18-10-2006

Low-complexity generation of scalable complete complementary sets of sequences

Darryn Lowe
University of Wollongong, darrynl@uow.edu.au

Xiaojing Huang
University of Wollongong, huang@uow.edu.au

Low-complexity generation of scalable complete complementary sets of sequences

Abstract

This paper presents extremely low-complexity boolean logic for the generation of coefficients suitable for filtering or correlation of scalable complete complementary sets of sequences (SCCSS). As the unique auto- and cross-correlation properties of SCCSS are of broad interest, the simplicity of the proposed coefficient generation technique allows arbitrarily long SCCSS to be used in resource constrained applications.

Keywords

complementary sequences

Disciplines

Physical Sciences and Mathematics

Publication Details

This paper was originally published as: Lowe, D & Huang, X, Low-complexity generation of scalable complete complementary sets of sequences, 2006 International Symposium on Communications and Information Technologies (ISCIT), Bangkok, Thailand, 18-20 October 2006. Conference information is available here.

Low-Complexity Generation of Scalable Complete Complementary Sets of Sequences

Darryn Lowe and Xiaojing Huang
School of Electrical, Computer and Telecommunications Engineering
University of Wollongong
Wollongong, Australia, 2522
Email: {darrynl, huang}@uow.edu.au

***Abstract*— This paper presents extremely low-complexity boolean logic for the generation of coefficients suitable for filtering or correlation of scalable complete complementary sets of sequences (SCCSS). As the unique auto- and cross-correlation properties of SCCSS are of broad interest, the simplicity of the proposed coefficient generation technique allows arbitrarily long SCCSS to be used in resource constrained applications.**

I. Introduction

Spreading sequences are fundamental to numerous signal processing applications such as spread-spectrum communication systems and image processing. Although the quality of a given spreading sequence is largely dependent on its auto-correlation and cross-correlation functions, the computational effort needed to generate the sequence is also an important consideration. For example, even though an evolutionary algorithm can find sequences that asymptotically meet almost any arbitrary requirement [1], resource constraints force many applications to use general-purpose sequences such as Walsh-Hadamard, Gold or Kasami sequences.

Spreading sequences are usually applied via a digital correlator or filter. To minimize gate count, it is often desirable to pool hardware for these kinds of expensive operations. For example, a single bank of hardware multipliers may realize different filters at different times to avoid unnecessarily duplicating logic.

With a goal of minimizing the cost needed to parameterize a shared filter or correlator, this paper shows how coefficients for powerful scalable complete complementary sets of sequences (SCCSS) [2] can be generated using as little as one or two boolean operations per filter tap. This is significant since low-complexity low-power applications are often unable to store large codesets in read-only memory (ROM).

As well as having very attractive auto- and cross-correlation functions, as detailed below, SCCSS have three other distinguishing characteristics. First, scalability means that each set includes all sets of smaller size. This is of particular benefit to communications systems as it can provide adaptable user-specific process gains. Second, completeness makes SCCSS more efficient than randomly generated sequences. For example, a SCCSS-based spread-spectrum mobile telephony system could offer higher data rates and/or support more users. Thirdly, the complementary property makes it possible to design novel synchronization algorithms that exploit the periodic zeros in the cross-correlation and auto-correlation functions for faster convergence.

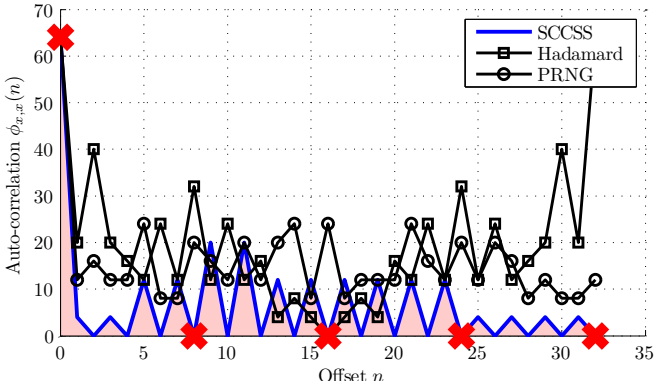

Fig. 1. Comparison of worst-case auto-correlation for length 64 sequences.

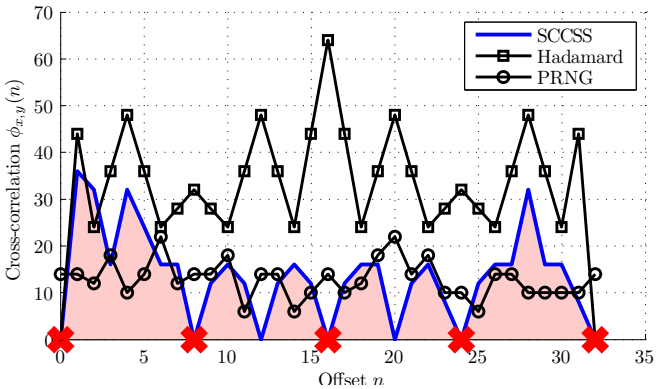

Fig. 2. Comparison of worst-case cross-correlation for length 64 sequences.

Fig. 1 and Fig. 2 show respective examples of the auto-correlation and cross-correlation functions of a SCCSS relative to sequences derived from Walsh-Hadamard and pseudo-random number generators (PRNGs).

Both examples use codesets comprised of 8 sequences of 64 chips each. Worst-case results are considered by taking the highest correlation for a given offset n over all 8 codes. It can be observed that while SCCSS and Walsh-Hadamard sequences are perfectly orthogonal at $n = 0$, the PRNG sequences are only ever approximately orthogonal. Further,

when $n > 0$, we see that the Walsh-Hadamard sequences have significant sidelobes that are much higher than those of SCCSS. The complementary nature of SCCSS, further discussed in Section II, is also apparent by virtue of how the auto-correlation and cross-correlations are perfectly orthogonal about all offsets that are integer multiples of 8.

This paper is organized as follows. In Section II, we begin by reviewing the construction and properties of SCCSS. Then, in Section III, we derive novel combinatorial logic that can generate coefficients appropriate for use in reconfigurable finite impulse response (FIR) filters. Using a VHDL example, we evaluate the complexity of the proposed scheme in Section IV. Finally, Section V concludes the paper and identifies future work.

II. Theory

A Hadamard matrix of order N is a square $\{-1, +1\}$ matrix of dimension $N \times N$ that satisfies

$$\mathbf{H}_N \mathbf{H}_N^T = N\mathbf{I}_N \tag{1}$$

where I_N denotes the $N \times N$ identity matrix and $\mathbf{H}_N^T$ denotes the transpose of Hadamard matrix $\mathbf{H}_N$. Further, a Golay-paired Hadamard matrix is defined by [2] as

$$\mathbf{H}_{2N} = \begin{bmatrix} \mathbf{H}_N & \widetilde{\mathbf{H}}_N \\ \mathbf{H}_N & -\widetilde{\mathbf{H}}_N \end{bmatrix}. \tag{2}$$

When $\mathbf{H}_1 = 1$ and $\widetilde{\mathbf{H}}_N = \mathbf{H}_N$, (2) defines the well-known Walsh-Hadamard family of matrices. Alternatively, when $\widetilde{\mathbf{H}}_N$ is constructed by commutating the upper- and lower-halves of $\mathbf{H}_N$, we have the basis to build sets of mutually orthogonal Golay-paired Hadamard matrices.

Golay-paired Hadamard matrices can also be generated through a closed-form expression [2]. Specifically, the value at row i and column j of the k^{th} matrix in a set of order $N = 2^n$ is denoted by

$$\mathbf{H}_{2^n}^{(k)}(i,j) = (-1)^{\Gamma} \tag{3}$$

where

$$\Gamma = \sum_{r=0}^{n-2} (j_{r+1} \oplus i_r \oplus k_r)\, j_r \oplus (i_{n-1} \oplus k_{n-1})\, j_{n-1} \tag{4}$$

and k_r denotes the r^{th} bit in the radix-2 expression[1] of k as per

$$k \equiv (k_{n-1}, k_{n-2}, ..., k_0)_2 = \sum_{r=0}^{n-1} 2^r k_r \tag{5}$$

and likewise for j_r and i_r.

[1]All radix-2 notations are written MSB to LSB

For example, the $k = 0$ Golay-paired Hadamard matrix in the set of order $N = 4$ is

$$\mathbf{H}_4^{(0)} = \begin{bmatrix} \mathbf{H}_4^{(0)}(0,0) & \mathbf{H}_4^{(0)}(0,1) & \mathbf{H}_4^{(0)}(0,2) & \mathbf{H}_4^{(0)}(0,3) \\ \mathbf{H}_4^{(0)}(1,0) & \mathbf{H}_4^{(0)}(1,1) & \mathbf{H}_4^{(0)}(1,2) & \mathbf{H}_4^{(0)}(1,3) \\ \mathbf{H}_4^{(0)}(2,0) & \mathbf{H}_4^{(0)}(2,1) & \mathbf{H}_4^{(0)}(2,2) & \mathbf{H}_4^{(0)}(2,3) \\ \mathbf{H}_4^{(0)}(3,0) & \mathbf{H}_4^{(0)}(3,1) & \mathbf{H}_4^{(0)}(3,2) & \mathbf{H}_4^{(0)}(3,3) \end{bmatrix}$$
$$= \begin{bmatrix} +1 & +1 & +1 & -1 \\ +1 & -1 & +1 & +1 \\ +1 & +1 & -1 & +1 \\ +1 & -1 & -1 & -1 \end{bmatrix} \tag{6}$$

A complete set of N mutually orthogonal Golay-paired Hadamard matrices of dimension $N \times N$ denotes a SCCSS of order $N = 2^n$ with each sequence in the SCCSS 2^{2n} chips in length. For example, when $n = 3$, we can construct $2^n = 8$ mutually orthogonal Golay-paired Hadamard matrices of dimension 8×8 via (3) with each matrix forming a single sequence in the SCCSS. If we denote the k^{th} sequence of a SCCSS of order 2^n as $c_{2^n}^{(k)}(t)$, then we can denote each chip in the sequence as

$$c_{2^n}^{(k)}(t) = c_{2^n}^{(k)}(i + 2^n j) = \mathbf{H}_{2^n}^{(k)}(i,j) \tag{7}$$

where $0 \le i < N$ and $0 \le j < N$.

The complementary property that distinguishes SCCSS means that each sequence in the set is orthogonal to all other sequences in the set about shifts of integer multiples of the set size. For example, if we choose two sequences $c_{2^n}^{(x)}(t)$ and $c_{2^n}^{(y)}(t)$ such that $x \neq y$, we can state that the normalized auto-correlation and cross-correlation are respectively

$$\phi_{c_{2^n}^{(x)}, c_{2^n}^{(x)}}(m) = \begin{cases} 1 & \text{if } m = 0 \\ 0 & \text{if } m = K2^n \end{cases} \tag{8}$$

and

$$\phi_{c_{2^n}^{(x)}, c_{2^n}^{(y)}}(m) = 0 \quad \text{if } m \neq \{0, K2^n\} \tag{9}$$

where $K = \{1, 2, ..., 2^n - 1\}$.

III. Digital Sequence Generation

As the realization of a digital filter can greatly increase a design's total gate count, it is often desirable to have mutually exclusive processes share hardware. In other words, rather than duplicate the filter logic, it can be more efficient to use a single digital filter with variable filter coefficients [3].

Fig. 3a shows a portion of a generic FIR filter where incoming data samples $d(0)$ through $d(t)$ are respectively multiplied by the filter coefficients $c(0)$ through $c(t)$, with the results added together to create the output y. Digital implementations of this filter structure are well understood [4][5] and it can be easily seen that N multiplications and $N - 1$ additions will be required for N taps. If the filter coefficients $c(t)$ are limited to $\{1, -1\}$, as is the case when correlating binary spreading codes like SCCSS, multiplication by $c(t)$ is equivalent to addition and subtraction for $c(t) = 1$ and $c(t) = -1$ respectively.

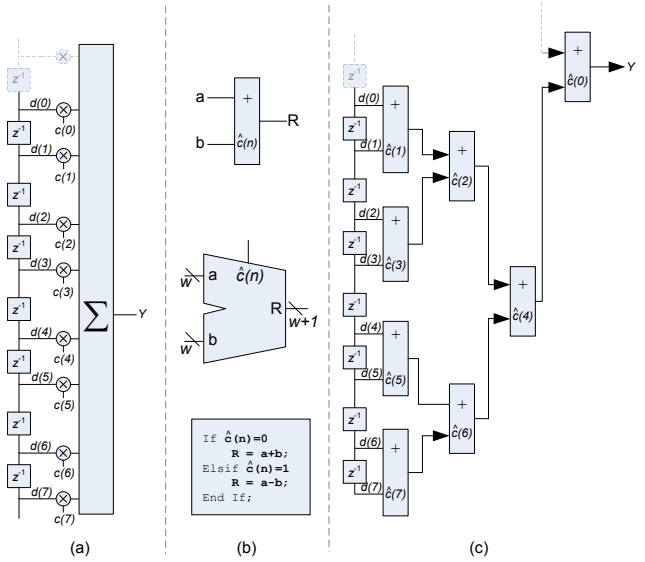

Fig. 3. Parallel FIR filter with a) General Architecture, b) 2-Port Adder/-Subtractor, c) 2-Port Adder/Subtractor Hierarchy

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity top is
  port(clk : in std_logic;
       t : in std_logic_vector(5 downto 0);
       k : in std_logic_vector(2 downto 0);
       c : out std_logic);
end top;

architecture rtl of top is
begin
  process(clk)
  begin
    if (clk='1' and clk'event) then
      if    t(0) = '1' then c <= t(3);
      elsif t(1) = '1' then c <= t(4);
      elsif t(2) = '1' then c <= t(5);
      elsif t(3) = '1' then c <= t(4) xor k(0);
      elsif t(4) = '1' then c <= t(5) xor k(1);
      elsif t(5) = '1' then c <= k(2);
      else                  c <= '0';
      end if;
    end if;
  end process;
end rtl;
```

Listing 1. VHDL to generate filter coefficients for a SCCSS of length 64.

Since efficient pipelined digital adder/subtractors are limited to two inputs, as shown in Fig. 3b, the large N port adder of Fig. 3a must be converted into an adder-tree with $\log_2 N$ levels as shown in Fig. 3c. Modified filter coefficients $\hat{c}(t) = 0, 1$ determine how each adder/subtractor is used. In other words, an adder/subtractor implements $R = A + B$ if $\hat{c}(t) = 0$ and $R = A - B$ if $\hat{c}(t) = 1$. Note that there is no loss of generality by limiting each adder/subtractor to $A \pm B$ rather than $\pm A \pm B$ [6].

With the cost of the shared filter hardware amortized over multiple operations, we can further reduce the gate count by storing each set of filter coefficients in as few gates as possible. In addition to their other properties, SCCSS are extremely attractive in this regard since it is possible to generate the modified 2-port adder/subtractor coefficients $\hat{c}(t)$ used in Fig. 3c through simple combinatorial logic.

With the full derivation provided in the appendix, modified 2-port adder/subtractor coefficients for a SCCSS of order 2^n can be denoted as

$$\hat{c}_{2^n}^{(k)}(t) = \begin{cases} t_{l+n} & 0 \le l \le n-1 \\ t_{l+1} \oplus k_{l-n} & n \le l \le 2n-2 \\ k_{l-n} & l = 2n-1 \\ 0 & otherwise \end{cases} \qquad (10)$$

where l is the index of the least significant non-zero bit in the radix-2 representation of t. For example, consider the filter coefficient at $t = 34$ for code $k = 5$ in a set of order $n = 3$. Here, $t = 34 = (100010)_2$ and $k = 5 = (101)_2$. The least significant non-zero bit in t is second from the right. Therefore, by substituting $l = 1$ into (10) we find that $\hat{c}_{2^n}^{(k)}(t) = \hat{c}_8^{(5)}(34) = t_{l+n} = t_{1+3} = t_4 = 0$.

An example of a VHDL realization of an SCCSS generator appropriate for the FIR filter of Fig. 3c is provided in Listing 1. The complexity of this generator is very low; the modified filter coefficients are calculated via simple boolean logic and no internal counters are needed. However, despite the low complexity, this component is functionally equivalent to a 2^{2n}-bit wide ROM 2^n words deep that is preloaded with modified SCCSS-coefficients.

IV. Results and Analysis

In this section, we compare the complexity of the modified SCCSS-coefficient generator of Listing 1 to that of ROMs and PRNGs. In all comparisons, logic utilization figures were obtained from Synplify Pro and the application specific integrated circuit (ASIC) gate count estimates were calculated as per [7].

Pseudo-random sequences, as well as derivatives like Gold codes, can be very efficiently implemented in digital hardware [8], [9] since a pseudo-random sequence 2^n chips long requires a linear-feedback shift-register (LFSR) with only n flip-flops. Although codes produced in this way are not perfectly orthogonal, the ease with which very long sequences can be generated makes this approach very common. Table I shows how a PRNG compares to the SCCSS generator. Although both techniques are low cost, the guaranteed orthogonality, scalability, and completeness of the SCCSS more than justifies its marginally higher complexity. Further, another weakness of LFSR-based codeset generation is that the code chips must be generated recursively. In other words, it is not possible to obtain the n^{th} chip without first calculating the $(n-1)$ preceding chips. The SCCSS generator does not have this limitation since it uses a single counter, the cost of which

	SCCSS			PRNG		
Code Length	64	256	1024	64	256	1024
Flip-Flops	10	13	16	6	8	10
4-Input LUTs	17	24	31	1	1	1
ASIC Gates	182	248	321	54	70	120

TABLE I

COMPARISON OF COMPLEXITY OF SCCSS AND PRNG GENERATION

is included in the comparison, to identify the desired chip.

Another alternative to dynamically generating the codeset is to pre-calculate the coefficients and store them in a ROM. Although this allows for sequences with arbitrary auto- and cross-correlation functions, the storage is grossly inefficient since roughly 2^{3n} ASIC gates are required to store 2^n sequences of length 2^{2n}. For example, in the case of an order $2^n = 32$ SCCSS, where each sequence is 1024 chips long, a 32kb ROM would be needed. This is several orders of magnitude more gates than needed for the SCCSS generator.

V. CONCLUSION

An extremely efficient boolean expression to calculate FIR filter coefficients for SCCSS were derived. Since SCCSS have auto-correlation and cross-correlation functions that are superior to pseudo-random and Walsh-Hadamard sequences in several ways, devices such as spread-spectrum communication transceivers can exploit SCCSS for improved end-to-end performance with nominal increase in computational complexity.

REFERENCES

[1] H. Dong, B. Wang, C. Gu, and G. Feng, "Synthesis of binary sequences with good auto- and cross-correlation properties by GA," in *Proceedings of International Conference on Communication Technology*, Oct. 1998.

[2] X. Huang and Y. Li, "Scalable complete complementary sets of sequences," in *GLOBECOM 2002, Taipei, Taiwan*, Nov 2002.

[3] S.-W. Kim and B. Daneshrad, "A 100 μW, 20 Mcps versatile correlator chip for third geneation WCDMA systems," in *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers, vol. 1, pp. 130134*, Oct 1999.

[4] B. A. Harvey and C. Bouey, "FPLD implementation of a PN-matched filter correlator," in *IEEE SoutheastCon*, 2003.

[5] P. M. Shriram Kulkarni and G. I. Haddad, "A high-speed 32-bit parallel correlator for spread spectrum communications," in *Proceedings of 9th International Conference on VLSI Design*, Jan 1996.

[6] A. M. Ken Chapman, Paul Hardy and M. George, "XAPP212: CDMA matched filter implementation in virtex devices," Xilinx, Inc.," Application Note, 2001.

[7] "XAPP059: Gate count capacity metrics for FPGAs," Xilinx, Inc.," Application Note, 1997.

[8] A. Miller and M. Gulotta, "XAPP211: PN generators using the SRL macro," Xilinx, Inc.," Application Note, 2004.

[9] M. H. Maria George and A. Miller, "XAPP217: Gold code generators in Virtex devices," Xilinx, Inc.," Application Note, 2001.

APPENDIX

The relationship between the original coefficients $c(t)$ and the modified 2-port adder/subtractor coefficients $\hat{c}(t)$ can be derived by equating the Y outputs in Fig. 3a and 3c as per

$$\begin{aligned}\sum_{t=0}^{N-1} d(t)c(t) = \quad & \\ & d(0)(-1)^{\hat{c}(0)} \\ & + d(1)(-1)^{\hat{c}(1)\oplus\hat{c}(0)} \\ & + d(2)(-1)^{\hat{c}(2)\oplus\hat{c}(0)} \\ & + d(3)(-1)^{\hat{c}(3)\oplus\hat{c}(2)\oplus\hat{c}(0)} \\ & + d(4)(-1)^{\hat{c}(4)\oplus\hat{c}(0)} \\ & + d(5)(-1)^{\hat{c}(5)\oplus\hat{c}(4)\oplus\hat{c}(0)} \\ & + d(6)(-1)^{\hat{c}(6)\oplus\hat{c}(4)\oplus\hat{c}(0)} \\ & + d(7)(-1)^{\hat{c}(7)\oplus\hat{c}(6)\oplus\hat{c}(4)\oplus\hat{c}(0)}\end{aligned} \tag{11}$$

Note that this equivalence is valid only for binary sequences, i.e. for original filter coefficients $c(t) = \{+1, -1\}$ and modified filter coefficients $\hat{c}(t) = \{0, 1\}$.

The like-terms of $d(t)$ can be equated and the resulting simultaneous equations solved to yield the modified filter coefficients $\hat{c}(t)$. By expressing the index t in radix-2 notation, we can state that

$$\begin{aligned}(-1)^{\hat{c}(000_2)} &= c(000_2) \\ (-1)^{\hat{c}(001_2)} &= c(001_2)c(000_2) \\ (-1)^{\hat{c}(010_2)} &= c(010_2)c(000_2) \\ (-1)^{\hat{c}(011_2)} &= c(011_2)c(010_2) \\ (-1)^{\hat{c}(100_2)} &= c(100_2)c(000_2) \\ (-1)^{\hat{c}(101_2)} &= c(101_2)c(100_2) \\ (-1)^{\hat{c}(110_2)} &= c(110_2)c(100_2) \\ (-1)^{\hat{c}(111_2)} &= c(111_2)c(110_2)\end{aligned}$$

Through induction, we can therefore denote the general case as

$$(-1)^{\hat{c}(t)} = \begin{cases} c(t) & t = 0 \\ c(t)c(m) & t \neq 0 \end{cases} \tag{12}$$

where m is equivalent to t with the least significant non-zero bit flipped.

When the filter coefficients denote a SCCSS, i.e. $c(t) = c_{2^n}^{(k)}(t)$, the modified filter coefficients can be calculated using simple boolean logic. We begin this derivation by observing that the least significant n bits of the radix-2 expression of t denote the row i while the most significant n bits denote the column j. In other words,

$$i = \sum_{r=0}^{n-1} 2^r t_r = (t_{n-1}, t_{n-2}, ..., t_0)_2 \tag{13}$$

and

$$j = \sum_{r=n}^{2n-1} 2^{r-n} t_r = (t_{2n-1}, t_{2n-2}, ..., t_n)_2 \tag{14}$$

By substituting (3), (13) and (14) into (12) we obtain five distinct cases that are delineated by the index t.

A. Case 1: t=0

The first modified filter coefficients, where $t = i = j = 0$, of any sequence in a SCCSS is

$$\begin{aligned}\hat{c}(t) &= \log_{-1} c(t) \\ &= \log_{-1} c^{(k)}_{2^n}(t) \\ &= \log_{-1} \mathbf{H}^{(k)}_{2^n}(i,j) \\ &= \sum_{r=0}^{n-2} (j_{r+1} \oplus i_r \oplus k_r)\, j_r \oplus (i_{n-1} \oplus k_{n-1})\, j_{n-1} \\ &= \sum_{r=0}^{n-2} (0 \oplus 0 \oplus k_r)\, 0 \oplus (0 \oplus k_{n-1})\, 0 \\ &= 0 \end{aligned} \tag{15}$$

We conclude that the first modified filter coefficient for all SCCSS sequences is 1.

B. Case 2: $0 \leq l \leq n-2$

For all chips $t > 0$, the modified filter coefficients can be denoted

$$\begin{aligned}\hat{c}(t) &= \log_{-1} c(t)c(m) \\ &= \log_{-1} \mathbf{H}^{(k)}_{2^n}(i,j)\mathbf{H}^{(k)}_{2^n}(i',j') \end{aligned} \tag{16}$$

where the values of i' and j' depend on the location of the least significant '1' bit in the radix-2 representation of t. We now expand (16) into (22) by substituting (3).

By definition, this requires the least significant '1' bit in t to be in the lower half of the radix-2 representation. This places the focus on the row index i since i' will differ from i by exactly one bit whereas the column indices are equivalent with $j' = j$. We denote the changed bit as $i_l = 1$ and $i'_l = 0$, which allows us to simplify (22) to

$$\begin{aligned}\hat{c}(t) &= \sum_{r=0}^{n-2} [(j_{r+1} \oplus i_r \oplus k_r)\, j_r \oplus (j_{r+1} \oplus i'_r \oplus k_r)\, j_r] \\ &= (j_{l+1} \oplus 1 \oplus k_l)\, j_l \oplus (j_{l+1} \oplus 0 \oplus k_l)\, j_l \\ &= j_l \end{aligned} \tag{17}$$

As the radix-2 representation of j is a subset of the radix-2 representation of t as per (14), we conclude that

$$\hat{c}(t) = t_{l+n} \tag{18}$$

C. Case 3: $l = n-1$

Beginning again with (22), we note that this case is similar to the previous one since $i'_r = i_r = 0$ and $j'_r = j_r$ for all $0 \leq r < n-1$. The same logic also reveals that $i_{n-1} = 1$ and $i'_{n-1} = 0$, allowing us to conclude

$$\begin{aligned}\hat{c}(t) &= (i_{n-1} \oplus k_{n-1})j_{n-1} \oplus (i'_{n-1} \oplus k_{n-1})j_{n-1} \\ &= (1 \oplus k_{n-1})j_{n-1} \oplus (0 \oplus k_{n-1})j_{n-1} \\ &= j_{n-1} \\ &= t_{l+n}. \end{aligned} \tag{19}$$

Since the simplified expression is unchanged from the previous case, we can combine them to yield a single expression $\hat{c}(t) = t_{l+n}$ for all $0 \leq l \leq n-1$.

D. Case 4: $n \leq l \leq 2n-2$

We have now, by a process of elimination, established that the least significant '1' bit in t must be in the upper half of its radix-2 representation. This means that the row indices are equivalent, with $i' = i = 0$, and the column indices differ, with $j'_{n-1} = j_{n-1}$. We therefore simply (22) to

$$\begin{aligned}\hat{c}(t) &= \sum_{r=0}^{n-2} \left[(j_{r+1} \oplus k_r)\, j_r \oplus \left(j'_{r+1} \oplus k_r\right) j'_r\right] \\ &= (j_{l-n+1} \oplus k_{l-n})\, j_{l-n} \oplus \left(j'_{l-n+1} \oplus k_{l-n}\right) j'_{l-n} \\ &= j_{l-n+1} \oplus k_{l-n} \\ &= t_{l+1} \oplus k_{l-n} \end{aligned} \tag{20}$$

E. Case 5: $l = 2n-1$

In this final case, $i' = i = 0$ and $j'_r = j_r = 0$ for $r < n-1$. With the least significant '1' bit forced to be the MSB, we can conclude that $j_{n-1} = 1$ and $j'_{n-1} = 0$. Eq. (22) is therefore simplified to

$$\hat{c}(t) = k_{n-1} \tag{21}$$

$$\hat{c}(t) = \sum_{r=0}^{n-2} \left[(j_{r+1} \oplus i_r \oplus k_r)\, j_r \oplus \left(j'_{r+1} \oplus i'_r \oplus k_r\right) j'_r\right] \oplus (i_{n-1} \oplus k_{n-1})\, j_{n-1} \oplus \left(i'_{n-1} \oplus k_{n-1}\right) j'_{n-1} \tag{22}$$