Université Mohamed Khider de Biskra
Faculté des Sciences et de la Technologie
Département de génie électrique

MÉMOIRE DE MASTER

Sciences et Technologies
Electronique
Systèmes Embarqué

Réf. : .....

Présenté et soutenu par :
DJOUDI Amina **HARABI Soumaia**

Le : mardi 03 juin 2025

Utilisation FPGA/VHDL pour la gestion et le contrôle des données

---

Jury :

Mr.	Tobbeche Souad	Pr	Université de Biskra	Président
Mr.	DHIABI Fathi	MCB	Université de Biskra	Encadreur
Mr.	Okba Benelmir	MCB	Université de Biskra	Examinateur

Année universitaire : 2024-2025

Université Mohamed Khider de Biskra
Faculté des Sciences et de la Technologie
Département de génie électrique

MÉMOIRE DE MASTER

Sciences et Technologies
Electronique
Systèmes Embarqué

Réf. : .....

Utilisation FPGA/VHDL pour la gestion et le contrôle des données

Le : .....

Présenté par :

DJOUDI Amina
HARABI Soumaia

Avis favorable de l'encadreur :

DHIABI Fathi

Signature Avis favorable du Président du Jury

Cachet et signature

Université Mohamed Khider de Biskra
Faculté des Sciences et de la Technologie
Département de génie électrique

MÉMOIRE DE MASTER

Sciences et Technologies
Electronique
Systèmes Embarqué

Utilisation FPGA/VHDL pour la gestion et le contrôle des données

Proposé par : DHIABI Fathi
Dirigé par : DHIABI Fathi

ملخص

تناول هذه المذكرة تصميم وتنفيذ نظام ذكي لإدارة مواقف السيارات باستخدام تقنية FPGA وبرمجة VHDL بهدف تحسين كفاءة عمليات ركن السيارات.يتناول الفصل الأول لغة VHDL، التي تُعد أداة أساسية لوصف وتصميم الدوائر الرقمية، مع شرح مفصل للمفاهيم الأساسية والقواعد البرمجية وطرق المحاكاة. يركز الفصل الثاني على تقنية FPGA، مع توضيح بنية FPGA ومميزاتها. يُقدم الفصل الثالث التطبيق العملي للنظام باستخدام برنامج Quartus II من شركة Altera ولوحة Cyclone II، ويشمل ذلك مراحل التصميم والبرمجة والمحاكاة والتنفيذ على العتاد. تم الجمع بين الجانب النظري والتطبيقي لإظهار فعالية استخدام FPGA وVHDL في تطوير أنظمة رقمية متقدمة وذات أداء عالي.

Summary

This thesis focuses on the design and implementation of an intelligent parking management system using FPGA technology and VHDL programming, aiming to enhance the efficiency and automation of vehicle parking operations.The first chapter introduces the VHDL language, which is a fundamental tool for describing and designing digital circuits. It provides adetailed explanation of core concepts, syntax rules, and simulation techniques.The second chapter centers on FPGA technology, explaining its architecture and advantages.The third chapter presents the practical implementation of the system using Quartus II software from Altera and the Cyclone II development board. It covers all stages of design, coding, simulation, and hardware implementation.The thesis successfully combines both theoretical and practical aspects to demonstrate the effectiveness of using FPGA and VHDL in the development of advanced and high-performance digital systems.

شكر وعرفان

نحمد الله تعالى اولا واخيرا حمدًا, يليق بجلاله وعظيم سلطانه، لما منّا به عليّنا من عونٍ وتوفيق، ومنحنا القوة والعزيمة والإيمان لإتمام هذا العمل المتواضع.

نتوجّه بخالص الشكر والتقدير إلى أستاذنا المشرف السيد ذيابي فتحي، الذي لم يبخل عليّنا بدعمه وتوجيهه السديد، وكان لعطائه المتواصل وملاحظاته الهادفة ودعمه المستمر الأثر البالغ في إنجاز هذا المشروع.

كما نتقدّم بجزيل الشكر والعرفان لأعضاء لجنة المناقشة، الأستاذ بالمير عقبة والأستاذة طبش سعاد، على قبولهم تقييم هذا العمل، وما تفضلوا به من ملاحظات قيّمة وجهتنا نحو الأفضل.

ولا يفوتنا أن نعبّر عن امتناننا لكل من ساندنا أو قدّم لنا يد العون، سواء بكلمة، أ و توجيه، أو دعم، فلكم منا جميعاً خالص الشكر والتقدير.

سمية حرابي

امينة جودي

الاهداء

(وَآخِرُ دَعْوَاهُمْ أَنِ الْحَمْدُ للهِ رَبِّ الْعَالَمِينَ)

الحمد الله حبا وشكرا وامتنانا ما كنت لأفعل هذا لولا فضل الله فالحمد لله على البدء والختام

ها أنا اليوم أهدي نجاحي إلى نفسي أولا ثم لكل من سعى معي لإتمام هذه المسيرة.

إلى الذي علمني أن الدنيا كفاح وسلاحها العلم والمعرفة ، إلى من أحمل اسمه بكل افتخار إلى أعظم أعز رجل في الكون أبي الغالي محمد.

إلى من جعل الله الجنة تحت أقدامها وسهلت لي الشدائد بدعائها ، إلى الإنسانة العظيمة التي لطالما تمنت أن تقر عينها لرؤيتي في يوم كهذا أمي العزيزة.

إلى ضلعي الثابت وأمان أيامي، إلى ملهمي نجاحي ... إلى خيرة أيامي وصفوتها ، إلى قرة عيني أخواني الغاليين حسين، خالد، عمر وأختي فاطمة الزهراء.

إلى نفسي المثابرة الطموحة

الى من كانو عونا وسندا في هذا الطريق ..... إلى صديقات المواقف لا السنيين شريكات الدرب الطويل

نرجس، سمية، سندس، سلسبيل، ليليا، رميسة، بشرى وشيماء

أهديكم هذا الإنجاز وثمرة نجاحي الذي لطالما تمنيته،

ها أنا اليوم أتممت أول ثمراته بفضل من الله عز وجل

فالحمد لله على ما وهبتني وأن يعينني ويجعلني مباركة أينما كنت

امينة جودي

الاهداء

(وَآخِرُ دَعْوَاهُمْ أَنِ الْحَمْدُ للهِ رَبِّ الْعَالَمِينَ)

الحمد لله حبا وشكرا وامتنانا على البدء والختام اهدي هذا المشروع لنفسي اولا بعد مسيرة دراسية حملت في طياتها الكثير من الصعوبات والمشقة والتعب ها انا اليوم اقف على عتبه تخرجي اقطف ثمار تعبي وارفع قبعتي بكل فخر.

الى التي جعل الجنة تحت اقدامها الى من اضاءت في ليالي العتمة طريقي الى من ضلت دعواتها تضم إسمي. ((أمي الغالية كريمة)).

الى من كلل العرق جبينه وعلمني ان النجاح لا يأتي إلا بالصبر وبالإصرار الى من لا ينفصل اسمي عن اسمه الى من كان لي مصدر الدعم والعطاء إلى داعمي الاول وسندي وقوتي بعد الله إلى فخري واعتزازي ((والدي الغالي محمد))

الى الأرواح التي تتحدى الموت يوميا لتبقى على الامل حيا ...الى غزة اهديكم فرحة تخرجي ,علها تحمل معها دعاء بالنصر والفرج القريب

الى ضلعي الثابت وامان ايامي الى من شددت عضدي بهم الى خيرة ايامي وصفوتها

اخواتي سندس , وسام , مريم فاطمة الزهراء ورقية

واخوتي مصعب, صهيب, زيد ونوح

الى صديقات المواقف لا السنين، شريكات الدرب الطويل

من كانوا في سنوات العجاف سحابا ممطر امينة

نرجس, اميمة, بشرى, شيماء, قدس, هيبة و رميسة احبكم

و ممتنة لوجودكم .

سمية حراب

الفهرس

الفصل الثالث: انشاء مشروع لادارة البيانات باستعمال FPGA مع VHDL

قائمة الاشكال والجداول

قائمة الاختصارات

A

ASIC: Application Specific Integrated Circuit.

C

CLB: Configurable Logic Block.

CISC: Complex Instructions Set Construction.

CPU: Central Processing Unit.

CPLD: Complex Programmable Logic Device.

D

DSP: Digital Signal Processor.

E

EPROM: Erasable Programmable Read Only Memory.

EEPROM: Electrically-Erasable Programmable Read Only Memory.

F

FPGA: Field Programmable Gate Arrays.

I

IEEE: Institut of Electrical and Electronics Engineers.

IOB: Input Output Block.

J

JTAG: Joint Test Action Group.

L

LCA: Logic Cell Array.

LC: Logic Cell.

LE: Logic Element.

LUT: Look-Up-Table.

P

PLD: Programmable Logic Device..

R

RAM: Random Access Memoy.

ROM: Read Only Memory.

RISC: Reduced Instructions Set Construction.

RTL: Register Transfert Language.

S

SRAM: Static Random Access Memory.

UAL: Unité Arithmétique et Logique.

UC: Unité de Contrôle.

V

VHSIC: Very High Scale Integrated Circuit.

VLSI: Very Large Scale Integration.

المقدمة العامة

المقدمة العامة

في عصرنا الحالي، تشهد التكنولوجيا الرقمية تطورًا مذهلاً، حيث تلعب الأنظمة الرقمية دورًا محوريًا في كل جانب من جوانب حياتنا، من الأجهزة المنزلية إلى أنظمة الاتصالات، مرورًا بالسيارات الذكية وأنظمة التحكم الصناعي. ويأتي من بين أهم هذه التقنيات تقنيات التصميم الرقمي القابلة للبرمجة، والتي مكنت المهندسين والمصممين من تطوير حلول مرنة وقابلة للتكيف مع متطلبات التطبيقات المختلفة دون الحاجة إلى تصنيع أجهزة جديدة.

تعتبر لغة VHDL (VHSIC Hardware Description Language) من أبرز الأدوات البرمجية في هذا المجال، حيث تتيح توصيف وتصميم الأنظمة الرقمية بطريقة منهجية ودقيقة، وتُستخدم على نطاق واسع في تصميم دوائر متكاملة قابلة للبرمجة. تتيح هذه اللغة إمكانية كتابة كود يعبر عن التصميم الرقمي بطريقة تسمح بمحاكاته واختباره قبل تنفيذه فعليًا، مما يقلل من الأخطاء ويوفر الوقت والتكلفة في عملية التطوير.

في الجانب الآخر، تمثل تقنية FPGA (Field Programmable Gate Array) نقلة نوعية في تصميم الأنظمة الرقمية، فهي شرائح إلكترونية قابلة للبرمجة تسمح بإعادة تكوين الدوائر الرقمية حسب الحاجة، وتتميز بسرعتها العالية، ومرونتها، وقابليتها للتعديل بعد التصنيع. سمحت هذه التقنية بابتكار أنظمة متطورة تلبي متطلبات التطبيقات المتغيرة والمتنوعة في مجالات عديدة مثل الاتصالات، التحكم، السيارات، والأنظمة الذكية.

انطلاقًا من أهمية هذه التقنيات ودورها الحيوي في تطوير أنظمة ذكية وقابلة للتكيف، تناولت هذه المذكرة موضوع تصميم وتنفيذ نظام عملي متكامل لإدارة مواقف السيارات باستخدام لغة VHDL على تقنية FPGA. حيث أن مواقف السيارات تمثل مشكلة متزايدة في المدن الحديثة، ويتطلب إدارتها حلولًا ذكية تضمن الاستخدام الأمثل للمساحات المتاحة وتوفير الوقت والجهد على المستخدمين. تتكون هذه المذكرة من ثلاثة فصول متكاملة:

- **الفصل الأول:** يركز على لغة VHDL، حيث يتم تقديمها بشكل مفصل كأداة توصيفية تستخدم في تصميم الأنظمة الرقمية. يتناول هذا الفصل أساسيات اللغة، قواعد الكتابة، أنواع البيانات، الهياكل البرمجية، وأدوات المحاكاة التي تمكن من التحقق من صحة التصميم قبل التنفيذ.

- **الفصل الثاني:** يستعرض تقنية FPGA، ويشمل شرحًا مفصلًا لبنية هذه الشرائح، مكوناتها الداخلية، آلية عملها، بالإضافة إلى المزايا التي تجعلها خيارًا مفضلًا في تصميم الأنظمة الرقمية مقارنةً بالتقنيات الأخرى. كما يعرض هذا الفصل التطبيقات العملية لتقنية FPGA في الصناعة والبحث العلمي.

- **الفصل الثالث:** يقدم التطبيق العملي للمذكرة، حيث تم تصميم وتنفيذ نظام ذكي لإدارة مواقف السيارات. في هذا الفصل تم استخدام برنامج **Quartus II** من شركة **Altera** (التي أصبحت جزءًا من شركة Intel) لتطوير وتصميم المشروع. ويعتمد البرنامج على عائلة الشرائح Cyclone II ، وهي من الشرائح الاقتصادية ذات الأداء الجيد التي تناسب التطبيقات التعليمية والتطبيقات العملية البسيطة والمتوسطة. يتضمن الفصل مراحل التصميم من تحديد المتطلبات، كتابة كود VHDL للنظام، وإجراء المحاكاة باستخدام أدوات البرنامج للتأكد من صحة الأداء الوظيفي للتصميم. لم يتم الانتقال إلى مرحلة تحميل التصميم على لوحة FPGA، حيث تم الاكتفاء بالتحقق من صحة العمل من خلال بيئة المحاكاة فقط. يعرض الفصل تفاصيل العمل البرمجي والعتادي، كما يناقش النتائج والتحديات التي واجهت تنفيذ المشروع والحلول التي تم اعتمادها.

تهدف هذه المذكرة إلى دمج الجانب النظري مع الجانب التطبيقي، لتوفير فهم عميق وشامل لتقنيات التصميم الرقمي الحديثة، وإبراز كيفية توظيفها لحل مشكلات واقعية بفعالية. كما تسعى إلى تنمية مهارات التصميم، البرمجة، والتحليل لدى الطالب، وتجهيزه للعمل في مجالات متقدمة تتطلب استخدام أنظمة FPGA ولغات توصيف العتاد مثل VHDL.

بهذا الأساس، تمثل هذه المذكرة خطوة مهمة في مسيرة التعلم والتطوير المهني، وتأمل في أن تفتح آفاقًا جديدة نحو مشاريع مستقبلية أكثر تعقيدًا وابتكارًا في مجال الأنظمة الرقمية والأنظمة المدمجة.

الفصل الأول

مفاهيم حول لغة وصف الأجهزة

(VHSIC Hardware Description Langage)

V.H.D.L

1.I. مقدمة :

لغة VHDL، وهي اختصار لـ VHSIC Hardware Description Language، تُعد واحدة من أهم لغات وصف العتاد الرقمي في عالم الإلكترونيات والأنظمة الرقمية. تم تطوير هذه اللغة في أوائل الثمانينيات من قبل وزارة الدفاع الأمريكية كجزء من مشروع تطوير دارات متكاملة عالية السرعة (VHSIC)، وكان الهدف الرئيسي منها هو توفير وسيلة دقيقة وموحدة لتوثيق وتصميم الدوائر الإلكترونية المعقدة.

تتميز VHDL بكونها لغة وصف سلوكي وبنيوي، أي أنها تتيح للمصمم أن يحدد كيف يجب أن يعمل النظام (من خلال وصف السلوك) وكذلك كيف يتكون هذا النظام من مكونات فرعية مترابطة (من خلال وصف البنية). وهذا ما يجعلها أداة قوية في تصميم النظم الرقمية بدءًا من أبسط البوابات المنطقية وحتى المعالجات المعقدة

أحد أهم استخدامات VHDL هو في برمجة شرائح FPGA (Field Programmable Gate Arrays)، حيث يمكن استخدام الكود المكتوب بلغة VHDL لوصف وظائف معينة يتم تحميلها على الشريحة، لتصبح الشريحة بعد ذلك تعمل تمامًا كدائرة مصممة خصيصًا لهذا الغرض. كما تُستخدم VHDL أيضًا في تصميم رقائق ASIC، وهي دوائر متكاملة يتم تصنيعها لتأدية وظائف محددة بشكل دائم

من مزايا VHDL أنها تدعم المحاكاة والتحقق من صحة التصميم قبل تحويله إلى شكل مادي، مما يقلل من الأخطاء ويوفر الوقت والتكلفة. كما أن طبيعتها المعيارية والمعتمدة من قبل IEEE جعلت منها لغة موثوقة وقابلة للاستخدام في مختلف البيئات الصناعية والأكاديمية

تُستخدم VHDL أيضًا في تعليمنا الجامعي كمادة أساسية ، حيث تتيح لنا كطلاب تعلم مفاهيم التصميم الرقمي بشكل عملي ومحاكاة عمل الأنظمة قبل تطبيقها فعليًا.[1]

2.I. نبذة تاريخية على ال VHDL:

في ثمانينات القرن الماضي، وفي إطار مبادرة تطوير الدوائر المتكاملة عالية السرعة (VHSIC)، اطلقت وزارة الدفاع الأمريكية مناقصة لإنشاء لغة وصف عتاد موحدة تسمح بوصف جميع الأنظمة الإلكترونية المستخدمة.

وفي عام 1983، شاركت شركات كبرى مثل IBM وIntermetrics وTexas Instruments في هذا المشروع. وفي عام 1985، تم تطوير أول نسخة رسمية من لغة VHDL (الإصدار 7.2). ثم في عام 1986، تم تسليم VHDL إلى معهد مهندسي الكهرباء والإلكترونيات (IEEE) من أجل توحيدها، وتم اعتماد معيار VHDL تحت الرقم IEEE 1076 في 10 ديسمبر 1987.

وفي عام 1993، قام معهد IEEE بوضع معيار IEEE 1164 الذي يُنظّم تمثيل الإشارات المنطقية متعددة القيم. أما آخر تطور في المعيار فقد كان IEEE 1076-2001.[2] [3] [4] [5]

3.I. مزايا لغة VHDL:

تُعتبر لغة VHDL:

- **لغة شاملة :** منذ بداياتها، كان أحد الأهداف الرئيسية للـ VHDL هو تغطية مختلف مراحل تصميم الأنظمة الرقمية: التحديد، النمذجة، المحاكاة والتخليق؛ أي نظام موحد لجميع هذه المراحل.

- **لغة مستقلة :** باعتبارها معيارًا تابعًا لـ IEEE، فهي لا تعود لأي جهة بعينها تحتكر تطويرها أو استخدامها. كما أن استقلاليتها عن المعماريات والتقنيات تامة، حيث إن استخدامها لا يقتصر على فئة معينة من المعماريات أو على تقنيات محددة.

- **لغة مرنة :** تتيح VHDL للمستخدم إجراء الأوصاف على مستويات مختلفة من التجريد: المستوى السلوكي، والمستوى الوسيط باستخدام المعادلات المنطقية كمثال، والمستوى التركيبي باستخدام الربط بين البوابات المنطقية الأساسية. كما توفر مرونة في تركيب الأوصاف مع أو بدون تسلسل هرمي، حسب رغبة المستخدم.

- **لغة حديثة :** تتميز بصياغة سهلة القراءة، وغالبًا ما تُغني عن الحاجة إلى التعليقات التوضيحية. كما أنها لغة قوية في تحديد الأنواع وتدقيقها، مما يقلل من مخاطر الأخطاء وانتشارها بشكل خفي، ويعزز الجودة.

- **لغة معيارية :** وهو ما يضمن التوافقية، والثبات، والاستمرارية. تم اعتماد VHDLكالغة معيارية من قبل IEEE، حيث أُصدرت أول نسخة معيارية سنة 1987، وتبعتها تحديثات في 1993، 2000، 2002، و2008الى يومنا هذا.

- **لغة مفتوحة :** تدعم VHDL آليات الوظائف والإجراءات التي تتيح للمصممين توسيع قدرات اللغة بكل سهولة مع الحفاظ على قابلية النقل.[3]

4.I. عيوب لغة VHDL:

- **تعقيد اللغة وطول الكود:** لغة VHDL معقدة نسبياً وتتطلب كتابة كود مطوّل حتى للوظائف البسيطة، ما يجعل عملية التطوير أبطأ مقارنة بلغات أخرى مثل Verilog.
- **منحنى تعليمي حاد :** تحتاج VHDL إلى تعلم عميق، خاصة لمن ليست لديهم خبرة سابقة في لغات البرمجة أو تصميم العتاد، مما يصعّب دخول المبتدئين إليها.
- **صرامة بناء الجملة :** بناء الجملة في VHDL صارم جداً، ما يزيد من احتمالية الوقوع في أخطاء أثناء الكتابة ويجعل تصحيحها يستغرق وقتاً أطول.
- **تفصيلية مفرطة :** تتطلب VHDL توصيفاً دقيقاً ومفصلاً للأنظمة، وهذا قد يكون مرهقاً في التصاميم الصغيرة والبسيطة.

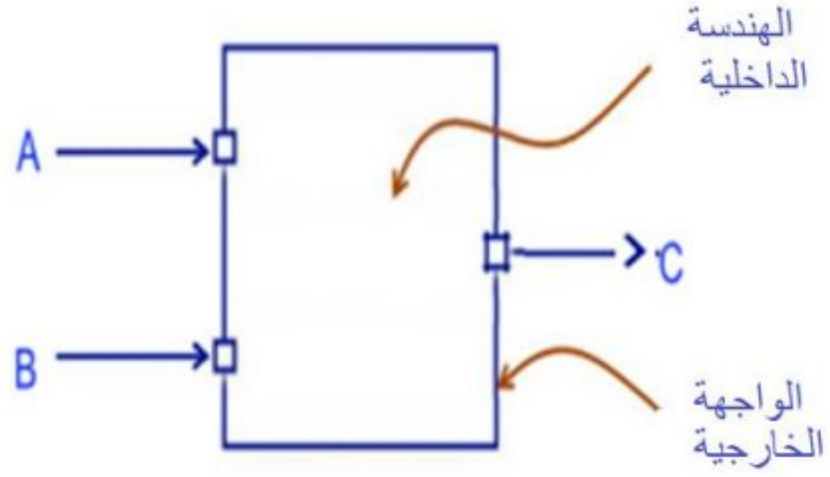
- **أوقات تطوير أطول :** الطبيعة المفصلة للغة تؤدي غالباً إلى زيادة زمن التطوير مقارنة بلغات توصيف العتاد الأخرى.[6]

5.I. وصف بسيط لبنية الVHDL:

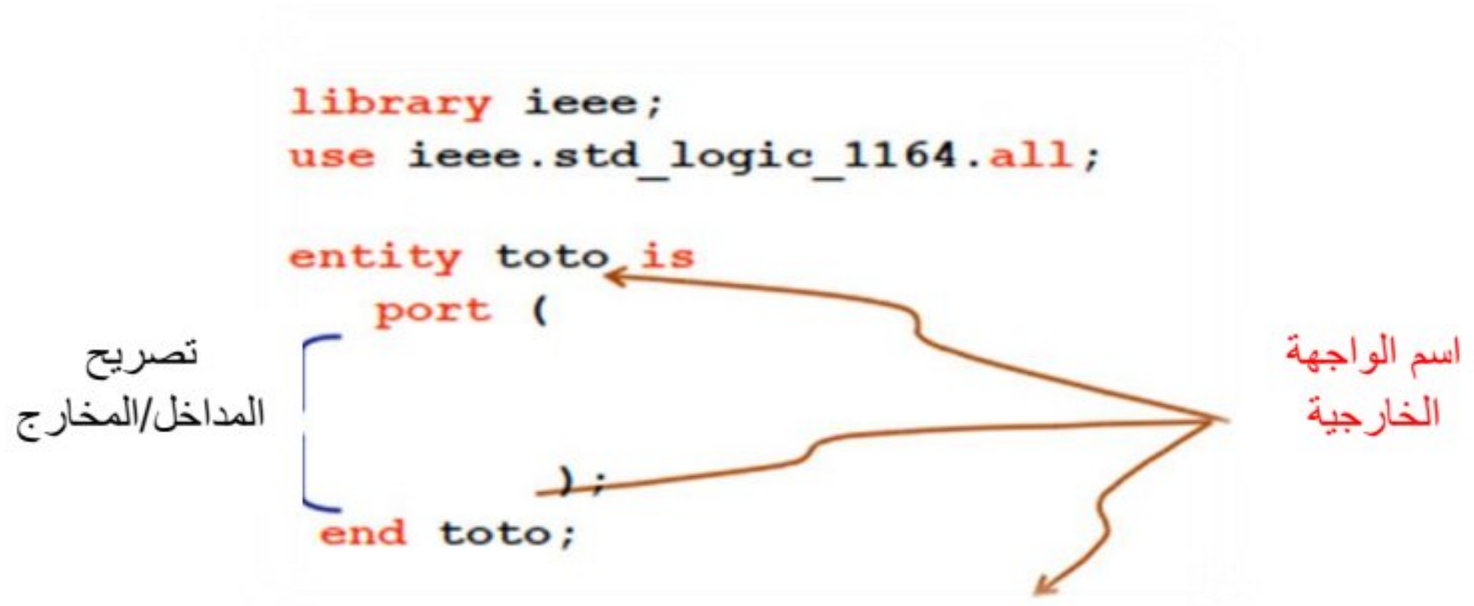
مشغل أولي، أو دائرة متكاملة، أو لوحة إلكترونية، أو نظام كامل يتم تعريفه بالكامل من خلال إشارات الإدخال والإخراج، ومن خلال الوظيفة المنفذة داخليًا. هذان الجانبان - إشارات الاتصال والوظيفة - يتكرران على جميع مستويات التسلسل الهرمي لتطبيق ما. العنصر الأساسي في أي وصف بلغةVHDL ، والذي يُطلق عليه اسم **كيان التصميم** (design entity) في اللغة، يتكون من زوج الواجهة الخارجية - هندسة النظام، والذي يصف المظهر الخارجي لوحدة التصميم وكذلك طريقة عملها الداخلية.

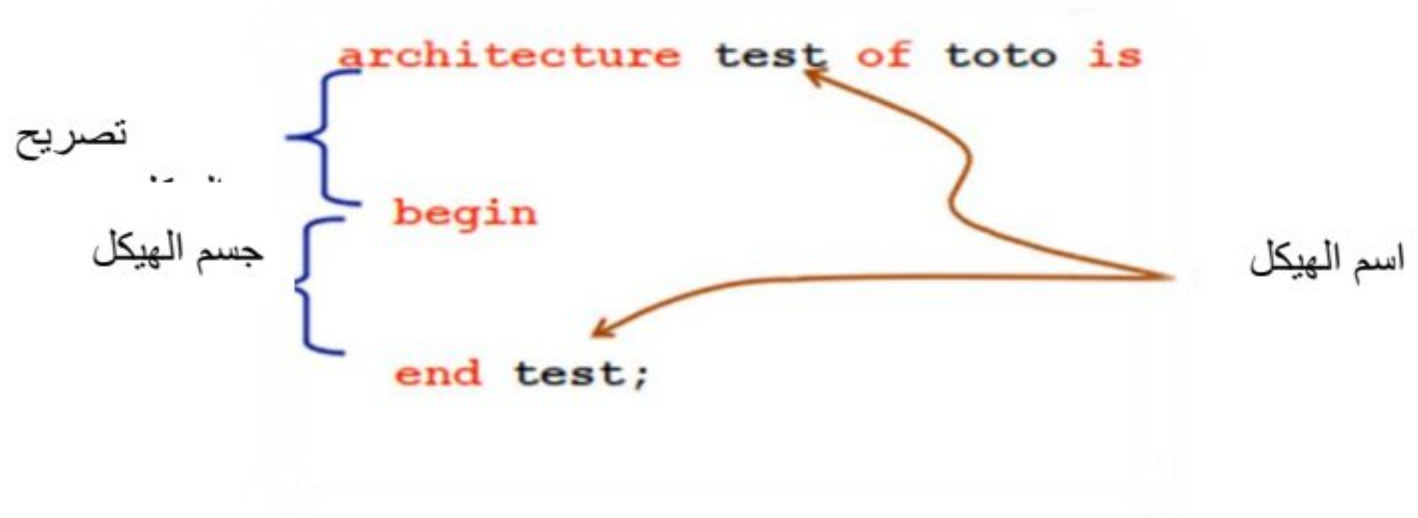
- **الواجهة الخارجية :** يصف الواجهة الخارجية للدائرة، أي إشارات الإدخال والإخراج.

- **هندسة النظام :** تصف الوظيفة الداخلية للدائرة. وهي دائمًا مرتبطة بواجهة خارجية معيّنة. ويمكن لواجهة واحدة أن تمتلك عدة معماريات.[9] [3]



الشكل 01.I: الهندسة الداخلية والواجهة الخاريجية [7]





الشكل I.02: بنية برنامج VHDL [7]

I.1.5. التصريح بالمكتبات:

كل وصف بالغة VHDL يُستخدم من أجل التوليف (synthèse) يحتاج إلى مكتبات. لقد قامت منظمة IEEE (المعهد الأمريكي لمهندسي الكهرباء والإلكترونيات) بتوحيد هذه المكتبات وخصوصًا المكتبة IEEE1164.

تحتوي هذه المكتبات على تعريفات لأنواع الإشارات الإلكترونية ، بالإضافة إلى الدوال والبرامج الفرعية التي تتيح تنفيذ العمليات الحسابية والمنطقية

-Library ieee ;	تصريح قياسي
-Use ieee.std_logic_1164.all;	لتعريف الأنواع std_logic ('X','U','L','H','0','1')
-Use ieee.std_logic_arith.all;	لاستخدام العمليات الحسابية'+'، '-'، '*'، '/' مع **std_logic** الأنواع
Use ieee.std_logic_signed.all;	النوع **std_logic** مع قيمة موقعة (ذات اشارة)
-Use ieee.std_logic_usigned.all;	النوع **std_logic** مع قيمة غير موقعة (ذات اشارة)
-Use ieee.numeric_std. all;	استخدام القيم العشرية مع النوع **std_logic**
-Use ieee.std_logic_textio.all;	استخدام القيم **ASCII مع النوع std_logic**
-Use ieee.numeric_bit.all;	استخدام القيم العشرية لنوع **std_logic**
-Use ieee.math_real.all;	
- Use ieee.math_complex.all;	

الجدول I.01: مكتبات ومعايير IEEE في لغة VHDL

ملاحظة:

IEEE 1164: هومعيارالقيم المنطقية المتعددة الذي نشرته منظمة IEEEبعام 1993.

IEEE 1076: هو معيار لغة VHDL الذي نشرته منظمة IEEE في عام 1987.

1/ تسمح التعليمة Use باختيارالمكتبات التي سيتم استخدامها.

2/ تسمح التعليمة All باستخدام جميع التطبيقات.

3/ الفرق بين Signed وUnsigned (حيث يُمثل المتجه عدداً إما بإشارة أو بدون إشارة).[4] [8]

2.5.I. تصريح الواجهة الخارجية:

تصف لواجهة الخارجية الواجهة بين العالم الخارجي ووحدة التصميم ويشمل:

إشارات الإدخال والإخراج ،وكذلك المعاملات العامة (وهي ثوابت يتم تحديد قيمتها من قبل بيئة الوحدة المعنية).

يمكن أن يحتوي تصريح الواجهة الخارجية أيضًا على تعليمات متزامنة سلبية أي تلك التي لا تحتوي على أي إسناد لإشارات؛ مثل هذه التعليمات لا تُنتج دوائر أثناء عملية التوليف (synthesis).يمكن ربط واجهة خارجية واحدة بعدة هياكل معمارية مختلفة. وبهذا فإنه يصف فئة من وحدات التصميم التي تقدم نفس الواجهة للعالم الخارجي ،لكن تختلف داخليًا في طريقة عملها.[9]

تحدد الواجهة الخارجية ما يلي:

- اسم الدائرة.
- منافذ الإدخال / الإخراج :

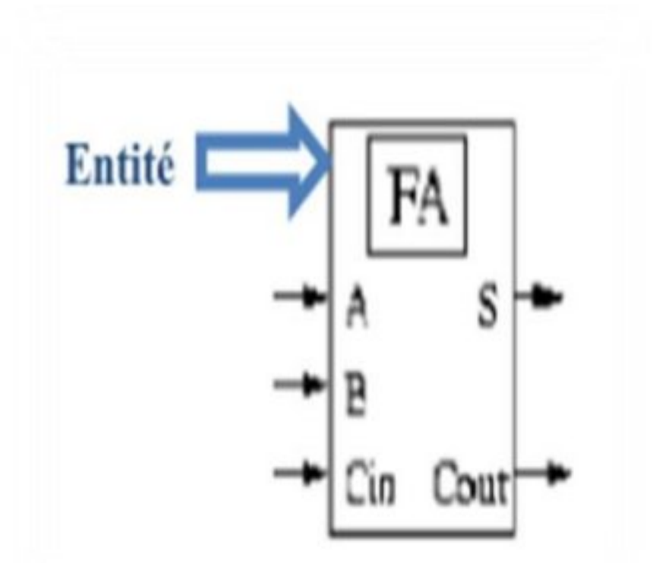
* أسماؤها.

* اتجاهاتها (in،out،inout، ...)

* أنواعها (bit،bit_vector،integer،std_logic، ...)

- المعاملات المحتملة للنماذج العامة (generic parameters).[10]

مثال على صيغة تصريح الواجهة الخارجية :




```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity FA is
port (
a, b, cin : in std_logic;
s, cout : out std_logic
);
end FA;
```

الشكل 03.I : تمثيل تخطيطي للواجهة الخارجية [10]

;

1.2.5.I. المنافذ:

تشكل إشارات الواجهة لواجهة خاريجية ما مايُعرف في مصطلحات VHDL باسم "منفذ" (Port). يجب أن تمتلك كل إشارة في المنفذ اسمًا ،ووضعًا (mode)، ونوعًا (type).

1.1.2.5.I. اسم الإشارة :

يتم اختيار اسم كل إشارة من قبل المستخدم، ويُعرف هذا الاسم داخل جميع الهياكل المعمارية التي تشير إلى الواجهة الخاريجية المقابلة. يتكون الاسم من سلسلة من الأحرف الأبجدية الرقمية تبدأ بحرف، VHDL لا يميز بين الحروف الكبيرة والصغيرة، لذا فإن AmI و aMi يُمثلان نفس الكائن.[3] [9]

2.1.2.5.I. وضع الاشارة :

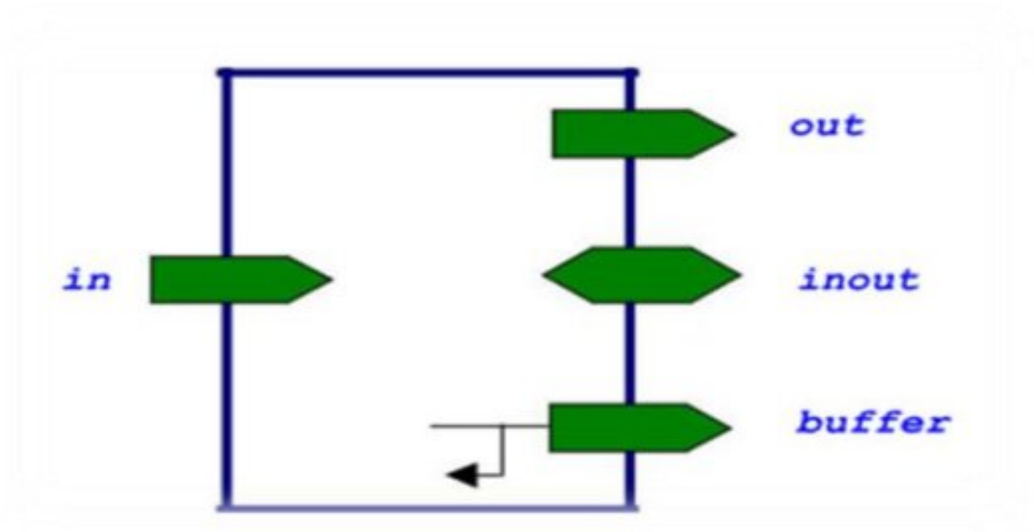
الوضع يحدد اتجاه الإشارة:

In: مدخل الاشارة .

Out:مخرج الاشارة .

Inout: مدخل ومخرج الاشارة .

buffer: اشارة متحكم فيها .



الشكل 04.I :الأوضاع الأربعة في VHDL [7]

3.1.2.5.I. نوع الاشارة :

النوع المستخدم لإشارات الإدخال/الإخراج هو:

- الـ **std_logic** لقيمة واحدة.
- الـ **std_logic_vector** لمصفوفة.

القيم التي يمكن أن تأخذها إشارة من نوع **std_logic**و **std_logic_vector** هي:

- 0:مستوى منخفض (0V)
- 1: مستوى مرتفع (1V)
- Z: مستوى منطقي عائم (إدخال مفصول)
- L: مستوى منطقي منخفض (جهد منخفض مثال 5v-)
- H:مستوى منطقي مرتفع (جهد مرتفع مثال 5v+)
- X:مستوى منطقي غير معروف (قد يكون '0' أو 'L' أو 'Z' أو 'H' أو '1')
- U: غير مُعرف.[11]

3.5.I. التصريح الهيكلي (الواجهة الداخلية):

الهيكلة تصف الوظيفة المطلوبة لدائرة منطقية. يتم وصف وظيفة الدائرة بواسطة تعليمات من ال VHDL .

هذه التعليمات هي العلاقة بين المداخل والمخارج التي تم وصفها في تصريح الواجهة الخارجية . وتكون على شكل وظيفتين الوظيفة التوافقية ، أو الوظيفية التتابعية، أو كلاهما معًا (التتابعية والتوافقية).[4]

الهيكلة مقسمة إلى جزئين: منطقة تعريفات ومنطقة تعليمات. في حالة التعليمات كتبت على التوازي فلا يهم ترتيب التعليمات داخل الهيكل ، ولا يعتمد الأداء على هذا الترتيب.[9]

يمكن استخدام ثلاثة أنماط للوصف في VHDL :

- النمط "الهيكلي": هو ربط المكونات من عدة هياكل اخرى .
- النمط "السلوكي": مجموعة من العمليات التي تعبر عن سلوك النظام .[13]
- تعريف الهيكلة يكون بالشكل التالي :

```
architecture nom_de_l_architecture of nom_de_l_entité is
    déclarations
begin
    instructions-concurrentes
end nom_de_l_architecture;
```

6.I. المعاملات في اللغة :

يسرد الجدول أدناه العوامل القياسية وأنواع المعاملات التي يتم التعامل معها .[3]

الدلالة	نوع المعامل		تصنيف
Et Non-et Ou Non-ou Ou exclusif Non ou exclusif Non	boolean bit ou bit_vector__		العوامل المنطقية and nand or nor xor xnor not
Egal Non égal Inférieur Inférieur ou égal Supérieur Supérieur ou égal	المدخل Tout type scalaire	النتيجة boolean	العمليات العلائقية = /= < <= > >=
+ unaire (signe +) ou addition - unaire (signe -) ou soustraction Multiplication Division Valeur absolue Exponentiation Modulo Reste	Integer, real integer		العمليات الحسابية + - * / Abs ** Mod rem
Logique gauche Logique droit Circulaire gauche Circulaire droi	bit_vector (amplitude : integer)		عمليات الازاحة Sll Srl Rol Ror
Concatenation	Bit, bit_vector	bit_vector	انواع اخرى &

الجدول 02.I: معاملات VHDL وانواعها [3]

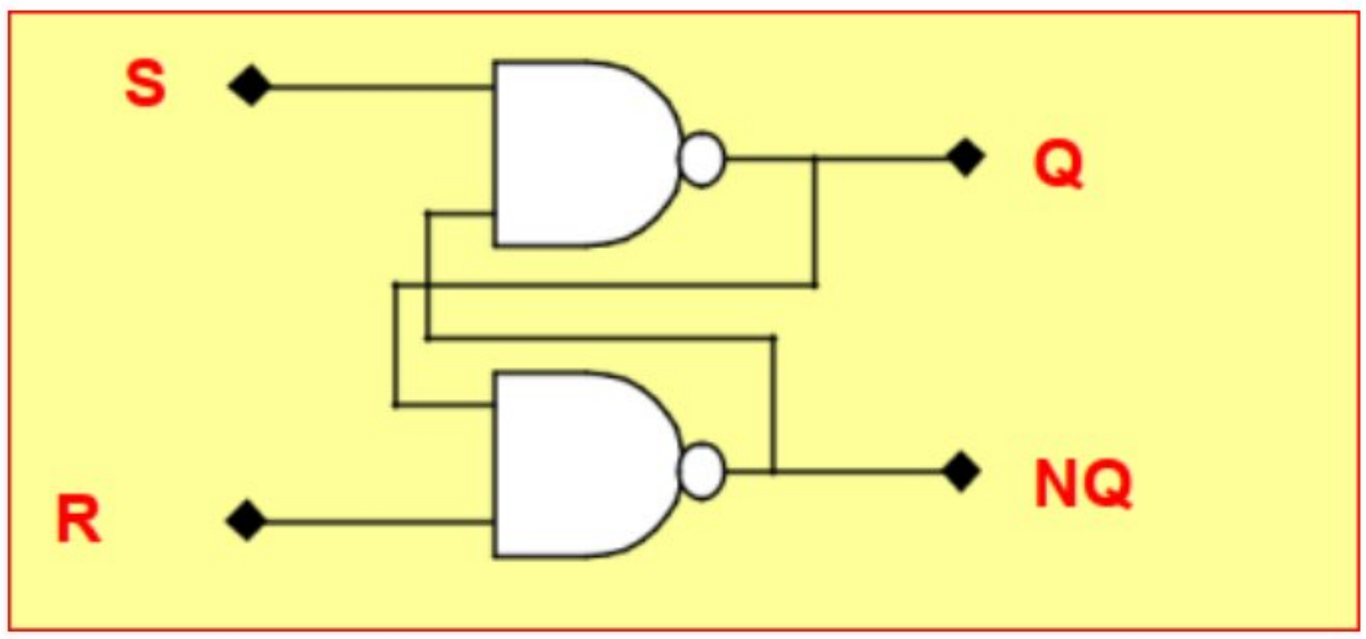
7.I. عمليات بتوازي و تسلسل :

1.7.I: عمل بتوازي (التفرع) :

يمكن وصف سلوك الدائرة المنطقية بمجموعة من العمليات التي تُنفذ بشكل متوازٍ. ولهذا السبب، يوفر VHDL مجموعة من التعليمات التي تُعرف بالتعليمات المتزامنة

التعليمة المتزامنة هي تعليمة يتم تنفيذها بشكل مستقل عن ترتيب ظهورها في شفرةVHDL

على سبيل المثال، نأخذ حالة جهاز Latch بسيط، كما توضح الشكل:



الشكل 05.I : دارة لاقط SR باستخدام بوابتين NAND [7]

البابان اللذان يشكلان هذا الجهاز Latch يعملان بشكل متوازٍ. يتم إعطاء وصف ممكن لهذا الدائرة في الكود التالي

(حيث يتم إعطاء الهيكل فقط) :

```
architecture comportement of VERROU is
begin
Q<= S nand NQ;
NQ <= R nand Q;
end comportement;
```

هاتان التعليمتان تنفذان في نفس الوقت هما على التوازي ؛ ترتيب كتابتهما ليس له أهمية؛ بغض النظر عن ترتيب هاتين التعليمتين، فإن الوصف يبقى كما هو.[13]

2.7.I. العمل التسلسلي:

في VHDL لا تُستخدم التعليمات التسلسلية إلا داخل العمليات (process) .

1.2.7.I. تعريف عملية (Process) :

العملية هي جزء من وصف االدائرة المنطقية يتم فيها تنفيذ التعليمات بشكل تسلسلي، أي واحدة تلو الأخرى (على عكس التعليمات المتزامنة). وتتيح إجراء العمليات على الإشارات باستخدام التعليمات القياسية للبرمجة الهيكلية كما في الأنظمة المعتمدة على المعالجات الدقيقة.

الصيغة:

```
[Nom_du_process :] process (Liste_de_sensibilité_nom_des_signaux)
begin
-- instructions du process
end process [Nom_du_process] ;[8]
```

2.2.7.I. قواعد عمل العملية:

1)- يتم تشغيل العملية نتيجة لتغيير أو تغييرات في حالات الإشارات المنطقية. يتم تعريف أسماء هذه الإشارات في قائمة الحساسية عند إعلان العملية.

2)- تنفذ تعليمات العملية بشكل تسلسلي

3)- التعديلات التي تطرأ على قيم الإشارات بواسطة التعليمات تصبح سارية المفعول في نهاية العملية. [8] [3]

المثال التالي يوضح هيكل (فقط) لجزء دالة D يحتوي على عملية تُنفذ عند تغيير حالة الساعة CLK. [13]

```
architecture comportement of basc_D is
 begin
Process (CLK)
Begin
If ( CLK= ‘1’) then Q <= D;
End if ;
End process ;
end comportement;
```

8.I. تعليمات :

1.8.I. تعليمات في وضع التوازي:

1.1.8.I. التخصيص غير المشروط:

شكله العام:

```
signal <= expression;
```

I.8.1.2. التخصيص المشروط:

التخصيص المشروط يظهر على النحو التالي:

```
signal <= exp1 when condition1 else
exp2 when condition2 else
....
expN-1 when conditionN-1 else expN;
```

I.8.1.3. التخصيص الانتقائي:

بالصيغة التالية:

```
with expression select
 signal <= exp1 when choix1,
                exp2 when choix2,
                ....
  expN when choixN;
```

I.8.1.4. إنشاء نسخة من المكون (تهيئة المكون):

يتكون من استخدام مجموعة فرعية موصوفة بلغة VHDL كمكون في مجموعة أوسع.

يتم إنشاء مثيل المكون في جسم البنية المعمارية بهذه الطريقة:

```
<nom_instance>:<nom_composant>port map (liste des connexions);
```
[14]

I.8.2. تعليمة في الوضع التسلسلي:

I.8.2.1. تخصيص غير مشروط للمتغير:

الصيغة العامة:

```
var := expression;
```

I.8.2.2. تعليمة الانتظار:

تعليمة "انتظار" توقف تنفيذ عملية حتى يحدث حدث أو شرط انتهاء الوقت (time out) إذا لم يتم تحديد شرط استيقاظ، فإن العملية تتوقف نهائيًا .[3]

- يمكن استخدام "انتظار" بالطرق التالية:

wait on [**nomSignal1, nomSignal2…**] ;

wait until [**expressionBooléenne**] ;

wait for [**expressionTemps**] ;

I.3.2.8. التعليميات الشرطية:

يمكن التعبير عنها باستخدام التعليميات if و case :

- صيغة if :

if condition1 **then** instructions

[**elsif** condition2 **then** instructions]

. . .

[**else** instructions]

end if ;

من الممكن تداخل عدة تعليمات if داخل بعضها البعض:

if ... then

if ... then

elsif ... then

end if;

else

end if;

- صيغة عبارة **case** :

case signal_de_slection **is**

when valeur_de_sélection =>instructions

[**when others** =>instructions]

end case;

I.3.8. الحلقات :

تسمح الحلقات بتكرار سلسلة من التعليمات. توجد ثلاث فئات من الحلقات فيVHDL ، وذلك حسب نمط التكرار المُختار:

–الحلقات البسيطة، بدون نمط تكرار، والتي لا يمكن الخروج منها إلا باستخدام تعليمة«exit» .

–حلقات«for» ، حيث يحدد نمط التكرار عدد مرات التنفيذ.

–حلقات«while» ، حيث يحدد نمط التكرار شرط الاستمرار داخل الحلقة . [15]

- **صيغة « for » :**

for parametre in minimum to maximum loop

séquence d'instructions

end loop;

او:

for parametre in maximum downto minimum loop

séquence d'instructions

end loop [etiquette] ;

- **صيغة «while» :**

while condition loop

séquence d'instructions

end loop [etiquette] ;

تعليمات **exit** و**next** :

- لكي لا نبقى داخل حلقة بسيطة إلى أجل غير مسمى، من الضروري وجود شرط للخروج (exit).
- **الصيغة :**

exit [etiquette] when condition;

- تعليمة next تتيح الانتقال إلى التكرار التالي داخل الحلقة:

next [etiquette] when condition;

9.I. البرامج الفرعية:

البرامج الفرعية هي الوسيلة التي يمكن من خلالها للمبرمج أن ينشئ مكتبة من الخوارزميات المتسلسلة التي يمكنه تضمينها في الوصف.

الفئتان من البرامج الفرعية، الإجراءات والدوال، تختلفان من حيث آليات تبادل المعلومات بين البرنامج المستدعي والبرنامج الفرعي

I.9.1. الدوال:

الدالة تُعيد إلى البرنامج المستدعي قيمة واحدة فقط، وبالتالي لها نوع. يمكنها استقبال وسائط، تكون إما إشارات أو ثوابت، حيث تُنقل إليها القيم عند الاستدعاء.

الدالة لا يمكنها في أي حال من الأحوال تعديل قيم وسائط الاستدعاء الخاصة بها . [15]

- **الصيغة :**

FUNCTION nom de la fonction(قائمة المعاملات الخاصة بالدالة مع نوعها)

RETURN نوع معامل الإرجاع **IS**

منطقة إعلان المتغيرات;

BEGIN

التعليمات المتسلسلة;

RETURN اسم متغير الإرجاع أو قيمة الإرجاع

END; [8]

جسم الدالة لا يمكن أن يحتوي على تعليمة wait والمتغيرات المحلية المعلنة داخل الدالة تتوقف عن الوجود بمجرد انتهاء الدالة.

I.9.2: الإجراءات:

الإجراء، مثل الدالة، يمكنه استقبال وسائط من البرنامج المستدعي: ثوابت، متغيرات أو إشارات. لكن هذه الوسائط يمكن أن تُعلن بأنماط «in» ، «inout» أو «out» (باستثناء الثوابت التي تكون دائمًا بنمط «in») مما يسمح للإجراء بإرجاع أي عدد من القيم إلى البرنامج المستدعي.

- **الصيغة :**

Procedure nom de la procédure (قائمة معلمات الإجراء مع اتجاهها ونوعها)

RETURN IS

; منطقة إعلان المتغيرات

BEGIN

; التعليمات المتسلسلة

END [اسم الإجراء] ;

10.I. الخاتمة:

في ختام هذا الفصل، تم استعراض المفاهيم الأساسية المتعلقة بلغة وصف الأجهزة VHDL، بدءًا من هيكلها العام وصولًا إلى أهم المشغلات والتعليمات المستخدمة فيها. وقد تبيّن أن VHDL ليست مجرد أداة لكتابة وصف رقمي، بل هي لغة متكاملة وشاملة تدعم مختلف مراحل تصميم الأنظمة الرقمية، من النمذجة السلوكية إلى الوصف البنيوي.

كما تم تسليط الضوء على الفوائد المتعددة التي تقدمها هذه اللغة، مثل الدقة في توصيف الوظائف، القدرة على المحاكاة والتحقق من صحة التصميم، وإمكانية إعادة استخدام الشيفرات في مشاريع متعددة. كل هذه الميزات تجعل من VHDL خيارًا فعالًا ومفضلًا لدى المهندسين والمصممين في تطوير الدوائر والأنظمة الرقمية المعقدة.

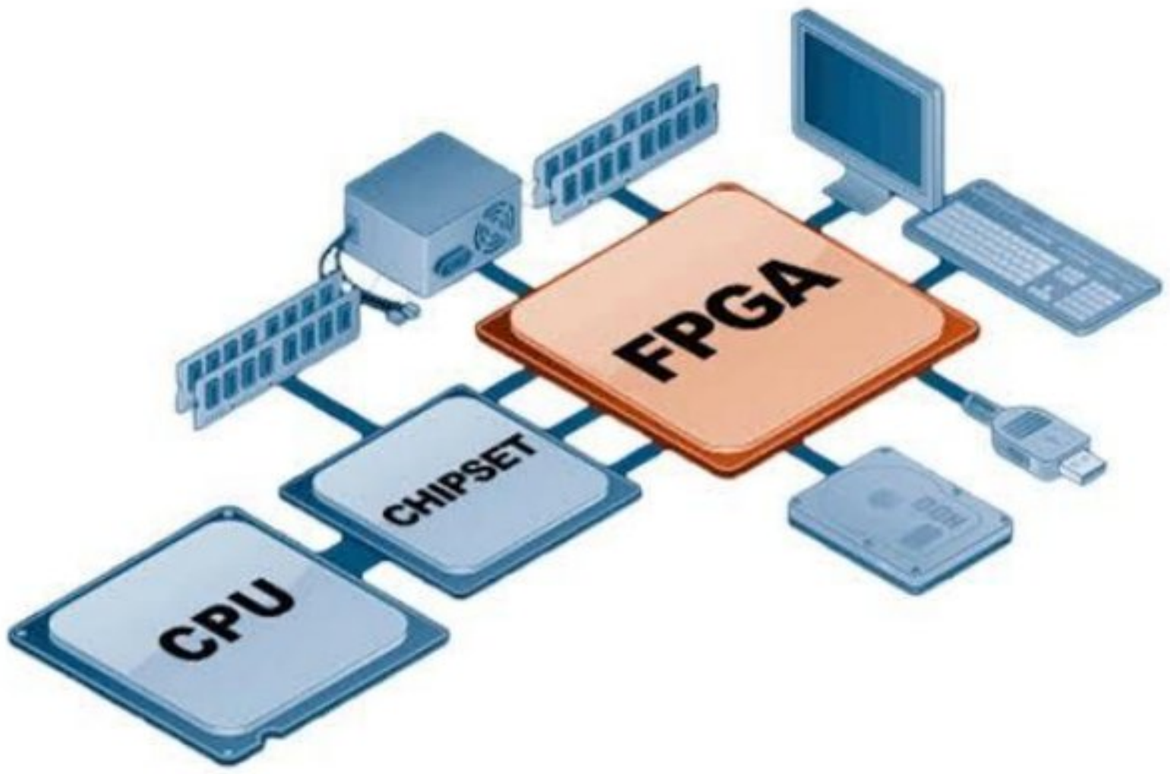
وبذلك، يوفر هذا الفصل الأساس النظري الذي يمكن البناء عليه لفهم أعمق واستخدام أكثر احترافية للغة VHDL في مراحل التصميم المتقدمة.

الفصل الثاني

مصفوفة البوابات المنطقية القابلة للبرمجة

(Field Programmable Gate Array)

F.P.G.A



1.II. مقدمة :

مع التطور المتسارع في تقنيات الدارات الرقمية وتزايد الحاجة إلى أنظمة إلكترونية مرنة وسريعة، برزت الدوائر المنطقية القابلة للبرمجة ميدانيًا أو ما يُعرف بـ FPGA (Field Programmable Gate Array) كأحد الحلول الثورية في عالم التصميم الرقمي. تتميز هذه الشرائح بإمكانية إعادة برمجتها عدة مرات لأداء وظائف مختلفة دون الحاجة إلى تصنيع دائرة جديدة، ما يجعلها مثالية لتطوير النماذج الأولية (prototyping) ، والأنظمة المدمجة (embedded systems)، وحتى في المنتجات النهائية ذات الإنتاج المحدود. تتكون الـ FPGA من عدد كبير من الكتل المنطقية القابلة للتهيئة (CLBs) ، متصلة عبر شبكة معقدة من الخطوط القابلة للبرمجة، بالإضافة إلى موارد إضافية مثل وحدات الذاكرة، مولدات الزمن، ومنافذ الاتصال عالية السرعة. يمكن للمصمم استخدام لغة وصف وانشاء نماذج المنطقية باستخدام VHDL لوصف الدوائر المتكاملة الرقمية المرغوبة فيها، ثم تحويل هذا الوصف إلى تكوين مادي يتم تحميلها على FPGA .[16]

تُستخدم FPGAs اليوم في مجموعة واسعة من التطبيقات، مثل معالجة الإشارات الرقمية (DSP) ، معالجة الصور، الذكاء الاصطناعي، الاتصالات، والتحكم الصناعي، نظرًا لما توفره من أداء عالٍ، مرونة، وتوازي في التنفيذ. ومع توفر أدوات برمجية قوية وسهلة الاستخدام، أصبح بإمكان المهندسين والباحثين تصميم أنظمة رقمية معقدة بسرعة وكفاءة باستخدام هذه التقنية . [17]

2.II. تاريخ ال FPGA :

تاريخ الـ FPGA (Field Programmable Gate Array) يعود إلى أوائل الثمانينات، وقد تطورت هذه التقنية بشكل ملحوظ منذ ذلك الحين :

- **البدايات :**

- في عام 1985، أطلقت شركة Xilinx أول شريحة FPGA تجارية تحت اسم XC2064 ، والتي احتوت على 64 وحدة منطقية قابلة للبرمجة. كانت هذه الشريحة تهدف إلى تقديم بديل أكثر مرونة وأسرع للتطوير من الدوائر المتكاملة المخصصة. [18]

- **التسعينات :**

- شهدت هذه الفترة تطورًا كبيرًا في FPGA من حيث الحجم وسرعة التشغيل وعدد البوابات المنطقية. كما ظهرت شركات منافسة مثلAltera ، التي استحوذت عليها Intel لاحقًا، وتم إدخال دعم لوحدات الذاكرة، والمضاعفات، ومنافذ الإدخال/الإخراج المتقدمة.[18] [19]

- **الألفينات :**

- بدأت الـ FPGA تُستخدم في تطبيقات أكثر تعقيدًا مثل معالجة الإشارات الرقمية، شبكات الاتصال، والأنظمة المدمجة. كما ظهرت أدوات تطوير أقوى وأكثر سهولة، مع استخدام لغات مثل VHDL وVerilog. [19] [20]

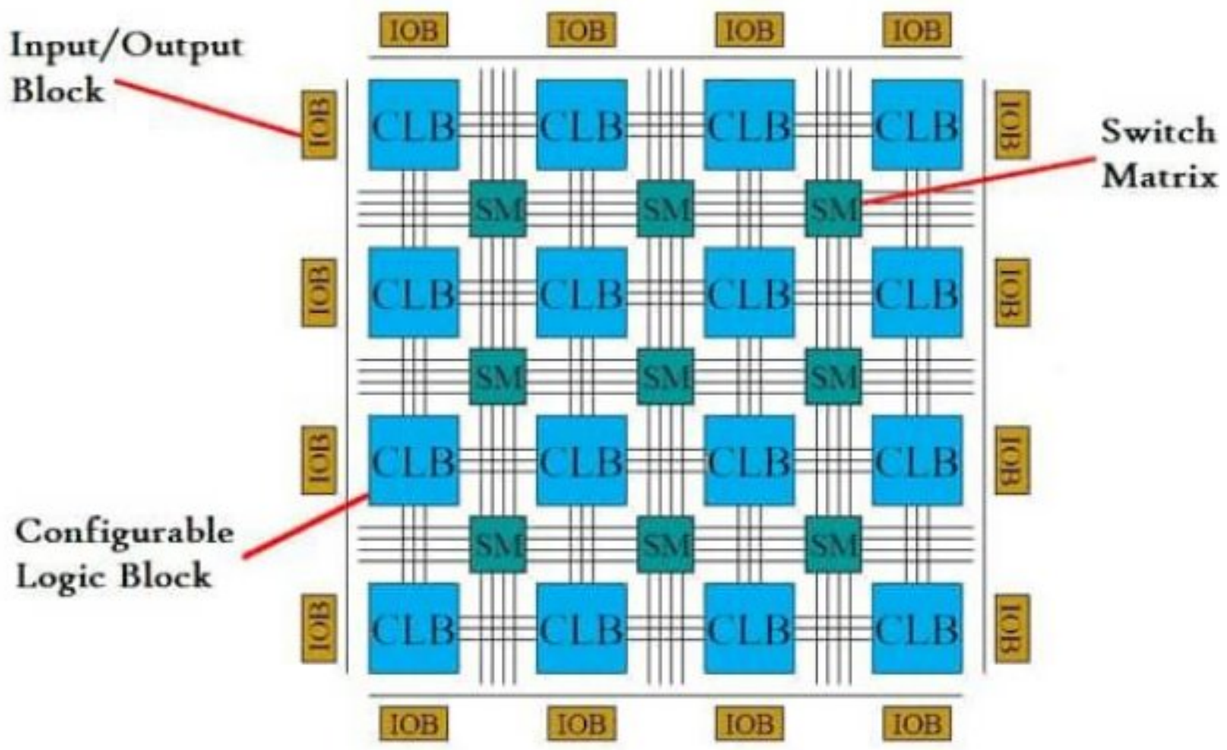
- **العقد الأخير (2010 - حتى الآن):**

- أصبحت الـ FPGA أكثر قوة، وتضم معالجات مدمجة مثل ARM cores في SoC FPGAs يتم استخدامها حاليًا في تطبيقات الذكاء الاصطناعي، تعلم الآلة، شبكات G5، مراكز البيانات، وحتى في السيارات ذاتية القيادة. تتنافس شركات كبيرة مثل Intel من خلال Altera و AMDمن خلال استحواذها على Xilinx عام 2022 بقوة في هذا المجال. [20] [19] [18]

3.II. بنية الداخلية لل FPGA :

تُعتبر الـFPGA بمثابة دوائر متكاملة خاصة (ASIC) قابلة للبرمجة من قبل المستخدم.
قوة هذه الدوائر تكمن في كونها قد تتكون من عدة آلاف، بل وحتى ملايين من البوابات المنطقية والقلابات flip-flops وتدمج الأجيال الحديثة من الـ FPGA حتى ذاكرة حية (RAM) ويُعد كل من XILINX و ALTERA أكبر مصنعين لهذه الدوائر.

تتكون من كتل منطقية أساسية (تضم عدة آلاف من البوابات) يمكن توصيلها ببعضها البعض . [21]



الشكل 01.II: البنية الداخلية FPGA [22]

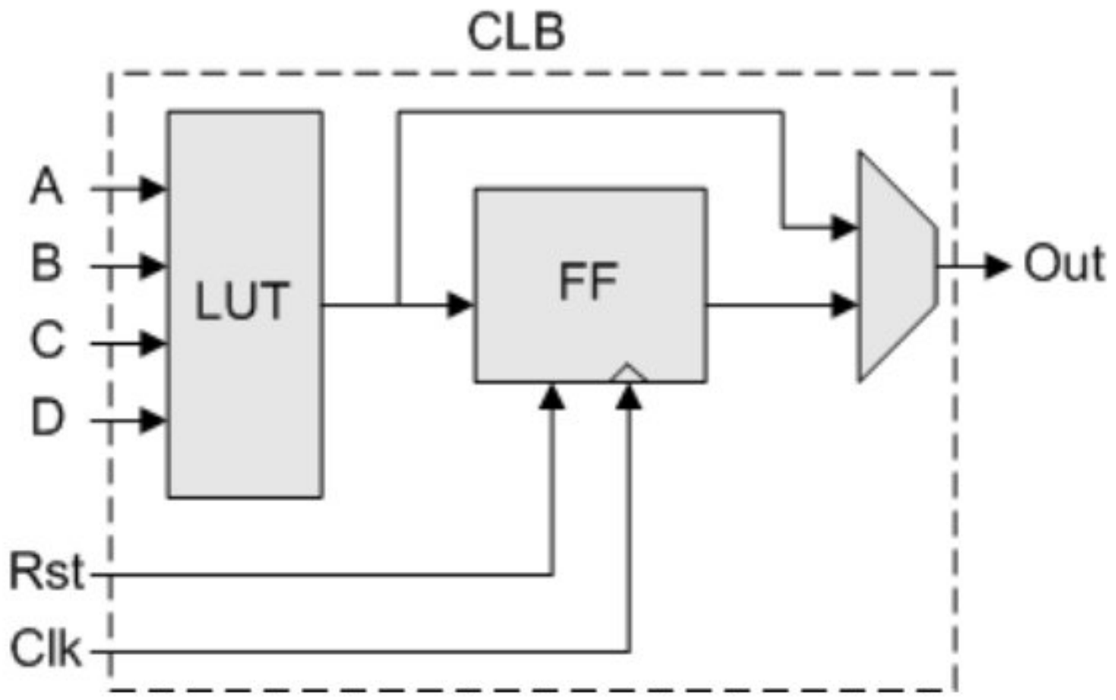
تتكون بنية FPGA (**الشكل 6.II**) من عدة عناصر أساسية:

- **الخلايا المنطقية :(Logic Blocks)** وهي وحدات صغيرة قابلة للبرمجة تحتوي على جداول إقتران (LUTs)، ومسجلات (Flip-Flops) ، وتُستخدم لتنفيذ الوظائف المنطقية الأساسية.
- **مصفوفة التوصيل :(Switch Matrix)** شبكات توصيل تسمح بربط الخلايا المنطقية مع بعضها البعض بطريقة مرنة.

- **بوابات الإدخال/الإخراج (I/O Blocks):** تسمح بالاتصال مع العالم الخارجي، سواء عبر إشارات رقمية أو تماثلية.
- **الساعات والمتحكمات (Clock Management):** مسؤولة عن تنظيم توقيت الإشارات والعمليات داخل الشريحة.
- **الذاكرة المدمجة (BRAM):** وحدات ذاكرة داخلية يمكن استخدامها لتخزين البيانات مؤقتًا أثناء المعالجة.

II.1.3. الخلايا المنطقية (Logic Blocks) في FPGA : [21]

الخلايا المنطقية تُعد العنصر الأساسي في تكوين دارات FPGA ، وهي المسؤولة عن تنفيذ **الوظائف المنطقية** المختلفة داخل الشريحة. تتميز هذه الخلايا بكونها **قابلة للبرمجة**، مما يسمح بتعديل سلوكها حسب الحاجة.



الشكل II.02: الخلايا المنطقية (Logic Blocks) في FPGA [22]

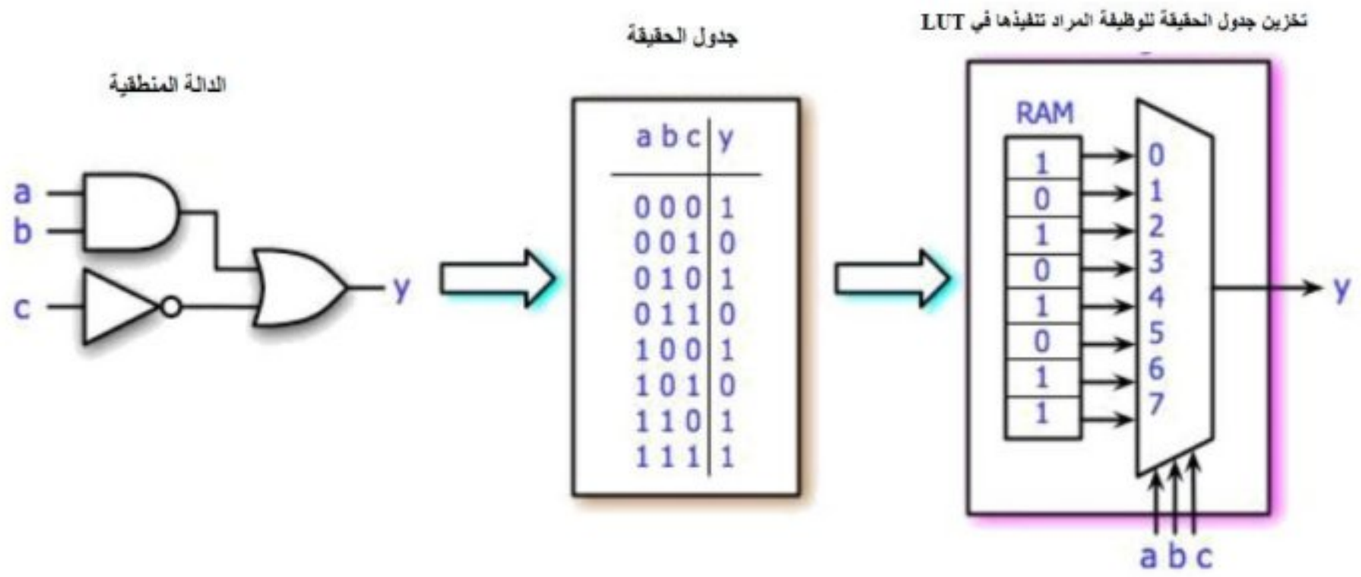
ما تتكون منه الخلية المنطقية الموضح في (**الشكل II.7**) :

1. **جدول الإقتران:(LUT - Look-Up Table)**
   - هو جدول يحتوي على جميع القيم المحتملة لدالة منطقية معينة.
   - يتم استخدامه لتخزين نتائج الدوال المنطقية مثلAND ، OR، XOR.....
   - يمكن اعتباره كـ"ذاكرة صغيرة" يتم استدعاء ناتجها بناءً على قيم الإدخال.
   - مثال: . (الشكل II.3) تخزين جدول الحقيقة للوظيفة المراد تنفيذها في LUT . يبين هذا المثال كيف تقوم FPGAs باستبدال العمليات المنطقية التقليدية مثل AND ، OR، NOT من خلال تخزين النتائج مسبقًا في شكل جدول فيLUT ، مما يسمح بتنفيذ سريع جدًا وبكمون منخفض، لأن كل عملية منطقية تتحول إلى عملية "بحث في جدول."
2. **مسجلات:(Flip-Flops)**
   - تُستخدم لتخزين الحالة المنطقية (0 أو 1) مؤقتًا.
   - تلعب دورًا مهمًا في **التزامن** و**تخزين النتائج الوسيطة** في الدارات الرقمية.

- تعمل بمزامنة مع إشارات الساعة (Clock) لضمان عمل النظام بشكل منظم.

3. **ممرات الربط : (Routing)**

- كل خلية منطقية تتصل بوحدات أخرى عبر شبكة معقدة من التوصيلات.
- هذه التوصيلات تسمح بتكوين علاقات منطقية أكثر تعقيدًا عند برمجة الخلايا معًا.



الشكل 03.II: تخزين جدول الحقيقة للوظيفة المراد تنفيذها في LUT [22]

تُعد الخلايا المنطقية في شرائح FPGA من العناصر الأساسية التي تُستخدم لتنفيذ الوظائف المنطقية، حيث تتكون غالبًا من جداول إقتران (LUTs) ، ومسجلات تخزين(Flip-Flops) ، ودوائر توصيل. لفهم كيفية عمل الـ LUT داخل خلية منطقية، نوضح المثال التالي.

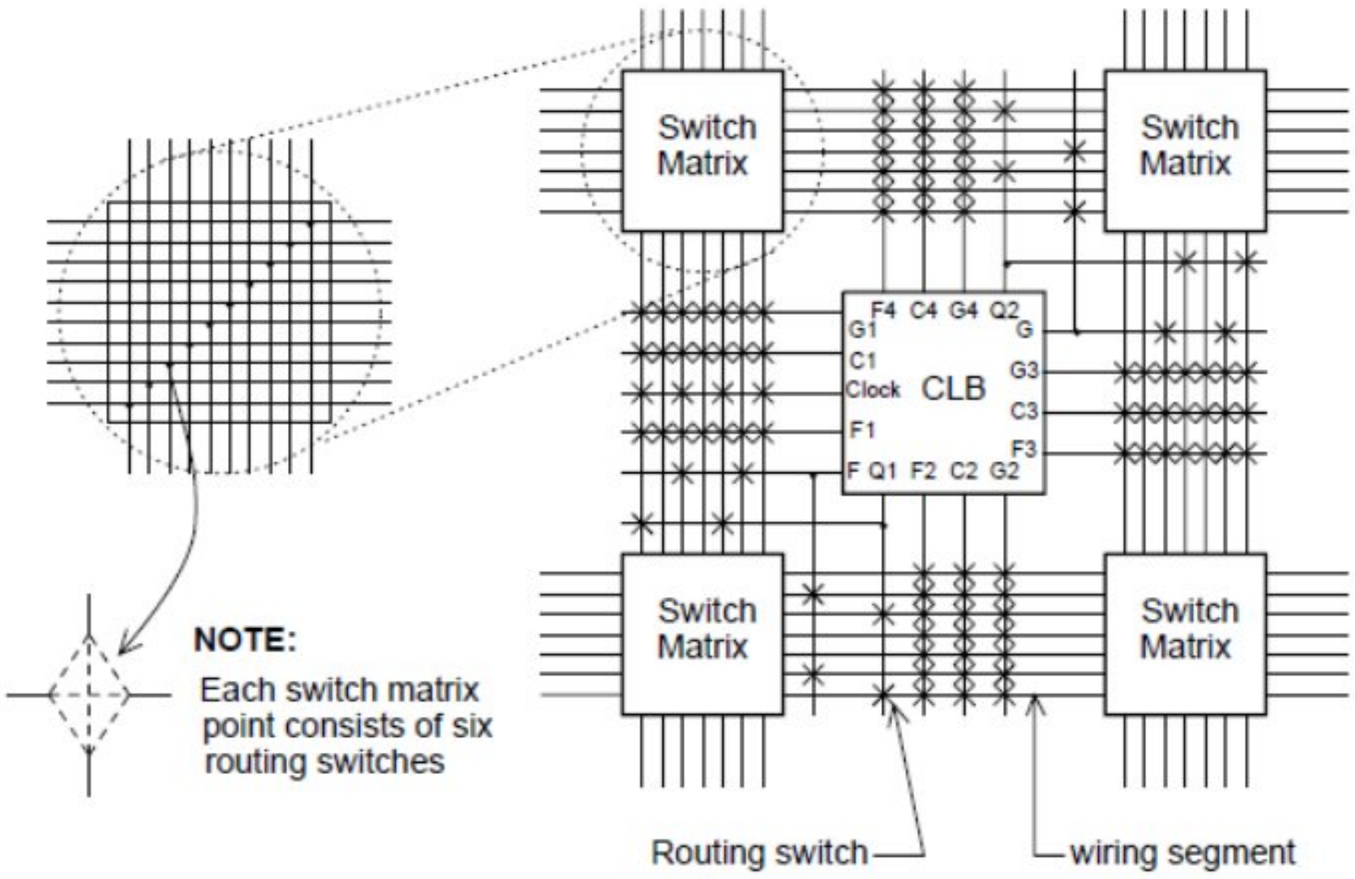
في الشكل 8.II ، لدينا دالة منطقية تعتمد على ثلاث مدخلات a,b,c، وتُعطى بالعلاقة التالية:

y = (a AND b) OR (NOT c)

تقوم هذه الدالة بتنفيذ عملية AND بين المدخلين a وb، ثم NOT للمدخل c، وأخيراً OR بين النتيجتين للحصول على المخرج y بعد تحديد هذه الدالة، يتم بناء **جدول الحقيقة** الذي يحتوي على جميع الحالات الممكنة للمدخلات الثلاثة وعددها 8 حالات (2^3)، مع تحديد القيمة الناتجة y لكل حالة.

يتم بعد ذلك تخزين جدول الحقيقة في LUT داخل الخلية المنطقية، حيث يُمثل LUT ذاكرة صغيرة (RAM) يمكن الوصول إليها باستخدام قيم المدخلات كمؤشر(Address) ، ليتم إرجاع القيمة المناسبة مباشرةً. هذا الأسلوب يُلغي الحاجة إلى تنفيذ العمليات المنطقية خطوة بخطوة، ويُستبدل بعملية "بحث" سريعة في جدول مبرمج مسبقًا.

هذا النهج يجعل من LUT وحدة فعّالة جدًا لتنفيذ الوظائف المنطقية بسرعة ومرونة عالية داخل FPGA ، مع إمكانية إعادة برمجة هذه الجداول لتغيير سلوك الدارة حسب الحاجة



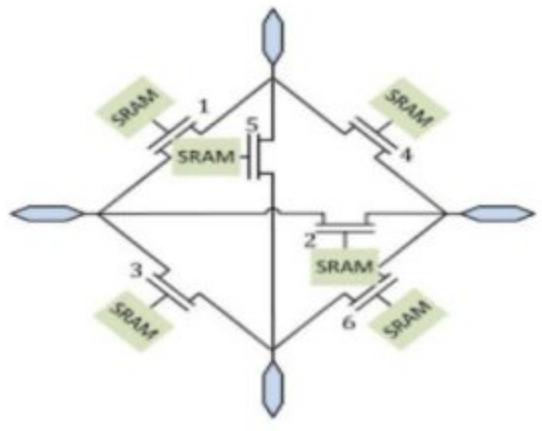
الشكل 04.II: مصفوفة التوصيل (Switch Matrix) [22]

2.3.II. مصفوفة التوصيل (Switch Matrix)

مصفوفة التوصيل (Switch Matrix) (**الشكل 9.II**)هي مكون أساسي في بنية FPGA ، وتُستخدم لربط مختلف الأجزاء داخل الشريحة مثل:

- الخلايا المنطقية (Logic Blocks)
- خطوط التوصيل (Routing Channels)
- مداخل ومخارج FPGA (I/O blocks)

تعمل هذه المصفوفة مثل "موزع إشارات قابل للبرمجة"، حيث تسمح بإنشاء مسارات مخصصة للبيانات بين المكونات المختلفة بناءً على التصميم المطلوب .كل نقطة تقاطع داخل مصفوفة التوصيل تتكون من 6 مفاتيح توجيه، أي أن هناك مرونة كبيرة في اختيار المسار المطلوب للإشارة (الشكل 9.II) كل نقطة في مصفوفة التوصيل تحتوي على **مفاتيح إلكترونية قابلة للبرمجة** (عادةً ترانزستورات أو مفاتيح تعتمد على ذاكرة (الشكل 10.II))، وتتحكم هذه المفاتيح في ما إذا كانت هناك **وصلة كهربائية بين مسارين معينين** أم لا. [21]



الشكل 05.II: نقطة التوصيل [22]

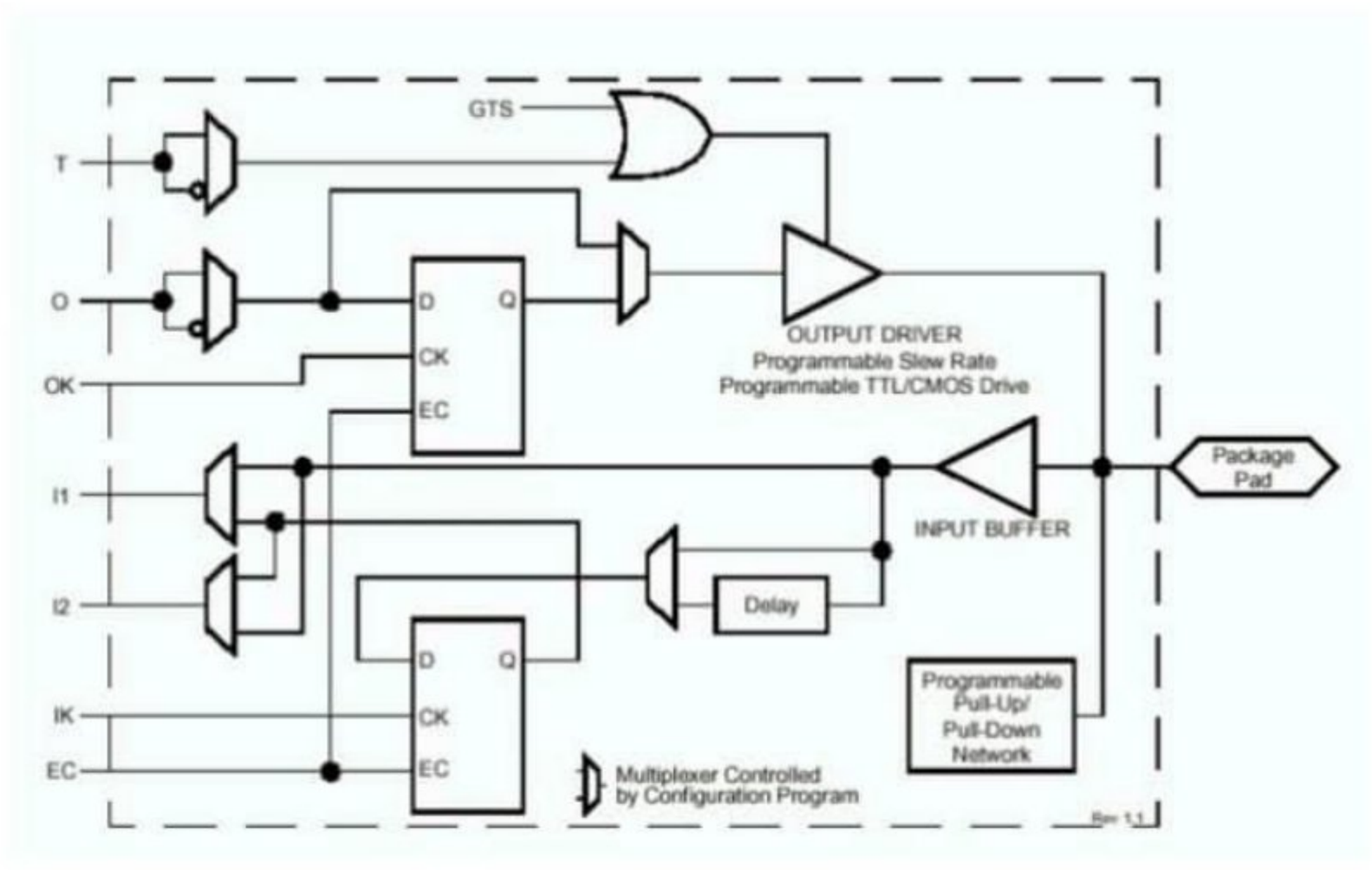
II.3.3. بوابات الإدخال/الإخراج:(I/O Blocks)

تُعد كتل الإدخال/الإخراج (IOBs) من المكونات الأساسية في الـFPGA ، حيث تمثل واجهة الربط بين المنطق الداخلي للشريحة و الخارجي. تحتوي كل كتلة IOB على مكونات قابلة للبرمجة تتيح التحكم الكامل في كيفية استقبال أو إرسال الإشارات من وإلى المسامير الفيزيائية .(Package Pads) يتضمن ذلك مخزن إدخال (Input Buffer) يعمل على تهيئة الإشارات الخارجية لتكون مناسبة للمعالجة داخلFPGA ، بالإضافة إلى مشغّل إخراج (Output Driver) يُستخدم لإرسال الإشارات إلى الخارج مع إمكانية التحكم في نوعية الخرج (TTL أو CMOS) وسرعة الانتقال (Slew Rate) .

تحتوي كل IOB على وحدات اختيار (Multiplexers) تسمح بتحديد مصدر أو وجهة الإشارة، إضافة إلى مسجلات (Flip-Flops)قابلة للبرمجة تُستخدم لمزامنة الإشارات مع إشارة الساعة، مما يساهم في تحقيق اتساق زمني عالي في الأنظمة عالية السرعة. كما توفر كتل IOB إمكانية إدخال تأخير زمني مبرمج (Programmable Delay) لتعديل توقيت الإشارات بما يتلاءم مع متطلبات التصميم. وتتميز هذه الكتل أيضًا بوجود إشارة تحكم عالمية ثلاثية الحالة (GTS) يمكن استخدامها لتعطيل جميع المخارج بشكل موحّد خلال فترات التهيئة أو عند الحاجة إلى التحكم الجماعي في الحالة المنطقية للمخارج.

بفضل هذه المرونة، تُمكن كتل الإدخال/الإخراج في FPGA من تصميم أنظمة رقمية دقيقة وموثوقة تُراعي متطلبات السرعة والتوافق الكهربائي لمختلف التطبيقات. [21]

يوضح الشكل التالي مثالا على SPARTRAN IOB مع مكوناته المختلفة :



الشكل II.06: بنية (IOB SPARTAN) [22]

4.II. مزايا وعيوب الـ FPGA :

1.4.II. المزايا: [23]

1. المرونة : يمكن إعادة برمجة الـ FPGA لتلبية احتياجات محددة، مما يسمح بتخصيص الأنظمة بسهولة دون الحاجة لتصنيع رقائق جديدة.
2. التحديث السهل : يمكن تحديث وظائف الـ FPGA ببرامج ثابتة جديدة، مما يتيح تحسينات مستمرة في الأداء والوظائف.
3. الأداء العالي: توفر الـ FPGAs أداءً عالياً في تطبيقات معالجة الإشارات الرقمية، حيث يمكن أن تدعم عددًا كبيرًا من القنوات في وقت واحد .
4. دعم تنسيقات متعددة : يمكن برمجة الـ FPGA لدعم مجموعة واسعة من تنسيقات الصوت الرقمية، مما يجعلها متعددة الاستخدامات .
5. الكفاءة في استهلاك الطاقة : في بعض التطبيقات، يمكن أن تكون الـ FPGAs أكثر كفاءة في استهلاك الطاقة مقارنةً ببعض المعالجات التقليدية.

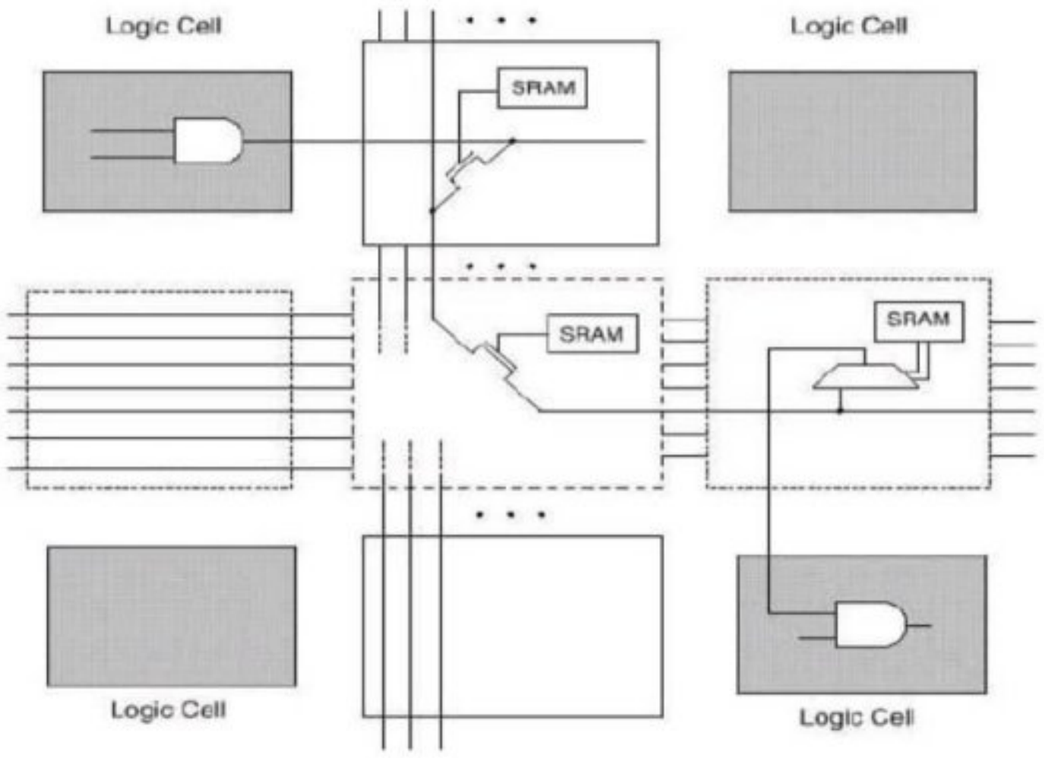
2.4.II. العيوب: [24]

1. التكلفة : عادةً ما تكون تكلفة الـ FPGA أعلى من رقائق معالجة الإشارات الرقمية التقليدية، مما قد يجعلها أقل جاذبية في التطبيقات ذات التكلفة المنخفضة .
2. استهلاك الطاقة : قد تستهلك الـ FPGAs طاقة أكثر من الحلول التقليدية، مما يجعلها غير مناسبة لبعض التطبيقات المحمولة أو التي تعمل بالبطارية.
3. تعقيد التصميم : تصميم وتصنيع أنظمة تعتمد على الـ FPGA يمكن أن يكون أكثر تعقيدًا، مما يتطلب مهارات متخصصة.
4. أداء أقل في بعض التطبيقات: على الرغم من أن الـ FPGAs تقدم أداءً عالياً في بعض المجالات، إلا أنها قد لا تتفوق على الحلول المخصصة مثل ASICs في التطبيقات ذات الحجم الكبير.

5.II. التقنيات المختلفة المستخدمة في شرائح FPGA :

1.5.II. ذاكرة الوصول العشوائي الساكنة (SRAM) :

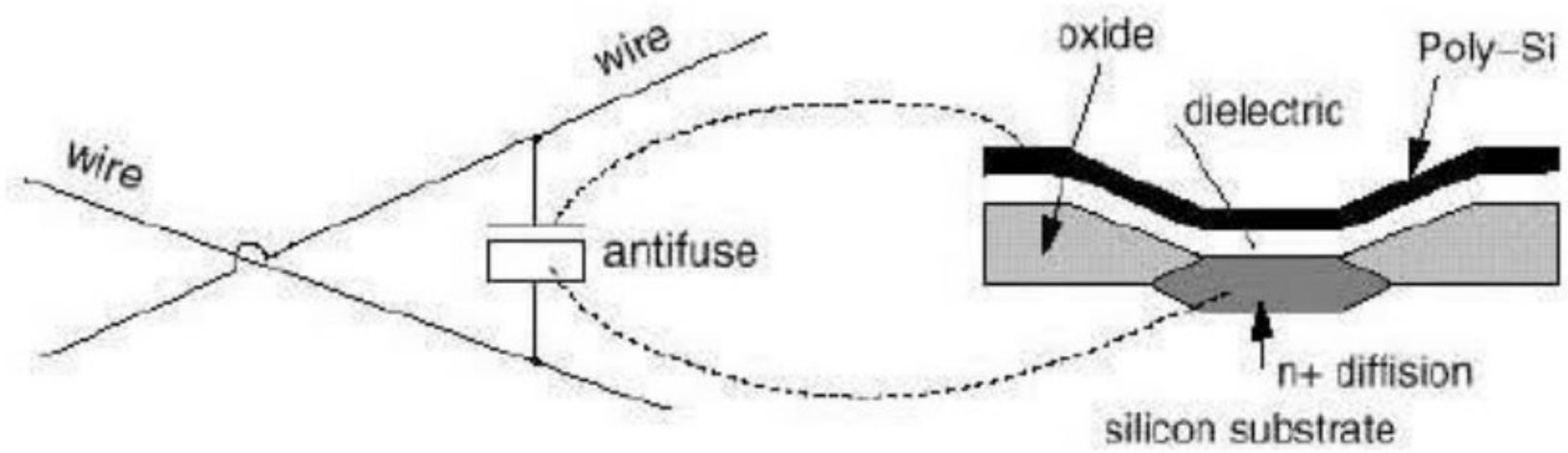
يتم إنشاء الاتصالات عن طريق جعل الترانزستورات في حالة تمرير. تتميز هذه التقنية بإمكانية إعادة التهيئة السريعة داخل نفس الدارة. أما العيب الرئيسي لها فهو المساحة الكبيرة التي تتطلبها خلايا SRAM كما تحتاج هذه الشرائح إلى ذاكرة قياسية يتم تحميلها عند بدء التشغيل، وتعتمد على تقنية CMOS . [25]



الشكل II.07: تقنية (SRAM) [26]

II.2.5. تقنية مضاد الانصهار(Anti-Fuse):

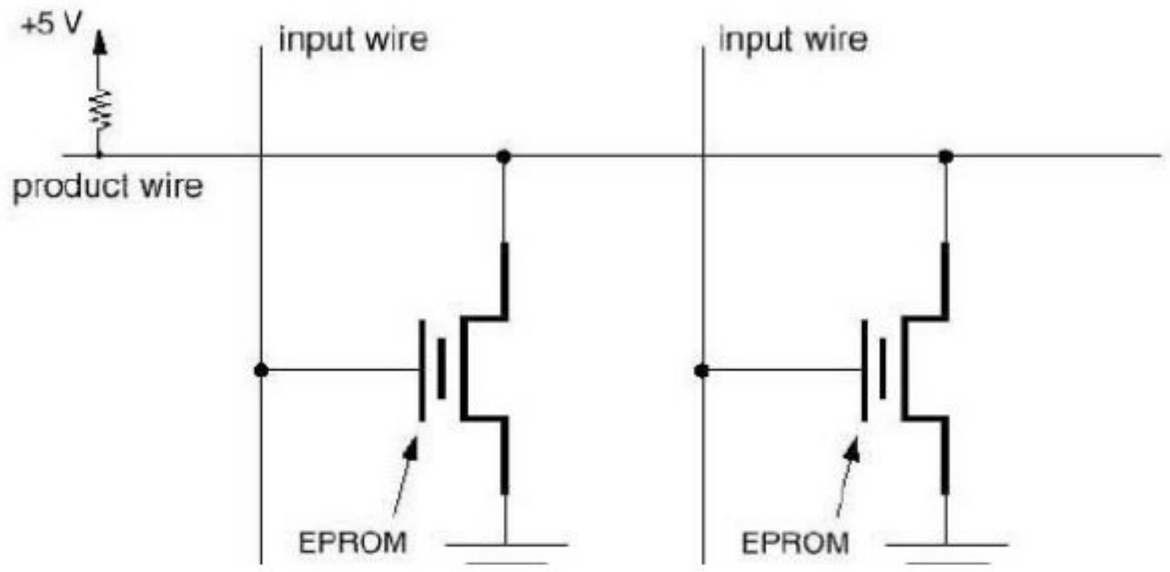
في حالتها الأصلية، تكون هذه التقنية في وضع مقاومة عالية(high impedance) ، ويمكن برمجتها لتصبح في حالة مقاومة منخفضة أو ما يُعرف بحالة "الانصهار". تُعتبر هذه التقنية أقل تكلفة منSRAM ، وتُتيح الوصول إلى سرعات أعلى مع احتلال مساحة أصغر على الشريحة. لكن من عيوبها أن شرائح FPGA المبنية بهذه التقنية لا يمكن برمجتها سوى مرة واحدة فقط. [26]



الشكل II.08: تقنية (Anti-Fuse) [26]

II.3.5. تقنية EPROM/EEPROM :

تعتمد هذه الطريقة على نفس التقنية المستخدمة في ذواكر EPROM وتُعتبر EEPROM قابلة لإعادة البرمجة. تعمل الشريحة بشكل مستقل دون الحاجة لذاكرة خارجية. وتُعد خصائصها الكهربائية ومساحتها المتوسطة مشابهة لتقنية SRAM . [25]



الشكل 09.II: تقنية (EPROM/EEPROM) [26]

6.II. الفرق بين fpga وتقنيات اخرى :

تعتبر مصفوفات البوابات الإلكترونية القابلة للبرمجة في الموقع (FPGAs) واحدة من التقنيات المهمة في تصميم الدوائر المتكاملة، وتختلف عن تقنيات أخرى مثل وحدات التحكم الدقيقة (MCUs) والدوائر المتكاملة محددة التطبيقات (ASICs) في عدة جوانب. إليك مقارنة توضح الفروق الرئيسية بين FPGAs وبعض التقنيات الأخرى:

1.6.II. البرمجة وإعادة التكوين :

FPGAs: تتميز FPGAs بإمكانية إعادة البرمجة على مستوى الأجهزة، مما يسمح للمستخدمين بتعديل تصميم الدائرة بعد التصنيع باستخدام لغات وصف الأجهزة مثل VHDL أو Verilog هذا يجعلها مرنة للغاية في التطبيقات التي تتطلب تغييرات متكررة أو تخصيصات خاصة. [27]

وحدات التحكم الدقيقة : على الرغم من أنها قابلة للبرمجة، إلا أن وحدات التحكم الدقيقة تسمح فقط بإعادة برمجة البرمجيات (firmware) وليس إعادة برمجة الأجهزة نفسها. هذا يعني أن تصميمها ثابت ولا يمكن تغييره بعد التصنيع. [28]

ASICs : لا يمكن إعادة برمجة ASICs بعد تصنيعها، حيث يتم تصميمها لتطبيقات محددة فقط. هذا يجعلها أقل مرونة مقارنة بـ FPGAs . [28]

2.6.II. الأداء والقدرة على المعالجة : [28]

FPGAs : توفر FPGAs قدرة معالجة عالية، حيث يمكنها تنفيذ عمليات متعددة في وقت واحد (parallel processing) ، مما يجعلها مثالية للتطبيقات التي تتطلب معالجة سريعة مثل معالجة الفيديو أو الذكاء الاصطناعي.

وحدات التحكم الدقيقة : عادةً ما تكون وحدات التحكم الدقيقة أقل قوة من حيث الأداء، حيث تعتمد على معالجة تسلسلية (sequential processing) مما قد يؤدي إلى اختناقات في الأداء في التطبيقات المعقدة.

ASICs :تقدم ASICs أداءً عالياً وكفاءة أفضل في استهلاك الطاقة، مما يجعلها مناسبة للتطبيقات التي تتطلب تصميمات مخصصة وفعالة.

II.3.6. التكلفة :

FPGAs : تعتبر FPGAs أكثر تكلفة من وحدات التحكم الدقيقة، ولكنها قد تكون أكثر فعالية من حيث التكلفة في التطبيقات التي تتطلب تغييرات متكررة أو تخصيصات خاصة. [28]

وحدات التحكم الدقيقة : عادةً ما تكون وحدات التحكم الدقيقة أقل تكلفة، مما يجعلها خياراً مفضلاً للتطبيقات البسيطة أو ذات الحجم الكبير. [27]

ASICs : على الرغم من أن تكلفة التصميم الأولية لـ ASICs أعلى، إلا أن التكلفة لكل وحدة تكون أقل في الإنتاج الكبير، مما يجعلها خياراً اقتصادياً في التطبيقات ذات الحجم الكبير.[27]

II.4.6. الاستخدامات :

FPGAs : تُستخدم بشكل شائع في التطبيقات التي تتطلب مرونة عالية وأداءً متقدماً، مثل النماذج الأولية وتصميم الدوائر المتخصصة. [28]

وحدات التحكم الدقيقة : تُستخدم في التطبيقات البسيطة مثل التحكم في الأجهزة المنزلية أو الأنظمة المدمجة التي لا تتطلب تغييرات متكررة. [28]

ASICs : تُستخدم في المنتجات التي تتطلب تصميمات مخصصة مثل الهواتف الذكية وأجهزة الكمبيوتر، حيث تكون الكفاءة والأداء مهمين. [27]

II.7. تطبيقات الFPGA :

تُستخدم مصفوفات البوابات المنطقية القابلة للبرمجة (FPGA) في العديد من التطبيقات، ومن بينها:

* نموذج أولي للدارات الجديدة : تُستخدم لتطوير واختبار تصميمات الدوائر قبل الإنتاج الضخم.

* تصنيع مكونات خاصة بكميات صغيرة : تُستخدم في إنتاج مكونات مخصصة بكميات محدودة.

* التكيف مع الاحتياجات أثناء الاستخدام : توفر مرونة لتلبية المتطلبات المتغيرة في التطبيقات المختلفة.

* أنظمة التحكم في الوقت الحقيقي : تُستخدم في التطبيقات التي تتطلب استجابة فورية ودقيقة.

* معالجة الإشارات الرقمية : (DSP) تُستخدم في معالجة البيانات الصوتية والمرئية.

* التصوير الطبي : تُستخدم في الأجهزة الطبية مثل أجهزة التصوير بالرنين المغناطيسي والأشعة السينية.[29]

8.II. تهيئة شرائح FPGA :

يتم حفظ إعدادات الدارة (Configuration) في طبقة الشبكة المصنوعة من ذاكرة SRAM ، بينما تُخزَّن هذه الإعدادات في ذاكرة ROM خارجية. وعند كل عملية تشغيل، يقوم جهاز داخلي بتحميل محتوى ROM إلى ذاكرة SRAM الداخلية .

من هنا يمكن بسهولة استخدام نفس الشريحة لتنفيذ وظائف مختلفة عبر تغيير ذاكرة ROM فقط، لأن برمجتها الداخلية ليست نهائية .

وهذا ما يُبرز الفائدة الكبيرة من هذه المرونة، خصوصًا خلال مرحلة تطوير واختبار التصميم. [25]

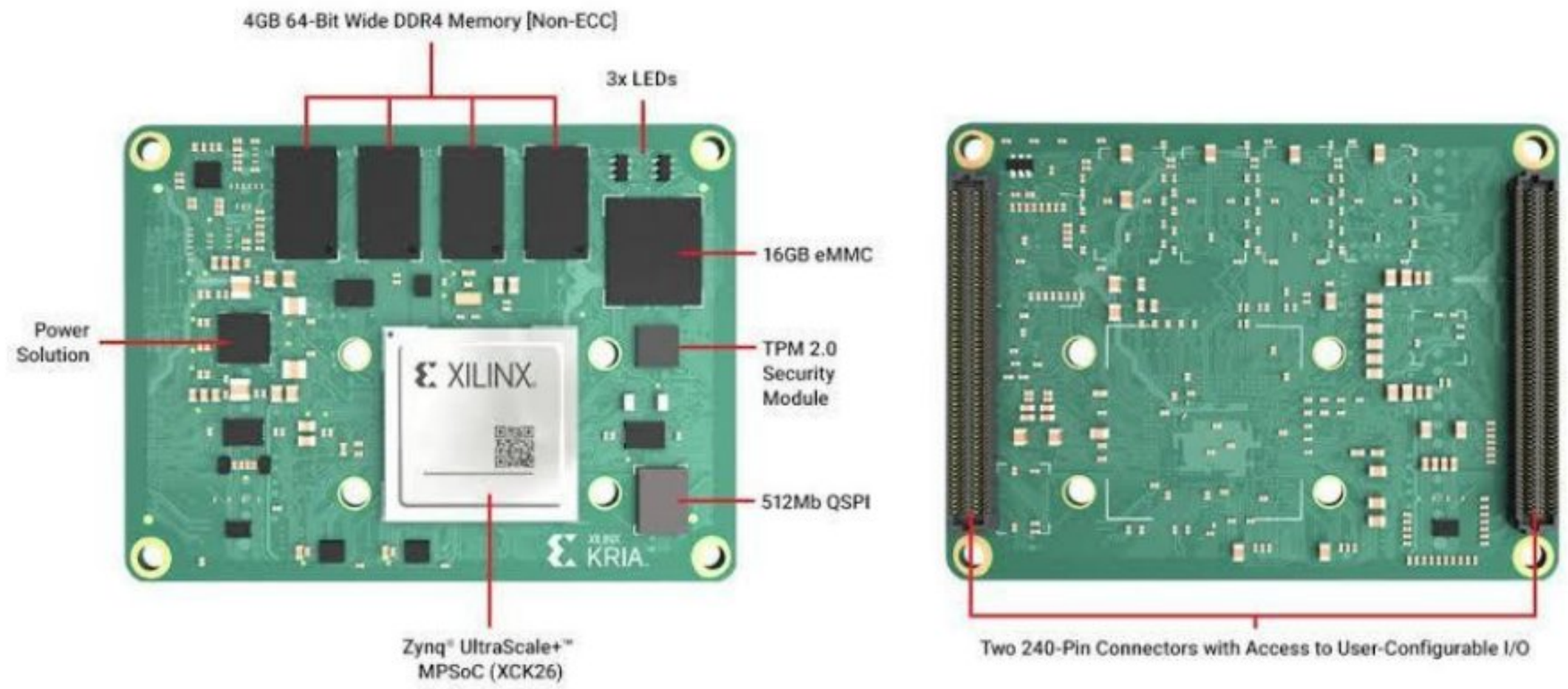
9.II. برمجة FPGAs :

نبدأ أولاً بوصف التصميم، إما باستخدام لغة وصف العتاد مثل VHDL أوVerilog ، أو من خلال إدخال المخطط مباشرة. بعد ذلك، يقوم أداة التوليف (Synthesizer) بتوليد قائمة الشبكات (Netlist) ثم تأتي مرحلة وضع المكونات (Placement)داخل شريحة FPGA إذا كان ذلك ممكنًا، حيث إن بعض شرائح FPGA لا تدعم محاكاة بعض العناصر مثل Latchs يليها توصيل هذه المكونات ببعضها (Routing) داخل الشريحة. عند انتهاء هذه المراحل، يكون قد تم توليد ملف البتات (Bitstream) الذي سيكون جاهزًا للإرسال إلى شريحة الـ FPGA وفقط في هذه المرحلة تبدأ عملية البرمجة الفعلية.

10.II. أهم الشركات المصنعة للـ FPGAs :

1.10.II. AMD Xilinx: [30]

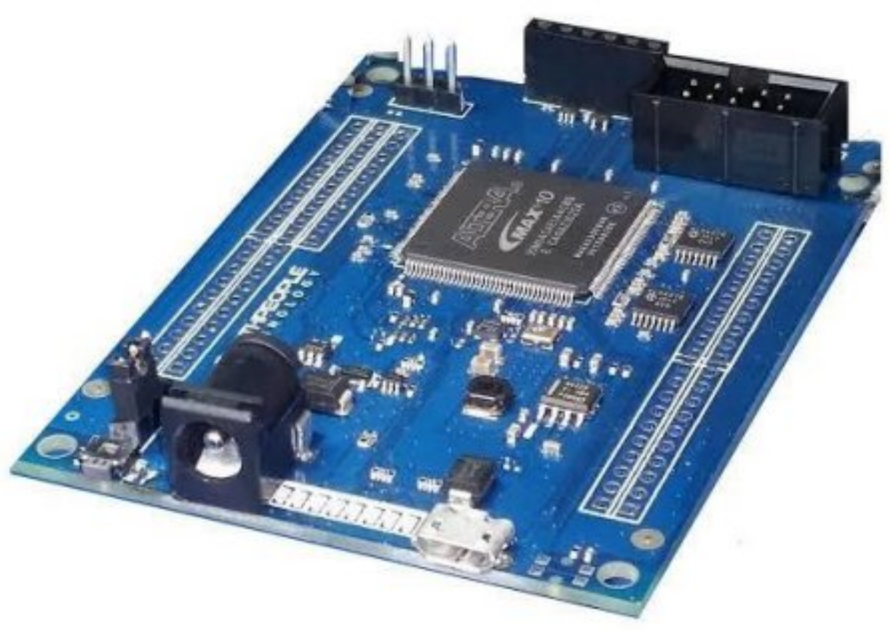
- تعتبرXilinx ، التي استحوذت عليها AMD ، من الشركات الرائدة في مجال FPGAs تقدم مجموعة واسعة من الحلول لتلبية احتياجات التطبيقات المختلفة، بما في ذلك الشبكات والتطبيقات ذات التكلفة المنخفضة.



الشكل 10.II: شريحة FPGA من شركة (AMD XLINIX)

II.10.2. Intel (Altera) : [30]

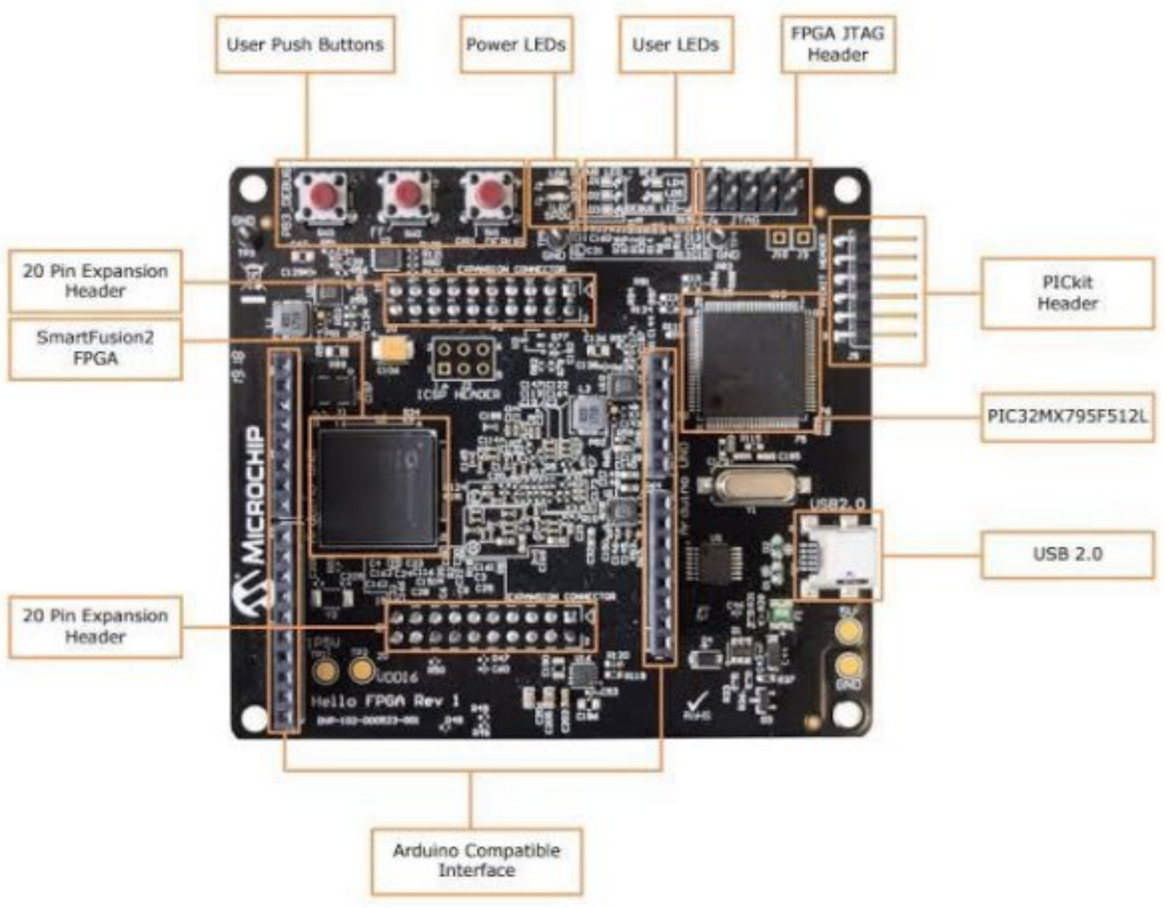
- استحوذت Intel على Altera في عام 2015، وتعتبر الآن جزءًا من مجموعة الحلول القابلة للبرمجة. تقدم Intel FPGAs مع مجموعة متنوعة من المكونات مثل SRAM و IP مدمج، مما يساعد في تقليل وقت الإنتاج والتكاليف.



الشكل II.11: شريحة FPGA من شركة (Intel Altera)

II.10.3. Microchip Technology : [31]

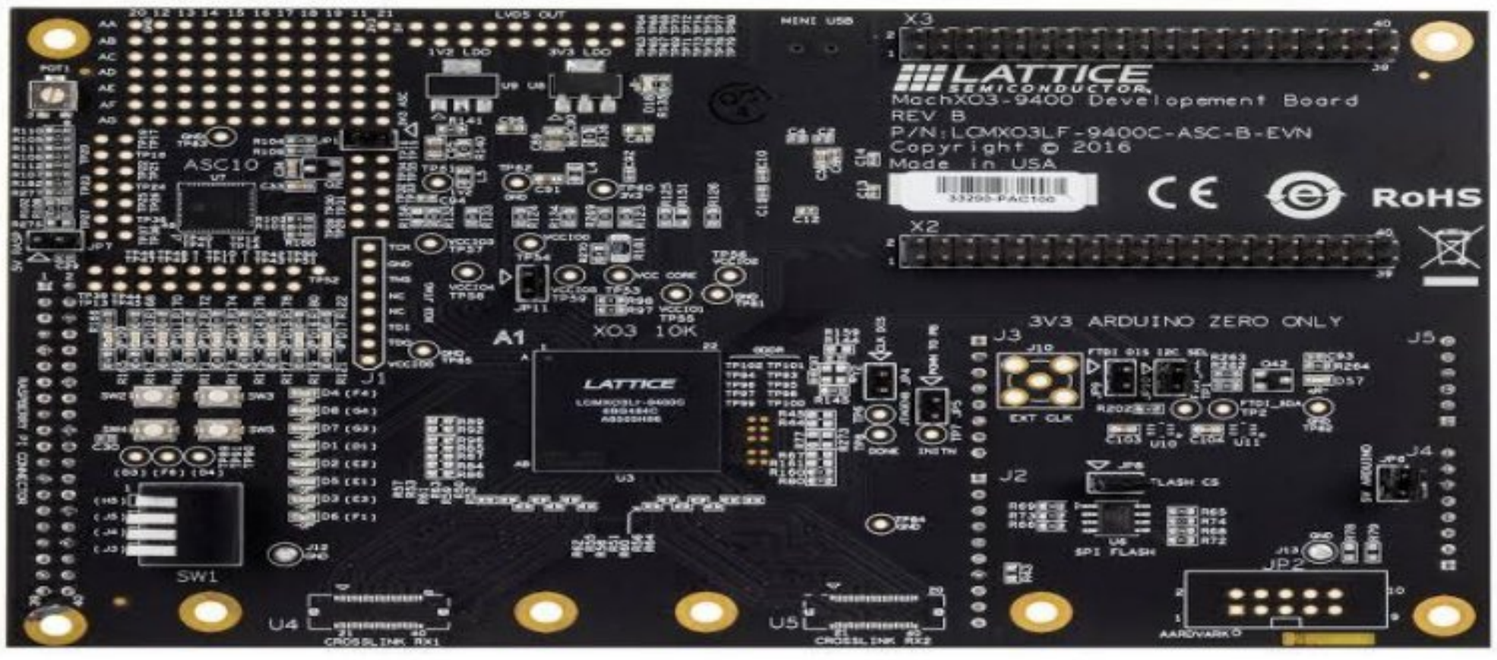
- تقدم Microchip FPGAs لتكامل المنطق، مع التركيز على السرعة والأمان. كما توفر حلولًا مسبقة البناء لتسهيل التصميم.



الشكل II.12: شريحة FPGA من شركة (Microchip)

II.10.4. Lattice Semiconductor : [32]

- تركز Lattice على تصنيع FPGAs منخفضة الطاقة، وتستخدم في مجموعة متنوعة من التطبيقات مثل الاتصالات والصناعات.



الشكل II.13: شريحة FPGA من شركة (Lattice Semiconductor)

II.10.5. Achronix Semiconductor : [33]

- تقدم Achronix حلول FPGAs عالية الأداء ومنتجات eFPGA ، مما يساعد في تسريع المهام الحسابية المعقدة.

الشكل II.14: شريحة FPGA من شركة (Achronix Semiconductor)

QuickLogic .6.10.II : [33]

- تركز QuickLogic على تطوير منصات كاملة للذكاء الاصطناعي ومعالجة الصوت، وتقدم أيضًا تقنيات eFPGA

الشكل 15.II: شريحة FPGA من شركة (QuickLogic)

Flex Logix Technologies .7.10.II : [34]

- متخصصة في تقنيات eFPGA وحلول تسريع الذكاء الاصطناعي، مما يجعلها لاعبًا مهمًا في السوق

الشكل 16.II: شريحة FPGA من شركة (Flex Logix Technologies)

11.II. تطور السوق : [35]

نمو السوق : من المتوقع أن ينمو سوق FPGAs بشكل كبير، حيث كانت قيمته حوالي 6.5 مليار دولار في عام 2022، ومن المتوقع أن تصل إلى 13.5 مليار دولار بحلول عام 2032، بمعدل نمو سنوي مركب يبلغ 7.8% .

زيادة الطلب : يتزايد الطلب على FPGAs في مجالات مثل الذكاء الاصطناعي، والتعلم الآلي، والشبكات G5، مما يعزز من استخدام هذه التقنية في مراكز البيانات والتطبيقات الصناعية.

التوجهات الحديثة : تشمل الاتجاهات الحديثة في السوق تطوير تقنيات جديدة مثل FPGAs المدمجة ، مما يوفر أداءً أفضل ومرونة أكبر.

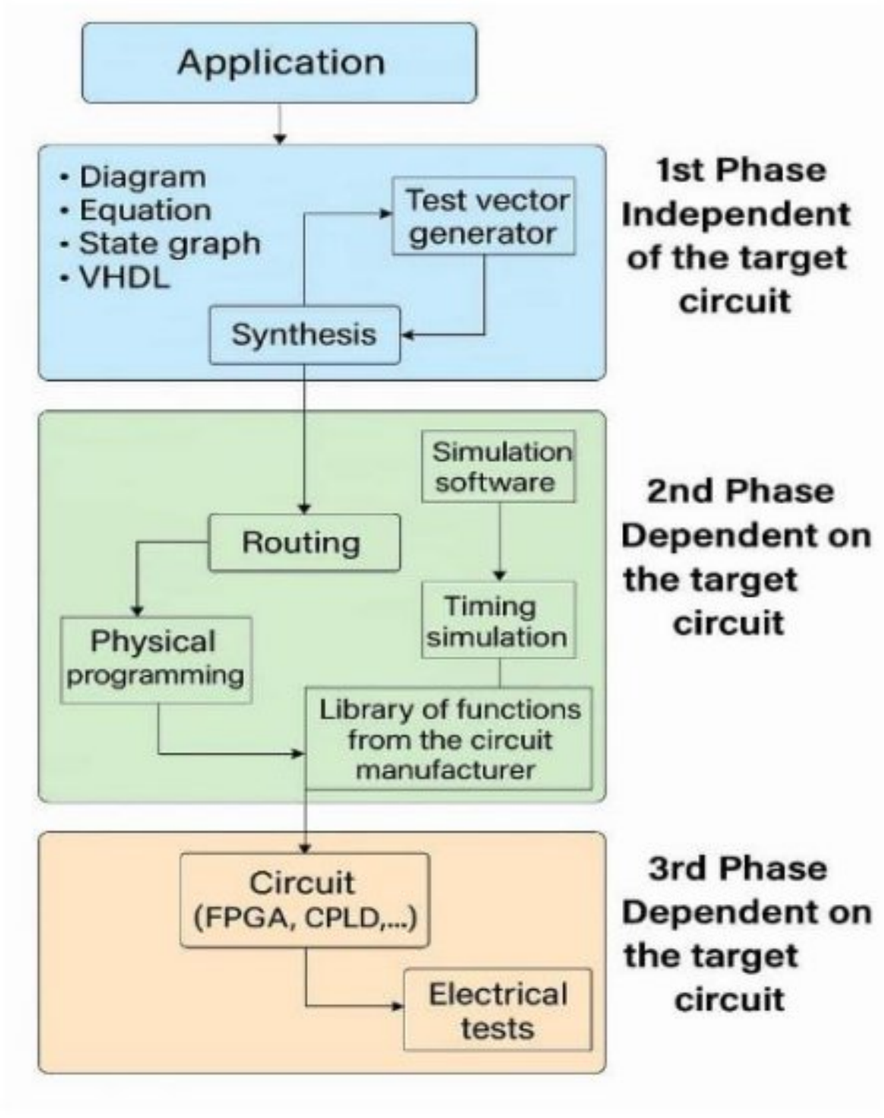
12.II. البرنامج المخصص لبرمجة FPGA :

حيث يتم الارسال من خلال برنامج QUARTUS :

QUARTUS: هو برنامج مطوّر من قبل شركة Altera، يوفّر بيئة متكاملة لإدارة سلسلة التصميم الخاصة بالدوائر المنطقية القابلة للبرمجة مثل CPLDو FPGA يتيح هذا البرنامج إمكانية إدخال التصاميم إما من خلال واجهة رسومية أو عن طريق لغات توصيف العتاد مثل VHDL أوVerilog ، مع إمكانية إجراء عمليات المحاكاة، والتوليف (synthèse)، والتنفيذ (implémentation) على الهدف البرمجي.

يضم Quartus مجموعة من الأدوات المتقدمة لتصميم الأنظمة على مستوى عالٍ، مع إمكانية الوصول إلى مكتبة واسعة من الوحدات الجاهزة (IP) التي توفّرها Altera, كما يحتوي على محرك فعال لعملية "الوضع والتوصيل " (placement & routing) مدعوم بتقنيات تحسين التوليف الفيزيائي وأدوات متقدمة للتحقق من صحة التصميم.

بشكل عام، تمرّ عملية تصميم وتكوين الدوائر القابلة للبرمجة بعدة مراحل متسلسلة تُعرف باسم سلسلة التصميم (Design Flow) ، تهدف في النهاية إلى تهيئة المكون البرمجي للعمل حسب المتطلبات المحددة.



الشكل 17.II: مراحل تصميم دائرة باستخدام FPGA

13.II. خاتمة :

في هذا الفصل، قمنا بدراسة موجزة حول الـFPGA ، حيث عرضنا بنيته ومعماريته بالتفصيل، وكذلك تقنيات تهيئة هذه الدوائر التي توفر أداءً لا نجده في دوائر أخرى من حيث المرونة، وإمكانية التهيئة وإعادة التهيئة. كما أنها تمتلك إدارة ديناميكية للاتصال مع بيئاتها، مما يسمح بتنفيذ خوارزميات مختلفة مناسبة لتطبيقات متنوعة، مع الحفاظ على مرونة كبيرة في الاتصال والتواصل.

الفصل الثالث

انشاء مشروع لادارة البيانات باستعمال FPGA مع VHDL

Créer un projet de gestion de données utilisant FPGA avec VHDL

1.III. مقدمة :

في عصرنا الرقمي، أصبحت إدارة البيانات في الأنظمة المضمنة مسألة مركزية في تصميم حلول ذكية وفعالة. من بين الوسائل التقنية التي أثبتت كفاءتها العالية في هذا المجال نجد الدارات المنطقية القابلة للبرمجة (FPGA) ، التي تتيح إمكانية تصميم أنظمة مخصصة ذات أداء عالٍ، إلى جانب لغة توصيف الدوائر المنطقية (VHDL) التي تُستخدم لبرمجة سلوك هذه الدارات بدقة وتحكم كبيرين.

في هذا المحور الأخير، سننتقل من المفاهيم النظرية إلى الجانب العملي عبر إنشاء مشروع لإدارة البيانات باستخدام FPGA و VHDLويأتي هذا المشروع كفرصة عملية لفهم كيفية تصميم أنظمة رقمية تعالج البيانات في الزمن الحقيقي (Real-Time) وتستجيب للأحداث الخارجية بطريقة فعالة وموثوقة.

إنشاء مشروع متكامل لإدارة البيانات باستخدام FPGA ولغة توصيف الدوائر المنطقية VHDL . يهدف هذا المشروع إلى فهم معمّق لكيفية تصميم نظام رقمي قادر على استقبال البيانات، معالجتها، تخزينها أو توجيهها، اعتماداً على شروط محددة ومخططات مدروسة.

سنقوم بتقسيم المشروع إلى مراحل واضحة تشمل: تحليل المتطلبات، تصميم مخطط البنية، كتابة كودVHDL ، المحاكاة والتحقق من صحة السلوك، ثم تحميل التصميم على لوحة FPGA واختباره عملياً. هذا العمل التطبيقي سيمكن من ربط المعارف النظرية بتطبيقات ملموسة في مجال إدارة البيانات والمعالجة الرقمية.

كمثال تطبيقي على هذا النوع من المشاريع، سنقترح تصميم نظام رقمي لإدارة موقف سيارات .يهدف هذا النظام إلى مراقبة عدد السيارات الداخلة والخارجة من الموقف، وعرض عدد الأماكن المتوفرة، مع إمكانية التحكم في الدخول عند امتلاء الموقف.

سيتم بناء هذا النظام باستخدام:

- وحدة معالجة على لوحة FPGA تقوم بحساب عدد السيارات والتحكم في منطق النظام.
- شاشة عرض 7-Segment Display عدد الأماكن المتاحة.
- كود VHDL لتوصيف سلوك النظام والتحكم في العمليات المختلفة بشكل متزامن.

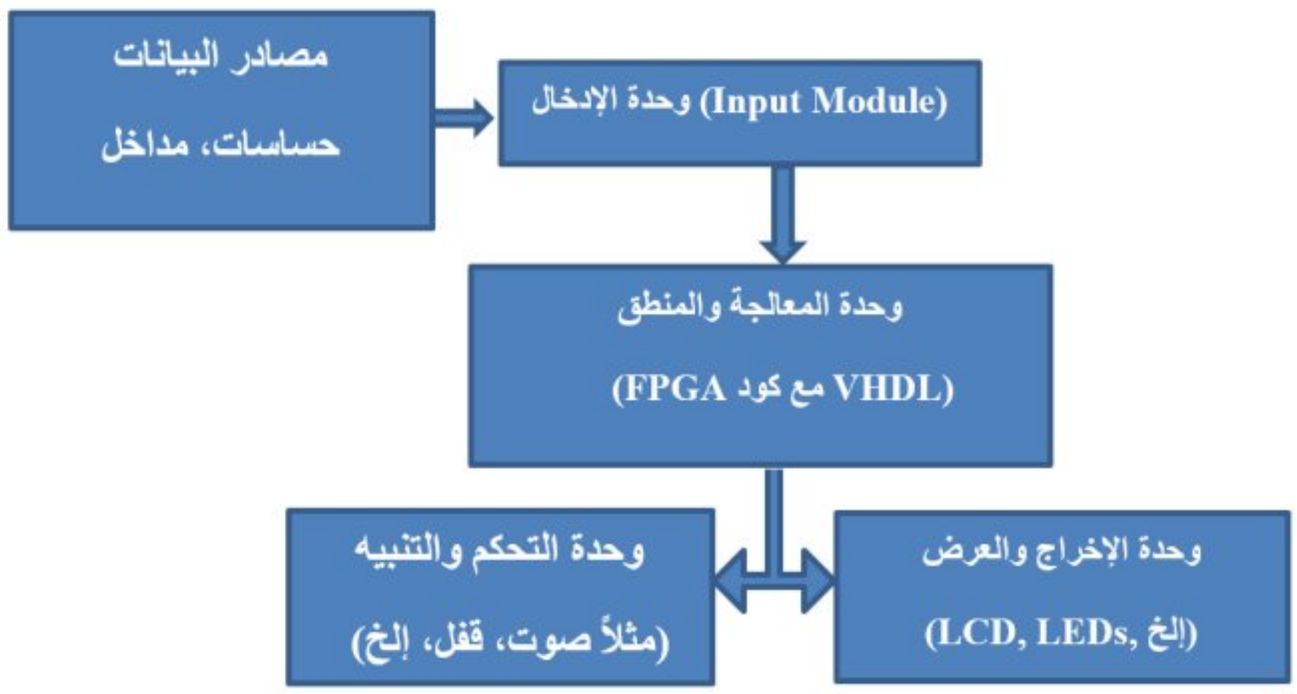
من خلال هذا المشروع، سيتعرف على كيفية:

- تحليل متطلبات النظام الرقمي.
- تصميم منطق التحكم في الموقف.
- توصيف العمليات بلغة VHDL.
- محاكاة النموذج واختباره على لوحة FPGA.

2.III. مخطط عام لنظام إدارة البيانات باستخدام FPGA :

إدارة البيانات هي عملية تنظيم، مراقبة، ومعالجة البيانات لضمان دقتها وتناسقها وفعاليتها ضمن نظام معين. عند استخدام FPGA ، يتم تحقيق هذه المهام عبر تصميم مكونات رقمية قادرة على استقبال البيانات من المصادر الخارجية، معالجتها وفق منطق محدد، وتوجيهها نحو وجهاتها المناسبة (مثل العرض، التخزين، أو التحكم).

في السياق العام، يمكن تبسيط مخطط إدارة البيانات كما يلي:



شكل III.01: إدارة البيانات باستخدام FPGA.

شرح المخطط (شكل **III.01**) إدارة البيانات باستخدام FPGA

1. **مصادر البيانات (حساسات، مداخل):** تمثل النقطة التي تبدأ منها البيانات بالدخول إلى النظام. قد تكون هذه المصادر:
   - حساسات حركة، حرارة، ضوء، مسافة... إلخ.
   - مفاتيح إدخال (Switches).
   - إشارات رقمية واردة من أنظمة أخرى.
2. **وحدة الإدخال:(Input Module)** تقوم هذه الوحدة بتهيئة البيانات القادمة من المصادر (مثلاً: إزالة الضوضاء أو تحويل الإشارة) وتجهيزها لتُرسل إلى وحدة المعالجة داخل FPGA.
   - يتم هنا تحويل الإشارات التناظرية (إن وجدت) إلى إشارات رقمية.
   - تعتبر حلقة وصل بين البيئة المادية (hardware) والمنطق البرمجي.
3. **وحدة المعالجة والمنطق مع كود(FPGA VHDL) :**هي القلب النابض للنظام، حيث تتم فيها:
   - تحليل البيانات القادمة من وحدة الإدخال.
   - تنفيذ العمليات المنطقية بحسب التعليمات المبرمجة بلغة VHDL
   - اتخاذ القرارات مثل عدد السيارات، تجاوز السعة، تشغيل إنذار، إلخ.
   - إرسال نتائج المعالجة إلى الوحدات الأخرى للتحكم أو العرض.
4. **وحدة الإخراج والعرض ,(7-Segment, LEDs…) :**

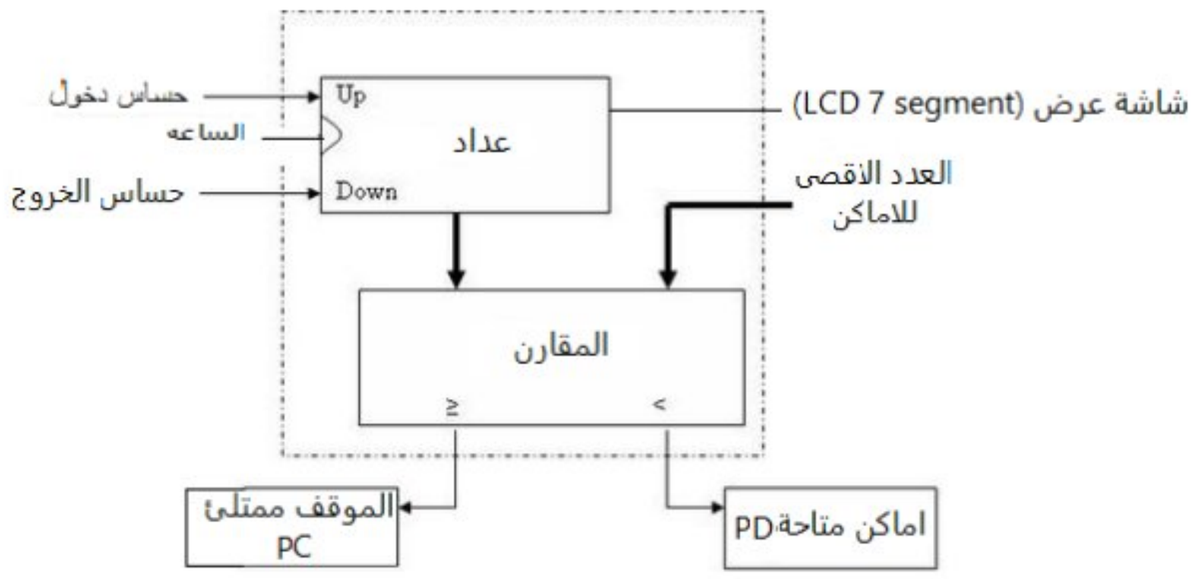
- تعرض معلومات النظام للمستخدم بشكل مرئي، مثل عدد السيارات المتبقية في موقف، رسائل تنبيه، أو إشارات ضوئية.
- يمكن أن تشمل:
  - شاشات رقمية (7-Segment) .
  - مصابيح LED .
  - شاشات رسومية.

5. **وحدة التحكم والتنبيه (مثلاً: صوت، قفل، إلخ):**
   - تنفذ إجراءات بناءً على القرارات المنطقية (مثل قفل البوابة، إطلاق إنذار صوتي، تفعيل محرك، إلخ).
   - تعبر عن استجابة النظام لحالة حرجة أو تجاوز شرط معيّن (مثلاً، الموقف ممتلئ).

هذا المخطط يُظهر دورة حياة البيانات في النظام، من لحظة استشعارها وحتى اتخاذ إجراء مناسب بناءً على معالجتها داخل الـ FPGA. كل مكون يؤدي وظيفة محددة لضمان تدفق البيانات، معالجتها، واتخاذ قرارات آلية في الزمن الحقيقي.وسنستعمل هذا المخطط في إدارة موقف سيارات.

3.III. العمل التطبيقي :

1.3.III. تصميم نظام لادارة موقف سيارات باستخدام VHDL :



شكل 02.III : نظام إدارة موقف سيارات رقمي مع شاشة عرض للأماكن المتاحة.

1.1.3.III. شرح النظام:

يهدف نظام إدارة موقف السيارات إلى تتبُّع عدد السيارات داخل الموقف والتحكّم في إشارات الموقف « ممتلئ » و« أماكن متاحة » بناءً على السعة القصوى المحددة (عدد الأماكن الأقصى)، وذلك عبر المكوّنات التالية: يدخل مستشعر الدخول والخروج (Up وDown) إلى عدّاد ثنائي بعرض 16 بت أو أكثر، حيث ترفع نبضة الدخول قيمة العداد بمقدار 1 وتنقصها نبضة الخروج بمقدار 1، وكل ذلك يتمّ تحت تزامن إشارة الساعة (clock) لضمان دقة التعديلات.

بعد ذلك، يُستخدم مكوّن شاشة عرض (7-Segment) مرتبط مباشرة بالعداد والسعة القصوى، ليعرض في الزمن الحقيقي عدد الأماكن المتبقية داخل الموقف. يتم حساب هذا العدد عبر طرح قيمة العداد الحالية من العدد الأقصى للمواقف، مما يوفر واجهة مرئية للمستخدمين أو للمراقبة التقنية.

ثم يُقارن المكوّن المقارن (Comparator) قيمة العداد بالعدد الأقصى للأماكن؛ فإذا بلغ العداد القيمة العظمى أصبحت إشارة « Parking Complete (PC) » مساوية لـ"1" بينما تصبح إشارة « Places Disponibles (PD) » مساوية لـ"0" للدلالة على اكتمال الموقف، أما إذا كانت قيمة العداد أقل من السعة فتكون "PD=1" للدلالة على وجود أماكن شاغرة و"PC=0".

وهكذا يوفِّر النظام تحكّماً رقمياً دقيقاً وسلساً في مخرجات إشغال الموقف باستخدام وصف VHDL وتنفيذه على FPGA، إلى جانب واجهة مرئية تسهّل قراءة عدد الأماكن المتوفرة بشكل مباشر عبر شاشة العرض.

2.3.III. انجاز المشروع :

لانجاز هذا المشروع نحتاج الى اتباع الخطوات التالية :

1.2.3.III. التنفيذ العملي (MANIPULATION) :

تم إجراء التنفيذ باستخدام برنامج Quartus .

حيث بدأنا بتشغيل البرنامج بالنقر عليه من سطح المكتب، أو من خلال النقر على:

- **ابدا (Start)**
- **البرامج (Programs)**
- **ALTERA**
- **Quartus II 9.0**

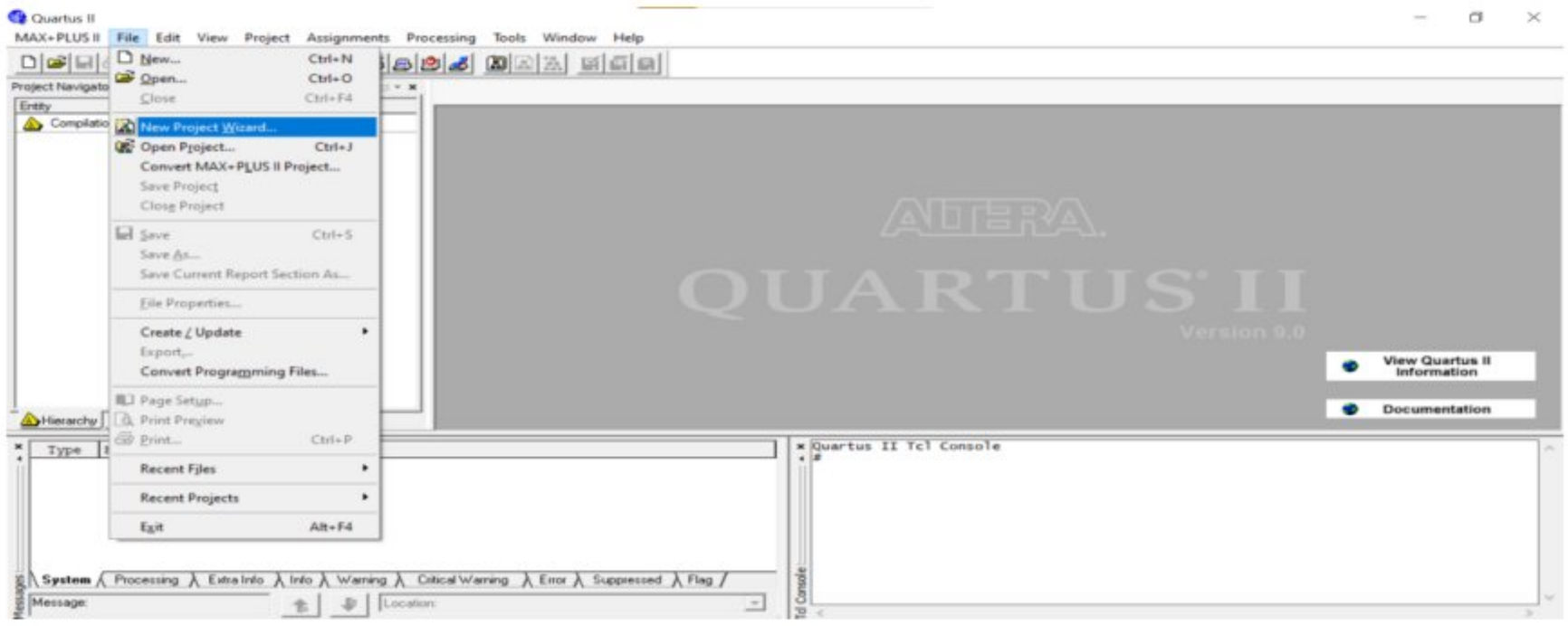
فتحت نافذة Quartus II .

2.2.3.III. إنشاء مشروع جديد (Creating a new project) :

لإنشاء مشروع جديد، ذهبنا إلى القائمة:

- **ملف (File)**
- **معالج المشروع الجديد (New Project Wizard)**

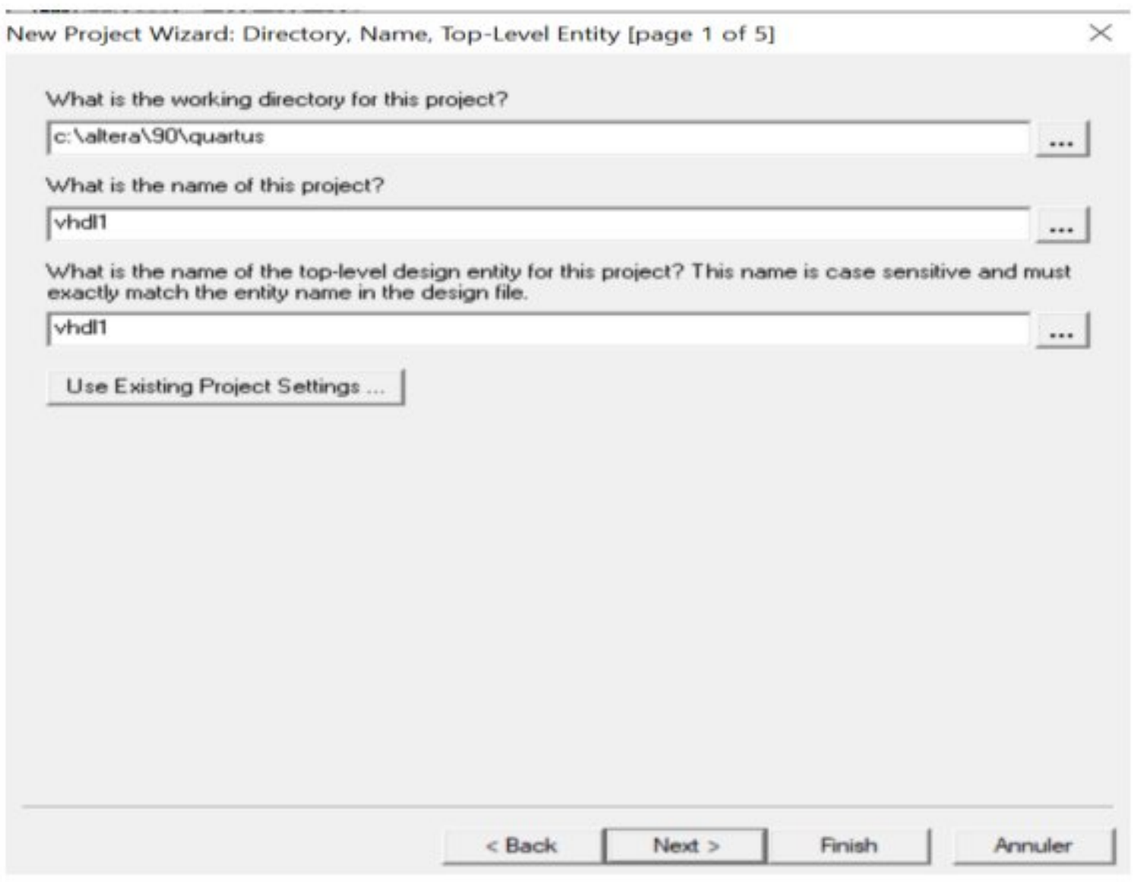
كما موضح في **الشكل 03.III:**



شكل 03.III : انشاء مشروع جديد.

تم فتح نافذة " **New: Introduction** " لتقديم شرح موجز، ثم ضغطنا على: التالي **(Next)**.

النافذة " **New Project Wizard : Directory , Name , Top-level Entity [Page 1 of 5]** " سمح لنا بتكوين المشروع عن طريق الحقول الثلاثة التالية **(شكل 04.III)** :



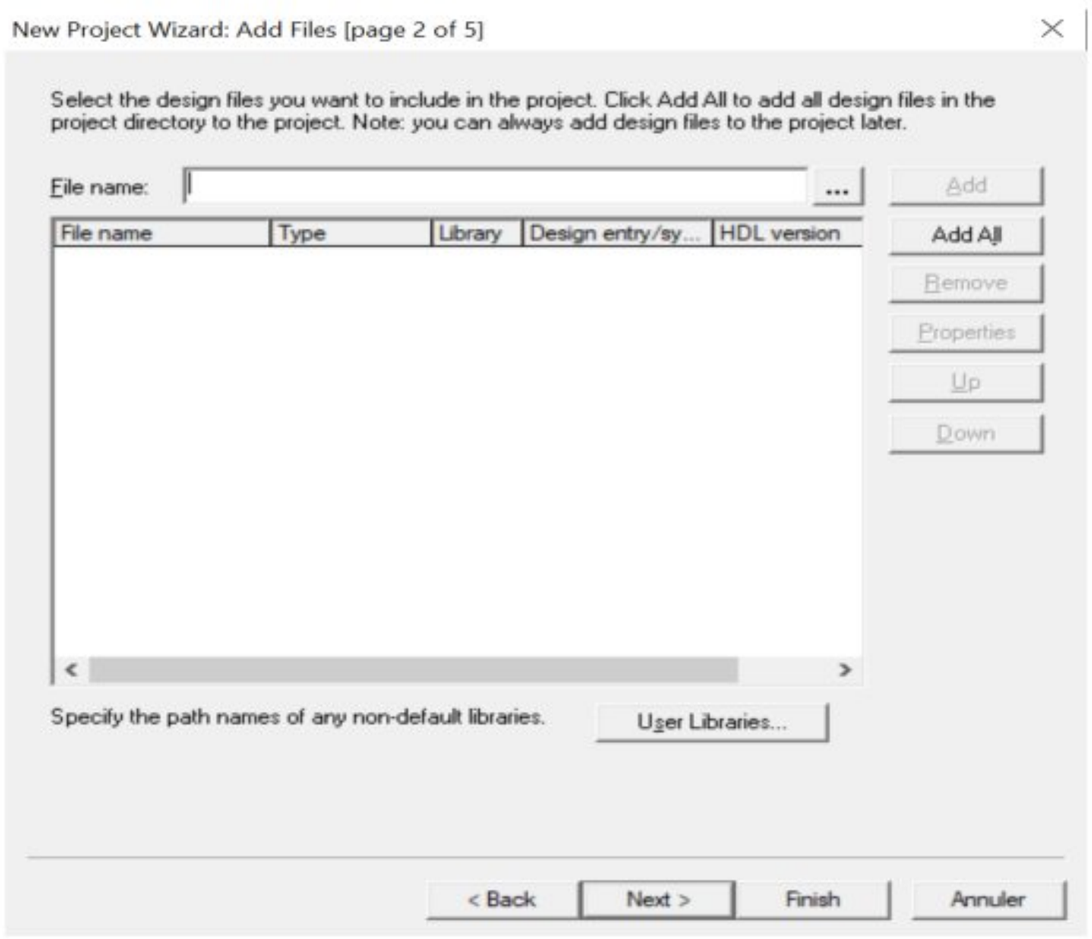
شكل 04.III : تكوين المشروع.

بعد ذلك، قمنا بالنقر على التالي **(Next)** ظهرت نافذة تُعلمنا بأن المجلد **(directory)** الذي اخترناه غير موجود، وطرحت علينا سؤالًا: هل ترغب في إنشائه؟

ننقر على نعم **(OUI)** لإنشائه، أو لا **(NON)** لإلغاء العملية.

ظهرت بعد ذلك صفحة **"New Project Wizard "** : إضافة ملفات **(Add Files)** [الصفحة 2 من 5].

في هذه المرحلة، ليس هناك ملفات لإضافتها، لذلك ضغطنا مرة أخرى على التالي **(Next)** لتظهر نافذة معالج إنشاء مشروع جديد: إعدادات العائلة والنوع **(Family & Device Settings)** [الصفحة 3 من 5] على الشاشة. كما في **شكل 05.III:**

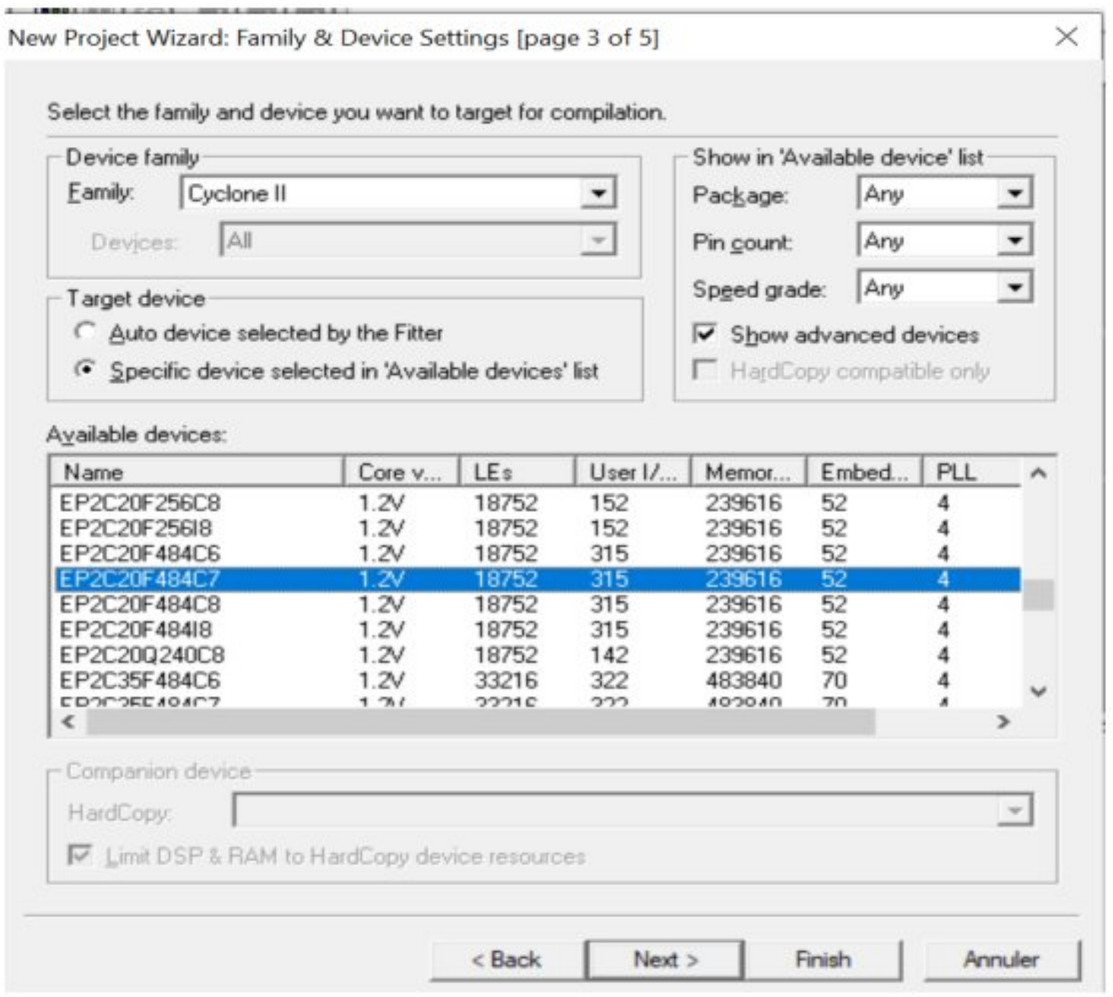


شكل 05.III: إضافة الملفات.

قمنا باختيار الدارة المنطقية القابلة للبرمجة المستهدفة والتي تم استخدامها أثناء عملية التنفيذ (التحميل على العتاد) وأثناء الترجمة **(compilation)** وغيرها من المراحل.

بدأنا أولاً باختيار العائلة **(Family)** ، ثم اخترنا الدارة التي تنتمي إلى هذه العائلة من الأجهزة المتاحة **(Available Devices)** كما في **شكل 06.III**.

إذا كنا نملك لوحة تطبيقات عملية (مثل DE0 أو DE1 أو DE2) لاستخدامها في التنفيذ، يجب علينا اختيار الدارة المنطقية القابلة للبرمجة الموجودة على تلك اللوحة. لذلك، من الضروري تحديد هذه الدارة من البداية، مع العلم أنه يمكننا ان نعدلها لاحقاً إذا لزم الأمر.



شكل 06.III: اختيار العائلة من الأجهزة المتاحة.

بعد ذلك، يمكننا النقر على التالي **(Next)** .

ظهرت لنا نافذة **"New Project Wizard"** : إعدادات أدوات EDA [الصفحة 4 من 5]، لكنها غير مستخدمة في الوقت الحالي، لذلك ضغطنا على التالي **(Next)**.

ظهرت لنا بعد ذلك صفحة **"New Project Wizard"**: الملخص **(Summary)** [الصفحة 5 من 5] ، والتي تتيح لنا معاينة ملخص المشروع،(مثل: مجلد المشروع، اسم المشروع، الكيان العلوي، وتعيينات الدارة مثل اسم العائلة ونوع الدارة).

عندما قمنا بالنقر على إنهاء **(Finish)**، رأينا في متصفح المشروع **(Project Navigator)** على يسار نافذة البرنامج ما يلي:

3.2.3.III. ادخال النص (Text input):

نأخذ توصيف بوابة vhdl1 باستخدام لغة VHDL قمنا بإدخال مكوّن موصوف بلغة VHDL بنفس الطريقة.

وللقيام بذلك، ذهبنا إلى القائمة وقمنا بالنقر على:

- **ملف (File)**
- **جديد (New)**
- **ملف VHDL (VHDL File)**

عندما فتحنا نافذة New ، اخترنا **"VHDL File"** :

ظهرت لنا نافذة محرر VHDL، وهي المكان الذي يتم فيه كتابة كود برنامج VHDL.

ثم كتبنا برنامج VHDL لبوابة vhdl1 كتالي:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity Vhdl1 is
Port)
    clk       : in STD_LOGIC;
    reset     : in STD_LOGIC;
    sensor_in  : in STD_LOGIC;
    sensor_out : in STD_LOGIC;
```

```
        PD      : out STD_LOGIC;
        PC      : out STD_LOGIC;
        seg1    : out STD_LOGIC_VECTOR(6 downto 0)الآحاد -- ;
        seg2    : out STD_LOGIC_VECTOR(6 downto 0)العشرات --
);
end Vhdl1;

architecture Behavioral of Vhdl1 is
    constant MAX_PLACES : integer := 16;
    signal counter : integer range 0 to MAX_PLACES := 0;
    signal ones    : integer range 0 to 9;
    signal tens    : integer range 0 to 9;
begin

عملية العد --
    process(clk, reset)
    begin
        if reset = '1' then
            counter <= 0;
        elsif rising_edge(clk) then
if sensor_in = '1' and counter < MAX_PLACES then
    counter <= counter + 1;
elsif sensor_out = '1' and counter > 0 then
    counter <= counter - 1;
```

```
        end if;
    end if;
end process;

-- إشارات توفر أو امتلاء المواقف
    PD <= '1' when counter < MAX_PLACES else '0;'
    PC <= '1' when counter = MAX_PLACES else '0;'

-- تقسيم الرقم إلى آحاد وعشرات
    ones <= counter mod 10;
    tens <= counter / 10;

-7 عملية تحويل خانة الآحاد إلى Segment --
    process(ones)
    begin
        case ones is
            when 0 => seg1 <= "1000000;"
            when 1 => seg1 <= "1111001;"
            when 2 => seg1 <= "0100100;"
            when 3 => seg1 <= "0110000;"
            when 4 => seg1 <= "0011001;"
            when 5 => seg1 <= "0010010;"
            when 6 => seg1 <= "0000010;"
            when 7 => seg1 <= "1111000;"
```

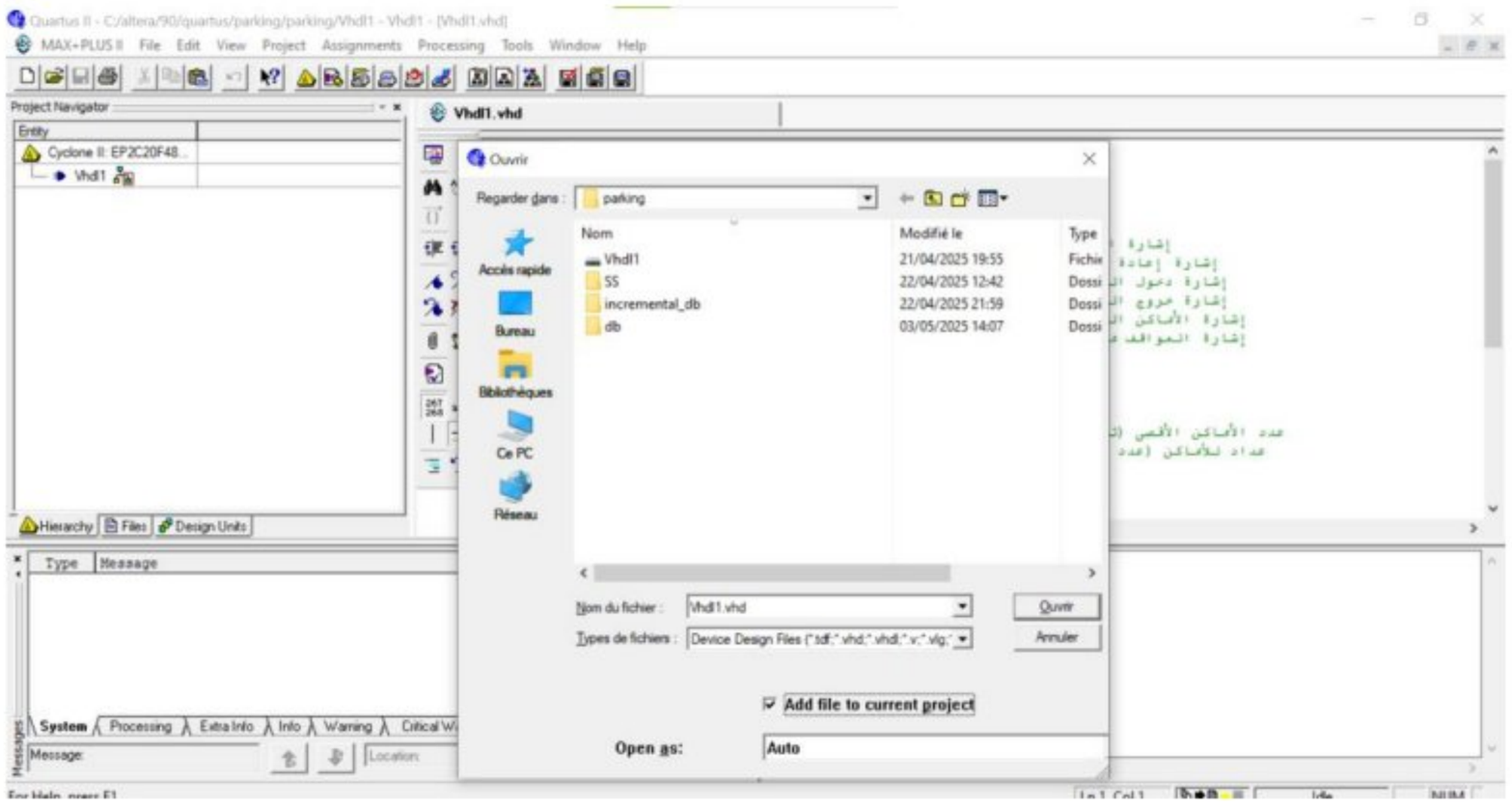
```
        when 8 => seg1 <= "0000000;"
        when 9 => seg1 <= "0010000;"
        when others => seg1 <= "1111111;"
    end case;
end process;

-- عملية تحويل خانة العشرات إلى 7- Segment
    process(tens)
    begin
        case tens is
            when 0 => seg2 <= "1000000;"
            when 1 => seg2 <= "1111001;"
            when others => seg2 <= "1111111 -- فارغ أو لا شيء ";
        end case;
    end process;
end Behavioral;
```

قمنا بحفظ هذا الملف، وللقيام بذلك نتبع الخطوات التالية:

- **ملف (File)**
- **حفظ باسم (Save As)**

في نافذة **"Save As"**، قمنا بإعطاء اسم للملف مع الامتداد vhd ، على سبيل المثال: **vhdl1.vhd**.



شكل 07.III: تسمية الملف.

مع عدم نسيان تفعيل الخيار **"Add File to current project"** قبل الحفظ، وذلك لضمان إضافة الملف إلى المشروع الحالي.

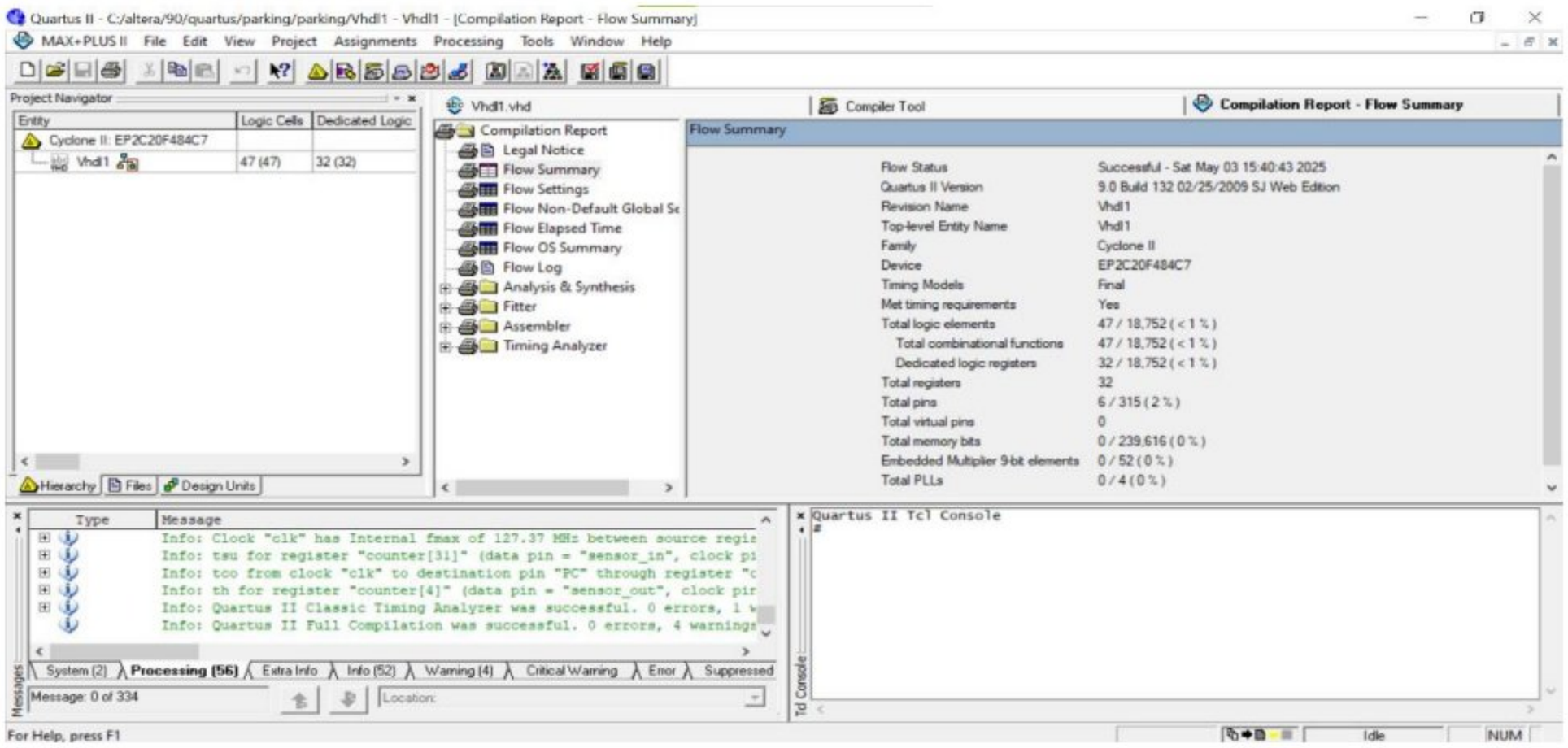
4.2.3.III. تجميع النص المدخل (Compiling text input) :

يجب أن يتم تجميع النص المدخل **(compilation)** لتصحيح أخطاء VHDL أو الأخطاء الأخرى.

لبدء التجميع ، قمنا بالنقر على**" Processing "** ثم على **" Start ".**

أو يمكننا اختيار: **" Processing → Compiler Tools "** ، ومن نافذة **" Compiler Tool "** ، ثم الضغط على **" Start Compiler ".**

عند انتهاء التجميع ، ظهرت لنا نافذة **" Compilation Report → Flow Summary "** ، وهي بمثابة تقرير ملخّص لنتائج عملية التجميع. كما في **شكل 08.III:**



شكل 08.III: نتائج عملية التجميع.

* إذا فشل التجميع ، يجب تصحيح الأخطاء التي تظهرلنا ثم إعادة تشغيل التجميع من جديد.

* إذا نجح التجميع ، نواصل باقي العمل.

مشاهدة المخطط في **RTL Viewer** :

بعدما تأكدنا من أن ملف VHDL خالٍ من الأخطاء، يمكننا إنشاء رمز رسومي (Symbol) يسمح لنا بإدراج هذا المكوّن في ورقة الرسم البياني.

وللقيام بذلك، من قائمة Quartus ، اختارنا :

- **ملف (File)**
- **انشاء/تحديث (Create/Update)**
- **انشاء ملف الرموز من الملف الحالي (Create Symbol File From Current File)**

ظهرت لنا نافذة تؤكد نجاح العملية برسالة : **"Create Symbol was Successful"**

ويتم حفظ الرمز في مكتبة **(WORK)** ، ليكون جاهزاً للاستخدام عند الحاجة .

لإضافة ملفات إلى المشروع ومشاهدة محتواه :

هناك طريقتان:

من القائمة : **Project → Add/Remove Files in Project**

أو من خلال : **Assignments → Settings**

ومن نافذة **" Settings "** ، اختارنا قسم **" Files "** لعرض جميع الملفات المضمّنة حالياً في المشروع ، ومن هناك يمكننا إضافة ملفات جديدة أو حذف ملفات غير مرغوب فيها.

III.5.2.3. المحاكاة(Simulation):

يُقدِّم برنامج Quartus نوعين من المحاكاة:

* **المحاكاة الوظيفية (Functional) :** وهي التي لا تأخذ في الاعتبار أزمنة الانتشار(Propagation Times)

* **المحاكاة الزمنية (Timing) :** وهي التي تأخذ أزمنة الانتشار بعين الاعتبار.

يرافق المحاكاة تعريف متجهات المحاكاة (Simulation Vectors) .

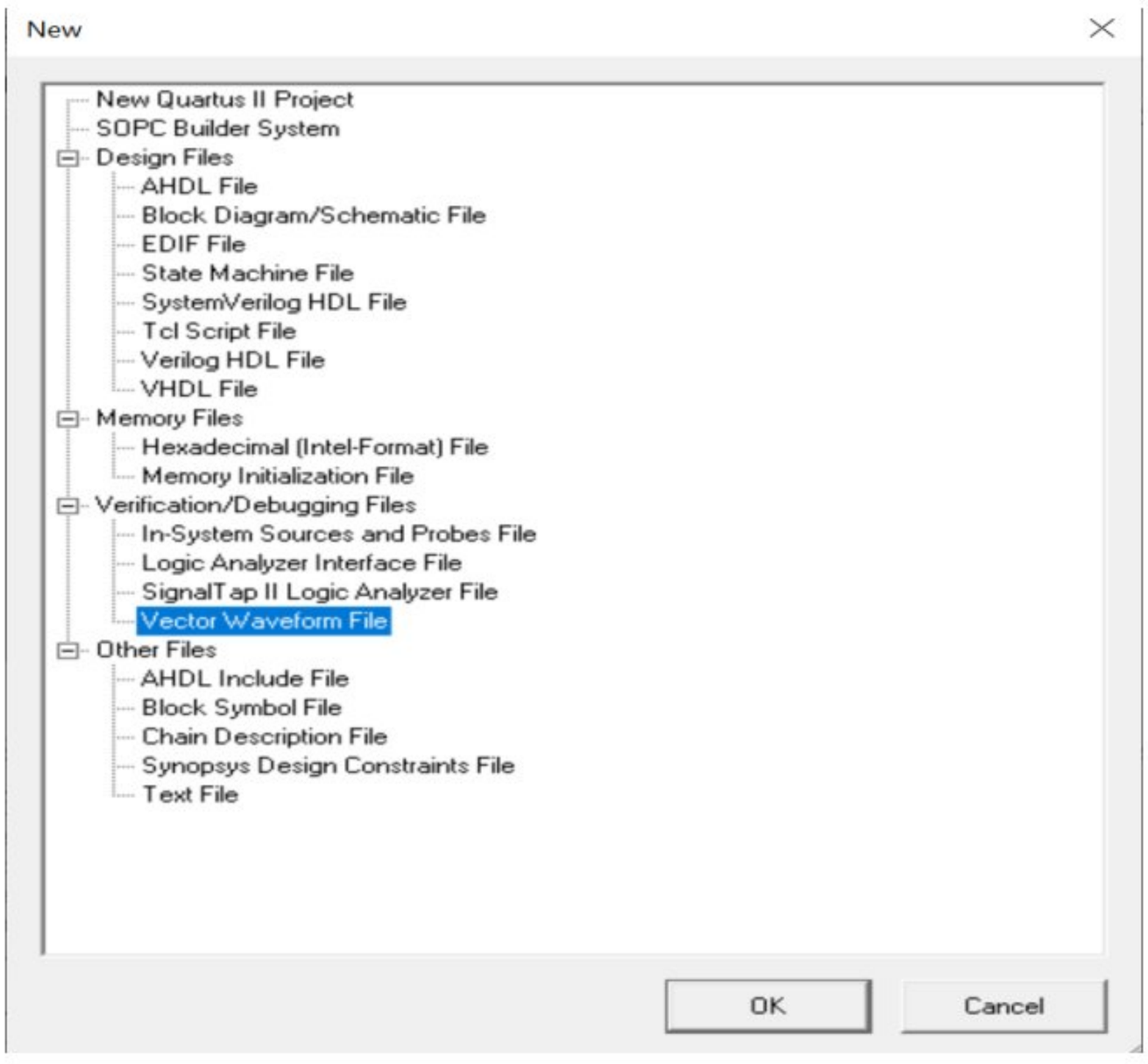
يُوفّر برنامج Quartus إمكانية محاكاة المخططات. وقبل بدء المحاكاة، يجب تعريف متجهات الاختبار، أي تحديد الإشارات التي ستُطبّق على المداخل.

ولمحاكاة التصميم المُنجز، قمنا بحقن بمثيرات(Stimuli) ، عندما يتم توليد هذه المثيرات من ملف **"Bench "**

قمنا بتوليد المثيرات باستخدام **Wave Editor** .

اختارنا من القائمة **File > New** : وعندما ظهرت نافذة **" New "** ، حددنا **" Vector Waveform File "**

ثم قمنا بضغط على **"OK "** . كما في **شكل III.09:**



شكل III.09: نافذة اطلاق المحاكاة Vector Waveform File.

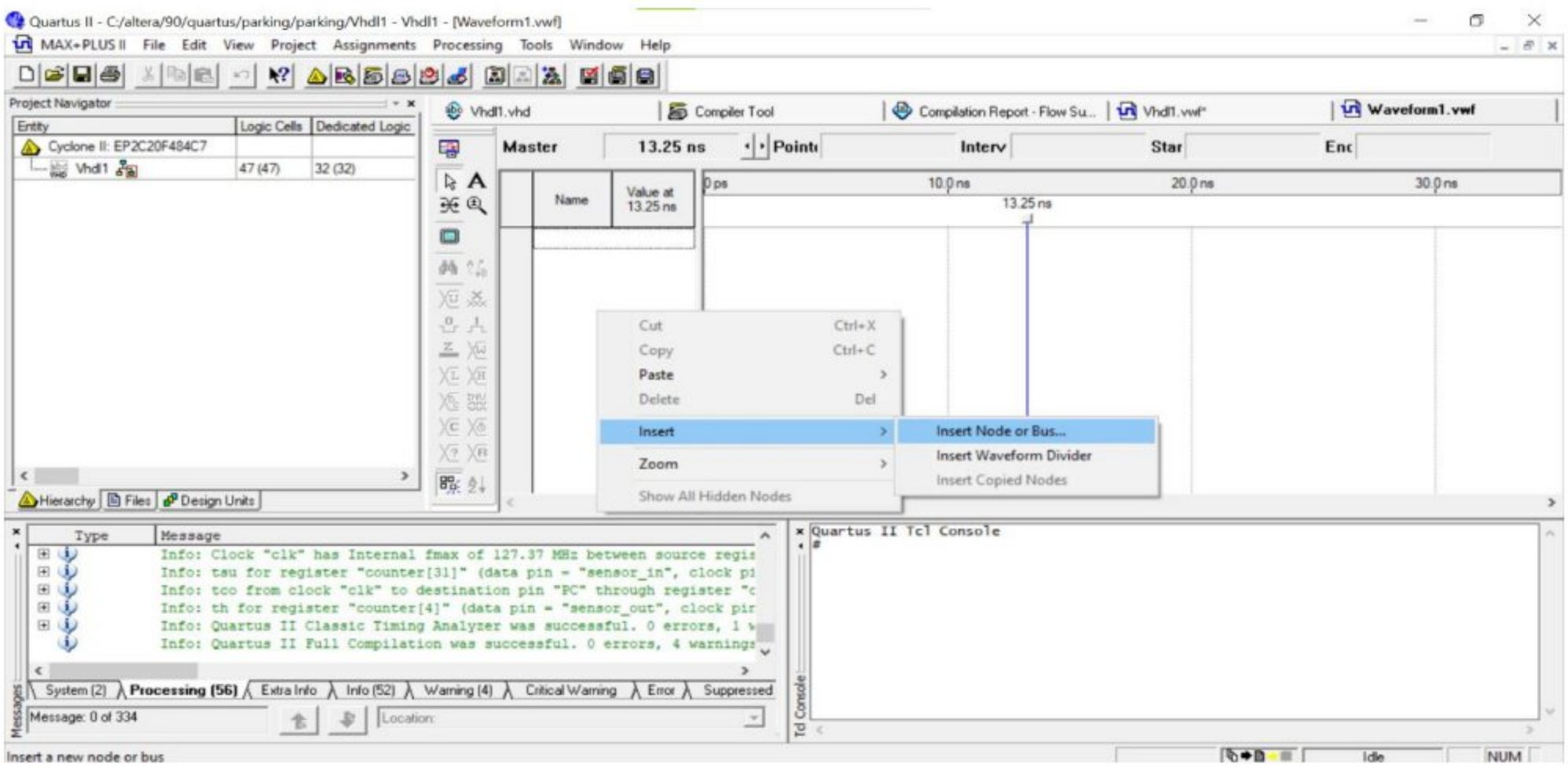
ظهرلنا نافذة " **Wave Form 1.vwf** " ، وسمحت لنا بحفظ ملف متجهات الاختبار.

قمنا باستبدال الاسم الافتراضي مثلاً بـ **(Simul\vhdl1.vwf)** لتحديد هذا الملف (مع عدم نسيان الامتداد vwf) ، ثم قمنا بالحفظ باستخدام " **Save as** " من قائمة " **File** " وضفناه إلى المشروع.

لإدراج إشارات المحاكاة المختلفة، أي إشارات الدخل والخروج (مثل Input ، Clk، Output ...)، استخدمنا الأمر:

" **Edit > Insert > Insert Node or Bus** " ، وذلك أثناء فتحنا ملف متجه الاختبار vwf، ثم ضغطنا على

" **Insert Node or Bus**".



شكل III.10: نافذة Insert Node or Bus.

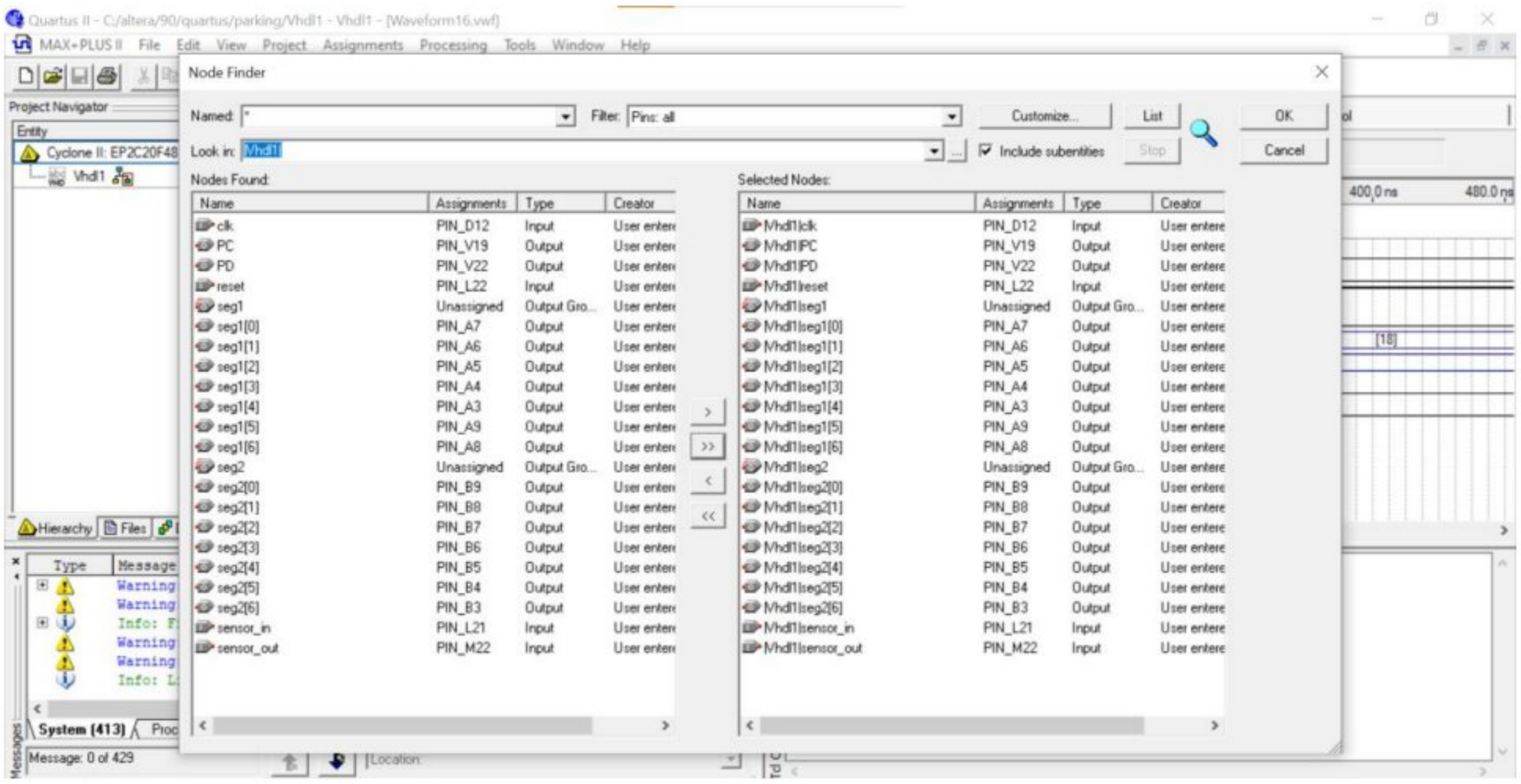
في نافذة " **Insert Node or Bus** " التي ظهرت، ننقر على" **Node Finder...** ".

في **شكل III.11** يظهر النافذة الجديدة " **Node Finder** " ، ذهبنا إلى " **Filter: Pins \:all** " وذلك بالنقر على السهم لإظهار القائمة .

بعد ذلك، ضغطناعلى " **List** " لكي تظهر المداخل والمخارج في العمود الأيسر المسمى " **Nodes Found** ".

الآن، قمنا باختيار المداخل والمخارج التي نريد استخدامها في المحاكاة (في العادة جميع الإشارات)، وذلك من خلال نقلها إلى العمود الأيمن " **Selected Nodes** " باستخدام السهم الفردي أو المزدوج بين العمودين.

ولإزالة الإشارات من " **Selected Nodes** " ، استخدمنا الأسهم في الاتجاه المعاكس.



شكل III.11: نافذة Node Finder.

في حالتنا، لدينا المدخلات " **clk ,reset ,sensor_in ,sensor_out** " والمخرجات " **seg2, seg1, PD,PC** " . باستخدام المخرجات، قمنا بتحديد فترة زمنية معينة واعطيناها القيمة 1 أو 0، بحيث تكون مجموعة قيم الإدخال في المحاكاة كما موضحة في **شكل III.12.**

ظهرت لنا إشارات المحاكاة على شكل مخطط زمني (Chronogramme).

❖ **المحاكاة الوظيفية (Functional Simulation) :**

للقيام بالمحاكاة، اخترنا: **Assignments > Settings > Simulator**

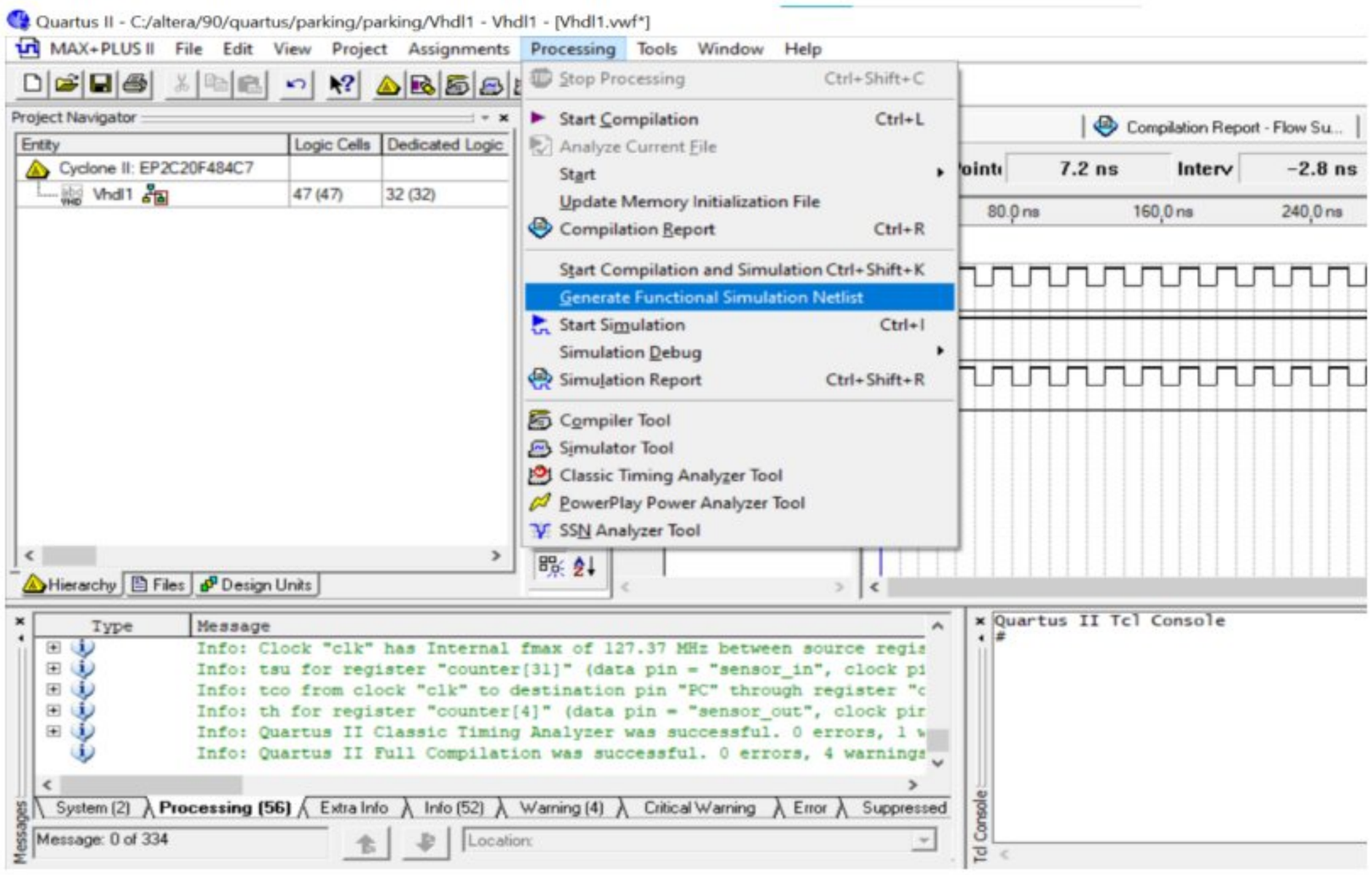
ثم اخترنا **Functional** كوضع المحاكاة (Simulation Mode) ، ثم ضغطنا "**OK** ".

وبعدها، ضغطنا على:

"**Processing > Generate Functional Simulation Netlist** " واكدنا بـ "**OK** ".

وأخيرًا، يتم إطلاق المحاكاة عبر:

" **Processing > Start Simulation** " أو باستخدام الأيقونة الخاصة بذلك.



شكل III.12: إطلاق المحاكاة.

في **الشكل III.13** تظهرلنا نتيجة المحاكاة الوظيفية (Functional Simulation) في برنامج Quartus II باستخدام واجهة الموجات الزمنية vwf بلغة VHDL:

- **شرح المخطط الموجي:**

* clk: إشارة الساعة (Clock)، تستخدم لمزامنة العمليات في الدارة الرقمية. يمكن ملاحظة التغيرات الدورية في التردد.
* PC (Parking Complet): إشارة تفعّل عندما يكون الموقف ممتلئًا (عدد السيارات = 16).
* PD (Places Disponibles): إشارة تفعّل عندما تكون هناك أماكن شاغرة (عدد السيارات < 16).
* reset: إعادة ضبط النظام (تُعيد العداد إلى الصفر).
* sensor_in (دخول): إشارة إدخال تشير إلى دخول سيارة للموقف (ترفع قيمة العداد).
* sensor_out (خروج): إشارة تشير إلى خروج سيارة من الموقف (تنقص العداد).

- **تفصيل عمل النظام في المحاكاة:**

* في بداية المحاكاة: يبدأ النظام مع تفعيل إشارة reset لإعادة ضبط العداد.
* مع تفعيل إشارة sensor _in : نلاحظ زيادة تدريجية في العدد (غير ظاهر رقميًا هنا ولكن يمكن تخمينه من تغير حالة PC و PD).
* عند الوصول إلى الحد الأقصى (16):

PC = 1 → موقف ممتلئ.

PD = 0 → لا توجد أماكن شاغرة.

*عند تفعيل sensor_out: يبدأ النظام بتقليل عدد السيارات.

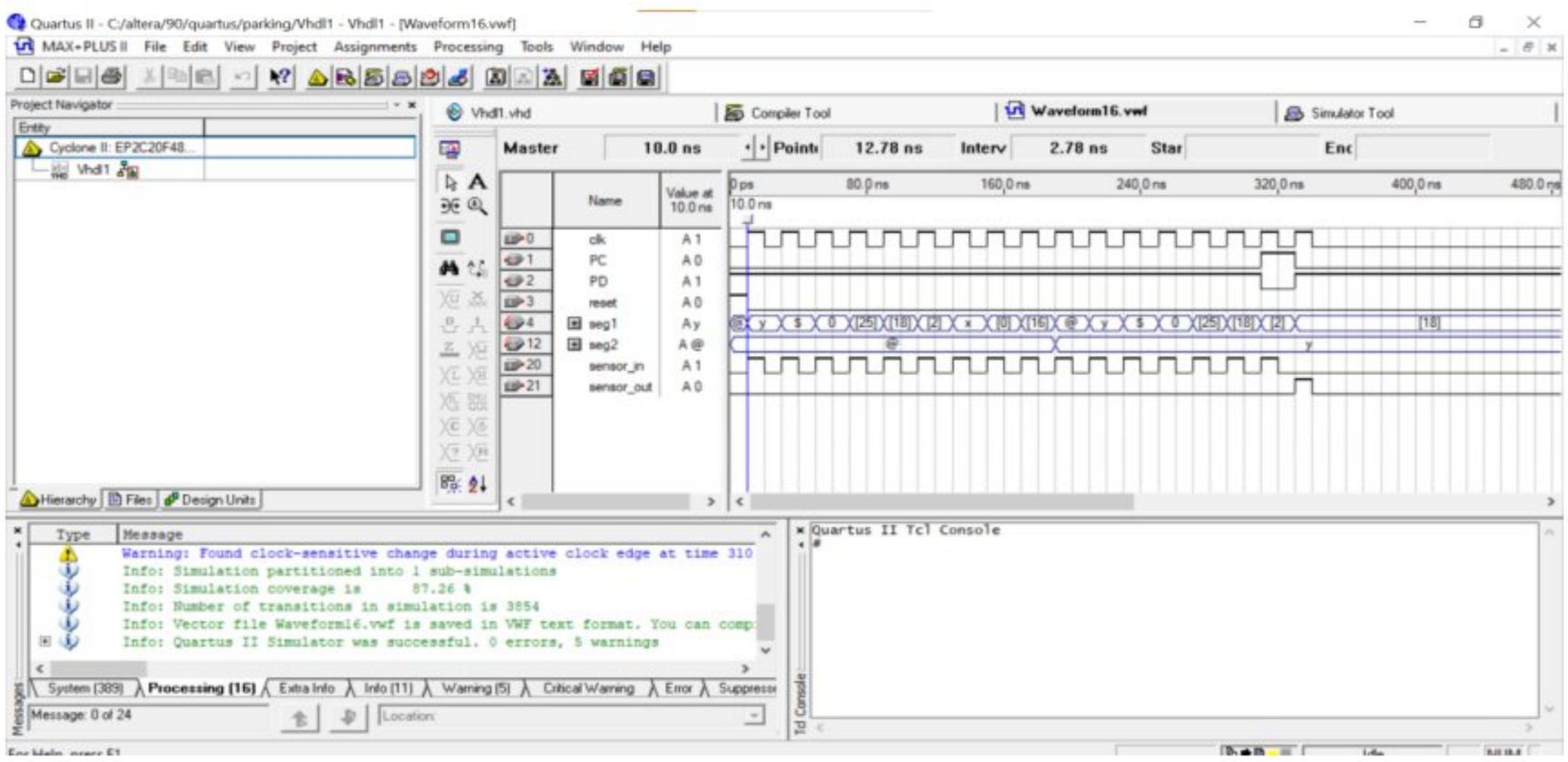
* PD تعود إلى التفعيل (1) عندما يصبح العدد أقل من 16.

لاحصاء عدد السيارات المتبقية اضفنا شاشة عرض 7-Segment:

* تُعرض القيمة الحالية للعداد (من 0 إلى 16) على شاشة 7-Segment.

* يتم تحويل العدد الثنائي إلى ترميز 7 قطع باستخدام Decoder BCD to 7-Segment.

* تُوصل الإشارة الخارجة من العداد مباشرة إلى وحدة العرض.



شكل 13.III: نتيجة المحاكاة الوظيفية.

يمثل **الشكل 14.III** إشارات seg1 حيث:

* seg1 تمثل خانة الآحاد في شاشة العرض 7-segment.

* وهي عبارة عن ناقل من 7 بتات: seg1(6 downto 0).

* تُستخدم لعرض الأرقام من 0 إلى 9 بناءً على قيمة ones = counter mod 10.

- **كيف تعمل؟**

في الكود الذي كتبناه، تم تعريف القيم التالية:

| الرقم | seg1 (7-segment) |

0	"1000000"
1	"1111001"
2	"0100100"
3	"0110000"
4	"0011001"
5	"0010010"
6	"0000010"
7	"1111000"
8	"0000000"
9	"0010000"

كل bit من seg1 يتحكم في جزء من الرقم المعروض على الشاشة، وغالبًا الإشارة 0 تعني أن الجزء (segment) مضاء.

- **شرح الإشارة في المحاكاة:**

1. الإشارة sensor_in تتغير كثيرًا (موجات صاعدة):
   يعني أن العدّاد counter يزداد.

2. كلما ازداد counter:

* ones = counter mod 10 يتغير.

* فتتغير قيمة seg1 لتمثل الرقم الجديد (من 0 إلى 9).

3. كيف ترى ذلك؟

* في الموجات، يتغير شكل الإشارات seg1(0) إلى seg1(6) كل مرة يزداد فيها counter.

* هذه التغييرات هي التي تُعبرعن الأرقام الظاهرة على الشاشة.

مثلاً:

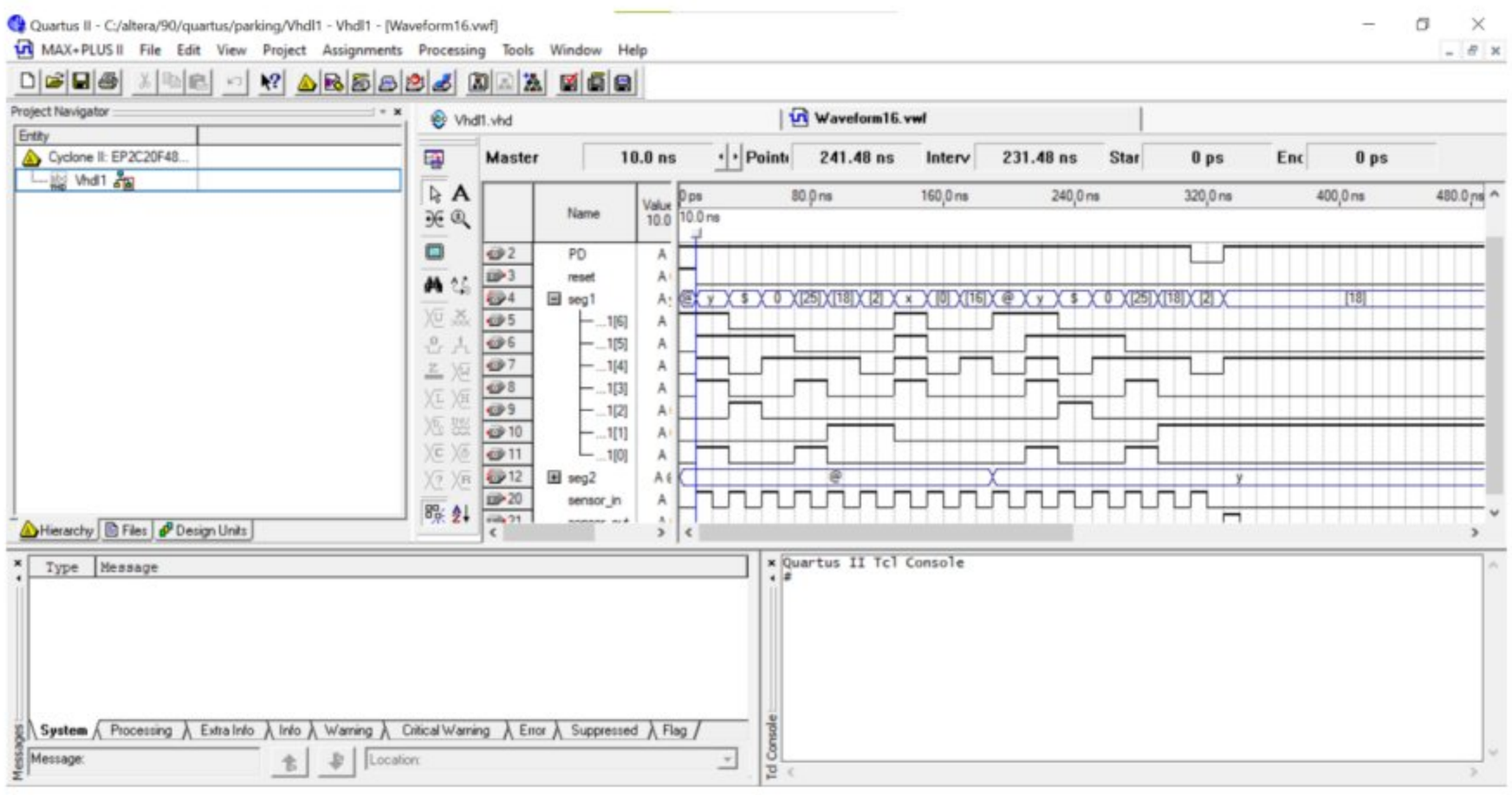
* إذا كانت seg1 = "1000000" → الرقم 0

* إذا أصبحت "1111001" → الرقم 1 وهكذا...

* بمجرد أن يصل counter إلى 10، تبدأ seg1 من جديد بـ 0 لأن:

;ones <= counter mod 10

وبالتالي، فإن seg1 تعمل بشكل دائري من 0 إلى 9 مهما كانت قيمة counter.



شكل 14.III: محاكاة إشارات seg1

يوضح **الشكل 15.III** محاكاة في برنامج Quartus II (نافذة المحاكاة: Waveform Editor).

* الإشارة المسماة seg2 تمثل خانة العشرات في شاشة العرض 7-segment.

* وهي عبارة عن ناقل (bus) من 7 بتات: seg2(6 downto 0).

- **كيف نقرأ seg2؟**

seg2 مقسم إلى إشارات فرعية:

seg2(6), seg2(5), ..., seg2(0)

كل واحدة تمثل جزءاً من الشكل الذي يظهر على شاشة العرض.

في 7-segment، كل خانة تمثل ضوء (segment) من A إلى G.

- **تحليل الإشارة:**

1. عند البداية :

* كل الإشارات تقريبا في حالة 1 (يعني OFF في بعض الشاشات).

* هذا يشير إلى أن شاشة العشرات لا تعرض شيئًا (حالة "1111111" => others").

2. عند أول مرة تنخفض بعض البتات (مثل seg2(3), seg2(0)):

* هذا يعني ظهور رقم معين.

* بالرجوع إلى الكود:

عندما يكون tens = 1 (counter من 10 إلى 19)

القيمة تكون: "1111001" → يعني:

seg2(6) = 1

seg2(5) = 1

seg2(4) = 1

seg2(3) = 1

seg2(2) = 0

seg2(1) = 0

seg2(0) = 1

أي يظهر الرقم 1.

3. ثم تختفي الإشارة (ترجع 1 كلها):

* يعني عودة الشاشة للحالة الفارغة عند tens = 2 أو أكثر.

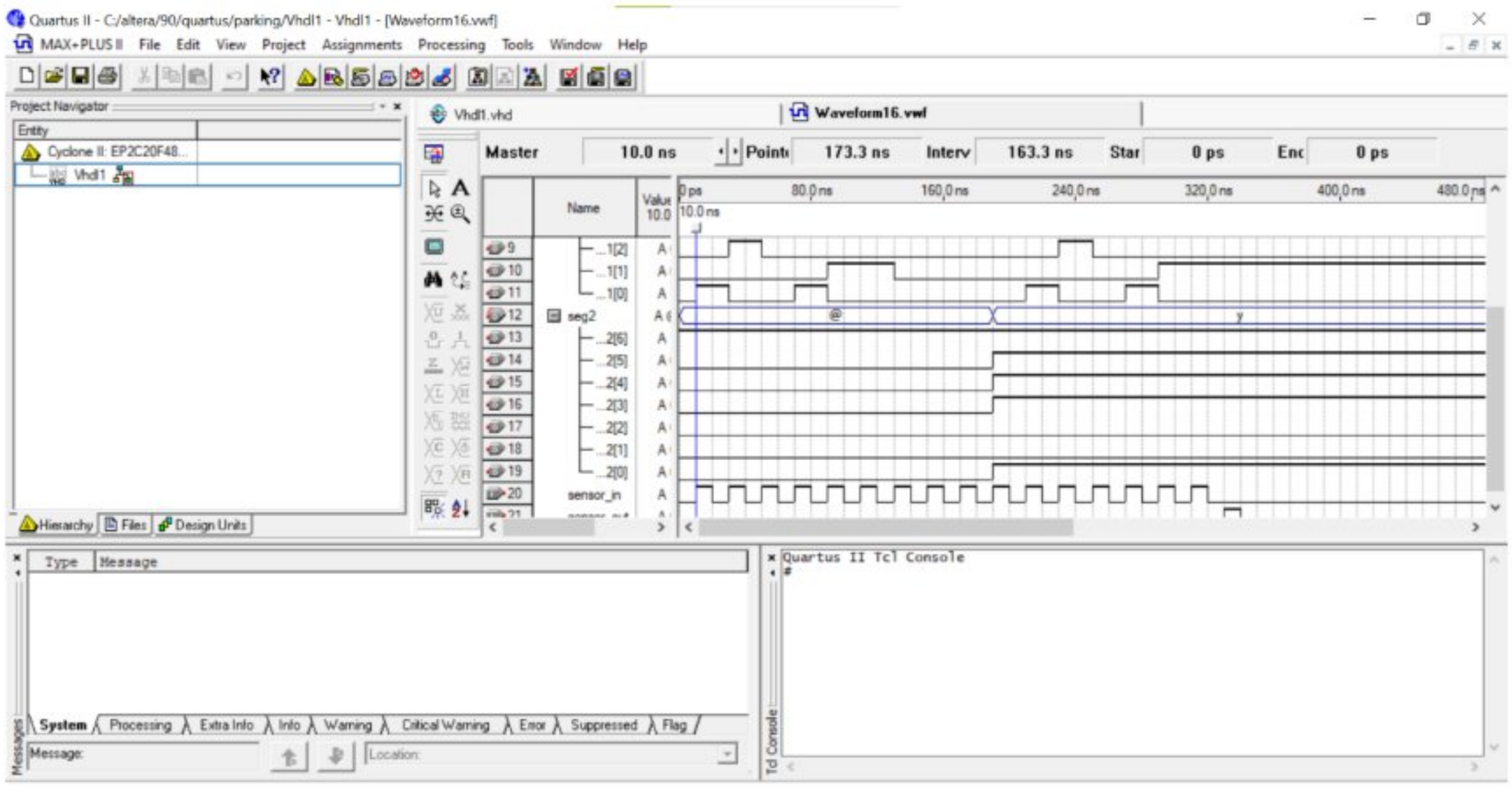
- **الربط مع sensor_in:**

* نرى أن sensor_in يتغير باستمرار (دورات موجة).

* هذا يُزيد العداد counter.

* عندما يصل counter = 10، tens = 1 → يظهر رقم 1 في seg2.

* عندما يتجاوز "counter = 19 → tens = 2 → seg2 = "1111111 (شاشة فارغة لأن الكود لا يدعم 2+).



شكل 15.III: محاكاة إشارات seg2

6.2.3.III. تخصيص الأرجل (Pin assignment):

من أجل اختيار أي رجل (pin) مادي في الدائرة يجب توصيله، قمنا بتشغيل أداة تعيين الأرجل.

من القائمة، قمنا بضغط على **"Assignement "** ، فظهرت ثلاث خيارات ممكنة:

❖ **عند النقر على:"Assignments Editor"**

عند فتح النافذة، وفي خانة Category ، اخترنا Pins.

* قمنا بضغط مرتين على **" New "** تحت عمود **"To "** .

* ظهرت لنا قائمة بجميع المداخل والمخارج (Input , Output) الخاصة بالتصميم.

* اخترنا المدخل أو المخرج الذي نريد تخصيص رجل له.

* ثم ضغطنا مرتين على **" New "** في عمود **"Location "**.

* تظهر قائمة الأرجل الخاصة بدائرة FPGA .

* اخترنا الرجل (pin) المناسبة للمدخل أو المخرج المطلوب.

أعدنا هذه العملية لكل المداخل والمخارج.

في النهاية، قمنا بحفظ تعيين الأرجل عند ظهور نافذة من Quartus II تسأل:

"Do you want to save changes to assignment?"

نختار **" نعم (Yes) "، " لا (No) "** أو **" إلغاء (Cancel) "**حسب الحالة.

❖ **عند النقر على :"Pin"**

عند فتح النافذة، ذهبنا **إلى "Fitter: Pins "** ثم قمنا بضغط على السهم.

يمكننا اختيار أحد الخيارات التالية:

"Pins: Input " لعرض أرجل المداخل فقط.

"Pins: Output " لعرض أرجل المخارج فقط.

"Pins: All " لعرض كل الأرجل.

لكل مدخل أو مخرج، ضغطنا على **"New "** في عمود **"Location "**، ثم اخترنا الرجل المناسبة من قائمة الأرجل الظاهرة.

في النهاية، قمنا بحفظ التغييرات كما في السابق.

❖ **عند النقر على:"Pin Planner"**

عندما فتحنا نافذة :**"Pin Planner "**

*العمود **"Node Name "** عرض لنا قائمة بالمداخل والمخارج.

*العمود الثاني **"Direction "** توضح لنا ما إذا كانت الإشارة input أو output .

*في العمود الثالث **"Location "**، نقرنا مرتين لإظهار قائمة الأرجل.

*اخترنا الرجل المناسبة لكل مدخل أو مخرج.

في العمود **"Location"** لكل إشارة اختر الأرجل المخصصة للشاشات:

لربط شاشة الآحاد (seg1): استهدف كل بيت من HEX0[0..6]

لربط شاشة العشرات (seg2): استهدف كل بيت من HEX1[0..6]

- **إشارات العرض 7-segment:**

العرض الأول (seg1) – خانة الآحاد:

رقم الإشارة	الأرجل (Pins)
PIN_A8	seg1(6)
PIN_A9	seg1(5)
PIN_A3	seg1(4)
PIN_A4	seg1(3)
PIN_A5	seg1(2)

seg1(1)	PIN_A6
seg1(0)	PIN_A7

العرض الثاني (seg2) – خانة العشرات:

رقم الإشارة	الأرجل (Pins)
seg2(6)	PIN_B3
seg2(5)	PIN_B4
seg2(4)	PIN_B5
seg2(3)	PIN_B6
seg2(2)	PIN_B7
seg2(1)	PIN_B8
seg2(0)	PIN_B9

- **إشارات التحكم:**

الإشارة	الأرجل	الاستخدام
Clk	PIN_D12	ساعة النظام
Reset	PIN_L22	إعادة الضبط
sensor_in	PIN_L21	دخول السيارة
sensor_out	PIN_M22	خروج السيارة
PD	PIN_V22	موقف غير ممتلئ
PC	PIN_V19	المواقف ممتلئة

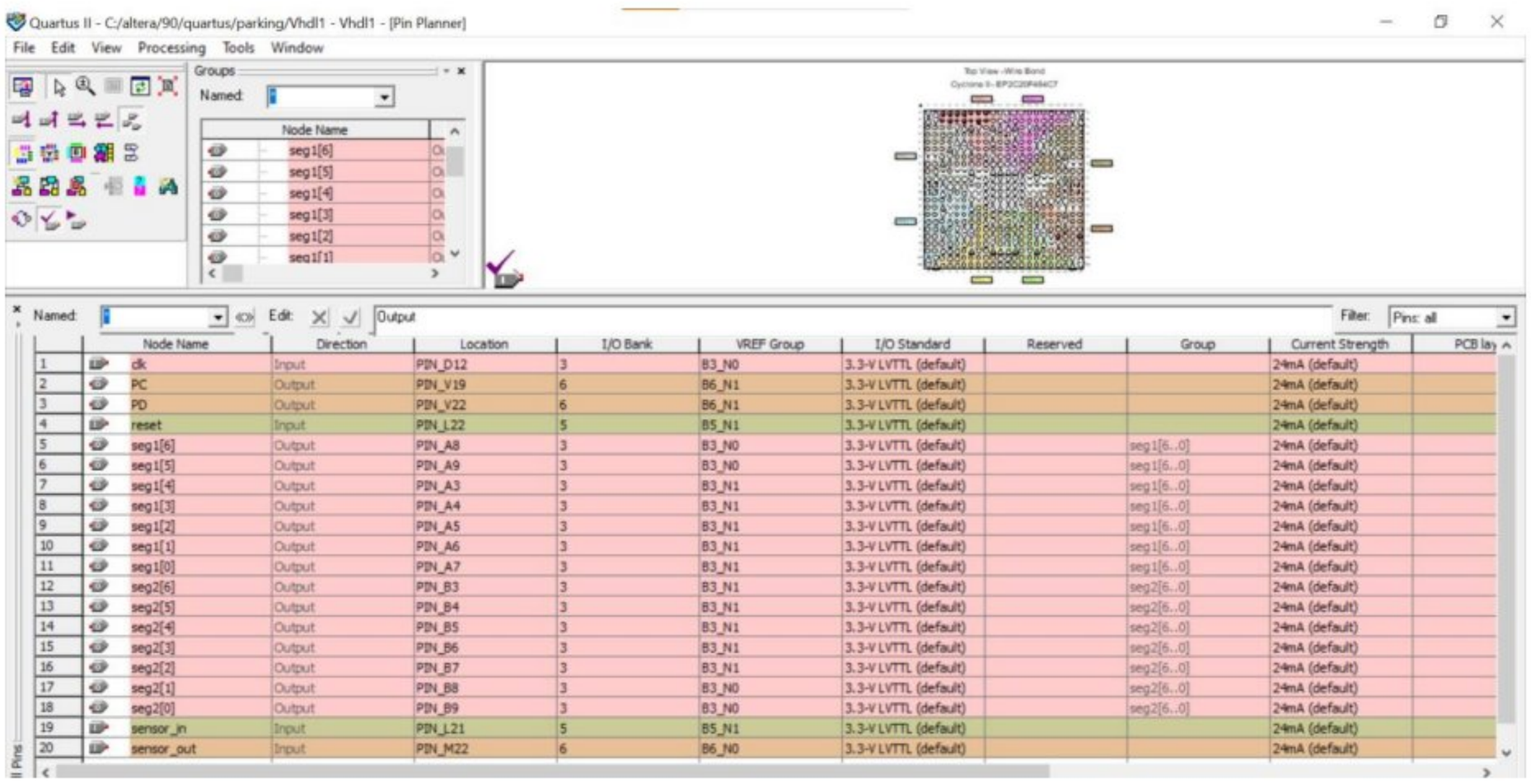
قمنا بملء خانة **"Location "** لكل الإشارات. ثم حفظنا التغييرات كما سبق.

في حالتنا استخدامنا بطاقة **DE1** من (Altera):

*العائلة **Cyclone II** .

*الشريحة (FPGA) **EP2C20F484C7** .

يظهر **الشكل 16.III** جدولًا من برنامج Quartus II يحتوي على أعمدة مهمة توضح كيفية توصيل إشارات التصميم إلى الأرجل (pins) الفعلية على الـ FPGA :



	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Group	Current Strength	PCB lay
1	clk	Input	PIN_D12	3	B3_N0	3.3-V LVTTL (default)			24mA (default)	
2	PC	Output	PIN_V19	6	B6_N1	3.3-V LVTTL (default)			24mA (default)	
3	PD	Output	PIN_V22	6	B6_N1	3.3-V LVTTL (default)			24mA (default)	
4	reset	Input	PIN_L22	5	B5_N1	3.3-V LVTTL (default)			24mA (default)	
5	seg1[6]	Output	PIN_A8	3	B3_N0	3.3-V LVTTL (default)		seg1[6..0]	24mA (default)	
6	seg1[5]	Output	PIN_A9	3	B3_N0	3.3-V LVTTL (default)		seg1[6..0]	24mA (default)	
7	seg1[4]	Output	PIN_A3	3	B3_N1	3.3-V LVTTL (default)		seg1[6..0]	24mA (default)	
8	seg1[3]	Output	PIN_A4	3	B3_N1	3.3-V LVTTL (default)		seg1[6..0]	24mA (default)	
9	seg1[2]	Output	PIN_A5	3	B3_N1	3.3-V LVTTL (default)		seg1[6..0]	24mA (default)	
10	seg1[1]	Output	PIN_A6	3	B3_N1	3.3-V LVTTL (default)		seg1[6..0]	24mA (default)	
11	seg1[0]	Output	PIN_A7	3	B3_N1	3.3-V LVTTL (default)		seg1[6..0]	24mA (default)	
12	seg2[6]	Output	PIN_B3	3	B3_N1	3.3-V LVTTL (default)		seg2[6..0]	24mA (default)	
13	seg2[5]	Output	PIN_B4	3	B3_N1	3.3-V LVTTL (default)		seg2[6..0]	24mA (default)	
14	seg2[4]	Output	PIN_B5	3	B3_N1	3.3-V LVTTL (default)		seg2[6..0]	24mA (default)	
15	seg2[3]	Output	PIN_B6	3	B3_N1	3.3-V LVTTL (default)		seg2[6..0]	24mA (default)	
16	seg2[2]	Output	PIN_B7	3	B3_N1	3.3-V LVTTL (default)		seg2[6..0]	24mA (default)	
17	seg2[1]	Output	PIN_B8	3	B3_N0	3.3-V LVTTL (default)		seg2[6..0]	24mA (default)	
18	seg2[0]	Output	PIN_B9	3	B3_N0	3.3-V LVTTL (default)		seg2[6..0]	24mA (default)	
19	sensor_in	Input	PIN_L21	5	B5_N1	3.3-V LVTTL (default)			24mA (default)	
20	sensor_out	Input	PIN_M22	6	B6_N0	3.3-V LVTTL (default)			24mA (default)	

شكل 16.III: نافذة Pin Planner

7.2.3.III. البرمجة (Programmation) :

بمجرد أن يتم تجميع (Compile) ومحاكاة (Simulation) التصميم بنجاح، يمكن تحميله على المكوّن القابل للبرمجة FPGA الموجود على بطاقة DE1 من شركة Altera المتوفرة في المختبر. وبهذا يمكن برمجة وتكوين الـ FPGA لاستقبال الدارة المراد تنفيذها.

يتم التكوين باستخدام وضعين معروفين:

❖ **وضع JTAG (Joint Test Action Group) :**

في هذا الوضع، حمَّلنا بيانات التكوين مباشرة إلى داخل شريحة الـ FPGA.

ثم ثبتنا تعريف التشغيل **USB Blaster Driver** .

ملاحظة مهمة : عند قطع التيار الكهربائي، يتم فقدان معلومات التكوين.

❖ **وضع AS (Active Serial) :**

في هذا الوضع، استخدامنا ذاكرة فلاش لتخزين بيانات التكوين.

يمكننا التبديل بين الوضعين (JTAG و AS على بطاقتي DE1 و DE2 باستخدام المفتاح RUN/PROG):

* الوضع RUN يفعّل وضع JTAG .

* الوضع PROG يفعّل وضع AS .

❖ **البرمجة في وضع JTAG :**

خطوات البرمجة والتكوين كالتالي:

1 .تحققنا من أن الاتصال بين الحاسوب وبطاقة FPGA (عبر وحدة **Byte-Blaster** أو **USB Blaster**) يعمل بشكل صحيح.

2 .قمنا بتشغيل البطاقة (تغذية بالطاقة) وتثبيت المفتاح **RUN/PROG** على الوضع **RUN.**

3 .من قائمة **Tools** في برنامج **Quartus** ، اخترنا **Programmer**.

4 .ظهرت لنا نافذة تحتوي على اسم المشروع بصيغة cdf ، مثل **NOM_DU_PROJET.cdf** :

5 .في هذه النافذة:

* حددنا **JTAG** في خانة **Mode**

* فعّلنا خيار **Program/Configure** بوضع علامة عليه.

6 .قمنا بحفظ ملف البرمجة كما تم مع الملفات السابقة، مثلاً **PORTE_vhdl1.cdf .**

7 .إذا لم يتعرف الحاسوب على البطاقة، تظهرلنا رسالة خطأ مثل:

"Unable to scan device chain. Hardware is not connected"

يمكننا استخدام خيار **Auto Detect** لمحاولة التعرف التلقائي.

8 .إذا لم يكن **USB-Blaster** محددًا بشكل افتراضي، نضغط على **Hardware Setup** ، ثم نختر **USB-Blaster** من القائمة.

9 .تأكدنا من أن الملف بصيغة **sof** الخاص بالمشروع موجود في خانة **File** .

10 .إن كان كل شيء صحيحًا، نضغط على **Start** لبدء البرمجة. ويمكن متابعة التقدم بالنسبة المئوية (%) .

11 .تُضيء **LED** في البطاقة لتؤكد أن التكوين تم بنجاح.

❖ **البرمجة في وضع AS :**

*اختبارنا الدارة بعد تحميلها في الـFPGA :

بعد إتمام البرمجة والتكوين، يمكننا اختبار والتحقق من عمل الدارة التي تم تحميلها في شريحة FPGA.

لن يتم استخدامها في هذا العمل التطبيقي.

4.III. خاتمة :

في هذا الفصل، تم التطرق إلى الجانب العملي من المذكرة والمتمثل في تصميم وتنفيذ نظام ذكي لإدارة مواقف السيارات باستخدام تقنية FPGA وبرمجة VHDL. تم شرح مراحل تطوير النظام بدءًا من دراسة المتطلبات وتحليل سلوك النظام، مرورًا بكتابة الكود بلغة VHDL، وصولًا إلى عملية المحاكاة واختبار الأداء باستخدام برنامج Quartus II من شركة Altera.

كما تم تطبيق التصميم فعليًا على لوحة تطوير من عائلة Cyclone II، مما أتاح التحقق من كفاءة النظام في التحكم في دخول وخروج السيارات بطريقة تلقائية، مع عرض عدد الأماكن المتاحة بشكل دقيق.

وقد أظهر هذا التطبيق العملي فعالية كبيرة من حيث السرعة والدقة في الأداء، إلى جانب سهولة إعادة برمجة النظام وتعديله، مما يعكس الإمكانات الكبيرة التي توفرها تقنيات FPGA في تصميم الأنظمة الرقمية الذكية. هذا الفصل يمثل تتويجًا للمفاهيم النظرية التي تم تناولها في الفصول السابقة، ويؤكد على أهمية الدمج بين الجانب النظري والتطبيقي في مشاريع التخرج الهندسية.

الخاتمة العامة

الخاتمة العامة

بعد تناولنا في هذه المذكرة لموضوع تصميم وتنفيذ نظام ذكي لإدارة مواقف السيارات باستخدام تقنية الـFPGA ولغة VHDL، يمكننا القول إن هذا العمل قد سمح لنا بالتعرف المتعمق على المفاهيم النظرية والتطبيقية المرتبطة بأنظمة التصميم الرقمي الحديثة، كما مكننا من الربط بين الجانب الأكاديمي والتقني العملي في إطار مشروع متكامل.

في الفصل الأول، استعرضنا أساسيات لغة VHDL باعتبارها أداة قوية لتوصيف وتصميم الدوائر الرقمية على مستوى عالي من التجريد. تعرفنا على بنيتها، أوامرها، طرق المحاكاة، وكيفية استخدامها في نمذجة وتصميم الأنظمة المعقدة. هذه اللغة تشكل حجر الأساس في البرمجة المنطقية القابلة للتهيئة، مما يجعل فهمها أمرًا ضروريًا لكل من يعمل في هذا المجال.

أما في الفصل الثاني، فتم التطرق إلى تقنية الـFPGA، حيث تم توضيح بنيتها الداخلية، آلية عملها، مميزاتها مقارنة بالتقنيات الأخرى مثل الـASIC، إضافة إلى أنواعها المختلفة وتطبيقاتها الواسعة في مختلف المجالات الصناعية والعلمية. وقد أظهر هذا الفصل أهمية الـFPGA كمنصة مرنة وقوية لتطبيق مشاريع إلكترونية متقدمة.

وجاء الفصل الثالث ليجسد الجانب العملي من المذكرة، حيث تم تطبيق المعارف المكتسبة في تصميم نظام عملي لإدارة مواقف السيارات. مررنا من خلال جميع مراحل الإنجاز، بدءًا من تحليل المتطلبات، مرورًا بكتابة كود VHDL، المحاكاة باستخدام أدوات مثل Quartus II. وقد مكننا هذا التطبيق من استكشاف التحديات التقنية الواقعية وكيفية التعامل معها، مما عمّق فهمنا العملي لكيفية تصميم وبناء الأنظمة الرقمية القابلة للتهيئة.

في النهاية، يمثل هذا العمل ثمرة جهد متكامل يجمع بين النظرية والتطبيق، ويؤكد أهمية امتلاك المهارات البرمجية والهندسية في مجال التصميم الرقمي الحديث. كما يشكل خطوة أولى نحو مشاريع أكثر تقدمًا في مجال الأنظمة المدمجة والمعالجات المنطقية، ويفتح آفاقًا واعدة في تطوير حلول ذكية وعملية لمشاكل الحياة اليومية. وبالنظر إلى النتائج المحققة، نطمح مستقبلًا إلى تطوير هذا النظام وتوسيعه ليشمل خصائص أكثر تقدمًا، وتطبيقه على لوحة FPGA متطورة من أجل تحسين الأداء وإضافة وظائف جديدة تلبي احتياجات المستخدمين بشكل أفضل.

المراجع

[1] https://ar.wikipedia.org/wiki/%D9%81%D9%8A_%D8%A5%D8%AA%D8%B4_%D8%AF

[2] https://fr.wikipedia.org/wiki/VHDL (site consulté le 27/01/2019).

[3] Philippe LARCHER .VHDL Introduction à la synthèse logique, 2000 .

[4] Anissa NASRI. Conception et implémentation d'un PIC 16F48 sur FPGA Thèse master université de Biskra 2018.

[5] Tangui RISSET. https//: comelec.enst.fr/hdl. Introduction à VHDL (site consulté le 27/01/2019).

[6] https://ae.ariat-tech.com/blog/vhdl.html?utm_source=perplexity

[7] Eduardo SANCHEZ EPFL/PDF. Le langage VHDL.

[8] Mr DHIABI Fathi SUPPORT DE COURS DE. université de Biskra.

[9] Jacques WEBER et Maurice MEAUDRE. VHDL du langage au circuit du circuit au Langage,1997.

[10] Laurent RODRIGUEZ et Benoit MIRAMOND Cours VHDL / PDF université de Cergy Pontoise. adresse url: https://docplayer.fr/5927659-Cours-vhdl-iv-l3-s6-universite-de-cergy-pontoise-laurent rodriguez-benoit-miramond .html (site consulté le 27/01/2019)

[11] https://fr.wikibooks.org/wiki/Conception_et_VHDL (site consulté le 27/01/2019)

[12] Laurence PIERRE. Modélisation des systèmes numériques - VHDL .Université Grenoble Alpes adresse url : http://users-tima.imag.fr/sls/lpierre/Teaching/MSN (consulté le 16/02/2019)

[13] C.ALEXANDRE. Introduction à la conception numérique en VHDL. pdf 2018 (site consulté le 25/02/2019)

[14] http://hdl.telecom-paristech.fr/vhdl_structurel.html . COURS EN LIGNE VHDL (site consulté le 16/02/2019)

[15] Jaques WEBER et Maurice MEAUDRE. circuits numériques et synthèse logique un outi.

[16] Jocelyn Sérot, Les circuits FPGA et le langage VHDL, une intro pour les programmeurs et par l'exemple, Editions Ellipses, 2019.

[17] https://nandland.com/introduction-to-vhdl-for-beginners-with-code-examples/ VHDL Tutorial - Introduction to VHDL for beginners.

[18] Carlis Collins, « THE QUARTERLY JOURNAL FOR PROGRAMMABLE LOGIC USERS »,Three articles discuss the history of Xilinx and the programmable logic industry,Issue 32 ,Second Quarter,1999.

[19] https://www.ibm.com/think/topics/field-programmable-gate-arrays

[20] https://biztechmagazine.com/article/2023/01/what-field-programmable-gate-array-and-how-does-it-work-perfcon

[21] François Verdier, « Les circuits FPGA Concepts de base, architecture et applications», Université de Cergy-Pontoise Laboratoire ETIS UMR CNRS 8051

[22] Conception d'un outil de prototypage rapide sur le FPGA pour des applications de traitement d'images Thèse de doctorat. Debyo SAPTONO Universite De Bourgogne NOV. 2011.

[23] https://fastercapital.com/kalimat/%D8%A7%D9%84%D9%85%D8%B3%D8%AA%D9%86%D8%AF%D8%A9-FPGA.html

[24] https://www.embeddedrelated.com/showarticle/195.php

[25] https://www.memoireonline.com/10/12/6158/m Mise-en-oeuvre-de-lauto-reconfiguration-[partielle-et-dynamique-sur-FPGA-Xilinx-Virtex-II-pro11.html (consulté le 25/03/2019)

[26] http://proxacutor.free.fr/index.htm (site consulté le 25/03/2019)

[27] https://www.ibm.com/sa-ar/think/topics/fpga-vs-microcontroller

[28] https://rushpcb.com/fpga-vs-microcontroller-understanding-the-key-differences/

[29] https://www.cder.dz/vlib/bulletin/pdf/bulletin_024_05.pdf Les circuits FPGA: description et applications GUELLAL Amar Attaché de recherche (site consulté le 25/03/2019).

[30] Ricardo Tapiador, Comprehensive evaluation of opencl-based convolutional neural network accelerators in xilinx and altera fpgas arXiv preprint arXiv:1609.09296, 2016.

[31] Task Force, High performance microchip supply, Annual Report. Defense Technical Information Center (DTIC), USA, 2005.

[32] Leveraging FPGA and CPLD Digital Logic to Implement Analog-to-Digital Converters, Lattice Semiconductor, A Lattice Semiconductor White Paper February, 2010.

[33] Chunxiao Lin, Yang Yi, Enhancing FPGA CAD Flow with AI-Powered Solutions, AI-Enabled Electronic Circuit and System Design : From Ideation to Utilization, 225-256.

[34] Xifan Tang, Edouard Giacomin, OpenFPGA: An opensource framework enabling rapid prototyping of customizable FPGAs, 2019 , 29th International Conference on Field Programmable Logic and Applications (FPL), 367-374, 2019.

[35] https://www.linkedin.com/pulse/leading-way-top-10-companies-global-fpga-market-curtis-croy