Titre: Title:	A design methodology for the implementation of embedded vehicle navigation systems
Auteur: Author:	Azizul Islam
Date:	2008
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Islam, A. (2008). A design methodology for the implementation of embedded vehicle navigation systems [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8341/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8341/
Directeurs de recherche: Advisors:	J. M. Pierre Langlois
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

A DESIGN METHODOLOGY FOR THE IMPLEMENTATION OF EMBEDDED VEHICLE NAVIGATION SYSTEMS

AZIZUL ISLAM
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AOÛT 2008

Your file Votre référence
ISBN: 978-0-494-46057-3
Our file Notre référence
ISBN: 978-0-494-46057-3

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

A DESIGN METHODOLOGY FOR THE IMPLEMENTATION OF EMBEDDED VEHICLE NAVIGATION SYSTEM

présenté par : ISLAM Azizul

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

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

M. FERNANDEZ José M., Ph.D., président

M. LANGLOIS J. M. Pierre, Ph.D., membre et directeur de recherche

M. BOYER François-Raymond, Ph.D., membre

To my parents...

Acknowledgement

I am grateful for the support and assistance that I received from my research supervisor Dr. Pierre Langlois, throughout the course of my time as a MScA candidate at the Computer Engineering Department of École Polytechnique de Montréal. I would like to thank him for his valuable insights, guidance, financial support and last but the most for his patience.

I would like to thank also Dr. Aboulmagd Noureldin and his "Navigation and Instrumentation" research group at Royal Military College, Kingston for providing the experimental GPS and TG600 tactical-grade IMU trajectory data and an existing Matlab code of 3D INS mechanization.

I have benefited from the stimulating discussions and the support given in many ways by several members of the Computer Engineering Department. I extend my gratitude to Dr. Mathieu Briere, Mr. Jérôme Collin, graduate students Bruno Girodias, Nasreddine Hireche, Stephane Tchoulack, Anka Stoykova, Gérard Bouyela and many others that I am not mentioning.

I would like to thank my mentor and teacher Ahmad El-Behery and my friend Mustafa Momen for helping me get through the difficult times, for all the emotional support and caring they provided. I wish to thank my relatives for providing their support. My brother, cousins, aunts and uncles were particularly supportive. Lastly, and most importantly, I wish to thank my parents. This paper would not have been possible without them in any ways and no words can express my gratitude towards them.

Abstract

Over the years, due to the increasing road density and intensive road traffic, the need for automobile navigation has increased not just for providing location awareness but also for enhancing vehicular control, safety and overall performance. The declining cost of Global Positioning System (GPS) receivers has rendered them attractive for automobile navigation applications. GPS provides position and velocity information to automobile users. As a result, most of the present civilian automobile navigation devices are based on GPS technology. However, in the event of GPS signal loss, blockage by foliage, concrete overpasses, dense urban developments viz. tall buildings or tunnels and attenuation, these devices fail to perform accurately. An alternative to GPS that can be used in automobile navigation is an Inertial Navigation System (INS). INS is a self-contained system that is not affected by external disturbances. It comprises inertial sensors such as three gyroscopes and three accelerometers. Although low-grade, low-cost INS performance deteriorates in the long run as they suffer from accumulated errors, they can provide adequate navigational solution for short periods of time. An integrated GPS/INS system therefore has the potential to provide better positional information over short and long intervals.

The main objective of this research was to implement a real-time navigation system solution on a low cost embedded platform so that it can be used as a design framework and reference for similar embedded applications. An integrated GPS/INS system with closed loop decentralized Kalman filtering technique is designed using trajectory data from low-cost GPS, accelerometer and gyroscope sensors. A data pre-processing technique based on a wavelet de-noising algorithm is implemented. It uses up to five levels of de-composition and reconstruction with non-linear thresholding on each level. The design is described in software which consists of an embedded microprocessor namely MicroBlaze that manages the control process and executes the algorithm.

In order to develop an efficient implementation, floating-point computations are carried out using the floating point unit (FPU) of MicroBlaze soft core processor. The

system is implemented on a Xilinx Spartan-3 Field Programmable Gate Array (FPGA) containing 200 thousands gates clocked by an onboard oscillator operating at 50 MHz, with an external asynchronous SRAM memory of 1 MiB. The system also includes the IBM CoreConnect On-Chip Peripheral Bus (OPB). As such the final solution for vehicle navigation system is expected to have features like low power consumption, light weight, real-time processing capability and small chip area. From a development point of view, the combination of the standard C programming language and a soft processor running on an FPGA gives the user a powerful yet flexible platform for any application prototyping.

Results show that a purely software implementation of the decentralized closed loop Kalman filter algorithm embedded platform that uses single precision floating point numbers can produce acceptable results relative to those obtained from a desktop PC platform that uses double precision floating point numbers. At first, the Kalman filter code is executed from a 1 MiB external SRAM supported by 8KiB of data cache and 4KiB of instruction cache. Then, the same code is run from high speed 64KiB on-chip Block RAM. In the two memory configurations, the maximum sampling frequencies at which the code can be executed are 80 Hz (period of 12.5 ms) and 119 Hz (period of 8.4 ms) respectively, while accelerometer and gyroscope sensors provide data at 75 Hz. The same two memory configurations are employed in executing a wavelet de-noising algorithm with 5 levels of de-composition, reconstruction and non linear thresholding on each level. Accelerometer and gyroscope raw data are processed in real-time using non-overlapping windows of 75 samples. The execution latencies in the two cases are found to be 5.47 ms and 1.96 ms respectively. Additionally, from the post synthesis timing analyses, the critical frequencies for the two hardware configurations were 63.3 MHz and 83.2 MHz, an enhancement of 26% and 66% respectively. Since the system operates at 50 MHz, there is thus an interesting processing margin available for further algorithmic enhancements.

Thus, by employing the combination of a low cost embedded platform, a flexible development approach and a real-time solution, the implementation shown in this thesis

demonstrates that synthesizing a completely functional low-cost, outage-resilient, real-time navigation solution for automotive applications is feasible.

Keywords: FPGA, MicroBlaze, INS, mechanization, wavelet de-noising, automobile navigation, Kalman filtering.

Résumé

Au fil des années, en raison de l'augmentation de la densité routière et l'intensité de la circulation, un système de navigation automobile devient nécessaire. Ce système doit fournir non seulement l'emplacement du véhicule mais, surtout, augmentera le contrôle, la sécurité et la performance globale de l'automobile. La baisse du coût des récepteurs de Géo-Positionnement par Satellite (GPS) a vulgarisé leur utilisation dans la navigation automobile. Le système GPS fournit les données de positionnement ainsi que l'information qui concerne la vitesse aux conducteurs. De ce fait, la plupart des dispositifs de navigation des automobiles civiles sont actuellement basés sur la technologie GPS. Cependant, en cas de perte du signal GPS par blocage par feuillage, passages en béton, dense agglomération urbaine, grands immeubles, tunnels et dans le cas d'atténuation, ces dispositifs ne parviennent pas à fonctionner avec précision. Une solution alternative au GPS, qui peut être utilisée dans la navigation automobile, est le système de navigation inertielle (INS). L'INS est un système autonome qui n'est pas affecté par des perturbations externes. Il comprend des capteurs inertiels comme trois gyroscopes et trois accéléromètres. Le coût des INS peut être faible mais leur performance se détériore à long terme car ils souffrent des erreurs accumulées. Cependant, il peut fournir des solutions précises sur de courts intervalles de temps. Un système intégré de GPS/INS à faible coût a donc le potentiel de fournir de meilleures informations de position pendant des intervalles courts et longs.

L'objectif principal de cette recherche était de mettre en place une solution d'un système de navigation véhiculaire temps réel sur une plateforme embarquée à faible coût. Ceci avait pour but de pouvoir l'utiliser comme un cadre de conception, et comme référence pour d'autres applications embarquées similaires. Pour améliorer la solution de navigation même en cas d'arrêt de fonctionnement du GPS, les données du système GPS/INS ont été fusionnées par la technique de la boucle fermée du filtrage de Kalman décentralisé en utilisant 15 équations d'états d'erreurs d'INS. En raison de l'utilisation d'accéléromètre à faible coût, ainsi que des capteurs gyroscopiques de données, une technique de prétraitement nommée algorithme de débruitage par ondelettes a été

adoptée. L'algorithme a un maximum de 5 niveaux de décomposition, de reconstruction, ainsi que du seuillage non linéaire à chaque niveau. La conception est décrite par un logiciel qui comprend un microprocesseur embarqué. L'implémentation est effectuée à l'aide d'un cœur du processeur MicroBlaze qui gère le processus de contrôle et exécute l'algorithme.

Afin de développer une implémentation efficace, des calculs en virgule flottante sont effectués en utilisant l'unité de virgule flottante (FPU) du cœur du processeur Microblaze. Le système est implémenté sur carte FPGA Spartan-3 de Xilinx. Elle contient 200 mille portes logiques cadencées par un oscillateur à 50 MHz, avec une mémoire externe asynchrone SRAM de 1 Mio. Le système comprend également un bus périphérique sur puce (OPB). À ce titre, la solution finale du système de navigation automobile devrait avoir des caractéristiques telles qu'une faible consommation de puissance, un poids léger, une capacité de traitement en temps réel ainsi qu'un petit espace occupé sur puce. D'un point de vue développement, l'utilisation du langage C et d'un cœur de processeur fonctionnant sur FPGA donne à l'utilisateur une plateforme flexible pour tout prototypage d'applications.

Les simulations montrent qu'une implémentation purement logicielle de l'algorithme de la boucle fermée du filtrage de Kalman décentralisé sur une plateforme embarquée qui utilise les nombres virgule-flottante à simple précision, peut produire des résultats acceptables. Ceci est conforme aux résultats obtenus sur une plateforme d'un ordinateur de bureau qui utilise les nombres virgule-flottante à double précision. Dans un premier temps, le code du filtrage de Kalman est exécuté à partir d'une mémoire externe SRAM de 1 Mio, soutenue par une mémoire cache de données de 8Kio et une cache d'instructions de 4 Kio. Puis, le même code est lancé à partir du bloc RAM sur puce, à grande vitesse, de 64 Kio. Dans les deux configurations mémoire, les fréquences d'échantillonnage maximales pour lesquelles le code peut être exécuté sont de 80 Hz (période de 12,5 ms) et 119 Hz (période de 8,4 ms), respectivement, tandis que les capteurs fournissent les données à 75 Hz. Les même deux configurations de mémoire sont employées dans l'exécution de l'algorithme de débruitage par ondelettes avec 5

niveaux de décomposition, de reconstruction et seuillage non linéaire à chaque niveau. Sur l'accéléromètre et le gyro, les données brutes sont fournies en temps réel en utilisant un mode de fenêtre de non-chevauchement, avec une longueur de fenêtre de 75 échantillons. Les latences d'exécution dans les deux cas sont 5,47 ms et 1,96 ms pour les deux configurations de mémoire précédemment citées, respectivement. En outre, l'analyse temporelle de l'après synthèse des deux configurations matérielles, reporte des apports de 26% et 66% respectivement. Puisque le système fonctionne à 50 MHz, il y a ainsi une marge de manœuvre disponible intéressante pour des perfectionnements algorithmiques.

Ainsi, en utilisant la combinaison d'une plate-forme peu coûteuse, une approche flexible de développement et une solution en temps réel, l'exécution montrée dans ce mémoire démontre que la synthèse d'une solution finale de navigation véhiculaire fonctionnant en temps réel, complètement fonctionnelle, panne-résiliente, peu coûteuse est faisable.

Mots-clés: FPGA, Microblaze, INS, mécanisation, débruitage par ondelettes, navigation automobile, filtrage de Kalman.

Condensé en français

Les systèmes de navigation ont toujours été essentiels pour les véhicules aériens et marins. Au cours des dernières années, on a observé une augmentation importante de la densité routière et de l'intensité de la circulation. Les besoins en systèmes de navigation pour les automobiles ont donc augmenté considérablement pour la localisation, la sécurité, le contrôle et la performance. La disponibilité d'une technologie appropriée doit permette d'offrir un produit à prix modéré comme système de navigation automobile [1]. Ainsi, cette composante deviendra une partie intégrale de l'industrie automobile et un choix attrayant à prix raisonnable pour le consommateur.

Avec l'avancement de la technologie, la navigation automobile est devenue une préoccupation de plus en plus pressante. En même temps, la baisse du coût des récepteurs de géo-positionnement par satellite (GPS) a généralisé leur utilisation dans la navigation automobile. Particulièrement durant la dernière décennie, le prix des receveurs GPS a été rendu plus intéressant pour les consommateurs. Ainsi le GPS est un système de navigation précis pour déterminer la position et la vitesse du véhicule [2]. Aujourd'hui la plupart des automobiles utilisent la technologie GPS. Cependant, dans le cas de perte, de blocage et d'atténuation des signaux GPS, si ceux-ci ne fonctionnent pas correctement, il en résulte alors une panne. Les pannes temporaires des signaux GPS peuvent être causées par les forêts, les gratte-ciel, les agglomérations urbaines, les tunnels etc. et donc nécessitent un système alternatif pour la navigation d'automobile [3].

L'alternative au GPS, qui peut être utilisée dans la navigation automobile, est le système de navigation inertielle (INS). Aujourd'hui presque tous les systèmes INS sont constitués de deux parties : le mesureur inertiel (Inertial Measurement Unit – IMU) et l'ordinateur de navigation. L'IMU comprend des capteurs inertiels, qui sont en général trois gyroscopes et trois accéléromètres montés orthogonalement dans un véhicule. L'IMU détecte et mesure les accélérations éprouvées et les degrés de changement de direction dans le repère inertiel de la terre. L'ordinateur de navigation transforme ces mesures en un point de repère de navigation. De cette manière, l'INS permet de calculer

la position, la vitesse et la direction incrémentale du véhicule. Ainsi, en réunissant ces valeurs incrémentales de position, vitesse et direction avec les valeurs initiales, la solution de navigation (position, vitesse et direction angulaire) peut être déterminée [4].

A l'inverse du GPS qui dépend de l'aide de plusieurs satellites externe, l'INS est un système autosuffisant qui n'est pas affecté par l'environnement. Il possède une excellente performance à court terme. Toutefois, les capteurs accéléromètres et les capteurs gyroscope sont dispendieux, surtout pour les modèles très performants. Les IMUs les moins chers peuvent fournir une solution de navigation provisoire et il s'en suit une accumulation d'erreurs et une performance inférieure [4].

Dans la plupart des cas, le problème du GPS comme système de navigation autonome peut être surmonté pour une courte période par un système intégré GPS et INS. Ainsi, l'idée principale est d'utiliser les deux systèmes (GPS et INS) ensemble. Dans ce système intégré, on utilise normalement la solution fournie par GPS. En même temps, l'INS continue à calculer la position, la vitesse et la direction. En présence d'une panne de GPS, la solution est alors fournie par l'INS. Ceci est la plus simple manière d'intégrer les deux systèmes.

Pour obtenir encore un meilleur résultat quand il y a une panne de GPS, on utilise le filtrage de Kalman sur les données de GPS et d'INS. Le filtrage de Kalman comprend un modèle dynamique des erreurs de GPS et d'INS, un modèle stochastique des erreurs des capteurs et aussi des valeurs initiales des données reliées avec le système GPS et INS. Durant une panne de GPS, le filtrage de Kalman prévoit les erreurs de position, de vitesse et de direction d'INS. En corrigeant ces erreurs, la solution de navigation est grandement améliorée [4].

A l'inverse du GPS, l'INS est exempté du blocage des signaux. Mais quand le GPS tombe en panne, la performance d'INS baisse progressivement s'il contient des capteurs accéléromètres et des capteurs gyroscopes de faible qualité. Plusieurs techniques de débruitage ont été proposées afin d'améliorer les sorties des accéléromètres et gyroscopes dont le débruitage par ondelettes [5].

L'objectif général de cette recherche est de mettre en place une solution d'un système de navigation véhiculaire temps réel sur une plateforme FPGA Spartan-3 utilisant le cœur de processeur MicroBlaze. Les objectifs spécifiques de cette recherche peuvent être résumés comme suit :

Objectif 1: Implémenter les algorithmes introduits dans la section 1.1 de ce mémoire utilisant le cœur de processeur Microblaze de Xilinx. Cette approche est particulièrement avantageuse au chapitre du coût.

Les capteurs GPS fournissent 3 positions et 3 vitesses directement. Par contre, les capteurs IMUs (les capteurs accéléromètres et gyroscopes) fournissent 3 accélérations et 3 vitesses angulaires par rapport au référentiel «Corps de l'automobile» (body frame). Idéalement, le système GPS/INS intégré inclut les 3 étapes suivantes:

a) Débruitage de données brutes qui sont fournies par les capteurs IMUs.

b) Transformation 2D ou 3D («Mécanisation 2D» ou «Mécanisation 3D») qui convertissent les accélérations linéaires et les taux de rotation du référentiel du véhicule a celui de navigation dans le but d'obtenir les informations de position, vitesse et direction.

c) Fusionnement du système GPS/INS par la technique de la boucle fermée et décentralisée du filtrage de Kalman.

Ainsi, le but était d'implémenter un ordinateur de navigation (NCU) qui peut exécuter les trois techniques mentionnées ci-dessus en utilisant les données fournies par des capteurs accéléromètres et gyroscopes. Plus particulièrement, le but était d'implémenter un ordinateur de navigation sur une plate-forme moins coûteuse au lieu d'un microprocesseur de bureau (qui coûte plus cher) qui utilise les nombres virgule-flottante à double précision.

Objectif 2: Fournir une analyse de résultat comparative (c.à.d. une solution de navigation: position et vitesse) d'implémentation de système GPS/INS intégrée entre les plateformes *«Embarquée»* et *«Microprocesseur»*. À cet égard, l'utilisation des ressources de système embarqué montre la simplicité de l'application de navigation d'automobile, de même que son aspect contact en surface nécessaire.

Objectif 3: Fournir une solution de système GPS/INS intégré de navigation en temps réel pour ces types d'application spécifiques.

Objectif 4: Présenter l'implémentation d'un système GPS/INS intégré de telle manière qu'elle puisse être employée comme un cadre générique pour l'implémentation des algorithmes de navigation. Ce cadre intègre la prise en charge de l'algorithme de débruitage de données d'IMU. Ce dernier s'avère efficace malgré sa complexité relativement à l'algorithme Fast Orthogonal Search (FOS) [5]. Cette technique de débruitage spectral présente une haute résolution pour les données d'IMU sur technologie MEMS. Aussi, ce cadre inclut également la technique d'intégration de GPS/INS pour résoudre des pannes de GPS avec une plus longue durée en utilisant la méthode par réseaux de neurones artificiels qui est en phase de recherche et développement [5] [6].

Pour ce projet de recherche, la carte Spartan-3 développée par Digilent inc. a été principalement utilisée. Cette plate-forme de développement inclut un FPGA Spartan-3 de Xilinx qui contient 200 mille portes logiques cadencées par un oscillateur à 50 MHz, avec une mémoire externe asynchrone SRAM de 1 Mio. Elle contient aussi une interface UART (port série), un port JTAG, un port VGA ainsi que plusieurs autres interfaces (port PS/2, boutons, etc.) [22]. Elle peut donc supporter des circuits de complexité variée, ainsi que des processeurs embarqués.

Dans une étape plus avancée de l'implémentation et pour rendre l'exécution du code plus rapide, la carte FPGA Xilinx University Program (XUP) Virtex-II Pro a été utilisée et elle comporte une quantité supérieure de Block RAM. Il s'agit d'une plate-forme haute performance Virtex-II Pro FPGA entourée par un nombre de périphériques. Pour cette recherche, nous avons utilisé spécifiquement le port série, le port JTAG et le générateur d'horloge [23].

L'une des motivations pour utiliser un cœur de processeur logiciel est son faible coût. En effet, un processeur MicroBlaze ne coûte que $0.48 US. Par contre, les micro-processeurs qu'on trouve habituellement dans les postes de travail coûtent plusieurs

centaines de dollar. De plus, la plate-forme FPGA utilisée coûte moins de $2.00 US, ce qui en fait une solution abordable pour les applications de navigation automobile [26].

Le microcontrôleur MicroBlaze est une solution intégrée toute désignée pour l'implémentation d'un contrôleur embarqué. Le cœur de processeur logiciel MicroBlaze embarqué est un processeur 32-bit suivant une architecture Harvard. L'architecture Harvard sépare physiquement la mémoire de données et la mémoire programme. L'accès à chacune des deux mémoires s'effectue via deux bus distincts. Le processeur MicroBlaze suit aussi une architecture RISC optimisée pour implémentation dans les FPGA de Xilinx. Une fois implémenté, le processeur est programmé sur la carte FPGA. Le processeur MicroBlaze est aussi configurable. Il permet aux utilisateurs de choisir uniquement les fonctionnalités dont ils ont besoin [24].

Le «Wizard Base System Builder» (BSB), module de l'outil EDK, a été utilisé pour construire le système matériel fonctionnel dédié pour la carte Spartan-3. Le BSB a été employé pour développer la plateforme matérielle dans le but de l'intégrer au logiciel de navigation développé dans ce mémoire. En remplacement d'un système d'exploitation embarqué tel que XilKernel ou ucLinux, une plateforme autonome a été retenu.

Le nombre de cycles nécessaires a été mesuré en utilisant un compteur watch-dog connecté avec le port de gestion des interruptions du MicroBlaze. Cette approche est plus précise que celle du profilage du code utilisant l'outil GCC/GPROF d'EDK.

Plusieurs recherches ont été conduites en vue de bien comprendre les algorithmes existants reliés au système GPS/INS et les traitements des données fournies par l'IMU. Le chapitre 2 de ce mémoire synthétise l'état de l'art dans ce domaine. L'étude des connaissances a permis de retenir le model de filtrage de Kalman, basé sur une boucle fermée décentralisée. Ce modèle se base sur une équation à 15 états pour modéliser les erreurs d'INS.

Le chapitre 3 présente le travail réalisé qui consistait à calibrer les paramètres de filtrage de Kalman dans le but de valider le modèle pour un cas de simulation de panne de GPS d'une duré de 20 seconds. En parallèle, une implémentation d'algorithmes de

débruitage par ondelettes a été faite pour les données d'IMU. L'implémentation de ces algorithmes a été faite sur la plate-forme MicroBlaze.

Le chapitre 4 présente les résultats de l'implémentation embarquée décrite au chapitre 3. Il débute par l'examen des sorties d'application embarquée (des algorithmes de navigation) en comparaison à ceux exécutés sur une plateforme d'ordinateur de bureau à l'aide de l'outil Matlab. Ensuite, on y trouve une discussion des performances temporelles des applications de navigation embarquée. Enfin, le chapitre résumé l'utilisation de matériel par les applications embarquées dans ce mémoire.

Les figures 4.1 et 4.2 montrent la comparaison des sorties de la «Mécanisation 2D» de position d'une automobile pendant la panne de GPS forcée pendant une période de 20 secondes respectivement entre le modèle de Matlab et celui du MicroBlaze. La figure 4.2 illustre le problème de précision lié à une représentation 32 bits à virgule flottante. Elle démontre que dans un intervalle étendu l'incrément de position de l'automobile à chaque 1/75 seconde n'est pas suffisamment grand pour être ajouté à une grande quantité de latitude et de longitude.

D'ailleurs, dans le calcul de latitude et de longitude, les points de référence sont sur l'équateur (latitude zéro) et sur le méridien de Greenwich (longitude zéro) respectivement. Les données de trajectoire de navigation utilisées ont été rassemblées de l'essai sur route effectué autour de la ville de Kingston, Ontario, Canada [28]. Cet endroit est loin de l'équateur et même plus loin de la ligne de méridien de Greenwich. La figure 3.6 montre que pendant la panne de GPS, le véhicule se dirigeait vers le nord. Ainsi le changement de la position vers la direction est-ouest dans chaque 1/75 seconde a été trop peu comparé à la grande distance de la ligne méridienne. En revanche, la figure 4.1 montre une croissance accumulée des erreurs, et l'effet du calcul d'addition est évident.

Les résultats de l'expérimentation sur la nouvelle plateforme embarquée avec un algorithme simple tel que la mécanisation 2D sont concluants. Ces résultats nous ont permis d'adresser le problème lié à la précision dans le calcul des solutions de navigation. Afin de surmonter ce problème, un point local a été retenu comme référence

dans le but de minimiser la distance de référence et de faciliter l'ajout d'incréments les plus petits possible.

Les figures 4.3 et 4.4 montrent les schémas des sorties de la «Mécanisation 2D», «Mécanisation 3D» et du filtrage de Kalman fonctionnant sur la plateforme MicroBlaze. Il reste évident que la performance des sorties relative au référence GPS de «Mécanisation 3D» excède celle de «Mécanisation 2D» alors que le filtrage de Kalman dépasse celle de la «Mécanisation 3D».

En examinant le schéma PSD de figure 4.11 (débruitage par ondelettes fonctionnant sur le cœur de processeur MicroBlaze), on peut non seulement atténuer des erreurs à court terme existant au delà de 2Hz, mais également rejeter une partie des erreurs à long terme dans le spectre au-dessous d'elle.

Les figures 4.7, 4.8 et 4.12 montrent les sorties de MicroBlaze par rapport aux résultats obtenus d'une modélisation Matlab à haut niveau. Les figures 4.5 et 4.6 illustrent l'implémentation dans des ordinateurs de navigation. Les figures 4.10, 4.11 et 4.14 montrent les résultats d'implémentation de débruitage par ondelettes sur MicroBlaze pour les données d'IMU.

Le temps d'exécution du code sur MicroBlaze pour deux configurations de mémoires séparées a été mesuré (voir tableaux 4.1 et 4.2). L'exécution dans un bloc de RAM s'avère plus rapide que celle sur la mémoire externe de SRAM.

Le cœur de processeur logiciel MicroBlaze a été utilisé avec succès parce qu'il offre la flexibilité d'implémenter arbitrairement des algorithmes en utilisant un langage de programmation de haut niveau tel que le langage C. Ceci permet d'éviter l'utilisation extensive d'autres langages de programmation de bas niveau comme HDL. Par conséquent, l'usage du MicroBlaze réduit significativement la complexité et le temps de développement en comparaison avec d'autres approches.

Le chapitre 5 présente les conclusions de la méthodologie d'implémentation présentée dans ce mémoire. Il fournit quelques recommandations dans le but d'une utilisation comme cadre de conception pour les futures applications embarquées de même nature.

Ce travail a permis de démontrer qu'une implémentation peu coûteuse est possible pour un système embarqué de navigation automobile. Cette implémentation, qui utilise une approche flexible, fournit des sorties en temps réel.

Les travaux décrits dans ce mémoire constituent un cadre structuré pour le développement éventuel d'une solution complète et intégrée d'un produit commercial. Plusieurs équipes de chercheurs travaillent au développement d'algorithmes d'intégration INS/GPS et de modélisation d'erreur de l'INS. La miniaturisation des senseurs INS créera par ailleurs des opportunités grandissantes, permettant de passer de la navigation véhiculaire à la navigation personnelle.

Table of Contents

List of Figures

List of Tables

List of Symbols

ω_x Vehicle platform rotation rate or gyroscope measurement about the X-axis of the *b*-frame.

ω_y Vehicle platform rotation rate or gyroscope measurement about the Y-axis of the *b*-frame.

ω_z Vehicle platform rotation rate or gyroscope measurement about the Z-axis of the *b*-frame.

f_x Vehicle platform acceleration or accelerometer measurement about the X-axis of the *b*-frame./

f_y Vehicle platform acceleration or accelerometer measurement about the Y-axis of the *b*-frame. *x*

f_z Vehicle platform acceleration or accelerometer measurement about the Z-axis of the *b*-frame.

$\vec{f}^b$ Vehicle platform acceleration or accelerometer measurement vector with respect to the *b*-frame.

R_b^l Transformation matrix from body frame to local level frame.

θ Vehicle platform pitch angle.

ϕ Vehicle platform roll angle.

ψ Vehicle platform azimuth angle.

V^e Vehicle platform velocity in East direction.

V^n Vehicle platform velocity in North direction.

V^u Vehicle platform velocity in Up direction.

φ Vehicle platform (geodetic) latitude position component.

τ Eccentric Latitude

θ' Geocentric Latitude

λ Vehicle platform longitude position component.

h Vehicle platform altitude position component.

σ^2 Variance.

β	In`verse of correlation time for Gauss-Markov process.
x	INS error state vector of KF
K_k	Kalman gain matrix of KF
$\hat{x}_k^-$	Predicted or a priori estimate of error state vector of KF
$\hat{x}_k$	Updated or a posteriori estimate of error state vector of KF
H_k	Design matrix of KF
z_k	Measurements vector of KF
$F_{k\text{-}1,k}$	State transition matrix of KF.
G	System noise distribution matrix of KF
Ω_{ib}^b	Skew-symmetric matrix containing vehicle platform rotation measurements.
Ω_{il}^b	Skew-symmetric matrix containing effect of the Earth's rotation and change to the local level frame orientation.
Ω_{ie}^l	Skew-symmetric matrix containing effect of the Earth's rotation.
Ω_{el}^l	Skew-symmetric matrix containing effect of change to the l-frame orientation.
g	Acceleration due to gravity
V^l	Vehicle platform velocity vector expressed in the l-frame.
r^l	Vehicle platform position vector expressed in the l-frame.
R_k	covariance matrix of the measurement noise $u\delta$.
P	Error state covariance matrix.
P_k^-	Predicted or a priori estimate of error state covariance matrix.
P_k	Updated or a posteriori estimate of error state covariance matrix.
Q	System noise covariance matrix.
N	Prime vertical radius of the best fitting Earth ellipsoid (East-West)
M	Corresponding meridian radius of curvature (North-South)
f_s	Sampling frequency
bps	bits per second
deg	degree

hr	hour
m	meter
km/hr	kilometer per hour
m/s	meter per second
rad	radian
s	second

List of Acronyms and Abbreviations

ANI	Advanced Navigation and Instrumentation Research Group of Royal Military College
BRAM	Block Random Access Memory
b-frame	body frame
DCM	Direction Cosine Matrix
DGPS	Differential Global Positioning System
DWT	Discrete Wavelet Transform
EDK	Embedded Development Kit
EMC	External Memory Controller
FFT	Fast Fourier Transform
FOS	Fast Orthogonal Search
FPGA	Field Programmable Gate Array
GHz	Giga Hertz
GNU	GNU is Not Unix
GPS	Global Positioning System
GUI	Graphical User Interface
Gyro	Gyroscope
HDL	Hardware Description Language
IMU	Inertial Measurement Unit
INS	Inertial Navigation System(s)
I/O	Input/Output
IP	Intellectual Property cores
JTAG	Joint Test Action Group
KF	Kalman Filter/Filtering
l-frame	local level frame
LOD	Level of De-composition
LMB	Local Memory Bus
LPF	Low Pass Filter

MBit/s	Mega Bit per second
MHz	Mega Hertz
NCU	Navigational Computing Unit
OPB	On-Chip Peripheral Bus (IBM CoreConnect Bus)
OS	Operating System
PSD	Power Spectral Density
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read-Only Memory
RS-232	Recommended Standard 232 (asynchronous serial line standard)
RTOS	Real-Time Operating System
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver/Transmitter
WMRA	Wavelet Multi-Resolution Analysis
SCP	Soft Core Processor

CHAPTER 1: Problem Statement

1.1 Automobile Navigation

Navigation systems have always been essential for the aerial and marine vehicles. However, over the years, due to the increasing road density and intensive road traffic, the need for automobile (land vehicle) navigation has increased not just by providing location awareness but also by enhancing the vehicular control, safety and overall performance significantly [1]. Availability of appropriate technology is enabling light and affordable (low-cost) automobile navigational systems. In this way, it is becoming to be an attractive product in general consumer navigation market day by day and is expected to be an integral part of future automobile industry.

1.1.1 GPS Navigation

With the advancement of technology, automobile positioning and navigation has become an integral part of the automobile industry. The declining cost of GPS receivers over the past few decades has rendered Global Positioning System (GPS) attractive for automobile navigation applications. GPS provides position and velocity information to automobile users [2]. As a result, most of the present civilian automobile navigation devices are based on GPS technology. But, in the event of GPS signal loss, blockage and attenuation, these devices fail to perform accurately. GPS signal outages are usually caused by the obstruction by foliage, concrete overpasses, dense urban developments viz. tall buildings or tunnels. Thus, an alternative method for determining position in automobile navigation becomes necessary.

1.1.2 Inertial Navigation System

The suitable method which is widely used in this regard is Inertial Navigation System (INS). In modern days, almost all INS systems are Strap-down Inertial Navigation System (SINS) and they have the following two components:

a) Inertial Measurement Unit (IMU): It comprises inertial sensors such as three gyroscopes ('gyro' in short) and three accelerometers. The assembly of three gyro sensors and accelerometer sensors mounted orthogonally and located inside a moving

platform to calculate linear accelerations and angular velocities respectively is known as an Inertial Measurement Unit (IMU).

b) Navigation Computer: An IMU detects the current acceleration and rate of change of attitude (i.e. pitch, roll and yaw rates) in the earth's inertial frame. INS transforms accelerations and angular rates from inertial frame of reference to the navigation frame of the automobile and mathematically integrates to calculate the incremental position, velocity and attitude of the automobile. Then by summing up these incremental values of position, velocity and attitude with the initial values respectively, the navigational solution (i.e. position, velocity and attitude information) can be determined [4].

The terms Inertial Measurement Unit (IMU) and Inertial Navigation System (INS) are often confused. As mentioned above, an IMU is an instrument that measures specific forces and angular rates relative to an inertial frame of reference using its three orthogonally placed accelerometer and gyro sensors.

An INS contains an IMU as one of its components and also includes a computation unit namely NCU to derive meaningful navigation solution (position, velocity and attitude information) from IMU measurements. Effort has been made to use the terms IMU and INS in their proper context throughout this thesis. The term "IMU sensors" has been used specifically to refer to inertial sensors, also known as accelerometer and gyro sensors.

Unlike GPS which depends on external satellite aiding, INS is a self-contained system and is not affected by any external disturbances. It has a good short-term performance. But IMUs (accelerometer and gyro sensors) tend to be expensive and their performance and cost varies depending on their grade. Low-grade IMUs can provide navigational solution for a shorter period of time and their performance deteriorates immensely in the long run as they suffer from accumulated error. While calculating a navigational solution, measurement noise of low grade IMU sensors is accumulated as error and it increases dramatically with the passage of time. This leads to an ever increasing error in the navigational solution in the long-term.

1.1.3 GPS/INS Integration

In most cases, the problems of GPS as a stand-alone navigation system can be overcome at least for a short period of time by a GPS/INS integrated system. The main idea is to use both GPS and IMU devices together. In this integrated system, GPS signal data is used when it is available (i.e. there is no GPS signal outage and the signal is in the least erroneous state) to obtain position and velocity information. In the mean time, the INS in real-time mode continues to calculate the position, velocity and attitude. But as soon as an outage of GPS data is detected, the navigational solution from INS is put into use. This is the simplest possible way of integrating the two systems.

To improve navigational solution even further during a GPS outage, GPS/INS data is fused by any one of several optimal estimation techniques. They are Kalman filtering (KF), artificial neural network etc. KF uses dynamic model of INS and GPS errors, stochastic model of the inertial sensor errors and prior information about the co-variances of the data provided by the systems [4]. In this way, while there is no GPS outage, using several initial error characteristics, the filter models the overall system error characteristics. During a GPS outage, KF predicts or estimates the future position, velocity and attitude errors and also the output of the inertial sensors (accelerometer and gyro sensors). By correcting these errors, the overall navigational solution can be improved.

1.1.4 IMU Sensor De-noising

INS is inherently immune to the signal jamming and blockage vulnerabilities of GPS. But as the time progresses (during a GPS signal outage while using INS solution), the performance of the INS degrades while using low grade accelerometer and gyro sensors. The accuracy of low-grade INS is significantly affected by the low and high frequency noise characteristics of its inertial sensors. One of the techniques to improve the accuracy of the raw data of accelerometer and gyro sensors is to use a de-noising technique [5]. By pre-processing the IMU data using a de-noising technique, the effect of complex short-term (high frequency) and long-term (low frequency) noise characteristics produced by different error sources is reduced considerably.

1.2 Research Focus

The main objective of this research is to introduce a real-time implementation of GPS/INS integration module for land vehicles applications on the Xilinx MicroBlaze soft core processor (SCP) running on a low-cost Spartan-3 FPGA platform.

The research objectives of this thesis can be summarized as follows:

1.2.1 Objective 1

To implement algorithms introduced in the section 1.1 of the thesis using low cost Xilinx MicroBlaze SCP embedded on a low cost Spartan-3 FPGA. Most of the GPS devices provide directly 3 velocities and 3 positions. In contrast, the low-grade Micro-Electro-Mechanical-System (MEMS) based IMUs (accelerometer and gyro sensors) provide raw data of 3 accelerations and 3 angular velocities in vehicle's body frame. In the simplest form, implementing a GPS/INS integrated system may include the following steps:

a) Pre-processing raw IMU data: using a wavelet de-noising technique [6].

b) 2D and 3D Mechanization: converting the linear acceleration and rotation rates from the vehicle reference frame to position, velocity and attitude information in the navigation reference frame.

c) GPS/INS data fusion: using KF.

In short, the goal is to implement a NCU (Navigational Computing Unit) that executes the abovementioned techniques using the accelerometer and gyro sensor measurements along with an external GPS sensor data. More specifically, it was aimed to implement a NCU on a low cost embedded platform using low-cost sensor data, instead of an expensive microprocessor.

1.2.2 Objective 2

To provide a comparative output (i.e. navigational solution: position and velocity) analysis of GPS/INS integrated system implementation between the embedded platform and the microprocessor. In this regard, the use of embedded resources and its compact area utilization metrics show the simplicity of the automobile navigational application.

1.2.3 Objective 3

To provide a real-time navigational solution of the GPS/INS integrated system implementation. That is to say, the system accepts data in real-time mode and processes them. And then after a short time delay, it will provide the solution. To process data for such specific applications (GPS/INS integrated systems), the number of clock cycles is measured. And in that way, a timing analysis of the implementation is shown.

1.2.4 Objective 4

To present the GPS/INS integrated system implementation in such a way that it can be used as a model or generic platform/reference for the implementation of similar navigational algorithms which are still in research and development phase. This includes more effective and computationally complex IMU data pre-filtering algorithm such as the FOS (Fast Orthogonal Search) algorithm which is a high resolution spectral de-noising of low-end (MEMS-based) inertial sensors [5]. It also includes GPS/INS integration technique for resolving GPS outages with longer duration using artificial neural network methodology which is under research and development stage [5][6].

1.3 Thesis Outline

This Thesis is organized as follows.

Chapter 2 presents relevant concepts for this thesis. It introduces the background information required and presents the necessary mathematical equations from implementation point of view to gain an appreciation for the work conducted in this thesis. It discusses the key concepts behind INS operation. IMU sensors (inertial sensors) and their errors are also discussed, and are followed by an elaborate discussion of the KF technique for GPS/INS integration. The chapter ends with a description of inertial sensor pre-filtering techniques such as wavelet de-noising.

Chapter 3 begins with a summary of implementation environment, Spartan-3 FPGA as embedded system platform and the SCP Xilinx MicroBlaze. The implementation methodology of the research that was used to develop the hardware and the software using the equations represented in chapter 2 is presented in details here.

Chapter 4 presents the result of the embedded implementation described in chapter 3. It discusses the timing performance of the embedded navigational application to validate the real-time operation capability of the implementation and its software profiling output. Subsequently, the chapter outlines the hardware utilization summary of the embedded software. This is followed by a discussion of all the abovementioned results.

Chapter 5 draws main conclusions of the implementation methodology presented in this thesis. It provides some recommendations from the point of view of embedded implementation methodology for future designers and engineers who will be involved with similar kinds of implementation.

CHAPTER 2: Background on Vehicular Navigation

In Chapter 1, the automobile navigation was introduced. In this chapter, the concepts will be further explored so that the subsequent chapters of the thesis become understandable. First, general information regarding GPS is presented, including its principles of operation as well as common sources of errors. Before presenting inertial navigation mechanization and error equations, various navigation frames (to which navigational information is defined), Earth's model (to define Earth's the variable geometry and gravity parameters) and required mathematical (mostly geometry and trigonometry) concepts related to attitude representation are discussed. Then, using the INS error equations, GPS/INS integration through Kalman filtering (KF) technique is presented. Finally the INS raw data pre-processing using wavelet de-noising method/technique is presented. The understanding and the mathematical equations required to construct an automobile Navigational Computing Unit (NCU) that has been carried out in the Implementation Methodology (chapter 3) of the thesis is presented in this chapter.

2.1 Global Positioning System

GPS stands for Global Positioning System. It is a satellite-based radio-navigation system that is able to calculate position on the Earth. It consisted of 24 satellites (in 1993) orbiting at an altitude of 20200 kilometers in an approximate circular path around the Earth. According to Wikipedia, as of March 2008, there are 31 actively broadcasting satellites in the GPS constellation. GPS satellites are arranged so that a minimum of four satellites are placed in each of six Earth's orbital planes. In this way, four to ten satellites are always visible above an elevation angle of ten degrees from any place on earth. These satellites are continuously monitored by numerous worldwide ground stations. GPS satellites broadcast navigation messages and provide a 24-hour all-weather navigation service globally. GPS provides three different observations. They are pseudo-code, carrier phase and phase rate (also known as Doppler). The position update measurements can be derived by solving either pseudo-code or carrier phase

observations from at least four satellites. Using phase rate or the Doppler frequency of the received signal, GPS receiver determines the receiver's velocity [7].

2.1.1 GPS Operation

There are three main segments in the Global Positioning System namely Space, Control and User segment as shown in figure 2.1 (modified and reproduced from [8]). The Space Segment consists of orbiting satellites with antennas pointed towards the Earth that broadcast signals. Each satellite contains atomic clocks. The Control Segment which consists of worldwide base-stations that monitor the satellites to track their exact orbital position, altitude and speed in space and to make sure that they are operating correctly. The User Segment consists of available GPS receivers that detect, decode and process the signals received from the satellites. They are made up of hardware (also an antenna) and processing software for positioning, navigation and timing applications [7].

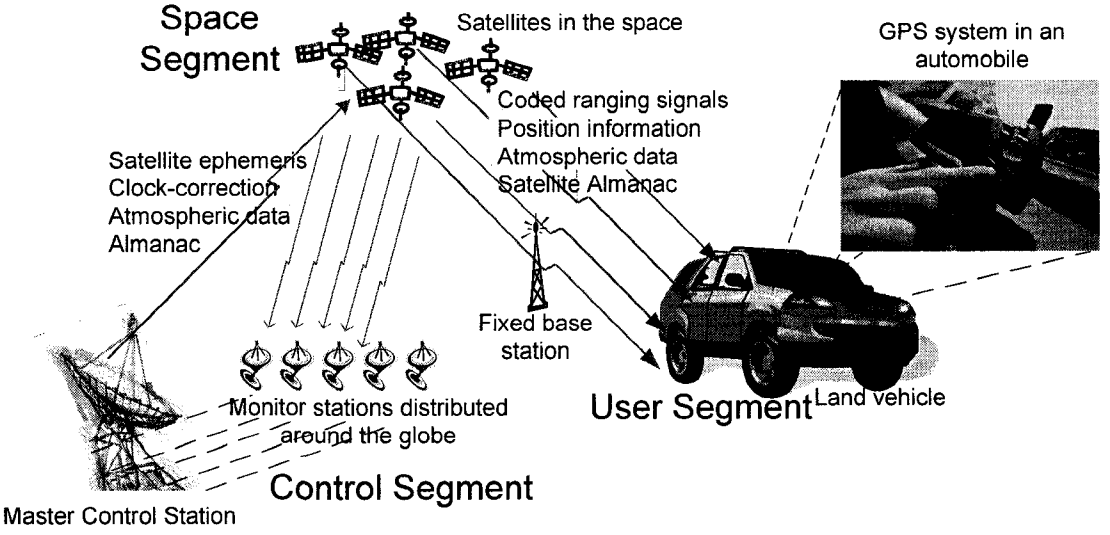

Figure 2.1: Space, Control and User segment of GPS.

Determining an exact position at the intersection of three spheres using three range information and exact co-ordinate information of the satellites is illustrated in figure 2.2 (modified and reproduced from [8]). A GPS receiver's position is calculated from the intersection of the signal propagating sphere. Here, the GPS satellites are located at the center of the spheres whose radiuses are the receiver-satellite distances.

The signal transmitted by a GPS satellite is detected by the antenna of a GPS receiver and is processed accordingly. It receives a signal using direct line of sight with any GPS satellite and determines the required time (time difference) taken by the received signal to travel from the corresponding satellite. The apparent transmit time of the satellite signal from a GPS satellite to a specific receiver is used to calculate the range in that GPS receiver. Mathematically, range = time difference × speed of light. If the GPS receiver clock and the GPS satellites clocks were synchronized with each others, only three range observations would be required to compute the receiver coordinates in 3-D space as shown below. But in reality GPS receiver clocks are not as precise as the GPS satellite clocks. This causes time synchronization error known as receiver clock bias. A range measurement with an error in time synchronization is referred to as a pseudorange. Thus, GPS signal contains the pseudorange information and the respective satellite coordinates as a function of time [7].

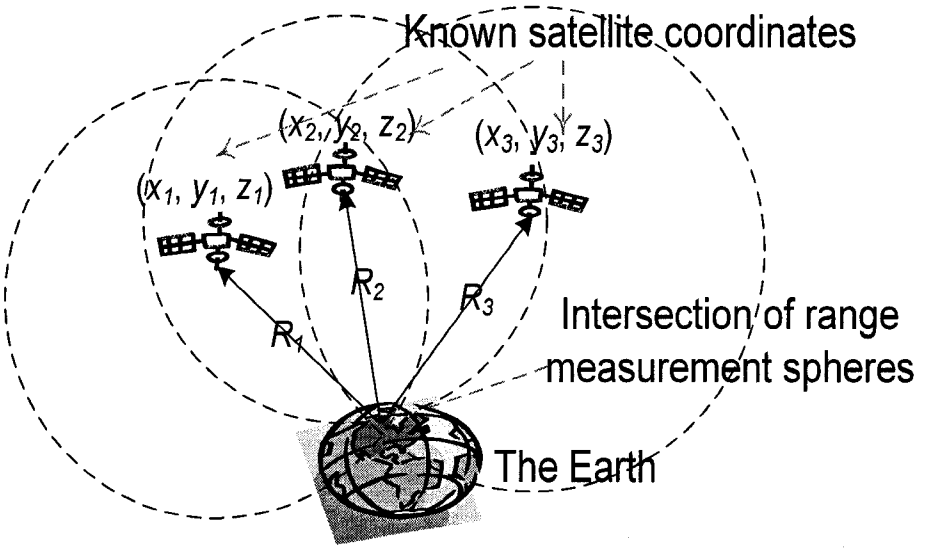

Figure 2.2: Illustration of Single Point Positioning using GPS.

Pseudorange includes the calculation of the range and receiver clock bias. By using at least four such measurements and the satellite position, the equation is reduced to determining four unknowns. They are the receiver's three position co-ordinates and

clock bias. In this way, pseudorange is used in a least square parametric model to solve for four unknowns [2]:

$$P_i = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} - b \quad (2.1)$$

where, i is the satellite index,

P is the pseudo range (m)

x_i, y_i, z_i are the coordinates of the i^{th} satellite (m, m, m)

x_u, y_u, z_u are the coordinates of user (m, m, m)

2.1.2 Drawbacks of GPS

GPS is able to provide precise positioning information to an unlimited number of users anywhere on the Earth. It is an absolute positioning system. Today, most vehicular (aerial, marine and also ground-based automobile) navigation systems rely mainly on the GPS receiver as it is a primary source of information to provide the position of the vehicle. [4]. Due to its availability through a comparatively low-cost and small-sized electronic receiver, the number of applications using GPS has increased dramatically over the last few years. However, under tunnels and overpasses, in downtown settings with high-rise buildings all around and in densely forest areas due to tall trees, the number of tracked satellites by a receiver may fall below four. This causes GPS outages when GPS receiver can no longer generate a navigation solution on its own. Thus, maintaining a direct line of sight between GPS receiver and with at least four GPS satellites is essential. In other words, without having a clear line of sight all the time between the satellite and the receiver, it is not possible to use GPS as a stand-alone navigation system for vehicular application [7]. The aftermath of a GPS outage can cause a discontinuity or a jump in the GPS carrier-phase measurement by an integer number of cycles and provide erroneous navigational solution.

Apart from signal outages, there are generally six standard errors that GPS signal data might contain. They are *ephemeris error*, *satellite clock error*, *receiver clock error*, *atmospheric disturbances*, *position dilution of precision* (caused by poor geometry of the satellites in the sky) and *multipath interference*. They contribute to the degradation of the accuracy of the receiver's performance [9]. Some of these error effects can be

reduced by combining GPS control segment and user segment with a GPS receiver in real-time mode.

The GPS control segment is made up of five monitor stations located around the world, four antennas and a Master Control Station (MCS) that tracks GPS satellites accurately. As the locations of the satellites are know by these stations precisely, an 'inverted' positioning process calculates the orbital parameters of the satellites and broadcast it in a regular time interval to the GPS receivers to minimize the GPS error effects [3]. The GPS MCS processes measurements taken at the monitoring base stations and develops predictions for the orbits (satellite orbital models) and satellite clock behavior. Then it sends this data to the antennas for transmission to the satellites for broadcast to the GPS receivers [10]. In this way, *ephemeris error* and *satellite clock errors* are significantly reduced.

In order to eliminate some more of those abovementioned errors, double differencing technique namely Differential Global Positioning System (DGPS) is used. DGPS is based on the simultaneous use of two or more receivers where one stationary reference or base receiver is located at a known location while the position of the other remote receiver is to be determined. The known position of the reference receiver is used to estimate corrections to the GPS derived position. These corrections are then transmitted to the remote receiver and thus the remote receiver computes position with more accuracy. DGSP employs the fact that GPS errors are very similar over a distance of up to several hundred km and ensures most of the time a meter level accuracy [2]. The *atmospheric disturbance* to the GPS signal propagation is countered to an extent by DGSP technique.

GPS signals propagate in the form of microwave radio waves and get reflected by solid objects like buildings, large canopy etc. as they cannot penetrate them. As a result, instead of coming from direct line-of-sight, the GPS satellite signal can arrive to the GPS receiver from different paths of reflection. This phenomenon is called *multipath interference* and it results in the distortion in the range measurement. Its impact on the

measurements is uncorrelated between two receivers (or even among control segments) and thus cannot be reduced using DGPS [7].

2.2 Navigation Frame

The various navigation frames to which the position information is defined are introduced in this section and presented in the figures 2.3 to 2.6 (all of them have been reproduced from [4][12]). These navigation co-ordinate systems are used in the subsequent sections to derive INS mechanization and KF equations.

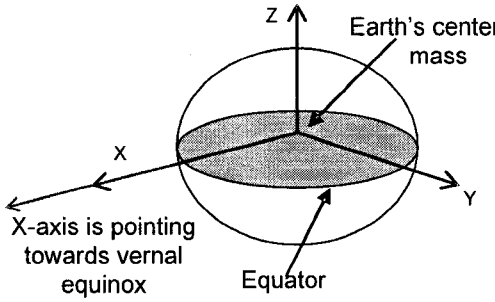

Figure 2.3: Inertial frame (*i*-frame).

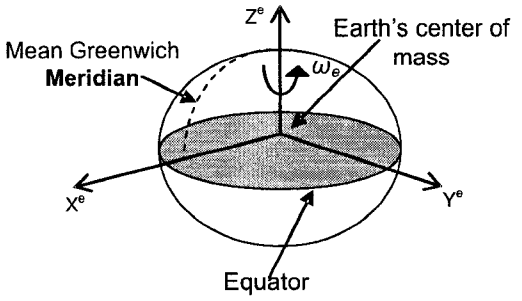

Figure 2.4: Earth frame (*e*-frame).

2.2.1 Inertial frame (*i*-frame)

Inertial frame has its origin at the centre of the Earth and its axes are **stationary** (non-rotating) with respect to the fixed stars [11]. In this frame, the *Z*-axis is coincident with the Earth's polar axis, the *X*-axis points towards the mean vernal equinox and the *Y*-axis points towards the direction to complete the right-handed orthogonal rule. All inertial sensors produce measurements relative to an inertial frame but resolved along the instrument-sensitive body frame [10].

2.2.2 Earth frame (*e*-frame)

Like the inertial frame, *e*-frame has its origin at the centre of the Earth. The axes are fixed with respect to the Earth, in the figure 2.4 defined by the axes X^e, Y^e, Z^e with Z^e along the Earth's polar axis. The axis X^e lies along the intersection of the plane of the Greenwich Meridian with the Earth's equatorial plane. The Y^e-axis points towards the direction to complete the right-handed orthogonal rule [11]. The Earth frame rotates with respect to the inertial frame at an angular rate $\omega_e = 2\pi / 24$ rad/hr $\approx$ 15°/hr which is also

referred to as earth's rotation rate about the Z-axis. In other words, the Z^e axis is parallel to the spin axis of the Earth.

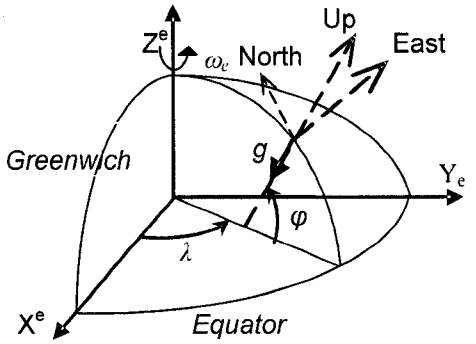

Figure 2.5: The local-level frame (*l*-frame)

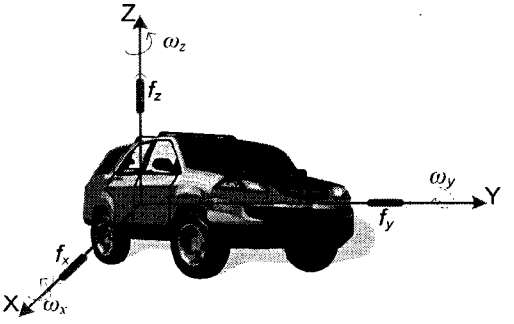

Figure 2.6: The body frame (*b*-frame).

2.2.3 Local level frame (*l*-frame)

Local level frame is a local geodetic frame and its origin coincides with the inertial sensors frame. With the local level ENU (East, North and UP) frame, the X, Y and Z axes of the inertial sensor are aligned with geodetic East, North and vertical up direction respectively. The ENU frame is used as the frame of reference in this thesis while converting the accelerometer and gyro measurements into position, velocity and attitude information. In the same way, the GPS "navigational solution" refers to position and velocity information in this frame. Figure 2.5 and figure 2.14 (reproduced from [11]) illustrate the ENU frame or the local level frame.

2.2.4 Body frame (*b*-frame)

It is an orthogonal axis set and is made coincident with the axes of the vehicle's moving platform in which the sensors are mounted. As shown in figure 2.6 (reproduced from [11]), the y axis is defined in the forward direction, the z axis is defined pointing to the vertical up direction of the vehicle and the x axis completes the right-handed orthogonal co-ordinate system.

2.3 Earth Models

In this section, approximations to the Earth's shape and gravity models tailored (simple and suitable) for land vehicle navigation amenable to simple mathematical descriptions are presented. An accurate model of the Earth's shape is necessary so that

an accurate solution (position, velocity and attitude) of a moving vehicle can be resulted from its inertial mechanization process. The Earth's gravity model is used to determine what part of the sensed acceleration by the inertial sensors is due to vehicle dynamics and what is due to the Earth's gravitational attraction [12] [14].

2.3.1 Ellipsoid Geometry

Owing to the slight flattening of the earth at the poles, it is customary to model the earth as a reference ellipsoid instead of considering the typical perfect sphere model. It approximates more closely to the true geometry than the spherical model as shown in figure 2.7 (modified and reproduced from [13] [14]) and 2.8 (modified and reproduced from [10] [11][12]).

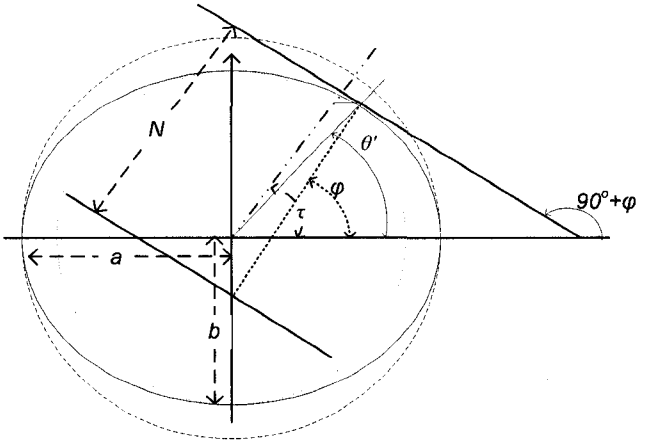

Figure 2.7: Illustration of eccentric (τ), geocentric (θ') and geodetic latitude (φ).

Here, N is the prime vertical radius of the best fitting Earth ellipsoid (East-West) and M is the corresponding meridian radius of curvature (North-South). They are also known as meridian radius and denoted as N in figure 2.7 and 2.8 and transverse radius of curvature respectively. They are expressed as [10] [11][14]:

$$N = \frac{a}{(1-e^2 \sin^2 \varphi)^{1/2}} \tag{2.2}$$

$$M = \frac{a(1-e^2)}{(1-e^2 \sin^2 \varphi)^{3/2}} \tag{2.3}$$

Where, Length of the semi major axis, a = 6378137 m
Length of the semi minor axis, $b = a(1-f) = 6356752.3142$ m
Fattening of the ellipsoid, $f = (a-b)/a$ = -0.0033528

Eccentricity of the ellipsoid, $e = [f(2-f)]^{1/2} = [(a^2-b^2)/a^2]^{1/2}$ = 0.0818

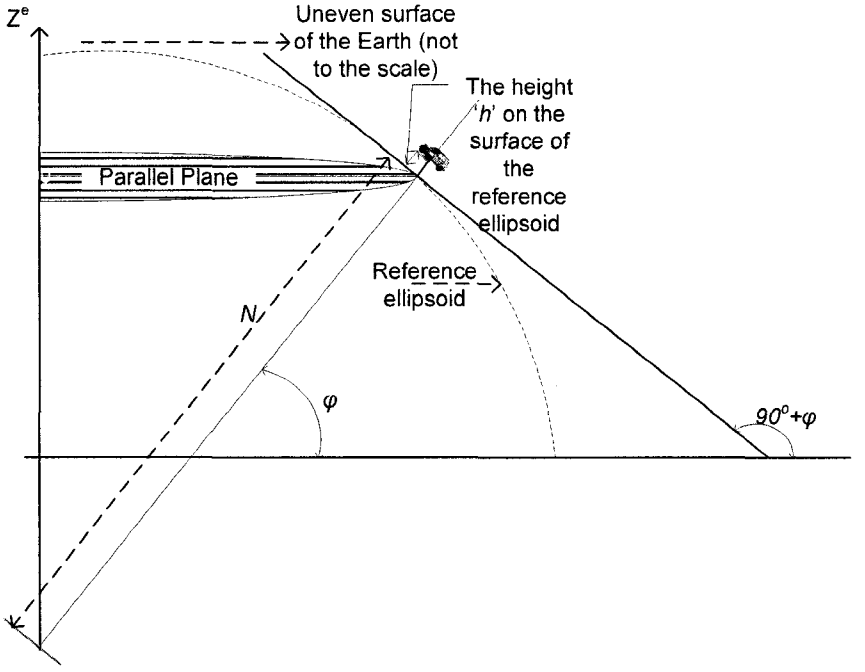

Figure 2.8: Local Meridian plane of reference ellipsoid.

2.3.2 Ellipsoid Gravity

Magnitude and the direction of the gravity vector vary with position on the Earth's surface and height above it. Inhomogeneous mass distribution of the Earth is a factor for this variation [11]. Precise knowledge of the gravity vector is important for certain high accuracy applications such as for marine navigation. Gravity deviations are represented by the following gravity vector referenced in the local level frame (ENU frame) [12]: $g = [-\zeta g \quad \eta g \quad -g]^T$

Where, ζ = meridian deflection of the vertical (+ve about east), η = normal deflection of the vertical (+ve about north) and g = gravity magnitude (+ve about up). As height deflection is not usually a major issue unless driving on a mountainous trajectory, in automobile navigation, the gravity vector is approximated: $g = [0 \quad 0 \quad -g]^T$

Here, g is computed from the following equation where C_{00}, C_{10}, C_{20}, C_{01}, C_{11}, $C_{21}$and C_{02} are constant values listed in table 2.1 [13]:

$$g = (C_{00} + C_{10}\sin^2\tau + C_{20}\sin^4\tau) + (C_{01} + C_{11}\sin^2\tau + C_{21}\sin^4\tau)h + C_{02}h^2 \quad (2.4)$$

Table 2.1: Constant coefficient (unit in m/sec^2) for normal gravity.

C_{00}	9.780326582929618
C_{10}	$5.197841463945455\times10^{-2}$
C_{20}	$-1.18852395328380\times10^{-4}$
C_{01}	$-9.411353888873278\times10^{-7}$
C_{11}	$.347079301177616\times10^{-9}$
C_{21}	$-3.034117526395185\times10^{-12}$
C_{02}	$6.685260859851881\times10^{-14}$

Here, the eccentric latitude (τ) is defined by the following equation:

$$\tau = \tan^{-1}[(b/a)\tan\varphi] \quad (2.5)$$

2.4 Attitude Representations

Various mathematical representations are used to define the attitude of a body with respect to a co-ordinate reference frame. The parameters associated with each method are updated as the vehicle rotates using turn/rotation measurements provided by the gyro [11]. The attitude representations discussed in this section are used to derive 3D INS mechanization equations. The three attitude representations are the following.

2.4.1 Direction Cosine Matrix

The direction cosine matrix (DCM) is a 3 by 3 matrix, the columns of which represent unit vectors in body axes projected along the reference axes. It is denoted by the symbol R_b^l. It can be written here in component form as follows:

$$R_b^l = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

The columns represent unit vectors in body axes projected along the reference axes. The element in the ith row and the jth column represents the cosine of the angle between the i-axis of the reference frame and the j-axis of the body frame [11].

2.4.2 Euler Angles

A transformation from one co-ordinate frame to another can be carried out as three successive rotations about different axes taken in turn [11] [12]. For instance, a transformation from reference axes to a new co-ordinate frame may be expressed as follows where ψ, θ and ϕ are referred to as the Euler rotation angles.

Rotation through angle ψ about reference z-axis can be expressed by as [11] [12]:

$$R_1 = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation through angle θ about new y-axis after the first rotation as the y axis has been transformed to a new position. It can be expressed as [11] [12]:

$$R_2 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

Rotation through angle ϕ about new x-axis (after the first and second rotation mentioned above, this axis has been transformed). It can be expressed as [11] [12]:

$$R_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$$

2.4.3 Quaternion

The quaternion attitude representation allows a transformation from one co-ordinate frame to another to be effected by a single rotation (about a vector $\bar{\mu}$ defined in the reference frame). Theoretically, quaternion is a four-element vector representation, the elements of which are functions of the orientation of this vector and the magnitude of the rotation [11]:

$$Q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} (\mu_x / \mu)\sin(\mu/2) \\ (\mu_y / \mu)\sin(\mu/2) \\ (\mu_z / \mu)\sin(\mu/2) \\ \cos(\mu/2) \end{bmatrix}$$

Here, Q is the vector of Quaternion parameters. μ_x, μ_y and μ_z are the components of the angle vectors and μ is the magnitude of $\bar{\mu}$. The rotation angle $\mu = \sqrt{\mu_x^2 + \mu_y^2 + \mu_z^2}$

2.4.4 Relationships between DCM, Euler Angles and Quaternion

To solve INS attitude mechanization equations, quaternion is used over the parameterization of the rotation matrix R_b^l due to computational simplicity. Four differential equations are solved numerically instead of six differential equations if the rotation matrix R_b^l is manipulated directly. It also avoids the singularity problem introduced with some other solutions methods (with DCM and/or Euler angles methods). Quaternion parameters can be expressed using the DCM elements in the following way [10] [11]:

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} 0.25(R_{32} - R_{23})/q_4 \\ 0.25(R_{13} - R_{31})/q_4 \\ 0.25(R_{21} - R_{12})/q_4 \\ 0.5\sqrt{1 + R_{11} + R_{22} + R_{33}} \end{bmatrix} \tag{2.6}$$

In the reverse way, DCM can be formed using the Quaternion parameters [10][11][12]:

$$R_b^l = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1 q_2 - q_3 q_4) & 2(q_1 q_3 + q_2 q_4) \\ 2(q_1 q_2 + q_3 q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2 q_3 - q_1 q_4) \\ 2(q_1 q_3 - q_2 q_4) & 2(q_2 q_3 + q_1 q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{2.7}$$

A transformation from reference to body axes may be expressed as the product of three separate transformations: $R_l^b = R_3 R_2 R_1$

In the same way, the inverse transformation from body to reference axes is given by [11][12]: $R_b^l = R_l^{bT} = R_1^T R_2^T R_3^T$

$$R_b^l = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

$$R_b^l = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{2.8}$$

As gyro sensors provide data at a higher sampling rate and in that way any attitude increment remains always small, it can be assumed that $\sin\phi \rightarrow \phi$, $\sin\theta \rightarrow \theta$, $\sin\psi \rightarrow \psi$ and $\cos\phi \rightarrow 1$, $\cos\theta \rightarrow 1$ and $\cos\psi \rightarrow 1$. Making these substitutions in the above equation and ignoring products of angles which also become small, the DCM expressed in terms of the Euler rotations reduces approximately to the skew symmetric form [11][12]:

$$R_b^l = \begin{bmatrix} 1 & -\psi & \theta \\ \psi & 1 & -\phi \\ -\theta & \phi & 1 \end{bmatrix} \tag{2.9}$$

The Euler angles may be derived directly from the DCM elements [11]:

$$\phi = \arctan\left[\frac{r_{32}}{r_{33}}\right] \tag{2.10}$$

$$\theta = \arctan\left[-r_{31}\right] \tag{2.11}$$

$$\psi = \arctan\left[\frac{r_{21}}{r_{11}}\right] \tag{2.12}$$

2.5 Inertial Navigation System

By measuring the accelerations and rotations applied to the inertial frame of vehicle (and using initial position, velocity and attitude), an Inertial Navigation System (INS) provides its position, velocity and altitude. Inertial Measurement Unit (IMU) refers to the equipment containing an orthogonal triad of accelerometer and gyro sensors. IMU is a part of INS and it detects the current acceleration and rate of change in attitude (i.e. pitch, roll and yaw rates).

The output of an accelerometer due to a gravitational field is the negative of the field acceleration. In vector notation, it is given as: $\vec{f} = \vec{a} - \vec{g}$. Here, $\vec{f}$ = specific force measured by accelerometer, $\vec{a}$ = acceleration with respect to the inertial frame and $\vec{g}$ = gravitational acceleration. Then the navigation computing unit (NCU) which is the other part of the overall INS instrument processes them using mechanization equations to find the total change from the initial position. In contrast to the *absolute* positioning system

like GPS, INS is a *relative* positioning system. INS derives its position and altitude from integrating inertial referenced accelerations and angular velocities [4].

2.5.1 Grades of IMU

IMU sensors are of different grades based on the type of technology used to build them and accordingly are of different cost. The higher the cost the better is the sensor output. According to the performance quality and cost, they are usually labeled in three different grades. They are *strategic*, *navigation* and *tactical* grades. The emergence of inertial sensors made of micro-electro-mechanical systems (MEMS) in the past decade has caused MEMS grade sensors to be widely used as well. Strategic grade IMUs are very expensive and cost more than US $250,000. They provide highly accurate navigational solution. Navigational grade IMUs are less accurate and cost in the range of $70,000 to $100,000 USD. Tactical grade IMUs are even less accurate and less expensive. MEMS-grade units are very small and inexpensive. The price is in the order of US $500 to $2000 and while produced in mass quantity, the unit price can go below $10. Due to inherent sensor noise, their solution tends to degrade rapidly (in the long run) [4].

Micro-Electro-Mechanical Systems (MEMS) technology has shown promise for the development of low cost IMUs. Advances in MEMS and computer technology combined with the miniaturization of electronics have made it possible to produce chip-based inertial sensors. They are inexpensive, small and consume low power. A recent development of a complete MEMS IMU/GPS integrated navigation system by the Mobile Multi-Sensor System (MMSS) Research Group at the University of Calgary had price range of US $100-200 [15]. Thus, if a low-cost navigation solution comprising integrated IMU and GPS sensors is produced in a very mass scale, the gyro and accelerometer sensors price should be as low as below US $10.00 [16].

2.5.2 Types of Inertial Navigation Systems

There are two distinct arrangements of accelerometer and gyro sensors and. they are. *Gimbaled mechanized* and *Strapdown*. In gimbaled mechanized arrangement, IMU sensors are commanded to maintain the platform frame alignment with a specific

navigation coordinate system. In this way, in spite of vehicle motion, the platform does not experience any rotation relative to the navigation frame. Thus, accelerometers aligned with the platform measure the specific force along the *navigation frame* axes. After proper scaling and direct mathematical integration, this measured acceleration yields the desired position and velocity vectors. Vehicle attitude is determined by measurement of the relative angles between the vehicle and the platform.

Strapdown Inertial Navigation System (SINS) is a self-contained positioning and attitude device that continuously measures (sampling rate higher than 50 Hz) three orthogonal accelerations and three angular rates. Using these measurements, it calculates the incremental position, velocities and attitude angles provided that initial position, velocities and attitude were known. In this way, it keeps track of the vehicle's attitude, more importantly the heading or azimuth angle of automobile vehicle with respect to a reference frame. Currently almost all INS are of SINS due to their advantages in reliability, higher output rate, low power consumption, light weight, low cost and flexibility (mechanically less complex). In SINS, the sensors experience the full dynamic motion specifically higher rotation rates of the vehicle and produce data with higher range of bandwidth. Due to this increased dynamic range, it contains more noise which causes gyro scale-factor error and nonlinearity [10].

2.6 IMU Sensor Errors

The accuracy of INS is significantly affected by the error characteristics of the IMU sensors. The nature of the sensor errors can be categorized into two parts. They are *Deterministic* and *Stochastic* Error.

Bias offset and Scale factor of accelerometer and gyroscope sensor are *deterministic* errors. Bias refers to the offset in the measurement provided by an inertial sensor. By definition, the bias of a signal is the signal it gives when there is no input. The deterministic part of bias is called bias offset and can be determined by calibration. The *stochastic* part of it is called bias drift as it is random in nature and it varies with different factors like time, temperature etc. [4].

The scale-factor of an inertial sensor is the relationship between the output signal and the quantity being measured. For example, a gyro provides a measurement of turn rate about a given axis. The output of the gyro may take the form of a voltage or current (an analog output) proportional to the applied turn rate plus a constant bias term caused by the various imperfections within the sector. To convert it to a meaningful quantity and unit, IMU calibration is used. Deviation from theoretical (measured by IMU calibration) scale factor with temperature and repeatability causes scale factor instability. Like bias drift, scale factors can be of random nature and can be modeled *stochastically*. Unknown random (non constant) variations in bias and scale factor also significantly contribute to inaccurate navigational solutions [4].

The IMU sensors in an INS have significantly complex short-term (high-frequency) and long-term (low frequency) noise characteristics and they are contributed by many different error sources. IMU sensor noise is a kind of common stochastic error resulting from the sensor itself and/or other electronic equipment that interfere with the measured output signals. Some kinds of noise are generally distributed across the frequency spectrum and others are frequency centered. Unknown zero-mean additive noise on the sensor outputs, quantization noise, computational noise (from mechanization) and electronic noise are also major factors and these noises are usually unpredictable. But its *statistical* properties are used in KF to estimate drifting scale factor and biases. Axes misalignment is another kind of stochastic error resulting from the imperfection of mounting the sensors [4]. IMU Calibration is discussed in the next section to counter the deterministic errors. IMU alignment is then presented as both of these procedures are crucial before inertial navigation starts.

2.6.1 IMU Calibration

Accelerometer sensor calibration provides the relationship between the sensed specific force and the actual specific force. In the same way, gyro sensor calibration relates the sensed rate of rotation and the actual rate of rotation. The reference acceleration is the magnitude of the local apparent gravity vector at the calibration site. And the reference angular is the Earth's rotation rate. Thus accelerometer and gyro

calibrations are performed at a location where the gravity vector magnitude and geodetic position have been determined with great precision. At various orientations of the sensors, the outputs of the accelerometers and gyros are compared to reference values and during navigation these differences are used to generate corrections to the measured specific force and angular rate respectively [4][14].

Table 2.2: Using six-position calibration technique, calculation of deterministic bias and scale factor (of accelerometer and gyro sensors).

	Measurement for up position	Measurement for down position	Calculated Bias	Calculated Scale Factor
Accelero-meter	$Z_{up} = b-(1+s)g$	$Z_{down} = b+(1+s)g$	$\frac{Z_{up}+Z_{down}}{2}$	$\frac{Z_{down}-Z_{up}-2g}{2g}$
Gyro	$\omega_{up} = b_\omega+(1+S_\omega)\omega_e \sin\varphi$	$\omega_{down} = b_\omega-(1+S_\omega)\omega_e \sin\varphi$	$\frac{\omega_{up}+\omega_{down}}{2}$	$\frac{\omega_{up}-\omega_{down}-2\omega_e \sin\varphi}{2\omega_e \sin\varphi}$

Deterministic bias drift and scale factor calculation for IMU sensors can be achieved by different calibration techniques namely local level frame calibration, six position static test and angle rate tests. In six position static test, the sensors are mounted on a level table with each sensitive axis pointing alternately up and down directions in a static mode. Measurements are taken for 10-15 minutes and they are averaged. Deterministic bias and scale factor are calculated by summing up and differencing combinations of the averaged measurements [4]. Table 2.2 contains the required mathematical notions where b, S, b_ω, S_ω, φ, ω_e and g represent accelerometer bias, accelerometer scale factor, gyro bias, gyro scale factor, latitude of the calibration location, Earth's rotation rate ($\omega_e = 2\pi/24$ rad/hr $\approx$ 15°/hr) and gravity, respectively.

2.6.2 IMU Alignment

The SINS body frame can take any arbitrary direction as the accelerometer and gyro sensors are strapped down to the vehicle which can be oriented in any direction with respect to the navigation frame. The principle of strap down inertial navigation (SINS) assumes that initial information about the system is already known. While the starting position and velocity can be obtained using GPS assistance, the initial orientation of the system is not typically available. Therefore, the INS requires

performing an initial alignment which produces coincidence between the sensor axes of the IMU with the local level frame (shown in figure 2.5 and figure 2.14). The purpose of this alignment is to establish the relationship between the body frame and the local level frame. Thus the initial parameters of the rotation matrix (R_b^l) between the body frame to the local-level frame are calculated. Once the alignment is done, the rotation rates measured by the gyros are used to constantly update the R_b^l matrix. This updated matrix is then used to transform the accelerometer measurements to the navigation frame [4].

Initial alignment is done in two steps, namely accelerometer leveling and gyro compassing. Accelerometer leveling aligns the z-axis of the accelerometer triad to the z-axis of the local frame by driving the horizontal accelerometer outputs to the value zero. After accelerometer leveling, gyro compassing is performed based on the principle of sensing a component of the Earth rotation by gyro sensors. This component ($\omega_e \cos\varphi$) is at its maximum when the sensitive axis points North and zero when it points East [4]. Accurate alignment is necessary to achieve satisfactory navigation solution as it can severely influence the performance of an inertial navigation [11].

2.7 2D INS Mechanization Equations

The idea of mechanization is to determine velocity and position in a desired co-ordinate system. The sequence of equations used to convert the IMU outputs of angular rates and accelerations to INS outputs of positions, velocities and attitudes are called Mechanization Equations. They include a set of first order differential equations that transform the raw IMU measurements into position, velocity and attitude components [10]. The computational processes required to perform the navigation task in 2D are much simplified compared with a full strap down system in a 3D space. In other words, being functionally identical, 2D Mechanization presents a foreword for the much complex 3D Mechanization equations [11].

A vehicle constrained to move in a single plane can use an inertial navigation system to find its navigational solution. In this case, the system contains two accelerometers and a single axis rate gyro attached rigidly to the body of the vehicle.

The azimuth angle (A) is obtained from mathematically integrating the gyro angular velocity. The initial azimuth angle A_0 is required for this.

$$A(t) = \int_0^t \omega_{gyro} dt + A_0 \tag{2.13}$$

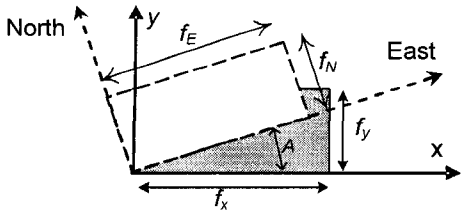

Figure 2.9: Rotation of body frame by about the vertical axis.

Measurements taken by the sensors are in the body frame as the sensors are mounted in this frame. These measurements are transformed to the navigation frame:

$$f^{navigation} = R_b^n f^{body}$$

Here, $f^{navigation}$ is the acceleration in the navigation frame (which is the local level frame). In a 2D scenario, the local level frame constitutes only East and North direction. f^{body} is the acceleration in the sensor's body frame and R_b^n is the rotation matrix which transforms body frame parameters to navigation frame. In detailed equation form:

$$f_E = f_y \sin A + f_x \cos A$$
$$f_N = f_y \cos A - f_x \sin A$$

In matrix form: $$\begin{bmatrix} f_E \\ f_N \end{bmatrix} = \begin{bmatrix} \cos A & \sin A \\ -\sin A & \cos A \end{bmatrix} \begin{bmatrix} f_x \\ f_y \end{bmatrix} \tag{2.14}$$

Here, as illustrated in figure 2.9 (reproduced from [4][12]) and shown in equation (2.14), f_x and f_y are the accelerometer measurements along the x and y directions of the body frame. f_E and f_N are the corresponding East and North accelerations in the local level frame and A is the azimuth angle. By mathematically integrating, the incremental values of position and velocity can be calculated.

$$V(t) = V(t_0) + \int_0^t f(t) dt \tag{2.15}$$

$$P(t) = P(t_0) + \int_0^t V(t) dt \tag{2.16}.$$

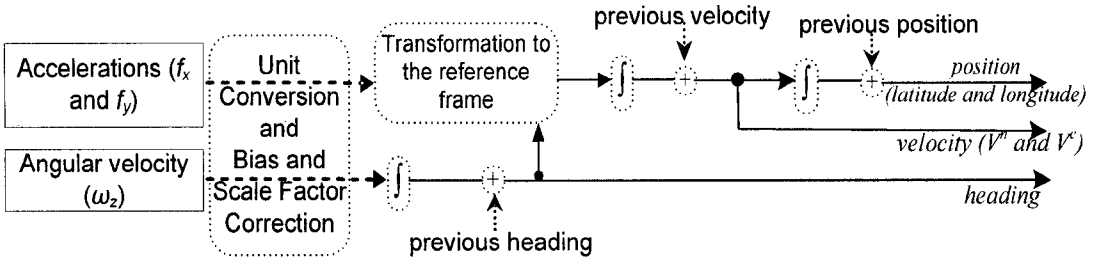

Figure 2.10: Block diagram illustrating 2D INS Mechanization.

Figure 2.10 (modified and reproduced from the [4]) summarize the mathematical equations involved in 2D Mechanization procedure.

2.8 3D INS Mechanization Equations

The previous section has outlined the basic form of the computing tasks to be implemented in a strap down system using a simplified 2D representation. In contrast, attitude information in 3D Mechanization cannot be obtained by a simple integration of the measured angular rates. In this scenario, the three gyro outputs contain not just only actual angular velocities of the moving body, but also both the Earth's rotation and the change in orientation of the local-level frame. The measured angular velocities by gyro can be decomposed as (in the skew-symmetric matrix equation form representation):

$$\Omega_{lb}^{b} = \Omega_{li}^{b} + \Omega_{ib}^{b} = \Omega_{ib}^{b} - \Omega_{il}^{b} \tag{2.17}$$

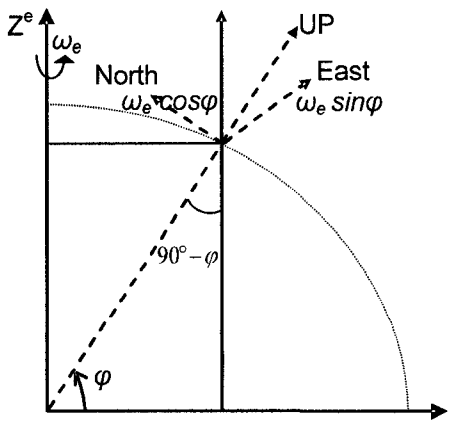

Figure 2.11: : De-composition of the Earth's rotation in local level frame

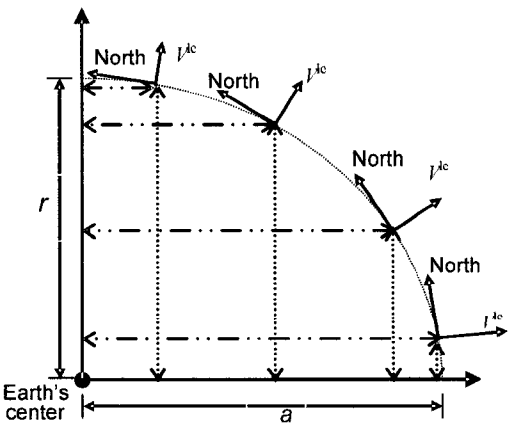

Figure 2.12: Change of local North and Vertical directions during motion over the surface.

Ω_{ib}^{b} is the skew-symmetric matrix of the measurements of angular velocities provided by gyroscopes and it can be calculated directly from gyro outputs:

$$\Omega_{ib}^{b}=\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.18}$$

Since, $\omega_{ie}^{b}=R_{l}^{b}\omega_{ie}^{l}$ and $\omega_{el}^{b}=R_{l}^{b}\omega_{el}^{l}$, then $\omega_{il}^{b}=\omega_{ie}^{b}+\omega_{el}^{b}=R_{l}^{b}(\omega_{ie}^{l}+\omega_{el}^{l})$. In elaborate form, it can be written as follows and in that way Ω_{il}^{b} can be calculated:

$$\omega_{il}^{b}=R_{l}^{b}\left[\begin{pmatrix} 0 \\ \omega_e\cos\varphi \\ \omega_e\sin\varphi \end{pmatrix}+\begin{pmatrix} \dfrac{-V^n}{M+h} \\ \dfrac{V^e}{N+h} \\ \dfrac{V^e\tan\varphi}{N+h} \end{pmatrix}\right] \tag{2.19}$$

Here, Ω_{ie}^{l} and Ω_{el}^{l} are the skew-symmetric matrices corresponding to the angular velocities ω_{ie}^{l} and ω_{el}^{l} respectively. The angular velocities ω_{ie}^{l} and ω_{el}^{l} are illustrated in the figure 2.11 and figure 2.12 (reproduced from [11]) can be expressed mathematically as:

$$\omega_{ie}^{l}=\begin{bmatrix} 0 & \omega_e\cos\varphi & \omega_e\sin\varphi \end{bmatrix} \tag{2.20}$$

$$\omega_{el}^{l}=\begin{bmatrix} \dfrac{-V^n}{M+h} & \dfrac{V^e}{N+h} & \dfrac{V^e\tan\varphi}{N+h} \end{bmatrix}^{T} \tag{2.21}$$

The mechanization process can be separated into three parts:

2.8.1 Attitude Mechanization

The initial attitude angles obtained (usually from the alignment procedure): pitch (θ), roll (ϕ) and azimuth (ψ) angles form the rotation matrix R_{b}^{l}. By solving the time derivative equation of the transformation matrix, the incremental attitude angles of the moving body is determined:

$$\dot{R}_{b}^{l}=R_{b}^{l}\Omega_{lb}^{b}=R_{b}^{l}(\Omega_{ib}^{b}-\Omega_{il}^{b}) \tag{2.22}$$

To solve the above differential form, quaternion parameters for the initial time step (Q_k at time t_k) are calculated from the initial DCM R_{b}^{l}. Then, the quaternion

parameters for the next time step (Q_{k+1} at time t_{k+1}) based on the values of the quaternion parameters Q_k at time t_k are determined: $Q_{k+1} = Q_k + \left(\frac{1}{2}\Omega(\omega_k)Q_k\right)\Delta t$

Once the quaternion parameters are determined for the next time step (Q_{k+1} at time t_{k+1}), the rotation matrix (in the DCM form) R_b^l can be obtained using the DCM expressed in quaternion parameters relation. The updated rotation matrix R_b^l is then used in transforming the more recent (newly sampled) accelerometer measurements to the navigation (or in the local level frame) frame. Using trigonometric relations between the Euler angles and the DCM, pitch (θ), roll (ϕ) and azimuth (ψ) angles can also be obtained [4].

2.8.2 Velocity Mechanization

The velocity is expressed by three components along the East (V^e), North (V^n) and vertical (V^u) directions, $V^l = [V^e \quad V^n \quad V^u]^T$.

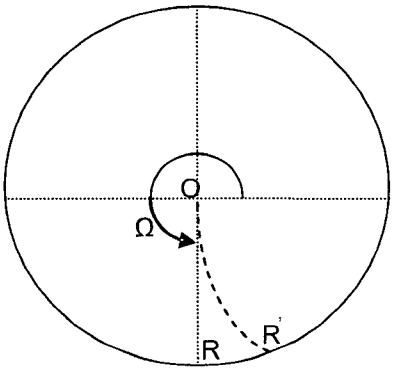

Figure 2.13: Coriolis acceleration on rotating Earth.

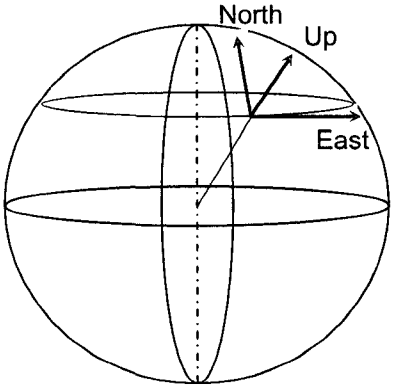

Figure 2.14: Local level ENU frame.

The acceleration of the moving platform is measured by the accelerometers in three mutually orthogonal directions in body frame $f^b = [f_x \quad f_y \quad f_z]^T$. Velocity components cannot only be deduced integrating directly from acceleration components in the local-level frame of the moving component. To transform these measurements into the local level frame, the updated rotation matrix (R_b^l) calculated in the previous section (2.7.1. Attitude Mechanization) is used as follows:

$$f^l = [f_e \quad f_n \quad f_u]^T = R_b^l f^b = R_b^l [f_x \quad f_y \quad f_z]^T \tag{2.23}$$

The $R_b^l f^b$ term transforms the acceleration measurements from the body frame to the local level frame. In addition to the sensed acceleration of the moving body and the Earth's gravitational acceleration, the accelerometer measurements include for the apparent acceleration sensed by the vehicle as it moves within a rotating coordinate frame. This apparent acceleration is caused by the Earth's angular rotation rate (illustrated in figure 2.11) and the change in orientation of the local-level frame (illustrated in figure 2.12). As previously discussed, the skew-symmetric matrix Ω_{ie}^l and Ω_{el}^l accounts for the effect of the Earth's rotation ω_e at the vehicle's position and the change in orientation of the local level with respect to the Earth respectively. It is also referred to as Coriolis acceleration. The Coriolis acceleration effect on a moving point on a rotating platform is illustrated in two dimensions in figure 2.13 (modified from [11]). As the point moves away from the axis of rotation, it traces out a curve in space as a result of the rotation. The dotted line in figure 2.13 is the trajectory required to travel from O to R on a rotating earth.

Taking the abovementioned three factors (sensed, apparent and gravitational acceleration) into consideration, the accelerometer output vector relation $\vec{f} = \vec{a} - \vec{g}$ can be transformed from the inertial frame to the local level frame (in this case the ENU frame) as the first order differential equation as (for details see reference [11]):

$$\dot{V}^l = R_b^l f^b - (2\Omega_{ie}^l + \Omega_{el}^l)V^l + g^l \tag{2.24}$$

2.8.3 Position Mechanization

In the local level frame, the position of a platform is expressed in terms of the curvilinear coordinates i.e. latitude, longitude and altitude: $r^l = [\varphi \quad \lambda \quad h]^T$.

In figure 2.15 (reproduced and modified from [11]), latitude calculations have been illustrated on a meridian plane, an imaginary great circle on the earth's surface passing through the North and South geographic poles. The change of latitude is expressed as:

$$\dot{\varphi} = \frac{V^n}{M+h} \tag{2.25}$$

In the same figure, the longitude calculations have been illustrated on a parallel plane (the Earth has been sliced parallel to the equatorial plane which perpendicularly intersects the axis of rotation) of the Earth. The change of latitude is expressed as:

$$\dot{\lambda} = \frac{V^e}{(N+h)\cos\varphi} \tag{2.26}$$

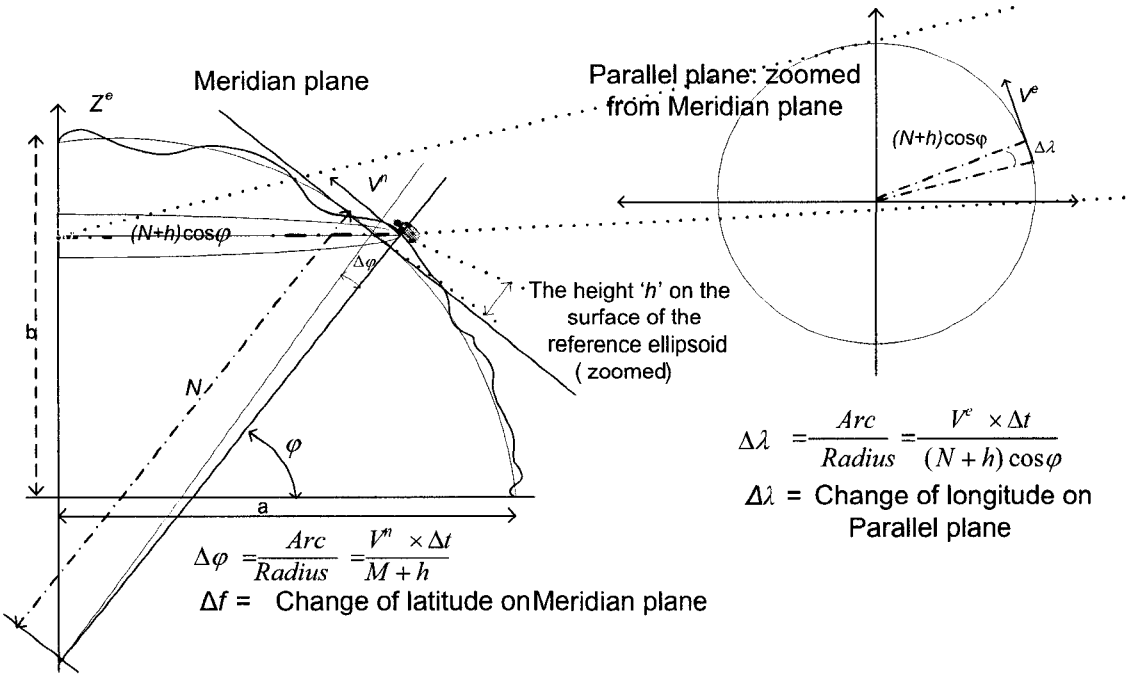

Figure 2.15: Illustration of the change of latitude on the parallel plane and change of longitude on meridian plane.

Here, φ, λ and h are the geodetic latitude, longitude and altitude. The figure 2.7 clarifies the difference between geocentric latitude and geodetic latitude. Geodetic latitude (φ) at a point on the surface of the Earth is the angle between the equatorial plane and a line passing through centre of Earth and the surface location point. In contrast, geodetic latitude at a point on the surface of the Earth is the angle between the equatorial plane and a line normal to the reference ellipsoid which passes through the point [11].

The calculation of the change of position of a moving body is performed based on the velocities in each direction. Once the velocities are known, the position can be

calculated accordingly. In matrix form the equation is expressed as the following, where D^{-1} is a 3 by 3 matrix whose non zero elements are functions of latitude (φ), height (h) and the Earth's radiuses (N and M):

$$\dot{r}^l = \begin{bmatrix} \dot{\varphi} \\ \dot{\lambda} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \frac{1}{(N+h)\cos\varphi} & 0 & 0 \\ 0 & \frac{1}{M+h} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V^n \\ V^e \\ V^u \end{bmatrix} = D^{-1}V^l \tag{2.27}$$

Finally, the 3D Mechanization equations in local level frame that represent the change in position, the change in velocity and the change in attitude can be summarized in the figure 2.16 (modified and reproduced [5]) following matrix form equation [4]:

$$\begin{bmatrix} \dot{r}^l \\ \dot{v}^l \\ \dot{R}_b^l \end{bmatrix} = \begin{bmatrix} D^{-1}V^l \\ R_b^l f^b - (2\Omega_{ie}^l + \Omega_{el}^l)V^l + g^l \\ R_b^l(\Omega_{ib}^b - \Omega_{il}^b) \end{bmatrix} \tag{2.28}$$

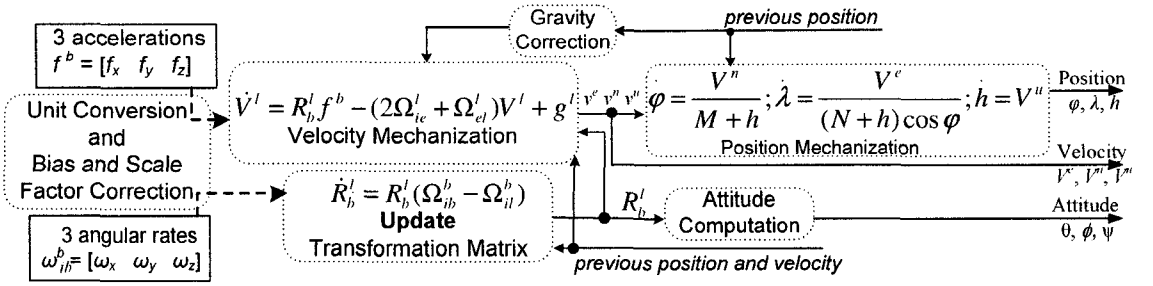

Figure 2.16: Block diagram of the procedures of 3D Mechanization.

The input to these mechanization equations are the accelerometer and gyro measurements (expressed as f^b and Ω_{ib}^b in the above equation). Thus the navigational function is fulfilled by combining the measurements of vehicle rotation and specific force (acceleration) with knowledge of the gravitational field to compute estimates of attitude, velocity and position with respect to a pre-defined reference frame [11].

2.9 INS Error Equations

The *navigational solution* of the INS tends to deteriorate in the long term mostly due to the mathematical integration involved in the INS mechanization procedure

presented in the previous section. To improve this situation, a dynamic error model is used that includes position coordinate, velocity component and attitude component errors (resulting form the INS mechanization procedure) and *stochastic error models* of IMU sensors (for analysis and estimation of different unaccounted error sources). These errors are variable in time and can be modeled by differential equations.

The mechanization equations discussed in the previous section provide no information about the errors of the system as they process data received from the IMU to obtain updated navigation parameters. To estimate the system errors in order to improve performance to a satisfactory level, the system (shown in equation 2.24) is perturbed.

The linearization approach of differential error equations used in this section is the perturbation representation for position, velocity and attitude errors to obtain dynamic error equations [12]. In addition, *stochastic modeled* bias drift of IMU sensors is a vital part of the dynamic error model. The first order Gauss-Markov is the most commonly used *stochastic error model* due to its simplicity. Thus, the dynamic error state vector for any INS system can be expressed as:

$$x = [\delta_\varphi \;\; \delta_\lambda \;\; \delta_h \;\; \delta_{V^e} \;\; \delta_{V^n} \;\; \delta_{V^u} \;\; \delta_\theta \;\; \delta_\phi \;\; \delta_\psi \;\; \delta_{\omega_x} \;\; \delta_{\omega_y} \;\; \delta_{\omega_z} \;\; \delta_{f_x} \;\; \delta_{f_y} \;\; \delta_{f_z}]^T \qquad (2.29)$$

Here, the prefix δ implies the variable is an error value.

δ_φ, δ_λ and δ_h are coordinate errors.

δ_{V^e}, δ_{V^n} and δ_{V^u} are velocity errors.

δ_θ, δ_ϕ and δ_ψ are attitude errors.

δf_x, δf_y and δf_z are accelerometer measurement errors.

$\delta\omega_x$, $\delta\omega_y$ and $\delta\omega_z$ are gyro measurement errors.

Using a Taylor series expansion to a 1st order approximation, neglecting the higher order terms and simplifying several factors so that only the most significant error components were included, the time derivative of the errors can be obtained. The error equations are expressed as follows.

2.9.1 Coordinate errors

The rate of change of position components is defined as: $(\delta \dot{r}^l) = \dot{r}'^l - \dot{r}^l = \frac{\partial(\dot{r}^l)}{\partial r^l}\delta r^l$. Applying Taylor series expansion linearization on position mechanization equation and afterwards neglecting higher order terms (i.e. the terms which involve dividing the velocity components by the large Earth radius value), coordinate errors can be expressed as [4]:

$$\delta \dot{r}^l = \begin{bmatrix} \delta\dot{\varphi} \\ \delta\dot{\lambda} \\ \delta\dot{h} \end{bmatrix} = \begin{bmatrix} \frac{1}{(N+h)\cos\varphi} & 0 & 0 \\ 0 & \frac{1}{M+h} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta V^n \\ \delta V^e \\ \delta V^u \end{bmatrix} \tag{2.30}$$

2.9.2 Velocity errors

By applying the linearization and the first order approximation criterion on the velocity mechanization equation (equation 2.24), the velocity error becomes:

$$\delta \dot{V}^l = \delta R_b^l f^b + R_b^l \delta f^b - (2\Omega_{ie}^l + \Omega_{el}^l)\delta V^l - (2\delta\Omega_{ie}^l + \delta\Omega_{el}^l)V^l + \delta g^l \tag{2.31}$$

Neglecting higher order terms (i.e. 1st order approximations), velocity errors can be expressed as:

$$\delta \dot{V}^l = \begin{bmatrix} 0 & f_u & -f_n \\ -f_u & 0 & f_e \\ f_n & -f_e & 0 \end{bmatrix} \begin{bmatrix} \delta\theta \\ \delta\phi \\ \delta\psi \end{bmatrix} + R_b^l \begin{bmatrix} \delta f_x \\ \delta f_y \\ \delta f_z \end{bmatrix} \tag{2.32}$$

2.9.3 Attitude errors

Ignoring the insignificant error terms (terms involving velocity components divided by the Earth radius and the Earth rotation rate components), the attitude error term becomes [4]:

$$\begin{bmatrix} \delta\dot{\theta} \\ \delta\dot{\phi} \\ \delta\dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{M+h} & 0 \\ \frac{-1}{N+h} & 0 & 0 \\ \frac{-\tan\varphi}{N+h} & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta V^e \\ \delta V^n \\ \delta V^u \end{bmatrix} + R_b^l \begin{bmatrix} \delta\omega_x \\ \delta\omega_y \\ \delta\omega_z \end{bmatrix} \tag{2.33}$$

2.9.4 Accelerometer bias errors

After removing the *deterministic* bias and scale factor of the IMU sensors (accelerometers and gyros), the residual part is modeled *stochastically* as a first order Gauss Markov process. The accelerometer random errors are usually correlated in time and expressed as follows [4]:

$$\dot{\delta f}^{b} = \begin{bmatrix} \dot{\delta f}_x \\ \dot{\delta f}_y \\ \dot{\delta f}_z \end{bmatrix} = \begin{bmatrix} -\alpha_x & 0 & 0 \\ 0 & -\alpha_y & 0 \\ 0 & 0 & -\alpha_z \end{bmatrix} \begin{bmatrix} \delta f_x \\ \delta f_y \\ \delta f_z \end{bmatrix} + \begin{bmatrix} \sqrt{2\alpha_x \sigma_{ax}^2} \\ \sqrt{2\alpha_y \sigma_{ay}^2} \\ \sqrt{2\alpha_z \sigma_{az}^2} \end{bmatrix} w(t) \tag{2.34}$$

Here, α_x, α_y and α_z are the reciprocals of the time correlation parameters of the random processes respectively associated with the acceleration measurements f_x, f_y and f_z. On the other hand, σ_{ax}, σ_{ay} and σ_{az} are the standard deviations of these processes associated with the gyro measurements. $w(t)$ is the unity variance Gaussian noise.

2.9.5 Gyroscope's drift errors

The gyroscope random errors are usually correlated in time and modeled as a first order Gauss Markov processes as follows [4]:

$$\delta\dot{\omega}^{b} = \begin{bmatrix} \delta\dot{\omega}_x \\ \delta\dot{\omega}_y \\ \delta\dot{\omega}_z \end{bmatrix} = \begin{bmatrix} -\beta_x & 0 & 0 \\ 0 & -\beta_y & 0 \\ 0 & 0 & -\beta_z \end{bmatrix} \begin{bmatrix} \delta\omega_x \\ \delta\omega_y \\ \delta\omega_z \end{bmatrix} + \begin{bmatrix} \sqrt{2\beta_x \sigma_{gx}^2} \\ \sqrt{2\beta_y \sigma_{gy}^2} \\ \sqrt{2\beta_z \sigma_{gz}^2} \end{bmatrix} w(t) \tag{2.35}$$

Here, β_x, β_y and β_z are the reciprocals of the time correlation parameters of the random processes respectively associated with the gyro measurements ω_x, ω_y and ω_z. On the other hand, σ_{gx}, σ_{gy} and σ_{gz} are the standard deviations of these processes associated with the gyro measurements. And, $w(t)$ is a unity variance Gaussian noise.

2.10 GPS/INS data fusion using KF

The 2D or 3D Mechanization solution using low-cost IMU sensor data is not useful in the long run (during a GPS outage) since sensor errors and the fixed-step integration errors in mechanization computation (resulted mostly from mathematical integration process) cause the solution to diverge. The most common algorithm used to integrate or fuse GPS and INS is KF that by accounting for these errors improves the

navigation solution. It operates with a set of mathematical equations and recursively processes the noisy measurements to compute estimates.

2.10.1 KF based GPS/INS Integration Schemes

KF can be implemented in different ways such as *loosely*, *tightly* and *ultra-tightly* coupled integration and can be implemented either in a *closed loop* or *open loop* framework [4]. In *tightly coupled* and *ultra tightly* coupled scenario GPS clock timing, *pseudoranges*, phase rate (Doppler) or carrier phase measurements are blended with the navigation solution generated by the IMU sensors [10]. Furthermore, tight integration provides a means for implementing a more sensitive fault detection and isolation. Navigational applications which use carrier-phase output for attitude determination and carrier-phase positioning benefit from *tightly coupled* integration [5].

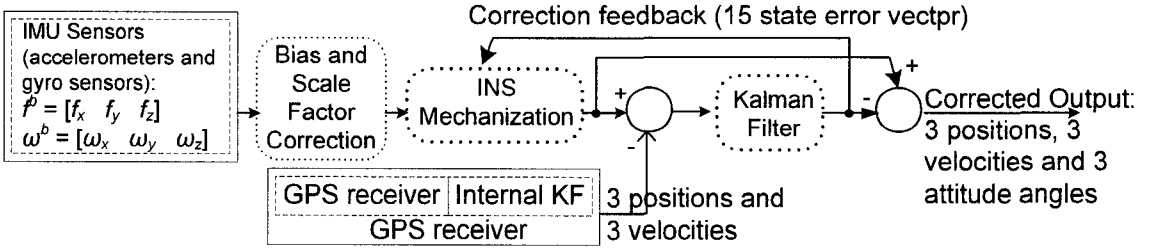

Figure 2.17: Decentralized and closed loop GPS aided SINS KF architecture.

In contrast, in a *loosely coupled* system, the GPS/INS integration KF uses GPS-derived position and velocity (the modified output of a GPS receiver) as a measurement instead of GPS-derived pseudo-range, phase and phase-rate. Loosely coupled integration treats GPS and INS as individual navigation systems, combining the two at the navigation solution level (3 positions and 3 velocities). The loosely-coupled filtering approach has been chosen in this thesis (for automobile navigational application) due to its modularity, smaller filter size, flexibility and simplicity [5]. For a loosely coupled system there are two separate and independently operating KFs, one for GPS (internal, as it comes with a GPS receiver) and the other for INS as shown in figure 2.17 (adapted from [17]). It is also referred to as "GPS aided INS" because it treats the outputs of the internal GPS KF as independent measurements.

Open loop KF approach does not account for the estimation of sensor errors. In contrast, in *closed loop* KF, IMU sensor errors are *stochastically* modeled (as mentioned in the previous section) and are fed back to correct the measurements. (as shown in figure 2.17) for the corresponding errors before being computed and fetched for the next epoch's/sampling interval's computation [17]. *Open loop* KF approach is used for high end expensive IMU sensors with low sampling rate.

2.10.2 KF Models for GPS/INS Integration

KF is used for optimally estimating the error state of the GPS/INS system for which the measurements are corrupted by variable noise. It uses a variable known as Estimated State Vector (x) that can contain three position errors, three velocity errors, three attitude errors and augmented by accelerometer and gyro sensors errors. Thus, a dynamic model with 15 error states of INS is applied for this kind of KF based integration in this thesis [18]. As shown in the previous section ("2.9. INS Error Equations") of this thesis, the error state vector x for navigational parameters is expressed as:

$$x = [\delta_\varphi \;\; \delta_\lambda \;\; \delta_h \;\; \delta_{V^e} \;\; \delta_{V^n} \;\; \delta_{V^u} \;\; \delta_\theta \;\; \delta_\phi \;\; \delta_\psi \;\; \delta_{\omega_x} \;\; \delta_{\omega_y} \;\; \delta_{\omega_z} \;\; \delta_{f_x} \;\; \delta_{f_y} \;\; \delta_{f_z}]^T \tag{2.36}$$

Another important variable used in KF is called Error Covariance Matrix (P). It is a measure of estimation uncertainty which take into consideration how the sensor noise and dynamic uncertainty contribute to the uncertainty of the estimated system state. By maintaining an estimate of its position, velocity and attitude output estimation uncertainty and the relative uncertainty in the sensor outputs, the KF is able to optimize the estimate to minimize the estimation error. It combines the estimate with the measurement using a variable called Kalman Gain (K_k) [17]. KF is essentially a set of mathematical equations which has a prediction stage and an update stage [19].

a) The Prediction Stage

The Estimated State Vector $\hat{x}$ (shown in equation 2.33) and its covariance matrix of estimation uncertainty P propagate from one time step to the next in this stage [17].

A linear model in discrete time can be shown as:

Prediction of error states: $\hat{x}_k^- = F_{k,k-1}\hat{x}_{k-1}$ (2.37)

Prediction of error covariance: $P_k^- = F_{k,k-1}P_{k-1}F_{k,k-1}^T + G_{k-1}Q_{k-1}G_{k-1}^T$ (2.38)

Here, $F_{k,k-1}$ is the state transition matrix, it relates the state from the previous step (denoted by subscript k-1) with present step (denoted by subscript k).

Q_{k-1} is the system noise covariance matrix given by $Q_k = E\left[w_k w_k^T\right]$

G_{k-1} is the system noise coefficient matrix which represents how the system noise w is distributed among the INS error state components.

P_k^- is the priori estimate of the covariance matrix for the estimate of the error state vector x where: $P_k = E\left[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^T\right]$

In most KF implementations the sensor noise portion of this State Transition Matrix is based on a 1st order Gauss Markov model (a *stochastic* model) as shown in equation 2.34 and equation 2.35.

The state transition matrix $F_{k,k-1}$ can be obtained using the following Taylor expansion approximation equation:

$F_{k,k-1} = \exp(F\Delta t) \approx \mathrm{I} + F\Delta t$ (2.39)

b) The Update Stage

The update stage corrects the Estimated State Vector $\hat{x}_k^-$ using new measurements. This is carried out in the following equation:

$\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H\hat{x}_k^-\right)$ (2.40)

The Kalman Gain Matrix K_k is the optimal weighting matrix for combining new GPS data with priori estimate $\hat{x}_k^-$. Here, the size of Kalman Gain Matrix (K_k) is 15 (for each Error State) by 6 (for each value of Design Matrix Variable). Mathematically it is expressed as the following equation:

$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$ (2.41)

GPS position and velocity measurement update is given in following equation:

$$z_k = \begin{bmatrix} \varphi_{INS} - \varphi_{GPS} & \lambda_{INS} - \lambda_{GPS} & h^{INS} - h^{GPS} & V^e_{INS} - V^e_{GPS} & V^n_{INS} - V^n_{GPS} & V^u_{INS} - V^u_{GPS} \end{bmatrix}^T \tag{2.42}$$

The Measurement Vector (z_k) of update stage can also be represented in the following form as it is the difference between the INS and GPS position and velocity:

$$z_k = H_k x_k + v \tag{2.43}$$

Here in equation 2.37 and 2.40, H_k is the Design Matrix which provides ideal noiseless relationship between the measurement vector z_k and the INS Error State Vector $\hat{x}_k$. The Matrix (H_k) is designed with the number of common parameters between INS and GPS (six in total: φ, λ, h, V^e, V^n and V^u). The number of rows is equal to the number of common parameters and the number of columns is equal to the length of the error state vector $\hat{x}$.

The Error Covariance Matrix (P) has the size 15x15 (where the initial values only on the diagonals) is updated in this stage with the following equation:

$$P_K = (I - K_k H_k) P_k^- \tag{2.44}$$

The Measurement Error Covariance Matrix (R_k) is one of the parameters of KF that needs to be tuned for successful KF implementation. It is expressed as:

$$R_k = \begin{bmatrix} \sigma_\varphi^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_\lambda^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_h^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{v^e}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{v^n}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{v^u}^2 \end{bmatrix} \tag{2.45}$$

The update procedure of KF is implemented at a lower rate than the prediction because IMU sensors provide data at a much higher rate than GPS (50 to 200 Hz vs. 1 Hz). GPS-derived accurate positions and velocities are excellent external measurements for updating the INS, thus improving the applications' long-term accuracy. The INS in an integrated GPS/INS system is responsible for interpolating position between updates (short-term positioning component) as well as providing attitude information [4]. This phenomenon has been illustrated at figure 2.18 (reproduced from [4]). When a GPS

outage is detected the KF runs only the prediction stage. As, the GPS signal returns, it resumes the update stage.

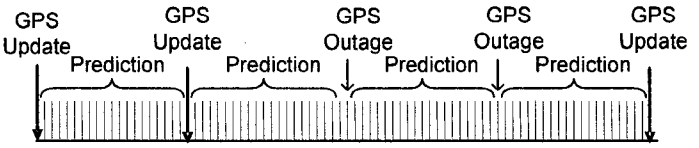

Figure 2.18: Illustration of GPS/INS data sampling and KF (in prediction and update mode).

Readers are encouraged to go through the reference [12] for more details on KF theory, its application and related issues.

2.10.3 Limitations of KF

KF only works well under certain predefined models and sensor outputs that fit properly. It is difficult to set the initial values for some parameters involved. An accurate stochastic model for IMU sensors (accelerometer and gyro) that works in all cases is also difficult to set. The performance of the inertial sensors is directly proportional to the sensor cost as the sensors do not conform to traditional error models [17].

The low-cost tactical or MEMS-based IMU sensors have very high and varying drift rates. As a result, in the absence of GPS update measurement, the KF based integration solution using a lower grade INS can degrade (during the prediction stage) dramatically over time and follow the same exponential error trend as the mechanized (without the benefit of KF) INS data [5, 6].

The measurement covariance matrix (R_k) is of great importance to determine the KF output. It determines how good the measurement model derived from the measurements by IMU sensors are. The shortcoming of the KF model for GPS/INS integration is usually due to non-modeling, mismodeling or ignoring one of the correlated noises of IMU sensors.

Another challenge is to set the initial values for the system noise covariance matrix $Q_{k,k-1}$ as it cannot be set to zero due to the inaccurate measurements (with noise) of accelerometers and gyros. The role of this covariance matrix Q is to define the width of the uncertainty after each step. In other words, it determines to what extent the predictions by the KF can be trusted. A large value of Q enlarges the uncertainty and

results in a noisy estimate while a smaller Q results in a smoother estimate. A correct value of Q is critical for achieving practically sound KF outputs as it depends on factors such as system dynamics and sensor noise level. Thus the requirements to tune KF parameters (Q and R_k) pose a challenge in GPS/INS integration [19].

2.11 IMU data preprocessing using Wavelet De-noising

The accuracy enhancement of IMU sensors to improve the navigation solution is an important field of research [5][6]. The short-term (high frequency) errors resulting from the INS mechanization process (described in section 2.9 of this thesis) section can be removed through signal processing techniques such as low-pass filtering (LPF). This is generally effective to reduce errors with frequencies above the true motion dynamics bandwidth. Attempting to remove noise in the true motion dynamics bandwidth with a LPF runs the risk of compromising the measurements of the actual vehicle motion [5]. Wavelet de-noising is based on Wavelet Multi Resolution Analysis (WMRA) which uses the Discrete Wavelet Transform (DWT). WMRA, DWT and their effectiveness on signal analysis and de-noising over LPF technique and for other techniques constitute a vast field of subject matter [20]. For simplicity, only a real-time implementation of this technique used on IMU sensor data will be discussed in this section.

The DWT of a discrete time sequence $x(n)$ is expressed as [6]:

$$C_{j,k} = 2^{(-j/2)} \sum x(n) \psi(2^{-j} n - k) \tag{2.46}$$

Where,

ψ (n)	is the wavelet function
$2^{(-j/2)} \psi(2^{-j} n - k)$	is the scaled and shifted version of ψ(n) based on i and j
j and k	scaling and shifting coefficients which are always an integer

For different scaled and shifted versions of $\psi(n)$, $C_{j,k}$ represents corresponding wavelet coefficients. The original signal $x(n)$ can be generated from the corresponding wavelet function [20]:

$$x(n) = \sum_j \sum_k C_{j,k} \psi_{j,k}(n) \tag{2.47}$$

The WMRA can be implemented using a bank of half-band low pass and half-band high pass discrete time filters. The low-pass portion contains the low frequency components of the signal, which are known as the *approximations*. The high pass portion contains the high frequency components of the signal, which are known as the *details*. The *approximations* and *details* are each down sampled to half the number of points. With enough levels of de-composition, high frequency noise components (i.e. white noise) can be separated from the signal. There are essentially three principle steps in wavelet de-noising. They are: (1) De-composition of the signal with a wavelet basis function to a chosen level; (2) Threshold the details coefficients at each level; (3) Reconstruct the signal using the thresholded wavelet coefficients [6] [20].

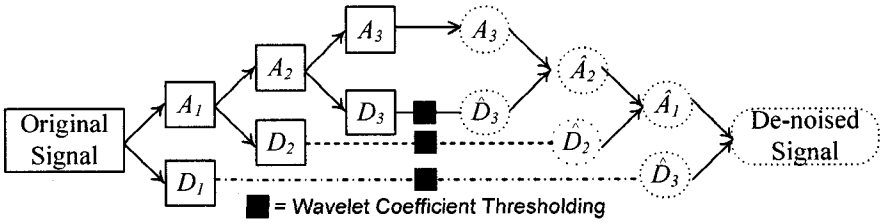

Figure 2.19: Illustration of the three steps of Wavelet De-noising Procedure a) de-composition, b) thresholding and c) reconstruction.

To reduce the impact of short term (high frequency) INS sensor errors, the bandwidth of true motion dynamics are identified by spectrum analysis and a de-noising algorithm is applied. The mathematical procedure involved for the implementation of WMRA is illustrated in the following sub-sections:

2.11.1 Signal De-composition and Reconstruction

As shown in figure 2.19 (modified and reproduced from [6]), the WMRA builds a pyramidal structure or an iterative application during signal de-composition. The low pass filter and the high pass filter initially act on the entire band and gradually reduce the signal band at each stage. In the figure, the high frequency band outputs are the detail coefficients ($D1$, $D2$ and $D3$) and the low frequency band outputs are the approximation coefficients ($A1$, $A2$ and $A3$). For an input signal $x(n)$, the approximation coefficient $a_{j,k}$ at the j^{th} resolution is [6] [20]:

$$a_{j,k} = 2^{(-j/2)} \sum_{n} x(n)\phi(2^{-j}n - k) \tag{2.48}$$

Here, $\phi(n)$ is called the scaling function. The scaling function is similar to the wavelet function except that they have only positive values. It smoothes the input signal and operates in a manner equivalent to low pas filtering. The *approximation* of $x(n)$ at the jth level can be computed as [6] [20]:

$$x_j(n) = \sum_{k=-\infty}^{\infty} a_{j,k}\phi_{j,k}(n) \tag{2.49}$$

The *details* coefficient $d_{j,k}$ at the jth resolution level and detail signal $g_j(n)$ are then computed:

$$d_{j,k} = \sum_{n} x(n)\psi_{j,k}(n) \tag{2.50}$$

$$g_j(n) = \sum_{k=-\infty}^{\infty} d_{j,k}\psi_{j,k}(n) \tag{2.51}$$

Here, $\psi_{j,k}(n)$ is the wavelet *basis function*. The above steps (equations 2.48 to 2.51) are repeated for the j+1 resolution level using the approximation signal $x_j(n)$.

The original signal $x(n)$ can be reconstructed using all the detail coefficients obtained during the de-composition process (starting from the first de-composition level until the last level) and the approximation coefficients of the last resolution level. The following equation illustrates this (where the de-composition was done till Jth resolution level)[6] [20]:

$$x(n) = \sum_{k=-\infty}^{\infty} a_{J,k}\phi_{J,k}(n) + \sum_{j=1}^{J}\sum_{k=-\infty}^{\infty} d_{j,k}\psi_{j,k}(n) \tag{2.52}$$

Here, in the above equation, the first term represents the approximation coefficient at level J. The second term represents the detail coefficients at resolution level J and lowers [6] [20].

Each time the signal is passed through a set of filter banks it is said to have a de-composition level of one. In figure 2.19, a level of de-composition (LOD) of 3 is shown as the original signal has passed through 3 banks of discrete filters. These filters initially act on the entire signal band at the small scale values (i.e. at the higher frequencies). Gradually the band size reduces in each de-composition level/stage. In the same figure,

the low-scale or the high frequency band outputs are presented by the detail coefficients (D_1, D_2 and D_3) and the high-scale or the low frequency band outputs are presented by the approximation coefficients (A_1, A_2 and A_3).

2.11.2 Wavelet Coefficient Thresholding

Thresholding operation is crucial on the detail coefficients of the (wavelet) decomposed signal to ensure the effective cancellation of the interference of noisy signal with a minimum distortion to the sharp transition details of the true signal. Unlike, signal de-composition and reconstruction of DWT, it is a nonlinear operation. Two thresholding operators, the soft and the hard are proposed by Donoho, D. [21]. In hard thresholding procedure, any wavelet coefficient with an absolute value below the threshold is replaced by zero and coefficients with an absolute value above the threshold are kept the same. In soft thresholding procedure, any wavelet coefficient with an absolute value below the threshold is replaced by zero as like the hard threshold procedure. But the coefficients with a magnitude above the threshold are reduced or shrunk in value by the threshold value. These procedures are expressed in the following table (after [21]):

Table 2.3: Wavelet coefficient (hard and soft) thresholding equations.

Hard thresholding equation	$th(y) = \begin{cases} y & if \quad \lvert y \rvert > T \\ 0 & if \quad \lvert y \rvert \le T \end{cases}$
Soft thresholding equation	$th(y) = \begin{cases} y - T.sign(y) & if \quad \lvert y \rvert > T \\ 0 & if \quad \lvert y \rvert \le T \end{cases}$

Here, T is the threshold value. The same value of T can be applied to *detailed* coefficients of every de-composition level assuming the original signal is affected by a *white* noise. Here, *sign*() represents the sign operator. T is selected according to the standard deviation of the Gaussian noise (σ) affecting the original signal and the length of the observations/samples of the original signal (N) [6][21].

$$T = \sigma\sqrt{2\log N} \tag{2.53}$$

The standard deviation can be estimated from the median of its finest scale wavelet coefficients, provided that $x(t)$ is piecewise smooth [6][21]:

$$\sigma \approx \frac{1}{0.6745} Med\left(|x(t)|\right) \tag{2.54}$$

$\text{Med}\left(|x(t)|\right)$ is the median value of the sequence $x(t)$ which is actually the *detail* coefficients of the finest resolution level.

Now, in case of the noisy components of the original signal are not *white* (*colored* signal which is mostly the case with raw data of IMU sensors), the Gaussian noise (σ) is estimated adaptively in each level of de-composition.

$$T = \lambda_j \sigma_j \sqrt{2\log(N/2^j)} \tag{2.55}$$

Here, j is the de-composition level and λ_j is a level dependent relaxation factor. In the orthogonal wavelet de-composition, the *detail* coefficients decrease with the increase of levels, λ_j is used to achieve a balance between the cancellation of noisy components and distortion to the signal details. A simpler version of the soft thresholding technique assumes λ_j to be equal to 1. In each level of de-composition, the number of detail coefficients is halved (constant N divided by 2^j where j is the de-composition level number) [21].

2.12 Chapter Summary

This chapter presented the principles of GPS operation and errors related to vehicle navigation. It then focuses mainly on Inertial Navigation System using IMU sensors suited for land based navigation. It defined inertial navigational equations by introducing the related theories/concepts and mathematical notations in the sections 2.2 to 2.6. The mathematical equations and operations/techniques (mechanization, KF and de-noising) presented in the later part of the chapter (from the section 2.7 to 2.10) are thus based on these basic concepts and mathematical notations. Then in the following chapter (chapter 3), a functional NCU has been built on an embedded platform using IMU sensor data and GPS receiver data.

In summary, a minimally functional NCU using IMU sensor data and off-the-shelf GPS receiver executes these techniques in the following order as presented in this chapter and as have been chosen to implement on a low cost embedded platform in order

to provide a navigational solution in conjunction to an existing GPS navigation technology (during a GPS outage) on a land vehicle:

a) Pre-processing of low-cost IMU sensor data:

- Wavelet de-noising.

b) INS mechanization:

- Correction of raw data for known or estimated errors (bias and scale factor correction).
- Attitude update.
- Transformation of specific force to navigation frame of interest.
- Calculation of velocity and position.

c) GPS/INS data fusion using 15 state KF:

- Calculate the INS error state equation
- Using the error equations, KF model equations are used for prediction and update (if GPS data is available) stages to enhance the reliability of the INS solution.

CHAPTER 3: Implementation Methodology

Chapter 3 begins with short summaries of the embedded system platform's components i.e. FPGA chip, the corresponding board and the Xilinx MicroBlaze soft core processor (SCP) that runs on its fabric— used in the implementation. Afterwards, the design and development of the hardware and the software components for implementation are detailed. The IMU sensor data set used to establish a navigational application implementation is presented in this context.

3.1 Hardware/Equipment Setup

Figure 3.1 shows the hardware/equipment setup in the form of connection between PC and the FPGA board that has been used to do the experiment to emulate the implementation methodology. The parallel cable was used to download the bitstream of FPGA configuration (generated by Xilinx EDK software tool on the PC) to the FPGA through its JTAG interface. The Serial cable was used to establish the data communication between the PC and the serial port of the FPGA.

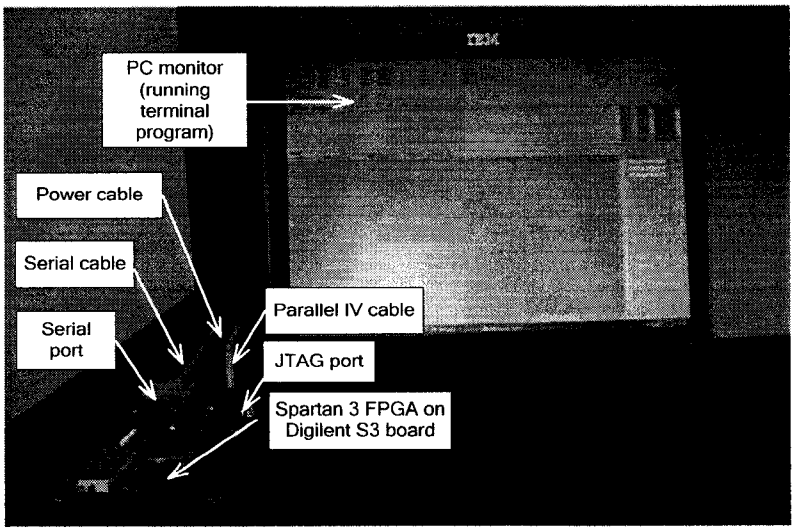

Figure 3.1: A snapshot of the hardware/equipment setup for the implementation.

3.1.1 Development Boards

An FPGA (Field Programming Gate Arrays) is a semiconductor device containing programming logic components called "logic blocks" as they are

interconnected through horizontal and vertical routing channels. FPGAs are used for prototyping any design as they offer great flexibility in design procedure.

The Spartan-3 Starter Board developed by Digilent Inc. was used primarily in this thesis which provides a development platform containing the Spartan-3 FPGA of Xilinx. It features Spartan-3 XC3S200 FPGA (containing 200 thousands gates) which is clocked by an onboard oscillator operating at 50 MHz, 1MiB asynchronous SRAM, a UART (serial port) interface, a JTAG port, a VGA display port and several other (PS/2 port, expansions slots, dipswitches, pushbuttons etc.) on-board I/O devices. Thus it can hold from a simple logic circuit to an embedded processor core [22]. It ships with a power supply and a programming cable and an adapter to power the board. The hardware designs were downloaded to the FPGA via the onboard JTAG interface.

Even though, the board comes with lots of features and peripherals most of the resources were not used as illustrated in figure 3.2. A final navigation solution on a platform with just only three hardware components of the board as shown in figure 3.2 is expected to be cost-efficient.

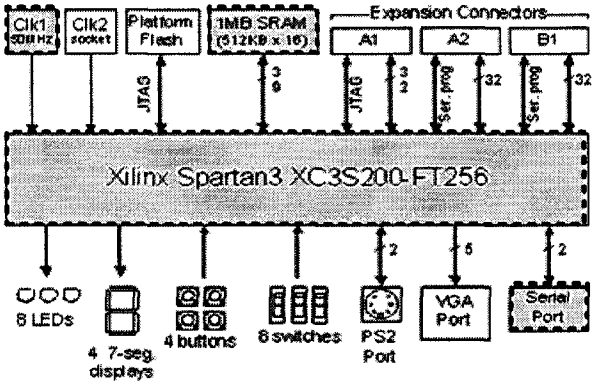

Figure 3.2: Block diagram of S-3 Board (resources used have been shaded)

Apart from the Spartan-3 XC3S200 FPGA chip of the Spartan-3 Starter board, the following onboard features are used in the implementation:

- 1 MiB of SRAM for storing programs.
- UART interface for output.
- JTAG interface to download the bitstream.

At the later stage of the implementation, in order to make the code run faster, Xilinx University Program (XUP) Virtex-II Pro Development Board was used as the FPGA chip with more high-speed block ram (BRAM) memory resource. It is an advanced hardware platform (compare to Spartan-3 Starter board) consisting of a high performance Virtex-II Pro FPGA surrounded by a collection of peripheral components. In this thesis, apart from its FPGA chip resources (specifically its increased size of BRAM), the serial port, the JTAG port and the digital clock are used [23].

3.1.2 Serial Cable

Serial port interface is simple, low-cost and comparatively low-speed interface to establish data communication between PCs' (also known as *terminal* equipment) and other devices (also known as *communication* equipment). Two RS-232 transceivers/voltage level translators are available in the S-3 board. One (labeled J2 on the S-3 board) connects directly to the FPGA chip by way of 9-pin header of female DB9 connector. It can be accessed by simply implementing a UART in the FPGA fabric. A standard *straight-through* serial cable (male-female) is used to connect the FPGA board (which has female DB9 connector) to the PC's serial port known as COM1 or COM2 (which has male DB9 connector) [22].

3.1.3 Parallel Cable IV

A Xilinx Parallel Cable IV (known as "Digilent Low-Cost Parallel Port to JTAG Cable") is used to connect the parallel port of a PC to the JTAG port of the S-3 board. This single cable comes with the S-3 board and allows users not only to download the bitstream from the PC through parallel port to the FPGA chip through the JTAG interface but also to debug FPGA implementation. In the same way, to configure the XUP Virtex-II Pro Development System externally through USB JTAG interface the embedded Platform Cable USB (supplied with the board) is used. Readers are encouraged to consult references [22] and [23] for more details on FPGA, serial port and parallel port.

3.1.4 Terminal Program

The application "Hyper Terminal" and/or "Terminal V1.9B" for Windows were used to retrieve data sent from the S3 and V-II Pro boards through the serial cable.

The terminal program lets a computer (PC) connect to other computer, internet telnet sites and host computers using a modem, a null modem serial cable or a network connection. In the implementation, both Hyper Terminal (comes with included with Microsoft Windows OS) and/or Terminal V1.9B (a freeware) was used as the interface to the FPGA board. Through the serial cable connecting to the COM1 port of the PC Data can be sent to the FPGA and receive back and display them on the Hyper Terminal. The terminal program settings for use with these projects are given in the table below.

Table 3.1: Terminal program settings used for retrieving data from FPGA boards.

Baud rate	115200 bps
Data bits	8 bits
Parity	None
Stop bits	1 bit
Flow control	None

3.2 Embedded Platform

This section starts with a description of the tool used (to implement the embedded system) in this thesis namely Embedded Processor Development Kit (EDK) developed by Xilinx. Before downloading the bitstream, it synthesizes the microprocessor hardware design, maps it to target FPGA chip and generates the bitstream. A brief description of MicroBlaze soft core processor (SCP) is then presented with its different bus connectivity and functionality. Finally, the different peripherals that were used to develop navigation algorithms on MicroBlaze processor have been described. Readers are encouraged to consult references [24] and [25] for more details.

3.2.1 Development tool Xilinx EDK

Xilinx Embedded Development Kit (EDK) is a development environment where the hardware is instantiated as different IP-blocks connected via buses and signals to develop embedded processors such as MicroBlaze and PowerPC. The software is developed on top of the generated libraries of the hardware design. It provides a

framework for designing hardware/software components of embedded processor systems on programmable logic fabric of FPGA. It includes an integrated development environment (IDE) with a GUI named Xilinx Platform Studio (XPS). It is used to design a complete embedded processor system for implementation specifically on Xilinx FPGA device [25].

The EDK contains the following tools [25][27].

Xilinx Platform Studio (XPS) Tool Suite which is a graphical IDE and command line support for developing and debugging HW/SW platforms for embedded applications. By compiling the software and implementing the hardware, XPS acts as a graphical front-end. It also has design wizards to configure the embedded system architecture, buses and peripherals

Software Development Tools for MicroBlaze and PowerPC which include GNU C/C++ compiler and debugger, Xilinx Microprocessor Debug (XMD) target server, Data2MEM utility for bitstream loading and updating, Base System Builder (BSB) configuration wizard and Platform Studio SDK (Software Development Kit) software-centric design environment based on Eclipse IDE etc.

The EDK also contains stand alone Board Support Packages (BSPs) for non-RTOS systems of MicroBlaze and PowerPC. It also contains BSPs for different RTOSs namely Wind River VxWorks and Embedded Linux (running on PowerPC) and support for Xilinx MicroKernel (XMK) Systems.

3.2.2 MicroBlaze Soft Processor Core

As mentioned in chapter one, one of the motivations behind using soft processor core (SCP) on a FPGA lies in its low-cost. A MicroBlaze SCP costs US $0.48 whereas any other microprocessor (found inside a PC) costs in the order of hundreds of dollars. Moreover, the FPGA chip price which is below US $2.00 with more than 100K gates system (such as Spartan-3 FPGA) makes it a feasible and cost-effective embedded platform to be used in applications related to automobile navigation [26].

Microblaze microcontroller is an integrated solution intended for implementation of an embedded controller in the FPGA. A *soft core* processor (SCP) like MicroBlaze is

downloaded into the FPGA chip as bitstream format. In contrast to hard core processors like most other processors (IBM's PowerPC, Intel's Pentium etc.) which are real *physical* processors. The MicroBlaze SCP embedded over FPGA fabric (i.e. as the bitstream is downloaded on it, a processor is created out of the configuration logic bloc) is a 32-bit Harvard architecture processor which has physically separate storage and signal pathways for instructions and data with an instruction set provided by Xilinx. The MicroBlaze embedded *soft core* is a Reduced Instruction Set Computer (RISC) optimized for implementation in Xilinx Field Programmable Gate Arrays (FPGAs). MicroBlaze is configurable as it allows the users to select a specific set of features required by their design. The processor's fixed feature set includes 32-bit general purpose registers, 32-bit instruction word with three operands and two addressing modes, 32-bit address bus and single issue pipeline [24].

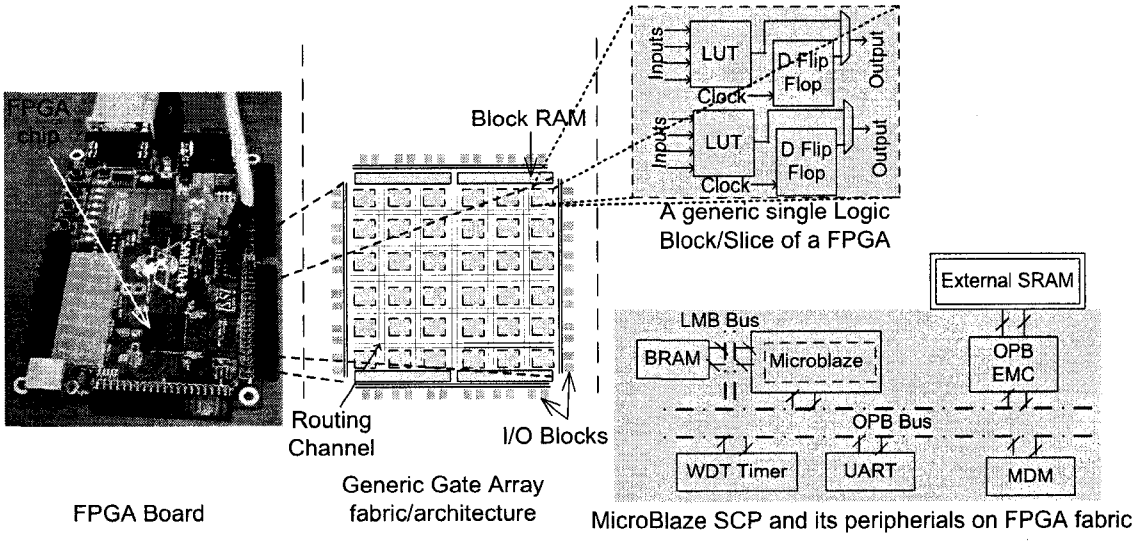

Figure 3.3: An illustration of the embedded platform used in this thesis work.

MicroBlaze supports three interfaces with separate bus interface units for data and instruction accesses. They are Local Memory Bus, On-chip Peripheral Bus and Xilinx Cache Link. Local Memory Bus (LMB) provides single-cycle access to on-chip dual-port block RAM. IBM's On-chip Peripheral Bus (OPB) interface provides a connection to both on-chip and off-chip peripherals and memory. Xilinx Cache Link

(XCL) includes up to 8 Fast Simplex Link (FSL) ports, each with one master and one slave FSL interface [25].

In addition to the static features, the MicroBlaze SCP is parameterized for additional functionalities such as: On-chip Peripheral Bus (OPB) data side interface, On-chip Peripheral Bus (OPB) instruction side interface, Local Memory Bus (LMB) data side interface, Local Memory Bus (LMB) instruction side interface, Hardware barrel shifter, Hardware divider, Instruction cache, Data cache, Hardware debug logic, Fast Simplex Link (FSL) interfaces, Machine status set and clear instructions, Cache Link support, Hardware exception support, Pattern compare instructions, Floating point unit (FPU), hardware multiplier etc. [24]

3.2.3 MicroBlaze Processor peripherals

The EDK tool allows users the ability to connect MicroBlaze with large number of commonly used peripherals available in the EDK peripheral libraries. It also allows users to implement custom peripherals functionality not available in the EDK peripheral libraries. These peripheral devices are connected mainly through its OPB bus and also through LMB bus.

The LMB is a fast local bus with separate read and write data buses. It is a single master bus and unlike IBM's OPB bus it requires no arbiter. It connects MicroBlaze instruction and data ports to high-speed peripherals, primarily BRAMs. The LMB provides single-cycle access to on-chip dual-port block RAM. Separate instruction and data cache units can be enabled. ILMB (Instruction side Local Memory Bus) and DLMB and (Data-side Local Memory Bus) are connected to the dual-port BRAM. This memory is also used for bootloop storage.

SRAM on the S-3 Board is connected through an External Memory Controller (EMC) module with the OPB bus of MicroBlaze processor.

JTAG port was used for transferring the bitstream as well as the ELF (execution and linking format) file containing the software architecture to the SRAM. It is also used for debugging. Microprocessor Debug Module (MDM) on the OPB bus is used for JTAG-based debugging.

A free-running Timebase and Watchdog Timer peripheral is attached to the OPB bus. One of the purposes of using this is to measure/calculate the number of clock cycles required to execute a certain portion of code/instructions on a MicroBlaze platform.

RS232 serial port on the S-3 Board, connected to a UART peripheral on the processor OPB bus of MicroBlaze, are used as standard in and standard out devices.

MicroBlaze SCP has one interrupt port and it can be connected to the peripheral that requires it. An interrupt controller peripheral is required for handling more than one interrupt signal. On interrupts, MicroBlaze jumps to address location 0x10 which is part of the C runtime library and contains a jump to the default interrupt handler. This function is part of the MicroBlaze Board Support Package (BSP), which is provided by Xilinx.

3.3 Hardware Platform Development

"Hardware platform" is a term used by Xilinx to describe the embedded processing subsystem created using EDK tool according to the need of the application being implemented. The hardware platform consists of one or more processors and peripherals connected to the processor buses. EDK captures the hardware platform in the MHS file (Microprocessor Hardware Specification). It allows the users to customize the hardware logic in the processor subsystem [25]

3.3.1 Building processor core

The Base System Builder (BSB) wizard of EDK tool helps users to quickly build a working system targeted at a specific development board. It was used to develop the hardware platform as the target of the software code of navigation application developed in this thesis. Instead of adding any embedded Operation System (embedded OS like XilKernel, ucLinux etc.) on top, a *standalone* platform has been chosen to keep the functionalities simple.

MicroBlaze running at 50 MHz with FPU unit option enabled was chosen as the processor for the hardware platform for Spartan-3 Starter Board. The LMB BRAM interface controllers only support power of 2 sizes (e.g.8KiB, 16KiB, 32KiB, 64KiB, etc.) and the Spartan-3 Starter Board can be configured with up to 16KiB of BRAM. The

navigation application code for (2D, 3D Mechanization, KF and wavelet de-noising) was found to be too large to fit in the on-chip BRAM. The external memory (1 MiB of SRAM) available on the Spartan-3 Starter Board (shown in figure 3.2) for storage of the code (both instruction and data) was included with the hardware platform design. It was connected to the OPB bus of the MicroBlaze processor through an OPB External Memory Controller (EMC) unit. While using the external memory, only 8KiB BRAM resource was used for storing the bootloop information and software stack/heap data and the rest of the BRAM resources were configured for caching the external memory. Up to 4KiB of OPB Instruction Cache and 8KiB of OPB Data Cache were enabled with available BRAM resources of the Spartan-3 Starter Board to accelerate the code execution from the external memory (connected to MicroBlaze through OPB bus).

With the XUP Virtex-II Pro board, the processor was configured with 64KiB BRAM as it successfully accommodated the decentralized KF code (developed in C) within 64KiB BRAM configuration. As a result, no external memory (and of course no caching) was required to configure with the hardware platform. MicroBlaze has been configured with 100.00 MHz clock frequency as it contains faster DCM. Also, the FPU option was enabled. Unlike Spartan-3 Starter board, the BSB wizard of Xilinx EDK does not contain the XUP Virtex-II Pro Board Definition file by default. It was downloaded from the manufacturer Digilent Inc.'s website [23].

RS-232 serial port with UART LITE IP (included with the BSB tool) was configured as STDIN and STDOUT for the MicroBlaze running on both the boards with parameter set according to the PC's terminal program shown in table 3.1. For both boards, hardware debug module was enabled.

3.3.2 Measuring the timing performance

Number of clock cycles spent/used by software code running on MicroBlaze was measured using a watch-dog timer connected to the processor's interrupt port. Software intrusive code profiling is supported by the MicroBlaze and for this GCC/GPROF tools are provided with EDK. But, to enhance the accuracy of the measured clock cycles, a watch-dog was used instead as illustrated in figure 3.4.

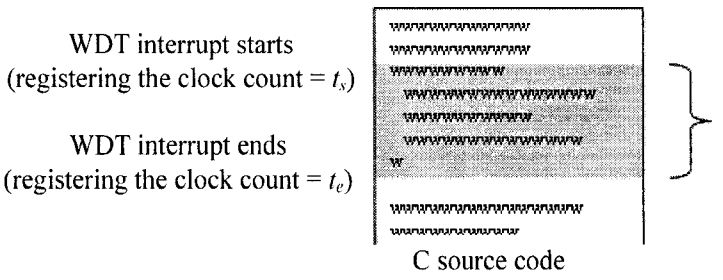

Figure 3.4: Illustration of the measurement of clock cycles (t_e-t_s) to execute a certain portion of C code on MicroBlaze.

Unlike PowerPC, MicroBlaze SCP does not have a time-base register inside the processor that works with the system clock. A watch-dog timer (WDT) was attached like a 32-bit peripheral with the OPB bus that contains a time-base register. While building the processor core using the Base System Builder (BSB) wizard of EDK, the TimeBase WatchDog Timer (TBWDT) was chosen as a peripheral with interrupt option. Thus, an interrupt signal is provided that pulses high for one clock period as the time-base counter rolls over from 0xFFFFFFFF to 0x00000000. Interval length/count bit option for the timer was chosen as 31 bit (the maximum possible length) and the option for "WDT can be repeatedly enabled/disabled via software" was selected.

3.4 Software Coding

In this section, the details associated with the C coding (on MicroBlaze platform) of the navigation algorithms are described.

The software coding of the navigation application (derived from the theoretical understanding/research shown in chapter 2) was carried out mainly in three phases. In the first phase, 2D, KF and wavelet de-noising (applied on IMU raw data) model was created in Matlab environment. To this end, a working 3D Mechanization Matlab model provided by Advanced Navigation and Instrumentation (ANI) Research Group of Royal Military College was used [28]. Then from the Matlab environment, the coding was carried out in Microsoft Visual Studio environment using C programming language. At the final stage of the coding, the debugged and validated code (with respect to the

Matlab models) from Microsoft Visual C environment was ported to the MicroBlaze environment.

As mentioned before, an existing 3D Mechanization Matlab model was supplied by ANI research group [28] along with the GPS and corresponding IMU sensor data in ASCII format. The existing 3D Matlab model was extended to a working KF model that was capable of providing a position error of just above 30 meter in the case of a 20 seconds GPS outage. By tuning and tweaking the initial parameters (such as covariance values of update position and velocities of GPS and the co-variances of the INS error states) this result could have been improved and the created Matlab model could have been validated by measuring its performance during numerous forced GPS outages. Wavelet de-noising and thresholding algorithm was developed in C and the plots showing de-noised IMU data (operating in real-time mode) were obtained (shown in figures 4.10 and 4.14 of chapter 4). However, its improvement in the solution domain could not be validated as the developed KF using de-noised data could not provide better result. As the data set for the KF was changed (from noisy data to de-noised data), the KF parameters were changed as well. The parameters could not be successfully tuned and tweaked in the way it was done for the noisy data set (of a specific *forced* outage shown in figure 3.7). As mentioned in reference [18] about the limitations of KF: "It requires a human expert to tune the optimal parameters of the Kalman filter (i.e. Q and R matrices). In addition, these parameters are sensor dependent."

As the focus of the thesis is to verify the feasibility of navigation algorithms on embedded platform and analyze its performance, a KF model implementation with an acceptable error margin for a single forced GPS outage is shown in chapter 4. For low end tactical grade, KF should be able to provide 20 m position errors maximum during 20 sec outages [29]. Thus the development of a fully functional KF model working perfectly for any outage (and providing better results on de-noised IMU data) can be considered as a future extension of the research.

TG6000 IMU's gyro's output data are in degree format. To facilitate the trigonometric operation, gyro data were converted from degree to radian. In the same

way, accelerometer sensors' raw data were transformed from gravitational acceleration unit ('*g*' unit) to m/s^2.

The IMU data file was supplied with *bias* and *scale factor* values for each gyro and accelerometer sensor. As illustrated in table 2.2 of the thesis, these values were obtained in a laboratory environment. After unit conversation, the *bias* and *scale factor* convertion was carried out for each sensor's output at every time interval. To illustrate this fact, a module titled **"Bias and Scale factor correction with Unit Conversion"** was added with the block diagrams of 2D, 3D Mechanization and KF (figure 2.10, 2.17 and 2.18 respectively)

The *initial* position and velocity before the GPS outage were obtained using the corresponding GPS position and velocities. The *initial* heading of the vehicle was determined using the incremental GPS latitude and longitude values just before the outage. For simplicity and assuming that vehicle was running on a smooth road surface, the *roll* and *pitch* values were initialized to *zero*.

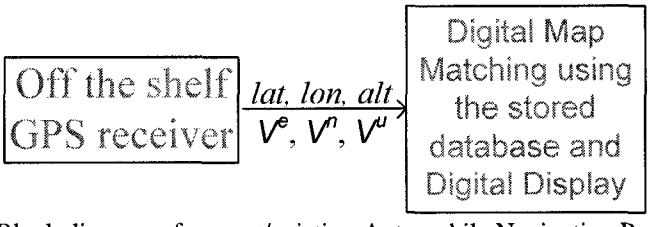

Figure 3.5: Block diagram of current/existing Automobile Navigation Product in the Market as portable in-car GPS device

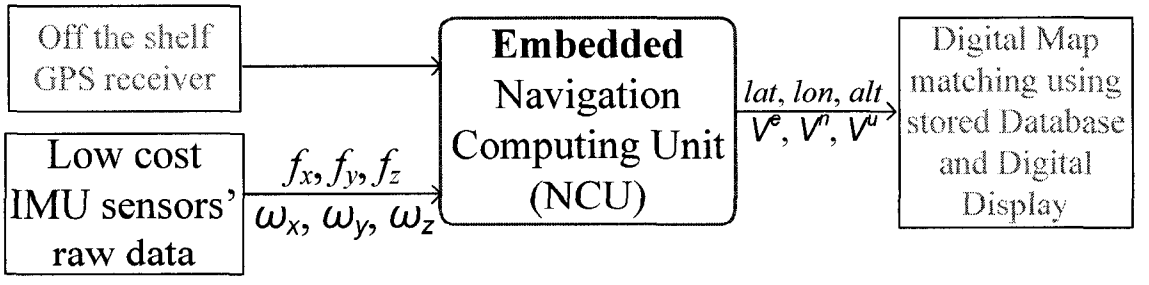

Figure 3.6: Block Diagram of the Navigation solution implemented this thesis.

As shown above in figure 3.5, the block diagram represents the existing navigation product available in the current market. The figure 3.6 summarizes the implementation work (both in terms of hardware and software) carried out in this thesis.

3.4.1 Land Vehicle Navigation Data

Implementing navigation algorithms involves processing navigation data. At the first stage of the development, the ideal scenario for a thesis/project establishing the feasibility of using embedded processor would require simulated IMU sensor data (accelerometer and gyro data) for a reference trajectory (with position and corresponding velocity information at each time instance). The second best scenario would be to use low-cost IMU sensor data collected from a field experiment.

Due to limited resources, simulated data could not be procured. A road test data involving GPS and tactical-grade IMU namely TG6000 (KVH Industries, Inc., Middletown, RI) have been obtained thanks to ANI research group [28]. The TG-6000 measures angular rate and linear acceleration in the X, Y, Z axes with three fiber optic gyroscopes and three accelerometers. The data has been provided to MicroBlaze to perform the navigational algorithms mentioned in chapter 2.

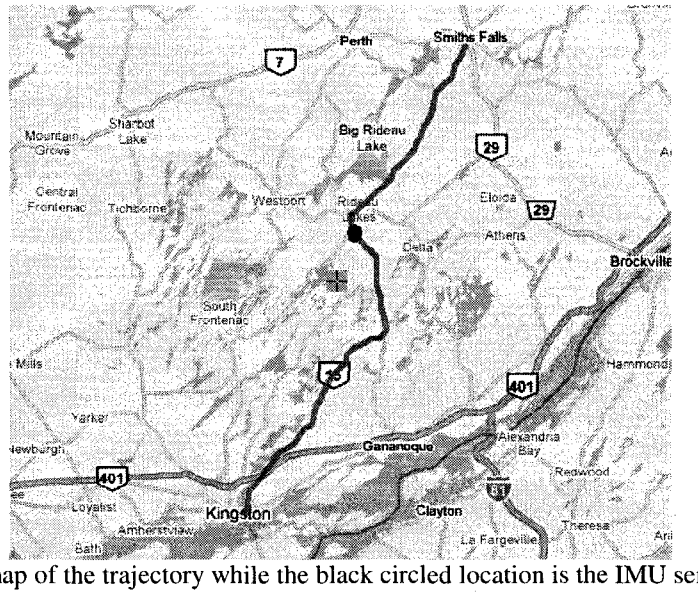

Figure 3.7: Complete map of the trajectory while the black circled location is the IMU sensor (during a simulated GPS outage) data used.

3.4.2 2D Mechanization

Yaw (heading) angle was calculated in radian using the equation (2.13). Using this value, accelerometer outputs (f_x and f_y) were transformed to the local level frame in equation (2.14). East and North velocities were calculated using equations (2.15) and (2.16), position information was obtained as a unit conversion is applied (shown in equation (2.23) and equation (2.24) and illustrated in the figure 2.15) to obtain the final output in radian.

3.4.3 3D Mechanization

Using the *initial* attitude information, a DCM matrix is constructed as shown in equation (2.8). This DCM matrix is then transformed to quaternion parameters using the equation (2.6). Now, from the *initial* parameters of the quaternion, the parameters for the next/current time step are obtained.

Using the new attitude parameters for the *current* time step, equation (2.24) and equation (2.27) are used to obtain velocity and position domain solution respectively.

A simple second order Runge-Kutta integration method has been used to transform the velocity increments to position increments. Instead of integrating by fixed time step integration method (Euler integration), the output has been smoothed to an extent in this way. The Euler integration algorithm works by assuming that the slope of a function is constant over the period of integration. The second order Runge-Kutta algorithm provides some compensation for changes in the slope over the integration interval [10].

3.4.4 Kalman filter

For low-cost IMUs like tactical-grade IMU, the mechanization procedure only gets fulfilled after combining it with a working KF Module which can be created after tuning many sensor noise variance values.

The KF code was made to run for the first 10 seconds using the aid of GPS data and the next 20 seconds in standalone mode. This 20s corresponds to the same forced GPS outage as applied to the 2D and 3D INS Mechanization implementation shown in the previous two sections. In both non-outage and outage scenario, the KF code went

through prediction and update stages but by using two different Error Covariance Matrix (R_k) values as shown in equation (2.45). During the outage, a higher value of R_k matrix was used as a higher standard deviation would decrease the KF's reliance on the Error Measurement Vector z_k (shown in equation (2.42) and (2.43)) and, thus, increase its confidence in the INS output.

Using the INS error equations, represented by equations (2.30) to (2.35), error state transition matrix $F_{k,k-1}$ of size 15x15 was constructed. As shown in equation (2.37), in *prediction* stage, the *15* error state shown in equation (2.36) gets propagated to the next time step by being multiplied with the *15*x*15* matrix. Then, using equation (2.38), the covariance matrix of this new predicted error state (P_k^-) is obtained.

In the *update* stage of the KF, the Kalman Gain Matrix (K_k), in equation (2.41), is obtained using the Covariance Matrix for the estimate of the Error (P_k^-) and Measurement Error Covariance Matrix (R_k).

The Kalman Gain (K_k) matrix is then used to update the Covariance Matrix for the estimated error state (P_k) using equation (2.44). It is also used to update the estimated error state vector $\hat{x}_k^-$ using equation (2.40) where the Measurement Vector is given by z_k.

The Measurement Vector z_k is obtained from GPS position and velocity measurement update as shown in equation (2.42). In the scenario of *forced* GPS outage, the last available GPS data (before the outage) is used in the calculation of z_k.

The final position, velocity and attitude output of KF is obtained by subtracting the corresponding value of the updated error state vector ($\hat{x}_k$).

At the update stage of the 15 state KF Model, a *6*x*6* matrix inversion is involved as shown in the equation (2.41). To implement it, "Gauss-Jordan elimination without pivoting" technique has been used deriving from the code snippet shown at the reference [30].Through debugging, it was observed that the matrix to be inverted $(H_k P_k^- H_k^T + R_k)^{-1}$ had always non-zero elements at the diagonal. As a result of this simplification, non-pivoting technique was chosen over pivoting technique. The

complexity of this computation is approximately ($2n^3/3$). The corresponding number of clock cycles consumed to perform the 6 by 6 matrix inversion operation is shown in chapter 4.

3.4.5 Wavelet De-noising

One of the motivations of presenting this sliding window-based wavelet multi-resolution based analysis and threshold is to show the real-time implementation on soft processor based low-cost FPGA. This thesis does not focus on the details of the accelerometer and gyroscope sensor data format (analog or digital, if digital then whether ASCII or in 32-bit format), sensor interface and other details. However, this module was built to verify the feasibility and to model the embedded implementation of any de-noising algorithm for low cost inertial sensors: may it be wavelet de-noising [4] or computation load intensive but highly effective high resolution spectral de-nosing algorithm like FOS [3].

Further research should be conducted on the following factors related to the *optimum* real-time de-noising of any accelerometer and gyroscope sensor data:

1) The non-overlapping window data length
2) Types of wavelet
3) Level of Wavelet Multi-resolution analysis: how many de-composition levels are appropriate?
4) Different threshold algorithms for the wavelet coefficients

Two types of windowing techniques are usually employed in IMU sensors pre-filtering technique implementation in real-time mode. They are: *sliding* window and *non-overlapping* window [31][32]. For *non-overlapping* window, the data set is portioned into sections of certain length. This non-overlapping style has been chosen due to the low level of complexity (in computation load) associated with it. Daubechies-5 (‘*db5*’) was chosen for implementing wavelet de-noising as it has been deemed appropriate for low cost IMU sensors as used in reference [32] in real-time implementation.

The optimal level of de-composition varies with the bandwidth of true motion dynamic in each sensor [18]. The sampling rate of the IMU data used in this thesis is 75 Hz, a *3* LOD for ω_x, ω_y and f_z data and a *5* LOD for f_x, f_y and ω_z would be appropriate according to the results presented in references [18] and [32]. Table 3.2 (adopted from [18]) illustrates the bandwidths of true motion dynamic sensed by different IMU sensors and also confirms, as $2^5 = 32$, a *5* LOD would result the finest *approximation* coefficients to contain frequency contents of 0 Hz to 1.172 Hz of f_x, f_y and ω_z sensors' data. In the same way, as $2^3 = 8$, a *3* LOD would result the finest *approximation* coefficients to contain frequency contents of 0 Hz to 9.375 Hz for ω_x, ω_y and f_z sensors' data.

Table 3.2: Bandwidth of True Motion Dynamics of IMU sensor data.

IMU Sensors	Gyro			Accelerometer		
	ω_x,	ω_y	ω_z	f_x	f_y	f_z
Bandwidth	<8 Hz	<8 Hz	<2 Hz	<2 Hz	<2 Hz	<8 Hz

The common thresholding methods (namely *rigsure*, *sqtwolog*, *heursure*, *minimaxi*) implemented in the Matlab Wavelet toolbox [33] can be applied to analyze the best possible result for low cost IMU sensors' performances and that relates to a vast field of research [32]. In this thesis, a *soft* thresholding technique (shown in table 2.3) is applied over raw IMU sensor data with the *level dependent* thresholding technique shown in equation (2.52). To calculate the median of each detailed coefficients, a *qsort* algorithm is used [30].

3.5 Software Design Issues

A "software platform" on MicroBlaze refers to a collection of software drivers and the operating system (if not a standalone application, then an OS such as XilKernel can be included) on which to build the given application. The software image created consists only of the portions of the Xilinx library used in the embedded design. The EDK tool captures the software platform in the MSS file (Microprocessor Software Specification). It automatically generates the memory map of the hardware platform as

well as assigning default drivers to the processor and each of its peripherals. Thus, as mentioned, the program running on the processor core is built using basic standard C [25]. Software coding in MicroBlaze was done using basic standard C libraries and device drivers since usually there is no operating system (known as 'standalone' application) between the software and the hardware platform. This is the language format supported by Xilinx Embedded Development Kit (EDK) tools. As mentioned before, EDK includes GNU C compiler and Xilinx Microprocessor Debug (XMD) module. The Base System Builder used for creating the hardware platform optionally generated a software project called "TestApp_Memory" which contained a sample application and linker script. The code developed in this thesis was built on top of this BSB generated sample software template.

The "float.h" header file was included at the top of the .C file. Also, all floating point variables were assigned with single precision variable type *float* as they correspond to the default *double* in C programming language. The floating point variables/constants declared are initialized in the following fashion:

```
float test = 0.00f;
```

The above code snippet illustrates the use of *f* at end of variable declaration. In the same way, the single precision floating point supporting trigonometric functions were used e.g. `sinf()`, `cosf()`, `sqrtf()` instead of `sin()`, `cos()` and `sqrt()` respectively to avoid any interaction with the default library implementation in double precision floating point format.

Debugging on the application code for MicroBlaze was carried out manually in two phases. The first phase involved application code validation. It was done by simultaneously comparing the output of Microsoft Visual Studio environment with that of the MicroBlaze. In the second phase, for each small segment for the code its timing performance was obtained and improved by software code optimization. It was found that declaring function and variables with the '*f*' suffix as shown above (for performing single precision floating point operation instead of double precision floating point operation) significantly reduces code execution time.

3.5.1 Data I/O

The EDK libraries contains *reduced* version of standard C functions for I/O, such as *print()*, *xil_printf()* suited for embedded processors due to the limitation of hardware area as opposed to desktop PC environment. This reduced version of I/O functions in size (only 1KiB) does not offer support for floating point numbers [34]. Moreover, the *xil_printf()* included in the "stdio.h" header file treats every floating point values as double. Usually, C converts *float* to *double* before passing it to a variable argument of any function. It was found that *"%f"* expects a double, not a float, as it always was converted before the call.

To overcome this problem, a snapshot of the memory content (i.e. performing a direct read of the memory content) is printed using the existing *xil_printf()* function. Following code snippet represents a typical scenario:

```
float test = 22.002f;
xil_printf("test = 0x%08x\r\n", *(int*)&test);
```

Here, the 08x forces *"%x"* to print at least 8 character which is equivalent to 32 bits. The float variable *test* is typecast to integer as floats and integers on the MicroBlaze are of the same size (4 bytes). Additionally, the reduced *xil_printf()* function on MicroBlaze provides full support for integer numbers. So, the value in memory is the same. A Matlab script was created to read each of the 8 ASCII character outputs (in Hexadecimal format) from the terminal, to translate it to the corresponding 32-bit representation (1's and 0's) and eventually to the corresponding IEEE-754 floating point decimal value for results analysis as performed in chapter 4.

By using a custom function instead of using the standard I/O functions of the C library, the code size could have been further reduced.

3.5.2 Run time errors

Due to the large code size of KF model, several run-time errors were encountered. They are described here.

In the process of debugging it was observed that after running a certain length of the code, all the variables values were turned into garbage values indicating a (negative or positive) overflow. After attempting different approaches to detect the reason and

overcoming the situation, it was found that an increase of stack size from the default size in the linker script resolves the issue. From this, it was concluded that a stack overflow caused run-time error.

Another run-time error that occurred while implementing KF model was related to memory management. As the KF model includes several matrix variable declarations as large as 15x15 dimensions, the 2D array declaration for them as local variable was generating run-time error. To overcome this situation, double pointers were used with initialization using *malloc()* function. Another approach could have been using global variable declaration and thus avoiding the use of local variables.

CHAPTER 4: Results and Discussion

Chapter 4 presents the result of the embedded implementation described in chapter 3. It starts with examining the embedded application outputs (of navigational algorithms) by comparing them with those performed on a desktop PC (Matlab models running on Pentium micro processor) platform using Matlab tool. Then it discusses the timing performance of the embedded navigational application. Finally, the chapter outlines the hardware utilization summary of the embedded software.

4.1 Navigation Solution using MicroBlaze

4.1.1 2D Mechanization

Figure 4.1 and 4.2 show the position output comparison between Matlab model and MicroBlaze computation of 2D Mechanization for land vehicle experiment position results during 20s forced GPS outage. Figure 4.1 brings out the precision issue related to 32-bit floating point representation. It shows that due to large dynamic range, a position increment of a moving automobile in every (1/75) second in east and north direction is too small to be added to a large latitude and longitude quantity respectively.

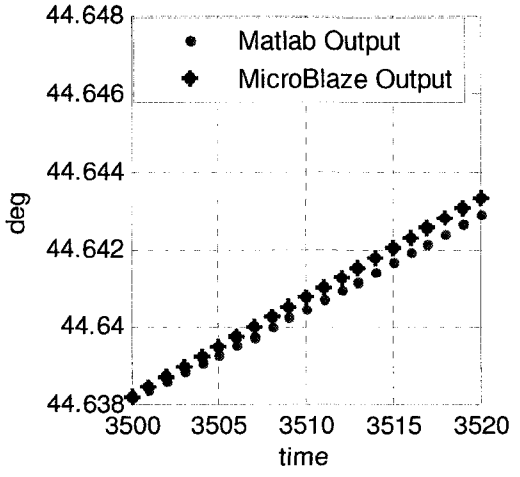

Figure 4.1: Latitude output comparison of 2D Mechanization.

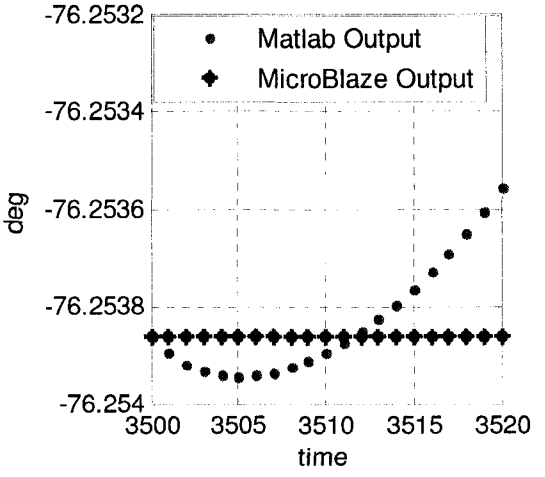

Figure 4.2: Longitude output comparison of 2D Mechanization.

Moreover, in *latitude* and *longitude* calculation, the *reference* points are on Equator (zero latitude) and on Greenwich Meridian (zero longitude). The navigation

trajectory data [28] used was collected from road test conducted around Kingston city, Ontario, Canada and this location is far away from the Equator line and even further away from the Greenwich Meridian line. From figure 3.6 it can be said that during the *forced* GPS outage, the vehicle was heading towards North. Thus the change of position towards East-West direction at every (1/75) second was too little compared to the large distance from the Meridian Line. Adding a very large number to an extremely small one produces no significant change in computation as evident in figure 4.2. In contrast, in figure 4.1, even though there is an accumulated error growth, at least, the effect of the addition computation is visible.

Starting experiments in a new embedded platform with the simplest algorithm such as 2D Mechanization thus proved helpful to get introduced to a precision related issue in navigation solution computation. To overcome this situation, a localized *reference* point (resulting a smaller distance) was used that can be easily added to the smallest possible east and north position increments calculated.

4.1.2 Mechanization and Kalman filter

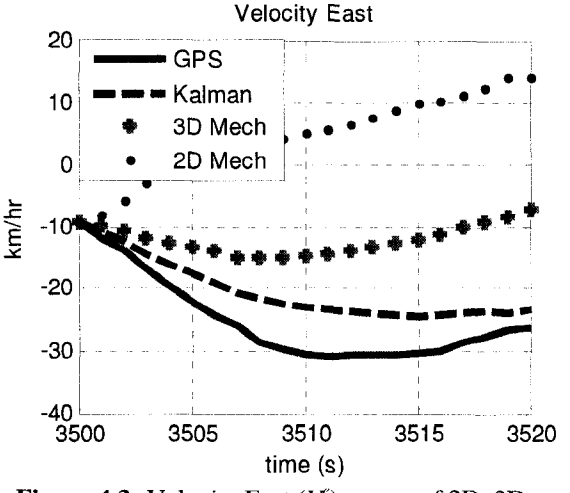

Figure 4.3: Velocity East (V^e) output of 2D, 3D Mechanization and KF.

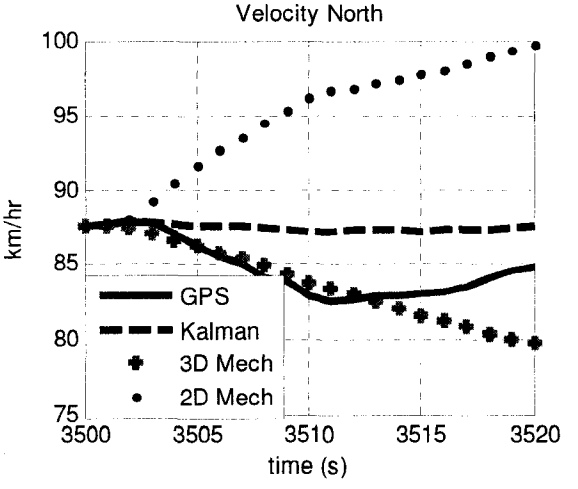

Figure 4.4: Velocity North (V^n) output of 2D, 3D Mechanization and KF.

Figures 4.3 and 4.4 show the velocity output plots of the 2D, 3D Mechanization and KF code running on MicroBlaze platform. The improvement of the performance (with respect to the reference GPS output) of 3D Mechanization over 2D Mechanization and that of KF over 3D Mechanization is evident.

Figure 4.5 and 4.6 demonstrates the performance of the KF in position domain with respect to the reference GPS (during the *forced* GPS outage). Position Error Plot as a function of time in North and East direction show the error growth as the time progresses. From these two position error plots, it can be deduced that (for this specific *forced* GPS outage) the total horizontal error is approximately 30 m $\left(\sqrt{14.8168^2 + 28.0588^2} = 31.73061\right)$

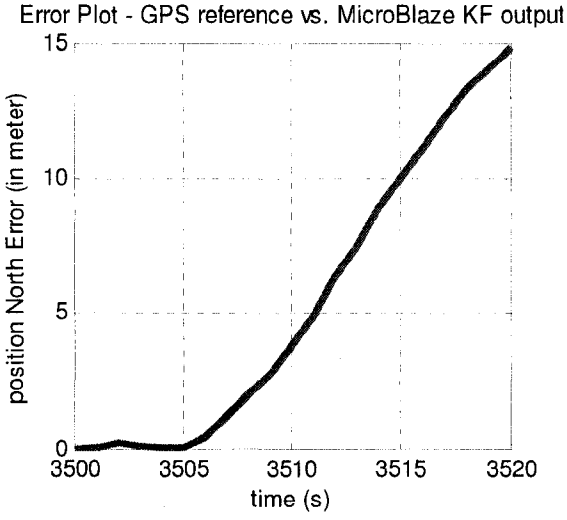

Figure 4.5: MicroBlaze output: Position North Error (in meter) with respect to GPS reference.

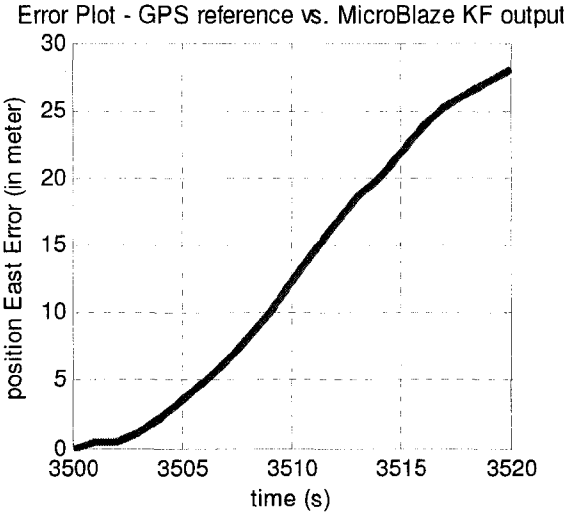

Figure 4.6: MicroBlaze output: Position East Error (in meter) with respect to GPS reference.

Figures 4.7 and 4.8 show the error plots between the computations carried out in Matlab Model and MicroBlaze. The reasons for the differences of the outputs can be described as follows.

In MicroBlaze platform, the computation was carried out in single precision (32-bit) floating point format compared to double precision (64-bit) floating point format used in Matlab which was running on desktop PC. In Matlab model, computing INS error equations in KF require *absolute latitude* (resulting from 3D Mechanization computation) values at every time instance. However, computation carried out in 32-bit platform cannot support the dynamic range of adding a very small latitude update (in every 1/75 second) to a very large initial latitude value (as described previously and shown in figures 4.1 and 4.2). To circumvent this scenario, the (absolute) latitude update in radian was carried out in every one second instead of every 1/75 second so that the

update value accumulates to a sufficiently large value enabling it to support the dynamic range of single precision floating computation. This different update method influences the Earth's radius (equation (2.2) and equation (2.3)) and gravity update (shown in equation (2.4)) computations.

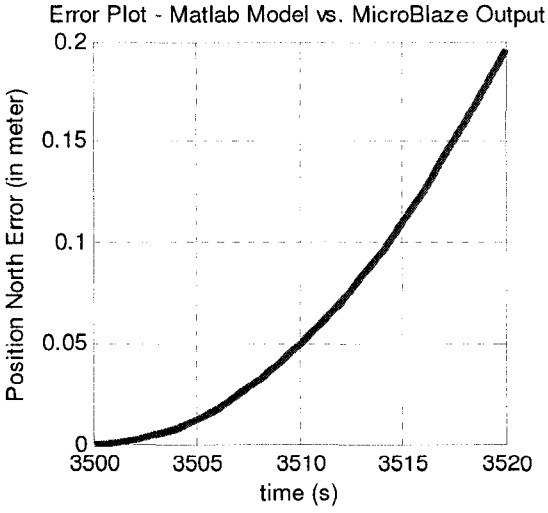

Figure 4.7: MicroBlaze vs. Matlab model output comparison – Position North Error (in meter)

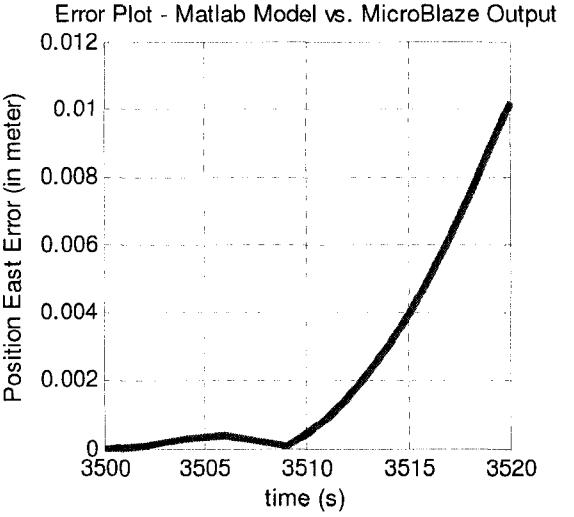

Figure 4.8: MicroBlaze vs. Matlab model output comparison - Position East Error (in meter)

Figure 4.7 and figure 4.8 show the overall position error/difference derived between Matlab outputs and MicroBlaze results is negligible as the overall error value here is 3.8 cm $\left(\sqrt{0.0102^2 + 0.1957^2} = 0.0384\right)$.

4.1.3 Wavelet De-noising

From examining the de-noised time domain plots (in figure 4.10 and figure 4.14) and PSD plot (figure 4.11) of de-noised signal, the de-noising effect of the algorithm implemented on MicroBlaze (even though there is no error free reference of IMU data) becomes apparent. In the above figure 4.9 and figure 4.10 and in the PSD plot of figure 4.11, only the data corresponding to Y-axis accelerometer is shown for the period of forced outage period of the trajectory (from 3500 second to 3520 second). This is because, being the sensor which detects the forward motion of the vehicle, the Y-axis accelerometer would provide data that is generally representative of the vehicle's overall motion dynamics.

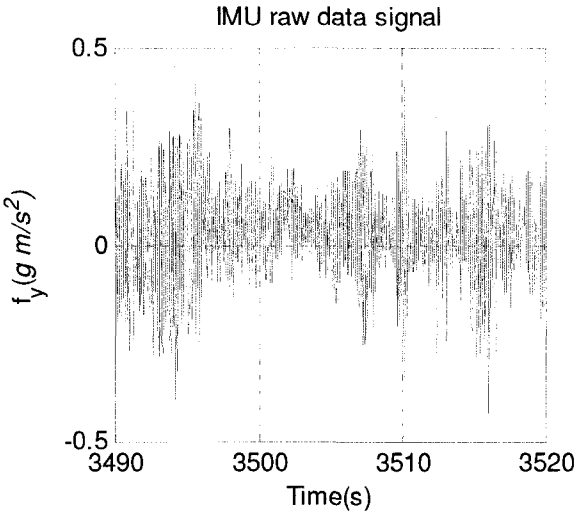

Figure 4.9: Raw time domain signal of *Y*-axis accelerometer.

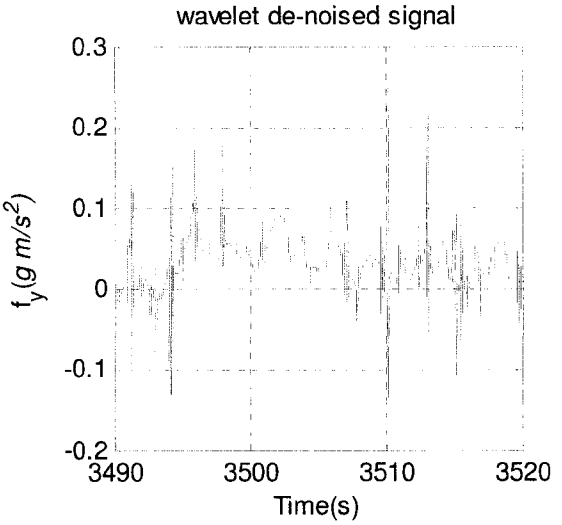

Figure 4.10: Wavelet De-noised time domain signal of *Y*-axis accelerometer.

By examining the PSD plot of figure 4.11, it can be said that Wavelet De-noising on Y accelerometer data carried out on MicroBlaze SCP is not only able to attenuate short-term errors existing beyond 2Hz, but it can also reject part of the long term errors present in the spectrum below it.

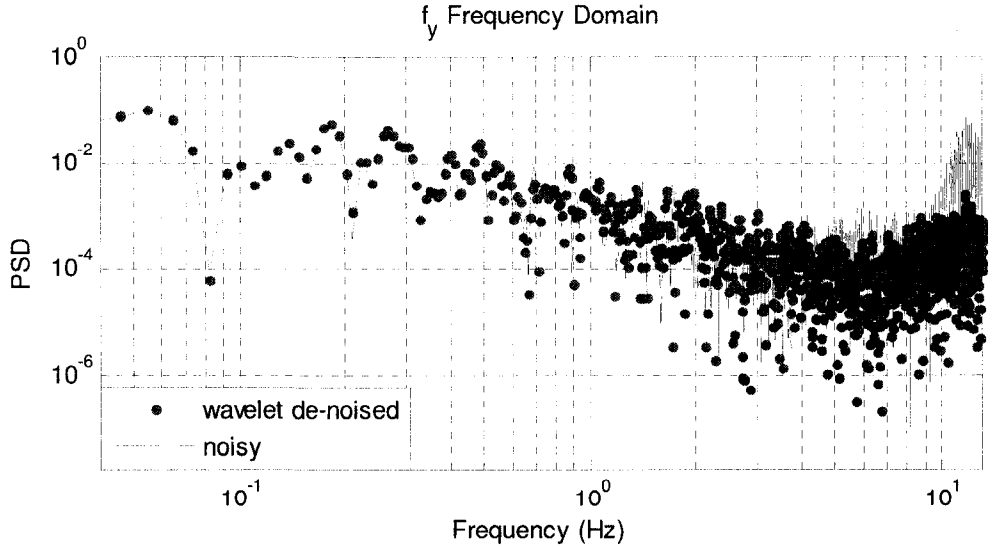

Figure 4.11: PSD of *Y*-accelerometer data using Wavelet De-noising.

The error plot of figure 4.12 clearly proves that Wavelet De-noising algorithm on low cost IMU data with high sampling rate (75 Hz in this case) can be successfully implemented on MicroBlaze SCP using its floating point hardware unit.

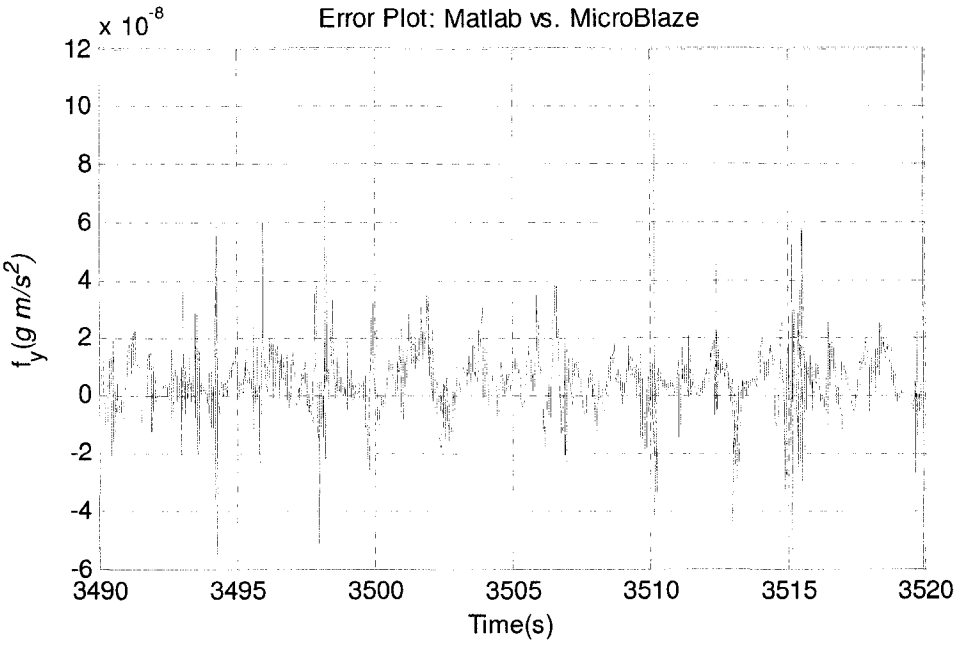

Figure 4.12: Error Plot – comparison between wavelet de-noised data outputs from Matlab and MicroBlaze

Figure 4.10 and figure 4.14 shows the visual effect of lowering the LOD of wavelet de-noising algorithm from 5 to 3.

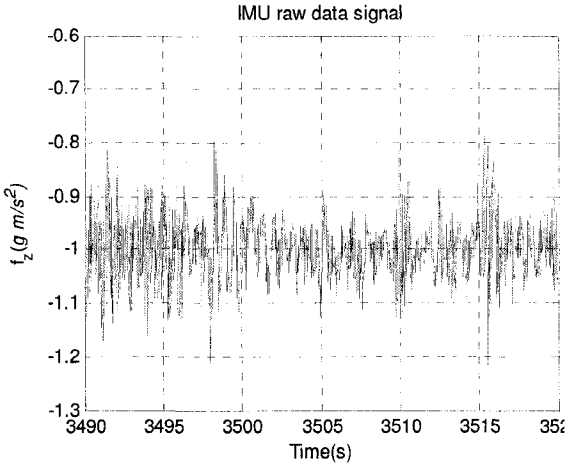

Figure 4.13: Raw time domain signal of *Z*-axis accelerometer.

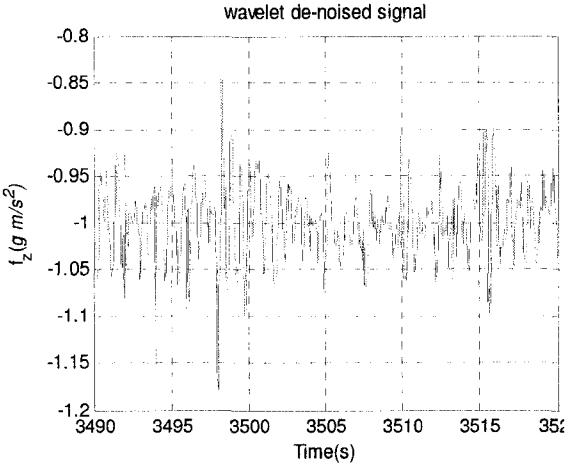

Figure 4.14: Wavelet De-noised time domain signal of *Z*-axis accelerometer.

4.2 Timing Measurements

Three different MicroBlaze configurations (using the BSB wizard of EDK tool) were implemented as the target of a software navigation application. In configuration 1,

the code executed entirely from the external SRAM. In configuration 2, 4KiB of Instruction cache and 8KiB of Data cache were enabled. In configuration 3, the entire application code ran from on chip BRAM of size 64 KiB. As the Spartan-3 Starter Board had only 24KiB of BRAM, the XUP Virtex-II Pro Board was used for configuration 3. Contrary to configurations 1 and 2 (both operated at the clock speed of 50MHz), configuration 3 operated at the clock speed of 100 MHz. In all three configurations, the stack/heap section of the code was kept at the BRAM section (along with the bootloop code for MicroBlaze).

The number of clock cycles taken/needed to execute the C code of following 4 algorithms (for a single iteration) is shown below. In contrast to the 2D and 3D Mechanization algorithms, both KF and matrix inversion algorithm uses loop operations intensively and it suits cache operation well (as it is shown in table 4.1).

Table 4.1: Timing results for 2D, 3D Mechanization and KF operation.

	Configuration 1 (clock cycles)	Configuration 2 (clock cycles)	Configuration 3 (clock cycles)
2D Mechanization	1772	1732	*390*
3D Mechanization	51937	37646	9471
Decentralized KF (*6x6* matrix inversion)	1843305 (50940)	619191 (12232)	420236 (11021)

It should be noted that the entry corresponding to 2D Mechanization implemented via Configuration 3 in Table 4.1 is italicized. The italicization highlights the fact that the code for 2D Mechanization could be run from 16KiB BRAM available on Spartan-3 Starter Board (i.e. there was no need to use the external 1MiB SRAM). In order to accommodate this code within this small memory limit, the calculations involving updating of Earth's Radii shown in equations (2.2) and (2.3) and gravity shown in equation (2.4) were omitted. Rather, these variables were assumed to be constant during this *forced* GPS outage period of 20 seconds. This assumption was extended to the 2D Mechanization performed by the other configurations to ensure that the timing output of all three configurations remain can be compared to each other. On the other hand, in the cases of 3D Mechanization and Decentralized KF, the

abovementioned variables i.e. Earth's radii and gravity were updated at the sampling frequency for the purpose of reporting timing performance of table 4.1.

The benefit of using Instruction and Data cache for external SRAM (which is connected through OPB bus of MicroBlaze) is clearly evident from table 4.1 and table 4.2. Configuration 1 can support up to IMU data rate 27 Hz (1843305 clock cycles) which is not fast enough to match the data rate of most of the MEMS based IMU sensors available in the market. By enabling the OPB cache option, up to 80 Hz (619191 clock cycles) data rate can be supported. By adding extra BRAM to the FPGA chip of Spartan-3 Starter board, the code can support up to 119 Hz data rate. Here, the application code in configuration 3 can support (119x2 = 238 Hz) as the clock frequency used for MicroBlaze SCP was 100 MHz..

Table 4.2: Timing results of Wavelet De-noising carried on 75 samples of IMU sensor data.

Level of de-composition and reconstruction plus adaptive thresholding	Configuration 1		Configuration 2		Configuration 3	
	clock cycles	ms	Clock cycles	ms	clock cycles	ms
3	922296	18.4459	199075	3.9815	147829	1.4783
5	1228636	24.5727	273555	5.4711	196361	1.9636

Table 4.2 shows that the wavelet de-noising algorithm execution time is in the order of millisecond i.e. less than 6 ms while OPB SRAM cache enabled. As the algorithm operates in real time using non-overlapping window mode, this execution latency is negligible compared to the delay associated with the IMU data buffering i.e. waiting for arrival for appropriate data to construct the non-overlapping window. By using cache, the execution latency of wavelet de-noising in real-time (by using non-overlapping window length of 75 in this case) was speeded up almost 5 times as shown in the table 4.2. The nature of the most low-cost IMU raw data pre-processing algorithms (wavelet de-noising, FOS etc.) involves intensive use of loops and this usage suits cache operation well. In the worst case scenario, even if there was a cache miss at the beginning of the loop, the subsequent loop operations are executed as if the code was running from high-speed BRAM.

4.3 Hardware Device Utilization Summary

Table 4.3 provides the device utilization summary and table 4.4 represents the post synthesis clock frequency of the critical modules for the three different hardware platform configurations (the same three different platforms shown in table 4.1 and table 4.2 for timing analysis purpose) used in the implementation. The data of the tables 4.3 and 4.4 are collected from the design report generated by the EDK tool for each hardware design. As a side note, it should be observed that choosing FPU leads to twofold increase (non-FPU version uses 950 logic cells while the current one employs at least 1616 logic cells) in the usage of logic cells. This observation can act as a motivation behind exploring future implementations which excludes the FPU.

Table 4.3: Hardware resources used by major IPs

	Configuration 1	Configuration 2	Configuration 3
Microblaze 32 bit soft processor (version: 4.00a)			
Slices	1628 (84%)	1717 (89%)	1616 (11%)
Slice Flip Flops	1335 (34%)	1343 (34%)	1332 (4%)
4 input LUTs	2519 (65%)	2721 (70%)	2382 (8%)
BRAMs	N/A	8 (66%)	N/A
MUL18X18s	7 (58%)	7(58%)	7 (5%)
Memory: Block RAM (BRAM) Block			
BRAMs	8 (66%)	4 (33%)	32 (23%)
On-chip Peripheral Bus (OPB) 2.0 with OPB Arbiter			
Slices	141 (7%)	141 (7%)	98 (~0%)
RS-232 OPB UART (Lite)			
Slices	51 (2%)	51 (2%)	54 (~0%)
SRAM_256Kx32 OPB External Memory Controller			
Slices	200 (10%)	200 (10%)	N/A

Inspired by the successful IMU data pre-processing implementation in real-time hardware in reference [35] and its optimization in the hardware area, the feasibility of using fixed point calculation in implementing the wavelet de-noising algorithm for IMU

data was explored. By *scaling* the de-composition and reconstruction filter parameters and the IMU raw data by an optimal value (through tuning), a single level of de-composition and reconstruction operation was carried out successfully. But as the LOD was increased to 3 and 5, an overflow was detected at the MicroBlaze output. The other obstacle in the *scaling* approach was the non-linearity introduced by the *log* function of wavelet thresholding (shown in equation 2.55). In short, due to unsatisfactory preliminary results, fixed point calculation scheme was replaced by the one using floating point.

The total area utilized by IPs not included in table 4.3 (such as LMB BRAM Controller, OPB Microprocessor Debug Module etc.) constitute less than 5% of the area of the single major IP Microblaze 32 bit soft processor (version: 4.00a).

Table 4.4: Post synthesis clock frequency for hardware configurations.

Modules	Configuration 1 (MHz)	Configuration 2 (MHz)	Configuration 3 (MHz)
Microblaze	**83.19**	**63.29**	**130.88**
OPB Interrupt Controller	118.38	118.38	190.10
OPB Timebase WDT	122.62	122.62	197.93
SRAM_256Kx32	131.80	131.80	N/A
RS-232	149.86	149.86	209.47

Table 4.4 shows the critical frequencies of the (some of the) corresponding IPs for three hardware configurations. The minor IPs such as SRAM_256Kx32 OPB External Memory Controller, RS-232 OPB UART (Lite) etc. runs much faster in each configuration than the single major IP MicroBlaze. Here, the clock on Spartan-3 Starter board and XUP Virtex-II Pro Board was configured to 50 MHz and 100 MHz respectively. Thus, application code running on prototyped MicroBlaze SCP would perform faster if the digital clock configured properly with the critical clock frequency value. In other words, there is thus an interesting processing margin available for further algorithmic enhancements.

Table 4.4 shows that, even though the application code run faster (shown in table 4.1 and table 4.2) on configuration 2 due to enabling OPB data and instruction cache in

comparison with configuration 1 (no cache), the critical clock frequency for the MicroBlaze core without the cache option (configuration 1) is higher. Not shown in table 4.4, the critical frequency for a MicroBlaze configuration without using external SRAM (on Spartan-3 Starter board) is found to be 83.19 MHz as well. Thus the critical frequencies for the MicroBlaze hardware configurations built Spartan-3 on Starter board had an enhancement of 66%, 26% and 66% for configuration 1, configuration 2 and for a similar of configuration 3 (built on Spartan-3 FPGA using only BRAM) respectively.

CHAPTER 5: Conclusion and Further Work

5.1 Summary of Contribution

From the work carried out in the thesis work (research, implementation) and with the navigational results and timing performances shown in chapter 4, the following developments can be considered as contribution of this thesis.

5.1.1 Development of navigational algorithms

Extensive research has been carried out with the aim of understanding some of the existing navigational algorithms related to GPS/INS integrated automobile navigational solution and to preprocess low-cost raw IMU data. This knowledge, found to be scattered in numerous contributions in the literature about this field, has been collated in an organized manner in Chapter 2 of the thesis. Applying the acquired knowledge, a closed loop decentralized KF filter model was built through 15-state INS error equations. Subsequently, by tuning the parameters appropriately, the model was validated successfully for a specific *forced* GPS outage lasting 20 seconds of an automobile. Additionally, an algorithm implementing Wavelet De-noising for preprocessing low-cost IMU data developed both in Matlab and C.

5.1.2 Porting to the Embedded Platform

The software implementation in C of navigational algorithms was successfully ported to the low cost MicroBlaze soft processor. The validated outputs, characterized by plots showing the resulting errors, are shown in figures 4.7, 4.8 and 4.12. The validation shows that a purely software implementation on a single precision embedded platform can produce acceptable results relative to the results obtained from a desktop PC platform that uses double precision floating point numbers.

In other words, as demonstrated by figures 4.5 and 4.6, this thesis introduced, a low cost embedded implementation of a navigational computing unit (NCU) – capable of providing satisfactory navigational solution for a very short GPS outage (lasting up to

20 seconds). The associated wavelet de-noised (applied on raw data and resulted from MicroBlaze) IMU sensor data are shown in figures 4.10, 4.11and 4.14.

5.1.3 Real-time solution

The number of clock cycles required for executing navigational algorithms on MicroBlaze platform was measured in order to validate the real-time requirement of the implementation as shown in table 4.1 and table 4.2. To improve timing criteria with the aim of meeting a sampling rate up to 119 Hz (not including the speedup resulting from increasing the clock frequency to critical value), the necessity of running the software application code (for MicroBlaze) from high speed Block RAM of the FPGA is highlighted for faster execution.

5.1.4 A reference for future developers

MicroBlaze processor core was used successfully as it provides the flexibility of using arbitrary algorithms to be coded in higher level programming language like C, thereby avoiding the need of using other HDL extensively. In this way, the usage of MicroBlaze reduces the development time and complexity to a great extent compared to the case of purely hardware implementation. Thus, by employing the combination of a low cost embedded platform, a flexible development approach and a real-time solution by running the code from BRAM, the implementation shown in this thesis work proves that synthesizing a completely functional low-cost, real-time navigation solution is feasible.

In the development of the KF model, a considerable amount of time and effort was devoted to make it functional for any random GPS outage on the given trajectory as shown in figure 3.7 and to extend the positive effect of wavelet de-noising in real-time mode on KF output to produce navigational solution domain (i.e. improved position, velocity and heading solution). Due to the lack of direct access to the collection of real-time sensor data (as GPS and INS sensor data was only available in an ASCII file format.), details related to INS and GPS hardware sensor data issues viz. data acquisition format, sensor setup, initial alignment, detailed sensor error characteristics and error

variance values, synchronization between IMU sensors and GPS, time accuracy and precision of the sensors outputs etc. could not be addressed in this thesis.

With a view to commercialize the end product of the proposed implementation i.e. a low cost integrated GPS/INS embedded navigational system capable of successfully bridging short-term GPS outages in the market, a team comprising of software and hardware developers and experts in vehicle dynamics and GPS/INS sensor equipments are required. Provided such a team can be coordinated, this thesis can be considered as an ambitious introductory step toward that final implementation.

5.2 Recommendations

A significant insight was gained throughout the work on this thesis. This section mentions some aspects or observations that were acquired during the process. In other words, important topics that were found essential for implementing a low-cost *complete* automobile navigational solution to be marketed (capable of bridging the GPS outages) but fell outside the scope of this thesis are proposed as recommendations for future work with the aim of completing an embedded product capable of providing navigation solution using GPS and IMU sensors.

5.2.1 Observation #1

Detection of either GPS signal outages or deteriorated performance of GPS is a complex procedure. In this thesis, GPS signal data is assumed to be capable of providing accurate navigational solution (position and velocity solution). Additionally, *intentional forced outages* were introduced. In practice, GPS signal degrades gradually. For this reason, GPS signal *outage detection module* can be considered as an essential step towards a complete GPS/INS integration module implementation. As discussed in chapter 2 of the thesis, GPS fails to provide accurate navigational solution for two reasons. Firstly, there is a possibility of GPS outage due to the loss or blockage of the line of sight between the GPS receiver antenna and the GPS satellites. Secondly, GPS data becomes erroneous mostly due to multipath and cycle slip error. Apart from complete GPS outage/blockage scenario, data fusion through KF technique with

erroneous GPS data may not only lead to a significant degrading and/or confusing navigation solution but also cause a divergence in KF [2].

A decentralized KF architecture module capable of detecting cycle slip is shown in reference [36]. Multipath interference producing unreliable GPS solution detection module that can notify a GPS outage is recommended which will maintain the GPS solution integrity. Most present day off-the shelf GPS receivers are not equipped with algorithms that can detect this interference and notify a GPS outage to the user. Methods to reduce the interference effects in GPS receiver hardware are the research topics that are recommended for developments as well to enhance the GPS solution permanence. In this way, accurate detection of real-life GPS outages scenario will pave the way for the implementation of a complete integrated GPS/INS system capable of providing better navigational solution.

Another significant step towards a final embedded navigation solution that needs to be undertaken is the interfacing with the digital map matching module (related to the research filed of GIS) shown in figure 3.4 and 3.5.

5.2.2 Observation #2

For real-time navigational solution, this thesis has adopted a purely software design implementation (as it provides flexibility in development) on MicroBlaze SCP. Results obtained from timing analysis and software profiling suggests the implementation of the time-intensive portion of the code/module in hardware directly for better performance. Therefore, adopting the software/hardware co-design technique through low latency MicroBlaze FSL (Fast Simplex Link) interface or through OPB (On-chip Peripheral Bus) interface will enhance performance. In this regard, the following modules are recommended to be implemented (for solution speed up/acceleration) on hardware directly:

a) To handle input/output operations, a modified hardware UART controller with customized (optimized for navigational application) buffer size.

b) To invert Matrix while implementing the KF algorithm.

c) To pre-process/de-noise IMU data module using FOS algorithm as its significant superiority over wavelet de-noising algorithm in real-time mode suggested in the reference [35].

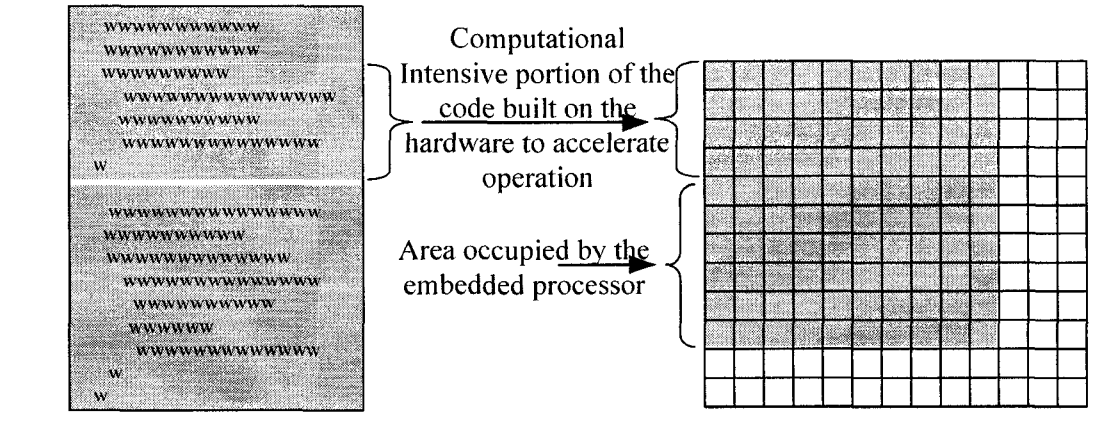

Figure 5.1: Illustration of Hardware-Software co-design on FPGA.

Critical portions of the software code can be executed faster by using multiple MicroBlaze SCPs connected to each other through FSL interface. As they are integrated with co-processing capability, both pre-processing/de-nosing IMU sensor data unit and KF can be implemented in two different processors. It would be worthy to measure and analyze the processing speed-up/acceleration as opposed to the purely implementation method illustrated in this thesis. Finally, the issue of lowering the power consumption of the embedded solution might be researched.

References

[1] Godha, S., "Land Vehicule Navigation System", Available: http://www.location.net.in/magazine/2006/sep-oct/34_1.htm. [Accessed Jan. 12, 2008].

[2] Misra, P. and P. Enge "Global Positioning System: Signals, Measurements, and Performance", Ganga-Jamuna Press, Lincoln, Massachusetts, 2001.

[3] Grewal, M.S., Weill, L.R., and Andrews, "Global Positioning Systems, Inertial Navigation, and Integration", A.P., John Wiley & Sons, Inc., 2001.

[4] Noureldin, A. Mobile Multi-Sensor System Integration, Course Notes provided for Royal Military College Course EE 513, Ontario, January 2006.

[5] Armstrong J., (M.A.Sc Thesis), "Application of Fast Orthogonal Search to Accuracy Enhancement of Inertial Sensors for Land Vehicle Navigation", Department of Electrical and Computer Engineering, Royal Military College of Canada, Ontario, April 2006.

[6] Abdel-Hamid W., (PhD Thesis), "Accuracy Enhancement of Integrated MEMS-IMU/GPS Systems for Land Vehicular Navigation Applications", Department of Geomatics Engineering, University of Calgary, Alberta, January, 2005.

[7] El-Rabbany, A.," Introduction to GPS: The Global Positioning System", Artech House, 2002.

[8] The Aerospace Corporation, Los Angeles, CA 90009-2957, USA. [Online]. Available: http://www.aero.org, [Accessed September 20, 2007].

[9] Spencer, J., Frizzelle, B., Page, P., Vogler, B., "Global Positionng System: A Field Guide for the Social Sciences", Blackwell Publishing, 2003.

[10] Farrell, J., Barth, M., "The Global Positiong System and Inertial Navigation", McGraw-Hill, 1999.

[11] Titterton, David H.; Weston, John L., "Strapdown Inertial Navigation Technology" (2nd Edition), Institution of Engineering and Technology, 2004.

[12] Rogers, Robert M., "Applied Mathematics in Integrated Navigation Systems", Volume 174, Progress in Astronautics and Aeronautics, AIAA Inc, 2007.

[13] Hsu, D.Y., "An accurate and efficient approximation to the normal gravity", Position Location and Navigation Symposium, IEEE 1998, 20-23 April 1998, Page(s):38- 44.

[14] Chatfield, Averil B., "Fundamentals of High Accuracy Inertial Navigation" Application of Fast Orthogonal Search to Accuracy Enhancement of Inertial Sensors for Land Vehicle Navigation", progress in astronautics and aeronautics, volume 174, 1997.

[15] Niu, X. and El-Sheimy, N., "The Development of a Low-cost MEMS IMU/GPS Navigation System for Land Vehicles Using Auxiliary Velocity Updates in the Body Frame", Mobile Multi-Sensor Systems Research Group, University of Calgary.

[16] Analog Devices Inc. (2007). Datasheet of ±150°/s Single Chip Yaw Rate Gyro with Signal Conditioning: ADXRS150. [Online] Available: http://www.analog.com/MEMS, [Accessed September 20, 2007].

[17] Eberts, M.D., (M.A.Sc. thesis), "Performance enhancement of MEMS based INS/GPS integration for low cost navigation applications", Department of Electrical and Computer Engineering, Royal Military College, Ontario, 2007.

[18] Chiang, K.W., (Ph.D. Thesis), "INS/GPS Integration Using Neural Networks for Land Vehicular Navigation Applications", Department of Geomatics Engineering, University of Calgary, Calgary, Alberta, 2004.

[19] Mohamed, A. H., (Ph.D. Thesis): "Optimizing the Estimation Procedure in INS/GPS Integration for Kinematic Application", Department of Geomatics Engineering, University of Calgary, Alberta, 1999.

[20] Burrus, C.S.; Gopinath, R. and Guo, H. (1998), "Introduction to wavelet and wavelet transforms a primer", Prentice Hall, 1998.

[21] Donoho, D., "De-noising by soft thresholding", IEEE Transactions on Information Theory 41(3), pp. 613-627, 1995.

[22] Digilent Inc., "Spartan-3 Starter Kit Board User Guide", UG130 (v1.1) May 13, 2005. [Online]. Available: http://www.digilentinc.com/Data/Products/S3BOARD/S3BOARD-rm.pdf [Accessed October 17, 2007].

[23] Digilent Inc., "Virtex-II Pro Development System", [Online]. Available: http://www.digilentinc.com/Products/Detail.cfm?Prod=XUPV2P&Nav1=Products&Nav2=Programmable [Accessed June, 2008]

[24] Xilinx Inc., "MicroBlaze Processor Reference Guide", Embedded Development Kit EDK 8.1i, February 21, 2006. [Online].Available: http://www.xilinx.com/ise/embedded/mb_ref_guide.pdf, [Accessed December. 12, 2006].

[25] Xilinx Inc., "Embedded System Tools Reference Manual", Embedded Development Kit, EDK 8.1i, October 24, 2005. [Online].Available: http://www.xilinx.com/ise/embedded/est_rm.pdf, [Accessed May. 12, 2007].

[26] Xilinx Inc., "Xilinx Shatters Price/Density Barrier for Low Cost FPGAs With New Spartan-3E Family Starting At Less Than $2.00", [Online]. Available: http://www.xilinx.com/prs_rls/silicon_spart/0531s3e.htm [Accessed May. 12, 2007].

[27] Xilinx Inc., "Embedded Development Kit: Programmable System Design is Now Easier than Ever", May, 2006 [Online].Available: http://www.xilinx.com/publications/prod_mktg/pn0010679-5.pdf , [Accessed March. 10, 2008].

[28] Advanced Navigation and Instrumentation Research Group, "TG6000 field test trajectory data", performed in summer 2006, Royal Military College of Canada, Kingston, Ontario.

[29] Noureldin, A, private communication, September 2007.

[30] Press, William H., "Numerical Recipes in C", 2nd ed., Cambridge University Press, 1992. [E-book] Available: http://www.nr.com/.

[31] Semeniuk L., (M.A.Sc Thesis), "Bridging Global Positioning System Outages using Neural Network Forward Prediction of Inertial Navigation Position and Velocity Errors", Department of Electrical and Computer Engineering, Royal Military College of Canada, Ontario, April 2006.

[32] Johnston, C. G., (M.A.Sc. thesis), "High Resolution Wavelet De-noising for MEMS-Based Navigation Systems", Department of Electrical and Computer Engineering, Royal Military College of Canada, Ontario, 2007.

[33] Misiti M., Misiti Y., Oppenheim G., Poggi J-M.: Wavelet Toolbox For Use with MATLAB, User's Guide, version 2. The MathWorks, Inc., 2002.

[34] Xilinx Inc., "OS and Libraries Document Collection", [Online].Available: http://www.xilinx.com/support/documentation/sw_manuals/edk91i_oslib_rm.pdf, [Accessed December. 12, 2007].

[35] Dybwad, S. , McGaughey, D. and Noureldin, A., "FPGA Implementation of High Resolution Spectral De-Noising Modules of MEMS-based Inertial Sensors" in Proceedings of the 20th International Technical Meeting of the Satellite Division of the Institute of Navigation ION GNSS, 2007, Pp. 121-126.

[36] Wei, M. and Schwarz, K.P., "Testing a decentralized filter for GPS/INS integration", Department of Surveying Engineering, The University of Calgary, Canada.