# DESIGN AND DEVELOPMENT OF DSP BASED GPS-INS INTEGRATED SYSTEM

A dissertation submitted in partial fulfillment of the requirements

for the degree of

**Master of Technology**

*by*

**Bhaktavatsala S**

(*Roll No. 02307407*)

*under the supervision of*

Prof. Vivek Agarwal

Prof. Hemendra Arya


Department of Electrical Engineering

Indian Institute of Technology, Bombay

Powai, Mumbai 400 076

June 2004

Indian Institute of Technology, Bombay

DISSERTATION APPROVAL

Dissertation entitled: **Design and Development of DSP based GPS-INS integrated system**

by **Bhaktavatsala S**

is approved for the award of the degree of **MASTER OF TECHNOLOGY in Electrical Engineering** with specialization of **Electronic Systems**.

Supervisor _____ (Prof. Vivek Agarwal)

Supervisor _____ (Prof. Hemendra Arya)

Internal Examiner _____ (Prof. K.Sudhakar)

External Examiner _____ (Mr. Anil Kulkarni)

Chairperson _____ (Prof. A K Verma)

Date:

# Acknowledgement

I would like to express my deepest gratitude to my guide Prof. Vivek Agarwal and Prof Hemendra Arya for guidance, support, and encouragement throughout the course of this project. I would like to thank Prof. K. Sudhakar for the suggestions I have received from him.

I am thankful to Vikas Kumar Naresh for his help in lab tests during development stage. I would also like to extend my sincere thanks to all my lab colleagues of our laboratory for all the help and support during this project.

I also thank Director, ADE, Bangalore and other office colleagues for their help and moral support.

At last, but not the least, I deeply appreciate all my family members especially my wife for their emotional support during the entire course work.

**Bhaktavatsala S**

June 2004

Bhaktavatsala S/ Prof. Vivek Agarwal and Prof Hemendra Arya (supervisor): "Design and Development of DSP based GPS-INS integrated system", *M.Tech. dissertation*, Department of Electrical Engineering, Indian Institute of Technology, Bombay, June 30, 2004.

# Abstract

In most autonomous navigation systems, reliable position information with high availability in real-time is essential. An integrated GPS-INS system provides more accurate estimates of position and attitude compared to individual systems. Availability of COTS inertial sensors and GPS cores has opened up new opportunities for low cost INS-GPS systems which find wide spread application. This paper describes an improved technique of acquiring INS and GPS signal data. Sampling of IMU signals is done simultaneously, which eliminates any phase delay between the acquired signals. GPS serial input link on specially designed FPGA reduces the processor's computing overhead. The FPGA is programmed for GPS data acquisition and the presence of internal DPRAM provides asynchronous communication. Processed GPS solutions have the advantage of a simple structure and easy implementation. Hence, a loosely coupled GPS-INS integration has been adopted in the proposed design. Real-time system discussed here is more suitable for autonomous navigation systems. Another attractive feature of the proposed system is the use of a single power supply for the entire system. All the details of proposed system development (including architecture design, technical specifications and the overall approach), along with its capabilities and salient features are presented.

# Contents

# List of symbols

| Symbol | Explanation |
|---|---|
| $a_x$, $a_y$, $a_z$ | Accelerations along the 3 body axes |
| $p$, $q$, $r$ | Angular rates |
| $U$, $V$, $W$ | Velocities along the 3 body axes |
| $?$ | Pitch angle |
| $?$ | Yaw or Heading angle |
| $f$ | Banking angle |
| $g$ | Acceleration due to gravity |
| $X$ | Position along the North axis in the navigation frame |
| $Y$ | Position along the East axis in the navigation frame |
| $Z$ | Position along the Down axis in the navigation frame |
| $H$ | Altitude |

# List of abbreviations

| Symbol | Explanation |
|---|---|
| GPS | Global Positioning System |
| INS | Inertial Navigation System |
| MAV | Mini air vehicle |
| MEMS | Micro Electro Mechanical System |
| COTS | Commercial Off The Shelf |
| IMU | Inertial Measurement Unit |
| FPGA | Field Programmable Gate Array |
| DPRAM | Dual Port RAM |
| RAM | Random Access Memory |
| ADC | Analog to Digital Convertor |
| DSP | Digital Signal Processor |
| RTOS | Real Time Operating System |
| ISR | Interrupt Service Routine |
| DGPS | Differential Global Positioning System |
| PC | Personal Computer |
| CPU | Central Processing Unit |
| NMEA | National Marine Electronics Association |
| GIDAC | GPS-INS data acquisition card |
| NPC | Navigation processor card |
| S/N | Signal-Noise ratio |
| CCS | Code Composer Studio |

# List of figures

# List of tables

# Chapter 1
# Introduction

## 1.1 Background

For accurate functioning of autonomous navigation systems, reliable position information in real-time must be available at frequent intervals of time. High performance inertial-grade **I**nertial **M**easurement **U**nits (IMUs) used in avionics systems exhibit superior performance specifications, however the high cost of standard **I**nertial **N**avigation System (INS) limits its widespread application in some general areas, such as navigation and positioning of **M**ini **A**ir **V**ehicle (MAV). With, the development of low cost **M**icro **E**lectro **M**echanical **S**ystem (MEMS) technology, **C**ommercial **O**ff **T**he **S**helf (COTS) based IMU system has proved to be a more feasible solution in autonomous navigation systems. However, the MEMS based INS systems are cost effective, compact and lightweight, but have certain inherent limitations on precision. In order to deal with some of the error characteristics of the MEMS based inertial system, the extended Kalman filter augmented by **G**lobal **P**ositioning **S**ystem (GPS) estimates is used. The time critical nature of autonomous applications requires that processing of INS computations and Kalman filter algorithm should be finished before the next update from the IMU sensors. This imposes a severe constraint on the integration software, the processor speed and the system architecture.

Early studies on GPS-INS integration techniques and hardware development have either used serial interface link or multiplexed **A**nalog-to-**D**igital **C**onverter (ADC) for INS data acquisition [1-4]. Also, all integrated systems discussed employ either two or three power supplies. Serial interface of sensor output involves a tedious processing overhead on navigation processor [2-3]. Further, with the multiplexed type ADC, the position information being considered for calculation, will not exactly the same at that instant of time [1, 4]. Different integration techniques of GPS-INS integration have been implemented to obtain accurate position in navigation systems [2, 5, 6]. A loosely coupled integrated approach [2] using a GPS receiver and a low cost strap down INS is chosen to overcome the problems and obtain high accuracy estimates of position and attitude. Most of the systems discussed [1,2,4] are

based on the commercially available embedded computer boards. This report concerns mainly on building an integrated system considering data acquisition, processing overheads, interfacing and time criticality issues.

## 1.2 Project objective

The objective of this project is to develop an integrated and miniaturized GPS-INS integrated system prototype for an MAV. The system should be compact, single supply operated, compact and light. As the IMU sensor outputs are accompanied by environmental noise, this needs to be removed by appropriate filtering techniques. GPS data acquisition using microcontroller or implementation on **F**ield **P**rogrammable **G**rid **A**rray (FPGA), and **D**ual **P**ort **R**andom **A**ccess **M**emory (DPRAM) interface between acquired GPS data and navigation processor is investigated. Feasibility of GPS-INS integration implementation on processor is studied. Suitable integration technique for GPS-INS integration and statistical combination of GPS and INS is performed by Kalman Filter to achieve more accurate position information.

## 1.3 Dissertation outline

The literature review of GPS-INS integrated system architecture, along with the system computation methods, is presented in Chapter 2. Chapter 3 describes GPS-INS integrated system besides briefing about GPS and INS errors. Also, the time critical issues related to GPS-INS integrated system is discussed. Chapter 4 provides suitable second order filter for removing noise components from sensors and discusses the comparison between multiplexed ADC versus simultaneous sampling ADC. Implementation of GPS data acquisition on FPGA chip is presented in Chapter 5. Chapter 6 explains the development of **D**igital **S**ignal **P**rocessor (DSP) based **N**avigation **P**rocessor **C**ard (NPC) for GPS-INS integration with necessary software and its peripherals for interfacing, followed by the results and conclusions of GPS-INS integrated system in Chapter 7.

# Chapter 2

# Literature Survey

In GPS-INS integration systems, IMU and GPS are used as sensors. Several integration techniques implemented on real-time embedded systems have been reported. Some of these are described here.

Faulkner *et al.,* [1] describes a development programme in which PC/104 computer based loosely coupled system and DSP based navigation processor is used for developing integrated system. Serial link communication operated at higher baud rates and a modular integrated navigation Kalman filter has been used in their prototype system.

Sung Wook Moon [2] has reported a loosely coupled integration technique, for implementing the GPS-INS integrated navigation system. The system is based on a high speed RISC processor with a floating-point unit. pSOS+ was chosen as the **R**eal **T**ime **O**perating **S**ystem (RTOS) and the software has three **I**nterrupt **S**ervice **R**outines (ISRs) and six tasks. Task was operated independently at different rates and the communication among the tasks was synchronized through unidirectional semaphore. Time synchronization between GPS and INS was carried out using events. Computer simulations and Van tests were carried out and the results show that the integrated navigation solution is better compared to operating GPS and IMU separately.

An integrated system, based on PC/104 embedded microcomputer operating on three power supplies is described by Shang *et al.,* [3]. Here, multiplexer based A/D conversion of inertial sensor signals is achieved using custom based data acquisition board. Real-time data calculation and rapid data exchanging is achieved using Micro-programming Controlled Direct Memory Access (MCMDA) technique.

Bridging GPS outages for tens of seconds using Crossbow's low cost inertial device, and integration with GPS information and its feasibility are described by Cao *et al.,* [4]. Results show that the position accuracy of low cost SINS/GPS is just the same as that of GPS receiver when GPS data is available, and with a position error of 10 meters after 10 seconds with complete loss of GPS signals are observed.

A highly accurate attitude determination algorithm was presented by Lichun *et al.,* [5] in which aircraft's inherent kinetic property is predicted by a linear-two-point polynomial predictor and combined with a short term highly accurate INS. On the other hand, aircraft's inertial motion is predicted using a quadratic-five-point predictor and is combined with the long-term accuracy of GPS. Linear, convex combination of linear-two-point and quadratic five-point predictors is done and the test results show that the INS attitude accuracy is improved by using updates from determined attitude of the GPS.

Dong-Hwan Hwang [6] in his work overcomes the drawbacks of loosely coupled integration by using a tightly coupled scheme [6]. Here, the raw measurements of the GPS are directly taken to Kalman filter and combined with inertial measurements. Even when less than four satellites are tracked, integrated navigation solutions are found. Various types of GPS raw measurement can be used, such as **D**ifferential **G**lobal **P**ositioning **S**ystem (DGPS) output, the carrier phase measurement, double differenced observations and attitude output from a multi antenna GPS receiver to acquire more accurate reliable navigation solutions.

D. T. Knight [7] has modularized integration software for tightly-coupled GPS/INS integration system, which can be run on Pentium PCs. In this work, integration was implemented on a high speed RISC processor with a floating point unit with pSOS+ RTOS and the software consisting of three ISRs and six tasks.

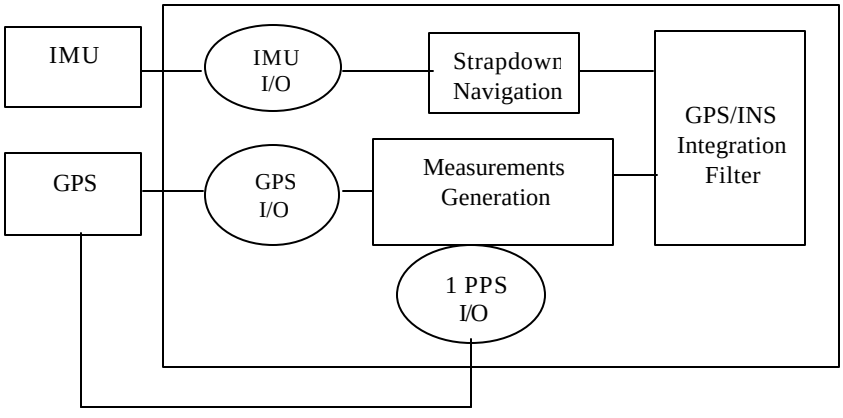



**Fig 2.1** Tightly coupled Integration

A 17-state GPS-INS integration Kalman filter was implemented [7] using an error model. Several experimental trials were carried out to evaluate the performance of the developed integration system. Integrated system performance was found to be as good as the highly accurate INS, MAPS, used in the trials of the system. Test trajectory was done for 35 minutes and the system was initially aligned for 15 minutes before navigation. Performance degradation was observed seen when the GPS signal was blocked. In spite of this degradation, much smaller errors were observed than those with the GPS.

Martin and Bruce [8], in their paper describe development of miniaturized, integrated low cost GPS-INS integration at *Boeing Guidance, Navigation, and Sensors.* In this paper they describe the coarse/acquisition (C/A) code, product architecture, and the overall design approach. Their work was based on combining a navigation processor, GPS and the integration processor in order to decrease the cost, size, weight, and power. Their architecture features a **T**exas **I**nstruments (TI) TMS320C31 DSP and three SPROC high speed signal processors. The inertial sensor data enters the processor system via memory mapped counter registers and serial ports in each SPROC. Gyro data and the accelerometer data are fed through ADC and filtering operation in each SPROC. The GPS receiver has five parallel channels of L1-only coarse/acquisition (C/A) code capability on a single board. The IMU software and integrated GPS-INS software algorithms is in the same **C**entral **P**rocessing **U**nit (CPU) TMS320C31 processor. GPS-INS interaction was accomplished using 28-error state Kalman filter. The corrections are updated by the Kalman filter at a rate of 1-Hz, and are then provided as control signals for the strapdown and navigation algorithms.

Wang Jin-Hang [9] describes the performance analysis relevant to system design and platform trajectory and/or maneuver optimization. Precise positional information from the INS was used for cycle slip detection and correction. As cycle slips are one of the limiting factors for high precision GPS positioning. Tightly coupled system was adopted and INS and GPS are treated as individual sensors and double differenced pseudo range and carrier phase measurements with fixed integer ambiguities are used to update the filter. For short range positioning, most of the systematic errors in the raw GPS measurements are eliminated during the double-differencing technique. Hence the error state vector of the Kalman filter only includes

the navigation parameters, and the accelerometer and gyroscope error states. The experimental results show that, when GPS signal blockages were present, better than half meter accuracy was achieved for up to 20 seconds. During the land based experiments, results showed that the higher the dynamic changes in North and East components, shorter the initialization time. Therefore, they proposed that a trajectory with the shape of CIRCLE is the optimal one for the initialization if there are high dynamic changes.

Salychev *et al.*, [10] describe low cost GPS-INS integration and testing system. In their paper, they assess the feasibility of using a motion sensor, integrated with GPS and DGPS information, to bridge GPS outages for tens of seconds. An algorithm was developed to integrate the GPS and motion sensor, data includes INS error damping, calculated platform corrections using GPS (or DGPS) output, velocity correction, attitude correction and error model estimation for prediction. This algorithm structure guaranteed a high level of software reliability and was implemented in GNSS-Aided Inertial Navigation (GAIN). Vehicular and aircraft test trials were conducted with the system in land vehicle mode. Simulated outages in GPS availability were made to assess the bridging accuracy of the system. The INS prediction accuracy was strongly dependent on changes in the vehicle dynamics during the DGPS gap. But, when the vehicle acceleration was changed within the prediction interval, the prediction accuracy was found to be 10 m over 10 seconds, and when the vehicle dynamics were kept constant, the results improved to 3 m over 10 seconds. For a 20 second outage, the prediction accuracy was found degraded to 10 m and 23 m for these two cases. Results also showed that a bridging accuracy of up to 3 m after 10 seconds in vehicular mode and a corresponding accuracy of 15 m after 60 seconds in aircraft mode can be obtained, depending on vehicle dynamics.

Petovello *et al.*, [11] in their paper dealt with little latency between INS and GPS information. The GPS measurements received from a GPS receiver are typically latent with respect to the IMU data, with the latency expected to be as large as half of a second. A Loosely coupled system was chosen here, this GPS/INS time offset is further compounded by the time required to process the GPS data before passing the position and velocity information to the INS filters. Two methods were dealt with this and in the first approach IMU data is buffered until appropriate GPS measurements

are received. In their second approach, once the IMU data is received, it was processed and the navigation solutions buffered in memory. Then, when GPS update was ready, it was applied to the appropriate INS solution, according to the GPS time tags. It was found that, the most current position was always the INS solution predicted from the last GPS update. Figure 2.2 gives a graphical representation of the updating process. Here the accuracy of the system depends on two things. First, the accuracy of the GPS solution will define the accuracy of the system. Second, the relative accuracy of the INS will determine the initial GPS error propagated forward in time.

Mathematically, the real time position accuracy can be expressed as:

$$s^2_{RT} = s^2_{GPS} + s^2_{INS}(I) \qquad (2.1)$$

where;

$s^2_{RT}$ is the real-time position error variance
$s^2_{GPS}$ is the GPS position error variance
$s^2_{INS}(I)$ is the INS prediction error variance as a function of time.

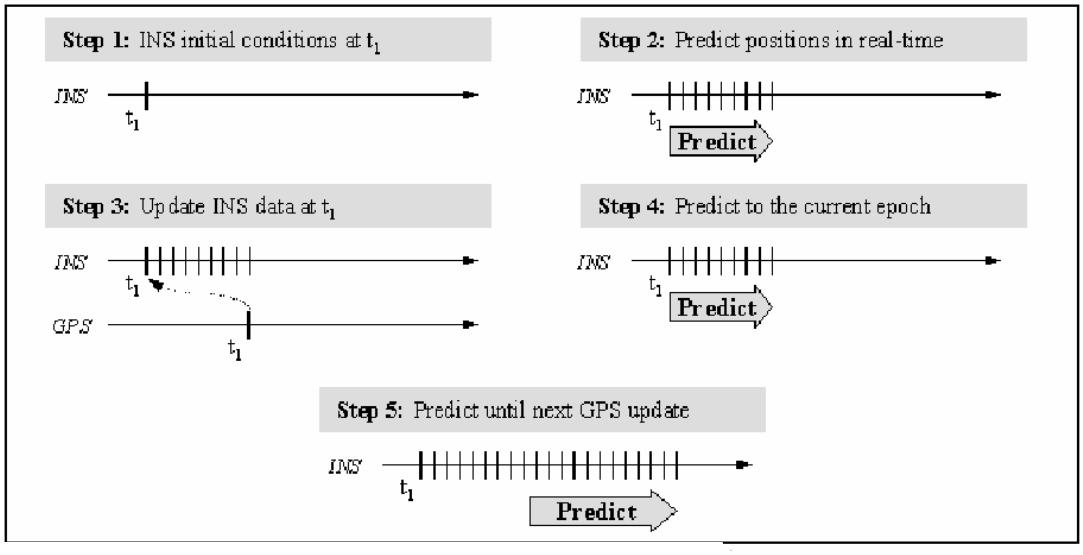



**Fig 2.2** Graphical Description of Updating the INS with Latent GPS Data [11]

The accuracy of the GPS solution was fairly known. To quantify the overall system performance, they considered analyzing the INS errors over time. Field tests performed under different operational conditions showed the system was capable of delivering position accuracies of the order of a few centimeters in real-time under ideal conditions. Under signal masking environments, accuracies better than half a

meter were expected for GPS data gaps of up to about seven seconds, depending on vehicle dynamics.

GPS-INS integration system with multiple GPS antennas provides navigation performance better than one GPS antenna [12]. The results show that, position precision is within 5m and attitude precision within $10^{o}$~ $15^{o}$ of the vehicle through the vehicle's moving test around the square road.

# Chapter 3

# GPS-INS integration

## 3.1 GPS

GPS receivers output data every one second and provide data in **N**ational **M**arine **E**lectronics **A**ssociation (NMEA) sentences format with one-second update. Each GPS satellite carries a Cesium and/or Rubidium atomic clock to provide timing information for the signals transmitted by the satellites. Each GPS satellite transmits two spread spectrum L-band carrier signals- an $L_1$ signal with carrier frequency $f_1$=1575.42MHz and an $L_2$ signal with carrier frequency $f_2$=1227.6MHz. The $L_1$ signal from each satellite uses binary phase-shift keying (BPSK), modulated by two pseudorandom noise (PRN) codes in phase quadrature known as C/A-code and P-code. The $L_2$ signal from each satellite is **B**inary **P**hase **S**hift **K**eying (BPSK) modulated by only the P-code. The **D**ilutions **O**f **P**recision calculations (DOP) are needed, in the calculations of the user position and velocity, and the best locations of satellites to be used.

### 3.1.1 GPS data

Most GPS receivers output data serially. There are 63 standard NMEA sentences, from which, most of the GPS receivers provide four to five NMEA sentences. Also, few of the GPS receivers provide proprietary sentences. These NMEA sentences will provide information about user latitude, longitude, altitude, time-of-fix, date-of-fix, number of satellites seen, Heading data, etc.,

A typical example of GPS sentence is as shown in Table 3.1:

$GPGGA,170834,4124.8963,N,08151.6838,W,1,05,1.5,280.2,M,-34.0,M,,*75

**Table 3.1** GPS sentence

| Name | Example Data | Description |
|---|---|---|
| Sentence Identifier | $GPGGA | Global Positioning System Fix Data |
| Time | 170834 | 17:08:34 UTC |
| Latitude | 4124.8963, N | 41d 24.8963' N or 41d 24' 54" N |

| Name | Example Data | Description |
|---|---|---|
| Longitude | 08151.6838, W | 81d 51.6838' W or 81d 51' 41" W |
| Fix Quality:<br>- 0 = Invalid<br>- 1 = GPS fix<br>- 2 = DGPS fix | 1 | Data is from a GPS fix |
| Number of Satellites | 05 | 5 Satellites are in view |
| Horizontal Dilution of Precision (HDOP) | 1.5 | Relative accuracy of horizontal position |
| Altitude | 280.2, M | 280.2 meters above mean sea level |
| Height of geoid above WGS84 ellipsoid | -34.0, M | -34.0 meters |
| Time since last DGPS update | blank | No last update |
| DGPS reference station id | blank | No station id |
| Checksum | *75 | Used by program to check for transmission errors |

### 3.1.2 GPS errors

The following types of errors affect the performance of GPS. The errors due to atmosphere (Ionospheric and Tropospheric) lead to phase lag in pseudorange calculation. Ephemeris errors occur when the GPS message does not transmit the correct satellite location and this affects the ranging accuracy. These tend to grow with time from the last update from the control station. Measurement noise affects the position accuracy of GPS pseudorange absolute positioning by a few meters. The propagation of these errors into the position solution can be characterized by a quantity called Dilution of Precision (DOP) which expresses the geometry between the satellite and the receiver and is typically between 1 and 100. If the DOP is less than 6, then the satellite geometry is considered good.

## 3.2 INS

Inertial Navigation refers to the determination of a vehicle's position, velocity and course with respect to earth. The navigation solution is obtained by numerically solving Newton's equations of motion using measurements of vehicle specific forces and rotation rates obtained from onboard instrumentation. The onboard sensing components consist of accelerometers and gyros, together with other platform

electronics comprise the Inertial Measurement Unit (IMU). The IMU along with the computer software and hardware, that implements the navigation solution make up the Inertial Navigation System. The INS may be mechanized in Gimbaled or Strapdown configuration.

In a Strapdown system [13], the sensors are tightly mounted to the vehicle airframe, and coordinate transformations are used along with the body-fixed acceleration and rate measurements to perform the navigation computations in the fixed frame. Strapdown systems can be subjected to higher dynamic conditions (such as high turn rate maneuvers) which can stress instrument performance. However, with the appearance of newer high quality gyros and accelerometers, strapdown inertial systems are becoming predominant mechanization, due to their low cost and reliability. There are numerous sources of error in inertial navigation systems that must be considered in evaluating system accuracy and performance. For example, the errors associated with the gyros and accelerometers typically include static and g-sensitive biases and drifts, scale factor errors, misalignment errors and random noise. Additional errors arise from calibration, initialization and transfer of alignment, computational inaccuracies and approximations in the navigation solution. Without compensation, all INS errors will propagate in time, and some errors (such as position) will typically diverge with time, others will be bounded and oscillate. The accuracy of an INS is therefore highly dependent on the sensor quality and navigation system mechanization.

### 3.2.1 Accelerometers and Gyroscopes

Inertial measurement of linear acceleration is possible with accelerometers. The signal is obtained mechanically and scaled in g[1g=9.81m/sec$^2$]. Gyroscopes (or gyros) measure rotational values without reference to external coordinates. A gyro's accuracy can be judged by its output errors. For rate sensors the most important are bias, scale factor error, noise and drift.

### 3.2.2 Noise and Sensor Errors

Most of the error sources that occur in navigation are sensor errors or random disturbances. To name a few, errors may be linear, bias and input/output nonlinearity. A linear error is proportional to the input signal. It usually exhibits some degree of nonlinearity. A bias is a constant signal on the output of a sensor, independent of the input. A bias will not change during a run, but may vary from turn-on to turn-on. A bias is modeled as a random constant.

#### 3.2.2.1 Accelerometers

For a cluster of three accelerometers with nominally orthogonal input axes, the effects of individual scale factor deviations and input axis misalignments from their nominal values can be modeled [13] by

$$z_{output} = S_{nominal}\{I+M\}z_{input} + b_z \tag{3.1}$$

where, $b_z$ represents the bias value of three sensors,
$z_{input}$ are the sensed values(accelerations or angular rates),
$z_{output}$ are the output values from the sensors,
$I$, $M$ represent the individual scale factor deviations and input axis misalignments,
$S_{nominal}$ is the nominal sensor scale factor

Or, $$z_{input} = \overline{\mathrm{M}}\{z_{output} - b_z\} \tag{3.2}$$

where $\overline{\mathrm{M}} = \dfrac{1}{S_{nominal}}\{I + M\}^{-1}$

The model [13]for combined three-accelerators is of the form given by

$$\begin{bmatrix} a_{i,input} \\ a_{j,input} \\ a_{k,input} \end{bmatrix} = \overline{\mathrm{M}}_{acc}\left\{\begin{bmatrix} a_{i,ouput} \\ a_{j,output} \\ a_{k,output} \end{bmatrix} - a_{bias}\right\} \tag{3.3}$$

where, $a_{bias}$ is the bias compensation
$\overline{\mathrm{M}}_{acc}$ (a 3 x 3 matrix) is the combined scale factor and misalignment compensation.

The non-linearities of sensors are typically modeled [eq 3.4] in terms of a MacLauren series expansion, with the first two terms being bias and scale factor. The next order term is called “g-squared” accelerometer error sensitivity, which is most common in inertial grade accelerometers.

$$a_{input} = \lambda_0 + \lambda_1 a_{output} + \lambda_2 a^2 output + ..... \tag{3.4}$$

### 3.2.2..2 Gyroscopes

After each turn-on, the general purpose gyroscope bias error model will have the form of a drift rate (rotation rate) about the gyroscope input axis:

$$w_{output} = w_{input} + dw_{bias} \tag{3.5}$$

$$dw_{bias} = dw_{constant} + dw_{turn-on} + dw_{randomwalk} \tag{3.6}$$

but $dw_{constant}$ is a known constant

$dw_{turn-on}$ is an unknown constant

$dw_{randomwalk}$ is modeled as a randomwalk constant

and $\frac{d}{dt} dw_{randomwalk} = w(t)$ where $w(t)$ is a zero-mean, white-noise process with known variance.

The gyroscope scale factor is usually specified in compensation form as $w_{input} = C_{scalefactor} w_{output}$ (3.7)

where, $C_{scalefactor}$ can have components that are constant, variable from turn-on to turn-on and drifting after turn-on.

The compensation for bias, scale factor, and input axis alignments for three gyroscopes with nominally orthogonal input axes is implemented in matrix form[13] as shown below:

$$\begin{bmatrix} w_{i,input} \\ w_{j,input} \\ w_{k,input} \end{bmatrix} = \overline{M}_{gyro} \left\{ \begin{bmatrix} w_{i,ouput} \\ w_{j,output} \\ w_{k,output} \end{bmatrix} - w_{bias} \right\} \tag{3.8}$$

where ? $_{bias}$ is the bias compensation

$\overline{M}_{gyro}$ (3 x 3 matrix) is the combined scale factor and misalignment compensation.

The nonlinearities of sensors are typically modeled in terms of a MacLauren series expansion, with the first two terms being bias and scale factor.

$$w_{input} = \lambda_0 + \lambda_1 w_{output} + \lambda_2 w^2_{output} + ..... \tag{3.9}$$

## 3.3 GPS-INS integration

INS provides self-contained independent means for three-dimensional positioning with high short-term accuracy and degrades over a period of time. The degradation is much faster for low-cost INS systems. INS provides high-accuracy three-dimensional positioning when the GPS positioning is poor or unavailable over short periods of time (e.g., due to poor satellite geometry, high electromagnetic interference, high multi-path environments, or obstructed satellite signals). In addition, the INS system provides much higher update positioning rates compared to the output rate conventionally available from GPS. The GPS signals are updated every one sec and also the GPS position errors are not good over a short term, but they do not degrade with time. The short- term position errors from the Inertial Navigation System (INS) are relatively small, but they degrade without bound over time.

Integration of GPS [13] with an Inertial Navigation System improves the quality and integrity of each navigation system: use of GPS permits calibration of inertial instrument biases, and the INS can be used to improve the tracking and re-acquisition performance of the GPS receiver. There are two error calibration techniques [13] that can be implemented in an integrated GPS/INS system: the feed forward (or open-loop) method and the feedback (or closed-loop) method as shown in Fig. 3.1.

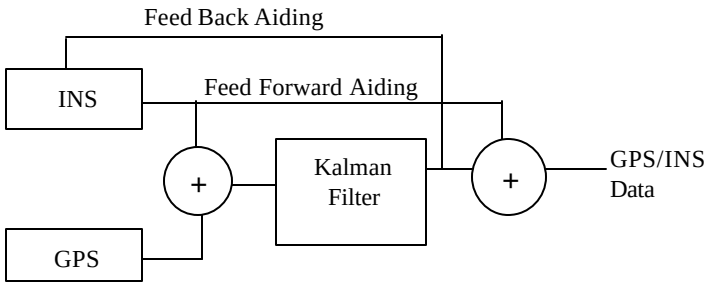



**Fig 3.1** Inertial aiding concept

There are two basic types of methods for integration of GPS and INS data in a system. These are *tightly coupled* and *loosely coupled.* With a loosely coupled integration[2,13], the GPS pseudorange measurements are pre-processed by a Kalman filter internal to the GPS receiver, which produces "GPS derived" geographic position and velocity as the receiver outputs. The block diagram for a loosely coupled

integration is shown in Fig. 3.2. Because most of the GPS receivers have a built in Kalman filter and output the "GPS derived" position and velocity outputs, we use the loosely coupled configuration in our experiments. One of the drawbacks of this is that the navigation should be performed on the INS measurements alone when the number of tracked GPS satellites is less than four.

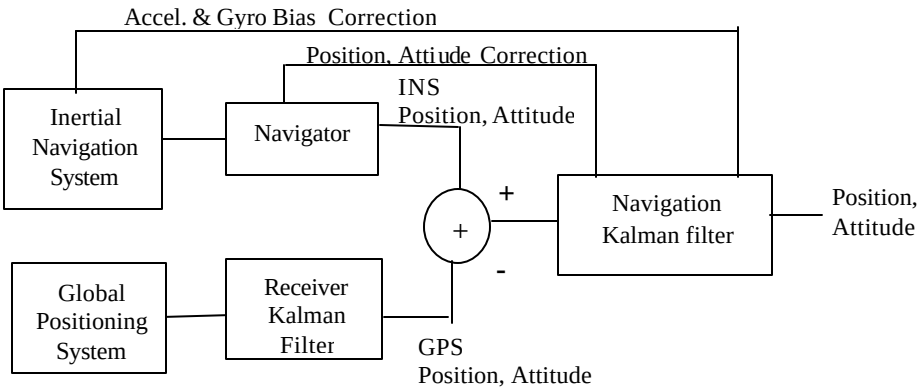



**Fig 3.2** Loosely coupled GPS-INS integrated system

The block diagram of a tightly coupled integration technique [6,13] is shown in Fig. 3.3. Internal to the GPS receiver, a set of pseudorange measurements is obtained for position estimation. A pseudorange is a distance estimate from the GPS receiver to one satellite before being error corrected. The straightforward (and most optimal) approach for integrating GPS with an INS is to directly utilize GPS pseudorange measurements from the GPS receiver to correct INS error growth with a specially designed Kalman filter that models the INS errors and the GPS measurement geometry.

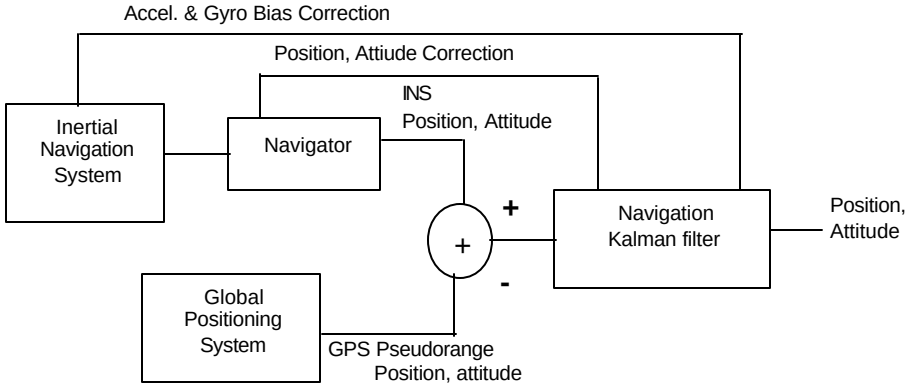



**Fig 3.3** Tightly coupled GPS-INS integrated system

Among several integration methods, the tightly coupled method provides better navigation performance than others. Separate filters for receiver and navigation processing are combined into a single integrated filter. The filter error states GPS pseudo range and delta range measurement residuals directly.

It can also give a filtered navigation result[13] even when less than four GPS satellites are tracked, but has a more complex structure.

In this integration technique as shown in Fig. 3.3, raw measurements of the GPS are directly utilized in the integrated Kalman filter to combine them with the inertial measurements. This scheme is generally known to provide an optimal navigation solution. When less than four satellites are tracked, integrated navigation solutions can be still in use. Though it is more difficult to implement, this scheme, gives better performance than other integration methods.

## 3.4 GPS-INS integration real-time issues

As per the literature review, the INS computations should be processed within 10 milliseconds, before the next INS data is considered. Also, during GPS updates (every one second), extended Kalman filter algorithm is used to correct the errors accumulated by the IMU. Thus, the extended Kalman filter algorithm should be finished 10 milliseconds before the next IMU data is accessed. This imposes severe constraint on selection of hardware, integration software, processor speed and system architecture. Besides the interfacing between the sensors assembly, control electronics circuitry and the navigation processor is of major concern. Some of the authors [1,2,6] have reported serial communication for interfacing the sensors and control electronics with navigation processor. This imposes more overheads on the processor for communicating with slow speed interfaces. The periodic tasks such as this, should be processed immediately before next update by the navigation processor. These requirements impose constraint on speed, selection of either RISC/CISC processor or a DSP, memory and interfacing techniques.

## 3.5 GPS-INS implementation

Many researchers and designers have implemented GPS-INS integrated systems with commercially available embedded computers and available data acquisition cards. Few cases of implementation of hardware architecture and the proposed system architecture based on floating point DSP (TMS320VC33) are discussed here:

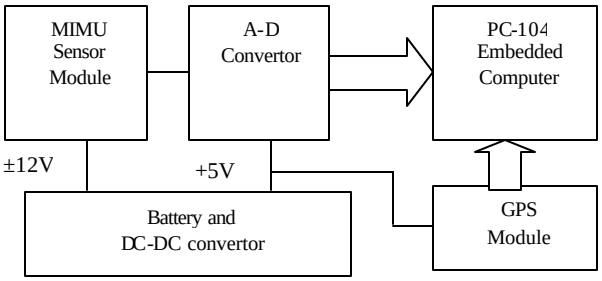



**Fig 3.4** Hardware architecture proposed by Jie Shang, *et al.*, [3]

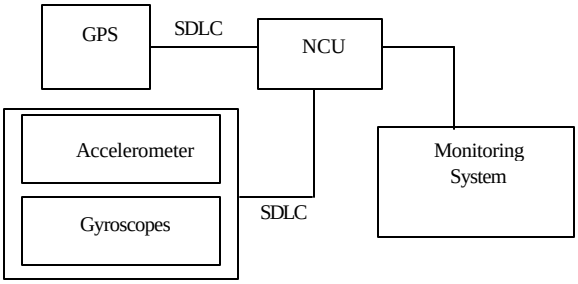



**Fig 3.5** Hardware architecture proposed by Liu [5]

The system structure shown in Fig. 3.4, is operated with three power supplies and uses PC/104 bus for data transfer from ADC to an embedded computer [3]. Input channels are selected using a multiplexer available within the ADC. In another system architecture shown in Fig. 3.5, the analog sensor signals are interfaced with NCU using serial data link [5]. NCU is based on a high speed RISC floating point processor and is operated in multitasking environment.

In the proposed system, TMS320vc33 is chosen as a navigation processor which is superior to RISC/CISC solutions in numerical performance and emulation capability. It is a floating point processor and provides upto 75 MIPS, 150MFLOPS. It has 34K words dual access SRAM, bootloader and onchip peripherals and operates on low power. This chip is inexpensive and easily available.

## 3.6 Proposed GPS-INS system integration

The proposed strap-down integrated system is developed using DSP from Texas Instruments. This system is operated on a single power supply.

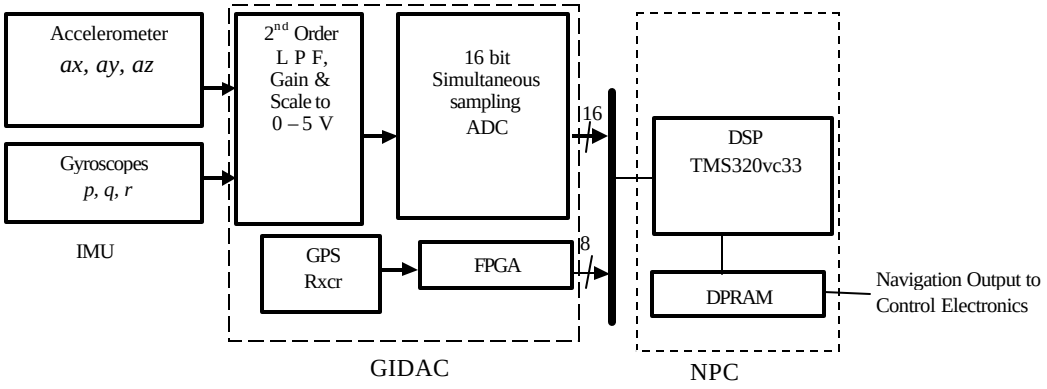



**Fig 3.6** Proposed system architecture

An integrated system with a lightweight package providing navigation system function is shown in Fig. 3.6. For better understanding, the proposed system can be divided into two blocks: (i) **G**PS and **I**NS **D**ata **A**cquisition **C**ard(GIDAC)

(ii) **N**avigation **P**rocessor **C**ard (NPC)

Following are the features of the proposed system architecture:

1. The analog signals from IMU are passed through a second order Butterworth low-pass filter, and scaled to data acquisition input range and sampled simultaneously thus preserving the phase information among all the signals at GIDAC card. There is a delay(? ) between sampling of each input signals in case of multiplexer at the input stage of ADC, and 5? delay at the instant of sixth input signal being sampled. Considering *ax, ay, az, p, q* and *r* as input channels ch1 to ch6, the position information being considered for calculation from all input signals, is not exactly the same at that instant of time[1-5]. Whereas, in the proposed system, relative information among all the signals (ch1 to ch6) is maintained by simultaneous sampling, thereby the position information calculated is more accurate compared to having a multiplexed input stage of the ADCs.
2. To relieve the main processor from computational processing overheads during slow speed serial I/Os, an FPGA based serial port interface is used. In the proposed architecture, the FPGA chip is programmed to receive the GPS data from GPS receiver, and generates a busy signal when accessing the internal

DPRAM of FPGA. This low-going busy signal interrupts the DSP processor, and the processor fetches the data from internal DPRAM of the FPGA chip.

3. Asynchronous communication is maintained between the GPS module and the NPC card using dual port RAM, thus saving the processor time during the transfer of data.
4. The INS computations and integration with GPS data using Kalman filter are carried out on floating point TMS320vc33 DSP in NPC card. Control logic signals for selecting peripheral chips are generated using **P**rogrammable **A**rray **L**ogic (PAL).
5. Asynchronous communication is maintained between the GPS/INS integrated system with the control electronics circuitry. The computed position is interfaced through dual port RAM to the external circuitry.
6. The system is compact, operates on a single power supply, and less weight.

Real-time systems are much more specific in their applications and the consequences of their failure are more drastic as compared to their general purpose counterparts. The correctness of these systems depends not only on the logical result of computation, but also on the time at which results are produced. Dedicated DSPs are used as the processor when the system has more computation intensive tasks to perform. In all these systems the system continuously interacts with the environment through various sensors. Periodic monitoring of the environment as well as timely processing of the sensed information is necessary. The next chapter describes the hardware and software implementation of the proposed scheme and operation with +5V supply.

# Chapter 4
# INS data acquisition

The signal obtained from IMU sensors is accompanied by noise, as well as vibrations from aircraft's body frame. This chapter outlines noise elimination, off-set and linearisation of sensor signals using analog technique and ADC operated with single power supply. Further, simultaneous sampling is used for preserving the phase information among all the input signals at the instant of sampling.

A low cost dual axis ±2g accelerometer, ADXL202E [14] and a ±300°/s gyro ADXRS300 [15] from analog devices are selected as inertial sensors. These two sensors are operated on a single +5V supply, which provides and gives 2.5V as their initial null value in both the sensors. The sensitivity of the accelerometer and gyro is 312 mV/g and 5 mV/°/s respectively. The signal conditioning circuit designed, eliminates the noise components and provides sufficient amplification for sensor signals.

## 4.1 Noise reduction by low pass filter

For noise elimination, low pass filtering was used. The low pass filter is a second-order Butterworth filter based on Multiple-feed back filter [16]. This filter is not immune to changes in resistor values as in case of Salen-Key filters.

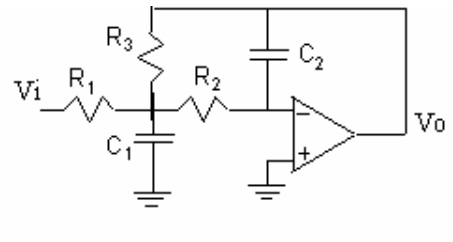



**Fig. 4.1** Multiple-feedback filter

Multiple-feedback filters shown in Fig. 4.1 utilize more than one feedback path, it exploits full open loop gain, and these are more popular single-op-amp realizations of second order responses. Resistor $R_3$ provides positive feedback; varying $R_3$ can adjust $w_0$, and $R_1$ to adjust $Q$. Where $w_0$ and $Q$ is the cut-off frequency and Quality factor of the circuit. Capacitor $C_2$ is chosen arbitrarily and

capacitor $C_1$ can be calculated as $C_1 = nC_2$, where $n$ is the capacitance spread, $n \geq 4Q^2(1+H_0)$ and $H_0$ is the dc-gain magnitude.

The resistors are given by:

$$R_1 = \frac{R_3}{H_0},\ R_3 = \frac{1+\sqrt{1-4Q^2(1+H_0)/n}}{2w_0QC_2} \text{ and } R_2 = \frac{1}{w_0^2 R_3 C_1 C_2} \quad (4.1)$$

Resistors $R_1$ and $R_3$ have same values. $C_1$ is chosen four times the value of $C_2$. This filter was simulated using Orcad package and used in the design.

## 4.2 Signal pre-processing

The signals received from sensors have less sensitivity as discussed earlier. Hence these signals are sufficiently amplified while maintaining a fixed bias voltage of 2.5V. The sensor signals are preprocessed to data acquisition range of 0 to 5V. Here, offsetting is provided through analog technique to shift the level of the signal by a predictable amount (2.5V), which restores the offset of transducer signals. The sensor signals are linearized to the input of data acquisition range 0 to 5V [Fig. 4.2]. Single-supply or dual-supply operated op-amp OP491 from analog devices [17] featuring rail-to-rail inputs/outputs is chosen. The output voltage of an op-amp swings to within mill volts of the supplies. Simulation results in Orcad and lab tests show that the signal conditioning circuit provides output swing in the range 0 to 5V, with offset of 2.5V is provided to filter op-amp and amplification stage. The signal conditioning schematic is shown in Appendix A.

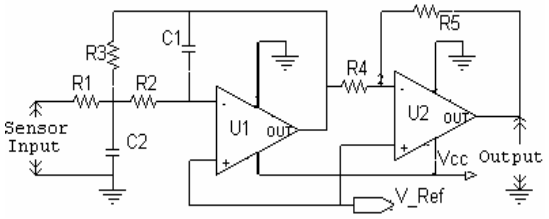



**Fig. 4.2** Signal conditioning circuit of one channel

## 4.3 Selection of ADC

Amongst the several types of ADC, Sigma-delta ADCs provide higher resolution output with lower update rate. Timing synchronisation of serial output ADCs is provided through an external clock from the host processor. Processor overhead is more in communicating with serial ADCs. General multi-input ADCs

have multiplexer at their input stage, which selects input sensor signals based on the control logic of the multiplexer. There is a certain delay between the first input signal and the last input signal of ADC chip. This gives additional phase difference between each axis of accelerometer and gyroscope. As seen in Fig 4.3, there is a delay (? ) between sampling of individual input signal in case of a multiplexer based input stage of ADC. Considering *ax*, *ay*, *az*, *p*, *q* and *r* as inputs to channels ch1 to ch6, the position information being considered for calculation from all input signals is not exactly the same at that instant of time. The relative phase delay information among the various signals (ch1 to ch6) can be maintained only by simultaneous sampling. The position information thus calculated is more accurate compared to multiplexer based input stage ADCs.

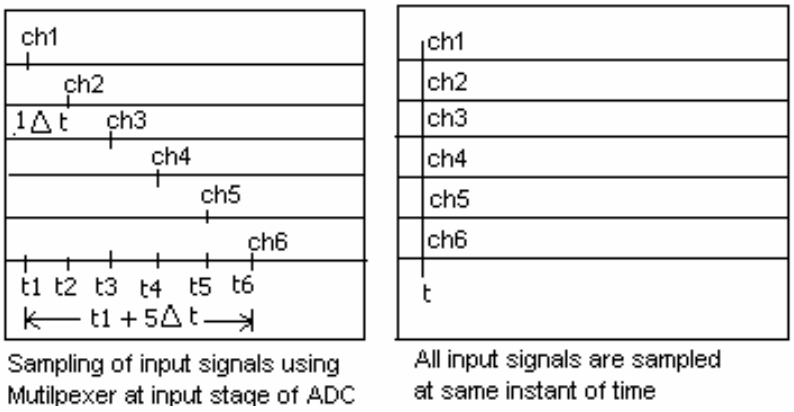



**Fig. 4.3** Comparision of Sampling of input signals using multiplexer and simultaneous sampling of input signals

Hence, 16-bit parallel output ADC ADS8364 [18] that allows simple interface to DSP from Texas Instruments is chosen. The input channels are grouped into pairs (A0, A1, B0, B1, C0 and C1) for high-speed simultaneous signal acquisition, and offers a high-speed parallel interface in cycle mode. The control signals can be selected by either hardware or by software that is accessible through the data bus and the control word. These pins are in OR configuration with the hardware pins. This ADC chip has two supply pins, +5V analog power supply, and 3.3V/5V for digital interface. The control signals are generated from the NPC card for accessing the signals. As shown in Fig 4.4, bringing */HOLDx* pin low, initiates conversion of all six channels, places all the sample-hold amplifiers in the hold state simultaneously and the conversion process is started on each channel. The output data for each channel is available as a 16-bit word from channel A0 through channel C1.

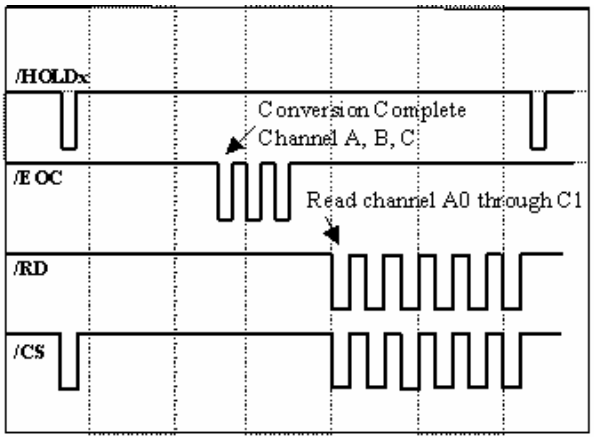



**Fig 4.4** ADC waveform obtained from ADS8364

The waveform of Fig. 4.4 is obtained with control signals byte and address signals low, and selecting control signals (*A2*, *A1*, *A0*) =110, the interface runs in cycle mode. Here data 15 down to 0 of channel A0 is obtained for first read cycle, followed by A1, B0, B1, C0 and C1 channels respectively.

The ADS8364 is a CMOS, low power, 16-bit ADC. It is capable of simultaneous sampling of 6 single-ended signals. In this board we have provision for giving 6 single-ended signals through an expansion connector (J1). The ADC has the capability to achieve a maximum of 250k samples per seconds conversion rate in continuous conversion mode [18]. The control signals of ADC can be either hardware or software controlled. The available data signal from the ADC is connected to the external interrupt /INT1 pin. The buffer within the ADC stores these converted values. The ADC control register is used to program the device into the desired mode.

Digitized data of sensor signals is interfaced with NPC. All the control signals for ADC are generated using programmable array logic PAL22V10 of NPC.

# Chapter 5

# GPS data acquisition

This chapter discusses present interfacing techniques for GPS data acquisition, with their merits and demerits. Further, realization on FPGA with its advantages over existing schemes is highlighted. FPGA implementation is explained in detail.

## 5.1 Introduction

For several years, GPS serial link interface with control electronics has been done using a microcontroller. The microcontroller is programmed to receive sentences from GPS receiver in NMEA format and the required data is extracted from this. The GPS data is connected to the serial port of the microcontroller and the baud rate is generated using internal timer of the microcontroller. The **U**niversal **A**synchronous **R**eceive **T**ransmit (UART) of microcontroller detects the start bit, receives 8 data bits, and stores the collected data as a byte, once it detects the stop bit.

### 5.1.1 Regular Program Flow adopted on Microcontroller

**Step 1:**

Program flow is constantly monitored to receive "$" (hex value = '24h'), and identifies this as start of sentence. Once this is received, it jumps into a loop, and looks out for character "G" (hex value = '24h'), character "P" (hex value = '50h'), character "G" (hex value = '24h'), character "G" (hex value = '24h') and for character "A" (hex value = '41h'). With this, the program identifies the correct sentence identifier, and continues in the loop. There are a total of fifteen comma characters "," (hex value = '54h') in this "$GPGGA" NMEA sentence.

**Step 2:**

At this point of the loop, a counter is initialized to zero and incremented once it receives character comma "," (hex value = '54h'). Every time the counter is checked with fifteen. Once the counter reaches fifteen, the counter is reset to zero.

**Step 3:**

When the counter has values equal to two, three or nine, it starts storing the received data in known memory locations (referred as LATITUDE, LONGITUDE and ALTITUDE). Once the counter value equals six, the program code checks for the immediate next character received.

**Step 4:**

When counter value equals six, the program code checks for the immediate character received. If the character received is '1', then the whole data received and stored is termed as VALID. If the character received is '0', then the whole data received and stored is termed as INVALID.

**Step 5:**

The program code continues to increment the counter up to fifteen. Once it is fifteen, it resets the counter to zero. The program code looks out for line feed character, at the end of the GPS sentence. With this the loop is terminated and this ends the $GPGGA data sentence.

Once the data is found VALID, the stored data are re-transmitted using UART of microcontroller for control electronics circuitry for necessary action. The program code continues to receive next GPS sentence every 1 second, and repeats step 1 to step 5.

**Limitations of the existing scheme**

Microcontroller is generally used in between GPS receiver and the control electronics circuitry in serial communication mode. Thus, during the transfer of data, there needs to be synchronisation between the microcontroller and the control electronics circuitry.

**Alternative:-1**

A dual port RAM (DPRAM) can be introduced in between the microcontroller and the control electronics circuitry. With this asynchronous communication is maintained between two processors. Thus, the computation overhead of the main processor of control electronics is drastically relieved.

**Limitations of Alternative-1:**

1. The microcontroller used to extract the GPS data from the GPS NMEA sentences, only makes use of the UART, and a few memory locations for storing the data. Thus full features of microcontroller are not fully used.
2. Additional components are required for interfacing DPRAM between microcontroller and control electronics circuitry. Also the full memory of DPRAM is not used. The number of chip count is more. And the total number of components required is: Microcontroller, DPRAM, and Latch 373.

**Alternative-2:**

The components, which are used for serial link interface, can be implemented on re-programmable FPGA chip.

### 5.1.2 Advantages of implementation on FPGA

The system is made more compact and reliable by using configurable **F**ield **P**rogrammable **G**ate **A**rray (FPGA). FPGA based dedicated serial protocol interface is used, to relieve the main processors from overhead of processing slow speed serial I/Os. Also, the total number of chips is reduced to one. Processing time on FPGA is less compared to other existing schemes.

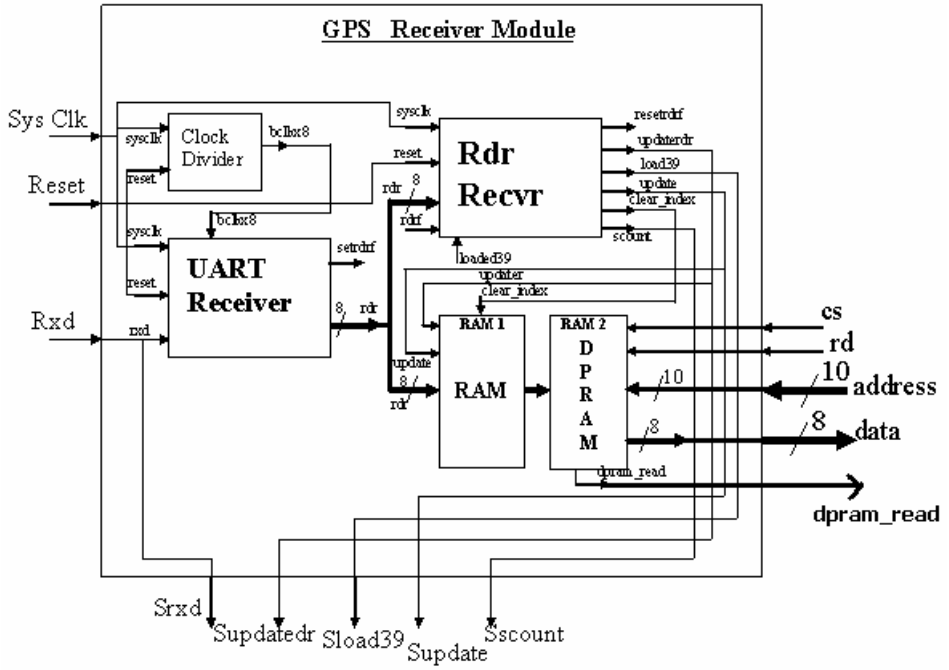



**Fig. 5.1** FPGA implementation of GPS data acquisition

## 5.2 Implementation on FPGA

The interface between serial link GPS receiver core and the main processor is implemented on FPGA. The main components of this implementation are: UART, RAM and DPRAM [Fig. 5.1]. This section explains in detail about realization of each component.

The serial communication interface, which receives and transmits serial data, is often called a UART (Universal Asynchronous Receiver-Transmitter). *RxD* is the received data signal and *TxD* is the transmitted data signal. Since there is no clock line, the data (*D*) is transmitted asynchronously, one byte at a time. When no data is being transmitted, *D* remains high as shown in Fig. 5.2. Transmission of data is marked by the start bit (keeps low for one bit time) and eight data bits, with the least significant bit first and followed by one stop bit (goes high one bit time). The number of bits transmitted per second is frequently referred to as the *BAUD* rate.

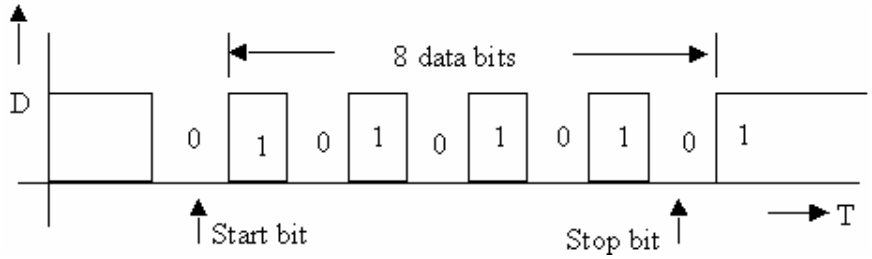



**Fig. 5.2** Standard Serial Format

When receiving, the UART detects the start bit, receives 8 data bits, and stores the data in parallel form when it detects the stop bit. Since no clock is transmitted, the UART must synchronise the incoming bit stream with the local clock.

### 5.2.1 Design of a simplified version of UART (receiver part)

The operation of UART receiver is as follows: -

1. Two 8-bit registers are used:
   (i) RSR Receive Shift Register
   (ii) RDR Receive Data Register
   Beside the registers, the two components of UART are the BAUD rate generator and the receiver control.

2. When the UART detects the start bit, it reads in the remaining bits and shifts them into the RSR [Fig.5.3].

3. When all the data bits and the stop bit have been received, the RSR is loaded into the Receive data register and RDRF flag is set high.
4. The microcontroller checks the RDRF flag, and if it set, the RDR is read and the flag is cleared.

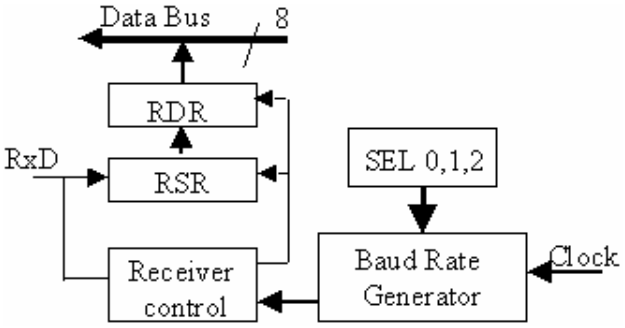



**Fig. 5.3** UART connected to 8-bit data bus

The baud rate generator divides down the system clock to provide the bit clock (*Bclk*) with a period equal to one bit time and also *Bclkx8*, which has a frequency eight times the *Bclk* frequency.

The bit stream coming in on *RxD* is not synchronised with the local bit clock (*Bclk*). If it is attempted to read *RxD* at the rising edge of *Bclk*, this could have a problem if *RxD* changes near the clock edge. This could lead to setup and hold time problems. If the bit rate of the incoming signal differs from *Bclk* by a small amount, it may lead to reading some bits at the wrong time. Ideally, the data bit value should be read at the middle of each bit time for maximum reliability as shown in Fig. 5.4. When *RxD* first goes to 0, the next data is read after four *Bclkx8* periods, and with this, the data is read near the middle of the start bit. The data is read once every eight *Bclkx8* clocks the stop bit has been read.

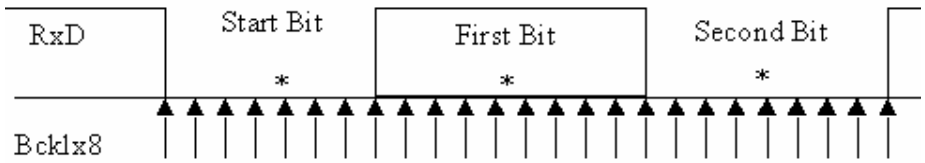



**Fig. 5.4** Sampling RxD with Bclkx8

### 5.2.2 State Machine of UART receiver

The state machine(SM) chart for the UART receiver is described here. Two counters are used. *Ct1* counts the number of Bclkx8 clocks. *Ct2* counts the number of bits received after the start bit. In the *IDLE* state, the SM waits for the start bit (*RxD='0'*) and then goes to the *start detected* state. The SM waits for the rising edge of Bclkx8 and then samples *RxD* again. Since the start bit should be '0' for eight *Bclkx8* clocks, we should read 0. *Ct1* is still 0, so *Ct1* is incremented and the SM waits for *Bclkx8*. If *RxD*='1', this is an error condition and the SM clears *Ct1* and resets to the IDLE state. Otherwise, the SM keeps looping. When *RxD* is '0' for the fourth time, *Ct1*=3, so *Ct1* is cleared and the state goes to receive data mode. In this state, the SM increments *Ct1* on every rising edge of *Bclkx8*. After the eighth clock, *Ct1*=7 and *Ct2* is checked. If it is not 8, the current value of *RxD* is shifted into RSR, *Ct2* is incremented, and *Ct1* is cleared. If *Ct2*=8, all 8 bits have been read and it should be the middle of the stop bit. If *RDRF* =1, the receiver has not yet read the previously received data byte, if *RxD='0'*, the stop bit has not been detected properly. If *RDRF* =1, RDR is loaded from RSR. In all cases, RDRF is set to indicate that the "receive" operation is completed and the counters are cleared.

The SM chart for the UART receiver is shown in Appendix B. The receiver contains the RDR and RSR registers and the receive control. RDR can drive data onto the data bus. The first process represents the combinational network, which generates the next state and control signals. The second process updates the registers on the rising edge of the clock. The signal *Bclk_rising* is '1' for one system clock time following the rising edge of *Bclkx8*.

## 5.3 Baud rate generator

Three bits in the SCCR are used to select any of the eight-BAUD rates. It is assumed that the system clock is 8 MHz and it is required to generate different BAUD rates (300, 600, 1200, 2400, 4800, 9600, 19200 and 38400). The maximum Bclkx8 frequency needed is 38400x8=307200. To obtain this frequency, it is required to divide 8 MHz by 26.04. Since divide by an integer is possible, it is required to either accept a small error in the BAUD rate or adjust the system clock frequency downward to 7.9877 MHz to compensate.

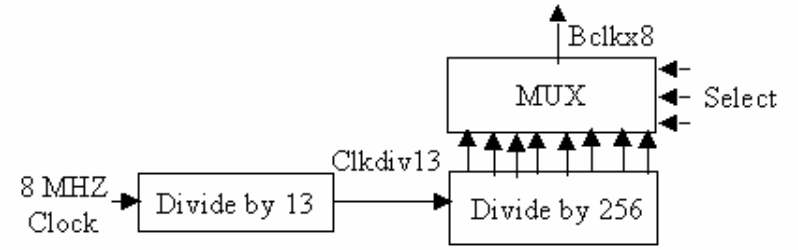



**Fig. 5.5** Baud Rate Generator

Fig. 5.5 shows a block diagram of the baud rate generator. The 8-MHz system clock is first divided by 13 using a counter. The counter output goes to an 8-bit binary counter. The outputs of the flip-flops in this counter correspond to divide by 2, divide by 4,,.., and divide by 256. One of these outputs is selected by a multiplexer. The MUX select inputs coming as the lower 3 bits. The MUX output corresponds to Bclkx8. Assuming an 8-MHz clock; the frequencies generated are given in Table 5.1.

The first process increments the "divide-by-13" counter on the rising edge of the system clock. The second process increments the "divide-by-256" counter on the rising edge of Clkdiv13. A concurrent statement generates the MUX output, Bclkx8. To complete the UART design, we need to interconnect the two components that have been designed.

**Table 5.1** Selection of different baud rates

| Select Bts | BAUD Rate |
|---|---|
| 000 | 38462 |
| 001 | 19231 |
| 010 | 9615 |
| 011 | 4808 |
| 100 | 2404 |
| 101 | 1202 |
| 110 | 601 |
| 111 | 300.5 |

## 5.4 RDR_receiver

This part of is based on SM chart. As explained earlier, the GPS receiver provides either NMEA sentences or the proprietary sentence. This design of RDR_receiver is approached based on the sentences received from GPS receiver.

### 5.4.1 NMEA Sentences

Firstly, the NMEA sentences are considered and the SM chart is shown in Appendix B. A counter is used, to count the number of commas. Counter counts the number of commas received after start of the GPS sentence identifier.

In the "check_dollar state", the SM waits for the first character '$'dollar (hex value=24), and then waits to receive character 'G', character 'P', character 'G', character 'G' and character 'A'. Once character '*' is received, the SM loops to "check_sum". Counter is incremented on every character received, until the character '*' is received. With this, the counter is reset to zero. The SM loops to "Receive_Data" state and starts updating the received data to a temporary RAM. Also, this SM does XOR calculation on the received data concurrently. In the check_sum state, the received data is stored and compared to the calculated check_sum done. Once the check_sums are found equal, the data stored in the temporary RAM is updated to the DPRAM. While updating the data to the DPRAM, a busy signal is initiated, which is used as an interrupt signal for the processor.

### 5.4.2 SIRF sentences

Secondly, if SIRF sentences are considered, the SM chart is shown in Appendix B. A counter is used, to count the number of characters received. The counter counts the number of character received after the start of the SIRF sentence identifier.

In the "check_A0" state, the SM waits for the first character 'A' (hex value= 41), and then waits to receive character '0', character 'A', character '2', character '0', character '0', character '2' and character '9'.

Once character '*' is received, the SM loops to "check_sum". Counter is incremented on every next received, until the character '*' is received. With this, the counter is reset to zero. The SM loops to Receive_Data state and starts updating the received data to a temporary RAM. Also, this SM does XOR calculation on the received data

concurrently. In the "check_sum" state, the received data is stored and compared to the calculated check_sum done. Once the check_sums are found equal, the data stored in the temporary RAM is updated to the DPRAM. While updating the data to the DPRAM, a busy signal is initiated, which is used as an interrupt signal for the processor.

## 5.5 Software Tools

Most of the software tools available in market provides flexibility in coding. The user doesn't need to know either VHDL or Verilog. The Xilinx web pack environment provides all the basic facilities required to implement, test and synthesize a complex logic design on XILINX FPGA family of devices. It has:

1. Project Navigator with schematic editor, state diagram editor and compiler for VHDL and VERILOG.
2. Model Sim for simulation and testing.
3. Test Bench facility.

Other facilities include editing the **U**ser **C**onstraints **F**ile (UCF) for logic synthesis on FPGA Kit.

### 5.5.1 Spartan II Kit

The Spartan-II trainer kit MXSFK-2008-002 is useful in realizing and verifying various digital designs using VHDL and Verilog. The Spartan-II family is a second generation high volume production FPGA solution. Devices in this family are available up to 200k gates with up to 200MHz system performance operated at 5V supply.

User can construct HDL code and verify the results by implementing physically into the target device (FPGA). With the help of this, we can simulate/observe various input and output conditions to verify the implemented design.

# Chapter 6

# Navigation Processor Card

NPC is the main block of the GPS-INS integrated system. The hardware design and system software flow of the NPC card is described in this chapter. As discussed earlier, due to real-time issues involved, floating point DSP from Texas Instruments is chosen for our design. The board features, hardware details, its programmability and interfacing with GIDAC card are also explained in this chapter.

## 6.1 HARDWARE

The INS computations and Kalman filter algorithm is simulated on a TMS320vc33 DSP running at 150 MHz. TMS320vc33 can perform parallel multiply and **A**rithmetic **L**ogic **U**nit (ALU) operations on integer or floating point data in a single cycle [22].

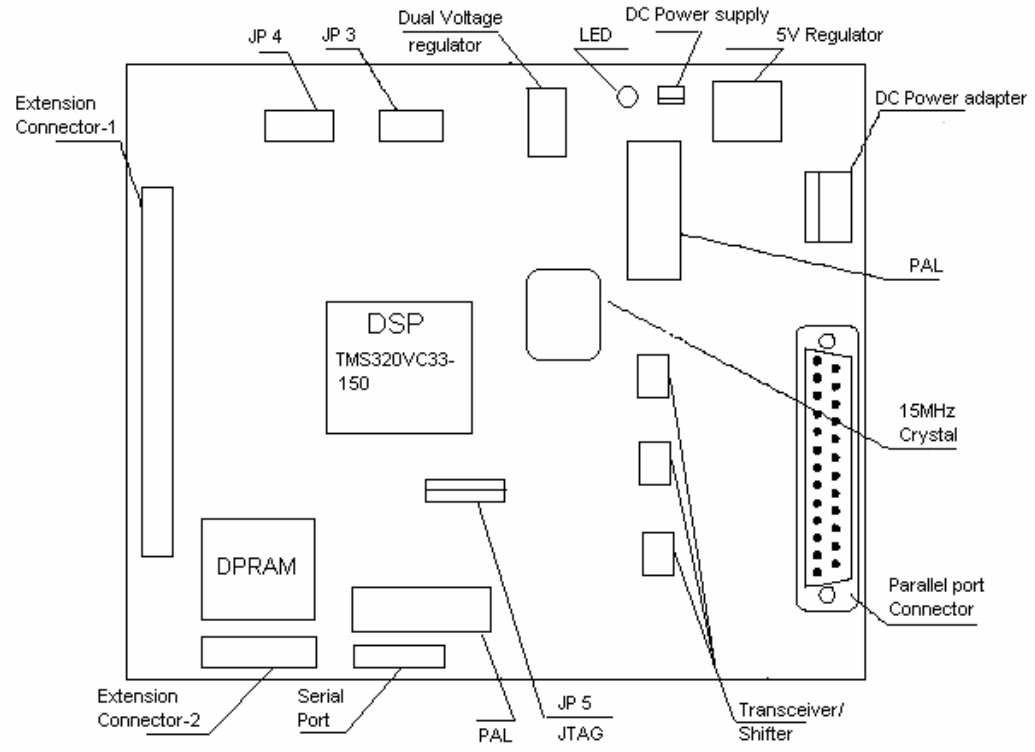



**Fig. 6.1** NPC board layout

The processor's internal bus and instruction set have the speed and flexibility to execute up to 150 MFLOPS. NPC is a single board with DSP processor, DPRAM and the control circuitry. The NPC board can communicate with the host PC either through parallel port or **J**oint **T**est **A**ction **G**roup (JTAG) cable. The applications are developed on host PC, downloaded and run on the NPC board in DOS emulator mode. The NPC board starts up by responding to a host reset command and bootloading a communication kernel. The communication kernel here provides the necessary I/O for interfacing the NPC board and the host system. Also the NPC card can be programmed using JTAG connector in **C**ode **C**omposer **S**tudio (CCS) environment. The detailed schematic diagram of the NPC board is given in Appendix C.

### 6.1.1 DSP

The TMS320vc33 device is a 144 pin quad flat-pack, capable of operating at a maximum internal clock frequency of 150 MHz. Presently, the DSP is operated with an external clock using 15 MHz crystal. The CLKMOD0 and CLKMOD1 pins of the DSP are used for selecting the clock multip lier available on jumper JP3. By setting pins CLKMOD0 and CLKMOD1 high, a multiplication factor of 5 is selected and makes the internal clock frequency 75 MHz. The micro computer/processor mode select pin MCBL/MP and the EDGEMODE pins are brought to the jumper JP4. The NPC board can hence select either the processor in microprocessor mode or the microcontroller mode. For INS computations and Kalman filter algorithm development the NPC board is selected in the microcontroller mode by setting the pin high. The edge mode is made high to enable the interrupt edge mode detection. The reset signal comes from the regulator IC (POW_RST) and also from the host PC through the parallel port P_RESET. Most details of the schematic are provided in Appendix C.

### 6.1.2 Po wer Supply

The power supply required for the NPC board is 1.8 V, 3.3V, and 5V. The DC power is fed either through a 9V DC adapter or can be operated directly with 5V DC supply. JP6 jumper is used to select either 9V adapter through DC jack or with 5V power supply. The input fed through the DC adapter is first regulated at 5V by a three terminal 5V regulator (TL7805C). The 3.3V and 1.8 V is derived using a dual voltage

regulator TPS767D318 [23]. This IC also provides the power on reset required by the DSP. The details of the power supply circuit can be referred to in Appendix C. The DSP chip requires 3.3V (DVDD) for the I/O pins and 1.8V(CVDD) for the DSP core. PAL and DPRAM ICs require 5V supply. 3.3V and 5V are also used by the level shifter and translator.

### 6.1.3 Host hardware interface

The host interface connects the TMS320VC33 parallel bus to the host's parallel printer port. The hardware interface consists of [Appendix-C]:

- A programmable array logic (PAL22V10)
- Two high-speed octal transceiver cum level shifters with tri-state outputs(74LS4245)
- JTAG port

The PAL determines when TMS320vc33 is accessing the host interface by using the STRB, and MSB of address signals. PAL provides three outputs P_RDY, P_INT2, and OE that are used in the control of data flow and also chip select signal DPRAM_CS and write data signal WR_DPRAM for DPRAM for transferring data to control electronics. The 74LS4245 level shifter cum transreceiver buffers data and level shifts from 5V to 3.3V and vice versa between the PC parallel port and the 'VC33 parallel bus. Here the interface supports the following transfers:

- The 8-bit bi-directional mode that allows faster transfers on parallel printer ports which support bi-directional transfers.
- Unidirectional printer ports support an 8-bit transfer from host to the VC33 while supporting 4-bit transfers from 'VC33 to the host.

### 6.1.4 DPRAM Interface

DPRAM provides asynchronous communication between the GPS-INS integrated system and the control electronics. IDT7130 is a high speed 1K x 8 DPRAM [24]. It is TTL compatible and has a PLCC package. This chip is operated on +5V and has two independent ports, separate control signals and I/O pins. This chip gives /BUSY signal when both the ports access the same location. It is memory mapped with the DSP of NPC. The control signals are generated through PAL. The

DPRAM can be accessed through a 20 pin extension connector-2 for easy interface with control electronics circuitry.

### 6.1.5 Additional Connectors/Jumpers on NPC board

This prototype NPC board has two extension connectors, jumpers for selecting serial interface with DSP, and additional test pads on the board for easy access to /PAGE3 to /PAGE0. This board has two extension connectors, 50 pin and 20 pin respectively. The 50 pin extension connector-1 is provided to interface the GIDAC card. Buffered data bus (D0-D15), address lines (A0-A9) and control signals are brought out on this connector. The expansion connector in addition to the data and address bits has timer/counter signals. Easy access to serial port of the DSP is provided through jumpers on the edge of the NPC card. Other control signals of the DSP for debugging can be accessed easily.

## 6.2 Development Tools

During development of NPC, the following development tools are used for programming. PALASM from AMD, Code Composer Studio 3x4x from TI, and vc33 debugger developed at IIT are discussed.

### 6.2.1 PALASM

Based on combinational logic, PAL22V10 chips were programmed using PALASM. PLDs have replaced the use of TTL decoders for address selection. Complex control logics involving combinational circuits, flip-flops, counter and other decoding logics can be very efficiently implemented on these chips. To program the PALs as per our needs AMD has developed a software called PALASM. All the logics to be fused inside PAL are programmable using AND array driving a fixed OR array matrix available inside.

When PAL chip is programmed, appropriate fuses are blown and the corresponding logic gates are implemented. So we finally arrive at a unique check sum for a particular program. This makes it easy to find out whether the IC is in good condition or not.

### 6.2.2 Code Composer Studio (CCS)

Code Composer [25] is a software program for high-level TI C and assembly language which supports high-performance, JTAG hardware assisted debugging. It directly runs on the target DSP by gaining access to the internal register set, peripherals, and bus of the target board in order to load, run, and debug applications. Also integrated into the CCS package is a code management subsystem for editing files as well as creating and compiling DSP projects. The programs included have a code composer project file (*.MAK) and workspace file (*.WSP) associated with them which may be used to recompile the example. CCS supports floating point C compiler toolset for the TMS320C3x/4x family. The compiler runs on *Windows* as a cross compiler, generating executable applications for the DSP processor, which are then downloaded and executed using the other tools. The compiler is ANSI C compatible and supports nearly all standard C functions. Assembly language may also be mixed with C code for higher performance where required. Typical application like this will consist of one or more C (*.C), header (*.H), and assembly language (*.ASM) source files, as needed. Additionally, target program generation requires use of a linker command file (*.CMD) which specifies the memory map for the target and optionally includes commands defining the libraries to be linked to the final application. This section discusses the tools used in the package.

#### 6.2.2.1 DSP Software Simulator

Simulator is a software program that simulates the TMS320c3x microprocessor and microcomputer modes for effective software development and program verification in non-real time. With this simulator, we can debug without target hardware. Time critical code, as well as individual portions of the program, can be tested. The clock's counter allows loop timing during code execution. Cycle counting displays the number of clock cycles in a single-step operation or in the run mode. The simulator uses the standard C/assembly source debugger interface and allows debugging code in C, assembly or both.

#### 6.2.2.2 DSP Emulator

Emulators are powerful and high speed tools used for system-level integration and debugging. They are user-friendly, and support hardware development on the target processor. Access is provided to every memory location and register of the target processor through JTAG cable connector. An optimizing compiler for

TMS320c3x device translates ANSI-standard C language files into highly efficient assembly language source files, which are then input to a TMS320c3x assembler/linker. Ability to link C programs with assembly language routines, allows coding of time-critical functions in assembly language.

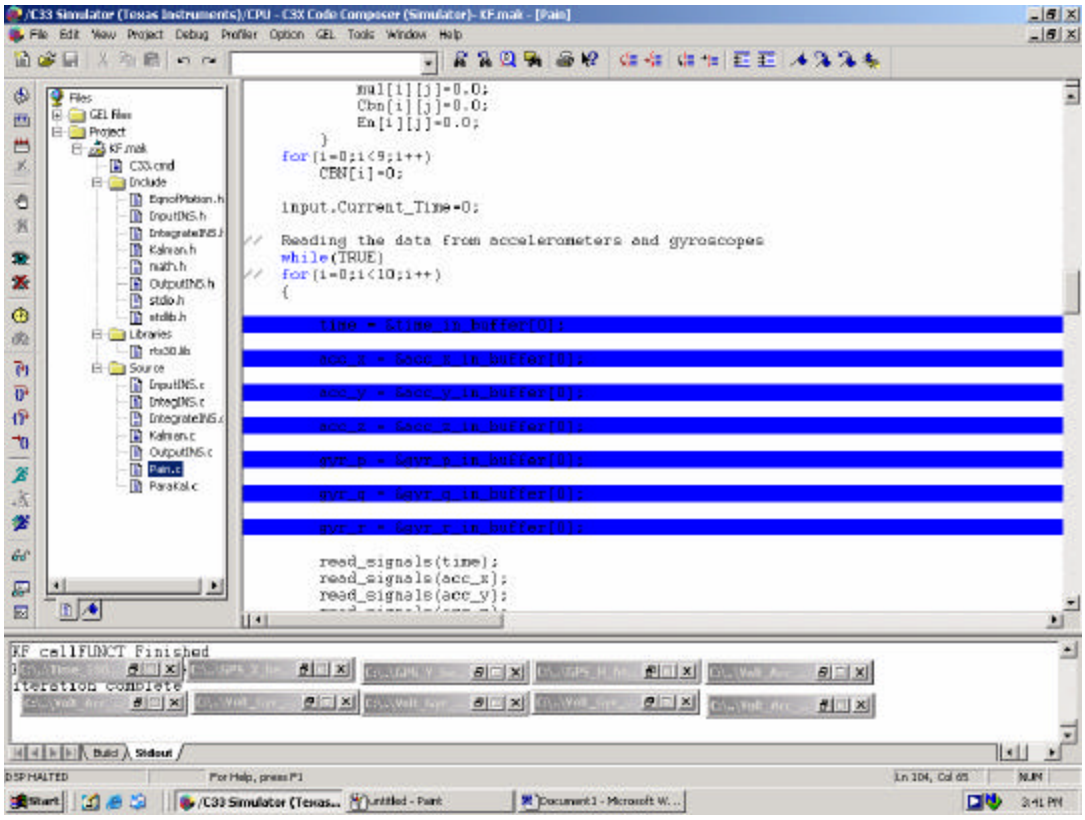



**Fig. 6.2** Snap shot of the code composer studio

### 6.2.3 vc33 Debugger

The vc33 debugger consists of software tools, integrated to work together to provide a complete DSP design environment for DSP boards. vc33 debugger is based around c31 debugger environment. The assembler, cross compiler runs under DOS environment, generating executable applications for the DSP processor which are then downloaded and executed using other tools. The target program requires linker command file (.CMD) which specifies the memory map for the target and optionally includes commands defining the libraries to be linked into the final application. If more than one source file is used in the project development, once after compilation, the (.OBJ) files can be appended in the corresponding linker command file. The linker generates an output file (.OUT) which is downloaded into DSP through parallel port connector.

## 6.3 System flow diagram

This section presents the software flow for INS computation and Kalman filter algorithm. The sensor data is generated using flight dynamic box (FDC) in Matlab for known trajectories [29]. These values are stored on hard drive of PC. The collected data is read from the hard drive and run our program taking this as input, as if the collection were happening in real-time. The software flow for GPS-INS integration code has several subprograms and header files, along with assembly coding for initialisation of DSP and its peripherals [Figs. 6.3 and 6.4]. Timers *Timer0* and *Timer1* are configured using *Timer Global control* registers and *Timer Period* registers in the initialisation routine. In the interrupt vector table, /INT1 and /INT0 is defined as interrupts from IMU and GPS data acquisition. Interrupt /INT0 received from FPGA chip (GPS update every one second) is considered as highest priority. A variable “count=2” is initialized for counting EOC (/INT1) signal from ADC. Variables “Start_INS” and “GPS_available” are set low. *Timer0* is dedicated for generating clock of 5 MHz (ADC) and *Timer1* for generating interrupts at a frequency of 100Hz update. These timers are enabled by configuring *Timer Count* register.

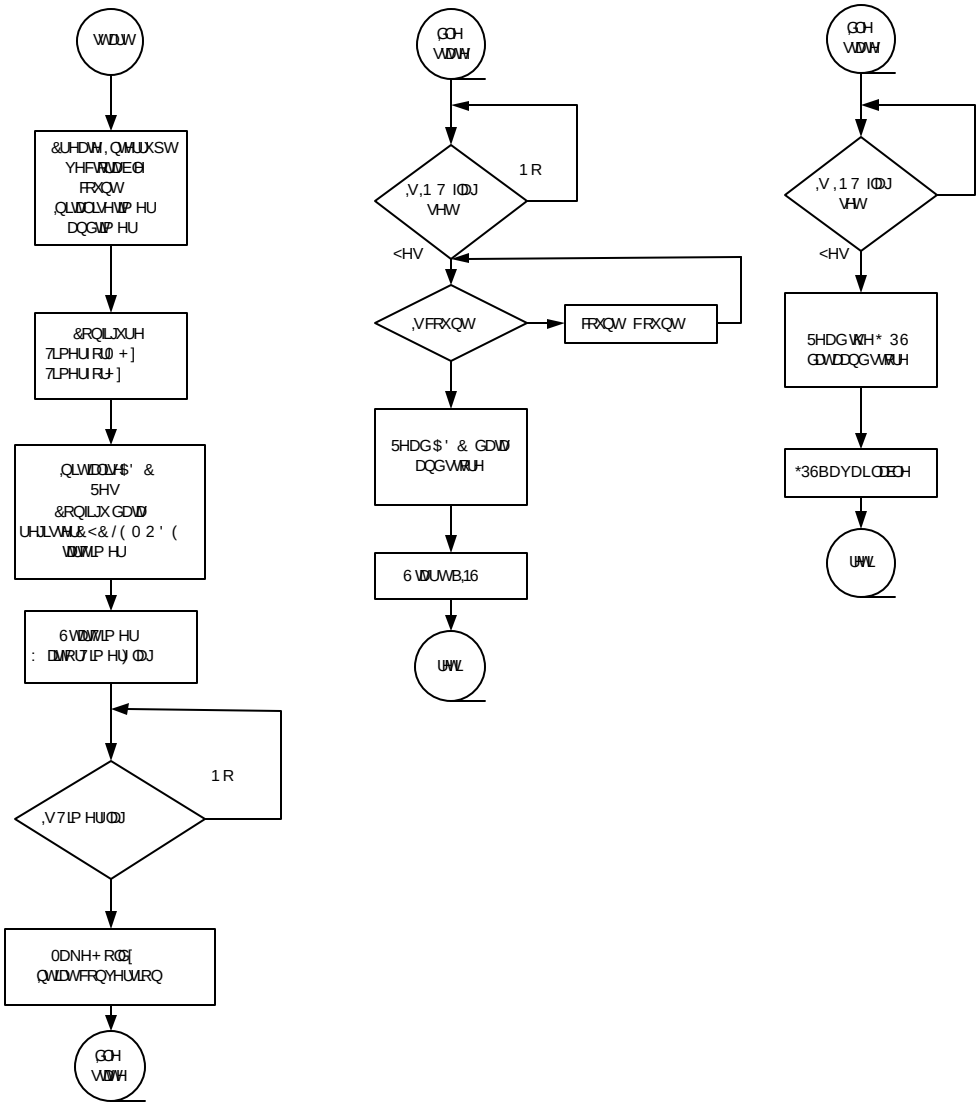


**Fig. 6.3(a)** Program flow of GPS-INS integration

The ADC chip is initialized by giving software reset and configured to operate in CYCLE MODE. *Timer0* is run, thus clock is provided to ADC continuously. *Timer1* generates interrupts every 100Hz for INS time step. On every *Timer1* overflow flag, the /HOLDx signals are made low, thus making ADC sample all six channels simultaneously. The main program waits in an infinite loop (*IDLE* mode). Based on the interrupts, the respective ISRs are enabled.

- When EOC of ADC (/INT1) occurs, the data of all six channels are stored. The Start_INS is set high. The program returns from /INT0 ISR.

- When GPS_read of FPGA (/INT0) occurs, the data is read from internal DPRAM within FPGA, variable GPS_available is set high and the program returns.

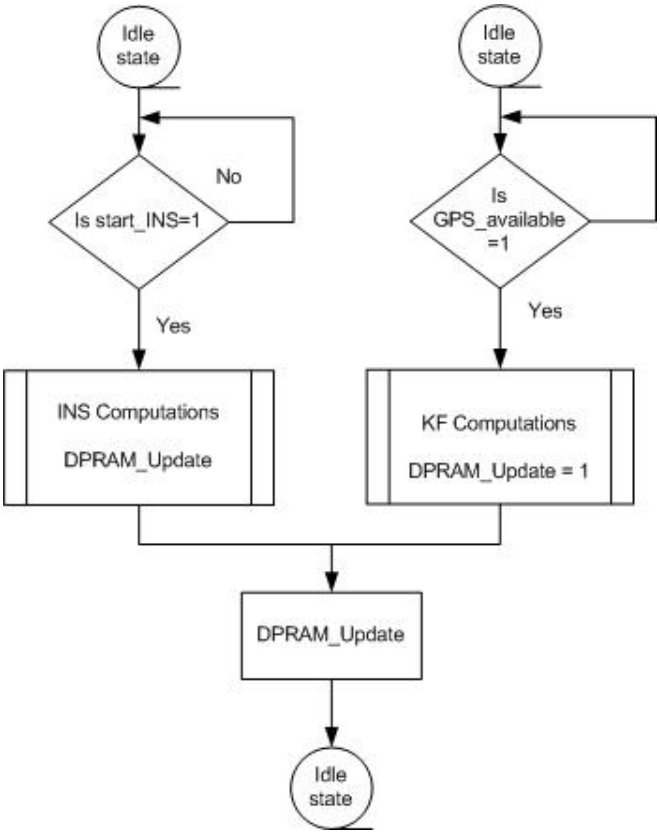



**Fig. 6.4(b)** Program flow of GPS -INS integration

In the main loop, INS computations and KF computations are performed when "Start_INS" bit and "GPS_available" is set high. The actual implementation of the filter in the program was done using a set of matrix functions built on standard ANSI C routines. ANSI C is used to aid in portability should the need ever arises to use the software on another platform. For example, one application may involve using the filter in a DSP or embedded microprocessor in a portable system, which would most likely require ANSI C language compliance to compile.

### 6.3.1 INS computation

Sensor digitized inputs are typically converted into meaningful values such as meters, degrees/sec, etc. The parameters that are used for these conversions are subject to errors and require correction for effective filtering operation. As shown in Fig. 6.4, the gyroscopes data are integrated by Euler parameter integration and Euler angles $(f, ?, ?)$ are calculated. Data from accelerometers are integrated to obtain $U, V, W$. The velocity components with respect to body frame are converted to local navigation frame using $DCM$. The velocity components $V_N$, $V_E$, $V_D$ obtained are integrated again to get the position X, Y, and Z along the three axes of local earth frame.

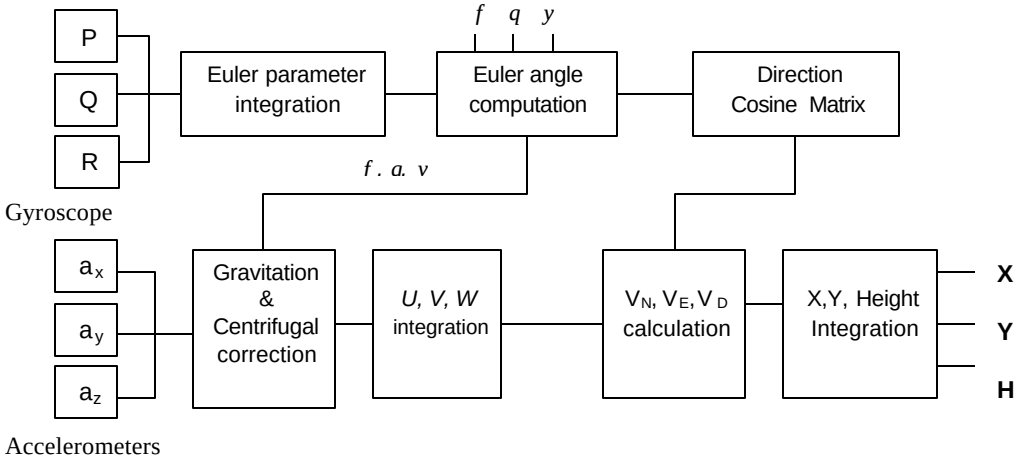



**Fig. 6.5** INS computations

We use Kalman Filter in this project only to determine the (relative) position from the inertial sensors. The most primary interest is in the parameters that affect the performance of the filter.

### 6.3.2 Kalman filter computation

A Kalman filter can be used to produce the optimal state estimate of a system given a set of measurements and relationships between the measurements and the state vector. It can be applied when the relationship between the state vector and the measurement vector can be written as:

$$X(k+1) = A(k)X(k) + w(k) \quad (6.1)$$

$$Y(k) = CX(k) + v(k) \quad (6.2)$$

where $X(k)$ the state vector is at time step $k$ and $Y(k)$ is the measurement vector at time step $k$. $A(k)$ is the transfer matrix and $C$ is the observation matrix. $w$ and $v$ are independent, white Gaussian noise variables representing process and measurement noise.

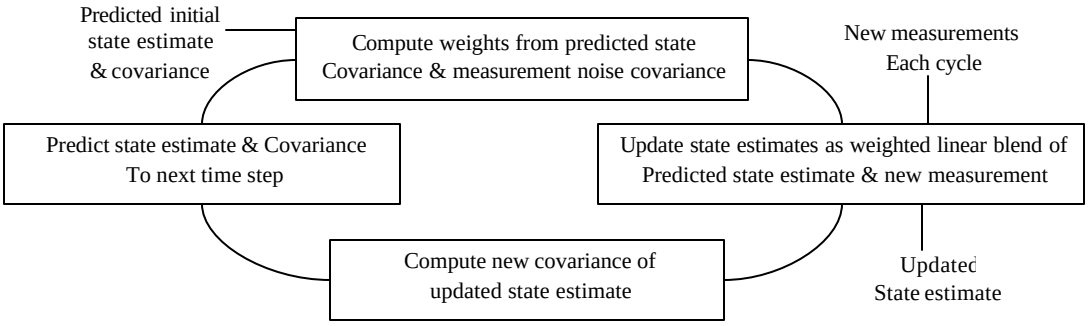



**Fig. 6.6** Kalman Filter flow diagram

At each cycle, the state estimate is updated by combining new measurements with the predicted state estimate from previous measurements. In our project we use a nine state Kalman filter [29] for correction of INS values in feedback mode. Each time a new inertial data is received in the system (at approximately 1sec) the Kalman filter is run to produce an optimal state estimate given the measurement information. During each iteration, updating the filter is accomplished in the following way. They are:

- Predication of covariance matrix of states

$$P_1(k) = A(k-1)P(k-1)A^T(k-1) + Q \tag{6.3}$$

- Calculation of Kalman gain matrix

$$K(k) = P_1(k)C^T\left(CP_1(k)C^T + R\right)^{-1} \tag{6.4}$$

- Update of the state estimation

$$\hat{X}(k) = A(k-1)\hat{X}(k-1) + K(k)\left[Y(k) - CA(k-1)\hat{X}(k-1)\right] \tag{6.5}$$

- Update of covariance matrix of states

$$P(k) = P_1(k) - K(k)CP_1(k) \tag{6.6}$$

where $P_1(k)$ is the predicted error covariance matrix at time $k$, $K(k)$ is the Kalman gain matrix at time $k$, and $P(k)$ is the updated error covariance matrix at time $k$. $R$ is the measurement noise matrix, which contains the expected covariance of the measurement data. $Q$ is the state noise matrix, which contains the expected covariance

of the state matrix values. Essentially, *R* is a measure of confidence in the measurements and *Q* is a measure of confidence in the state estimate. That is, if an extremely precise instrument were being used to quantify a particular measurement variable, the corresponding value in the measurement noise matrix, R, would be very low. Conversely, if a noisy instrument were being used, the corresponding value in R would be very high,

## 6.4 Prototype GPS – INS integrated system

The proto-type GPS-INS integrated system operated on either 9V DC power supply adapter (for bench tests) or 5V DC power by selecting jumper JP6. System is based on TMS320vc33 DSP running at 150 MHz. TMS320vc33 can perform parallel multification and ALU operations on integer or floating point data in a single cycle [22]. The processors internal bus and instruction set have the speed and flexibility to execute up to 150 MFLOPS. This system has two blocks:

(1) GIDAC is interfaced with NPC using extension connector-1. This card basically performs data acquisition of IMU and GPS signals. Analog inputs from the sensors are connected to the system at location J1. And GPS signals from GPS receiver is connected to FPGA. The digitized signal from ADC and FPGA are connected to NPC card through the extension connector-1.

(2) NPC is a single board with DSP processor, DPRAM and the control circuitry. Control signals for GIDAC block, DPRAM and communication with host PC are generated by PAL. The NPC board can communicate with the host PC either through a parallel port or JTAG cable. The applications are developed on the host PC, download and run the program on the NPC board. The NPC board starts up by responding to a host reset command and bootloading a communication kernel. The communication kernel provides the necessary I/O for interfacing the NPC board and the host system. Also, the NPC card can be programmed using JTAG connector in CCS environment.

## Key Features of the Proto-type GPS-INS integrated system:

1. Parallel port – to communicate with the host PC for download of communication kernel, program code and debug facility (NPC)
2. JTAG port – programming using emulator (NPC)
3. Serial port – for interfacing with serial port devices or used as general I/O Pins (NPC)
4. JP6 – 9V DC power supply adapter or 5V DC power supply (NPC)
5. JP3 – Clock mode select [x1, x5] (NPC)
6. JP4 – Select Microprocessor/Microcontroller mode and edge interrupt select (NPC)
7. Low power reset to system (NPC)
8. Extension connector-2 – interface with control electronics (NPC)
9. J1 – Input six analog signals (GIDAC)
10. GPS signal interface (GIDAC)
11. Weight of the System- 183 gms
12. Size of the Board

    NPC – 15`` x 12.5``
    GIDAC – 4.5`` x 3.5``
13. Power

    Active = 1.7 W (300mW+450mW+550mW+400mW)

    Idle = 0.062W (50mW + 10mW+ 1mW +1mW)

The key features of the TMS320vc33 DSP board are:

- TMS320vc33 floating point DSP
- 13-ns instruction cycle time, 150M FLOPS, 75 MIPS
- Standard or enhanced printer port interface which connects to a host PC and allows the TMS320VC33 to communicate with PC programs.
- Standard 50-pin connector interface with external circuitry.
- DPRAM 7130 for parallel data transfer through standard 20-pin connector.
- Standard 6-pin interface for serial I/Os or can be configured as a general purpose I/O pins.
- Standard- 20pin output interface.
- Standard-14 pin emulator connector.
- Operated either by 9V DC power supply or 5V DC power supply.

- User selectable Microprocessor/Microcomputer mode.
- User selectable Clock multiplier and edge interrupt.
- Control logic generation using PALs
- Test Pads provided can be used for debugging or for future expansion.
- Operate on a single power supply

The key features of the GIDAC block are:

- Single ended analog input signal
- Low Pass filter with cut off- 40Hz
- 16-bit, Parallel ADC using ADS8364 from TI
- 250 kHz, Simultaneous sampling
- Total time of ADC - 1µsec (Conversion + Acquisition)
- GPS data acquisition using FPGA chip
- 5V tolerable Spartan-II xc2s200
- Operate on a single power supply
- Standard 50-pin connector interface with external circuitry.

# Chapter 7

## Results and conclusions

In this chapter the results obtained from different blocks of GPS-INS integrated system are discussed. Block layouts of GIDAC and NPC blocks is given. The developmental results obtained is explained in separate sections.

### 7.1 Signal conditioning of signals

The signals received from IMU sensors are passed through Butterworth filter, level-shifted and scaled to fall within the range of 0-5V. The effect of filtering on the sensor signals bandwidth was investigated. Single supply operated, two-stage op-amp to achieve filtering, scaling with level-shifting is implemented after its feasibility. The simulated output of signal conditioning circuit is shown in Fig. 7.1.

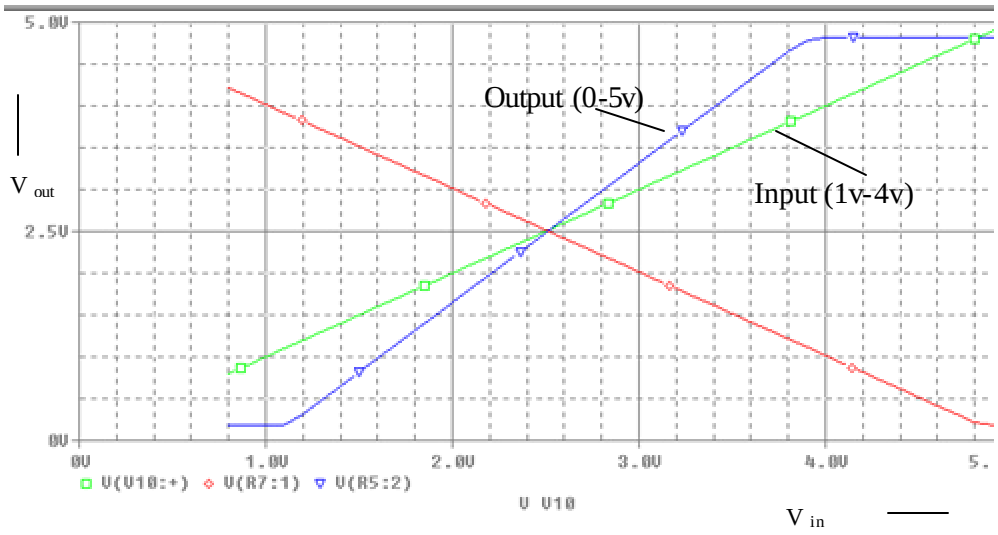



**Fig. 7.1** Output of gyro (one axis) signal conditioning circuit

### 7.2 ADC

Use of multiplexed ADC in data acquisition of IMU signals reported in earlier methods, was investigated for not considering the signal information at the same instant of time. For this purpose, simultaneous sampling of signals has been implemented. During every power-on of the system, ADC is initialized to reset state. And the chip is configured to acquire and convert in CYCLE mode, in which all the six channels (*A0, A1, B0, B1, C0* and *C1*) are read continuously. As seen from Fig.7.2,

the signals are sampled and read continuously in CYCLE mode. When sampled at 250kHz, the acquisition time and conversion time is 0.8µsec and 3.2µsec respectively.

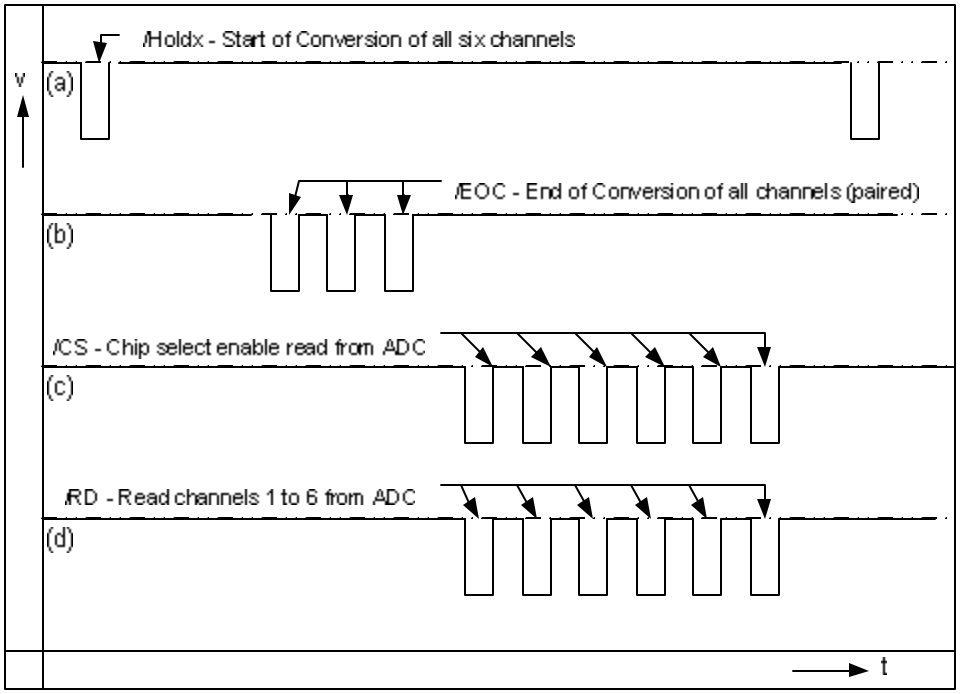



**Fig. 7.2** Output of simultaneous sampling ADC stage

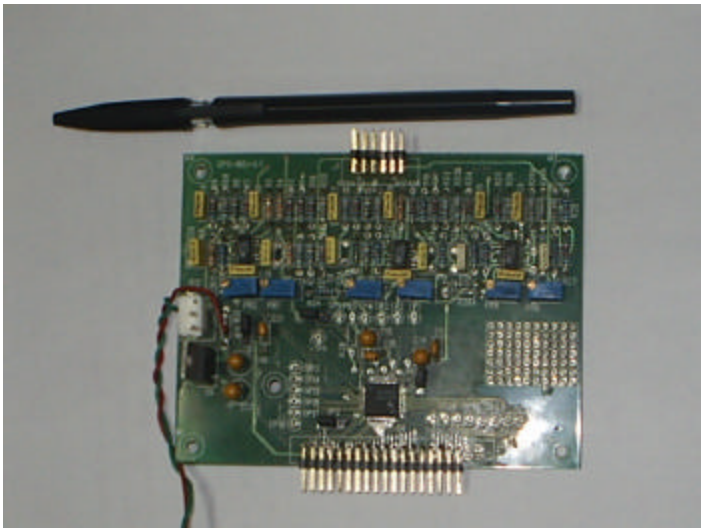


**Fig. 7.3** Board layout of GIDAC block

## 7.3 GPS

The regular approach for acquisition of GPS data was studied. First, the microcontroller receives the GPS data serially and after processing the received data, the latitude, longitude and altitude information is extracted and transmitted serially for necessary action. The GPS data obtained by using the microcontroller needs serial interface at both ends, i.e., from GPS receiver and to the main processor for necessary action. Thus, serial interface of GPS receiver output involves a tedious processing overhead on navigation processor, asynchronous communication is preferred in order to maintain serial data output GPS data acquisition. Hence, asynchronous interfacing techniques considering DPRAM were used for obtaining GPS data for further analysis. As the microcontroller and DPRAM have doubled the chip count to two, the same circuit can be effectively reduced by implementing it on FPGA, which is more reliable, compact and reprogrammable. It is observed from the synthesis report that the number of slices required for SIRF sentences is much less compared to the NMEA sentences. Also the SIRF sentence configured FPGA chip can be operated at a higher speed than that of NMEA sentence configured FPGA chip. The overall combinatorial delay in processing the signals from the time input to the FPGA chip and the output is almost same in both the cases. The comparison obtained from the synthesis report is reported in Table 7.1.

| Selected Device 2s200pq208-5 | For NMEA sentence | For SIRF sentence |
|---|---|---|
| **Device utilization summary** | | |
| Number of Slices | 279 out of 2352 11% | 194 out of 2352 8% |
| Number of Slice Flip Flops | 393 out of 4704 8% | 307 out of 4704 6% |
| Number of GCLKs | 1 out of 4 25% | 1 out of 4 25% |
| **Timing Summary** | | |
| Minimum period | 14.288ns (Maximum Frequency: 69.989MHz) | 13.440ns (Maximum Frequency: 74.405MHz) |
| Maximum combinational path delay | 23.999ns | 22.402 nsec |
| | | |

**Table 7.1** Comparison between NMEA and SIRF sentence

## 7.4 DSP Computations

System performance with INS data alone on Code Composer Studio simulator environment, was conducted for 50 sec (5001 data), the navigation results shows that the system output is same upto 4 decimal places when compared with the Matlab results. As solving differential equation algorithm adopted in Matlab is much accurate than the $4^{th}$ order Runge-Kutta adopted in our system flow, the output observed was considered almost correct. As far as the time constraint and the memory of the system is considered, the DSP TMS320VC33 chosen as navigation processor has a better edge over other processors used so far. Moreover, there is no requirement of any external interface of RAM as the internal SRAM of the processor is quite sufficient for computations. It was observed from the map report, that the total memory usage is 17K words. The RAM consumption generated from map report is shown in Table 7.2.

| Name | Origin | Length | Used | Words |
|---|---|---|---|---|
| ROM | 00000000 | 000001000 | 00000120 | 4K |
| RAM2 | 00800000 | 000004000 | 00002F81 | 16K |
| RAM3 | 00804000 | 000004000 | 00000e18 | 16K |
| MMRS | 00808000 | 000000100 | 00000000 | |
| RAM0 | 00809800 | 000000400 | 00000400 | 1K |
| RAM1 | 00809c00 | 000000400 | 00000000 | |
| | | Total words | 16.58 K | |

**Table 7.2** Map Report generated using CCS

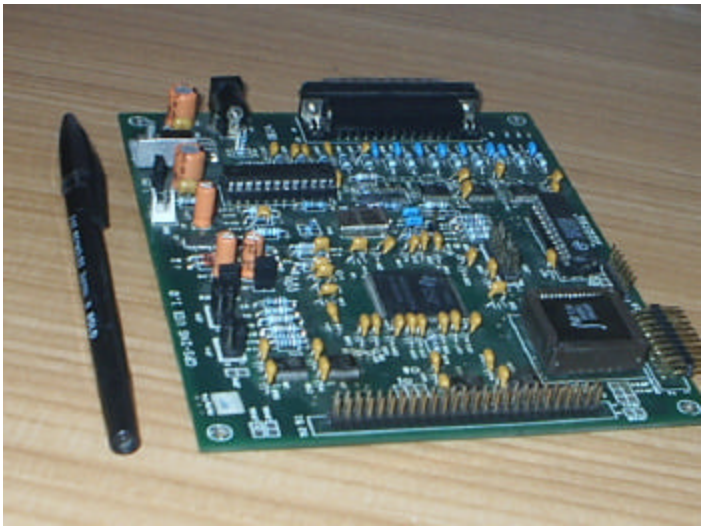


**Fig. 7.4** Block layout of DSP based NPC block

The execution time for processing INS and Kalman filter algorithms was calculated from the number of instruction cycles generated during simulation on CCS. The INS program was run for 50s or 5001 steps (with a time step of 10 milliseconds) and the total number of instruction cycles found to be 224744224, and each INS computation comes to 44940 steps. As, the DSP on NPC card is operated at 75Mhz, which corresponds to 26nsec per instruction cycle, the execution time is considered based on this. On an average each INS computation will take 1.17ms to execute on DSP. The maximum time and the minimum INS computation time will lead to 1.34ms and 1.16msec respectively to execute on DSP. Similarly, the time for one Kalman filter computation will take 6.34msec with 2443314 instruction cycles to execute on DSP.

| Computation During | No of cycles | Execution time on DSP operated with 75MHz | Execution time on DSP operated with 150MHz |
|---|---|---|---|
| INS | 44940 | 1.17msec | 0.58msec |
| Kalman Filter | 243314 | 6.33msec | 3.17msec |
| Simultaneous Sampling ADC | | 10.4 µsec | 10.4 µsec |
| DPRAM access | - | 1 µsec | 1 µsec |

**Table 7.3** Computation time of INS and Kalman Filter algorithm

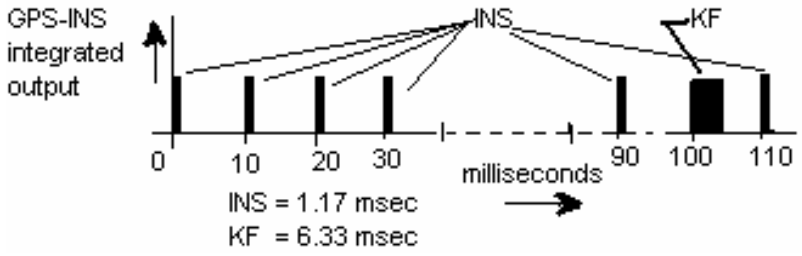



**Fig. 7.5** INS and KF computations

The developed GPS-INS integrated system weighs about 183gms and consumes 1.7 watts and 62 mW when the system is in active mode and idle mode respectively. The overall size of the prototype system is 15`` x 12.5``. The results obtained from the GPS-INS integrated system operating INS alone, shows that the output values are almost equal to 4 decimal places to that of Matlab results. Further, the system with both INS and GPS inputs are implemented using 9 state Kalman filter. The output waveforms obtained from system is comparatively same as that of Matlab results. The output waveforms obtained from the GPS-INS integrated system is shown in Fig. 7.6 to Fig. 7.8.

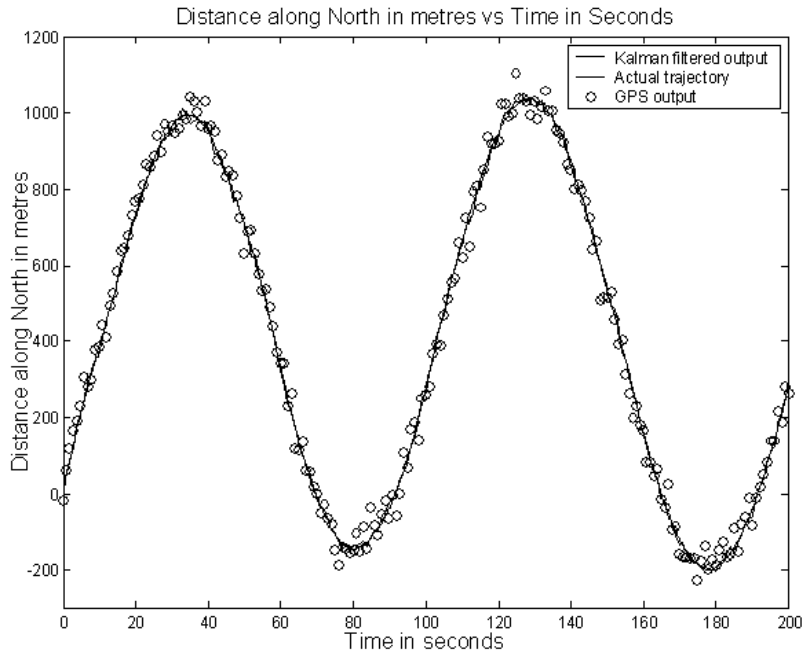



**Fig. 7.6** Distance along North in meters

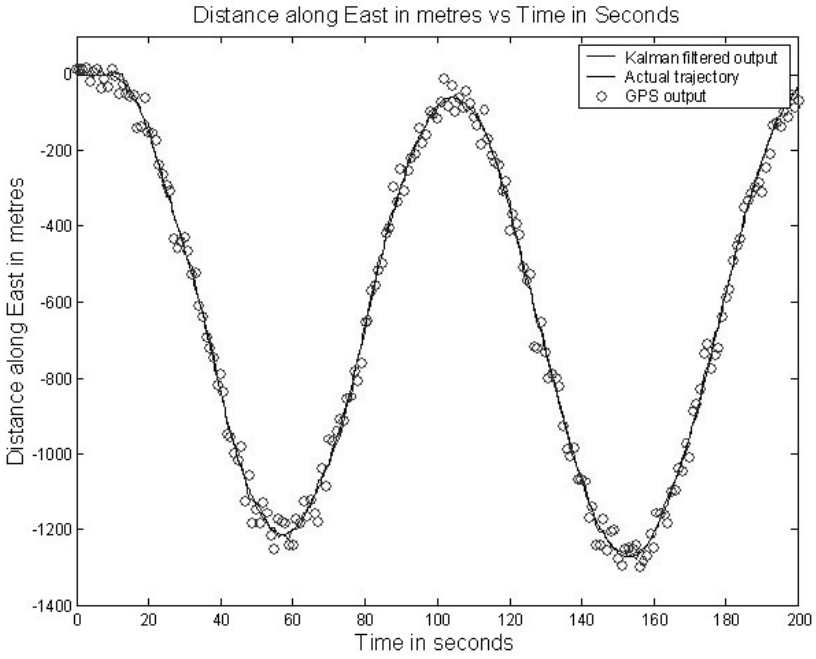



**Fig. 7.7** Distance along East in meters

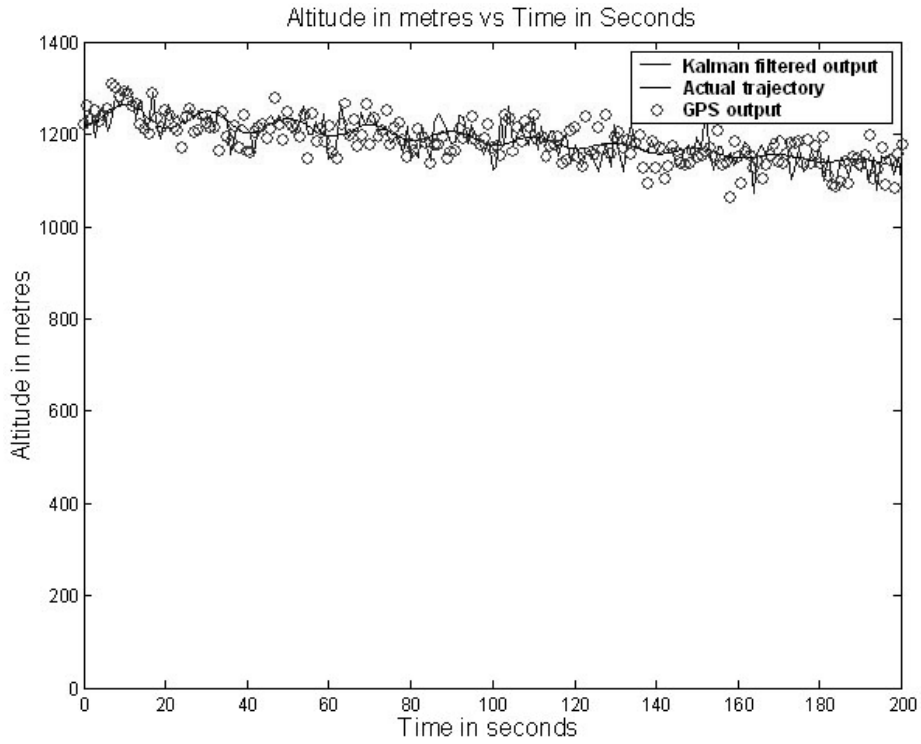



**Fig. 7.8** Altitude in meters

To summarize, the following are obtained from the system:

- With two stage op-amp, operating on a single-supply, filtering, offset and linearising was achieved.
- From the output waveform of the ADS8364, it is seen that all the six channels are sampled at same instant of time and read continuously for further processing.
- GPS data is acquired using Spartan-II FPGA chip, and the synthesize report shows that Proprietary sentence data consumes less memory and can be operated at higher speed compared to NMEA sentences.
- Use of DSP TMS320vc33 has an inbuilt 34K words SRAM, hence no external RAM interface is required.
- The results show that the INS and Kalman filtering computation processing speed becomes faster with the increase in speed of the processor.
- GPS-INS integrated system output is almost same as that of the Matlab results.
- Use of Single supply for the entire system, consumes around 1.7W during active mode and 60mW in active mode,
- The system weighs around 183gms.

**Suggestions for future work**

- The effect of interfacing IMU sensors data using data acquisition board through PC with the proto-type GPS-INS integrated system need to be analysed.
- Performance evaluation of the proto-type GPS-INS integrated system with actual sensor inputs conducting field trials need to be carried out.
- Presently, the FPGA chip is programmed to receive data from GPS receiver. By default almost all GPS receiver outputs NMEA sentences with certain baud rate. For the first time, on every power-on of the GPS receiver, the receiver has to be intialised/ programmed, to receive either NMEA or proprietary sentences with suitable baud rate. This part of the VHDL code can be added to the existing VHDL code and the system can be made more compact.
- Higher state Kalman filter algorithm can be studied and implemented on this system for better accurate position information.

# Appendix A

## Schematics of GIDAC board

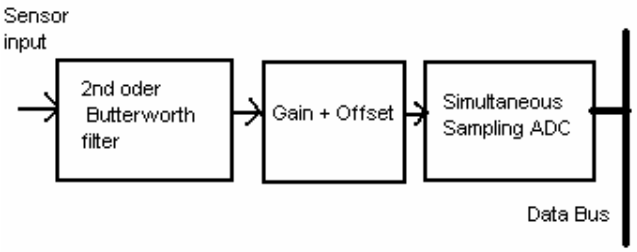



**Fig. A.1** Block Diagram of GIDAC Block (INS)

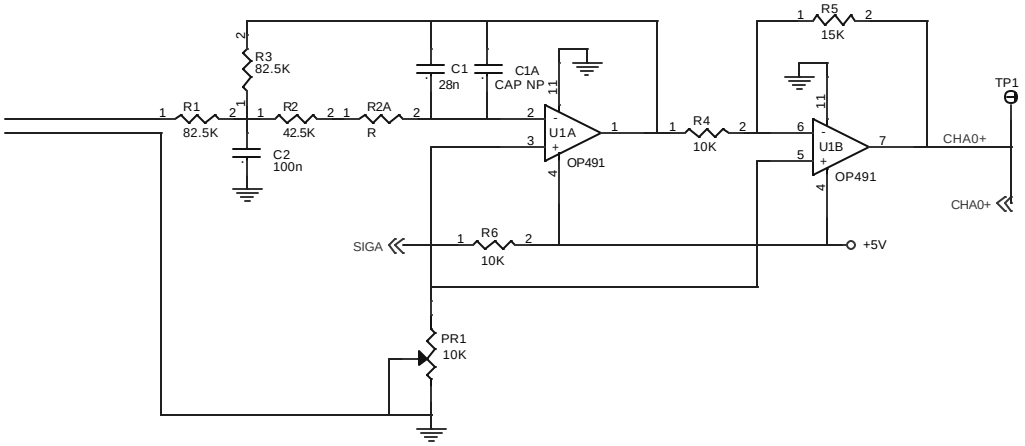



**Fig. A.2** Signal conditioning for accelerometer input [one axis] (GIDAC block)

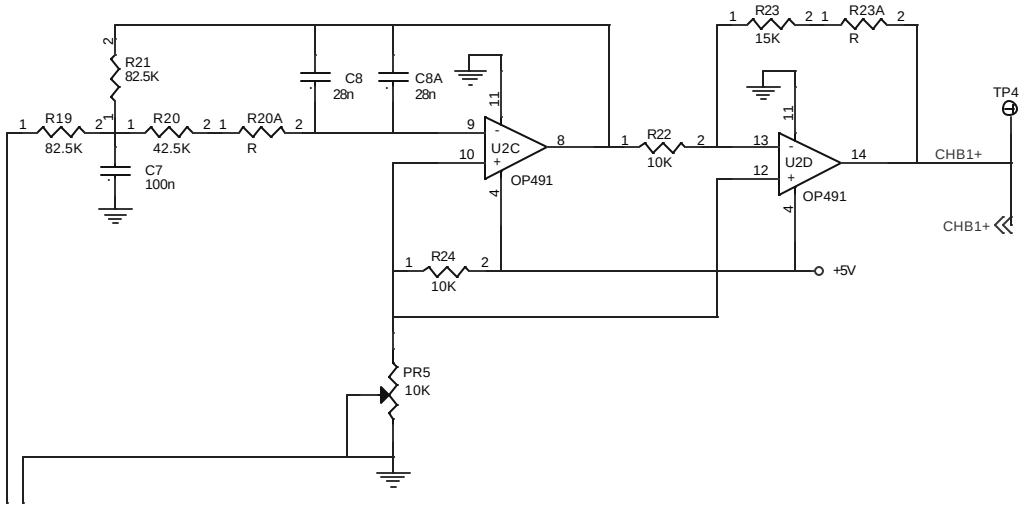



**Fig. A.3** Signal conditioning for gyroscopes input [one axis] (GIDAC block)

**Fig. A.4** INS data acquisition using Simultaneous ADC (GIDAC block)

# Appendix B

## GPS Receiver module implementation on FPGA- Program flow

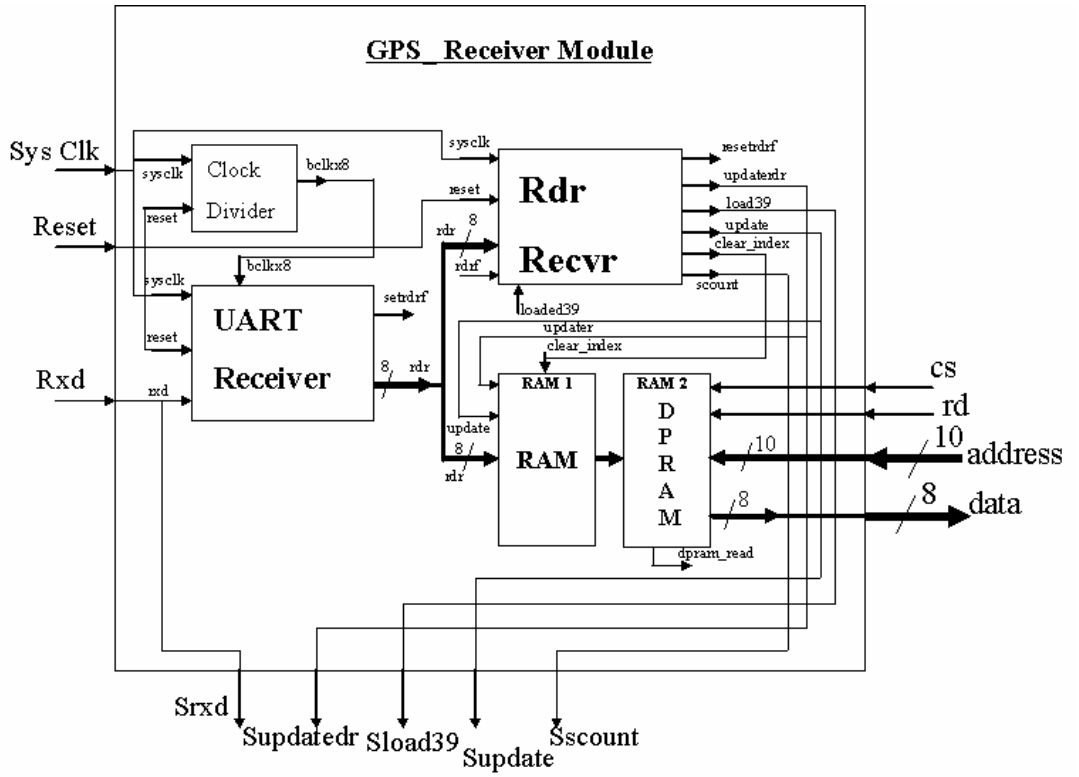



**Fig. B.1** GPS block implementation on FPGA chip

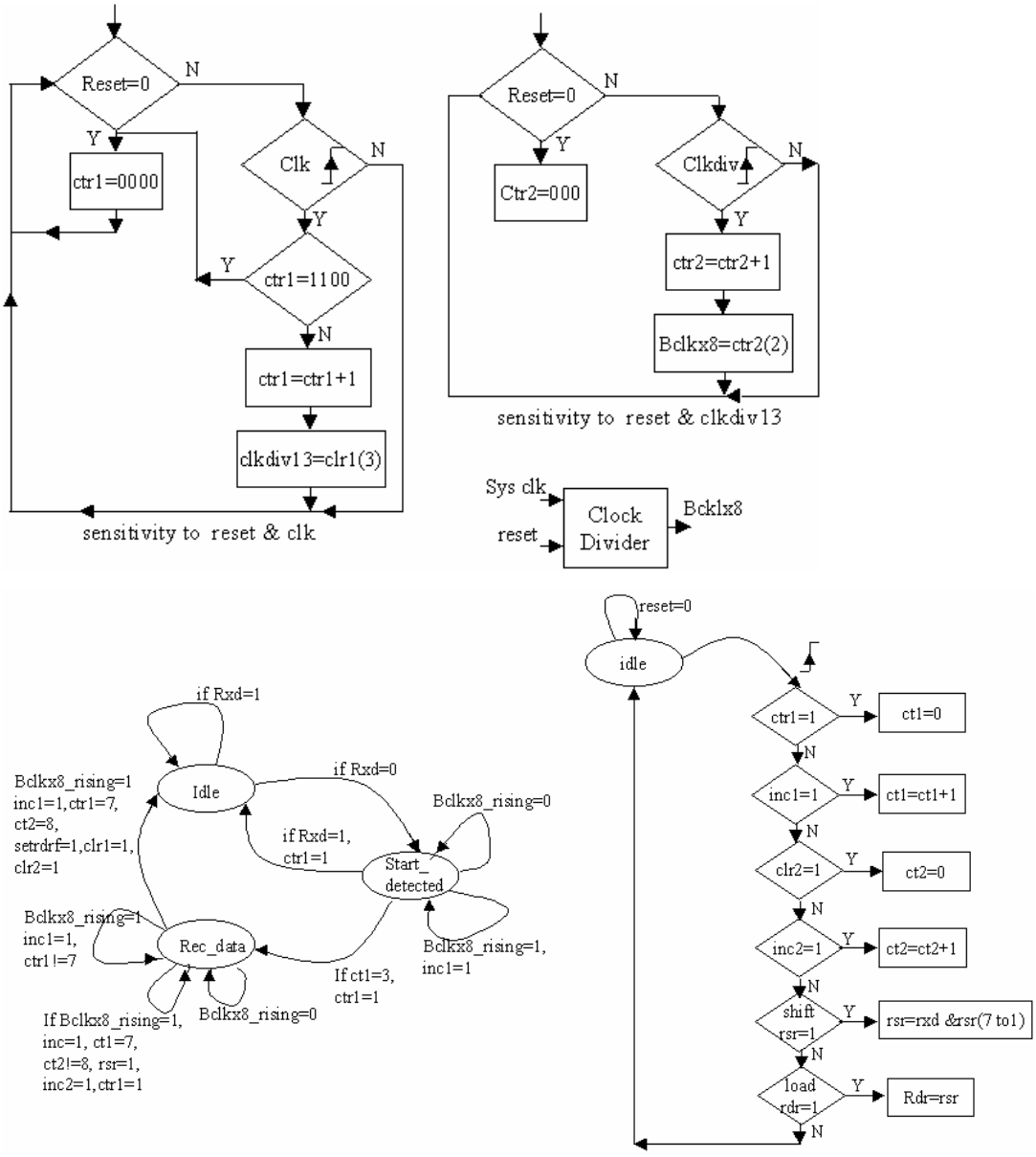



**Fig. B.2** Program flow of baud rate generator block

(GPS module)

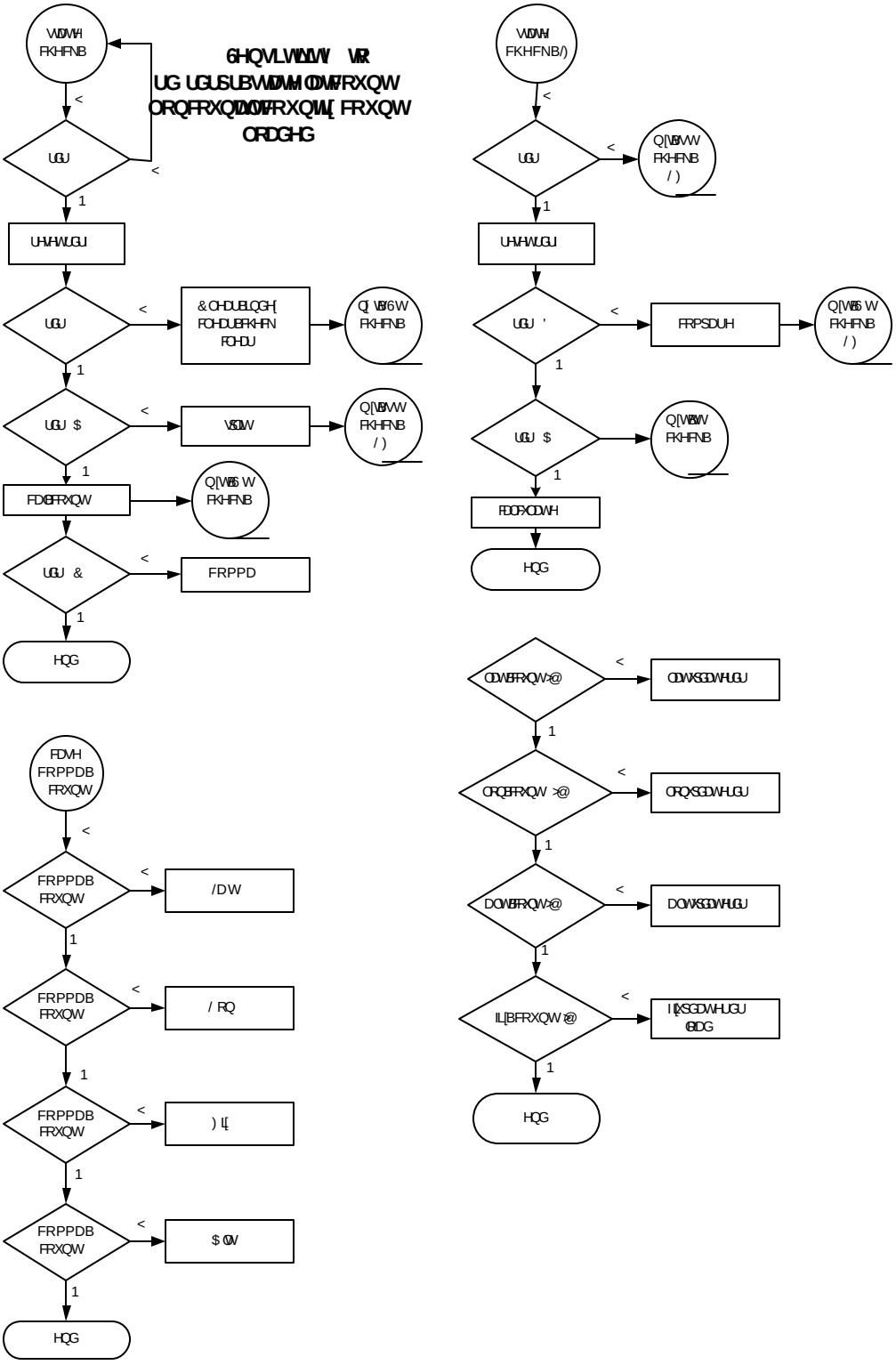
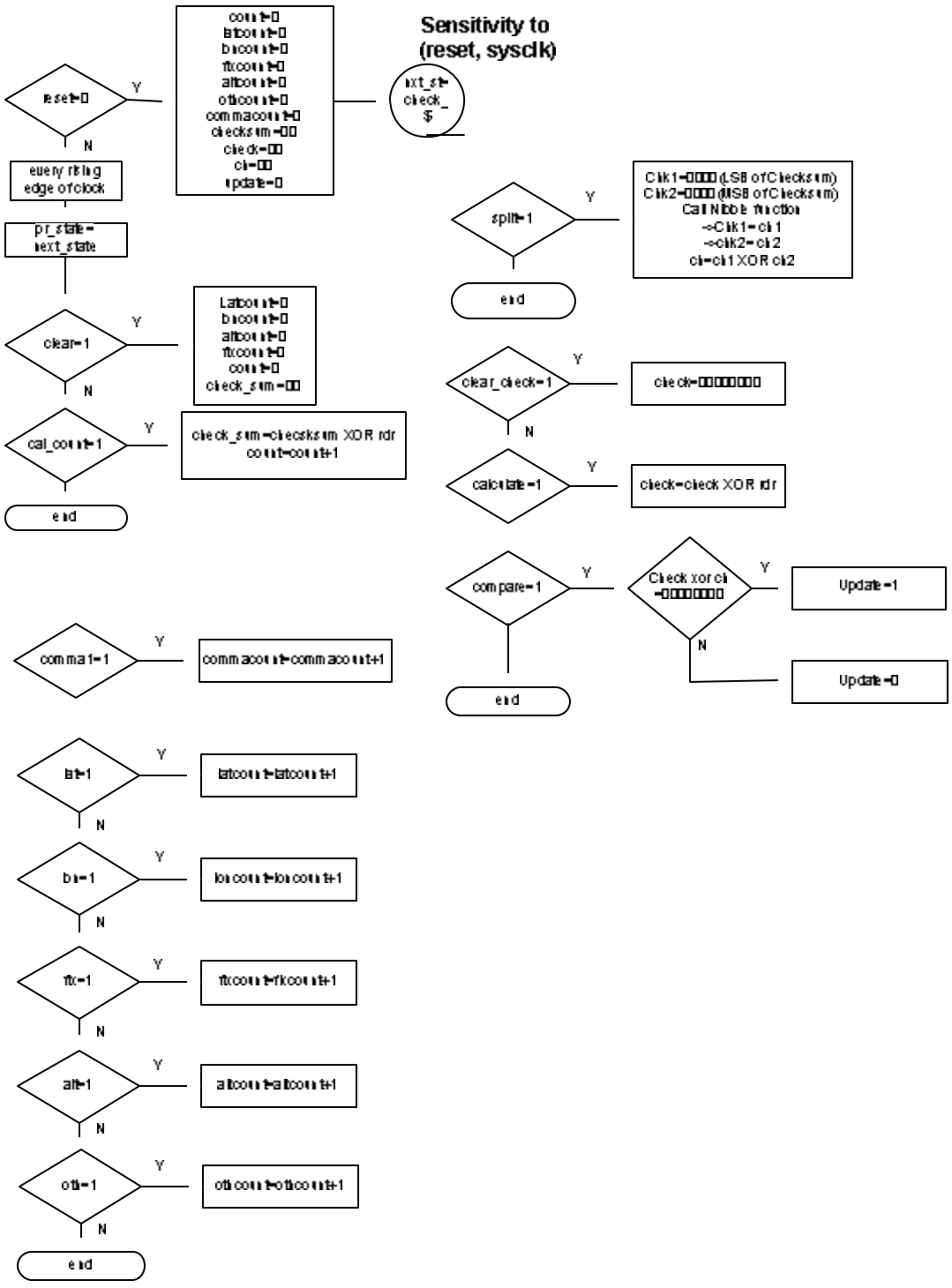


**Fig. B.3** Program Flow of UART Receiver Block (GPS Module )


Sensitivity to
(reset, sysclk)
reset='0'
Y
N
every rising
edge of clock
pr_state <=
next_state
clear='1'
cal_count='1'
end
comma='1'
bit='1'
bir='1'
ftr='1'
alt='1'
oth='1'
split='1'
clear_check='1'
calculate='1'
compare='1'
Update='1'
Update='0'

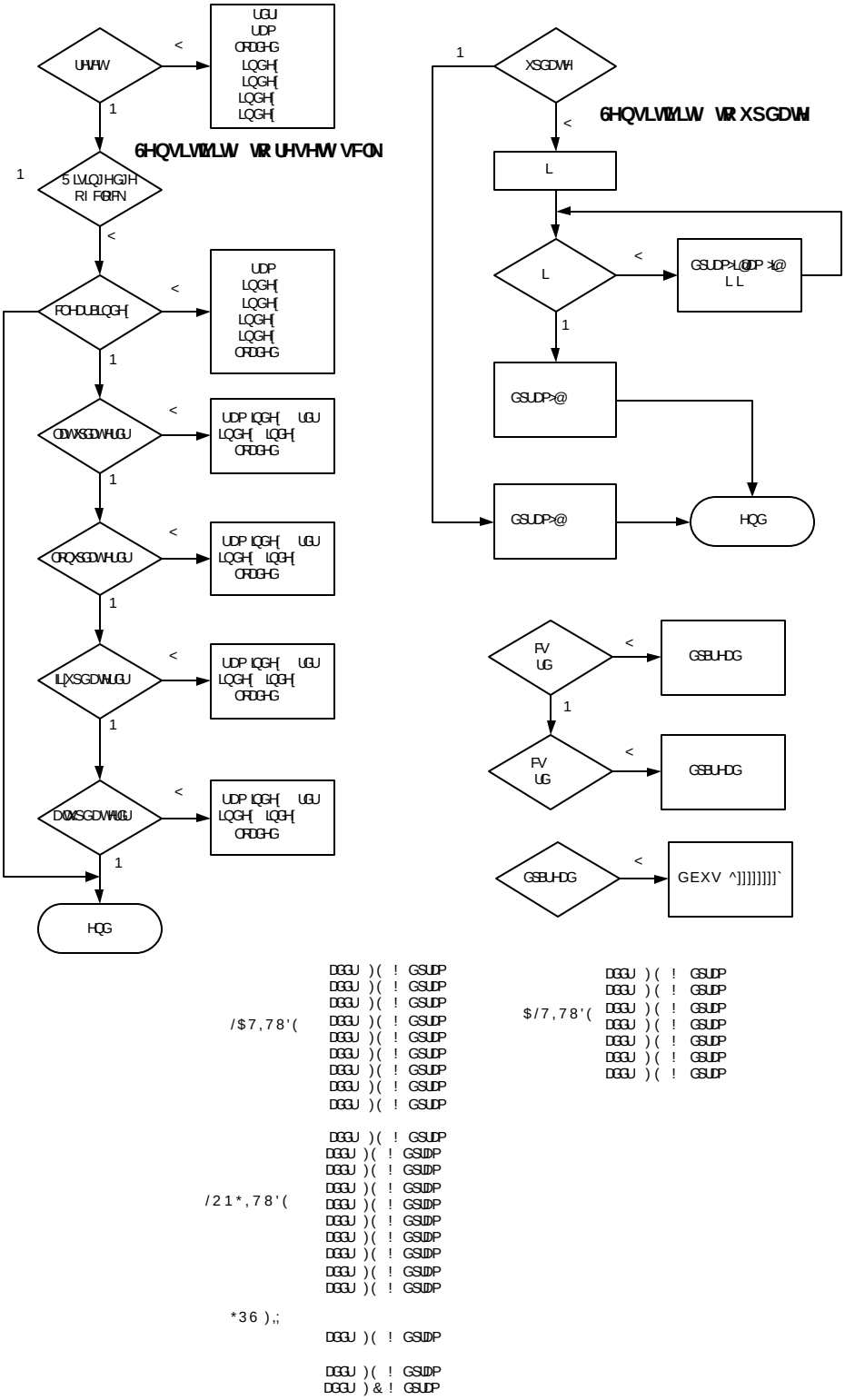


**Fig. B.5** Program flow of GPS Top Block (GPS Module)

# Appendix C

## Schematics of Navigation Processor Card

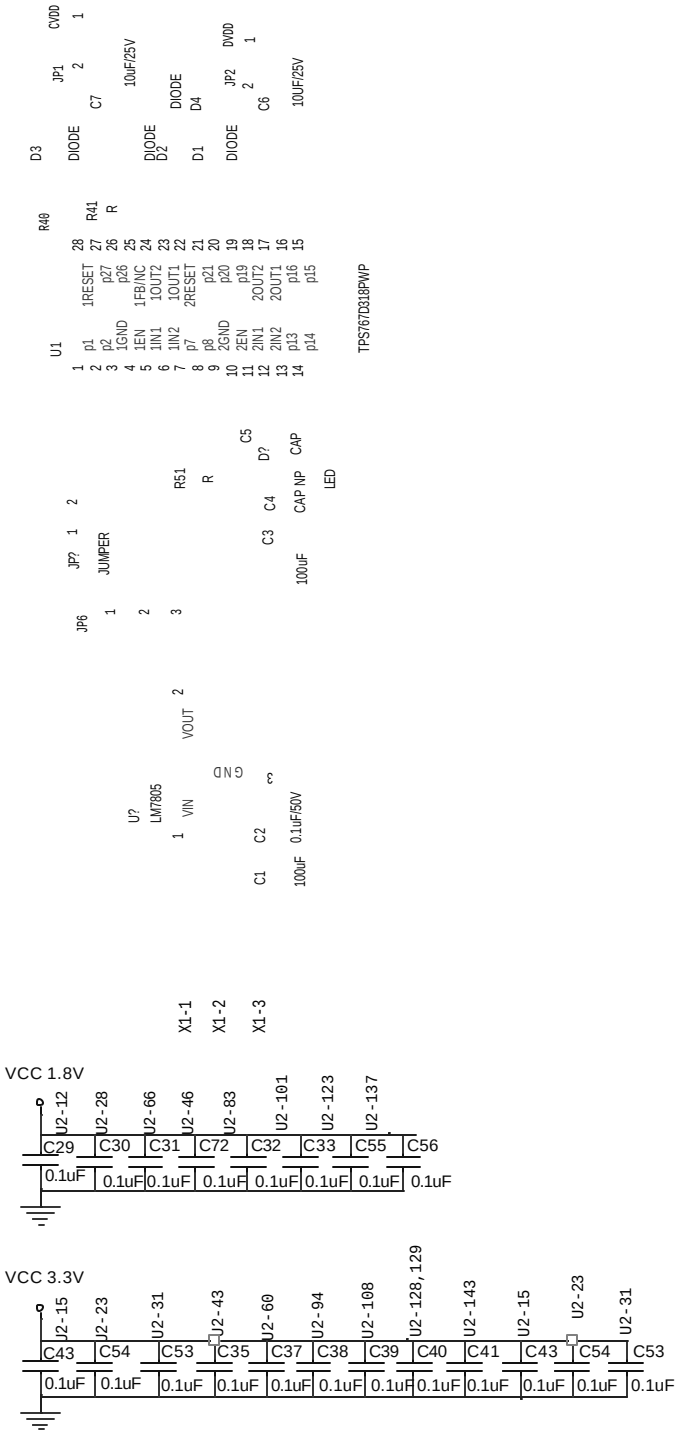



**Fig. C.1** Power Supply schematic (NPC block)

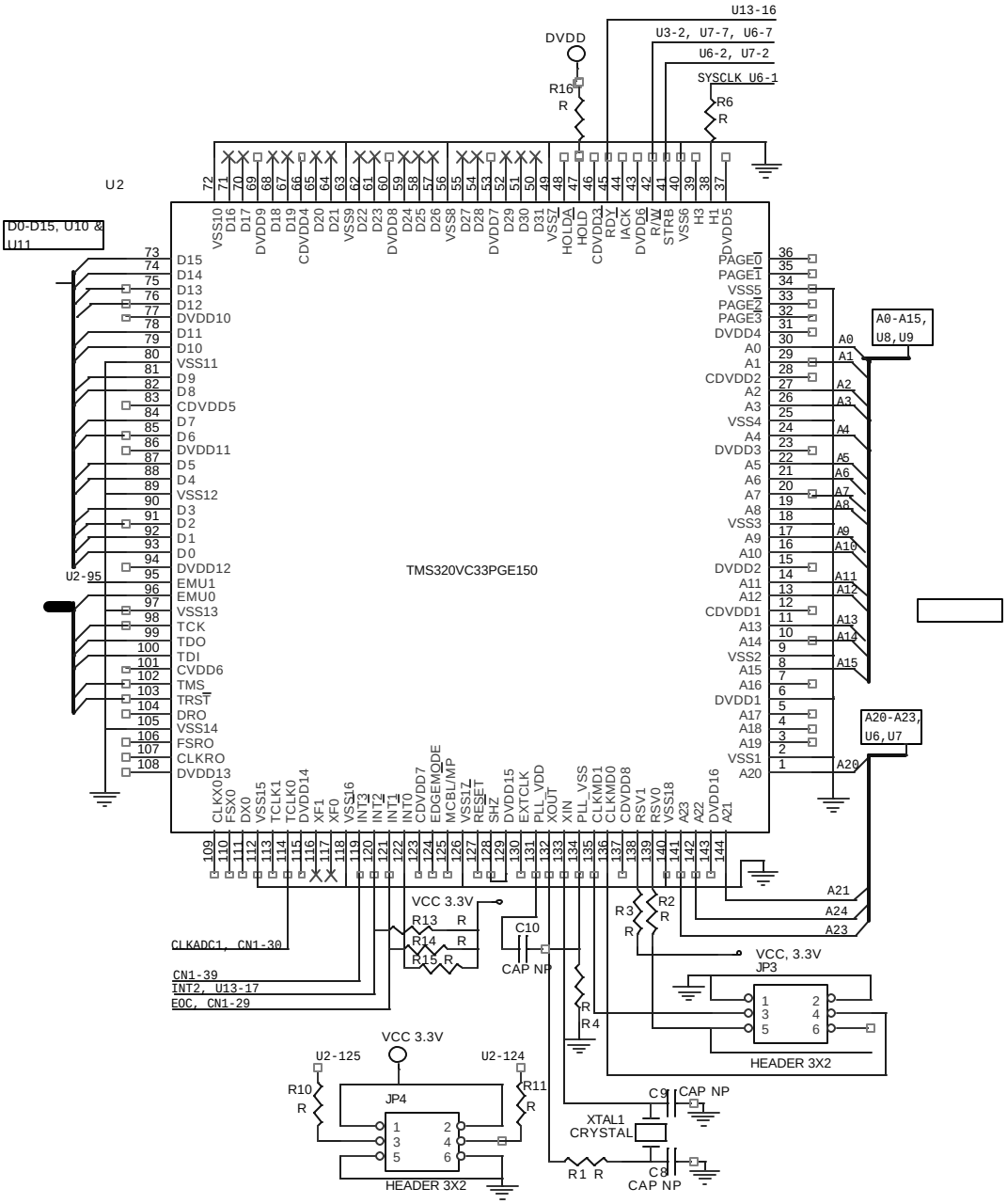



**Fig. C.2** TMS320VC33 based NPC block

**Fig. C.3** Control signal generation using PAL22v10 circuit and parallel port (NPC block)

**Fig. C.4** Address and Data bus Transreceivers/shifters(NPC Block)

# Appendix D

## PALASM design description

```
;---------------------------------- Declaration Segment ------------
TITLE    PC Communication WITH DSP
PATTERN
REVISION GPS_INS V 1.0
AUTHOR   S.BHAKTAVATSALA, M Tech(ES), C/o Prof VIVEK AGARWAL
COMPANY  APEL LAB, IIT BOMBAY, POWAI, MUMBAI-400076
DATE     03/02/04
CHIP   U6_V1  PAL22V10
;---------------------------------- PIN Declarations ---------------
PIN  1          SYSCLK                  COMBINATORIAL ; INPUT
PIN  2          /STRB                   COMBINATORIAL ; INPUT
PIN  3          A23                     COMBINATORIAL ; INPUT
PIN  4          A22                     COMBINATORIAL ; INPUT
PIN  5          A21                     COMBINATORIAL ; INPUT
PIN  6          A20                     COMBINATORIAL ; INPUT
PIN  8          T1                      COMBINATORIAL ; INPUT
PIN  10         R_W                     COMBINATORIAL ; INPUT
PIN  12         GND                                   ; INPUT
PIN  13         HPIS                    COMBINATORIAL ; INPUT
PIN  15         OE                      COMBINATORIAL ; OUTPUT
PIN  17         INT2                    COMBINATORIAL ; IO
PIN  18         RDY                     COMBINATORIAL ; OUTPUT
PIN  19         SRAM                    COMBINATORIAL ; OUTPUT
PIN  20         USERX                   REGISTERED    ; OUTPUT
PIN  22         Q1                      REGISTERED    ; OUTPUT
PIN  23         Q0                      REGISTERED    ; IO
PIN  24         VCC                                   ; INPUT
;---------------------------------- Boolean Equation Segment ------
EQUATIONS

INT2 = HPIS
OE = /(A23 * A22 * A21 * /STRB)
Q0.CLKF = SYSCLK
Q1.CLKF = SYSCLK
Q0 := INT2
Q1 := Q0
RDY =/(Q0 * /Q1) * (A23 * A22*A21*A20*/STRB)
SRAM=/(A23 * A22 * /A21 * /A20 * /STRB)
USERX:=T1

;---------------------------------- Simulation Segment ------------
SIMULATION
;------------------------------------------------------------------
```

This program was developed on PALASM software.

# References


1. M Faulkner, S J Cooper and P A Jeary, "Integrated MEMS/GPS navigation systems," Position Location and Navigation Symposium, *IEEE* , April 2002, pp. 306-313.

2. Sung Wook Moon, Dong-Hwan Hwang, Tae Kyung Sung and Sang Jeong Lee, "Design and Implementation of an Efficient Loosely-Coupled GPS/INS Integration Scheme", Chungnam National University, KOREA.

3. Jie Shang, Gang Mao and QiTai Gu, "Design and implementation of MIMU/GPS integrated navigation systems," Position Location and Navigation Symposium, *IEEE*, April 2002, pp. 99 - 105.

4. F X Cao, D K Yang, A G Xu, J Ma, W D Xiao, C L Law, K V Ling and H C Chua, "Low cost SINS/GPS integration for land vehicle navigation," Intelligent Transportation Systems, *IEEE*, Sep 2002, pp. 910 – 913.

5. Liu Lichun Tian Zengshan Huang Shun-ji, "An algorithm for integrating GPS/INS attitude determination system," Radar, CIE International Conference on, Proceedings, *IEEE*, 2001, pp. 167-170.

6. Dong-Hwan Hwang, Sung Wook Moon, Tae Kyung Sung and Sang Jeong Lee, "Design and Implementation of an Efficient Tightly-Coupled GPS/INS Integration Scheme", Chungnam National University, KOREA.

7. D. T. Knight, "Rapid Development of Tightly-coupled GPS/INS Systems," *Proceedings of ION International Technical Meeting*, Nashville, Tennessee, 1999.

8. Dr. Michael K. Martin and Bruce C. Detterich, "C-MIGITS™ II Design and Performance", *Proceedings of: The Satellite Division of The Institute of Navigation 10th International Technical Meeting ION GPS-97 Kansas City*, Missouri Sep 16 - 19, 1997.

9. Wang Jin-ling, Lee H K, Rizos C "GPS/INS Integration: A Performance Sensitivity Analysis" Sydney, Australia.

10. O.S. Salychev, V.V. Voronov M.E. Cannon, R. Nayak, G. Lachapelle, "LOW COST INS/GPS INTEGRATION: CONCEPTS AND TESTING", *Canada.*

11. M.G. Petovello, M.E. Cannon and G. Lachapelle, "Development and Testing of a Real-Time GPS/INS Reference System for Autonomous Automobile Navigation"

12. Chang-sun Yoo, Iee-ki Ahn, " Low cost GPS/INS sensor fusion system for UAV integration, Korea Aerospace Research Institute, Daejon, Korea.

13. Mohinder Grewal, Lawrence R Well and Angus P.Andrews "GPS positioning systems, Inertial navigation, and integration", Willey-Inter science, Edition 2001.

14. "ADXL202AE" website of Analog Devices, http://www.analog.com/ UploadedFiles/Data_Sheets/70885338ADXL202_10_b.pdf accessed in 2004.

15. "ADXRS300" website of Analog Devices, http://www.analog.com/ UploadedFiles/Data_Sheets/732884779ADXRS300_b.pdf accessed in 2004.

16. Sergio Franco, " Design with operational amplifiers and analog integrated circuits", Tata McGraw-Hill edition 2002.

17. "OP491" website of Analog Devices, http://www.analog.com/ UploadedFiles/Data_Sheets/813995969OP191_291_491_c.pdf accessed in 2004.

18. ADS8364, 6 Channel, Simultaneous Sampling Parallel ADC, data sheet, Texas Instruments, SBAS219.

19. GPS-Receiver JP3 users manual, Falcom

20. Bhaskar. J, *VHDL Primer*, 3e, Prentice Hall, 1999.

21. TMS320C3X User's Guide, Texas Instruments SPRU031e.

22. TMS320VC33 User's Guide, Texas Instruments SPRS087E.

23. "TPS767D318PWP" website of Texas Instruments, http://focus.ti.com/ lit/ds/symlink/tps767d318.pdf accessed in 2004.

24. "IDT7130" website of Integrated devices technology" http://www1.idt.com/ pcms/products.taf?catID=&genID=7130&sort=IO Level& sDir accessed in 2004.

25. Code Composer Studio User's Guide, Texas Instruments SPRU296a.

26. TMS320 floating point DSP Assembly Language Tools , Texas Instruments SPRU035B

27. Kun-Shan Lin, Editor, *Digital Signal Processing Applications with TMS320 Family Volume 1*, Prentice-Hall and Texas Instruments, Digital Signal Processing Series, 1987

28. Panos Papamichallis, Editor, *Digital Signal Processing Applications with TMS320 Family Volume 3,* Prentice-Hall and Texas Instruments, Digital Signal Processing Series, 1990

29. Vikas Kumar Naresh, "Integration of Inertial Navigation System and Global Positioning System using Kalman Filtering", M Tech Report, 2004