Fast Code-Phase Alignment of GPS Signals Using Virtex-4 FPGAs

Usama Malik, Oliver Diessel
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
umalik@cse.unsw.edu.au, odiessel@cse.unsw.edu.au

Andrew G Dempster
School of Surveying and Spatial Information Systems
University of New South Wales
Sydney, Australia
a.dempster@unsw.edu.au

***Abstract* — This paper describes a Virtex-4 based system for aligning the code-phase of a received GPS signal. The core operation involves a multiplication of the received signal with a local replica of the code followed by integration of all possible alignments within the period of one code epoch. A speedup proportional to the code length is thus achieved. We outline the proposed system, which stores the code in on-chip memory blocks and uses both dedicated DSP hardware and user logic to perform MAC operations in parallel. We study area, time and energy usage as the ratio of user logic to custom blocks is varied and identify a design point and corresponding device size for which energy usage is minimized.**

I. INTRODUCTION

The basis for the new GPS receiver design is the *Namuru* open GNSS receiver developed at the School of Surveying and Spatial Information Systems, UNSW [2]. The FPGA-based receiver can either be incorporated into a development kit or can be made available as an integrated IP block. A fully functional L1 GPS receiver is currently implemented on an Altera device and a new version aims to incorporate the Galileo L1 and GPS L2C signals. This paper presents an FPGA design for fast acquisition of the GPS L1 signal that can later be modified to operate on the new signals. "L1" denotes the carrier frequency 1575.42 MHz; the L2 carrier is at 1227.6 MHz.

Acquiring a GPS signal involves a two dimensional search in the frequency and code-phase space. The *Namuru* implementation currently uses the conventional approach of sequentially searching the frequency/phase space for code alignment. A copy of the code is generated at a given frequency and multiplied by the incoming signal. The product is integrated for a suitable period. The code is then shifted by one bit position and the procedure repeated to test alignment at the new phase. A range of frequencies needs to be examined to account for offsets due to Doppler shift and drift in the local oscillator. Signal acquisition can therefore take many seconds. Due to their greater code length, acquiring the L2C and Galileo signals will take longer [1].

This paper exploits parallelism to reduce the L1 acquisition time. The use of parallel techniques for this purpose is well known. Software receivers often use an approach that requires both forward and inverse FFTs [7]. This method can be computationally expensive, particularly for longer codes. In digital hardware, parallel signal "search engines" have been used by several manufacturers (e.g. [6]). Unfortunately, very little detail is published on the architecture of search engines. The language used to describe them suggests they consist of a large number of parallel correlators (2,046 per satellite in the case of [6]). Such an approach uses a significant amount of circuitry compared to the simple sequential approach, and requires a correspondingly large amount of power to complete the search, although for a shorter time. In this paper we outline a time-efficient design suited to the current Virtex-4 and Virtex-5 FPGA device families from Xilinx (there is a plan to port the receiver from Altera to Xilinx in order to exploit the greater reconfigurability). The design is parameterized on the packing density of internal memory blocks so as to provide flexibility in the area and power required. Our best design uses 2% of the logic and 75% of the block memory resources of the (smallest) XCV4VFX12 device operating at less than 0.5 W for 2 ms to acquire any L1 GPS signal.

The paper is organized as follows. Section 2 presents background to GPS signals and the problem of aligning code phase. Section 3 presents the target FPGA architecture. Section 4 describes the new hardware search engine and evaluates its performance. Section 5 concludes the paper with references to future work.

II. BACKGROUND TO GPS SIGNALS

GPS employs direct-sequence, spread spectrum communications. Signals are coded using *pseudorandom noise* (PRN) codes and modulated onto a carrier. PRN codes are long sequences of pseudorandom bits such that the dot product of any two code vectors ideally results in zero (in practice this does not occur because codes are not perfectly orthogonal). A PRN encoding of a binary bit-stream replaces each 1 in the input data by one or more epochs of the entire PRN code (e.g. 20 epochs for the GPS L1 signal) and each 0 by the inverse of that code. This process is also

called *spreading*. The resulting spread signal is then modulated (e.g. the L1 signal uses BPSK). The receiver demodulates and de-spreads the signal by performing the reverse of this process. However, the receiver must determine the start of each bit in the message by correctly aligning its copy of the code with samples of the sent signal.

A spread spectrum signal is more immune to fading, cross-correlation and interference, and the PRN code length is critical in defining the signal behavior. In general, longer codes provide more immunity to these problems. The downsides are the effort needed to acquire the signal and added complexity of the receiver. The PRN codes for the L1 signal are 1,203 *chips* long where a *chip* refers to a single bit in the spreading code. The term *chip* is used instead of the term *bit* because the individual bits in the code bit-stream carry no data. Each satellite is represented by a separate PRN code. The new L2C and Galileo signals have much longer code lengths. Table 1 lists the important parameters of various signals.

This paper addresses the problem of determining the correct code-phase of the received signal. In other words, the GPS receiver must determine when a PRN code begins in the input bit-stream before it can be de-spread. The accuracy with which this process can be performed plays a major role in the accuracy with which the distance to the satellite, and hence position, can be calculated. The correlation of a received signal, $r(t)$, with a locally generated signal, $s(t)$, can be described using the following equation:

$$R(\tau) = \int_0^T r(t)s(t+\tau)dt \quad \text{.....Eq. 1}$$

The received signal is observed for time T and multiplied by a local replica of the code which is phase shifted by τ seconds. The result is summed and compared to a threshold. Correct phase is said to be detected if the threshold is exceeded. Otherwise, τ is changed and the process repeats. In Eq. 1, T is referred to as the integration period. Longer integration periods are desirable because noise is further averaged. For L1, an integration period of 1 msec gives a 93% probability of detection which is acceptable. Various parameters of GPS signals are shown in Table 1.

Parameter	L1	L2c		L5	
	C/A	CM	CL	I5	IQ
Chipping Rate (chips/sec)	1.023M	0.5115M	0.5115M	10.23M	10.23M
Code length(chips)	1,023	1,023	767,250	10,230	10,230
Code epoch (ms)	1	20	1.5	1	1

Table 1: Important parameters of GPS signals.

III. VIRTEX-4 ARCHITECTURE

The target FPGA in this work is the Virtex-4 Field-Programmable Gate Array (FPGA) from Xilinx Inc [3]. This device was chosen because a goal of our project is to examine runtime reconfigurable GPS receivers and the Vertex-4 family offers the best options. A Virtex-4 device is organized into a two-dimensional array of *configurable logic blocks* (CLB). Each CLB consists of four *slices* where each slice contains two 4-input Look-Up-Tables (LUTs) and a pair of single-bit registers. Special carry chains are provided to expand the size of arithmetic circuits. Each CLB is connected to a *switch matrix* that can be configured to form connections between the CLB and a hierarchical interconnection network. Direct connections to neighboring CLBs are also possible.

A Virtex-4 device also contains other embedded resources such as blocks of RAM (BRAMs) and DSP slices. Each BRAM is 18Kb in size and can be configured to aspect ratios ranging from 16K×1-bit to 512×32-bit. A Virtex-4 BRAM is a dual-port structure and can be clocked at up to 500MHz. The number of BRAMs differs depending upon the device family (LX, FX or SX) and device size. BRAMs can also be combined to form larger memories. A Virtex-4 *XtremeDSP* slice contains an 18×18 multiplier followed by a multiplexer and an adder/subtracter that can take three 48-bit inputs. The slice operates as an optional three stage pipeline with a feedback loop. This feature can be used to construct high throughput MAC units. Each slice is located close to a BRAM and can form direct connections with it using a special interconnect. A DSP slice can sequentially generate addresses for the BRAM thus allowing the designer to construct FIFOs and cyclic buffers without using any extra logic resources. Several DSP slices can be cascaded to form larger circuits. Support for propagating carry bits and accumulating partial products is hardwired.

IV. HARDWARE MAPPING

In a typical GPS receiver, the RF unit downconverts the satellite signal to an intermediate frequency (IF) and digitizes it. As an example, Zarlink's GP2015 has an IF of 1.428MHz and outputs 2-bit sign/ magnitude samples at 5.714MHz. These samples are considered 3-bit signed integers here because they take the levels (±1, ±3). For an integration period of 1 ms, a vector of 5,714 integers must be multiplied by a vector of 5,714 PRN samples. This process must be repeated for various code phases until the signals are aligned. If the two vectors can be multiplied at a rate of 5.714MHz then correct phase can be determined within 2 ms as all phases can be examined.

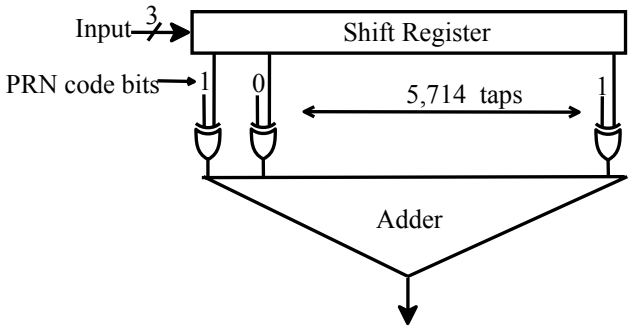

Fig 1: High-level operation of the phase-alignment system.

The operation of the phase-alignment system can be understood with the help of Fig 1. Sampled data from the Zarlink RF unit is serially entered into a 5,714 element shift register at the rate of 5.714MHz. Assume the shift register is full. The sign of each integer is multiplied by the corresponding bit of the PRN code which remains static. The output is summed to form the value to be compared against a threshold. It takes 1 ms to fill the shift register and another 1 ms to check all phases in the worst case.

A 5,714 element shift register can be easily built using CLB logic of Virtex-4 devices. However, building an adder that can add 5,714 3-bit integers at the rate of 5.714MHz requires significant logic. A solution is to use the fact that Virtex-4 chips contain many memory blocks that can be internally clocked at an order of magnitude faster than the input data rate. Therefore, it is conceivable to partition the system of Fig 1 such that each partition operates in time, rather than in space, at a much faster internal clock rate.

The XtremeDSP user guide describes an implemention of a phase-alignment system for CDMA-based applications on Virtex-4 devices [5]. It uses BRAMs and DSP blocks to implement a parallel matched filter. A single BRAM stores both data samples and filter coefficients. Data values and corresponding filter coefficients are read from memory and are fed into the DSP block which is configured as a multiply-accumulate (MAC) unit. A separate address control unit is configured in the CLB logic to generate successive memory addresses. As the internal clock rate is several times faster than the input data rate, the same logic is used to simultaneously align the signal phases for several satellites in a time-multiplexed manner.

The system we present is similar to the system described in [5] with two main differences. First, we use the difference in the internal and external clock rates to reduce the number of BRAMs to be used. This, however, does not preclude the possibility of handling multiple satellite signals. Second, the system described in [5] is generic. A mismatch between input data granularity and possible BRAM configurations can result in inefficient use of resources as discussed later.

We take these factors into account to derive a solution that makes better use of the available resources.

The approach adopted here is to construct a cascade of *basic blocks*. A basic block is essentially the system described in [5] and is also shown in Fig 2. Each basic block operates on a small subset of data values and mimics the operation of a shift register. The 1,023-bit PRN code is therefore stored in a distributed fashion in the memories of the cascaded basic blocks. Data values are read into the memory and are output to the next connected basic block. As each sample is input all values are shifted through the chain by one position and correlated with the stored code before the next sample arrives. A complete alignment is thus attempted for every input sample. Phase alignment is achieved when the product between the stored samples and the code being searched for is minimized, which is obtained after at most 2msec for 5,714 input samples as explained above.

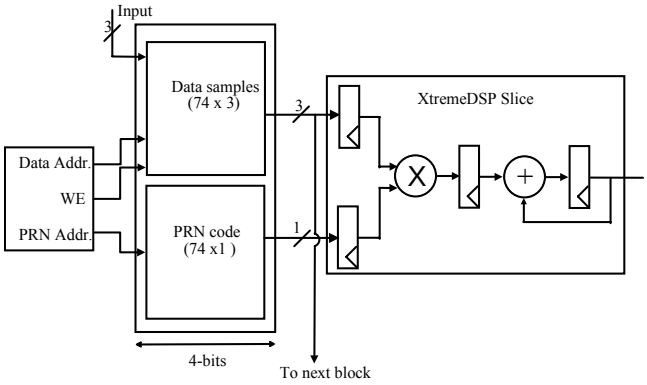

Fig 2: The architecture of a *basic block* (adapted from [3]).

In our base design, the BRAMs are configured as 4×4k words. Each basic block can operate at 450MHz which is almost 78 times faster than the external clock rate of 5.714MHz. Thus, each basic block can perform the MAC operation on 78 3-bit words before the next data value is input to the system. For an integration period of 1msec, one would require a sum of 5,714 integers, or in other words, a cascade of 74 basic blocks is required to implement the entire phase alignment system. This fits easily onto the smallest SX device but larger FX and LX devices are required to accommodate the circuit.

Each basic block collapses the sum of 78 input values within one external cycle. A separate adder circuit adds the contents of 74 MAC units. A single DSP slice can achieve this operation. The final sum is compared to a threshold to determine whether the current phase is correct or not.

Note that each basic block uses only a small proportion of the BRAM it occupies. Moreover, the required multiplication involves a simple 1-bit XOR, but the entire 18-bit multiplier of the target DSP block is used.

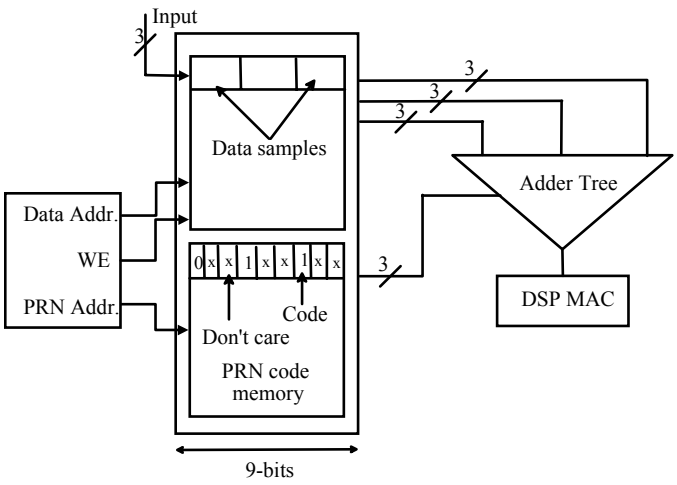

Fig 3: The architecture of Design 2.

To make better use of the available resources, alternative configurations were studied as shown in Table 2. For Design 2 each BRAM was configured as a 9×2k memory (Fig. 3). The memory was partitioned into data and PRN code values as before. However, each row contained three words instead of one. Since the DSP block can perform only one MAC at most, three 3-bit adder/subtracters were implemented in the CLBs whose sum was output to the DSP block to produce the final MAC. Table 2 shows that the clock rate of the new design was reduced due to the propagation delay of the adder logic. The total number of slices (design size) was reduced since fewer BRAMs were needed, thus saving on control circuitry.

Xilinx's XPower tool was used to estimate the power requirement of each circuit. The figures in Table 2 record the average (static + dynamic) power consumption. Significantly, Design 2 consumes less power than Design 1. The BRAM configuration 18*x*1k required the use of an adder tree in the CLB logic that multiplies 6 bits of PRN code by 6 data words. Similarly, the 36 x 512 configuration uses a circuit that adds 12 3-bit integers. The larger adder trees in Designs 3 and 4 result in higher CLB usage despite the savings in control logic for the BRAMs. This increased use of CLB logic results in higher power dissipation and increases critical path lengths, which thus reduces the clock rate. Table 2 shows that Design 2 provides a reasonable compromise over area/time and power across the range of configurations that were examined.

A direct comparison of Design 2 with commercially available systems is complicated by a lack of information. However, the sequential design reported in [2] was simulated using Xilinx ISE8.1 and it was found to use around 1% of the slices of an XCV4VFX12. In the worst case, this design aligns the phase in 1,023x1,023x2 = 2,093,058 clock ticks or around 366 ms. Operating at the same frequency as the input data rate, this design consumes around 156 mW of power or 70 times more energy than Design 2. It should be noted that the sequential design contains a code generator, whereas Design 2 stores an entire code directly. The power budget of Design 2 does not increase significantly because the BRAMs and DSP blocks are hardwired and are more power efficient. The principal shortcomings of the new designs are that the BRAMs and DSP blocks are still not fully used.

Design	BRAM Configuration	#BRAM	#Slices	Clock (MHz)	Power (mW)
1	4x4k	74	308	450	528
2	9x2k	27	255	400	411
3	18x1k	20	352	280	593
4	32x512	17	573	200	654

Table 2: Mapping results for the L1 signal assuming Zarlink's GP2015 front-end.

The system can only handle one channel of the incoming GPS signals. However, significant space exists in the BRAMs to store PRN codes for several satellites. Different satellites can thus be tracked one after another without altering the memory contents. The design can also be enhanced to handle several satellites simultaneously.

V. Conclusions and Future Work

The paper describes a system for acquiring L1 GPS signals in a time-efficient manner. The proposed system is over 180 times as fast as a sequential approach on the same device and uses less than 1.5% of the energy. This dramatic improvement results from the proposed system internally running several times faster than the data sample rate and performing the phase alignment process in parallel. The design uses power-efficient memory and DSP blocks while minimizing logic usage. Future work involves implementing a parallel design to overcome the relatively inefficient use of the memory blocks and to gauge the impact of this approach on the overall power usage.

References

[1] A. Dempster. Correlators for L2c. Inside GNSS, 2006

[2] P. Mumford, K. Parkinson and A. Dempster. An open GNSS receiver research program. International Global Navigation Satellite Systems Society, 2006

[3] Virtex-4 Datasheet. Xilinx Inc. 2005

[4] CDMA Matched Filter Implementation in Virtex devices. XAPP212, Xilinx Inc, 2001

[5] XtremeDSP for Virtex-4 FPGAs User Guide. Xilinx Inc, 2005.

[6] Frank van Diggelen and Charles Abraham, "Indoor GPS Technology", Global Locate, Inc. http://www.globallocate.com/SEMICONDUCTORS/Semi_Libr_Piece/IndoorGPSTechnology.pdf

[7] J B-Y Tsui, "Fundamentals of Global Positioing System Receivers: A Software Approach", 2nd ed., Wiley, 2005.