République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

École Nationale Polytechnique

Département d'Électronique

Mémoire de Fin d'Études

Présenté par

TIRCHI Saïd
TOUADI Mohammed Saïd

Pour l'obtention du diplôme

Ingénieur d'État en Électronique

Intitulé

Implémentation d'un routeur sur SoC (System on Chip)

Devant le jury composé de :

M^{r}	S.BOUSBIA	Président
M^{elle}	A.MOUSSAOUI	Examinatrice
M^{r}	R.SADOUN	Rapporteur

Promotion Juin 2008

مـــلـــخـــص

بالنظر الى معدلات نمو كبيرة للأنتقال من تدفق البيانات ذات الصلة الى النمو الهائل للانترنت ، فإن فكرة ابتكار كفاءه قادرة على تتبع مسارات تدفق صلات جيجابت وقد مثيرة للاعجاب. هذه المسارات يجب ان تلبي المعايير المختلفة (ديون الملكيه الفكريه فى وقت هام فى ايجاد الحد الادنى من التكامل رقاقة الاستهلاك وانخفاض التكاليف) التي غالبا ما تكون صعبة لتنفيذ في وقت واحد. مع ظهور النظم التي تتسم على رقاقة وادوات تطوير اساليب تقديم حلول مبتكرة لضمان خفض تصميم الوقت ، أكثر وأكثر وظيفيه ، ووضع مسار وعلى هذا الاساس جدا قوية. في هذا البيان الموجز ، وهو يمثل حزب دولة كمبوديا على مسار التنفيذ في البيئة من اجل التنمية جزءا لا يتجزأ من نظم edk مع ادماج البروتوكول الملكيه الفكريه باستخدام المكتبه مفتوحة المصدر وتنفيذ lwip خوارزميه بحث الاسراع في البحث عن وظيفة الملكيه الفكريه.

الكلمات المفتاحية : ، المسارات ، Soc ، FPGA .

Résumé

Face à la croissance considérable des débits de transmission des flux de données lié au développement exponentiel d'Internet, l'idée de concevoir des routeurs performants aptes à suivre les débits des liaisons Gigabit s'est vue imposante. Ces routeurs doivent satisfaire à différents critères (débits important, temps de recherche IP minimale, intégration sur puce, consommation et couts réduits). Avec l'avènement des Systems on Chip caractérisés par des méthodes et des outils de développement fournissant des solutions innovantes en garantissant un temps de conception réduit, pour de plus en plus de fonctionnalités, la conception d'un routeur sur cette base s'avère très performante. Dans ce mémoire, on présente une implémentation sur SoC d'un routeur sous l'environnement de développement de systèmes embarqués ***EDK*** avec l'intégration de la suite protocolaire TCP/IP à l'aide de la librairie open source lwIP et l'implantation d'un algorithme de recherche accélérant la fonction IP lookup

Mots clefs : Routeur , SoC , FPGA.

Abstract

Given the considerable growth rates of transmission of data flows related to the exponential growth of the Internet, the idea of devising efficient routers able to follow the flow of Gigabit connections has been impressive. These routers must meet various criteria (debits important time finding IP minimal integration chip consumption and lower costs) that are often difficult to implement simultaneously. With the advent of Systems on Chip characterized by methods and development tools providing innovative solutions ensuring a reduced design time, more and more functionality, design a router on this basis is very powerful. In this brief, it presents a SoC implementation on a router in the development environment for embedded systems EDK with the integration of the following protocol TCP / IP using the library open source lwIP and implementation of A search algorithm accelerating IP lookup function.

Key words: Router , SoC ,FPGA.

TIRCHI Said

Je dédie ce modeste travail à :

– ma mére, mon pére mes fréres surtout Sofiane et ma soeur, qui ont toujours cru en moi
– ma grande mére
– mes tantes et oncles
– mes cousins et cousines
– mes amis ainsi qu'à tous les membres de ma promotions

TOUADI MOHAMED Said

Je dédie ce modeste travail à :

ma mère et mon père qui m'ont toujours soutenu tout au long de ce parcours

mes soeurs

ma tante et mes oncles

mes cousins et cousines

mes amis

Remerciements

Nous tenons particulièrement à exprimer notre profonde gratitude à Monsieur SADOUN Rabah,enseignant chercheur à l'Ecole Nationale polytechnique, pour son encadrement et son suivi. Son soutien, ses conseils, ses encouragements permanents tout au long de ces années de recherche, son dynamisme et sa disponibilité ont joué un rôle déterminant dans ce travail. Qu'il trouve ici, l'expression de notre profonde reconnaissance.

Nous tenons à remercier tout particulièrement, Monsieur BOUSBIA, Professeur à l'Ecole Nationale Polytechnique, d'avoir accepté de présider le jury de notre mémoire.

Nous adressons également nos plus vifs remerciements à Madame MOUSSAOUI, professeur à l'Ecole Nationale Polytechnique, pour l'honneur qu'elle nous a fait en acceptant de juger ce travail.

Nos remerciements vont aussi à tout le corps enseignant de l'Ecole Nationale Polytechnique spécialement les enseignants du département d'Electronique pour leur apport de connaissance durant nos cinq années d'études.

Enfin, tous nos remerciements à toutes les personnes qui ont contribué de prés ou de loin à l'aboutissement de ce travail. Nous leurs somme très reconnaissants.

Table des matières

Table des figures

Introduction

Actuellement, on assiste à une évolution fulgurante dans le domaine des technologies de la communication. Les progrès réalisés dans ce domaine ont largement contribué à l'évolution et à l'extension du réseau Internet, le projet d'interconnexion de plusieurs réseaux hétérogènes initié en 1973 est devenu aujourd'hui le réseau des communications universelles.

La croissance importante du nombre d'internautes et la disponibilité des modules de connexion à ce réseau pendant ces deux dernières décennies ont considérablement augmenté la taille des réseaux constituant l'Internet, ainsi que le volume du trafic. Ceci s'est traduit par un besoin d'infrastructure capable de supporter un trafic de taille importante, à savoir des liens, des routeurs et des périphériques réseau pouvant supporter des débits de transmission de l'ordre de Gigabit.

Notre travail s'inscrit dans un projet de conception et de réalisation d'un routeur Gigabit. Le besoin de ce type d'équipement est dicté par une nécessité de plus en plus pressante en terme de débit avec le nombre de plus en plus élevé d'utilisateurs et une évolution vers des bandes passantes plus large au niveau des supports de transmission.

Les noeuds (routeurs) réseaux constituant la toile Internet deviennent des goulots d'étranglement, d'ou la nécessité de routeurs plus performant.

Notre contribution dans le projet consistera à mettre en oeuvre les briques de base, à savoir le choix d'un algorithme de recherche permettant d'optimiser le temps de la recherche de route et d'un modèle d'implémentation proposant une architecture d'un routeur supportant des débits plus important selon les technologies offertes. Actuellement les progrès réalisés dans le domaine de la microélectronique permettent d'intégrer un

système complet sur puce, bien connu sous le nom de System on Chip (SoC).

Ce travail est alors organisé en cinq chapitres de la manière suivante :

- Le premier chapitre présente une introduction à notre thème de recherche, en fournissant une description de l'architecture du réseau Internet avec un bref aperçu sur la pile protocolaire TCP/IP avec quelques détails sur la version du protocole Internet (IPv4) actuellement utilisé par les réseau afin de pouvoir mieux comprendre la nécessité de la conception de routeurs plus performants ;
- Le second chapitre met en valeur les concepts de base du routage dans le réseau Internet, en illustrant ses divers types avec une description de l'architecture interne d'un routeur ;
- Le troisième chapitre, relatif aux algorithmes de recherche avec une comparaison justifiant notre choix ;
- Le quatrième chapitre décrit en détail l'algorithme de recherche choisi, Dynamic Tree Bitmap, suivi d'une évaluation des performances ;
- Enfin, le dernier chapitre qui traite la partie développement. Dans cette partie on présente le système d'implémentation sur puce (SoC) du routeur sous l'environnement EDK puis on cite les différentes adaptations effectuées sur la plateforme matérielle ainsi que les librairies nécessaire au fonctionnement de notre routeur.

Chapitre 1

Internet

Dans ce chapitre, nous allons aborder la notion de réseau Internet en fournissant une architecture simplfiée, puis présenter les éléments relatifs à la couche réseau (IP) et un bréf aperçu sur les deux protocoles de la couche transport à savoir TCP et UDP.

1.1 Introduction

Les agences gouvernementales américaines ont, depuis des années, compris l'importance et le potentiel du projet internet [1] , à tel point qu'elles ont subventionné des travaux de recherche qui ont conduit à la réalisation d'un Internet mondial.

La technique Internet de l'ARPA *(Advanced Research Projects Agency)* [2] définit un ensemble de standards qui spécifient les détails des communications entre ordinateurs ainsi qu'un ensemble de conventions pour l'interconnexion des réseaux et le routage des informations. Officiellement baptisée suite à des protocoles d'interconnexion TCP/IP et couramment nommée TCP/IP, environnement TCP/IP, protocoles TCP/IP, il s'agit d'un ensemble de protocoles permettant d'établir des communications entre divers types de réseaux interconnectés.

TCP/IP constitue la technique de base du système Internet mondiale qui relie aujour-

[1] Nous suivrons la convention de désigner par Internet (avec un i : majuscule) le réseau mondiale, tandis qu'un internet (ou intranet) désigne un réseau privé utilisant la technologie TCP/IP *(Transmission Control Protocol et Internet Protocol)*

[2] Au cours des années 1980, l'ARPA a été très souvent dénommée DARPA *(Défence Advanced Research Projects Agency).*

d'hui plusieurs centaines de millions de personnes dans des organismes de recherche, des universités, des écoles, des entreprises de nombreux laboratoires gouvernementaux etc. Le succès éclatant d'Internet prouve la maturité de la technique TCP/IP et montre qu'elle s'adapte à une grande variété de réseaux.

1.2 Architecture du réseau Internet

Internet est un ensemble de réseaux interconnectés, qui consistent en une combinaison de plusieurs éléments hétérogène (hôtes, liens, routeurs), utilisant le protocole TCP/IP pour échanger des informations à travers le monde. La figure (FIG.1.1) illustre un exemple d'une architecture Internet, les réseaux locaux sont connectés au réseau Internet via des passerelles ou des routeurs de frontières *(Edge routers)*. Les informations sont acheminées dans le coeur du réseau Internet via des routeurs centraux *(core routers)*. Les modules de communication sont des hôtes (hosts) qui peuvent être des ordinateurs personnelles, stations de travail (workstation), des serveurs PDA (Personal Digital Assistants) ou des satellites transmettant et recevant des paquets contenant des blocs de données.

Les liens servent à connecter les éléments du réseau (les hôtes et les routeurs), ce sont généralement des fils de cuivre torsadés, câbles à fibre optique ou une variété de technologies de liaisons sans fil comme la radio infrarouge et les micro ondes. Il existe diverses stratégies de mise en place de liens dans un réseau. Ces stratégies prennent en considération la bande passante, la localisation géographique, le déploiement et les frais d'exploitation.

Le rôle fondamental des routeurs est la commutation de paquets entrant à partir des liens d'entrées vers les liens de sorties appropriés menant à la destination. Un paquet traverse plusieurs liens et noeuds aussi appelé sauts (hop), afin d'atteindre sa destination. Les liens du réseau son de nature transitoire (congestion, échec, addition, suppression), pour cela les protocoles permettent aux routeurs d'échanger continuellement des informations sur l'état du réseau [1].

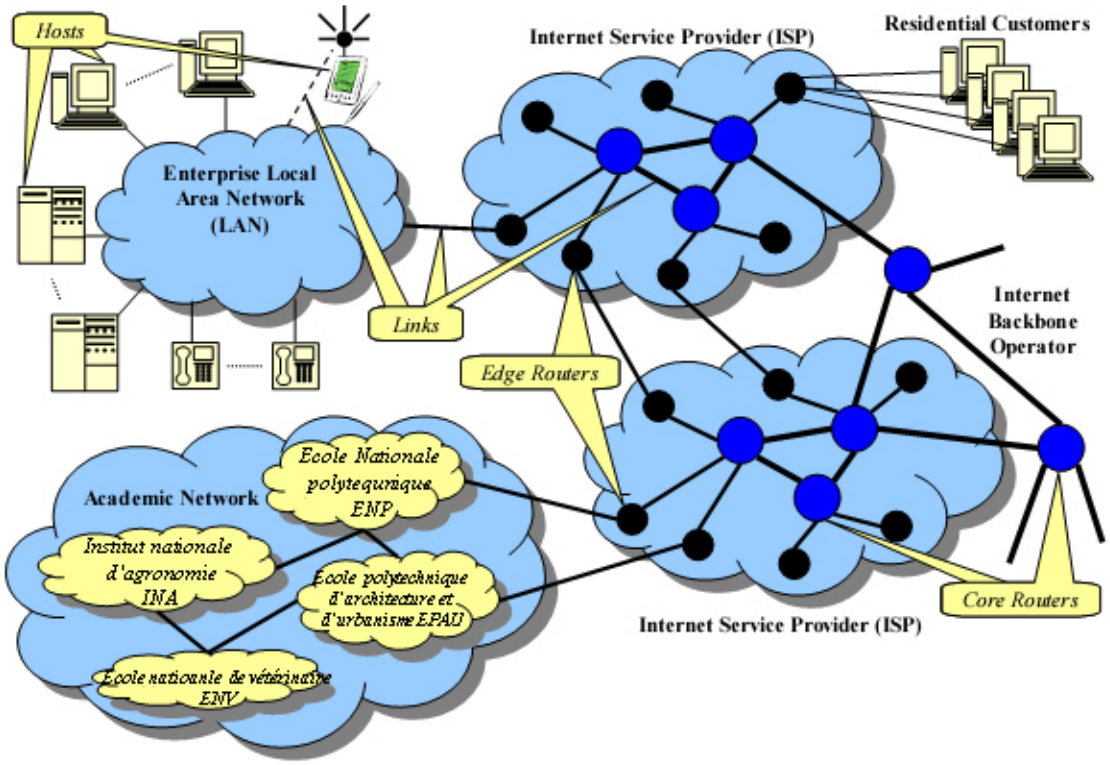


FIG. 1.1 – Un diagramme simplifie d'une architecture Internet[1]

1.3 IP (*Internet Protocol*)

Comme nous l'avons vu précédemment, le réseau Internet se présente comme le fruit de l'interconnexion de très nombreux réseaux locaux par des réseaux moyen et longue distances. La diversité des éléments constituant un tel système a nécessité la mise au point d'un protocole de communication universel. C'est à la fin des années 70 qu'a été présenté le protocole de communication IP permettant l'interconnexion de systèmes hétérogènes, indépendamment des supports de transmission, de la nature du réseau, ou des systèmes d'exploitation et des applications utilisées. Son intégration dans les grands systèmes UNIX, au début des années 80 puis progressivement dans tout les systèmes d'exploitation [2]. Le protocole IP fournit un service de remise des datagrammes en mode non connecté . Trois définitions importantes sont liées à ce protocole. La première définit le datagramme comme l'unité de donnée de base circulant dans un internet TCP/IP. La second correspond à la fonction d'adressage et la troisième est la fonction de routage.

La version actuelle, celle utilisée dans le réseau Internet est *(IPv4)*, une nouvelle *(IPv6)*

prendra bientôt sa place.

1.3.1 IPv4 (*Internet Protocol version 4*)

Les machines travaillant sous le protocole IPv4 possèdent une adresse tenant sur 32 bits. Cette adresse est représentée par une suite de quatre nombres séparés par des points ; par exemple 191.92.34.223.

L'adresse est constituée de deux parties : un identificateur de réseau et un identificateur de la machine à l'intérieur de ce réseau. L'identificateur de réseau est précédé par un numéro de classe de réseau. Il existe quatre classes d'adresses, chacune permettant de coder un nombre différent de réseaux et de machines, la figure (FIG.1.2) indique ces quatre classes :

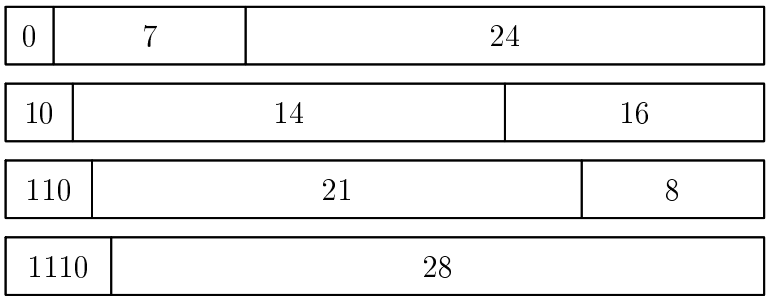


FIG. 1.2 – Les zones d'adresses

- classe A - 128 réseaux (codés sur 7 bits) ;
- classe B - 16 384 réseaux (codés sur 14 bits) ;
- classe C - 2 097 152 réseaux (codés sur 21 bits) ;
- classe D - adresses de groupe (codés sur 28 bits).

Une des difficultés majeures que doit affronter l'Internet est l'épuisement des adresses et en particulier des adresses de classe A et B . Une solution, définie en 1992 est appelée CIDR *(Classless Internet Domain Routing)* ou routage inter domaine hors classe [3]. De façon conceptuelle, CIDR fusionne ou agrège un ensemble contigu d'adresse de classe C (agrégation d'adresse) en une seule entrée représentée par le doublet : (Adresse réseau, Compteur) dans lequel l'adresse réseau est la plus petite adresse réseau du bloc et un compteur qui spécifie le nombre total d'adresses réseau du bloc [4]. C'est ainsi

que (192.5.48.0 ,3) permet de spécifier les trois adresses réseaux 192.5.48.0 ,192.5.49.0 et 192.5.50.0. À l'heure actuelle, le plan d'adressage dit agrégé est une généralisation des règles définies en IPv4 par CIDR. L'adressage contient 3 niveaux d'hiérarchie :

1. TLA (Top Level Aggregator) qui correspond à des opérateurs ;
2. NLA (Next Level Aggregator) correspondant aux fournisseurs d'accès ;
3. SLA (Site Level Aggregator) géré par le site.

Les faiblesses d'IPv4 concernent d'abord l'adressage qui est limité par les quatre octets disponibles. En fait, la distribution des adresses n'a pas été faite avec suffisamment de soin et de nombreuses adresses A et surtout B sont excessivement mal utilisées. Le second problème concerne l'arrivée d'applications multimédias qui contiennent des synchronismes forts comme celui de la parole.

Une autre lacune importante concerne la sécurité de la communication dans le réseau, la version IPv6 résout ces problèmes.

1.3.2 Format du Datagramme IPv4

Le datagramme est l'unité de donnée fondamentale circulant dans une session de communication. Le paquet (datagramme) IP est constitué des champs suivants comme indiquer dans la figure (Fig.1.3) :

- Version : vaut 4 dans les versions IPv4 et 6 dans la prochaine version du protocole. C'est pour cette raison que ces protocoles sont généralement désignés par IPv4 et IPv6 ;
- Longueur de l'en-tête : donne la longueur de l'en-tête en mots de 32 bits ;
- Type de service : la sémantique de ce champ a récemment été modifiée. Initialement, ce champ devait servir à prendre des décisions de routage différentes.
- Fragmentation : ce champ de 32 bits, permet d'adapter la taille des paquets IP à la taille des trames de niveau 2 ;
- Durée de vie : ce champ permet d'éviter les boucles dans le réseau. Il porte ce nom car il devait initialement limiter la durée de séjour d'un paquet dans le réseau ;
- Protocole : donne le protocole de niveau supérieur (ICMP) ;

- Checksum : permet de valider le contenu de l'en-tête IP. À noter que comme le champ durée de vie est modifié à chaque traversée de routeur, le champ checksum doit être modifié par ce dernier ;
- Les champs suivants : contiennent les adresses source et destination. Le paquet IP peut contenir des informations optionnelles. À noter le champ bourrage en fin d'option qui permet, le cas échéant, de maintenir l'alignement du reste des informations sur des mots de 32 bits. Le champ longueur de l'en-tête permet de connaître leur taille. Pour les raisons de performances décrites précédemment, il est très rare de trouver des options dans un en-tête IP. Parmi les options prévues par le protocole, certaines concernent le routage : - l'option router alert permet d'indiquer aux routeurs qu'ils doivent examiner le contenu du paquet. Cette option est utilisée par le protocole de gestion des groupes multicast IGMP *(Internet group message protocol)* et le protocole de signalisation et de réservation de ressource RSVP*(réservation protocol)* ; - l'option routage par la source (source routing). Cette option permet de préciser un chemin dans le réseau, c'est-à-dire l'adresse IP des routeurs par lesquels le paquet devra passer.

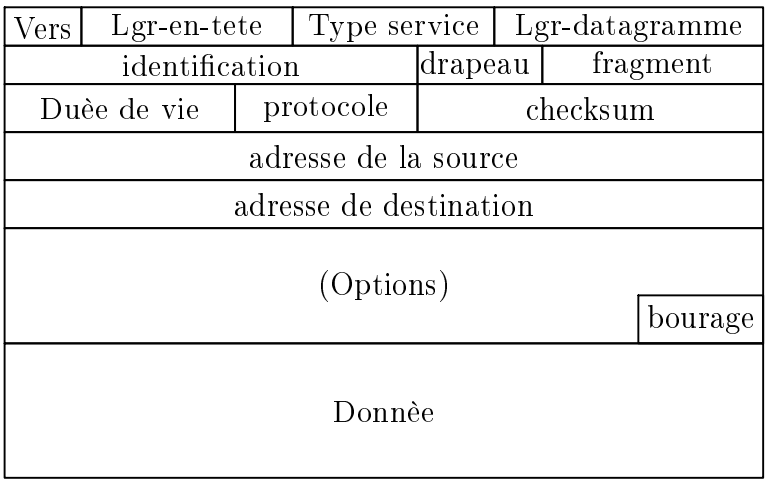


FIG. 1.3 – Format du paquet IPv4

1.4 TCP (*Transport Control Protocol)*

Le protocole TCP définit la structure des données et des acquittements échangés, et les mécanismes permettant de rendre le transport fiable. Il spécifie comment distinguer plusieurs connexions sur une même machine, et comment faire la détection et la correction, lors de la perte ou duplication de paquets. Il définit comment établir une connexion et comment la terminer. L'unité de protocole de TCP est appelée un segment. Ces segments sont échangés pour établir la connexion, pour transférer des données pour les acquittements, pour modifier la taille de la fenêtre et enfin pour fermer une connexion. Le protocole IP fournit un sérvice en mode non-connecté (sans accusé de réception ou reprise sur échec), mais s'il est combiné à TCP alors il sera en mode connecté.

1.5 UDP(*User Datagram Protocol)*

Le protocole UDP*(service de transport non fiable)* permet aux applications d'échanger des paquets en mode non-connecté. Ce protocole utilise la notion de " port " qui permet de distinguer les différentes applications qui s'exécutent sur une machine. En plus du datagramme et de ses données, un message UDP contient, à la fois, un numéro de port source et un numéro de port destination. UDP s'appuie sur les services du protocole Internet et fournit un service en mode non connecté, sans reprise sur erreur. Il n'utilise aucun acquittement, ne régénère pas les messages et ne met en place aucun contrôle de flux. Les messages UDP peuvent être perdus, dupliqués, remis hors séquence ou arriver trop tard pour être traités en réception.

1.6 Conclusion

Dans ce chapitre on a pu exposer des notions de réseau Internet ainsi que la suite protocolaire TCP/IP qui tire son intérêt de sa viabilité prouvée à grande échelle. Elle constitue la technique de base du système Internet mondiale qui relie aujourd'hui plusieurs millions d'utilisateurs. Par la suite nous nous interesseront encore d'avantage à la couche

IP notament le routage des paquets dans le réseau Internet.

Chapitre 2

Le Routage IP

2.1 Introduction

Définitions et principes

Avant de commencer à décrire techniquement les méthodes utilisées dans l'Internet, il est intéressant de clarifier le vocabulaire, car il règne une certaine ambiguïté dans les termes employés :

- Le terme **routage** est souvent confondu avec le terme de reliage (forwarding) qui fait référence à la fonction principale d'un routeur. Cette fonction consiste à recopier le plus vite possible un paquet sur l'interface de sortie ;
- La fonction de **routage** construit les bases de données (appelées tables de routage) indiquant l'interface de sortie pour une destination. Cette information servira pour le reliage. A première vue, ces définitions peuvent s'appliquer aux équipements de niveau 2 du modèle de référence de l'ISO[1](Ponts) et de niveau 3 (routeurs). Le routage consistera essentiellement à attribuer logiquement les adresses aux équipements.

[1]ISO est connu sous le nom de modèle de référence pour l'interconnexion des systèmes ouverts, souvent appelé modèle OSI*(Open Systems Interconnexion)*.

2.2 Principes Fondamentaux du Routage

Les ordinateurs sont constitués de quatre composants de base : un processeur, la mémoire, des interfaces et des port d'entrées/sorties. Comme un routeur est aussi doté de ces éléments, on peut en conclure qu'il s'agit d'un ordinateur. Mais c'est un ordinateur réservé à un usage particulier. Le routeur étant entièrement dédié au routage, aucun de ses composants n'est consacré aux dispositifs de sortie vidéo et audio, aux dispositifs d'entrée, tels que le clavier et la souris, ainsi qu'aux autres logiciels conviviaux typiques d'un ordinateur multimédia moderne.

À l'image des ordinateurs qui ont besoin de systèmes d'exploitation pour exécuter les applications, les routeurs doivent être équipés d'une plate forme logicielle IOS *((Internetworking Operating Software)* pour exécuter les fichiers de configuration,ces fichiers contrôlent le flux de données en direction du routeur. Plus précisément, en utilisant des protocoles de routage et des tables de routage pour rediriger les protocoles routés, ils choisissent les meilleures voies pour les paquets. Pour contrôler ces protocoles et décisions, il convient de configurer le routeur.

Une bonne partie de ce module est consacrée à la création de fichiers de configuration à partir de commandes IOS, de telle sorte que le routeur exécute les fonctions réseau de votre choix.

Le routeur est un dispositif qui choisit les meilleures voies de transition de trafic et qui gère la commutation des paquets entre deux réseaux différents.

2.3 Routage IP

Concept En théorie, le routage IP est simple, si une machine de destination est directement connectée à une autre machine ou sur un réseau partagé (par exemple : Ethernet), alors le datagramme IP est envoyé sans intermédiaire à cette destination. Par contre, le routage est plus complexe sur un routeur ou sur une machine avec plusieurs interfaces. Le routage IP est effectué sur la base de "saut à saut" (hop to hop routing). Les étapes du routage IP peuvent être découpées de cette manière :

- Recherche, dans une table de routage, de l'entrée associée à l'adresse IP de destination ; S'il trouve une correspondance entre la table de routage et l'adresse de destination, le datagramme IP est envoyé au routeur de "saut suivant"(next-hop router). Ce cas de figure est utilisé pour les liaisons point à point.
- Recherche, dans la table de routage, de l'entrée correspondant exactement à l'identificateur du réseau de destination. Si cette adresse est localisée, envoi du paquet au routeur de saut suivant indiqué ou à l'interface directement connecté (par exemple : si l'interface existe sur le routeur). C'est ici aussi que l'on tient compte des masques de sous réseau ;
- Recherche, dans la table de routage, de l'entrée par défaut. Envoi du paquet au routeur "de saut suivant" si cette entrée est configurée. Si le déroulement de ces 3 phases est correct, alors le datagramme IP est délivré au prochain routeur ou host. Par contre, si cela n'est pas le cas, un message ICMP (host unreachable ou network unreachable) est envoyé au host d'origine et le datagramme IP est jeté ;

2.4 Routage des datagrammes dans un Réseau Internet

Les ordinateurs et les routeurs participent au routage des datagrammes IP, lorsqu'un programme d'application d'un ordinateur entreprend de communiquer, cela conduit les protocoles TCP/IP à générer un ou plusieurs datagrammes IP. L'ordinateur prend une décision de routage lorsqu'il doit choisir ou envoyer les datagrammes qu'il produit. C'est le cas sur la (Fig.2.1),où un ordinateur doit prendre une décision de routage bien qu'il ne soit relié qu'à un seul réseau. Les routeurs prennent eux aussi des décisions de routage (c'est leur rôle principal et cela justifie leur appellation de routeur) [4].

Divers types de routages ont était mis au point avec l'évolution du réseau Internet, dans ce qui suit on présente un aperçu

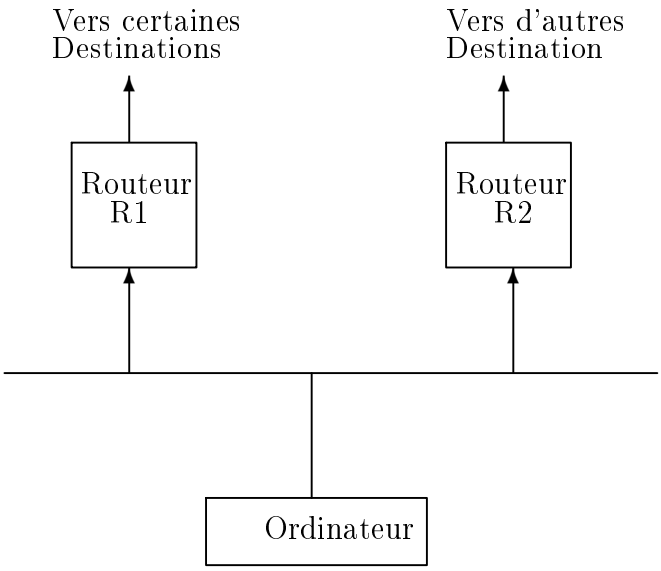


FIG. 2.1 – Exemple d'un ordinateur qui doit prendre une décision de routage[6]

2.4.1 Remise directe et indirecte

On peut distinguer deux formes de routage : Celui par remise direct et celui par remise indirecte des datagrammes.

La remise directe correspond au transfert direct d'un datagramme entre deux ordinateurs, c'est le fondement sur lequel s'appuient les communications dans un internet (réseau local) ;

La remise indirecte Dans ce cas les données échangées entre les ordinateurs doivent transiter par au moins un routeur, ce qui nécessite un calcul de routage.

2.4.2 Routage géré par des tables de routage

Les algorithmes de routage IP utilisent une table de routage qui contient les informations relatives aux différentes destinations possibles d'Internet et à la façon de les atteindre. A chaque fois que le logiciel IP d'un routeur doit transmettre un datagramme ,il consulte sa table de routage pour déterminer ou envoyer le datagramme [4].

Quelles informations doit contenir une table de routage ? Si chaque table de routage contenait les informations relatives a toutes les destinations possibles d'Internet, les routeurs devrait disposer de capacités de mémoire considérables pour stocker des tables de tailles gigantesques .Pour que le routage puisse être effectué malgré cet handicap, un mé-

canisme de masquage d'informations est mis en oeuvre pour permettre aux routeurs de prendre décisions de routage en n'ayant qu'un minimum d'informations.

Note : les tables de routage ne doivent contenir que des adresses réseaux et non la totalité des adresses IP.

2.4.3 Routage par sauts successifs

L'utilisation du préfixe réseau id-res, des adresses IP au lieu des adresses complètes assure l'efficacité du routage et réduit la taille des tables de routage. Ceci permet en plus de masquer des informations locales relatives a des ordinateurs spécifiques situes sur un même sites .En général, les tables de routage contiennent des paires d'adresses (N,R) ou N est l'adresse IP du réseau de destination et R l'adresse du routeur suivant sur le chemin qui mène au réseau N. Le routeur R est appelé saut suivant (*next hop*). La solution qui consiste à utiliser une table de routage par saut successifs (next hop routing), ainsi chaque table de routage n'indique qu'une seule étape du cheminmenant du routeur R au réseau N : donc un routeur ignore le chemin complet qui mène au destinataire final. Tous les routeurs référencés dans les tables de routage d'un ordinateur M doivent appartenir à des réseau auxquels cet ordinateur est directement reliés. Lorsqu'un M s'apréte à transmettre un datagramme le logiciel IP localise l'adresse IP du destinataire et en extrait la partie réseau, id-res. M utilise ensuite l'identificateur id-res pour prendre une décision de routage, en choisissant un routeur directement accessible.

2.4.4 Routage par défaut

Une autre technique utilisée pour masquer les informations et conserver des tables de routage de petites taille consiste à associer plusieurs entrées à un cas par défaut .L'idée consiste à faire en sorte que le logiciel IP recherche d'abord l'identificateur du réseau dans la table de routage .Si aucune route n'apparaît dans la table, les procédures de routage envoient le datagramme à un routeur par défaut [4].

2.4.5 Routage par les adresses IP

Il est important de comprendre que, hormis le calcul de la durée de vie et du total de contrôle, le routage IP n'altère en rien un datagramme en transit. En particulier, ses adresses source et destination demeurent inchangées ; elles indiquent toujours l'adresse IP de l'émetteur initiale et l'adresse IP du destinataire final [2]. L'exécution de l'algorithme de routage par le logiciel IP permet de calculer une nouvelle adresse IP, celle de la machine suivante vers laquelle le datagramme sera transmis. L'adresse IP calculée par l'algorithme est appelée adresse de saut suivant (next hop adresse).

Ou le protocole IP conserve t'il les adresses de sauts suivants ? Pas dans le datagramme ; aucune place n'y est réservée .L'adresse de saut suivant n'est pas conservée du tout par le protocole IP. Celui-ci, après avoir exécuté l'algorithme de routage, transmet le datagramme et l'adresse de saut suivant a l'interface réseau par laquelle le datagramme doit être émis. Cette interface fait correspondre l'adresse de saut suivant à une adresse physique, élaboré une trame ou un paquet, y encapsule le datagramme puis transmet le tout sur le réseau. L'adresse de saut suivant est détruit par l'interface réseau après avoir être utilisée pour la résolution de l'adresse physique correspondante.

2.5 Gestion des datagrammes entrants

Nous avons décrit jusqu'a présent le protocole IP en indiquant comment les décisions relatives aux datagrammes sortants étaient prises. Il est clair cependant que le logiciel IP doit également gérer les datagramme entrant.

Lorsqu'un datagramme arrive dans un ordinateur, le logiciel de l'interface réseau le transmet au logiciel IP chargé de le traiter. Si l'adresse de destination du datagramme coïncide avec l'adresse IP de l'ordinateur, le logiciel IP accepte le datagramme et le transmet au protocole de haut niveau approprie. Dans le cas contraire, le datagramme est ignoré et détruit (en effet, les ordinateurs n'ont pas le droit de tenter de corriger les erreurs de routages). Déterminer si un datagramme IP a atteint sa destination finale n'est pas aussi

[2] il y a une seule exception, lorsque le datagramme contient l'option de routage par la source.

évident qu'il y parait. Un ordinateur peut avoir plusieurs points d'accès à plusieurs réseaux, chacun repéré par une adresse physique. Lorsqu'un datagramme arrive, l'ordinateur doit comparer l'adresse IP du datagramme avec celle de chacun de ses accès réseau. S'il y a correspondance avec l'une d'entre elles, l'ordinateur conserve le datagramme et le traite.

2.6 Architecture génerale d'un routeur

L'architecture d'un routeur s'articule autour d'un crossbar, d'une zone mémoire, de ports d'entrée contenant principalement un calculateur de route et de ports de sortie (Fig.2.2)

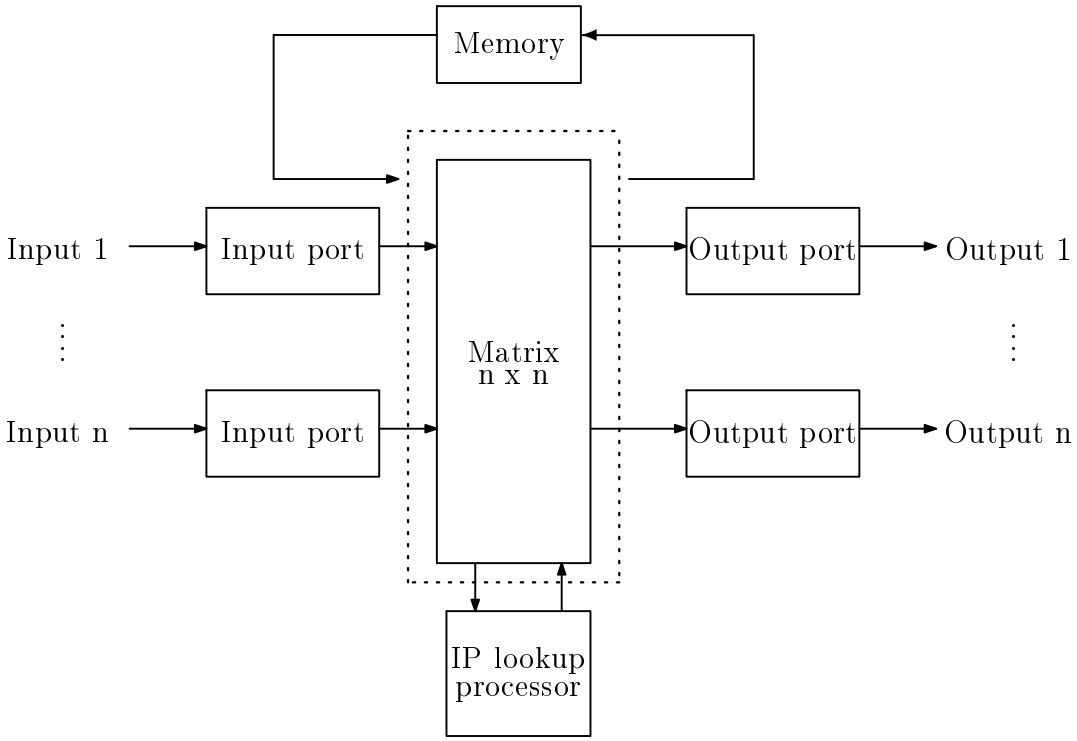


Fig. 2.2 – Architecture générale d'un routeur

.

– Le crossbar est constitué de deux parties principales, à savoir une matrice de commutation permettant de réaliser physiquement la connexion d'un port d'entrée vers un port de sortie, et un allocateur de sortie. Ce dernier a plusieurs fonctions :

1. Il réalise l'arbitrage entre les requêtes simultanées faites par plusieurs ports d'entrée désirant accéder au même port de sortie ;

2. Il effectue et mémorise l'allocation d'un port de sortie à un port d'entrée ;
3. Il commande la matrice de commutation de façon à réaliser les connexions demandées.

- Le calculateur de route détermine vers quelle sortie orienter le paquet, selon l'information de destination qu'il transporte ;
- La zone mémoire a pour rôle de stocker les données reçues lorsque des contentions se produisent. Citons comme exemple de contention le cas ou :
  1. Deux paquets issus de deux ports d'entrée différents sont en compétition pour accéder au même port de sortie qui ne peut en accepter qu'un ;
  2. Le port de sortie que doit emprunter un paquet n'a pas l'autorisation d'émettre, à cause d'une contention en aval.

2.7 Conclusion

Dans ce chapitre on a pu decourvrir les notions fondamentales de routage nécessaire à la transition du flux de données via le réseau Internet. Par la suite on a vu les types de routage existant afin d'établir la communication entre deux réseaux ainsi que le schéma synoptique d'un routeur. Le type de routage le plus répandu et pour lequel nos travaux se focaliseront par la suite est le routage par sauts successives géré par les tables de routage.

Chapitre 3

Algorithmes de Recherche

3.1 Introduction

La rapidité est le critère d'évaluation fondamental des algorithmes de routage. On distingue deux formes de rapidité : celle avec laquelle l'algorithme trouve le saut suivant (Next Hop) et celle relative à la mise à jour des données dans la table de routage. Les algorithmes de recherche (Lookup) ont une influence considérable sur la rapidité de routage des paquets dans les réseaux Internet.

Dans le présent chapitre, on vous propose de découvrir les principaux algorithmes présentés dans la littérature afin de répondre au critère de rapidité pour la classification de paquets. Nous commençons par un aperçu sur les solutions *Exact matching*, suivie d'une description des algorithmes *Longest Préfix Matching* (LPM) et *Range Matching*. Plus loin on met le point sur l'algorithme de recherche *Dynamique Tree Bitmap* (DTBM) qu'on choisira pour l'accélération de la recherche de route.

3.2 *Exact Matching*

L'idée principale des algorithmes de correspondance exacte est de vérifier l'appartenance d'un élément à un ensemble *(membership query)* i.e : Spécifier si une clé[1] donnée x appartient à l'ensemble de clés X.

[1] la clé est l'attribut sur lequel le tri est porté

Pour commencer on vous propose de découvrir deux structures de données classiques dont l'objectif est de réduire le nombre d'accès mémoires par recherche, B-trees et Hash tables. Puis nous enchaînerons par un type assez particulier de structures de données conçue afin exploiter de manière efficace l'espace mémoire, les Bloom Filters.

3.2.1 B-Tree

Dans le but d'améliorer les performances de la recherche, les données de la table de routage sont organisées sous forme d'une structure hiérarchique arborescente. Le mécanisme de recherche consiste alors à parcourir cette hiérarchique dont les objectifs principaux sont de gérer plusieurs niveaux d'index et de répartir équitablement les données par rapport à ces différents niveaux d'index.

A l'origine, les $B-trees$ (arbres balancées) ont été conçus pour limiter le nombre d'accès mémoires [5] [6], en organisant les clés dans une structure de données en arbre.

Le nombre maximum de branches qu'un noeud peut avoir se réfère typiquement au degré de l'arbre. Le nombre de clés stockées dans n'importe quelle nœud du tri (à l'exception du noeud racine) est limité par le degré minimum du B-tree. Par définition, chaque nœud de l'arbre peut contenir n clés, tel que : $B-1 < n < 2B-1$; $B \geq 2$, B : degré.

La hauteur d'un B-Tree contenant n clés est donnée par la formule (3.2) :

$$h \leq \log[(n+1)/2] \tag{3.1}$$

Dans la figure (FIG.3.1) un exemple de B-tree stockant les entiers multiples de 3. Les clés stockées dans un noeud sont arrangées dans un ordre décroissant. Chaque noeud interne stocke aussi un ensemble de pointeurs liés avec les clés. Chaque pointeur oriente vers un noeud fils : les clés arrangées à droite du noeud parent sont plus grande, et celles arrangées à gauche sont plus petites.

Plusieurs structures de données B-tree ont été proposés afin de répondre aux exigences de rapidité de recherche IP dans les tables de routage, on peut citer la structure de données proposé par SURI [7] effectuant une recherche LPM en un temps $O(logN)$, ou N est le

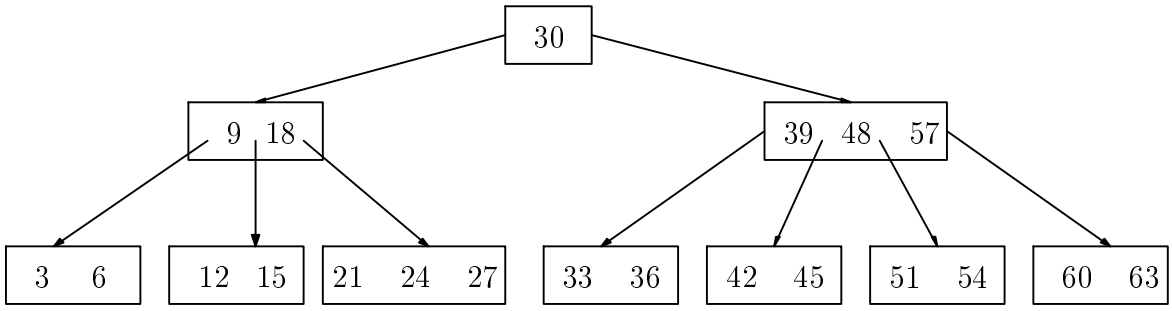


FIG. 3.1 – un exemple de B-tree stockant les entiers multiples de 3.[1]

nombre de préfixes de la table (Nombre de clés de la structure).

3.2.2 Hashing

Les techniques de hachage ont pour objectif de permettre l'accès direct à des données répondant à un critère simple d'égalité. Ces techniques sont basées sur les deux principes suivant : - l'espace destiné à stocker les données est découpé en P pages numérotées de 0 à P - 1 ; - une fonction, appelée fonction de hachage [2], permet de convertir la valeur de la clé de chaque donnée en un numéro compris entre 0 et P - 1, correspondant au numéro de la page dans laquelle la donnée est placée, ainsi le cheminement permettant de retrouver la page contenant une donnée est remplacé par un appelle à la fonction de hachage [8].

Dans notre cas lorsqu'une adresse IP est reçue, la recherche est effectuée dans une table de routage qui est l'espace contenant les pages, le routeur extrait son préfix réseau N et l'utilise comme clé de hachage. Il calculle avec cette clé une fonction de hachage h(N) et utilise le résultat du calcul comme index dans la table de routage pour obtenir l'adresse du saut suivant (Next hop), comme l'illustre la figure (FIG.3.2)

3.2.3 Bloom filter

Bloom Filter est une structure de données qui représente un ensemble de clés de manière optimal. Formulé en 1970 par Burton H.Bloom, elle a attiré l'attention des communautés de recherches pour différentes applications comme : la détection d'intrusion, et routage à base de contenu. Par l'intermédiaire de représentations implicites de clés dans

[2] Il existe plusieurs fonctions de hachages qui sont choisit selon les applications.

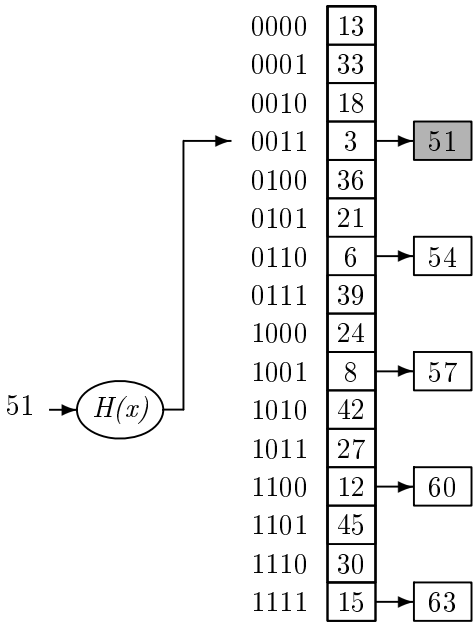


FIG. 3.2 – Exemple de l'algorithme hashing en utilisant les quatre premiers bits du MSB comme hash index[1]

un ensemble, la structure de données répond au critère d'appartenance d'un élément à un ensemble, mais n'enregistre pas l'information associée à chaque clé enregistrée. Bloom Filter est par définition : un vecteur de m bits tel que la clé x est représentée par un sous ensemble de ce vecteur.

Soit un ensemble de clé X , à n membres, on introduit une clé $x_i \in X$ dans le Bloom filter de la manière suivante :

On calcul k fonctions de hachage sur x_i, obtenant ainsi k valeurs dans l'intervalle [0 :m-1] . Chacune de ces valeurs adresse un bit unique dans le vecteur à m bits, par conséquent chaque clé x_i met k bits du vecteur de m bits à 1 . Dans la figure (FIG.3.3) on donne un exemple d'insertion de deux clés dans un Bloom Filter . Notez que, si une des k valeurs de la fonction de hachage adresse un bit qui est déjà mis à 1, ce bit reste à 1.

Afin de déterminer si une clé donnée $x_i \in X$ en utilisant les filtres est similaire au processus d'insertion décrit ci avant. Soit une clé x donnée, on génère k indices de hachage en utilisant les mêmes fonctions de hachages utilisées pour insérer les clés dans le filtre. On vérifie les positions des bits correspondant aux k indices de hachage dans le vecteur à m bits :

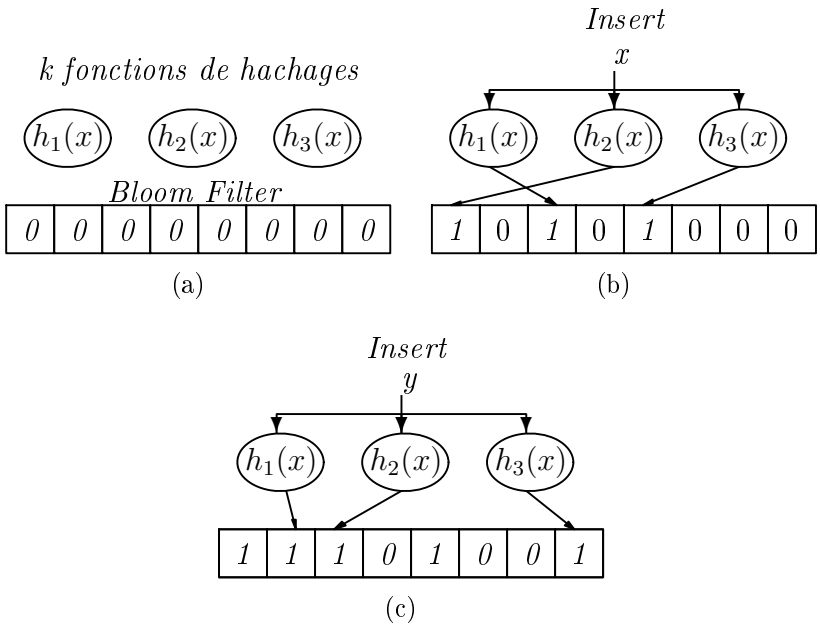


FIG. 3.3 – Exemple d'insertion de deux clés x et y dans un Bloom filter[1]

- Si au moins un des k bits est égale à 0, alors on affirme que la clé n'est pas un membre de l'ensemble.
- Si tous les bits sont égaux à 1, par conséquent la clé appartient probablement à l'ensemble des clés mais pas certainement (la probabilité $\neq 1$) ;
- Si les k bits sont égaux à 1 et x n'est pas un membre de X, alors on dit que c'est une fausse correspondance positive.

Cependant, la découverte d'un bit à 0 implique certainement que la clé n'appartient pas à l'ensemble, si c'était un membre, alors tous les k bits ont été mis à 1 lors de l'insertion de la clé dans le Bloom Filter. Un exemple des trois cas décrit précédemment est donné sur la figure (FIG.3.4)

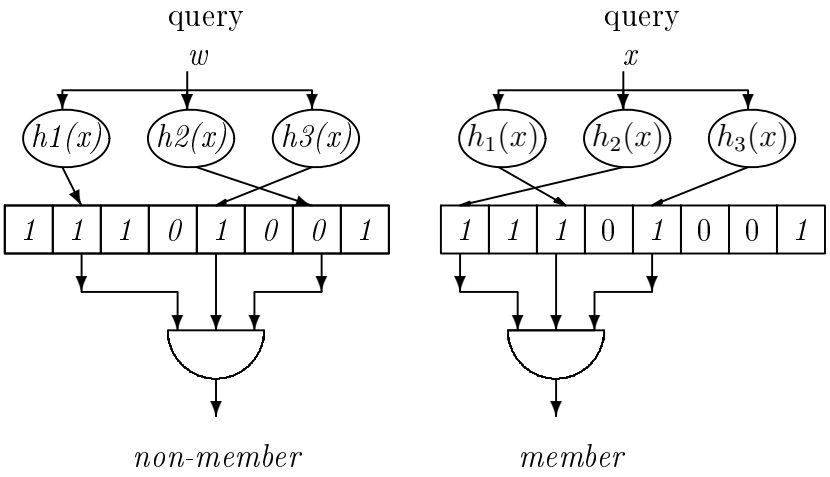
Ce qui suit est une dérivation de la probabilité d'une fausse correspondance positive, f . La probabilité qu'un bit aléatoire du vecteur de m-bits soit mis à 1 par la fonction de hachage est tout simplement $1/m$, par conséquent, la probabilité qu'il ne soit pas mis à 1 est $(1-1/m)$. La probabilité qu'il ne soit pas mis à 1 par un des n membres de l'ensemble X est $(1-1/m)^{nk}$ donc, la probabilité que ce bit soit repéré à 1 est $(1-(1-1/m)^{nk})$. Pour qu'une clé soit prétendue comme membre possible de l'ensemble, tout les k bits générés par la fonction de hachage doivent être à 1. La probabilité que ceci se produit, f

est donnée par (3.2) :

$$f = (1 - (1 - 1/m)^{nk})^k \tag{3.2}$$

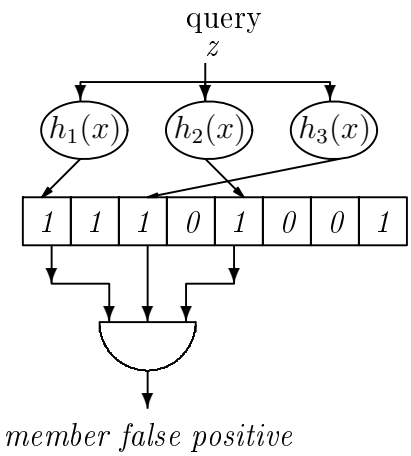
pour de grandes valeurs de m , l'équation ci-dessus devient (3.3) :

$$f = (1 - e^{-nk/m})^k \tag{3.3}$$



(a) w n'est pas un membre ,parceque le bit pointé par $h_2(x) = 0$

(b) x est membre ,il s'agit d'une correspondance exacte



(c) z n'est un membre ,il s'agit d'une fausse correspondance positive

FIG. 3.4 – Example de recherche de clés en utisant le Bloom Filter[1].

3.3 *Longest Prefix Matching* (LPM)

La correspondance par le plus long préfixe est un algorithme qui jou un rôle majeur dans l'évolution des performances des routeurs Internet. La croissance explosive d'Internet, ces dernières années a favorisé l'adoption de l'adressage CIDR (Classless Inter Domain Routing) (§ 1.3.1), qui assigne des adresses IP de longueurs variables. L'utilisation de cet algorithme permet après une recherche dans une table de préfixes de longueurs variables, de trouver le plus long préfixe correspondant à l'adresse IP de destination, afin de trouver le "next hop". Cette tâche de calcul intensif est appelé IP Lookup (Recherche IP). Dans les paragraphes suivant on met le point sur les développements les plus important dans les techniques LPM pour la recherche IP.

3.3.1 Recherche linéaire

Une recherche linéaire le long d'un ensemble de préfixes trié par ordre de longueurs décroissantes, implique nécessairement que le premier préfixe compatible avec l'adresse IP est le LPM (le plus long préfixe correspondant). Une application de cette méthode est illustrée dans la figure (Fig.3.5) . Cette technique est généralement considérée comme la plus efficace en terme de capacité mémoire qui est donnée par O(N) ou N est le nombre de préfixes dans la table. Il est à noter aussi que le temps de recherche est également O (N), ce qui implique que la recherche linéaire n'est pas une approche viable pour la recherche IP lorsque le nombre de préfixes est important.

3.3.2 Content Adressable Memory (CAM)

La mémoire adressable par contenu est incontestablement la solution de recherche la plus rapide. Actuellement, plusieurs concepteurs industriels de routeurs investissent dans cette technique de recherche IP afin de fournir des performances répondant aux critères de débits plus important. Les CAMs réduisent le nombre d'accès mémoire nécessaire pour effectuer une recherche, en comparant la clé d'entrée avec toutes les cases mémoire en parallèle (1 seul accès mémoire) ; par conséquent, une recherche efficace nécessite un cycle

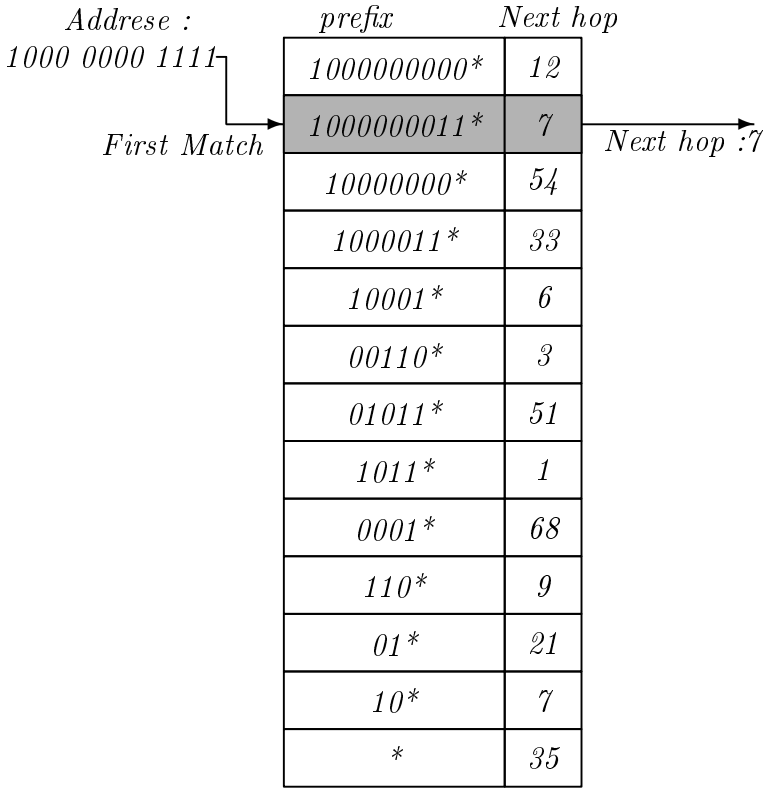


FIG. 3.5 – Exemple d'application de l'algorithme LPM pour une adresse de 12 bits en utilisant la recherche lineaire[1]

d'horloge (un temps O(1)). La seule contrainte en plus du coût important de ce type de mémoire, est la limitation en terme de capacité mémoire ne permettant pas de dépasser un certain nombre de préfixes.

3.3.3 Tree Based Schemes

Ce sont des techniques de recherche qui construisent des arbres en utilisant les bits des préfixes. La structure de données arborescente la plus courante est un arbre binaire dans lequel la valeur successive des bits de l'adresse IP définit le chemin à parcourir pour descendre l'arbre depuis sa racine jusqu'au préfixe correspondant, ce qui implique que le temps de recherche est indépendant du nombre de préfixes de l'ensemble. Dans ce cas il est plutôt lié à la longueur des préfixes par O(W), tel que W est la taille du plus long préfixe correspondant (LPM) à l'adresse IP. Un exemple d'un tri [3] est donné dans la (FIG.3.6) ,il s'agit de l'exemple illustré dans la section (3.1.1). La structure de l'arbre est obtenue

[3] un tri est un arbre ordonné dans lequel la clé stockée à chaque nœud est identifie par sa position dans l'arbre.

de la manière suivante :

Premièrement le noeud racine est inséré, il représente le préfixe (*) dans la table qui lui associe le next hop par défaut. Ensuite les autres préfixes sont insérés dans l'arbre en suivant leur séquence de bits en commençant par le MSB qui prendra la valeur 1 (à droite) ou 0 (à gauche) par rapport au noeud racine, puis le bit suivant de la même façon et ainsi de suite jusqu'a la fin de la séquence qui mènera sur le noeud représentant le préfixe qui va être marqué par un noeud noir. Ainsi les noeuds noirs représentent les préfixes de la table avec leur "Next Hop" associé.

Une recherche est menée en traversant le tri en utilisant les bits d'adresse, à commencer par le bit du poids le plus fort (MSB). Comme dans l'exemple précédent, le préfixe correspondant assorti est 1000000011 et le Next Hop associé est le 7.

Afin d'accélérer le processus de recherche, multi-bit tri schème a été développés , qui effectue une recherche en utilisant plusieurs bits de l'adresse à la fois. Srinivasan et Varghese, ont présentés deux techniques importantes pour le tri multi bit : Controlled Préfixe Expansion (CPE) et leaf pushing [9] :

CPE (Controled Prefixe Expansion) : transforme un ensemble de préfixe trié dans l'arbre en un ensemble équivalent de préfixe ayant une longueur plus courte.

La deuxième technique de Leaf pushing : permet de réduire la quantité d'informations Stockée dans chaque entrée de la table en introduisant " l'information du meilleure préfixe" assorti le long des chemins vers les nœuds de feuilles [10] .

D'autres techniques comme celle de lulea ,Eathertone [9] et tree bitmap de Dittia [11] employant les tris multi bits avec des nœuds compressés.

La technique Tree Bitmap consiste en une représentation compréssée des préfixes stockés dans chaque nœud multi bits et emploi également un système d'indexation intelligent pour réduire le nombre de pointeurs nécessaire à la recherche de next hop à deux pointeurs par noeud multi bit (pointeur des next hops et pointeur du child node) .

Cette technique a été exploitée pour développer l'algorithme Dynamique Tree Bitmap qu'on a choisi pour l'implémentation sur SOC afin d'accélérer le processus de recherche du next hop , dont les détails sont expliqués dans les sections ci-dessous.

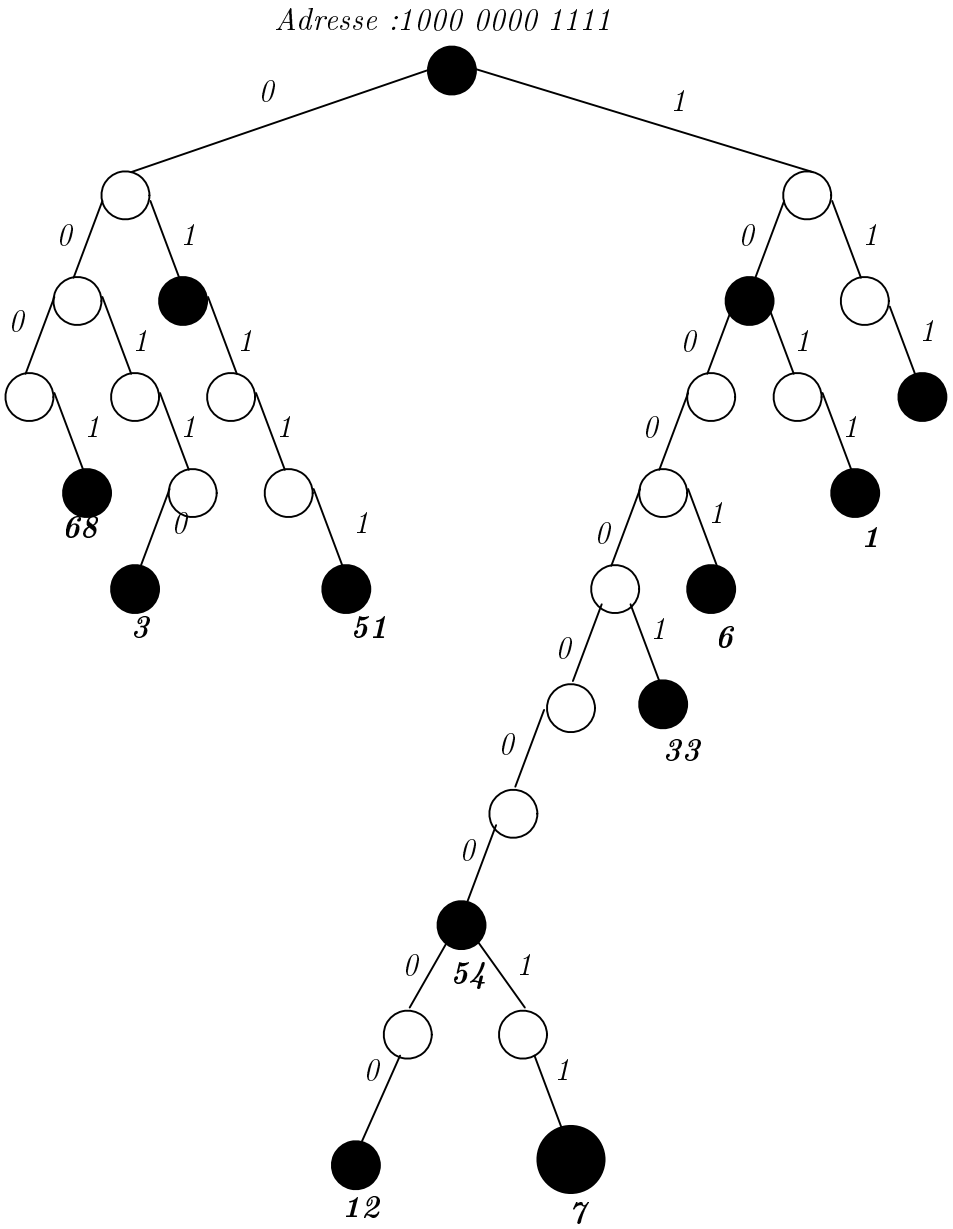


FIG. 3.6 – Example de la technique LPM utilisant un tri binaire[3]

3.3.4 Multiway and Multicolumn Search

Les algorithmes de recherche Multiway et Multicolumn sont des techniques qui ne sont pas basées sur le tri, présentées par Lampson, Srinivasan, et Varghese , ils ont été conçus pour optimiser les performances des implémentations logicielles sur des processeurs banalisés [12].

En spécifiant un ensemble de bits initiaux contigûs, les préfixes définissent des intervalles de valeurs. Par exemple, si 10* est un préfixe pour un bloc de quatre bits, alors il définit l'intervalle [1000 : 1011]. Les préfixes ne définissent jamais des intervalles de

recouvrement, sauf pour les intervalles nichés. Par exemple, [0 : 3] et [2 : 4] sont des intervalles recouverts, alors que [0 : 3] et [1 : 2] sont des intervalles nichés i.e : que l'un est inclut dans l'autre. Une des techniques d'optimisation, est d'utiliser une rangée d'index préalablement calculée [4] , dans la figure (FIG.3.7) un exemple de cette alternative pour les trois premiers bits de l'ensemble des préfixes de notre exemple précèdent de la figure (FIG.3.5) .

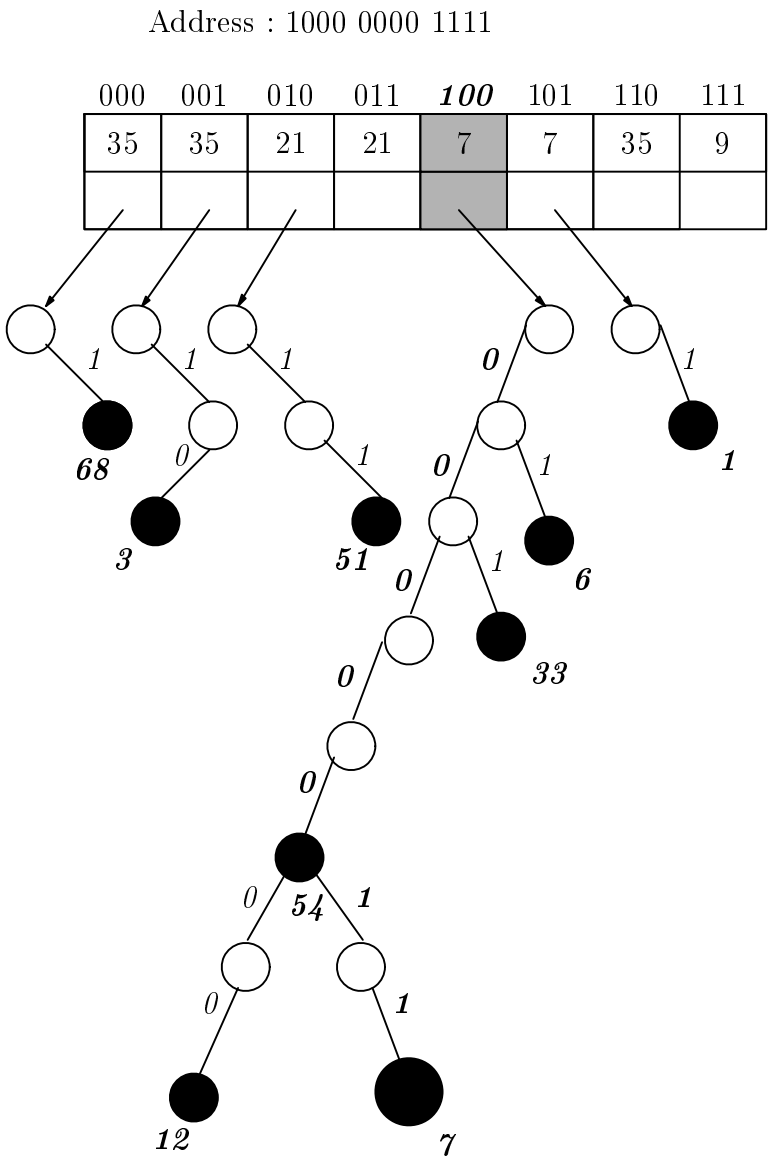


FIG. 3.7 – Recherche direct par rangée pour les trois premiers bits

Les étapes à suivre : On commence par stocker les préfixes dans un tri binaire, ensuite on exécute CPE pour une longueur de pas égale à trois. Le next hop associé avec chaque

[4] une rangé d'index préalablement calculé aussi connu sous le nom Initiale array et Direct lookup array .

nœud du niveau trois est inscrit dans la case adressée par les bits marquant le chemin depuis la racine jusqu'au nœud.

Si le nœud à des branches (child node) alors un pointeur indiquant le tri binaire contenant les branches de ce noeud est stocké. La structure est consultée en utilisant les trois premiers bits de l'adresse qui désignent une case de la rangée.

Si aucun pointeur n'est stocké alors le next hop est celui de cette case, par exemple la case désignée par 111_2 dans de la figure (FIG.3.7). S'il existe un pointeur dans cette case, alors le next hop est considéré comme le meilleur correspondant jusqu'ici et la recherche continu dans le tri désigné par le pointeur, si la case désignée par 100_2 dans l'exemple de la (FIG.3.7).

NB

: Cette structure de données nécessite $2^a * q$ bits de capacité mémoire ou :

a : est le nombre de bits utiliser pour indexer les cases ;

q : est le nombre de bits nécessaire pour le stockage de next hops et du pointeur.

En générale cette approche nécessite un temps de recherche $O(W + \log N)$ et un espace mémoire $O(2N)$, tel que N : le nombre de préfixe W : longueur de l'adresse. La contrainte fondamentale de cette technique est la relation linéaire avec la longueur d'adresse.

3.4 Range Matching

Les solutions de correspondance par intervalles parviennent généralement dans beaucoup de problèmes de recherches dans les domaines des réseaux informatiques, géométrie informatique , conception de base de données. Il existe plusieurs types de solutions Range matching. Dans ce qui suit on fait un bref aperçus sur les différentes approches de cette techniques. Soit un ensemble X d'intervalles fermées $[i,j]$ et un point p, cet algorithme consiste à trouver tout les intervalles de X contenant le point p . On distingue deux structures de données classiques [13][14] :

– Segment tree ;

– Interval tree.

3.4.1 Segment Tree

C'est une structure de données qui stocke un ensemble de segments sur l'axe des réels [1] . Dans notre cas, il s'agit d'un ensemble d'intervalles fermés X. Cette technique utilise spécifiquement comme structure de données des arbres de recherche binaires. Pour utiliser de telles structures de données, les deux extrémités des segments sont projetées sur l'axe des réels afin de former des intervalles élémentaires non recouverts. Soit un ensemble de segments X contenant |X| segments, l'ensemble Y d'intervalles élémentaires contient au plus (2|X|-1) segments. Un exemple est illustré dans la figure (FIG.3.8)

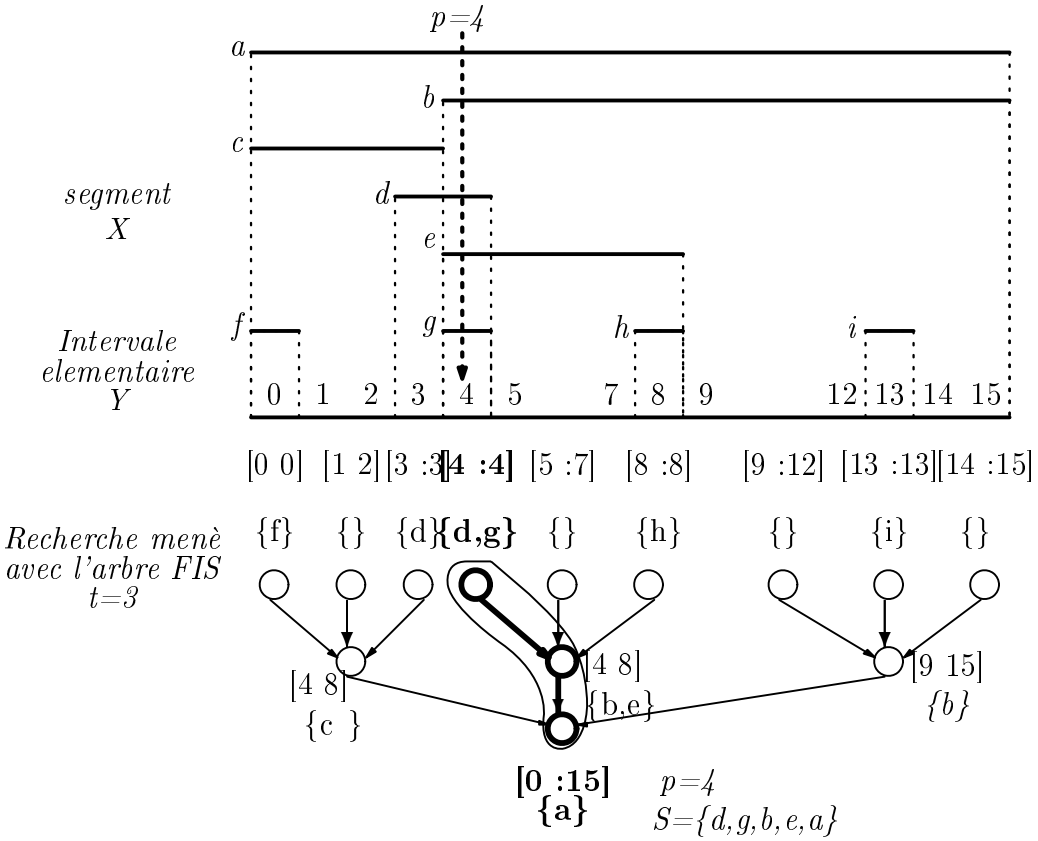


FIG. 3.8 – Projection des bornes des intervals pour former des segments non recouvert sur l'axe des réels[15].

La recherche dans un arbre de segments stockant les intervalles élémentaires [15] donne un ensemble de segments correspondant S pour un point donnée p dans un temps O($Log|y|$) tel que $\forall [i,j] \in S.i \leq p \leq j$.

Afin d'améliorer la performance de la recherche et de la mise à jour, Feldman and Muthukrishnan ont proposé l'arbre FIS (Fat Inverted Segments)[14].

L'arbre FIS est un B-tree de rang t=3. Un exemple d'arbre FIS est illustré dans la figure (FIG.3.8) . Chaque noeud v représente un intervalle I(v) qui est l'union des intervalles représentés par leurs branches .

Dans notre cas, les caractéristiques les plus importantes de FIS tree sont : la hauteur de l'arbre qui peut être limitée par un bon choix de t. Chaque noeud v stocke uniquement un intervalle X si $I(v) \subseteq x$ et $x \subset I$ (parent de v). L'ensemble de segments S recouvrant un intervalle élémentaire y peut être trouvé en parcourant le chemin depuis la feuille représentatif y jusqu'à la racine de l'arbre (i.e le chemin inverse) et annexant l'ensemble de segments stockés à chaque noeud v vers S . Dans la figure (FIG.3.8) on présente un exemple pour p=4.

Pour M= (2|x|+1), l'arbre FIS nécessite un temps de recherche O($log_t(M)$) , un temps de mise à jour O($M.log_t(M)$) et un espace mémoire de O($M.log_t(M)$).

3.4.2 Interval Tree

Un arbre d'intervalles stocke un ensembles d'intervalles fermés X en utilisant comme structure de données fondamentale B-tree binaire. A la différence de l'arbre de segments, cette arbre n'utilise pas des intervalles élémentaires ; chaque noeud dans cet arbre stocke un intervalle $x \in X$. La borne inférieur de l'intervalle est utilisée comme clé pour les noeuds du B-tree. Afin d'accélérer la recherche, les noeuds de l'arbre stockent généralement des variables additionnelles tel que la valeur maximale de toutes les bornes des intervalles stockés dans leurs sous arbres. Un exemple d'arbre d'intervalles est donnée dans la figure(FIG.3.9). Pour trouver l'ensemble S de tous les intervalles correspondant pour un point p donné , on suit les étapes supplémentaires. Premièrement on localise l'intervalle correspondant au point p et qui est stocké dans le noeud extrême gauche dans l'arbre. A partir de ce noeud, on parcourt les noeuds de l'arbre dans l'ordre, on s'arrête dans un noeud dont la clé est plus grande que p ou dans le dernier noeud de l'arbre. Dans la figure (FIG.3.9) un exemple est prit pour p=4.

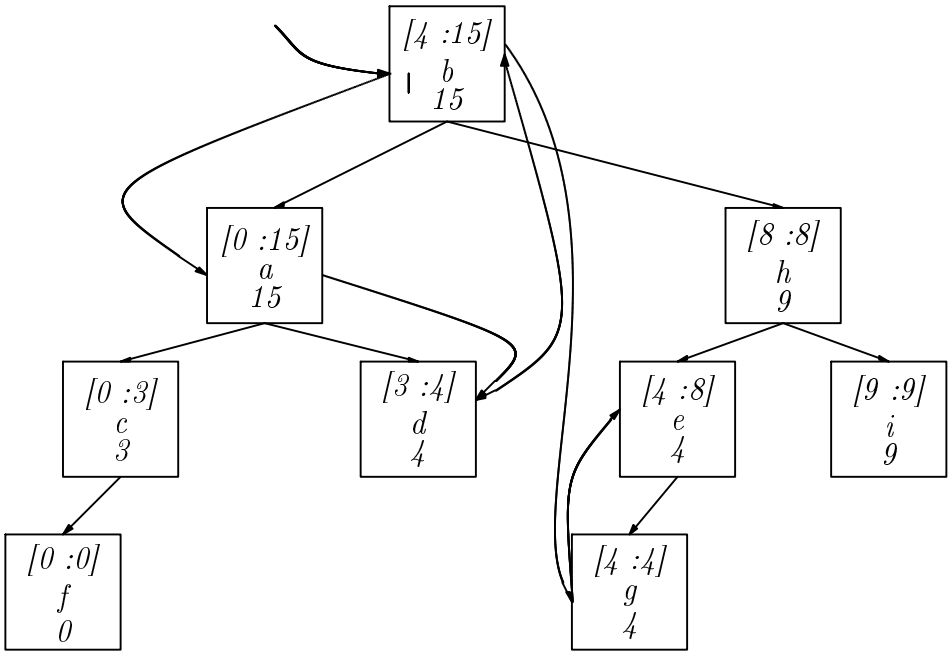


FIG. 3.9 – Interval tree[15]

Si $|S|$ est le nombre d'intervalles correspondant, la recherche nécessite un temps $O(log|X| + |S|)$. L'arbre d'intervalles nécessite un temps de mise à jour O(log|X|) et un espace mémoire O(|X|) [1].

3.5 Comparaison

Les techniques de recherche étudiées ci-dessus présentent des performances aussi diverses que variées, dont les avantages et les inconvénients s'évaluent en fonction de trois paramètres : le temps nécessaire pour une recherche, la capacité mémoire requise et le nombre d'accès mémoire dans la structure de données. Dans les algorithmes d'Exact Matching on trouve les Bloom Filter qui se caractérise par une utilisation optimale de la mémoire avec des temps de recherche assez performant, mais la seule contrainte avec ce type de filtre c'est la notion de probabilité qui peut engendrer des résultats de recherche erronés. Les B-tree réduisent considérablement le nombre d'accès mémoires, permettant ainsi la réduction du temps de recherche. Les techniques de hachage fournissent des temps compétitif mais l'espace mémoire disponible est limité. La technique LPM (Longest Prefix Matching) est idéal pour les recherches dans des tables de routages contenant des préfixes

de longueurs différentes (CIDR). La recherche linéaire est connue pour ces performances en matière de gestion de l'espace mémoire, sauf que le temps de recherche pour cette technique est linéairement lié au nombre de préfixes de la table ce qui présente un inconvénient majeur pour les tables de tailles importantes. Les CAMs sont aucun doute les meilleurs solutions pour la recherche IP dans des tables de tailles moyennes sauf que le coût très important ainsi la limitation de capacité mémoire ne permet de les utiliser pour les routeurs Gigabit. Les techniques Tree Based Schemes ont des temps de recherche indépendant du nombre de préfixes de la table ce qui les rand avantageuses pour les tables de routages de taille importante. Pour les solutions Range Matching les temps de recherches et l'espace mémoire requis sont liés au nombre de préfixes, ce qui est un inconvénient pour les tables de routages de taille quelconque.

3.6 Conclusion

D'après l'évualuation des perfomances de differentes techniques de recherche ,les techniques Tree Based Schemes se sont avérées comme étant les meilleurs, notamment pour des tables de routage de très grandes taille et contenant des préfixes de longueurs différentes, ce qui les rends parfaitement adéquates pour l'adressage CIDR. Par conséquent, on a opté pour une technique Tree Based Schemes dite Dynamique Tree Bitmap (DTBM) qu'ont détaillera dans le chapitre suivant.

Chapitre 4

Dynamic Tree Bitmap *(DTBM)*

4.1 Introduction

Le routage des paquets IP est le rôle fondamental des routeurs Internet .Il consiste en l'acheminement de chaque paquet le long d'un itinéraire, constitué de sauts successives menant de la source vers la destination. La vitesse à laquelle les paquets traversent le routeur (i.e temps du choix du port de sortie) est fondamental dans l'évaluation des performances de ce dernier . Afin d'améliorer ces performances, on fait appel à une solution algorithmique, cette alternative sera ensuite implémentée sur SoC (System on Chip) dans le but d'optimiser le temps de recherche de route. Dans ce chapitre on introduit un nouvel algorithme, Dynamique Tree Bitmap (DTBM),une solution developpée à partir des techniques Tree Based Schemes, permettant d'exécuter une recherche LPM à très grande vitesse avec une utilisation efficace de la mémoire .

4.2 Structure de données

Un routeur Internet oriente les datagrammes IP vers les ports de sortie correspondant en se basant sur deux paramètres : l'adresse IP de destination et les informations de la table de routage .Chaque ligne de cette table est constituée d'une paire (filtre d'adresse, Next Hop correspondant). On considère que le filtre est le préfixe d'une adresse de destination, dont le rôle est de sélectionner toutes les adresses de destinations ayant le même préfixe

que le filtre.

La table de routage constitue la base de donnée du routeur ,qui va être représenter sous forme d'arbre binaire. Il s'agit d'un tri ou le parcours est déterminé par la donnée elle même ,la figure (FIG.4.1) illustre la représentation de l'ensemble de préfixe R=*,01*,1* en tri binaire. Le préfixe x est la séquence de bits représentant le chemin dans l'arbre menant de la racine jusqu'au noeud correspondant au préfixe. Dans le figure (FIG.4.1)(a) , les noeuds en noire représentent les préfixes de l'ensemble R.

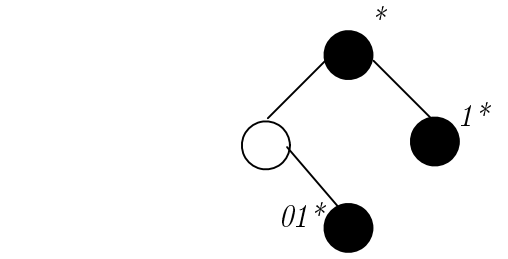


FIG.4.1.(a) Trie binaire representant l'ensemble des prèfixes R

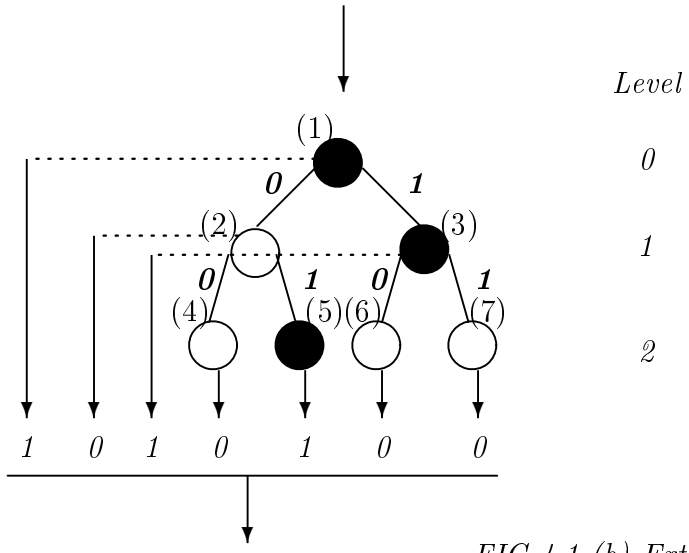


FIG.4.1.(b) Extension de trie (a) en trie multi bit de hauteur h=2

FIG. 4.1 – Extension d'un tri mono-bit en un tri multi-bits binaire de hauteur h=2

En Complétant le tri ,en noeud multi-bit (FIG.4.1)(b) avec une hauteur $h = 2$, le nombre de noeuds devient égale à $2^{h+1} - 1 = 7$ noeuds ;en les numérotant de 1 à 7, on pourra les représenté par un vecteur de 7 bits appelé l'IBM qu'on explicitera plus tard.

La structure de donnée est construite de la manière suivante : Premièrement, les préfixes de la table de routage sont triés dans l'arbre binaire. Deuxièmement l'arbre binaire

sera décomposé en partitions de hauteur h,les partitions contenant un tri non complet sera prolongé en noeuds multi-bits, ainsi toute les partitions de l'arbre auront un tri qui contient le même nombre de noeuds donc chaque partitions contiendra $w = 2^{h+1} - 1$ noeuds.

Les noeuds constituants le tri de chaque partition seront représentés par une séquence de w bits appelées *Internal Prefix Bitmap* (IBM). Telque chaque noeud de la partition est représenté dans l'IBM par un seul bit [16].

Le noeud racine du tri multi-bit sera représenté par le bit MSB de l'IBM et le dernier noeud le plus à droite dans le tri sera représenté par le bit LSB de l'IBM. Un noeud noire (un préfixe) est représenté dans l'IBM par un bit égal à 1, un noeud blanc (vide) est représenté dans l'IBM par un bit égal à zéros. Ainsi, chaque préfixe du tri multi-bit est repéré par la position de son bit correspondant dans l'IBM . Par exemple, l'IBM du tri de la (FIG.4.1)(b) est égale à 1010100. Par conséquent, après avoir représenter les tris multi-bit de chaque partition, on peut se déplacer à travers les noeuds du tri en se basant sur la position des noeuds dans l'IBM de la manière suivante :

Si on se trouve dans le i^{eme} bit, on peut se déplacer dans l'arbre avec la formule suivante,en changeant la position (i) du noeud courant comme suit :

- Vers le noeud fils à gauche (left child) :$2i$
- Vers le noeud fils à droite (right child) :$2i$+1
- Pour un déplacement vers le noeud parent : $i/2$ si i est pair et $(i-1)/2$ si i est impair (FIG.4.1)(b).

Chaque tri de ces partitions peut être représenter par un format unique ,qui est le noeud DTBM . Ce noeud est constitué des champs suivants :

- L'IBM représentant les neouds de chaque tri : $2^{h+1} - 1$ bits
- Le tableau des next hops qui renferme le numéro de port de l'interface de sortie pour chaque préfixe : $2^{h+1} - 1$ cases
- Le tableau des pointeurs, pour sortir du noeud DTBM courant vers les noeuds DTBM générés par celui-ci : 2^{h+1} pointeurs

Dans la figure FIG.4.2 ,on présente le format générale du noeud DTBM .

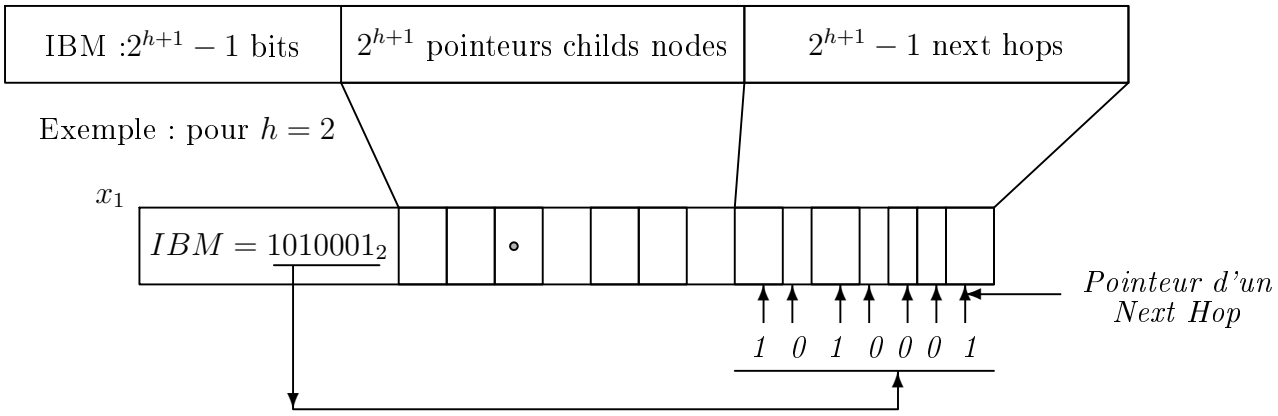


FIG. 4.2 – Format du noeud DTBM[18]

4.2.1 Mise en forme des noeud DTBM à partir d'une table de routage

On considére une table de routage contenant 9 préfixes ayant différentes longueurs comme illustré dans le tableau (FIG.4.3) .

Préfixe	Next Hop
*	35
1*	19
01*	7
0010*	15
1110*	12
1111*	33
00111*	21
11101*	14
1110110*	5

FIG. 4.3 – Table de routage

La première étape est de représenter les préfixes de cette table dans un arbre binaire . Apres la construction de l'arbre ,on le décompose en partitions équivalentes de hauteur $h = 2$:tous les tris doivent être complétés par des noeuds vides (blanc) de manière à ce que chaque partition contienne 7 noeuds car h=2(i.e : $2^{h+1} - 1$). Ainsi l'arbre binaire est

décomposé en 4 partitions équivalentes x_1,x_2,$x_3$et x_4 contenant chacune 3,2,3 ,1 préfixes respectivement avec leurs next hops, comme schématisé dans la figure (FIG.4.4). Chaque partition de l'arbre binaire sera représentée par un noeud DTBM contenant l'IBM approprié et le tableau de pointeurs qui sert à orienter vers les childs nodes de la partition courante .

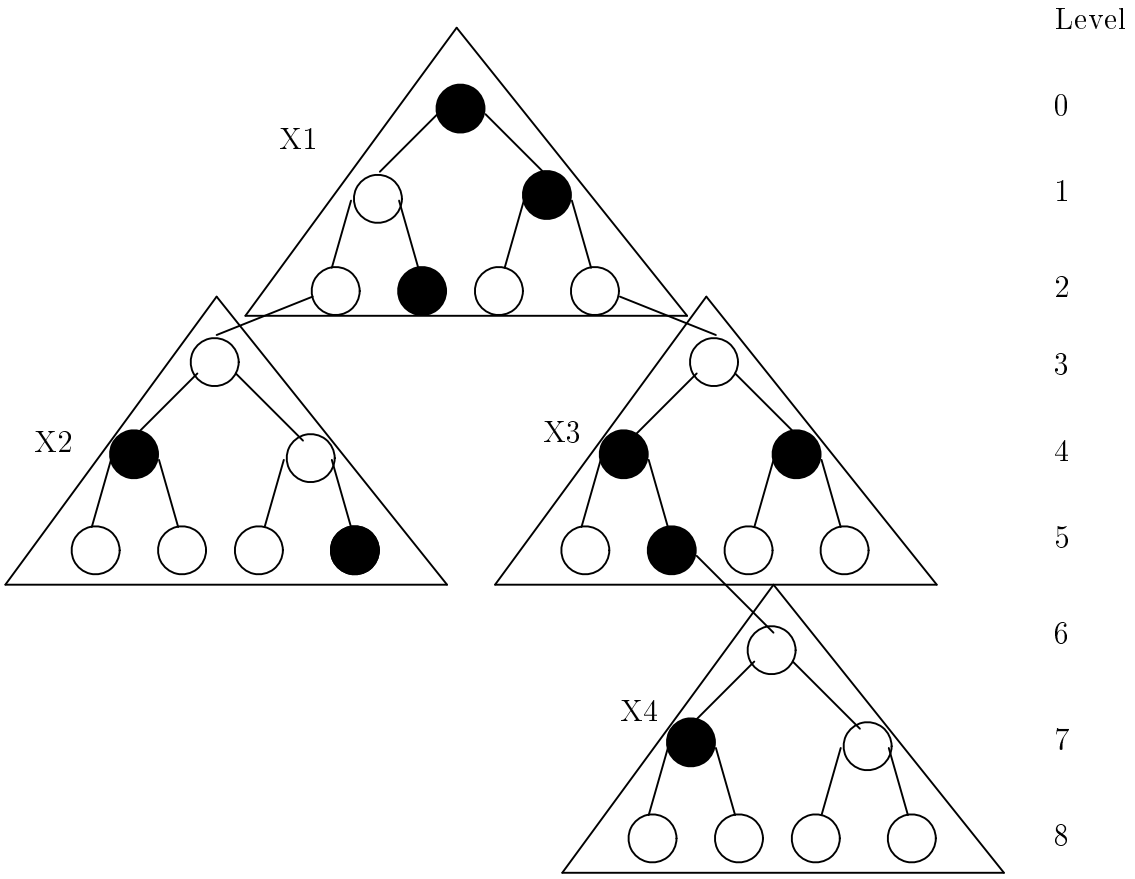


FIG. 4.4 – Organisation des prefixes en arbre binaire

La figure (FIG.4.5) montre les quatre noeuds DTBM qui représentent la table de routage précédente.

4.3 Algorithme de recherche et de mise à jour DTBM

Dans cette partie on présente les deux algorithmes relatifs à cette technique. Celui de la mise à jour (*algorithme Insert*) qui est appliqué afin de construire les noeuds DTBM

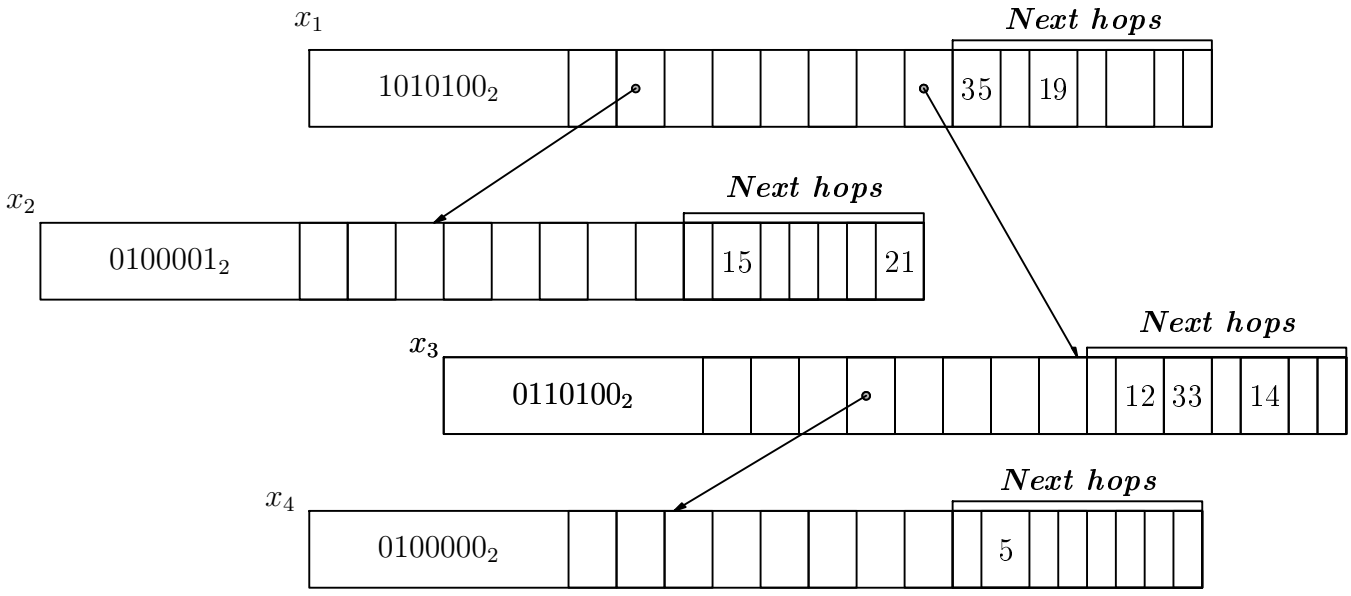


FIG. 4.5 – Les noeuds DTBM construits a partir de la table de routage

à partir d'une table de routage . Le second (*algorithme Lookup*) est celui de la recherche du plus long préfixe correspondant à une adresse d [16].

4.3.1 Lookup

A premier lieu, on considère que la structure de donnée est construite , à partir d'une table de routage. La recherche de plus long préfixe correspondant (LPM) à une adresse d consiste a parcourir l'arbre binaire de haut en bas ,en commençant par le premier noeud DTBM et en utilisant les bits de l'adresse de destination d du bit MSB vers le LSB ,en suivant les étapes décrites ci-dessous, on considère pendant toutes les étapes que les termes *partition* et *noeud DTBM* sont équivalente , on prend la hauteur des partitions $h = 2$:

Premièrement on cherche dans la première partition, la position du plus long préfixe de d (i.e :la position du bit dans l'IBM). Pour ce faire,on prend les deux premiers bits de d pour indexer les noeuds du niveau h=2 (i.e : les noeuds (la fig 4.1.b) d'indexe 4,5,6,7) du noeud DTBM. Puis on remonte vers le haut du tri jusqu'à trouver un préfixe (ce préfixe est le plus long des préfixes assorti avec d dans le noeud DTBM courant). Pour cela, on calcule la position du noeud représentant ces deux bits dans le tri de la partition courante par la formule :$i = 2^h + d.bits(h)$,tell que $d.bits(h)$ reperesente les (h) premiers bits de d convertis en décimale. Une fois la position du noeud trouvée ,on effectue un test sur le

bit représentant ce noeud afin de vérifier si ce noeud est un préfixe de la table de routage ou pas :

- Si l'i^{eme} bit de l'IBM ($IBM(i)$) est =1 ,alors ce noeud est un préfixe et c'est le LPM de d dans cette partition ;dans ce cas on enregistre l'index i qui sera considéré comme le meilleur résultat de recherche jusqu'ici.
- si $IBM(i)$=0, alors ce noeud est vide (ce n'est pas un préfixe),on doit donc remonter vers le noeud parent en changeant la position i par $i/2$, et le test est refait avec le bit $IBM(i)$ et ainsi de suite jusqu'à trouver un bit =1 ,qui sera considéré comme dans le cas précèdent.

Donc la recherche dans cette partition se clôture par : la position (i) de LPM d dans l'IBM , cette fonction est réalisée par l'algorithme POS, cette position servira par la suite à indexer dans le tableau des next hops, le next hop associé au préfixe LPM trouvé. Une fois la recherche dans la première partition terminée ,on entame la recherche dans la partition qui suit ,elle sera sélectionnée par le pointeur de la première partition dont la position dans le tableau des pointeurs est d.bits(h) ; arrivé à ce stade on se déplace de la partition (noeud DTBM)courante vers celle pointée.

Aprés avoir changer de noeud DTBM, l'adresse d est décalée vers la gauche de 3 bits, puis on refait la même opération de recherche dans ce nouveau noeud avec les deux nouveaux premiers bits de d, elle donnera comme résultat la position (i>0) du bit de l'IBM qui représente le plus long préfixe correspondant à d dans le noeud DTBM courant, dans ce cas ce résultat (i) remplace celui trouvé dans le noeud DTBM précédant. Mais si jamais la position trouvée i est égale à 0, alors il n'existe pas de préfixe correspondant à d dans le noeud DTBM courant (car les bits de l'IBM sont indexés par i : de 1 à 7),et le résultat de recherche dans le noeud DTBM précédant (i) est retenu [16].

Quelque soit le résultat de la recherche dans le noeud DTBM courant ,on poursuit la recherche en changeant le noeud DTBM courant par celui indiqué par l'élément du tableau des pointeurs indexé par $d.bits(h)$. Si ce pointeur se dirige vers un autre noeud DTBM ,alors la recherche s'effectue dans cette partition similairement à la précédente. Or si le pointeur est vide ,alors cela signifie qu'il n'y a plus de noeud DTBM pour continuer

la recherche ,on est arrivé au bout de l'arbre FIG.4.4. La recherche se close ici,en donnant comme résultat la position (i) du dernier préfixe correspondant à d (qui est le plus long dans l'arbre),ainsi que son next hop associé désigné par :$nexthops(i)$.

Ci-dessous est illustré l'algorithme de recherche *(Lookup)* ,pour un arbre binaire structuré en noeuds DTBM, tel que :

- node.pointeurs(i) est :l'i^{eme} pointeur du tableau des pointeurs ,il permet de poursuivre la recherche dans un autre noeud DTBM généré par le noeud courant ;
- next hops(i) est : l'i^{eme} entrée du tableau des next hops du noeud DTBM courant ;
- l'index du tableau des pointeurs est comprit de 0 à 7 .
- l'index du tableau des next hops est comprit de 1 à 7 ,comme celui des bits de l'IBM ;
- la fonction POS ,calcule la position du plus long préfixe correspondant à d dans l'IBM du noeud DTBM courant,cette position sert à indexer le next hop dans le tableau des next hops.

l'algorithme lookup se présente comme suit :

Algorithme lookup

```
node =root;
do {t=node.IBM.POS(d);
if (t\neq 0)
{y=node;hopIndex=t;}
node=node.pointeurs[d.bits(h+1)];
shift d left by h+1 bits;
}
while (node\neq null);
return y.nexthops[hopIndex];
end
```

l'algorithme POS se présente comme suit :

Algorithme $POS(d)$

```
i=2^{h}+d.bits(h);
while (i\neq 0 and (IBM(i)==0))
{i/=2;}
```

```
return i;
end
```

4.3.2 Organigramme du l'algorithme DTBM

Comme schématiser dans la figure (FIG.4.6)

Exemple d'application de l'algorithme *lookup*

Soit à touver la route (next hop) d'une adresse d = 11101110... à rechercher dans la table de routage FIG.(FIG.4.3) . Les préfixes de cette table sont triés dans un arbre binaire (tri) avec des partitions de hauteur $h = 2$ (FIG.4.4), ensuite les noeuds DTBM sont structurés à partir de l'arbre binaire (FIG.4.5).

Pour commencer ,la recherche est menée dans la partition représentée par le noeud DTBM x_1 ,en utilisant les deux (h=2) premiers bits : (11) de d ,on calcule la position du plus long préfixe correspondant à d par $x1.IBM.POS(d)$ qui donne t=3 ,ainsi hopindex=3 désignant le 3^{eme} élément du tableau des nexthops, y correspond au noeud x_1 ,il sert à enregistrer le dernier noeud DTBM ou la recherche a été fructueuse .Puis , on a $d.bits(h+1)$=$d.bits(3)$ =7 (qui donne l'accès vers le noeud DTBM suivant), $x_1.pointeurs(7)$ nous dirige vers le noeud x_3 . Ainsi le noeud courant devient x_3 et l'adresse d est décalée de $h+1 = 3$ bits vers la gauche et devient 01110... qui vont correspondre à la recherche dans x_3.

On suit la même procédure dans le noeud courant x_3 en utilisant les deux premiers bits de d (01), $x_3.IBM.POS(d)$ donne t=5 d'ou hopindex =5 et on met à jour $y = x_3$ puisque la recherche dans ce noeud a donné des resultats . Ensuite on change de noeud, $d.bits(h+1) = d.bits(3) = 3(011)_2$ donc $x_3.pointeurs(3)$ sélectionne x_4 .

Le noeud courant est maintenant $x4$ et l'adresse d est décalée de trois bits vers la gauche qui devient 10... ,$x_4.IBM.POS(d)$ donne t=0 ,ce qui implique qu'il n'y a pas de préfixe correspondant à d dans le noeud x_4,on calcule l'index du pointeur pour continuer dans un autre noeud DTBM , or tout les pointeur de x_4 sont vides, ($x_4.pointeurs(i) = null$) on sort de la boucle *do*, et on trouve enfin la sortie (next hops) donnée par $y.nexthops[hopIndex]$

qui est la 5^{eme} entrée de la table des next hops du noeud DTBM $x3$, ($x_3.nexthops[5]$).

4.3.3 Algorithme de mise à jour

Cet algorithme a pour rôle de construire les noeuds DTBM et les mettre à jour ,à partir d'une table de routage en insérant les préfixes dans les noeuds , selon les étapes suivantes :

1. Si la longueur du préfixe est supérieur à $h+1$,ce qui implique que le préfixe ne sera pas introduit dans le premier noeud DTBM, alors on calcule l'index du pointeur menant au noeud DTBM suivant et le préfixe est décalé de $h+1$ bits vers la gauche . Le test sur sa longueur est refait dans ce neoud DTBM, et ainsi de suite jusqu'à ce que sa longueur soit inférieur à $h+1$. Dans ce cas, le préfixe sera inséré dans le noeud DTBM courant, comme l'indique l'étape suivante .

2. Si la longueur du préfixe est inférieur à $h+1$, alors on affirme que le préfixe doit être insérer dans le noeud DTBM courant . La position du noeud représentant la partie restante du préfixe (après décalage) par $i = 2^h + P.bits(h)$, une fois cet index calculé,le i^{eme} bit de l'IBM est mis à 1, et le next hop associé est inséré dans la i^{eme} case du tableau des next hops.

Algorithm insert(p)

```
if (root is null) root=new DTBMnode;
node = root;
while (p.length = s and node.pointeurs[p.bits(s)]= null) {
node = node.pointeurs[p.bits(s)];
shift p left by s bits;
}
if (p.length =s){
do {
node = node.pointeurs[pbits(s)] = new DTBMNode;
shift p left by s bits;
} until (p.length <s)
}
Insert p into node;
```

4.4 Evualuation des performances

La structure de donnée explicitée ci-dessus s'applique pour la représentation des tables de routage dynamique de routeurs exigeant des débits de recherches et de mises à jour très élevés. Cet algorithme est capable d'effectuer des recherches et des mises à jour au même débit, avec un temps de recherche très compétitif avec celui des autres solutions algorithmiques qu'on a traité dans le chapitre 3. Le temps de recherche peut être évaluer à $O(N)$, ou N est la taille du plus long préfixe correspondant à une adresse d. L'un des avantages principaux de cette technique est que le temps de recherche est indépendant du nombre de préfixes de la table de routage, ce qui le rend idéal pour les routeurs Gigabit des grands réseaux de transition qui contiennent des tables de routage de taille importante. La structuration des éléments de la table de routage en noeuds DTBM permet la distribution des données en mémoire de manière optimale. L'espace mémoire requis pour cette structure est $O(N)$, qui est indépendant du nombre de préfixe de la table de routage ce qui permet d'ameliorer la gestion de la mémoire pour des routeurs Gigabits. Le nombre d'accès mémoire pour la recherche de route dépend de la hauteur h des partitions ainsi que la taille du plus long préfixe. Un bon choix de la hauteur h permet de minimiser considérablement le nombre d'accès mémoire. Des parallélisme de recherche sont envisageable afin de réduire le temps de recherche.

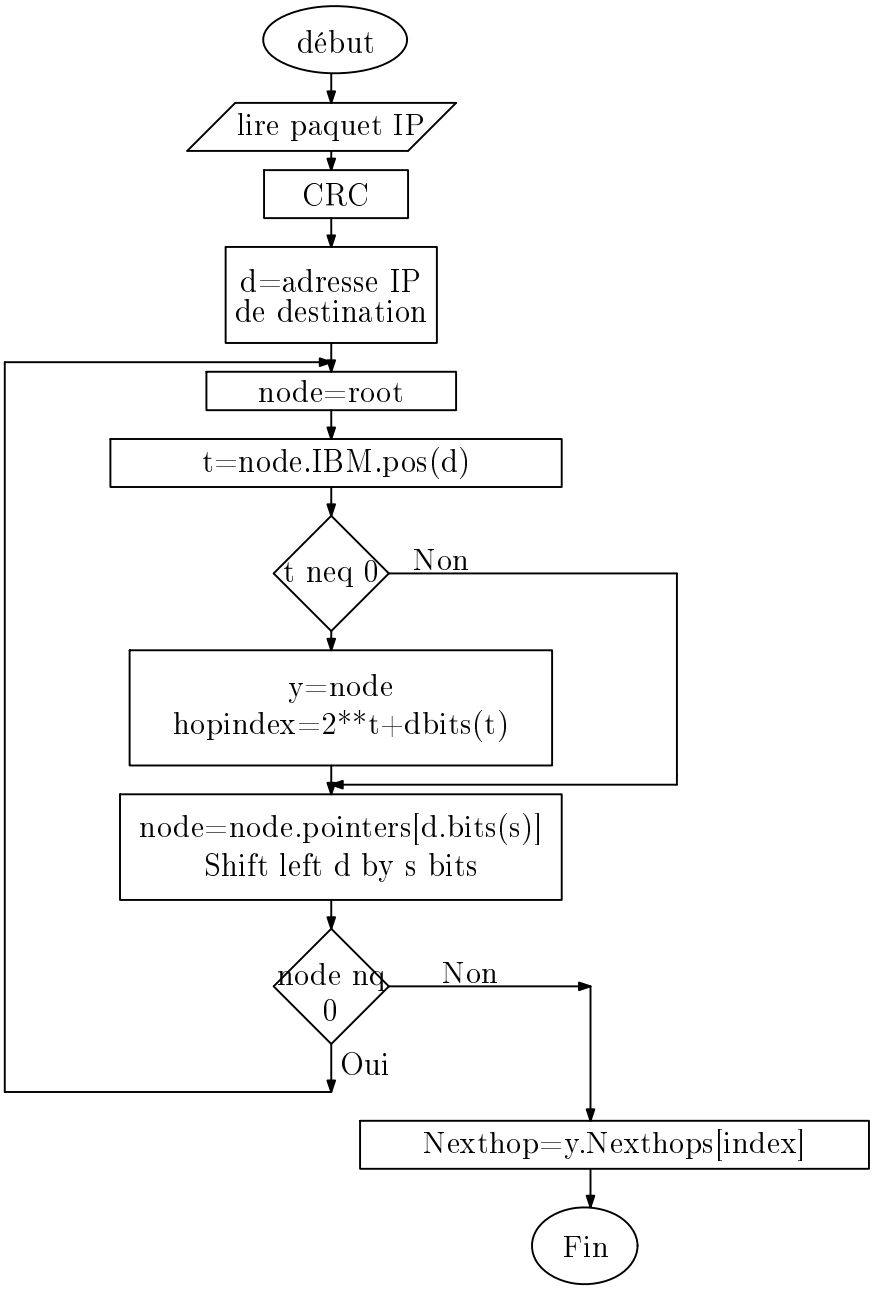


FIG. 4.6 – L'organisgrame de l'algorithme DTBM

4.5 Conclusion

Dans ce chapitre, nous avons pu exposer en détails l'algorithme de recherche *Dynamique Tree Bitmap* (DTBM), et passer en revue les différentes étapes de la recherche et de la mise à jour de la base de donnée. Comme nous l'avons souligné précédemment, cette technique est parfaitement applicable aux routeurs Gigabit des réseaux centrales. Ainsi d'après les performances évaluées de cette technique, elle sera choisit par la suite afin d'accélérer le temps de recherche de route de notre routeur, notamment après l'avoir implémentée sous forme d'une application software sur SoC (Systéme on Chip).

Chapitre 5

Implémentation sur SoC

Les progrès réalisés en microélectronique permettent aujourd'hui d'intégrer des millions de transistors sur un seul circuit. Les évolutions technologiques ont particulièrement favorisé la famille des circuits programmables FPGA(Field Programmable Gate Array). La logique programmable est ainsi devenue une solution incontournable, dès lors que les volumes de production sont faibles, ou que la reconfigurabilité et les coûts non récurrents réduits ainsi que le temps de conception font parti des critères de décision[17]. Dans ce chapitre nous allons étudier la solution technologique utilisé pour l'implémentation de notre routeur ainsi qu'une présentation de l'outil de développement pour finir par la mise en oeuvre de la plateforme matérielle et software.

5.1 Approche de développement

Avec l'évolution des densités d'intégration en microélectronique, on est aujourd'hui capable de concevoir des systèmes entier sur une même puce ; d'ou la naissance des "Systems on Chip". Notre approche est basée sur l'utilisation d'un circuit programmable de type FPGA, c'est le XC3S200. Ce circuit est fourni sur la SPARTAN 3 Starter Board qui chargera un noyau soft (MicroBlaze). Le tout nous donne une plateforme matérielle prête à accueillir les applications software.

5.1.1 La solution On Chip

Un SoC (System on Chip) est un ensemble de blocs fonctionnels intégré dans un composant électronique avec au moins un processeur. Ainsi, un systéme en SoC typique contient un ou plusieurs noyaux de processeurs (Microblaze ou Power Pc), un nombre arbitraire de périphériques, une mémoire on chip et des bus qui interconnectent tout ces dispositifs. Ces éléments se trouvent sous la forme de coeurs, aussi appelés "IP cores". En pratique un System on Chip peut aussi contenir une multitude d'interfaces entrées/sorties vers d'autres circuits (périphériques Off Chip).

Le but de la conception d'un System on Chip est d'utiliser un seul circuit pour l'integration d'un système en entier y compris son application.

Un System on Chip travail généralement à une fréquence d'horloge beaucoup plus faible qu'une CPU standard. En effet, cette dernière nécessite une fréquence entre 500 MHZ et 3GHZ, tandis qu'une CPU SoC nécessite moins de 100MHZ.

Avec une faible fréquence de travail, la consommation de l'énergie ainsi que la température du circuit sont réduites, ce qui diminue considérablement les besoins en refroidissement et en énergie donc les frais d'exploitation[18].

L'un des principaux objectifs de la conception SoC, est d'intégrer autant de composants que possible à l'intérieur de l'FPGA pour permettre une meilleur flexibilité. Les périphériques constituants la plateforme matérielle sont des IP cores qu'on intègre à partir d'une conception personnelle ou d'une bibiothéque.

On peut en dégager deux grandes catégories :

-Les noyaux Hard, qui ont une disposition physique (layout) fixe, et qui sont incorporé à la conception en tant qu'une puce standard. L'inconvénient d'un IP HARD est qu'il est beaucoup moins portable que les autres catégories d'IP à cause de sa dépendance vis-à-vis d'une technologie cible. Un "hard core" est perçu par le développeur d'un système comme une boîte noire.

-Les noyaux Firm, qui sont délivrés en tant qu'élément de librairie.

-Les noyaux Soft, qui sont fournis sous forme de Netlist, ou bien d'un code source en HDL. Le principal avantage de cette classe d'IP est que grâce à cette description de haut

niveau, la flexibilité et la portablité du noyau IP est assurée.

Trois options sont à la disposition de concepteur SoC :

- utilisation des IP cores commerciaux
- utilisation IP cores open source libre
- concevoir personnellement l'IP core selon l'application

Dans notre réalisation sur SoC, le dévloppement s'est effectué sous l'environnement EDK (Embedded Development Kit), qui d'après le principe de Co-design permet d'une part la configuration de la plateforme matériel et d'autres part le développement software de l'application.

5.1.2 L'environnement de développement EDK

Xilinx Embedded Development Kit (EDK) est un logiciel de développement compatible avec de nombreuses plateformes vendues par Xilinx. Cet environnement de développement permet de configurer la plateforme matériel en sélectionnant les composants (IP cores) appropriés puis en compilant et en débuggant les programmes que l'on veut y éxecuter. Il contient :

Un environnement de développement IDE : appelé Xilinx Platform Studio (XPS) qui est une interface graphique d'EDK qui permet de définir le matériel et le logiciel qui seront implémentés sur le circuit FPGA (Fig.5.1) ;

Une plateforme SDK : qui peut être utilisée pour développer les applications softwares en langages C ou C++.

5.1.3 Le flux de conception sous EDK

Le flot de conception d'un projet sous l'environnement EDK est représenté dans la figure (Fig.5.2) suivante :

L'aspect co-design du flot de conception sous EDK nous permet de développer la plateforme matérielle en parallèle avec la plateforme logicielle.

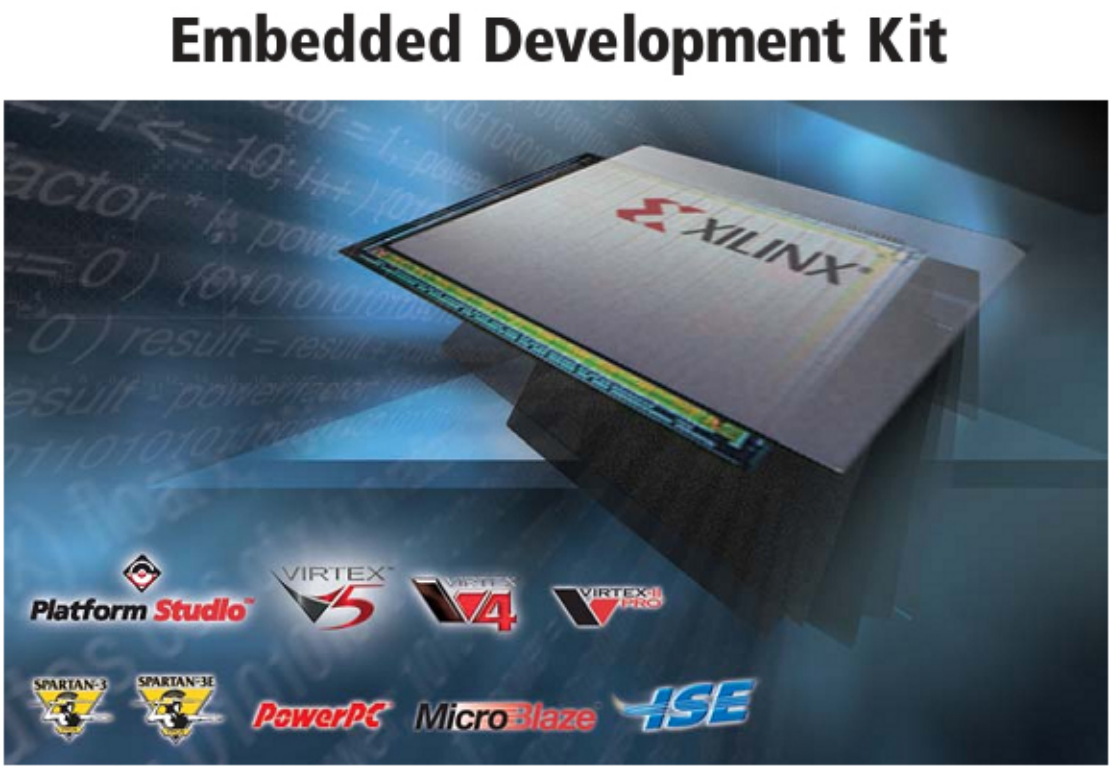


FIG. 5.1 – La plate forme EDK

– Dans le flux de conception de la partie Hardware, on choisi un processeur IP et les différents périphériques qui lui seront connectés puis on exécute l'outil PlatGen de EDK qui les configure en se basant sur les caractéristiques spécifiées dans le fichier MHS (Microprocessor Hardware Specifications) qui définit les composantes matérielles du système. Une fois la plateforme matérielle spécifiée, une Netlist [1] est synthétisée à partir des codes sources des IP cores propriétaires fournis par l'environnement ISE [2]) en langage VHDL. La configuration de la carte et les connexions des différents périphériques avec le circuit FPGA spécifiés dans le fichier de contrainte UCF (User Constraints File) qui est utilisé pour définir les pins du circuit FPGA qui sont utilisée dans la conception, qui sera ensuite lié avec la Netlist pour former un fichier .bit qui est le fichier de configuration de toute la plateforme matérielle d'un projet sous EDK ;
– Dans le flux de conception logiciel, le programme de l'application doit être écris soit en langage assembleur soit en langage C ou C++. Le code source obtenu doit être

[1] Netlist est une forme synthétisée du matériel qui contient les données logiques de la conception et des contraintes
[2] ISE est un produit de Xilinx nécessaire pour implémenter une conception sur le circuit FPGA

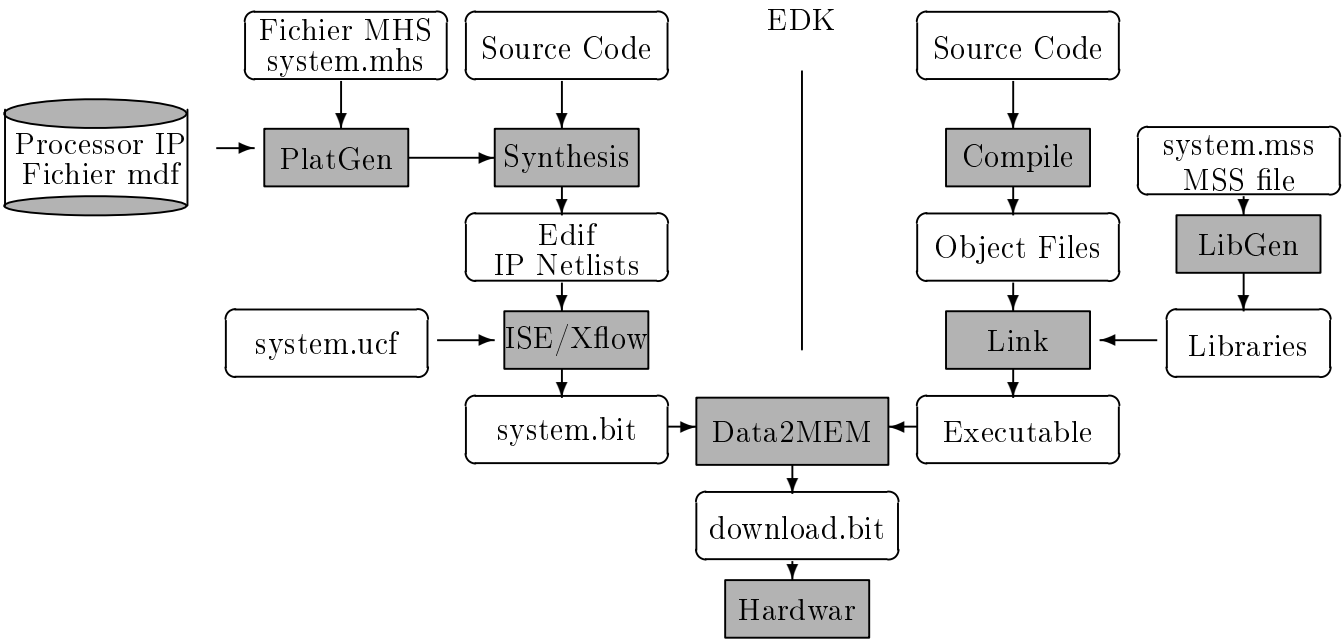


FIG. 5.2 – Flux de conception sous EDK

compilé pour avoir des fichiers en code objet qui seront liés avec des librairies spécifiques. Ces dernières sont générées lors de l'exécution de l'outil PlatGen de EDK pour les spécifications du software qui sont données dans le fichier MSS (Microprocessor Software Specifications) utilisé comme entrée pour l'outil de génération de librairies LibGen, il contient des directives pour les librairies, les pilotes et les systèmes d'exploitation. On obtient à la fin du flux software, un fichier exécutable . Le fichier .bit obtenu à la fin du flux matériel et le fichier exécutable de l'application obtenu à la fin du flux software sont liés par l'outil DATA2MEM pour former un seul fichier download.bit qui configure tout le système en entier et qui sera chargé sur le circuit FPGA [17].

5.2 Architecture matérielle

L'approche sur laquelle notre réalisation sera baser est l'utilisation d'un circuit programmable du type FPGA avec un processeur soft (MicroBlaze).

La taille réduite d'un circuit FPGA exige son implémentation sur un support matériel doté de périphériques et de pins. Le support matériel de notre implémentation est la carte SPARTAN 3 qui inclut le circuit FPGA ainsi que d'autres périphériques (switches, leds,

afficheurs sept segments, mémoire flash,connecteur RS232 ...etc.).

Le but essentiel de notre travail est d'implémenter sur SoC d'un routeur Internet effectuant des recherches de routes à haute vitesse. Pour ce faire, on doit configurer l'architecture du processeur virtuel à savoir ses bus de données et d'instructions, ses ports d'entrée/sortie, ainsi que ses périphériques on-chip et off-chip d'une manière appropriée pour accueillir l'application software contenant l'algorithme de recherche IP. Ensuite, on doit la compiler pour pouvoir la charger sur la mémoire du processeur (BRAM ou SRAM).

Dans cette réalisation, le routeur doit être en mesure d'intéragir avec le PC, qui sera connecter à la carte Spartan 3 renfermant le circuit FPGA où le processeur virtuel MicroBlaze sera virtuellement intégré. Pour que notre routeur soit en mesure d'intérpreter les datagrammes IP et d'en identifier les champs correspondant y compris l'adresse IP de destination, on doit intégrer la pile protocolaire TCP/IP dans notre systeme (spartan 3E annexe C). Pour ce faire, l'environnement de développement EDK nous fournit une librairie open source contenant la pile protocolaire TCP/IP, light-weight Internet Protocol (lwIP). Il s'agite d'une implementation TCP/IP pour des systèmes embarqués qui ne necessite pas de système d'exploitation.

Afin de pouvoir intégrer la pile lwIP, il est nécessaire de préparer l'environnement matériel adéquat, à savoir une intérface Etehrnet MAC, un timer programmable et un controlleur d'interruption, on devra donc passer par deux étapes :

1. L'adaptation de la plateforme matérielle : le processeur soft (Micro Blaze), interface Ethernet Mac, le timer et le contrôleur d'interruption connectant ces deux IP cores au Micro Blaze ;
2. La configuration de " lwip-v3-00-a " afin de l'introduire dans l'application software.

Le schéma bloc de notre plateforme matérielle après adaptation est illustré ci après ; figure ((FIG.5.3) ;

Les différents périphériques inclus sont comme suit :

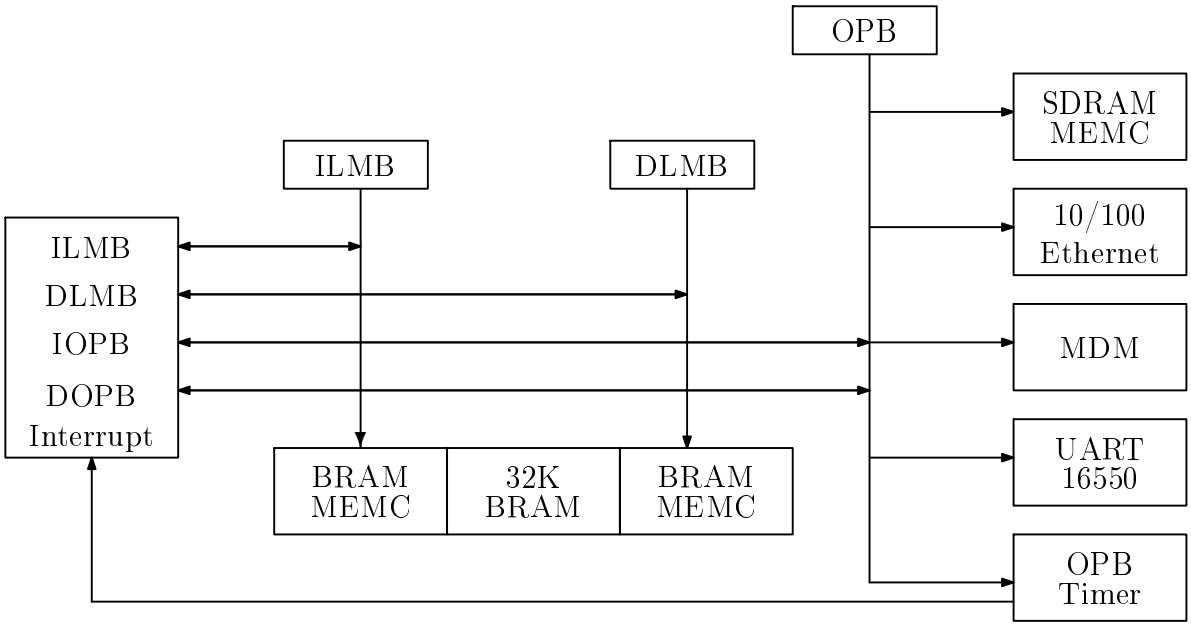


FIG. 5.3 – Plateforme matérielle du routeur

5.2.1 Le soft processeur (Microblaze)

Le coeur du système, un processeur Micro Blaze ; il s'agit d'un processeur soft 32bits à jeu d'instruction réduit (RISC), ayant une architecture HARVARD. Avec une telle architecture, les instructions et les données stockées dans la mémoire on chip du processeur (BRAM) sont accéssible simultanément en un cycle d'horloge via le Bus LMB. Développé par Xilinx, le processeur soft MicroBlaze est disponible sous l'environnement EDK.

MicroBlaze est caractérisé par :

- 32 registres banalisés à 32 bits ;
- Un mot d'instruction à 32 bits avec trois opérandes et deux modes d'adressage ;
- Des bus de données et d'instructions séparés à 32 bits conforment aux spécifications des bus OPB d'IBM ;
- Des bus de données et d'instructions séparés de 32 bits connectant le processeur au bloques mémoire internes via le bus LMB ;
- Des bus d'adresse à 32 bits ;
- Un pipeline qui assure l'exécution d'une instruction à chaque cycle d'horloge ;
- Un cache d'instructions et de données qui permet d'augmenter la bonde passante entre le processeur et la mémoire ;

– Un debugger logique du matériel ;

– Un support FSL (Fast Simplex link) qui permet une seule fonction à la fois soit la lecture soit l'écriture(FIG.5.4).

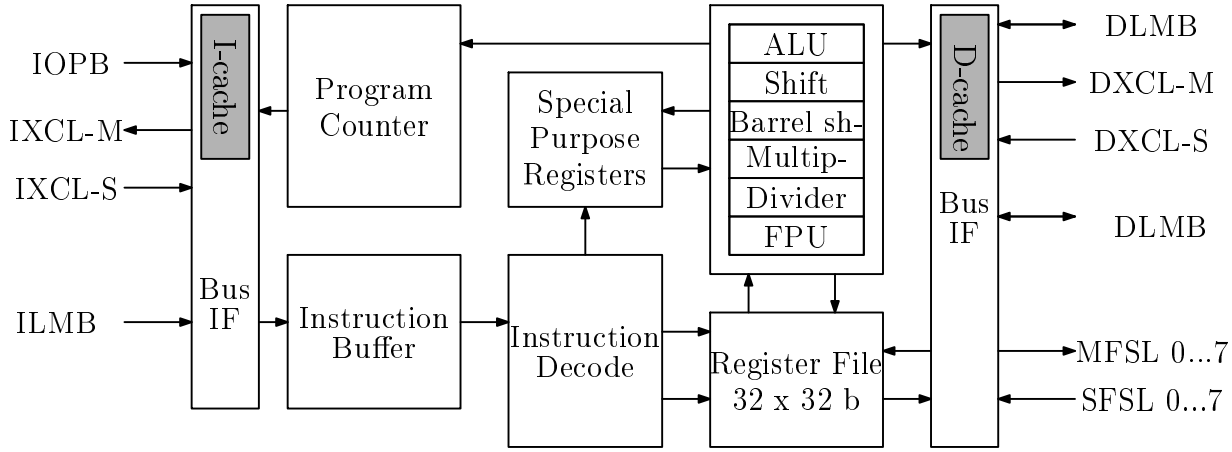


FIG. 5.4 – Architecture du processeur Microblaze

5.2.2 Architecture des bus

Le noyau MicroBlaze communique avec les modules intégrés sur FPGA via les deux Bus[17] :

– Local Memory Bus : le LMB est un Bus de 32 bits single master, qui permet l'accès à la BRAM. La structure Harvard de MicrBlaze divise ce bus en DLMB (Data LMB) et en ILMB (Instrcution LMB) afin de séparer le chemin de données du chemin d'instructions. Le MicroBlaze communique avec les controleurs de LMB sur chaque partie du bus, DLMB et ILMB, permettant l'accès aux instructions et aux données en un cycle d'horloge. Le contrôleur LMB est capable de supporter des mémoires de tailles allant de 8 à 64 kB selon le nombre de bloc BRAM utilisée[17].

– On-chip Peripheral Bus : l'OPB est un bus multiple master utilisé pour les communications avec les périphériques tel que les contrôleurs de mémoires externes (SRAM), Ethernet MAC et les timers. L'OPB est aussi divisé en DOPB (pour les données allant de ou vers les périphériques on chip), et en IOPB (pour les instructions). Ce bus à une largeur de 32 bits par défaut, mais les signaux "byte enable" peuvent être utilisées pour limiter la taille des données transférées à 8,16 et 32 bits[17].

5.2.3 Les mémoires

Dans notre conception, on a utilisé deux types de mémoire, la mémoire on-chip du processeur (BRAM) et la mémoire éxterne (SRAM) dont la capacité est plus importante.

- La Block RAM , est une mémoire on chip dont la taille est configurable lors de la création du projet sous EDK. Pour notre réalisation 32Ko de BRAM conviennent.
- La SRAM est une mémoire off-chip de capacité 1MB, cette éspace suffisant pour acueillir les programmes code sources et bibliothéques de l'algorithme de recherche ainsi que ceux du traitment de la couche IP.

5.2.4 Les périphériques

Les IP cores inclus dans notre conception matérielle sont :

- XPS-Ethernetlite qui sert d'interface fast Ethernet Mac 10/100 Mbps ;
- XPS-Timer pour synchroniser les interruptions et les fonctions de traitement IP périodiquement ;
- XPS-intc, le controleur d'interruption ;
- XPS-uartlite, c'est l'interface RS232 permettant au systéme d'interagir avec le PC en envoyant le résultat de la recherche du next hop ;
- XMD module de debugage hors puce permettant de connecter le cable de chargement (JTAG) avec la carte spartan3(annex C) ;

5.2.5 Mise au point de la plateforme matérielle

La plateforme matérielle est crée en utilisant l'outil de génération de plateforme PlatGen. Cet outil a pour entrée le fichier MHS qui définit l'architecture du système, les périphériques, les processeurs enfouis, les connections du système, la configuration des adresses de chaque périphérique dans le système et les options de configuration de chaque périphérique. PlatGen crée donc une Netlist qui décrit le matériel utilisé en plusieurs formes (NGC, EDIF). Après l'exécution de cet outil, les outils d'implémentation sur FPGA (ISE) s'exécutent pour compléter l'implémentation du matériel. A la fin du flux ISE, un fichier

de configuration (Bitstream) est généré pour configurer le circuit FPGA. Le Bitstream contient l'information pour l'initialisation de la mémoire BRAM sur le circuit FPGA.

Plusieurs périphériques utilisés souvent, sont fournis par Xilinx avec l'outil EDK, cependant on peut définir nos propres périphériques et les inclure dans le fichier MHS.

Il est possible d'éditer son propre fichier MHS, et d'y inclure d'autre périphériques mis à part ceux définis sur EDK, mais cela implique l'écriture (sur ISE) de codes VHDL descriptifs du matériel ajouté.

Le flux de configuration du matériel est donné dans la figure suivante (FIG.5.5) :

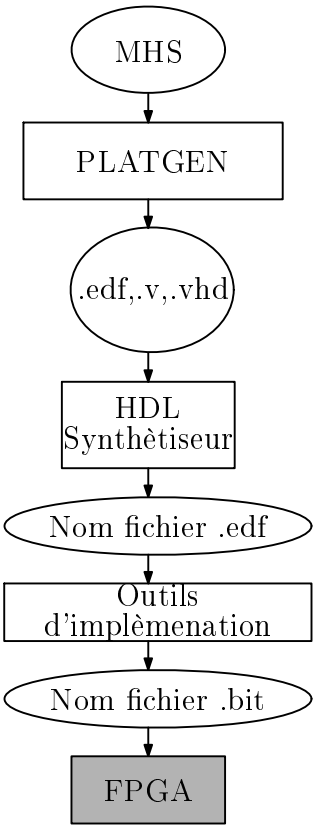


FIG. 5.5 – l'organigramme de la conception matérielle

5.3 Application software

Le deuxième élément du co-design est l'application software. Elle est constitué de deux parties : une partie liée à la recherche de route développé à partir de l'algorithme DTBM et celle relatif au traitement des paquets IP en utilisant la librairie lwIP.

Dans le flux de conception logicielle, le programme de l'application est écrit en langage C. Le code source obtenu doit être compilé pour avoir des fichiers en code objet qui seront liés avec les librairies spécifiques. Ces dernières sont générées lors de l'exécution de l'outil PlatGen de EDK pour les spécifications du software qui sont données dans le fichier MSS. On obtient à la fin du flux software, un fichier exécutable (FIG.5.8).

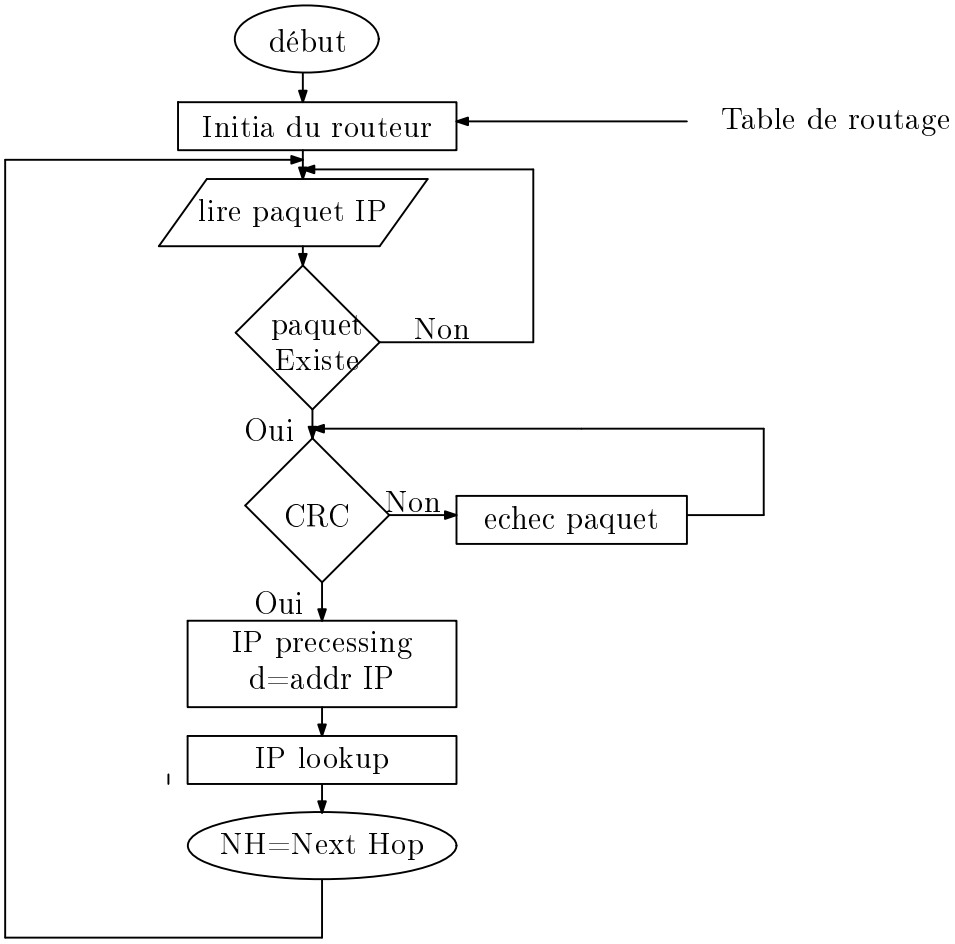


FIG. 5.6 – l'organigramme de fonctionnement du routeur

En recevant un paquet, le routeur extrait l'adresse IP du destinataire et calcul, en exécutant la recherche IP, le next hop correspondant et envoie finalement ce résultat sur l'HyperTerminal du PC [19].

5.3.1 La recherche de route

La stratégie générale utilisée dans cette partie consiste à trouver le plus long préfixe correspondant à l'adresse IP de destination (cette adresse est fournit après traitement IP)

en éxecutant l'algorithme Dynamic Tree bitmap, puis envoyer le next hop associé comme résultat sur le PC. Pour ce faire, cette partie a été structurée en module bibliothéque.h et bibliothéque.c avec un programme principal (lookups.C).

- La bibliothéque.h contient les déclarations des differentes foncions ainsi que les structures relatifs à l'algorithme Dynamic Tree bitmap à savoir le préfixe qui contient les champs : valeur, taille et next hop associé. La deuxième constitue le noeud DTBM avec les champs : l'IBM, les pointeurs, les next hops.
- La bibliothéque.c renférme les codes sources des fonctions nécessaire à l'éxection de l'algorithme DTBM.
- Le programme principal où est incluse la bibliothéque.h sous forme d'entête. lookups.c fait appel au fonction déclarées dans les header afin d'executer une recherche IP.

5.3.2 Le traitement IP sous lwIP

Dans notre systéme on utilisera la pile lwIP standalone (sans système d'exploitation). La librairie EDK qu'on utilisera est lwip-v3-00. lwIP fournit un support pour les protocoles :

- Internet Protocol (IP) ;
- Internet Control Message Protocol (ICMP) ;
- User Datagram Protocol (UDP) ;
- TCP (Transmission Control Protocol (TCP) ;
- Address Resolution Protocol (ARP) ;
- Dynamic Host Configuration Protocol (DHCP).

5.3.2.1 Définition lwIP

lwIP (lightweight IP) est une pile TCP/IP open source largement utilisée pour le développement de systèmes embarqués, elle est maintenant développée et maintenue par un réseau de développeurs répartis dans le monde entier. L'un des objectifs de l'implémentation de lwIP est de réduire l'utilisation des ressources tout en ayant un module TCP

le plus complet possible. Cela rend l'utilisation de lwIP parfaitement adapté dans des systèmes embarqués avec quelques dizaines de kilooctets de RAM disponible et environ de l'espace pour 40 kilooctets de code en ROM.

Le block diagramme de la conception hardware et software de notre système peut se présenter comme suit (FIG.5.7) ;

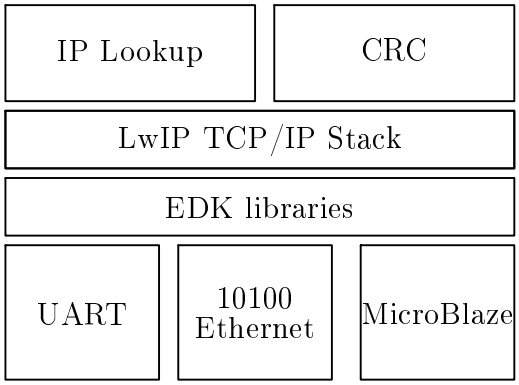


FIG. 5.7 – Le block diagramme de la conception de notre systéme

Les application software construite sous lwIP doivent suivre une certaines structure. Le signal d'horloge génégé par le timer doit etre géré par la pile lwIP pour permettre de synchroniser l'execution des fonctions avec un certain nombre de cycles d'horloges.

RAW API

Le mode Raw API permet à l'application d'avoir un accès directe à la pile lwIP et vice versa. Par conséquent, il n'est pas nécessaire d'enregistrer les données en mémoire. Le mode Raw API fournit des performances excellentes au prix de la compatibilité avec les autres pile TCP .

5.3.3 Mise en œuvre de la couche lwIP en mode raw API

- Initialisation de toutes les structures de lwIP à l'aide de la fonction : init-lwIP ; - une fois la couche lwIP initialisée, on lui associe la couche Ethernet MAC par un appel à la fonction : xemac-add ; - gestion de la lecture des paquets entrants en générant des interruptions à des intervalles de temps régulier. Pour cela, il est nécessaire d'activer

les interruptions au niveau du processeur et du contrôleur d'interruptions, ainsi que la validation du signal d'horloge émis par le timer. En général, une interruption chaque 250ms ; - une fois l'application initialisée, le programme principal entre dans une boucle infinie, afin de pouvoir attendre la lecture des paquets IP ; - le paquet reçu est lu par l'opération xemacif-input , la prise en charge des paquets entrants est dictée par les signaux d'interruptions périodiques, puis sont passés à la pile lwIP qui gère les opérations nécessaire sur le paquet.

5.3.3.1 L'initialisation de lwIP

LWIP fait appel à des fonctions d'initialisation avant d'être utilisé. Le code source d'initialisation est décrit dans l'annexe A.

5.3.4 Mise au point de la plateforme software

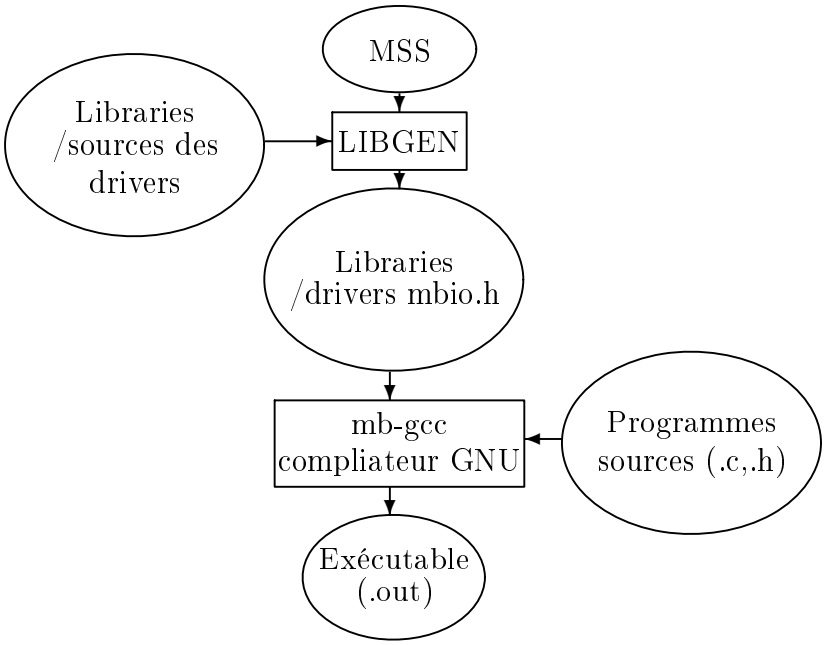


FIG. 5.8 – l'organigramme de la l'application software

La plateforme logicielle est spécifiée par le fichier MSS (Microprocessor Software Specification) qui définit les librairies, les pilotes, les paramètres de personnalisation du processeur, les dispositifs d'entrées/sorties, les routines de traitement d'interruptions et d'autres caractéristiques du software. Le fichier MSS est utilisé comme entrée pour l'outil de généra-

tion de librairies (LibGen). Cet outil nous permet de configurer les librairies et les pilotes avec les adresses des périphériques du processeur enfoui. Les librairies et les pilotes configurés et les programmes sources (d'extension .c ou bien .h) seront compilés grâce à mb-gcc qui est un compilateur GNU adapté à MicroBlaze. Le résultat est un fichier exécutable qui sera chargé dans la BRAM ou la SRAM de MicroBlaze pour être exécuté(FIG.5.8).

la figure (FIG.5.9) suivante represente le schéma bloc du systeme sur SoC.

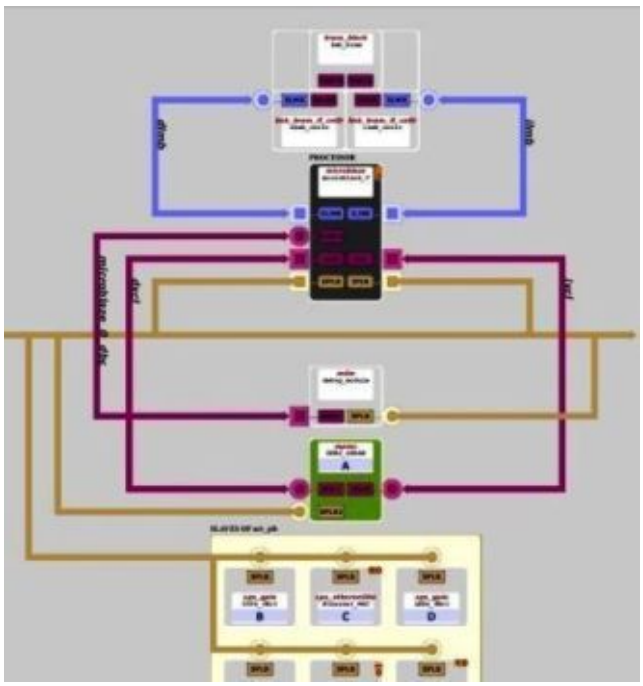


FIG. 5.9 – schéma bloc du système sur SoC

5.4 Conclusion

Dans ce chapitre on a pu découverir les System on Chip permettant l'utilisation d'un seul circuit pour l'intégration entière du co-design hard et soft ainsi que son outil de développement EDK qui prend en charge la totalité de la conception de la plateforme matérielle et logicielle. Ce qui en fait un outil idéal pour le concept de Co-design. L'utilisation d'un processeur virtuel est bien plus avantageux qu'un processeur matériel dont la configuration depuis sa conception est figé. Les processeurs virtuels sont configurable selon les besoins de l'application. Vu ces propriétés intéressante ; nous avons choisis de concevoir notre système on chip à base d'un soft processeur : MicroBlaze dont les périphériques et

les ports nécessaires pour notre réalisation sont configurés pour notre réalisation. Il a été nécessaire d'intégrer lwIP afin d'implémenter la couche TCP/IP. Ainsi on a pu voir les étapes de développement de notre réalisation sous EDK.

Conclusion et perspectives

Notre projet de fin d'études s'inscrit dans un projet de conception d'un routeur gigabit. Il a aboutit au choix et la mise en œuvre de l'algorithme Dynamic Tree Bitmap et son implémentation.

Le choix de cet algorithme s'est fait après l'étude des diverses approches possibles à savoir les techniques Exact Matching, Longuest Prefix Maching et Range Matching. Le Dynamic Tree Bitmap fait partie des algorithmes de la classe des Longuest Prefix Matching. Cet algorithme se caractérise par un temps de recherche indépendant de la taille de la table de routage contrairement à la majorité des autres algorithmes. Sa performance est essentiellement liée à la longueur du plus long préfixe correspondant à l'adresse de destination.Tenant compte du temps imparti à notre projet, le portage de cet algorithme s'est fait et testé sous forme de module logiciel écrit en langage C.

Le développement de l'implémentation Soc s'est fait sous l'environnement EDK (Xilinx). Ce choix a été adéquat et opportun sachant la flexibilité de cet environnement pour non seulement le déploiement de l'implémentation mais aussi pour l'évolutivité vers l'intégration d'accélérateurs matériels.

On a fait appel à la pile protocolaire lwip pour l'implémentation des fonctionnalités de la couche réseau. Elle est disponible sous forme libre et reste bien adaptée pour les systèmes embarqués. Nous nous sommes restreints évidemment à l'utilisation des deux couches liaison (MAC) et réseau (IP).

La traduction de l'algorithme précédemment cité sous forme matérielle, l'intégration d'adaptateur gigabit et le développement de la forme MPSOC à notre développement pourront aboutir à une architecture d'un routeur plus performante.

Annexe A

Initialisation de la pile protocolaire

```
/*****************************************************************
* Call lwIP Initialization Functions
/*****************************************************************/
sys-init() ;
mem-init() ;
memp-init() ;
pbuf-init() ;
netif-init() ;
tcp-init() ;
*****************************************************************
* Setup our 1 network interface (netif)
*****************************************************************/
netif = netif-add(ipaddr,netmask,gw,XEmacIF-ConfigTable[0] xemacif-init, ip-input) ;
netif-set-default(netif) ;
/*****************************************************************
* IDLE Loop - handle lwIP timer functions and poll EMAC
*****************************************************************/
while (1)
while (waiting-for-timer)
/* poll network interface */
```

```
xemacif-input(default-netif);
/* get current PPC405 timer value */
XTime-GetTime(ml-new);
/* check to see if we reached the terminal count */
if (ml-new >= ml-base)
waiting-for-timer = 0;
ml-base = ml-new + ml-offset;
my-tmr() /* calls the various lwIP timer functions */
/* wait for next terminal count */
waiting-for-timer = 1;
```

Annexe B

La carte de contrôle : La Spartan3 Starter Board

B.1 Présentation générale

La carte de control est fournit selon un kit comprenant la carte les câbles de programmations ainsi qu'un module d'alimentation. Ce kit fait office de plateforme de développement "tout en un" universelle spécialement conçue pour l'apprentissage rapide des techniques de conception numérique. De part la présence de son FPGA très largement dimensionné (près de 200 K portes) et de ses dispositifs de commandes et de visualisation divers (boutons-poussoirs, afficheurs, Leds, port PS2, Port VGA...), cette platine convient tout aussi bien pour la réalisation d'applications de décodage logique très simple comme pour la mise au point de réalisations extrêmement complexes et puissantes. L'ensemble est livré avec un bloc d'alimentation, un câble de programmation "JTAG" permettant de programmer votre application en mémoire Flash non volatile [20].

B.2 Caractéristiques principales du kit

- Base de développement FPGA complète avec câble de programmation "JTAG" livré
- Equipé d'un FPGA Spartan-3 avec 216 Kbits de bloc RAM et horloge interne jusqu'à 500 MHz

- Oscillateur 50 MHz inclus - support pour second oscillateur
- Plate-forme flash 2 Mbits (XCF02S) intégrée à la carte
- Mémoire SRAM (256 Kb x 32) intégrée à la carte
- 3 connecteurs d'extensions inclus sur la carte
- 4 afficheurs 7 segments à Leds
- 9 Leds
- 8 interrupteurs et 4 boutons-poussoirs
- Port série, VGA et PS2
- 3 régulateurs de tension (3,3 V / 2,5 V et 1,2 V)
- Livré avec câble de programmation (prévoir une alimentation de : +5 V)

(Fig.B.1)

Fig. B.1 – la Spartan 3 starter board de DIGILENT

B.3 Composants essentiels de la Spartan 3 Starter board

(FIG.B.2)

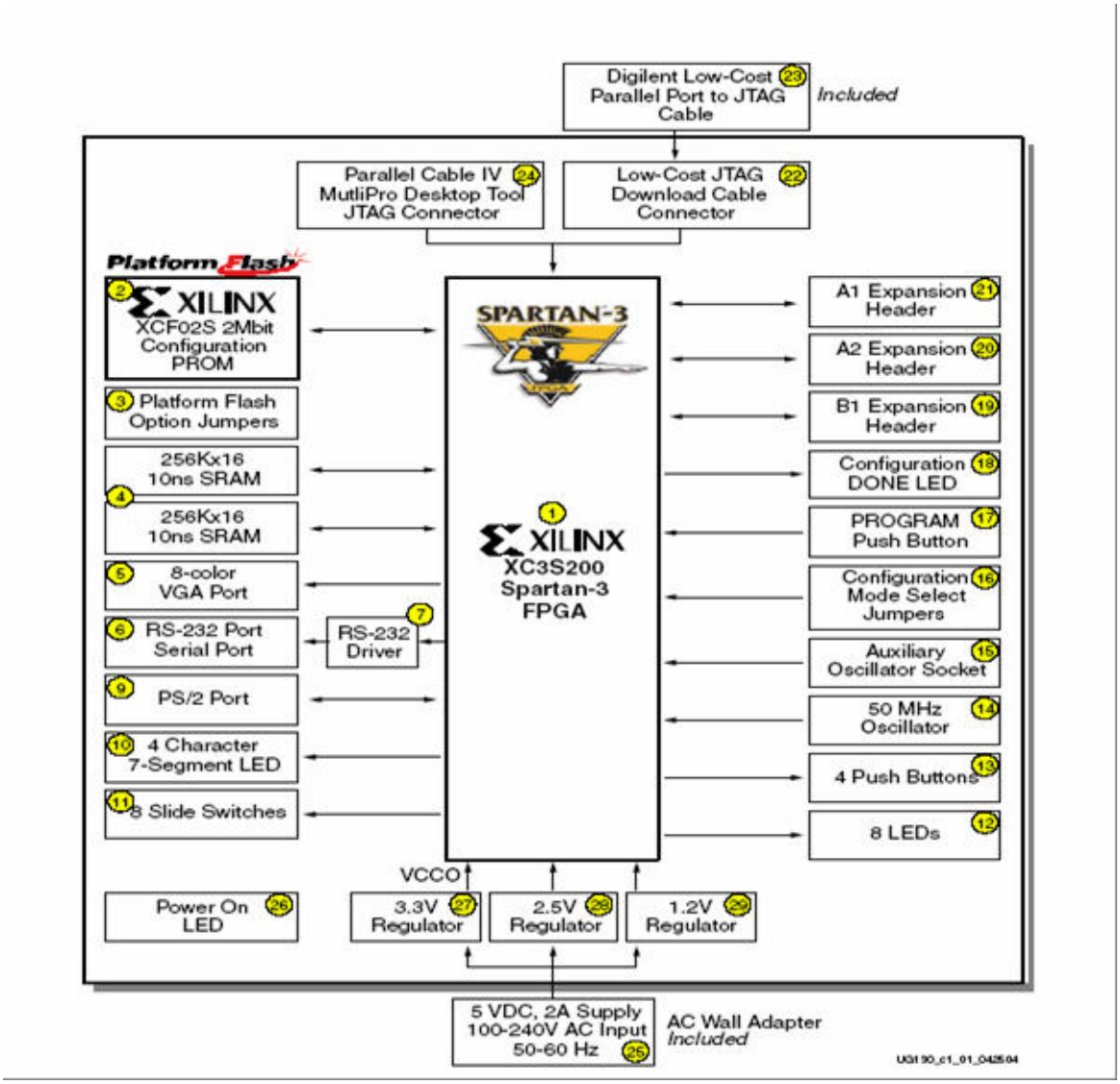


FIG. B.2 – diagramme block de la Spartan 3

1. Circuit programmable de type FPGA (constructeur : Xilinx) avec une capacité d'intégration de 200 000 portes logiques. Le circuit FPGA est fournit suivant le package XC3S200FT256 (1) ;
2. Flash mémoire PROM d'une capacité de 2Mbit (2) :
   - 1Mbit de mémoire non volatile comme support de stockage de données et de programmes disponibles après la configuration du circuit FPGA ;

- Les options du cavalier permettent au FPGA de lire des données PROM ou une configuration du FPGA à partir d'autres sources (3) ;

3. 1M byte de SRAM(4) ;
4. Port à 3 bits pour afficheur VGA à 8 couleurs(5) ;
5. Port série RS-232 à 9 pins (6) ;
6. Pilote pour le RS-232 (7) ;
7. Port PS/2 pour sourie/clavier(8) ;
8. Afficheur sept segments à quatre caractères(9) ;
9. huit swithes(10) ;
10. Huit Leds individuelles(11) ;
11. quatre boutons poussoirs (12) ;
12. Cristal oscillateur à 50 MHZ(13) ;
13. Socket pour une horologe supplémentaire(14) ;
14. Mode de configuration du FPGA sélectionné selon la position de cavaliers(15) ;
15. Bouton poussoir servant à forcer la reconfiguration du FPGA (une fois le bouton appuyé l'ancienne configuration est écrasée)(16).
16. Led indiquant que le FPGA a été correctement configurée(17) ;
17. 3 ports de connection à 40 pins chacun(18)(19)(20) ;
18. Port JTAG (21) pour câble de chargement(22) ;
19. Câble JTAG Digilent pour chargement et debuggage connecté au port parallèle du PC(23) ;
20. Port JTAG de chargement /debuggage compatible avec le câble parallèle de Xilinx(24)
21. Entrée pour adaptateur AC(25)
22. Led indicatrice (s'allume lors de la mise sous tension de la carte)(26)
23. Régulateurs de tension 3.3 V(27) , 2.5 V(28) et 1.2 V(29) .

B.4 Localisation des composants

(FIG.B.3)

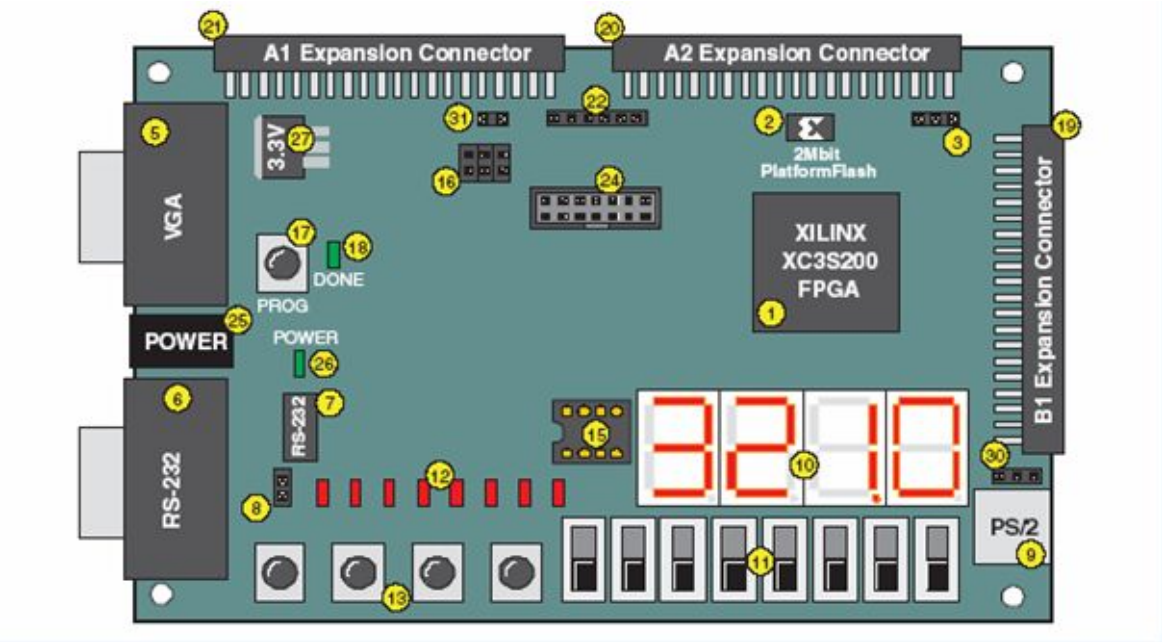


FIG. B.3 – Le Kit Spartan 3 vu d'en haut

(FIG.B.4)

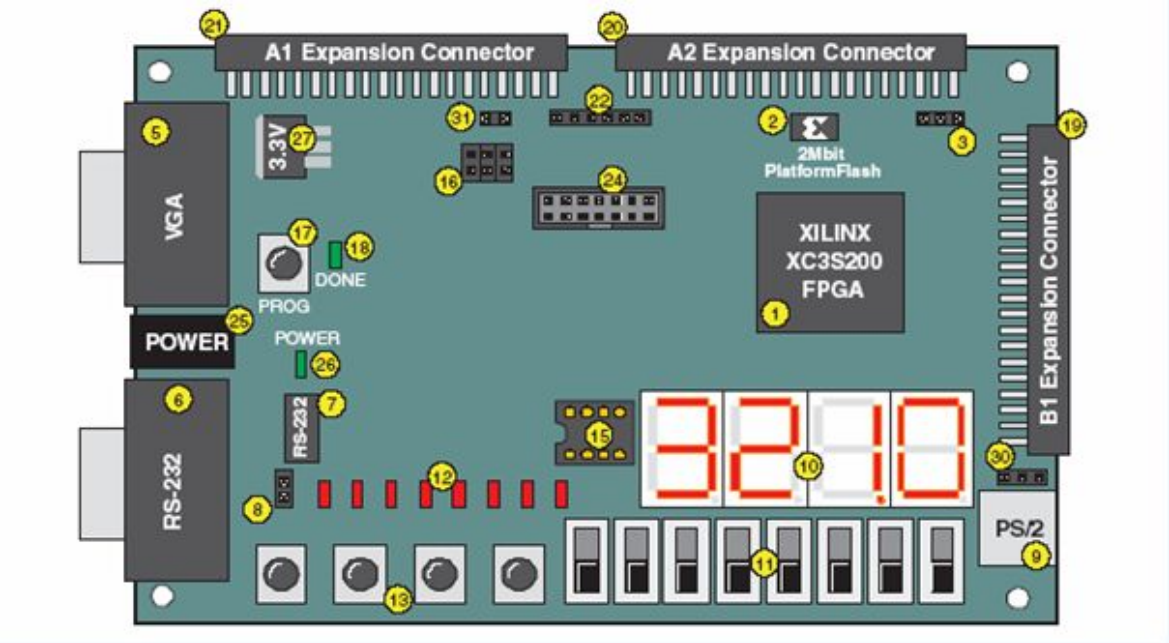


FIG. B.4 – le kit Spartan 3 vu d'en bas

Annexe C

Xilinx-spartan-3E

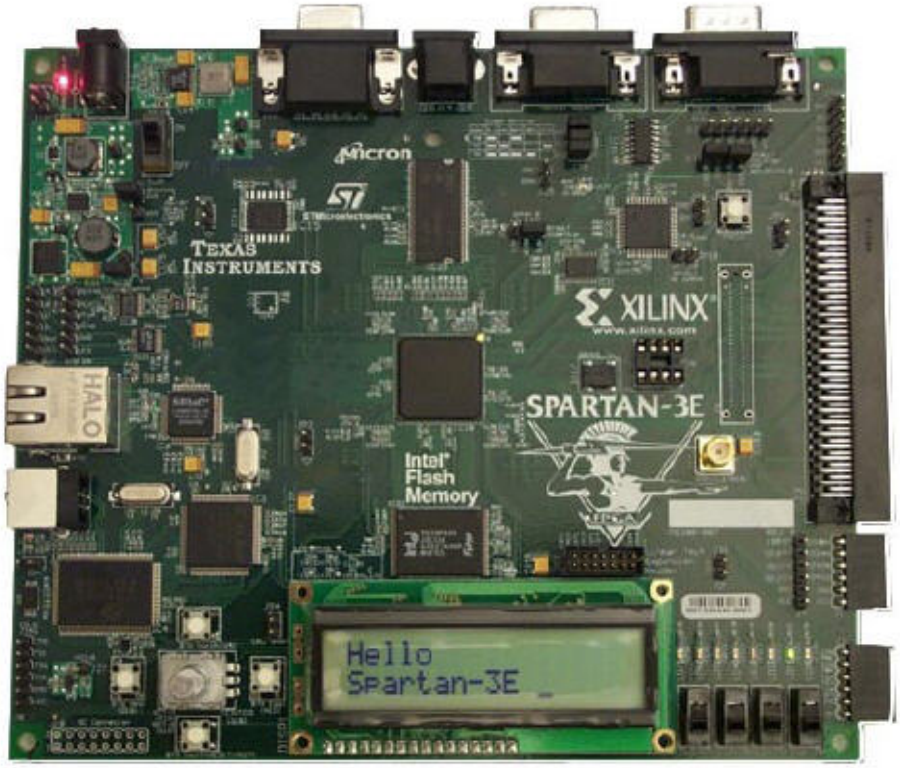


FIG. C.1 – Xilinx-spartan-3an-starter-kit

C.1 Composants essentiels de la Spartan 3E

Les dispositifs principaux du kit de démarreur de Xilinx Spartan-3AN

Dispositifs :

– Xilinx Spartan-3E XC3S500E-4FG320

Memory :

– 128 Mbit

– 8 Mbit SPI Flash

– 64 MB DDR SDRAM

– 4 Mbit platform Flash

Board Interfaces :

– Ethernet 10/100 PHY

– USB programming

– VGA display port

– 9-pin RS-232 serial port

– PS/2-style mouse/keyboard port

– Three expansion connection ports

Additional Features :

– 2 line LCD

– 4 slide switches

– 2 push-buttons

– 8 individual LED outputs

– Variable crystal clock oscillator

– convertisseur de 4-channel D/A,

Target Applications :

– Consumer

– Telecom/Datacom

– Servers

– Storage

– General Prototyping

Bibliographie

[1] E.David D.Taylor. *Models, Algorithms, and Architectures for Scalable Packet Classification.* PhD thesis, School of engineering and applied science, Washington University in St .Louis, july 2004.

[2] B.Petit. *Architecture des réseaux.* 2002.

[3] T.Laurent R.Gerardo. Routage dans les réseaux internet. *Techniques de l'ingénieur*, 2003.

[4] C.Douglas. *TCP/IP Architecture ,Protcoles,applications.* H.Prentice, Dunod,Paris ,2001 ISBN 2100081810, 4 edition, Septembre 2001.

[5] P.Bertrand petit. *Architecture des reseaux.* 2002.

[6] C. Leiserson T. Cormen and R. Rivest. *Introduction to Algorithms.* McGraw-Hill Book Company, 1990.

[7] P.Warkhede S.SURI, G.Varghese. Multiway range trees scalable ip lookup with fast updates. *GLOBCOM*, 2001.

[8] V.Fabienne T.Jean-Marc. Architecture des systèmes de gestion de bases de données. *Techniques de l'ingénieur*, 2918, 2003.

[9] V. Srinivasan and G. Varghese. *Faster IP Lookups using Controlled Prefix Expansion.* SIGMETRICS, 1998.

[10] Srinivasan and G. Varghese. Hardware-based internet protocol prefix lookups. Master's thesis, UniversityinSt.Louis, 1998.

[11] S. Carlsson "M. Degermark, A. Brodnik and S. Pink". *Small Forwarding Tables for Fast RoutingLookups.* inACMSigcomm, 1997.

[12] G.Varghese B.Lampson, V.Srinivasan. Ip lookups using multiway and multicolumn search. *IEEE/ACM Transactionson Networking*, 7(3) :324–334, 1999.

[13] C.-F. Su. *High-Speed Packet Classification Using Segment Tree.* GLOBECOM, 2000.

[14] A.Feldmannand S.Muthukrishnan. Tradeoffs for packet classification. *IEEEInfocom*, March 2000.

[15] R.E.Tarjan. *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics, 1983.

[16] H.Lu S.Sahni. Dynamic tree bitmap for ip lookup update. (FL 32611) :20, 2004.

[17] B.Hons J.David. An fpga coprocessor for real-time bathymetric synthetic aperture sonar. Master's thesis, Electrical and Computer Engineering at the University of Canterbury, Christchurch, New Zealand, February 2007.

[18] R.PIRIOU. Apport de la modèlisation et de la synthèse haut niveau dans la conception d'architecture flexible dèdiè aux turbocodes en blocs. Master's thesis, L'Ecole Nattionale Supèrieure de Tèlècommunications de Bretagne, Janvier 2007.

[19] Embedded Development Kit 10.1. *Platform Studio User GuideXPS; Embedded Development Kit 10.1.* Xilinx, 2008.

[20] Xilinx. *Spartan 3 starter Kit Board reference guide*, octobre 2004.