The Implementation of a Multiplexing Gold Code Generator using a Xilinx Logic Cell Array

Steven J. Merrifield and John C. Devlin
School of Electronic Engineering,
La Trobe University, Bundoora,
Victoria, Australia, 3083

Abstract— **This paper presents a novel system for generating, and switching between, a large number of Gold codes for use in a spread spectrum network. The underlying theory behind the choice of Gold codes is briefly discussed, and a method for reducing the amount of hardware required is introduced. The system is implemented using a Xilinx Logic Cell Array, and it is shown that use of a multiplexing arrangement can reduce the amount of hardware by a factor proportional to the number of transmitters in the network.**

I. Introduction

IN order for a code division, multiple access (CDMA) network to function effectively it is essential for each transmitter to use a unique, identifying code. In a spread spectrum network, this code is referred to as the transmitter's PN code (or Pseduo-random Noise code), and it is this code which is used to modulate the transmitted data, thereby spreading the signal.

The network discussed here assumes the use of a single receiver with multiple transmitters. Each transmitter is allocated a unique spreading code and the receiver switches rapidly between each code, resynchronising to the required transmitter's PN code as it switches.

In order for a receiver to de-spread the required signal it must be able to generate the relevant spreading code and synchronise itself to the incoming bitstream. With a conventional transmitter-receiver pair, the amount of hardware can be excessive and difficult to optimise, but with a large network it is possible to multiplex the receiver's locally generated code to switch between transmitters. By multiplexing at the receiver, a significant reduction in the amount of hardware is possible (directly proportional to the number of transmitters). The reduction in hardware can be achieved through the use of a conventional spread spectrum receiver which is able to dynamically reconfigure itself to synchronise to different spreading codes. Such a system is presented here, and consists of a Xilinx Logic Cell Array (LCA) containing a lookup table, the code generating logic, and a multiplexing subsystem.

II. Choosing a Gold Sequence

In spread spectrum applications it is desirable to use balanced sequences, that is, there is an almost equal number of ones and zeros in the pseudo random code (in fact the number of ones must exceed the number of zeros by one). This is essential since it ensures that any DC component of the sequence can be neglected since any code imbalance will be clearly obvious as unwanted peaks in the resulting spectrum; thus giving a clear indication of the code clock rate. Such a feature is undesirable as it allows a third party to locate, and possible interfere with, the transmitted signal. Since one of the motivations behind a spread spectrum system is to conceal the signal, there is little point in advertising it's presence with clearly visible peaks at multiples of the chipping rate.

Gold codes are ideally suited for use in a CDMA network because of their near optimum cross correlation properties [1]. In addition to this, they also possess the desirable characteristic of being able to generate a large number of different sequences for a given polynomial combination.

It has been established that Gold codes have three–level cross correlation values [2]. If we assume a generator with n stages, producing a PN code of length $L = 2^n - 1$, then it is seen that when n is even (and not 0 mod 4), 75% of the codes generated have a cross correlation bound at $-1/L$.

Gold codes are readily constructed using two related maximal length sequences (also known as m sequences). It is generally accepted that two fixed length shift registers are used, each with fixed feedback taps. The position of these taps is derived from the characteristic phases of the chosen maximal sequences, and determines the overall characteristics of the code generator. Each shift register is loaded with a set of initial conditions - these initial conditions determine the actual code to be generated. In order to generate a number of different codes with the same hardware, all that is required is a change in the contents of one of the initial shift registers; the other does not need to be adjusted. The configuration of the hardware remains the same, and does not need to be altered. Hence, we can produce a large number of Gold codes with a simple piece of fixed hardware, and a variable input to one of the shift registers.

The choice of initial conditions to one of the shift registers is very important. It is possible that any random choice of bits will produce a PN code, but it is highly likely that such a code will not be balanced, and will not possess an optimum cross correlation value. In order to produce an efficient code, a fixed procedure must be followed. Such a procedure is complicated, and involves the long division of polynomials. Once such a division has been performed, the calculation of optimum cross correlation codes then follows relatively easily. By shifting one of the sequences past the other, and xoring them together, another Gold code is produced. This Gold code may, or may not be balanced, but since balanced codes are preferable, it is important to choose the original code combination carefully.

Bekir [3] used moments to establish the upper and lower

bounds on the probability functions for cross correlation values of sets of Gold sequences. Weber *et al.* [4] was then able to conclude that for a set of Gold sequences of degree 13, the effect of other users in a CDMA network can effectively be modelled as a Gaussian random variable at the receiver. Thus, for the network discussed here, a code sequence of degree 13 was chosen.

Following the Gold–derived algorithm outlined by Dixon [5], a pair of preferred polynomials was chosen to give the required cross correlation properties. Peterson and Weldon [6] present a table of irreducible polynomials up to degree 34. Choosing the initial polynomial of degree 13 as 20033, it follows from Dixon [5] that the second polynomial can be calculated using the relation $2^{\frac{(n-1)}{2}} + 1 = 65$ where n is the degree of the sequence. Using the tables from Peterson and Weldon [6], we find that for a sequence of degree 13, with polynomial root 65, our required polynomial becomes 33343. Hence our two polynomials are

$$20033 \rightarrow 010\ 000\ 000\ 011\ 011$$

ie,

$$f(x) = x^{13} + x^4 + x^3 + x + 1 \tag{1}$$

and,

$$33343 \rightarrow 011\ 011\ 011\ 100\ 011$$

ie,

$$g(x) = x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^6 + x^5 + x + 1 \tag{2}$$

The characteristic sequence generated by the shift register corresponding to polynomial 20033 is given by $h(x)/f(x)$ where

$$h(x) = \frac{d(x \cdot f(x))}{dx} \tag{3}$$

with co–efficients interpreted mod 2 ie,

$$\frac{\frac{d}{dx}(x + x^2 + x^4 + x^5 + x^{14})\ mod\ 2}{1 + x + x^3 + x^4 + x^{13}} \tag{4}$$

now the numerator is given by

$$(1 + 2x + 4x^3 + 5x^4 + 14x^{13})\ mod\ 2 = 1 + x^4 \tag{5}$$

thus the characterisitic phase (the initial conditions) can be found from the quotient

$$\frac{1 + x^4}{1 + x + x^3 + x^4 + x^{13}} \tag{6}$$

$$= 1 + x + x^2 + 0x^3 + x^4 + x^5 + 0x^6 + x^7 + x^8 + 0x^9 + x^{10} + x^{11} + 0x^{12} + \ldots \tag{7}$$

hence the initial conditions, 1 1 1 0 1 1 0 1 1 0 1 1 0.

These initial conditions determine the power–on state of the lower shift register, and do not need to be altered to produce different Gold codes. In order to generate an alternative Gold code, the initial conditions of the upper shift register are loaded with different values. The only restriction on these values is that the first stage (the one on the right) contain a zero. If this is not the case, an unbalanced code will be produced and as a result will possess a non–zero DC offset. The structure of the resulting Gold code generator is shown in figure 1, where the lower shift register has a fixed set of initial conditions, and the upper register can be dynamically configured.

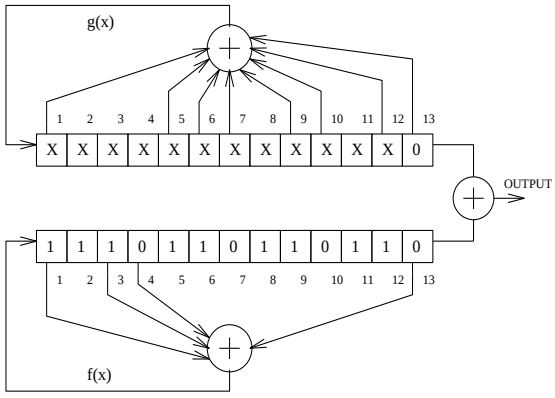

Fig. 1. 13 stage Gold code generator

III. Hardware Implementation

Two different hardware configurations were implemented. The first was a very simple multiplexing arrangement with a number of obvious disadvantages; and was constructed to provide a reference with which to compare a more realistic design. The second hardware implementation is a much more efficient structure, and has a large number of advantages over the simple multiplexer.

The hardware designs presented here were verified using a Xilinx XC4003A Logic Cell Array (LCA), commonly known as a field programmable gate array (FPGA). This device was chosen simply because a development system was already available which allowed immediate verification of hardware performance. The XC4003A has 100 configurable logic blocks (CLBs), and a typical gate count of 3000 gates. This device also has the ability to provide user–defined ROM, which is exploited in the optimised design presented here.

In order to prove the effectiveness of an optimised multiplexing arrangement, an initial hardware layout was constructed whereby copies of the basic Gold code generator were used. Each copy used an identical shift register configuration, but the initial conditions fed to the upper register in each case differed. The output of each of these individual Gold code generators were then time division multiplexed into a single output. The overall structure of this device is shown in figure 2.

As would be expected, such an implementation is not very efficient, and there are obvious drawbacks to this design. In particular, the number of CLBs required is directly proportional to the number of transmitters in the network. For a large network, the available on–chip resources will very quickly become exhausted. It is clear that only one particular sequence generator is in use at any one in-

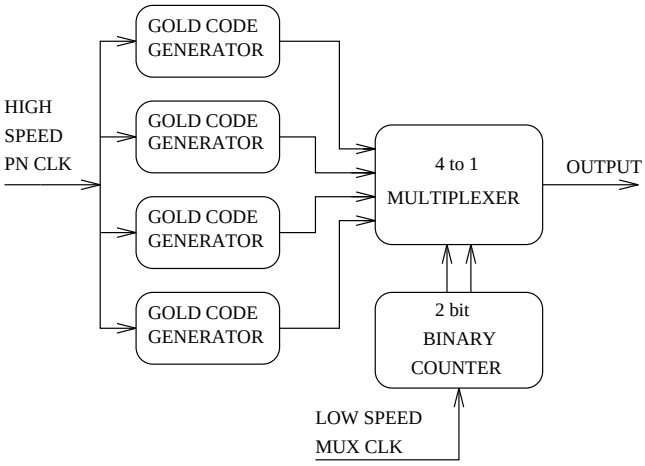

Fig. 2. Simple multiplexing system

stant in time. The others are simply occupying resources which may be better utilised.

Using an XC4003APC84–5 as the target device, the structure shown in figure 2 occupied 106 of the possible 200 CLB flip–flops. The maximum clock speed was 29.6 MHz before any optimisation. By specifying critical nets, and applying a lower priority to the slow speed multiplexing circuitry, the maximum clock rate was increased to 36.6 MHz. The number of CLBs is very limiting, since there is only provision for seven transmitters per receiver before the gate array is fully utilised. It is clear that another method of multiplexing between sequences is required.

Since each Gold code generator has 13 stages, the maximal length of the resulting code is $2^{13} - 1 = 8191$ bits. We must subtract one from the total since we have excluded the all–zeros state which would cause the sequence generator to become locked, only generating a zero output. It is important to note that in a Xilinx FPGA implementation, the conventional XOR gates in the shift register feedback are replaced with XNOR gates and the all–ones state is excluded. This is because under normal operation, Xilinx flip–flops wake up in the all–zeros state [7].

In order to reduce the amount of hardware required, and to increase the utilisation of the available FGPA resources, an alternative method of generating a large number of Gold codes is presented. Such a method only requires a single pair of shift registers, with one having a fixed preset condition, and the other having a dynamic allocation of initial conditions. The block diagram is presented in figure 3; and includes a lookup table in ROM which contains the necessary initial conditions for the upper shift register.

Alfke [8] presents a method of producing very efficient pseudo–random sequence generators using a Xilinx gate array. By replacing the conventional flip-flop design with a RAM-based design, an enormous reduction in CLB occupancy results. Alfke was able to take a 63 bit shift register, normally requiring 32 CLBs, and implement it using only four CLBs. Unfortunately, such a RAM-based approach is not possible in our case because of the selective feedback

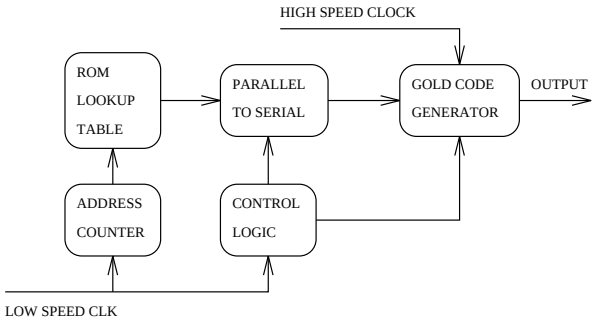

Fig. 3. Optimised hardware solution

required to produce Gold codes. Alfke's choice of a 63 bit generator was deliberate as it provides conveniently simple feedback. Although using a RAM-based approach was not possible, the method employed by Alfke to load the initial values is the same as that used here.

It was decided to use a lookup table of eight bit words, since this allowed the use of exisiting Xilinx symbols such as an eight bit parallel to serial convertor. The choice of eight bit words limits the number of configurations to $2^8 = 256$, which is considered sufficient for the network under discussion. The depth of the lookup table determines the number of possible transmitters in the network. For example, an initial choice of four entries was decided upon to allow a comparison with the simple multiplexing circuit. The complete system utilising a 4 x 8 lookup table required 48 CLB flip-flops, which is less than that of the basic multiplexer circuit. This optimised design was able to achieve a clock speed of 62.9 MHz, again exceeding the performance of the basic design. The comparison between these two designs in shown in table 1.

TABLE I
System comparison using four transmitters

	Simple MUX	Lookup Table
CLB flip-flops	106	48
Function gen's.	15	29
Clock rate	36.6 MHz	62.9 MHz
Routing time	4:30 min	3:40 min

Where possible, all parallel paths were converted to serial bit streams in an effort to minimise logic congestion, and thus increase the routing performance. Although it is simpler to load the initial conditions into the shift register using a parallel load technique, a serial based approach is more efficient. The parallel output from the ROM was converted into a serial stream, and then clocked into the Gold code generator using the same high speed clock as used for code generation. This technique required the implementation of dedicated control logic to disable the feedback from the intermediate shift register stages, and allow the loading process to take place.

In order to show the effectiveness of implementing this

design using a ROM based approach, the length of the table was increased to a maximum of 256 entries. This is the maximum length of a RAM or ROM based memory element implemented in a Xilinx XC4000 device. Figure 4 shows the performance characteristics of this design for a varying number of entries in the lookup table. It can be seen that such a design far outperforms the simple multiplexing system, both in terms of resource utilisation, and the maximum number of transmitters which can be accomodated. The maximum clock speed attainable by both designs using a varying number of transmitters is shown in figure 5. Again, it is seen that a ROM based multiplexing Gold code generator offers superior performance to a simple time-division multiplexing sequence generator.

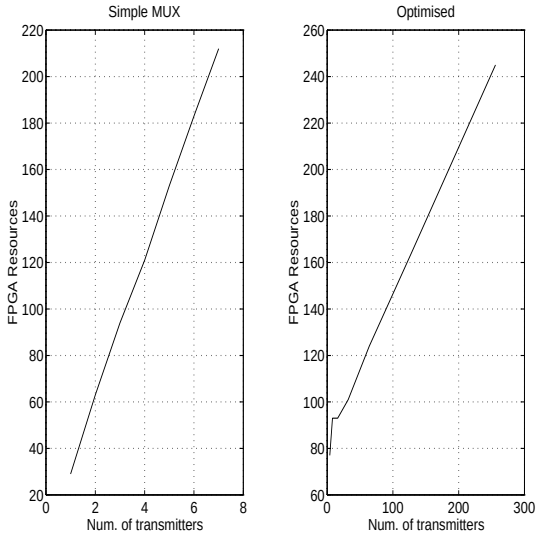

Fig. 4. FPGA utilisation for both structures

IV. Conclusion

A method of multiplexing between a large number of psuedo–random Gold codes has been presented. Such a system is likely to prove invaluable for CDMA networks where a single receiver and several transmitters are used. A comparison was drawn between a simple time division multiplexing system, and a design optimised to reduce the amount of hardware. The ability to configure the Xilinx 4000–series devices as ROM was exploited in the design of the optimised solution. It was shown that use of a dedicated Gold code generator for each PN sequence dramatically reduces the number of transmitters in a single receiver network. Using a lookup table to dynamically configure a single Gold code generator was shown to not only increase the number of allowable transmitters, but would also reduce the amount of hardware required. It is the authors' opinion that with further work an increase in clock speed is likely, along with a possible reduction in the CLB occupancy. Currently work is progressing towards the development of a delay-lock loop based receiver using the optimised switching method presented here.

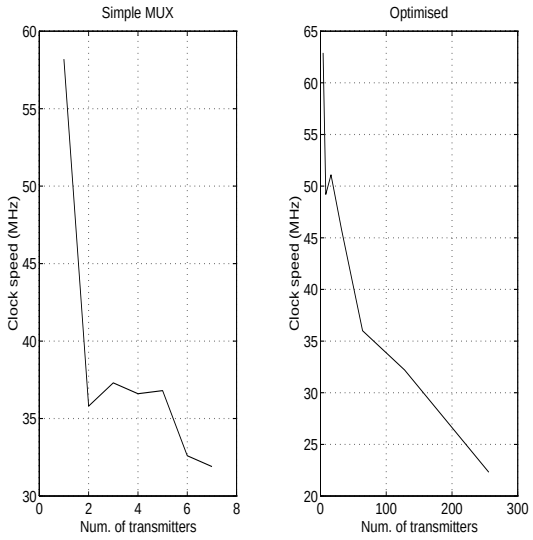

Fig. 5. Maximum clock speed for both structures

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful suggestions, and proofreading of this paper.

References

[1] R. Gold, *Optimal Binary Sequences for Spread Spectrum Multiplexing*, IEEE Trans. Information Theory, Oct 1967, pp. 619-621
[2] R. Gold, *Maximal Recursive Sequences with 3-valued Recursive Cross-Correlation Functions*, IEEE Trans. Information Theory, Jan. 1968, pp. 154-156
[3] N.E. Bekir, *Bounds on the distribution of partial correlation for PN and Gold Sequences*, Ph.D. dissertation, Dep. Elec. Eng., Univ. of Southern California, Los Angeles, Jan. 1978
[4] C.L. Weber, G.K. Huth, B.H. Batson, *Performance Considerations of Code Division Multiple-Access Systems*, IEEE Trans. Vehic. Tech., Vol. VT-30, No. 1, Feb 1981, pp. 3-10
[5] R.C. Dixon, *Spread Spectrum Systems*, 2nd Ed. John Wiley and Sons, New York, 1984, pp. 403-405.
[6] W. Peterson and E. Weldon, *Error Correcting Codes*, 2nd Ed. MIT Press, 1972, pp.472-491.
[7] P. Alfke, *The Programmable Logic Data Book*, Xilinx Corp., 1984, p. 9-24
[8] P. Alfke, *Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators*, Application Note, Xilinx Corp., Aug. 1995