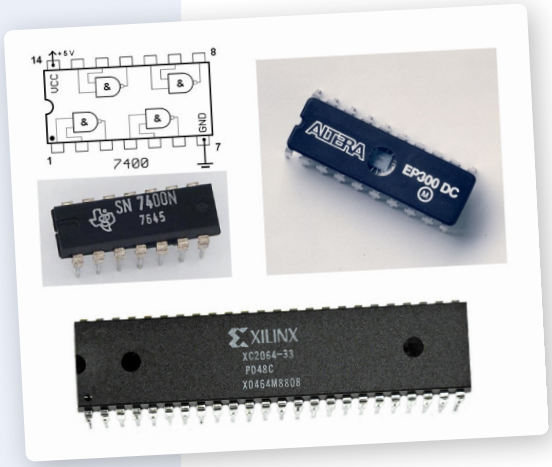


Figure 2. Quartus Lite Edition, avec ElektorFirstTop.bdf.

Historique : du TTL au FPGA

Avant l'avènement des FPGA, les circuits numériques s'appuyaient principalement sur des puces TTL et des circuits programmables tels que les FPLD, les PLA et les PAL. La série TTL SN7400 fonctionnait jusqu'à 20 MHz. Il y a 40 ans, Altera a introduit l'EP300, suivi par le XC2064 de Xilinx. L'EP300 était gravé avec une finesse de 5000 nm et avait des vitesses allant jusqu'à environ 10 MHz, tandis que les FPGA de milieu de gamme modernes sont gravés avec une finessde 20 nm et atteignent des vitesses d'horloge d'au moins 250 MHz. L'EP300 était un circuit reprogrammable à 20 broches utilisant le logiciel A+PLUS sur des PC IBM fonctionnant sous DOS. Le XC2064 de Xilinx comportait des cellules logiques de 1200 portes chacune et utilisait une technologie CMOS à faible consommation, avec un logiciel de conception comprenant Future-Net, VIEW-logic et XACT-Design-Editor. Le XC2064 était annoncé avec un taux de basculement interne de 100 MHz, mais il atteignait généralement 70 MHz.
Au départ, les outils logiciels FPGA utilisaient la capture schématique, mais celle-ci a ensuite été complétée par des langages de description de matériel (HDL) basés sur le texte, tels que AHDL pour Altera, et VHDL et Verilog pour Xilinx. Aujourd'hui, les deux sociétés prennent en charge les langages VHDL et Verilog. Les anciennes versions de l'outil Quartus d'Altera utilisaient l'AHDL avec des fonctions de simulation intégrées, nécessitant par la suite l'utilisation de ModelSim pour la simulation due au manque de prise en charge de l'AHDL. Aujourd'hui, tous les fabricants proposent une version gratuite de leurs outils.



Après avoir consulté la fenêtre de bienvenue, cliquez sur *Next*. Lorsque vous êtes invité à indiquer le répertoire de travail, choisissez le répertoire que vous avez créé. Vous devez choisir un nom de projet, ainsi qu'un nom pour l'entité de conception de premier niveau, que le compilateur utilisera comme racine du design. Dans cet exemple, nous avons utilisé *ElektorFirst* et *ElektorFirstTop*. Ensuite, choisissez *Empty Project*. Vous accéderez alors à l'étape *Add files*. Comme nous n'avons pas de fichiers, il suffit de cliquer à nouveau sur *Next*.
Vous accéderez alors à une page où vous pourrez sélectionner un appareil, ce qui peut être effectué via l'un des deux onglets : *Device* et Board. Dans ce dernier, plusieurs cartes de développement sont listées dans les familles MAX10 et Cyclone V et il est possible d'installer des cartes supplémentaires ultérieurement. Dans l'onglet *Device*, vous pouvez choisir la famille et la référence exacte du FPGA que vous utilisez, par exemple le *10LC025YU256C8G* dans la famille Cyclone 10 LP si vous avez la carte CYC1000. La famille Cyclone 10 est constituée de modules à faible coût et à faible consommation d'énergie, dérivés du Cyclone V.
Si vous n'avez pas de carte de développement, vous pouvez toujours sélectionner cette carte pour avancer dans le tutoriel et expérimenter le processus de compilation. Cliquez sur *Next*, *Next* à nouveau et *Finish*.

Structure du répertoire du projet

Jetez un coup d'œill à votre répertoire de travail. Vous remarquerez qu'il y a un répertoire db que Quartus utilisera plus tard. Le fichier nommé *ElektorFirst.qpf* est le fichier de projet *Quartus (QPF)*. Lorsque vous souhaitez reprendre votre travail plus tard, il vous suffit de double-cliquer sur ce fichier pour relancer le logiciel Quartus. Il existe également un fichier appelé *ElektorFirst.qsf*, qui est le *Quartus Settings File (QSF)*. Ces deux fichiers sont des fichiers texte, vous pouvez donc les examiner si vous êtes curieux - il suffit de cliquer avec le bouton droit de la souris et de choisir *Open with* pour choisir un éditeur de texte.

Ouvrir un projet

Dans le coin supérieur gauche, on voit *Project Navigator* avec l'option *Hierarchy* sélectionnée. Recherchez *ElektorFirstTop*, que nous avons nommé plus tôt. Lorsque vous cliquez dessus, une fenêtre contextuelle apparaît et indique « Can't find design entity ElectorFirstTop », car nous n'avons pas encore créé de fichier. Cliquez sur *OK*, puis allez dans le menu *File*. Ici, vous pouvez voir les différents types de fichiers que vous pouvez créer. Choisissez *Block Diagram/Schematic File* et cliquez sur *OK*.
Vous avez maintenant un canevas blanc dans le volet central, intitulé *Block1.bdf* en haut. Allez dans *File*, sélectionnez *Save As* et le logiciel vous proposera d'enregistrer le fichier sous *ElectorFirstTop.bdf*. Parfait, cliquez sur *Save*. Si vous cliquez sur *ElektorFirstTop* dans le *Project Navigator* et *Hierarchy*, vous arrivez sur cette feuille de schéma vide.

Ajouter des composants à notre premier schéma

Afin que tous nos lecteurs, même ceux qui ne maîtrisent pas les langages Verilog et VHDL, puissent suivre ce tutoriel, nous allons utiliser la programmation schématique. Cliquez sur l'onglet *ElektorFirstTop.bdf* pour ouvrir la feuille, faites un clic droit n'importe où sur la feuille et choisissez *Insert Symbol*. Le menu Bibliothèques apparaît. Cliquez sur le petit triangle à côté de *Libraries* pour développer la liste, où vous verrez des catégories comme *Megafunctions*, *Other*, *Primitives*, et plus encore. Il y a beaucoup à découvrir ici, alors n'hésitez pas à naviguer.
Faites défiler jusqu'au bas de la fenêtre

Libraries et dans le champ *Name* tapez « Input ». Un symbole d'entrée apparaît ; cliquez sur *OK*. Placez ce symbole quelque part sur la page, idéalement en haut à gauche. Ajoutez ensuite une deuxième entrée en cliquant avec le bouton droit de la souris et en choisissant *Insert Symbol*. Puisque « Input » devrait toujours figurer dans le champ Name, cliquez à nouveau sur *OK* et placez cette nouvelle entrée en dessous de la première sur votre feuille.
Ensuite, nous allons ajouter une bascule D. Cliquez avec le bouton droit de la souris, sélectionnez *Insert Symbol* et, dans le champ *Name*, tapez « dff ». Choisissez *dff* parmi les options proposées et cliquez sur *OK*, puis placez-le sur la feuille. Pour terminer cette étape, cliquez avec le bouton droit de la souris, sélectionnez *Insert Symbol*, saisissez « Output » dans le champ *Name* et cliquez sur *OK*. Placez également cette sortie sur la feuille.

Renommer les broches

Nous devrons attribuer des noms significatifs aux broches. Il est possible d'ajouter des préfixes ou des suffixes pour donner des détails selon nos besoins, comme i pour entrée (*input*), o pour sortie (*output*), sync ou async pour synchrone ou asynchrone, respectivement, etc. Nous allons donc renommer l'entrée *pin_name1* en *iDinpAsync*. Pour ce faire, double-cliquez sur *pin_name1* et saisissez le nouveau nom. Renommez également *pin_name2* en *iClk* et *pin_name3* en *oDout_sync*. Cliquez ensuite sur *File* et *Save*.

Ajouter des fils

Cliquez sur la broche *iDinpAsync*. Ceci activera automatiquement l'outil *Orthogonal Node* utilisé pour dessiner des fils. Commencez à dessiner à partir de cette broche et connectez-la à l'entrée D de la bascule D (DFF). Ensuite, procédez à la connexion de *iClk* à l'entrée d'horloge de la DFF, qui est indiquée par un triangle dans le symbole. Enfin, créez une connexion de la sortie Q de la DFF au symbole de sortie *oDout_sync*. Sauvegardez à nouveau.

Compiler

Pour compiler le dessin ou modèle, il suffit de cliquer sur l'icône du triangle bleu dans la barre d'icônes. Gardez un œil sur le volet des tâches à gauche pour suivre la progression et surveillez les messages qui défilent au bas de l'écran. Il s'agit d'un processus assez intéressant.
Après la compilation, le volet central affiche le volet *Flow Summary*. Ce résumé indique qu'un seul des 24 624 éléments logiques a été utilisé. Il est rare d'utiliser 100 % des éléments logiques d'un composant ; dans les circuits les plus complexes et les plus optimisés, environ 80 % des éléments logiques peuvent être utilisés. Au-delà de ce seuil, vous pouvez rencontrer des difficultés de routage ou le composant peut ne pas répondre aux exigences de synchronisation en raison de l'encombrement.
Le résumé montre également qu'un seul registre a été utilisé et que trois des 151 broches possibles sont utilisées, ce qui équivaut à 2 % du nombre total de broches. Il faut toutefois garder à l'esprit que certaines broches peuvent être réservées et non disponibles pour un usage général.
Vérifiez à nouveau votre répertoire de projet. Vous y trouverez de nouveaux sous-dossiers. Jetez un coup d'œil dans le dossier *db* pour vous faire une idée du travail de fond effectué. Le fichier *Elektor-FirstTop.sof* que vous utiliserez pour programmer votre FPGA se trouve dans le dossier *output_files*.

Choisir le FPGA idéal pour votre projet

Lorsqu'on choisit des FPGA et des cartes, il y a plusieurs paramètres à prendre en compte. Le nombre d'éléments logiques, qui donne une idée approximative de la taille du FPGA, en est un. Il faut également tenir compte de la nécessité de disposer d'entrées/sorties à faible ou à grande vitesse. Les FPGA haut de gamme dotés d'une mémoire rapide et d'émetteurs-récepteurs sont coûteux, tout comme les outils logiciels associés. Bien entendu, les classifications de haut, moyen et bas de gamme sont subjectives et peuvent varier d'un fournisseur à l'autre. Pour les deux principaux fournisseurs de FPGA, ce qui est considéré comme une performance de milieu de gamme peut être qualifié de haut de gamme par Lattice.
En ce qui concerne les E/S, les émetteurs-récepteurs sont souvent mentionnés. Il s'agit de connexions série qui offrent des taux de transfert élevés (par exemple de 4 à 32 Gbps), mais qui affectent considérablement le coût. Ils ne sont pas nécessaires pour toutes les applications. Au lieu de connexions PCIe ou COM Express à haut débit, une solution plus économique et plus pratique pour de nombreuses applications consiste à utiliser l'USB, éventuellement avec un dispositif FTDI pour l'interfaçage. Pour connecter des ADC ou DAC à grande vitesse à un FPGA, il est possible d'utiliser l'interface JESD204B, mais cela nécessite des émetteurs-récepteurs à la fois du côté FPGA et du côté ADC/DAC, ce qui peut s'avérer coûteux. Une attention particulière à l'intégrité des signaux est nécessaire lors du routage de la carte, mais la disposition du circuit est quelque peu simplifiée par le nombre réduit de pistes. Une autre option lorsque des débits importants sont nécessaires est d'utiliser plusieurs paires LVDS (Low Voltage Differential Signaling) plus lentes en parallèle. Dans tous les cas, il est utile de faire un calcul assez précis de vos besoins en termes de débit pour pouvoir choisir un composant adéquat.
J'ai tendance à considérer que les interfaces sont à bas débit lorsqu'elles fonctionnent à moins de 1000 Mbit/s, et à haut débit au-delà. Le coût des FPGA augmente avec le nombre d'interfaces à haut débit. Cependant, dans de nombreux cas, même un LVDS à vitesse modérée n'est pas nécessaire : des broches d'E/S à usage général sont suffisantes.
Sur les FPGA, les broches d'E/S à usage général sont les plus lentes des interfaces disponibles, mais elles ont toujours des taux de basculement plus rapides que les broches d'E/S à usage général sur les microcontrôleurs. Elles sont généralement compatibles avec une logique de 3,3 V, 2,5 V ou 1,8 V. Si vous pensiez utiliser un microcontrôleur pour votre prochain projet, mais avec des E/S plus rapides que ce que les microcontrôleurs typiques peuvent produire, les FPGA constituent une solution compacte, car il est possible d'implémenter un processeur (RISC-V ou autre) dans la structure FPGA.

VHDL ou Verilog?

Pour les débutants, le choix entre Verilog et VHDL dans le cadre de l'apprentissage des FPGA dépend de plusieurs facteurs. Pour ceux qui souhaitent travailler dans les secteurs de la défense ou de l'avionique, VHDL est le langage prédominant, en particulier dans les pays européens comme l'Allemagne et la France. En revanche, aux États-Unis et dans de nombreux secteurs autres que la défense, Verilog est plus couramment utilisé.

Cependant, toute personne souhaitant faire carrière dans le domaine de la conception des circuits numériques aura probablement besoin de maîtriser les deux langages. Il est important de vous adapter à votre objectif d'apprentissage ; s'il s'agit d'un objectif académique, alignez-vous sur le langage enseigné dans vos cours, tandis que pour les environnements professionnels, il est préférable d'utiliser ce que vos collègues utilisent.

D'un point de vue technique, Verilog a tendance à être plus compact et à ressembler au langage C à certains égards, ce qui n'est pas toujours avantageux car cela pourrait conduire à une vision axée sur le logiciel plutôt que sur le matériel. Le caractère textuel de VHDL peut permettre d'éviter certaines erreurs que Verilog pourrait manquer. Cette distinction peut faire du VHDL un meilleur point de départ pour ceux qui préfèrent un langage qui impose un processus de pensée centré sur le matériel, ce qui est crucial pour la conception de FPGA.

Enfin, examinez le fichier *ElektorFirstTop.pin* en l'ouvrant avec un éditeur de texte. Ce fichier montre les connexions de broches que le compilateur a configurées. Comme nous n'avons pas spécifié de conditions particulières, Quartus a sélectionné les broches au hasard. En règle générale, en tant que concepteur, vous devriez définir vos affectations de broches. Recherchez les broches *iDinpAsync*, *iClk* et *oDout_sync*. Dans ma situation, *iDinpAsync* est connecté à la broche T4, fonctionne à un niveau de 2,5 V, et se trouve dans le *bank* 3 du FPGA.

Programmer le FPGA

Bien sûr, nous sommes curieux de voir si cela fonctionne avec un vrai FPGA, alors essayons ! Tout d'abord, connectez votre carte à un port USB disponible sur votre ordinateur et suivez les étapes pour charger le fichier de configuration dans la SRAM du FPGA. Vous pouvez également le charger sur la mémoire flash, ce qui permet au FPGA de conserver le circuit même après les cycles d'alimentation.

Pour ce faire, allez dans *Tools* puis dans Programmer, ou double-cliquez simplement sur *Program Device* dans la liste des tâches. Cliquer sur Hardware Setup, et sélectionner *USB-Blaster [USB-0]*. Cliquez ensuite sur *Add File* et choisissez le fichier objet *SRAM ElektorFirstTop.sof* qui a été généré dans le répertoire *output_files* après la compilation. Cliquez sur *Start*. Pensez à sauvegarder la configuration du programmateur, par exemple sous le nom de *example1.cdf*, afin qu'il s'ouvre avec ces paramètres lors de sa prochaine utilisation. Après quelques secondes, le transfert est terminé ! Nous avons réussi à charger le circuit sur le FPGA, où il restera jusqu'à ce que l'alimentation soit déconnectée. Félicitations, vous avez maintenant une bascule D très sophistiquée, vous pouvez la tester en branchant des fils et des interrupteurs sur les broches appropriées de la carte CYC1000 !

Design hiérarchique

Il peut être très utile de concevoir une partie d'un système une fois pour toutes, et de pouvoir ensuite l'utiliser à plusieurs emplacements du système. Pour ce faire, nous pouvons intégrer une partie du schéma dans ce que l'on appelle un symbole. À titre d'exemple, créons un symbole pour notre circuit.

Tout d'abord, créez un nouveau fichier de diagramme fonctionnel/schéma comme nous l'avons fait dans la section « Ouvrir un projet » : *File*, *New*, etc. Nommez-le *Synchronizer.bdf*. Ensuite, dans la fenêtre *ElektorFirstTop.bdf*, cliquez et maintenez la souris enfoncée pour sélectionner tous les composants dans un rectangle. Copiez et collez ensuite le contenu dans le fichier *Synchronizer.bdf* vide et enregistrez. Ensuite, la fenêtre *Synchronizer.bdf* étant toujours ouverte, allez dans File, sélectionnez *Create/Update*, sélectionnez *Create Symbol Files for Current File* et cliquez sur *Save* (avec le nom suggéré *Synchronizer.bsf*). Dans la fenêtre contextuelle, cliquez sur *OK*.

Un symbole pour un DFF a été créé sous le nom de *Synchronizer.bsf*. Vous pouvez fermer la fenêtre Synchronizer. Dans *ElektorFirstTop.bdf*, vous pouvez tester et utiliser ce symbole nouvellement créé. Enlevez la bascule D, déplacez les entrées et les sorties sur les côtés pour faire de la place, et faites un clic droit dans l'espace vide pour insérer le symbole comme nous l'avons fait dans la section « Ajouter des composants à notre premier schéma ». Le symbole *Synchronizer* nouvellement créé sera disponible dans le menu *Insert Symbol*. Vous pouvez maintenant connecter le symbole aux entrées et sorties appropriées et enregistrer à nouveau. Vous pouvez à nouveau compiler ce design hiérarchique.

Utiliser un langage de description du matériel

La plupart des utilisateurs préfèrent utiliser un langage de description du matériel (ou HDL), tel que Verilog ou VHDL, plutôt que des schémas de blocs. De nombreuses ressources sont disponibles pour ces langages. En outre, les schémas sont notoirement difficiles à maintenir et à mettre à jour à mesure qu'ils prennent de l'ampleur. Convertissons le fichier *Synchronizer.bdf* en un fichier Verilog. Cliquez sur *Synchronizer.bdf*, accédez à *File*, puis à *Create/Update*, et choisissez *Create HDL Design File from Current File*. Sélectionnez *Verilog HDL* et cliquez sur *OK*. Un fichier *Synchronizer.v* devrait maintenant apparaître dans votre répertoire de conception. Vous pouvez ouvrir ce fichier avec n'importe quel éditeur de texte ou avec Quartus lui-même. Allez dans File, sélectionnez Open, trouvez *Synchronizer.v*, et ouvrez-le. Voilà, vous avez maintenant la description Verilog de votre circuit de synchronisation en main. Cela peut être utile à des fins d'apprentissage ! Bien entendu, il est également possible de créer un fichier VHDL.

Vous devrez créer un nouveau symbole pour ce nouveau fichier *Synchronizer.v*. Suivez les mêmes étapes que celles décrites dans la section « Design hiérarchique » et utilisez la commande *Create Symbol for Current File*.

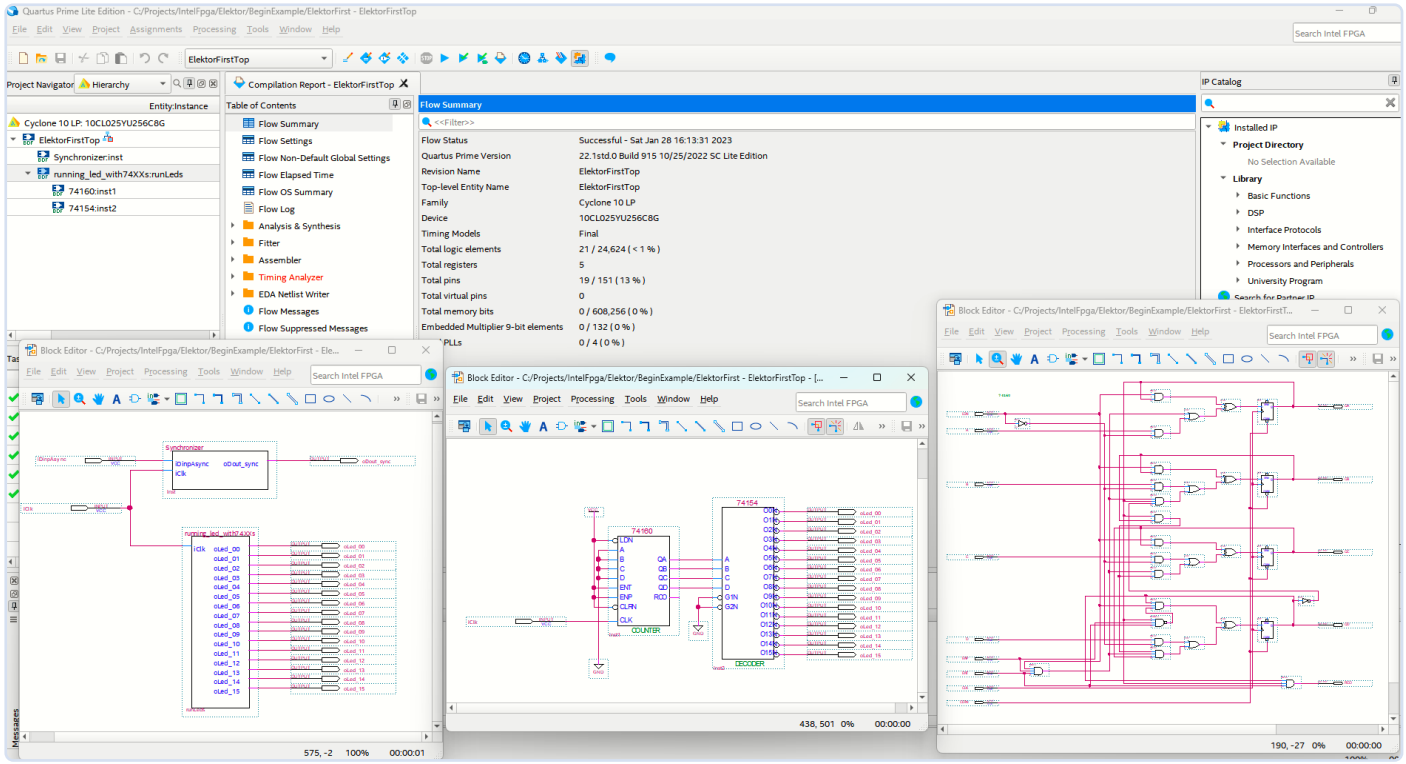


Figure 3. Quartus Lite Edition, avec l'exemple running_led_with74XXs.

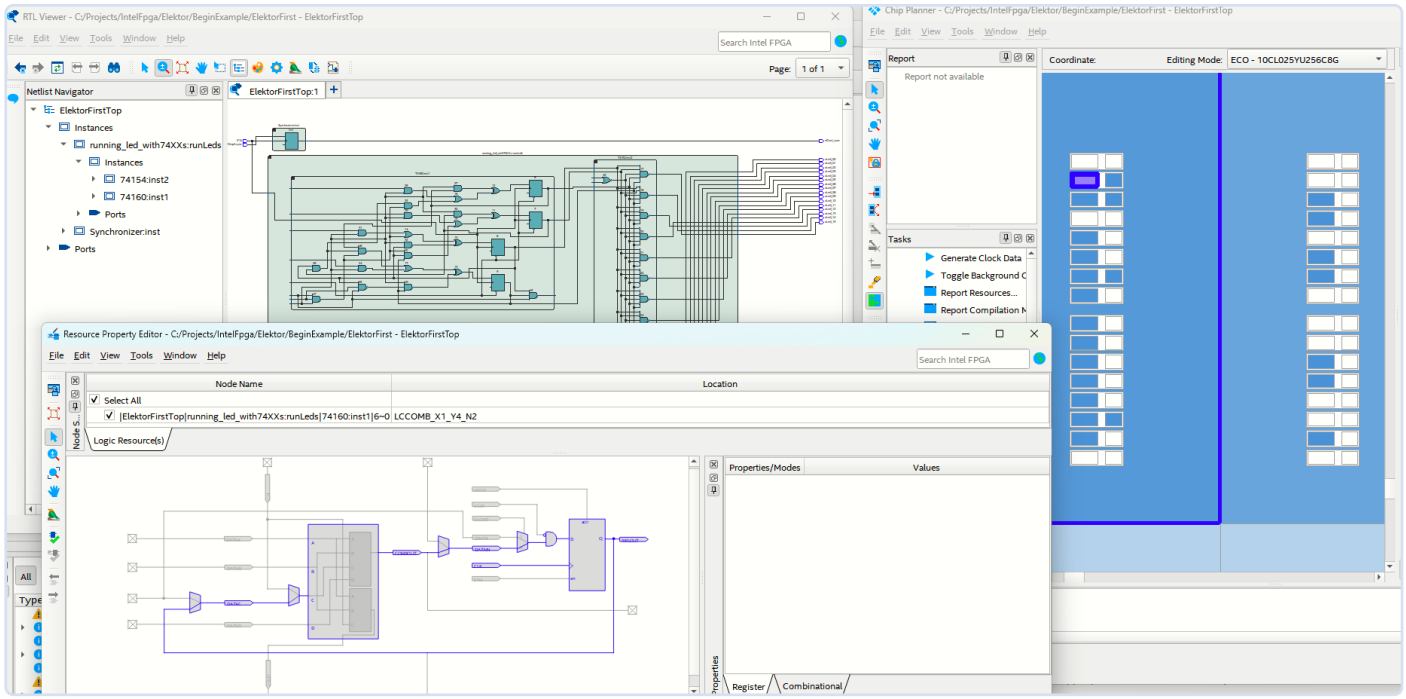


Figure 4. Quartus Lite Edition, montrant certaines des implémentations de bas niveau de l'exemple running_led_with74XXs.

Expérimenter par vous-même

N'hésitez pas à explorer et à expérimenter. Dans *ElectorFirstTop.bdf*, cliquez avec le bouton droit de la souris sur la zone schématique, choisissez *Insert Symbol* et cliquez sur le petit triangle situé à côté de la rubrique inférieure. Explorez les primitives, puis la logique, et vous découvrirez une série de composants logiques de base tels que AND, OR, XOR, etc. Allez-y et créez votre propre circuit !

De plus, pour retrouver les circuits logiques de la série 7400 dans la bibliothèque, cliquez avec le bouton droit de la souris sur la zone schématique du fichier *ElectorFirstTop.bdf*, sélectionnez *Insert Symbol* et tapez « 7400 » dans le champ *Name*. Vous trouverez une large sélection de composants de la série 7400. Vous pouvez les parcourir et prévisualiser les symboles dans le volet de droite. Il existe de nombreux circuits logiques fascinants utilisant la logique de la série 7400 disponibles en ligne, que vous pouvez émuler sur un FPGA. Ce n'est peut-être pas la méthode la plus efficace, mais cela peut certainement être une expérience divertissante et éducative ! Vous pouvez voir un exemple de circuit utilisant la logique de la série 74 à la **figure 3**.

Vous pouvez également jeter un coup d'œil au circuit complet à l'intérieur du FPGA. Dans la barre supérieure, choisissez *Tools*, *Netlist viewers*, *RTL viewer*. Vous pouvez voir le résultat à la **figure 4**. Les schémas sont visibles en haut à gauche. Pour examiner les éléments placés dans le FPGA, sélectionnez Tools, *Chip planner*. Ici, vous pouvez zoomer sur la zone contenant les cellules logiques. De plus, dans l'image, en haut à droite, une petite cellule violette est marquée. En sélectionnant une cellule logique et en cliquant dessus, vous pouvez visualiser son contenu, comme indiqué en bas à gauche.

Pour aller plus loin

Nous n'avons fait qu'effleurer la surface de la programmation FPGA en présentant la version gratuite de Quartus Prime Lite. Il offre une gamme complexe de fonctions, dont beaucoup n'ont pas pu être abordées dans cette introduction, telles que la simulation, l'analyseur logique Signal Tap, le Platform Designer, et bien plus encore. Les cartes FPGA accessibles aux débutants sont nombreuses. Pour les amateurs de logiciels libres, le projet Icestorm et l'EDI graphique Icestudio sont des points d'entrée intéressants dans ce domaine.

En outre, il existe une pléthore de ressources d'apprentissage en ligne. Des sites web comme fpga4fun [3], NandLand [4] et VHDLwhiz [5] sont d'excellents points de départ. La liste de Joel des cartes FPGA abordables [6] peut également vous aider à sélectionner le matériel adéquat.

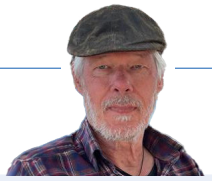
Pour les livres, la version numérique de *Free Range VHDL* de Bryan Mealy et Fabrizio Tappero est offerte gratuitement par les auteurs, et *Getting Started with FPGA*, de Russel Merrick (l'auteur des tutoriels NandLand) est également un complément intéressant à votre bibliothèque.

Le monde des FPGA est à portée de main, avec plus d'outils d'apprentissage et de communautés disponibles que jamais auparavant. C'est le moment idéal pour commencer votre voyage dans la conception numérique !

230067-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (tf.mulder@outlook.com) ou contactez Elektor (redaction@elektor.fr).



À propos de l'auteur

Theo Mulder est ingénieur en électronique. Il est expérimenté dans le traitement des signaux et a travaillé dans le domaine de l'électronique analogique et numérique, en particulier dans l'électronique médicale pour les appareils à ultrasons. Il est spécialiste du C++, des DSP et des FPGA. Il transforme les idées de recherche théoriques en application électroniques. Certains de ses propres concepts ont été brevetés. Il pense en termes de systèmes, au niveau de la carte électronique, pour trouver un bon équilibre entre le matériel analogique/numérique et le logiciel, afin de réaliser des systèmes qui fonctionnent bien avec un bon rapport prix/performance.

Produits

- **M. Dalrymple, *Microprocessor Design Using Verilog HDL* (E-book, Elektor)**
  www.elektor.fr/18518
- **Carte de développement Alchitry Au FPGA (Xilinx Artix 7)**
  www.elektor.fr/19641

LIENS

[1] Trenz Electronic — CYC1000 board: https://tinyurl.com/trenzcyc1000
[2] Intel Quartus Prime Design Software: https://tinyurl.com/downloadquartus
[3] fpga4fun: http://fpga4fun.com
[4] NandLand: https://nandland.com
[5] VHDLwhiz: https://vhdlwhiz.com
[6] Joel's list of FPGA boards: https://joelw.id.au/FPGA/CheapFPGADevelopmentBoards