Generalized Bent and/or Negabent Constructions

Yngve Ådlandsvik

April 2012

Abstract

In this thesis, we generalize the Maiorana-McFarland construction for bent, negabent and bent-negabent Boolean functions and describe a way to computationally search for constructions using these generalizations. We present some of these constructions and their properties.

Acknowledgments

First, I would like to thank my supervisor, Matthew G. Parker, for his great help in guiding me with writing this thesis and for him always being supportive, even in the face of impending deadlines. I would also like to thank the Selmer Center and the Department of Informatics at the University of Bergen for giving me this opportunity.

Finally, I would like to thank my family for their continuing support and encouraging words.

Contents

1 Introduction

Cryptographic algorithms and error correction codes are today vital parts of the digital world. As computers advance in speed and the cost of computing decreases, the need for stronger encryption algorithms increases. At the same time, the added computing power makes it possible to use better, but more processor-intensive error correction. At the core of both these research fields are Boolean functions – functions that take several digital bits as input and produce a single bit as output. Amongst these functions we find the *bent* and *negabent* Boolean functions – highly nonlinear functions that allow us to build cryptographic and error correction algorithms. As the demand for more advanced algorithms increases, the demand for more advanced Boolean functions likewise increases. In this thesis we will expand on one of the classical constructions for bent functions, the Maiorana-McFarland construction, allowing us to create bent and negabent functions of arbitrary size.

We start in chapter 2 with the definition of Boolean, bent and negabent functions, continue to introduce some facts about them, and create the theoretical framework that allows us to generate these functions. In chapter 3, we describe two different software programs we developed for function generation and discuss their limitations. In chapter 4 we present some discoveries we've made and a list of interesting functions we've discovered. Chapter 5 sums up our findings and presents possible further work on these functions.

2 Boolean, Bent and Negabent functions

2.1 Definitions

- $GL(2,n)$ – the general linear group, which is the set of $n \times n$ matrices with entries from $\mathbb{Z}_2$ which are invertible.
- $O(2,n)$ – the orthogonal group, which is the set of $n \times n$ matrices with entries from $\mathbb{Z}_2$, which are invertible and orthogonal, where orthogonal means that the dot product of any two different rows or any two different columns in the matrix is 0, and the dot product of a row or a column with itself is 1. Note that $O(2,n) \subset GL(2,n)$

2.1.1 The Kronecker Product

The Kronecker Product is an operation between two matrices where the second matrix overlays each of the values in the first matrix. The operation is represented with the symbol $\otimes$, and its precise definition is

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{2n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{bmatrix}$$

As an example, this is the Kronecker product between two matrices:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 5 & 6 & 7 \\ 8 & 9 & 0 \end{bmatrix}, \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} 1\cdot5 & 1\cdot6 & 1\cdot7 & 2\cdot5 & 2\cdot6 & 2\cdot7 \\ 1\cdot8 & 1\cdot9 & 1\cdot0 & 2\cdot8 & 2\cdot9 & 2\cdot0 \\ 3\cdot5 & 3\cdot6 & 3\cdot7 & 4\cdot5 & 4\cdot6 & 4\cdot7 \\ 3\cdot8 & 3\cdot9 & 3\cdot0 & 4\cdot8 & 4\cdot9 & 4\cdot0 \end{bmatrix}$$

In this thesis, we often apply the Kronecker product repeatedly over the same matrix, that is $\mathbf{A} \otimes \mathbf{A} \otimes \mathbf{A} ... \otimes \mathbf{A} = \mathbf{A}^{\otimes n}$. For example, $\mathbf{A}^{\otimes 1} = \mathbf{A}$, $\mathbf{A}^{\otimes 2} = \mathbf{A} \otimes \mathbf{A}$, $\mathbf{A}^{\otimes 3} = \mathbf{A} \otimes \mathbf{A} \otimes \mathbf{A}$ and so on. This means the Kronecker Matrix $\mathbf{A}^{\otimes n}$ is a matrix of size $2^n \times 2^n$.

2.1.2 Subsets of functions in $V_n \to V_n$

Let V_n be the vector space consisting of vectors, $\mathbf{x} = (x_0, x_1, x_2 ... x_{n-1})$, of length n with elements in $\mathbb{Z}_2$. We then have a set of functions $f(\mathbf{x}) = \mathbf{y}$ of the form $V_n \to V_n$, which can be seen as a transform of the n-length input vector. This set of functions can further be subdivided into several subsets:

- First, there is the full set of transformations, $\mathcal{F}_n$, which contains all functions in $V_n \to V_n$.
- Then there is the set of permutations, $\mathcal{P}_n$, which is the set of invertible functions in $V_n \to V_n$: For a function $f(\mathbf{x}) \in \mathcal{P}_n$, there always exists a function $f'(f(\mathbf{x})) = \mathbf{x}$.
- The set of invertible affine functions, $\mathcal{L}_n$ is the set of functions that can be represented by $\mathbf{y} = f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$, where $\mathbf{A}$ is a matrix in the general linear group $GL(2, n)$ and $\mathbf{b}$ is a vector in V_n. An example of this function is $f(x_0, x_1, x_2) = (x_0 + x_2, x_1 + 1, x_2)$, where $\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$.
- Finally, there is the set of identity functions, $\mathcal{I}_n$ which has as its only member the transformation represented by the identity matrix I_n , where $\mathbf{y} = f(\mathbf{x}) = \mathbf{I}_n\mathbf{x} = \mathbf{x}$

The important thing to note is that these function sets form a hierarchy, where $\mathcal{I}_n \subseteq \mathcal{L}_n \subseteq \mathcal{P}_n \subseteq \mathcal{F}_n$, and a function can be classified with regards to the smallest set it's a member of. For example, for $f(x_0, x_1, x_2) = (x_0, x_1, x_2)$, $f \in \mathcal{I}_3, f \in \mathcal{L}_3, f \in \mathcal{P}_3, f \in \mathcal{F}_3$. Likewise, $f(x_0, x_1, x_2) = (x_0, x_0, x_0)$ is only in $\mathcal{F}_3$. We also note that if $n \geq 3$, then $\mathcal{I}_n \subset \mathcal{L}_n \subset \mathcal{P}_n \subset \mathcal{F}_n$, so a function with $n \geq 3$ can be uniquely classified.

2.2 Boolean functions

Boolean functions are functions of the form $V_n \to V_1$, meaning that they take n binary variables as input and output a single binary variable as the result. These functions are an essential part of cryptography, coding theory and digital logic. Even the basic building blocks of computing, the AND, OR and NOT logic gates, are in fact Boolean functions.

The most basic way to represent a Boolean function is the *truth table*, which lists all possible values of V_n together with the corresponding V_1. As an example, here is the function $f(x_0, x_1, x_2) = x_0 + x_1 + x_2$:

x_0	x_1	x_2	$f(x_0, x_1, x_2)$
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	0
0	0	1	1
1	0	1	0
0	1	1	0
1	1	1	1

It is clear that a Boolean function of n variables will have a truth table of size 2^n, which means that as the number of variables grows, the size of the truth table grows exponentially. So as n grows larger, the truth table becomes too large to be used directly.

Another way of representing a Boolean function is the *Algebraic Normal Form*, where $ANF(f, \mathbf{x}) = 1$ if and only if either the elements that are 1 in $\mathbf{x}$ corresponds to a term in f, or $ANF(f, \mathbf{0}) = 1$ if 1 is a term of f. This is an example of the function $f(\mathbf{x}) = 1 + x_0 + x_0x_2 + x_1x_2 + x_0x_1x_2$:

Term	x_0	x_1	x_2	$ANF(f, \mathbf{x})$
1	0	0	0	1
x_0	1	0	0	1
x_1	0	1	0	0
x_0x_1	1	1	0	0
x_2	0	0	1	0
x_0x_2	1	0	1	1
x_1x_2	0	1	1	1
$x_0x_1x_2$	1	1	1	1

One can easily transform the ANF table to the truth table and vice versa – to convert from the ANF, we first need to use the Kronecker product to generate a transformation matrix. We do this by first calculating $\mathbf{A}^{\otimes 1} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, $\mathbf{A}^{\otimes 2} = \mathbf{A} \otimes \mathbf{A}$, $\mathbf{A}^{\otimes 3} = \mathbf{A} \otimes \mathbf{A} \otimes \mathbf{A}$ and so on. Then, we define the

truth table vector $f = (f(\mathbf{x}), \mathbf{x} \in V_n)$, and we have the ANF vector $\hat{f} = \mathbf{A}^{\otimes n} f$. Since $\mathbf{A}^{\otimes n}$ is its own inverse, we also have that $f = \mathbf{A}^{\otimes n} \hat{f}$. As an example, we convert the function with truth table $f^T = [1, 1, 0, 1]$:

$$\mathbf{A}^{\otimes 2} f = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \hat{f}$$

which means that $f(\mathbf{x}) = 1 + x_1 + x_0 x_1$.

2.3 Graphs

Boolean functions can also be viewed as graphs. The simplest case is when all terms in the function have degree 2 – one can then create a simple graph where each vertex represent one input variable, and an edge between two vertices represents a quadratic term with those variables, as shown in the figure below.

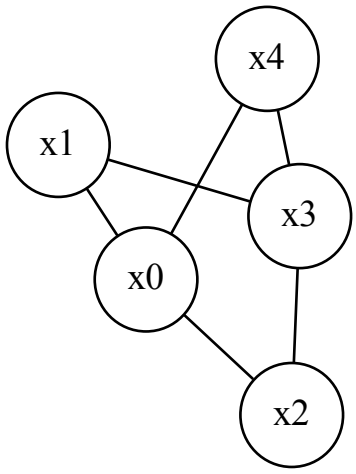

Figure 2.1: Graph representation of $f = x_0x_1 + x_0x_2 + x_0x_4 + x_1x_3 + x_2x_3 + x_3x_4$

It's also possible to model functions with a higher degree, by using hyper-

graphs (also called designs). A hypergraph can also be represented by a normal bipartite graph, where one set of vertices represent the vertices in the hypergraph and the other set represents the edges in the hypergraph, as shown in the example below:

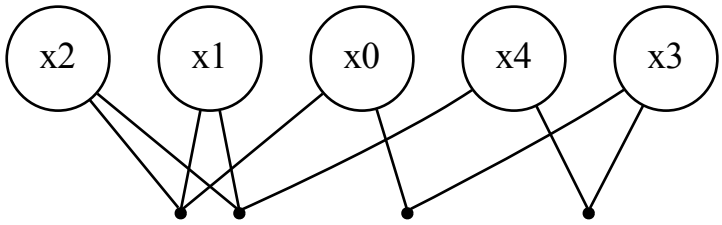

Figure 2.2: Graph representation of $f = x_0x_1x_2 + x_0x_3 + x_1x_2x_4 + x_3x_4$

These graph and hypergraph representations will later be used in the generation of all Boolean functions of a certain number of variables and certain degrees, with the help of the *nauty* software [11]. The graphs and hypergraphs are then turned back into function descriptions, which is used as input to our software.

2.4 Bent functions

Given a Boolean function, one can compute the Walsh spectrum of the function via the Walsh-Hadamard transform which is essentially a Fourier transform in $\mathbb{Z}_2$, which gives us the linear spectrum of our function.

$$H(f)(\mathbf{u}) = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in V_n} (-1)^{f(\mathbf{x})+(\mathbf{u}\cdot\mathbf{x})}$$

The spectrum, evaluated over all values of $\mathbf{u} \in V_n$ indicates how "close" the function is to the set of linear functions.

Another way of computing the spectrum is to create a *Hadamard matrix* – a $2^n \times 2^n$ matrix where all entries are either -1 or 1, and any two rows are orthogonal, that is, the dot product of them is 0. This can be done by taking

the base matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

and computing the Kronecker product in the usual way. The entries of this matrix are exactly the values of $(-1)^{\mathbf{u}\cdot\mathbf{x}}$, where $\mathbf{u}$ gives the row and $\mathbf{x}$ gives the column of the matrix. We can also define the matrix by

$$\mathbf{H}^{\otimes 0} = [1]\,,\ \mathbf{H}^{\otimes n} = \begin{bmatrix} \mathbf{H}^{\otimes(n-1)} & \mathbf{H}^{\otimes(n-1)} \\ \mathbf{H}^{\otimes(n-1)} & -\mathbf{H}^{\otimes(n-1)} \end{bmatrix}$$

This follows from the fact that if $(-1)^{(u_1,u_2,\ldots,u_{n-1})\cdot(x_1,x_2,\ldots,x_{n-1})} = a$, then $(-1)^{(u_1,u_2,\ldots,u_{n-1},u_n)\cdot(x_1,x_2,\ldots,x_{n-1},x_n)} = a$ if $u_n \cdot x_n \neq 1$ and $a(-1)$ when $u_n \cdot x_n = 1$, which only happens in the lower right part of the matrix. Finally, we sum up

$$\frac{1}{\sqrt{2^n}}\mathbf{H}^{\otimes n}(-1)^{f(\mathbf{x})} = (H(f)(\mathbf{u}), \mathbf{u} \in V_n) = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in V_n} (-1)^{f(\mathbf{x})+(\mathbf{u}\cdot\mathbf{x})}$$

If the resulting spectrum is flat, with the result being -1 or 1 for any u, the function is *bent*, which means it is as far away from all linear functions as possible. These functions were first described by O. S. Rothaus[1] in 1976, and were later found to be very useful in cryptography and coding theory.

In cryptographically secure block ciphers, functions of the form $V_m \to V_n$, called *S-boxes*, are often used to diffuse and mix the bits so it's hard to find a relation between the input and output. These functions must fulfill several criteria to be strong, one of them being that they have to be *balanced*, which means the output takes on the values of 0 and 1 equally often over the function space of V_m. Another criteria is that they have to be highly *nonlinear*. The latter criteria is exactly the definition of bent functions, so in theory they should be ideal for use in encryption. However, bent functions are never balanced, which can easily be seen by doing the Walsh-Hadamard transform and examining the $\mathbf{u} = \mathbf{0}$ component: In a balanced function, $(-1)^{f(\mathbf{x})}$ takes on the values 1 and -1 equally often. But then the sum of all values should be 0, which clearly disagrees with the result we would get if f was bent. Therefore, bent functions can't be used directly, but are instead used as a stepping stone to create proper S-boxes. For more detail on this, see [2].

Bent functions are also useful in coding theory – it was shown by Olsen

et al in [3] that sequences based on bent functions have good cross-correlation and autocorrelation properties, making them very suitable for use in coding for a shared channel. A further survey on bent function with regards to their application in coding theory can be found in [4].

It is easy to see that the Walsh-Hadamard transform of a function can only be bent when n is even, because otherwise the $\frac{1}{\sqrt{2^n}}$ constant would force the result to be an irrational number, which of course can never become 1 or -1. Rothaus proved in [1] that bent functions have a maximum degree of $\frac{n}{2}$, and it's easy to see that a bent function also needs to have a minimum degree of 2, since any functions of a lower degree are affine. Since the Hadamard matrix $\mathbf{H}_n$ is its own inverse, the resulting spectrum of a bent function is also a bent function. This is called the function's *dual*. There also exist bent functions where the Hadamard transform results in the same values in the spectrum as the input, and we call these functions *self-dual* bent functions [5]. Given a bent function, it is possible, using affine symmetry, to generate other bent functions g with the same number of variables, where $g(\mathbf{x}) = f(\mathbf{A}\mathbf{x} + \mathbf{b}) + \mathbf{c} \cdot \mathbf{x} + d$, and where $\mathbf{A}$ is a matrix in the general linear group $GL(2, n)$, $\mathbf{b}$ and $\mathbf{c}$ are vectors in V_n and d is a constant.

Whilst finding bent functions for a small number of variables is relatively easy, and can be done by exhaustive search, it is of interest to create bent functions with an arbitrary number of variables. One such construction is the Maiorana-McFarland (from now on referenced as MM) construction

$$f(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \theta(\mathbf{y}) + g(\mathbf{y})$$

where θ is a permutation $V_n \to V_n$ and g is any function $V_n \to V_1$. Any function constructed this way will be bent [6]. We also have the *extended* Maiorana-McFarland class, which contains all affine transformations of the concatenation of the input variables $\mathbf{x}$ and $\mathbf{y}$.

In this thesis we will consider generalizations of the MM construction.

2.5 Negabent functions

Another transform is the Nega-Hadamard transform, which can be computed as

$$N(f)(\mathbf{u}) = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in V_n} (-1)^{f(\mathbf{x}) + (\mathbf{u} \cdot \mathbf{x})} i^{weight(\mathbf{x})}$$

where $weight(\mathbf{x})$ is the Hamming weight of the vector $\mathbf{x}$, that is, the number of positions in $\mathbf{x}$ that have the value 1.

If the spectrum created with $N(f)(\mathbf{u})$ is flat, that is $|N(f)(\mathbf{u})| = 1$ for all $\mathbf{u} \in V_n$, then the function is called *negabent*. These functions were first described by M.G. Parker [7] in 2000. As with bent functions, we can alternatively compute the spectrum with the use of the matrix

$$\mathbf{N} = \begin{bmatrix} 1 & i \\ 1 & -i \end{bmatrix}$$

which we then use to get the Kronecker product $\mathbf{N}^{\otimes n}$. This time, the entries of the matrix are the values of $(-1)^{\mathbf{u}\cdot\mathbf{x}} i^{weight(\mathbf{x})}$, and as with the Hadamard matrix, an alternative representation is

$$\mathbf{N}^{\otimes 0} = [1]\,,\ \mathbf{N}^{\otimes n} = \begin{bmatrix} \mathbf{N}^{\otimes(n-1)} & \mathbf{N}^{\otimes(n-1)} i \\ \mathbf{N}^{\otimes(n-1)} & \mathbf{N}^{\otimes(n-1)}(-i) \end{bmatrix}$$

and then, as for bent functions, we have

$$\frac{1}{\sqrt{2^n}} \mathbf{N}^{\otimes n} (-1)^{f(\mathbf{x})} = (N(f)(\mathbf{u}), \mathbf{u} \in V_n) = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in V_n} (-1)^{f(\mathbf{x}) + (\mathbf{u}\cdot\mathbf{x})} i^{weight(\mathbf{x})}$$

While not as extensive as for bent functions, there has been some recent study of negabent functions too, with most of the focus concentrated on functions that are both bent and negabent, called bent-negabent functions. In [8], it is shown that these bent-negabent functions exist, and several techniques for constructing such functions are presented. It is shown that all affine functions are negabent, in contrast with bent functions, where none of the affine functions are bent (and are in fact as far away from bent as possible), and the *clique property* is described: If f is a bent function in n variables, and $c(x_1, ..., x_n) = \sum_{i<j} x_i x_j$, then $f + c$ is negabent. Likewise, if n is even and f is negabent, then $f + c$ is bent. For a quadratic function, this process is the same as representing the Boolean function as a graph and generating the complement graph. It's also proven that a bent-negabent's dual is also bent-negabent. In [9], bent-negabent functions are further studied, and a construction based on the previously mentioned Maiorana-McFarland class is detailed. Most importantly, it is shown that for $g(\mathbf{x}) = f(\mathbf{A}\mathbf{x} + \mathbf{b}) + \mathbf{c} \cdot \mathbf{x} + d$, where $\mathbf{A}$ is a matrix in the orthogonal group $O(2, n)$, $\mathbf{b}$ and $\mathbf{c}$ is in V_n and $d \in \{0, 1\}$, if f is a negabent

Algorithm 1 Fast Kronecker transform

Input n: number of 2x2 matrices in $\mathbf{T}$
Input $\mathbf{x}$: vector of length 2^n
Input $\mathbf{T}$: list of 2x2 matrices giving the Kronecker product
Output $\mathbf{y}$: $\mathbf{y} = \mathbf{T}\mathbf{x}$

for $i = 0$ to $n - 1$ **do**
  for $j = 0$ to $2^n - 1$ **do**
    if $j >> i \pmod 2 = 0$ **then**
      $\mathbf{y}_j = \mathbf{T}_{i,(0,0)} \cdot \mathbf{x}_j + \mathbf{T}_{i,(0,1)} \cdot \mathbf{x}_{j+2^i \pmod{2^n}}$
    else
      $\mathbf{y}_j = \mathbf{T}_{i,(1,0)} \cdot \mathbf{x}_{j-2^i+2^n \pmod{2^n}} + \mathbf{T}_{i,(1,1)} \cdot \mathbf{x}_j$
    end if
  end for
  $\mathbf{x} \leftarrow \mathbf{y}$
end for
return $\mathbf{y}$

function, then g is a negabent function. This is the same property that we had for bent functions, only with $O(2, n)$ instead of $GL(2, n)$. A further thorough exploration of negabent functions is undertaken in [10]. Amongst other things, it is proven that the maximum degree of a negabent function is $\lceil \frac{n}{2} \rceil$, which gives us an upper bound when searching for negabent functions. It also shows how to generate bent-negabent functions from the Maiorana-McFarland construction by requiring $weight(\mathbf{x} + \mathbf{y}) = weight(\theta(\mathbf{x}) + \theta(\mathbf{y}))$ and requiring $g(\mathbf{y})$ to be a bent function.

2.6 Fast Kronecker product computation

The naive way of multiplying with a Kronecker product $\mathbf{A}^{\otimes n}$ is by first creating the matrix, then doing the matrix multiplication. This, however, gives a computational complexity of $O(2^{2n})$. A better way is the Fast Kronecker transform, which is described in Algorithm 1. This algorithm gives a computational complexity of $O(2^n \log 2^n)$, and can be used to significantly speed up ANF/truth table transforms and checking functions for bentness and/or negabentness.

2.7 Going further

The Maiorana-McFarland construction hints at a property of bent and/or negabent functions – by replacing each of the variables in a function with a vector

of size m, one can use these functions to generate infinite constructions for bent and/or negabent functions. For example, using the bent function $f(\mathbf{x}) = x_0 \cdot x_1$ as a template function:

$$\begin{aligned} \mathbf{x}_0 &= (y_0, y_1) \\ \mathbf{x}_1 &= (y_2, y_3) \\ f'(\mathbf{x}) &= \mathbf{x}_0 \cdot \mathbf{x}_1 = (y_0, y_1) \cdot (y_2, y_3) = y_0 y_2 + y_1 y_3 \end{aligned}$$

While the straight insertion of vectors in place of the variables doesn't give very interesting results, it's possible to generalize by apply subfunctions to the input vectors. Continuing the previous example,

$$f(\mathbf{x}) = \theta_{01}(\mathbf{x}_0) \cdot \theta_{10}(\mathbf{x}_1) + g_0(\mathbf{x}_0) + g_1(\mathbf{x}_1)$$

with the θ_{ij} being functions from $V_m \rightarrow V_m$, and the g_i being functions from $V_m \rightarrow V_1$. There are, however, limitations on the kind of subfunctions we can use in the various positions whilst still maintaining the bent/negabent property. In the above case, one finds, to within symmetry, that θ_{01}needs to be the identity function in $\mathcal{I}_m$, while θ_{10} can be a permutation in $\mathcal{P}_m$ and g_1 can be any function in $V_m \rightarrow V_1$. This result is the Maiorana-McFarland construction.

2.8 Decomposing Kronecker products

Given a Kronecker product $\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}....\mathbf{Z}$ where the matrices are all $n \times n$ matrices of the same size, then

Lemma 1. $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$

Lemma 2. $\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C} \otimes \mathbf{D}...\mathbf{Z} = (\mathbf{A} \otimes \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{I}...\mathbf{I})(\mathbf{I} \otimes \mathbf{B} \otimes \mathbf{I} \otimes \mathbf{I}...\mathbf{I})(\mathbf{I} \otimes \mathbf{I} \otimes \mathbf{C} \otimes \mathbf{I}...\mathbf{I})...(\mathbf{I}...\mathbf{I} \otimes \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{Z})$

Proof. Let $\mathbf{A}_n$ and $\mathbf{B}_n$ be matrices of size $n \times n$, and let $\mathbf{I}_n$ be an identity matrix of size $n \times n$. Because of the mixed-product property we have

$$(\mathbf{A}_n \otimes \mathbf{I}_m)(\mathbf{I}_n \otimes \mathbf{B}_m) = \mathbf{A}_n\mathbf{I}_n \otimes \mathbf{I}_m\mathbf{B}_m = \mathbf{A}_n \otimes \mathbf{B}_m$$

We then get that$(\mathbf{A}\otimes\mathbf{I}\otimes\mathbf{I}\otimes\mathbf{I}...\mathbf{I})(\mathbf{I}\otimes\mathbf{B}\otimes\mathbf{I}\otimes\mathbf{I}...\mathbf{I}) = (\mathbf{A}\otimes\mathbf{B}\otimes\mathbf{I}\otimes\mathbf{I}...\mathbf{I})$, then $(\mathbf{A}\otimes\mathbf{B}\otimes\mathbf{I}\otimes\mathbf{I}...\mathbf{I})(\mathbf{I}\otimes\mathbf{I}\otimes\mathbf{C}\otimes\mathbf{I}...\mathbf{I}) = (\mathbf{A}\otimes\mathbf{B}\otimes\mathbf{C}\otimes\mathbf{I}...\mathbf{I})$ and so on. □

This decomposition means that instead of doing the Hadamard/Nega-Hadamard transform on all the variables of a Boolean function at once, we can split the set of variables into subsets, and do partial transforms over a subset of the variables at a time:

$$\mathbf{H}(-1)^{f(x_0,x_1)} = \mathbf{H}_{x_1}\mathbf{H}_{x_0}(-1)^{f(x_0,x_1)}$$

where $\mathbf{H}_{x_n}$ is a partial Hadamard matrix of the form $\mathbf{I}\otimes\mathbf{I}...\mathbf{I}\otimes\mathbf{H}\otimes\mathbf{I}...\mathbf{I}\otimes\mathbf{I}$ with $\mathbf{H}$ in the positions corresponding to x_n, for example if we have the function $f(x_0,x_1,x_2,x_3)$, we have $\mathbf{H}_{x_1} = \mathbf{I}\otimes\mathbf{H}\otimes\mathbf{I}\otimes\mathbf{I}$, or if we have $f(\mathbf{x}_0,\mathbf{x}_1)$ where $\mathbf{x}_0$ and $\mathbf{x}_1$ are each length 3 vectors of variables, then we have $\mathbf{H}_{\mathbf{x}_0} = \mathbf{H}\otimes\mathbf{H}\otimes\mathbf{H}\otimes\mathbf{I}\otimes\mathbf{I}\otimes\mathbf{I}$.

2.9 Partial Hadamard and Nega-Hadamard function transforms

Let $\mathbf{x},\mathbf{y}$ be vectors of variables of length m, and $\mathbf{z}$ be a vector of variables of length m'. A subclass of Boolean functions, $f(\mathbf{x},\mathbf{y},\mathbf{z})$, can be rewritten to be of the form $f(\mathbf{x},\mathbf{y}) = \mathbf{x}\cdot A(\mathbf{y}) + B(\mathbf{y},\mathbf{z})$ where $\mathbf{y}$ are the variables "connected" to $\mathbf{x}$, and $\mathbf{z}$ are the variables "unconnected" to $\mathbf{x}$. For example, for $m=2$, $f = x_0x_2 + x_1x_2 + x_1x_3 + x_2x_3 + x_4x_5$ and $\mathbf{x} = (x_0,x_1)$ becomes $f = (x_0,x_1)\cdot(x_2,x_2+x_3) + (x_2x_3 + x_4x_5)$ with $\mathbf{y} = (x_2,x_3)$ and $\mathbf{z} = (x_4,x_5)$. Using this property, we are able to deduce four function transforms that work on a single part of a function at a time. They are listed here together for convenience:

$$\begin{aligned}
\mathbf{H}_{\mathbf{x}}(-1)^{\mathbf{x}\cdot A(\mathbf{y})}\alpha^{B(\mathbf{y},\mathbf{z})} &= \prod_j\left((-1)^{x_j+A(\mathbf{y})_j}+1\right)\alpha^{B(\mathbf{y},\mathbf{z})}\\
\mathbf{H}_{\mathbf{x}}\prod_j\left((-1)^{\mathbf{x}_j+A(\mathbf{y})_j}+1\right)\alpha^{B(\mathbf{y},\mathbf{z})} &= (-1)^{\mathbf{x}\cdot A(\mathbf{y})}\alpha^{B(\mathbf{y},\mathbf{z})}\\
\mathbf{N}_{\mathbf{x}}(-1)^{\mathbf{x}\cdot A(\mathbf{y})}\alpha^{B(\mathbf{y},\mathbf{z})} &= i^{3([\mathbf{x}+A(\mathbf{y})]\cdot\mathbf{1})}\alpha^{B(\mathbf{y},\mathbf{z})}\\
\mathbf{N}_{\mathbf{x}}(-1)^{\mathbf{x}\cdot A(\mathbf{y})}\alpha^{B(\mathbf{y},\mathbf{z})} &= \mathbf{H}_{\mathbf{x}}(-1)^{\mathbf{x}\cdot A(\mathbf{y})}i^{\mathbf{x}.\mathbf{1}}\alpha^{B(\mathbf{y},\mathbf{z})}\\
\mathbf{N}_{\mathbf{x}}\prod_j\left((-1)^{\mathbf{x}_j+A(\mathbf{y})_j}+1\right)\alpha^{B(\mathbf{y},\mathbf{z})} &= (-1)^{\mathbf{x}\cdot A(\mathbf{y})}i^{A(\mathbf{y})\cdot\mathbf{1}}\alpha^{B(\mathbf{y},\mathbf{z})}
\end{aligned}$$

In these transforms, the $(i)^a$ and $(-1)^a$ parts are called the *phase* parts, as their absolute value stays the same no matter the value of a, and the $\prod_j$ parts

are called the *magnitude* parts, as the result is always a nonzero integer.

For the theorems below, note that the equations are interpreted as vectors, that is, $\mathbf{H_x} f(\mathbf{x}) = g(\mathbf{x})$ means that for the left hand side, $\mathbf{H_x}$ acts on the length 2^n vector $\mathbf{f} = (f(\mathbf{x}), \mathbf{x} \in V_n)$, which then gives the vector resulting from the multiplication $\mathbf{H_x f}$. Likewise, on the right hand side, we have the length 2^n vector $\mathbf{g} = (g(\mathbf{x}), \mathbf{x} \in V_n)$. All the theorems are true to within a constant factor.

2.9.1 Hadamard pure phase to magnitude-phase transform

Theorem 3. Let $A(\mathbf{y}) = (A(\mathbf{y})_0, A(\mathbf{y})_1, ..., A(\mathbf{y})_{m-1})$ be a function from $V_m \to V_m$, and $B(\mathbf{y}, \mathbf{z})$ be a function from $V_{m+m'} \to \mathbb{Z}_r$ for some integer r.

$$\mathbf{H_x}(-1)^{\mathbf{x}\cdot A(\mathbf{y})}\alpha^{B(\mathbf{y},\mathbf{z})} = \prod_j \left((-1)^{x_j + A(\mathbf{y})_j} + 1\right)\alpha^{B(\mathbf{y},\mathbf{z})}$$

Proof. $A(\mathbf{y})$ and $B(\mathbf{y}, \mathbf{z})$ do not change as $\mathbf{x}$ varies, therefore $A(\mathbf{y})$ can be written as the vector $\mathbf{a}$, and $\alpha^{B(\mathbf{y},\mathbf{z})}$ can be considered a constant. We only the Hadamard transform over only the $\mathbf{x}$ part, and we keep it as $\mathbf{H_x}$ to avoid confusion.

$$\mathbf{H_x}(-1)^{\mathbf{x}\cdot\mathbf{a}} = H_{\mathbf{x}}(\mathbf{u}) = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in V_n} (-1)^{\mathbf{x}\cdot\mathbf{a}+\mathbf{x}\cdot\mathbf{u}} = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in V_n} (-1)^{\mathbf{x}\cdot(\mathbf{a}+\mathbf{u})}$$

Again, for convenience, we remove the constant $\frac{1}{\sqrt{2^n}}$, and expand the dot product

$$H_{\mathbf{x}}(\mathbf{u}) = \sum_{\mathbf{x}\in V_n} (-1)^{x_1(a_1+u_1)+x_2(a_2+u_2)+...+x_n(a_n+u_n)}$$

We note that $\mathbf{x}$ takes on all possible values of V_n, and we can expand the summation. As an example, this is the expansion for V_2:

$$\begin{aligned} H_{\mathbf{x}}(\mathbf{u}) = \big((-1)^{0(a_1+u_1)+0(a_2+u_2)} + (-1)^{1(a_1+u_1)+0(a_2+u_2)} \\ +(-1)^{0(a_1+u_1)+1(a_2+u_2)} + (-1)^{1(a_1+u_1)+1(a_2+u_2)}\big) \end{aligned}$$

$$H_{\mathbf{x}}(\mathbf{u}) = \big((-1)^0 + (-1)^{a_1+u_1} + (-1)^{a_2+u_2} + (-1)^{a_1+u_1+a_2+u_2}\big)$$

We note that this kind of expansion can be contracted back to a product.

$$H_{\mathbf{x}}(\mathbf{u}) = \prod_j \left((-1)^{a_j+u_j} + 1\right)$$

Finally, we rename $\mathbf{u}$ to $\mathbf{x}$, and we have proven the transform

$$\mathbf{H}_{\mathbf{x}}(-1)^{\mathbf{x}\cdot\mathbf{a}} = \prod_j \left((-1)^{a_j+x_j} + 1\right)$$

□

2.9.2 Hadamard magnitude-phase to pure phase transform

Theorem 4.

$$\mathbf{H}_{\mathbf{x}} \prod_j \left((-1)^{\mathbf{x}_j+A(\mathbf{y})_j} + 1\right) \alpha^{B(\mathbf{y},\mathbf{z})} = (-1)^{\mathbf{x}\cdot A(\mathbf{y})}\alpha^{B(\mathbf{y},\mathbf{z})}$$

Proof. $\mathbf{H}_{\mathbf{x}}$ is its own inverse, so

$$\begin{aligned}
\mathbf{H}_{\mathbf{x}}(-1)^{\mathbf{x}\cdot A(\mathbf{y})}\alpha^{B(\mathbf{y},\mathbf{z})} &= \prod_j \left((-1)^{\mathbf{x}_j+A(\mathbf{y})_j} + 1\right) \alpha^{B(\mathbf{y},\mathbf{z})} \\
\mathbf{H}_{\mathbf{x}}\mathbf{H}_{\mathbf{x}}(-1)^{\mathbf{x}\cdot A(\mathbf{y})}\alpha^{B(\mathbf{y},\mathbf{z})} &= \mathbf{H}_{\mathbf{x}} \prod_j \left((-1)^{\mathbf{x}_j+A(\mathbf{y})_j} + 1\right) \alpha^{B(\mathbf{y},\mathbf{z})} \\
\mathbf{I}(-1)^{\mathbf{x}\cdot A(\mathbf{y})}\alpha^{B(\mathbf{y},\mathbf{z})} &= \mathbf{H}_{\mathbf{x}} \prod_j \left((-1)^{\mathbf{x}_j+A(\mathbf{y})_j} + 1\right) \alpha^{B(\mathbf{y},\mathbf{z})}
\end{aligned}$$

□

2.9.3 Nega-Hadamard pure phase to pure phase transform

In this transformation, the notation $[a_0 + a_1] \pmod 4$ means that $a_0 + a_1$ is "embedded" mod 2 in $\mathbb{Z}_4$, so that $[1 + 1] = 0$. Generalizing, we have

$$\left[\sum_{i=1}^{n} a_i\right] = \sum_{i=1}^{n} a_i + \sum_{i,j=1,i<j}^{n} 2a_i a_j \pmod 4$$

For example, for $a, b, c \in V_n$, $[a+b+c] = a+b+c+2ab+2ac+2bc \pmod 4$.

Theorem 5.

$$\mathbf{N}_{\mathbf{x}}(-1)^{\mathbf{x}\cdot A(\mathbf{y})}\alpha^{B(\mathbf{y},\mathbf{z})} = i^{3([\mathbf{x}+A(\mathbf{y})]\cdot\mathbf{1})}\alpha^{B(\mathbf{y},\mathbf{z})}$$

Proof. As before, we let $A(\mathbf{y})$ be a vector $\mathbf{a}$ and $\alpha^{B(\mathbf{y},\mathbf{z})}$ be a constant.

$$\mathbf{N_x}(-1)^{\mathbf{x}\cdot\mathbf{a}} = N_{\mathbf{x}}(\mathbf{u}) = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in V_n} (-1)^{\mathbf{x}\cdot\mathbf{a}+\mathbf{x}\cdot\mathbf{u}} i^{weight(\mathbf{x})} = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in V_n} i^{2(\mathbf{x}\cdot(\mathbf{a}+\mathbf{u}))+\mathbf{x}.\mathbf{1}}$$

Note that $weight(\mathbf{x}) = \mathbf{x}\cdot\mathbf{1}$. Again we remove $\frac{1}{\sqrt{2^n}}$ as it is constant. We expand the vectors:

$$N_{\mathbf{x}}(\mathbf{u}) = \sum_{\mathbf{x}\in V_n} i^{2x_1(a_1+u_1)+x_1+2x_2(a_2+u_2)+x_2\ldots+2x_n(a_n+u_n)+x_n}$$

Again we expand the summation and contract it back into a product, like we did with the Hadamard phase-magnitude/phase transform

$$N_{\mathbf{x}}(\mathbf{u}) = \prod_j \left(i^{2(a_j+u_j)+1} + 1\right)$$

This means that for each j, if a_j is equal to u_j, the product is multiplied by $(1+i)$, and if they are different, the product is multiplied by $(1-i)$. First, let us assume that $\mathbf{a} = \mathbf{u}$, and we can call the resulting product α, which is a constant depending on n. Then, for each position in $\mathbf{a}$ and $\mathbf{u}$ where they are different, the product α is multiplied by $-i$, or, $3i$, giving us $\alpha i^{3([\mathbf{a}+\mathbf{u}]\cdot\mathbf{1})}$. When the constant α is removed and $\mathbf{u}$ is renamed to $\mathbf{x}$ we get

$$N_{\mathbf{x}}(\mathbf{u}) = i^{3([\mathbf{a}+\mathbf{x}]\cdot\mathbf{1})}$$

where [] means embedded mod 2 as previously described. □

2.9.4 Nega-Hadamard pure phase to magnitude-phase transform

Theorem 6.

$$\mathbf{N_x}(-1)^{\mathbf{x}\cdot A(\mathbf{y})}\alpha^{B(\mathbf{y},\mathbf{z})} = \mathbf{H_x}(-1)^{\mathbf{x}\cdot A(\mathbf{y})} i^{\mathbf{x}.\mathbf{1}} \alpha^{B(\mathbf{y},\mathbf{z})}$$

Proof. First, remove the constant $\alpha^{B(\mathbf{y},\mathbf{z})}$ and let $i^f = i^{2(\mathbf{x}\cdot A(\mathbf{y}))}$. We then have

$$N_{\mathbf{x}}(\mathbf{u}) = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in V_n} i^f (-1)^{\mathbf{x}\cdot\mathbf{u}} i^{weight(\mathbf{x})} = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in V_n} i^{f+weight(\mathbf{x})} (-1)^{\mathbf{x}\cdot\mathbf{u}}$$

Let $g = f + weight(\mathbf{x}) = f + \mathbf{x} \cdot \mathbf{1}$ and we get

$$\frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in V_n} i^g (-1)^{\mathbf{x} \cdot \mathbf{u}} = H_{\mathbf{x}}(\mathbf{u})$$

We now insert g back in the original function, and can then apply the normal Hadamard pure phase to magnitude-transform □

2.9.5 Nega-Hadamard magnitude-phase to pure phase transform

Theorem 7.

$$\mathbf{N}_{\mathbf{x}} \prod_j \left((-1)^{\mathbf{x}_j + A(\mathbf{y})_j} + 1 \right) \alpha^{B(\mathbf{y},\mathbf{z})} = (-1)^{\mathbf{x} \cdot A(\mathbf{y})} i^{A(\mathbf{y}) \cdot \mathbf{1}} \alpha^{B(\mathbf{y},\mathbf{z})}$$

Proof. Here, let $\mathbf{N}_{\mathbf{x}}$ be the Nega-Hadamard matrix with the same size as $\mathbf{x}$. As before, we remove $\alpha^{B(\mathbf{y},\mathbf{z})}$ and consider $A(\mathbf{y})$ a constant $\mathbf{a}$.

$$\mathbf{N}_{\mathbf{x}} \prod_j \left((-1)^{\mathbf{x}_j + \mathbf{a}_j} + 1 \right) = (-1)^{\mathbf{x} \cdot \mathbf{a}} i^{\mathbf{a} \cdot \mathbf{1}}$$

It's easy to see that $\prod_j \left((-1)^{\mathbf{x}_j + \mathbf{a}_j} + 1 \right)$ will only be nonzero when $\mathbf{x} = \mathbf{a}$. This means that over the range of $\mathbf{x} \in V_n$, there will only be one nonzero value in the resulting vector. Now consider what happens when we multiply $\mathbf{N}_{\mathbf{x}}$ with this vector – the resulting vector will be one column of $\mathbf{N}_{\mathbf{x}}$ (within a constant factor), determined by the value of $\mathbf{a}$. We know from earlier that the entries of the matrix are $(-1)^{\mathbf{u} \cdot \mathbf{x}} i^{weight(\mathbf{x})} = (-1)^{\mathbf{u} \cdot \mathbf{x}} i^{\mathbf{x} \cdot \mathbf{1}}$ with $\mathbf{x}$ indexing the column and $\mathbf{u}$ indexing the rows. Since $\mathbf{x} = \mathbf{a}$, we have the vector $((-1)^{\mathbf{u} \cdot \mathbf{a}} i^{\mathbf{a} \cdot \mathbf{1}}, \mathbf{u} \in V_n)$. It only remains to rename $\mathbf{u}$ to $\mathbf{x}$ for convenience, and it's proven. □

2.10 Application of partial transforms

We now have all the tools required to find which subfunctions we can place where so as to get an infinite construction for bent and/or negabent functions. We will call these limits the *constraints* of a template function. To find these constraints for a specific bent/negabent function we successively apply the previously discussed transformations to a bent and/or negabent template function until the function has been evaluated over all input variables. The resulting constrained template function can then generate bent and/or negabent functions of mn variables, where n is the number of variables in the template function,

and m is the length of the vectors used in the construction. We will present two examples, the first is the $n = 2$ function that underlies the Maiorana-McFarland construction, and the second example is the $n = 4$ path graph, which is both bent and negabent. The latter construction was originally found in [9], but here we present a general way of discovering the construction.

2.10.1 Function over 2 variables

Our first example is the simple $2m$ variable function, where $F = (-1)^f$ and $\mathbf{x}_0, \mathbf{x}_1$ are vectors in V_m:

$$F = (-1)^{\theta_{01}(\mathbf{x}_0)\theta_{10}(\mathbf{x}_1)+g_0(\mathbf{x}_0)+g_1(\mathbf{x}_1)} \tag{2.1}$$

We wish to apply the Walsh-Hadamard transform to F, and wish to ensure that f is bent. To begin, we chose to apply $\mathbf{H}_{\mathbf{x}_0}$. First, we rewrite the function to have two parts, one containing all terms "connected" to $\mathbf{x}_0$, and one containing those not connected. To do this, we have to restrict some of the functions, namely $\theta_{01}$and g_0, to the identity function and zero, respectively. For simplicity in this description, $g_0(\mathbf{x}_0)$ is completely removed from the function, but observe that if f is bent and/or negabent, then $f + \mathbf{v} \cdot \mathbf{x}_0, \forall \mathbf{v} \in V_m$ is also bent and/or negabent. Given these restrictions, we have

$$F = (-1)^{\mathbf{x}_0(\theta_{10}(\mathbf{x}_1))}(-1)^{g_1(\mathbf{x}_1)}$$

Applying the $\mathbf{H}_{\mathbf{x}_0}$ transform gives, from theorem 3,

$$\mathbf{H}_{\mathbf{x}_0}F = \prod_j ((-1)^{(\mathbf{x}_0+\theta_{10}(\mathbf{x}_1))_j} + 1)(-1)^{g_1(\mathbf{x}_1)}$$

We can here see that the resulting function is only nonzero when $(\mathbf{x}_0 + \theta_{10}(\mathbf{x}_1))_j$ is zero mod 2 for all j, which also means $\mathbf{x}_0 = \theta_{10}(\mathbf{x}_1) \bmod 2$. It is possible to apply an arbitrary function $P(\mathbf{a})$ to both sides as long as $P(\mathbf{a}) = P(\mathbf{b})$ when $\mathbf{a} = \mathbf{b}$ and $P(\mathbf{a}) \neq P(\mathbf{b})$ when $\mathbf{a} \neq \mathbf{b}$. The functions with this property are the permutation functions in $V_m \to V_m$. We then get $P(\mathbf{x}_0) = P(\theta_{10}(\mathbf{x}_1))$. By picking P to be the inverse of θ_{10}, we get $\theta_{10}^{-1}(\mathbf{x}_0) = \theta_{10}^{-1}(\theta_{10}(\mathbf{x}_1)) = \mathbf{x}_1$. For this to be possible, we need to constrain θ_{10} to be a permutation too. Inserting

back and rearranging, we get

$$\mathbf{H}_{\mathbf{x}_0} F = \prod_j ((-1)^{(\mathbf{x}_1 + \theta_{10}^{-1}(\mathbf{x}_0))_j} + 1)(-1)^{g_1(\mathbf{x}_1)}$$

Because $\mathbf{x}_1 = \theta_{10}^{-1}(\mathbf{x}_0)$, we can replace $\mathbf{x}_1$in the phase part

$$\mathbf{H}_{\mathbf{x}_0} F = \prod_j ((-1)^{(\mathbf{x}_1 + \theta_{10}^{-1}(\mathbf{x}_0))_j} + 1)(-1)^{g_1(\theta_{10}^{-1}(\mathbf{x}_0))}$$

We are now ready to apply $\mathbf{H}_{\mathbf{x}_1}$, which we can do as $\mathbf{x}_1$ only occurs in the magnitude part of the function. By theorem 4,

$$\mathbf{H}_{\mathbf{x}_1} \mathbf{H}_{\mathbf{x}_0} F = (-1)^{\mathbf{x}_1 \theta_{10}^{-1}(\mathbf{x}_0) + g_1(\theta_{10}^{-1}(\mathbf{x}_0))} \tag{2.2}$$

Since we are left with a phase-only function, this means the transformed function's spectrum is flat, and, as $\mathbf{H}_{\mathbf{x}_0}\mathbf{H}_{\mathbf{x}_1}$is the Walsh-Hadamard transform, we've found a valid bent construction from (2.1), where the constraints are as follows:

- g_1 is a function in $V_m \to V_1$
- θ_{10} is a permutation in $\mathcal{P}_m$
- θ_{01} is an identity function, and g_0 is zero

which is easily recognized as the MM construction. The Hadamard dual of (2.1) is also given by (2.2).

2.10.2 Function over 4 variables

Consider this $4m$ variable function, which has the four variable bent-negabent function $x_0x_1 + x_1x_3 + x_3x_2$, corresponding to the four vertex path graph, as its template:

$$F = (-1)^{\theta_{01}(\mathbf{x}_0)\theta_{10}(\mathbf{x}_1) + \theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3) + \theta_{32}(\mathbf{x}_3)\theta_{23}(\mathbf{x}_2) + g_0(\mathbf{x}_0) + g_1(\mathbf{x}_1) + g_2(\mathbf{x}_2) + g_3(\mathbf{x}_3)} \tag{2.3}$$

We wish to apply the Walsh-Hadamard function to F, and restrict so that f is bent. We chose to apply $\mathbf{H}_{\mathbf{x}_0}$ first. As in the previous example, we separate out

$\mathbf{x}_0$ and the variables connected to it. As before, g_0 is removed.

$$\begin{aligned} F &= (-1)^{\mathbf{x}_0(\theta_{10}(\mathbf{x}_1))}(-1)^{\theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3)+\theta_{32}(\mathbf{x}_3)\theta_{23}(\mathbf{x}_2)+g_1(\mathbf{x}_1)+g_2(\mathbf{x}_2)+g_3(\mathbf{x}_3)} \\ \text{Let } \alpha &= (-1)^{\theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3)+\theta_{32}(\mathbf{x}_3)\theta_{23}(\mathbf{x}_2)+g_1(\mathbf{x}_1)+g_2(\mathbf{x}_2)+g_3(\mathbf{x}_3)} \\ \text{Then } F &= (-1)^{\mathbf{x}_0(\theta_{10}(\mathbf{x}_1))}\alpha \end{aligned}$$

Applying the transformation $\mathbf{H}_{\mathbf{x}_0}$ gives, by theorem 3,

$$\mathbf{H}_{\mathbf{x}_0}F = \prod_j((-1)^{(\mathbf{x}_0+\theta_{10}(\mathbf{x}_1))_j}+1)\alpha$$

We now wish to apply $\mathbf{H}_{\mathbf{x}_1}$, so we arrange $\mathbf{x}_0+\theta_{10}(\mathbf{x}_1)$ as $\mathbf{x}_1+\theta_{10}^{-1}(\mathbf{x}_0)$. This is only possible if θ_{10} is a permutation.

$$\mathbf{H}_{\mathbf{x}_0}F = \prod_j((-1)^{(\mathbf{x}_1+\theta_{10}^{-1}(\mathbf{x}_0))_j}+1)\alpha$$

and we replace $\mathbf{x}_1$ with $\theta_{10}^{-1}(\mathbf{x}_0)$ in α to obtain

$$\begin{aligned} \mathbf{H}_{\mathbf{x}_0}F = \prod_j((-1)^{(\mathbf{x}_1+\theta_{10}^{-1}(\mathbf{x}_0))_j}+1) \\ \cdot(-1)^{\theta_{13}(\theta_{10}^{-1}(\mathbf{x}_0))\theta_{31}(\mathbf{x}_3)+\theta_{32}(\mathbf{x}_3)\theta_{23}(\mathbf{x}_2)+g_1(\theta_{10}^{-1}(\mathbf{x}_0))+g_2(\mathbf{x}_2)+g_3(\mathbf{x}_3)} \end{aligned}$$

We can now use the second transform $\mathbf{H}_{\mathbf{x}_1}$, so by theorem 4,

$$\mathbf{H}_{\mathbf{x}_1}\mathbf{H}_{\mathbf{x}_0}F = (-1)^{\mathbf{x}_1\theta_{10}^{-1}(\mathbf{x}_0)+\theta_{13}(\theta_{10}^{-1}(\mathbf{x}_0))\theta_{31}(\mathbf{x}_3)+\theta_{32}(\mathbf{x}_3)\theta_{23}(\mathbf{x}_2)+g_1(\theta_{10}^{-1}(\mathbf{x}_0))+g_2(\mathbf{x}_2)+g_3(\mathbf{x}_3)}$$

Now we wish to apply $\mathbf{H}_{\mathbf{x}_2}$. This constrains θ_{23} to the identity and g_2 to zero. So

$$\begin{aligned} \mathbf{H}_{\mathbf{x}_1}\mathbf{H}_{\mathbf{x}_0}F &= (-1)^{\mathbf{x}_2(\theta_{32}(\mathbf{x}_3))}(-1)^{\mathbf{x}_1\theta_{10}^{-1}(\mathbf{x}_0)+\theta_{13}(\theta_{10}^{-1}(\mathbf{x}_0))\theta_{31}(\mathbf{x}_3)+g_1(\theta_{10}^{-1}(\mathbf{x}_0))+g_3(\mathbf{x}_3)} \\ \text{Let } \alpha &= (-1)^{\mathbf{x}_1\theta_{10}^{-1}(\mathbf{x}_0)+\theta_{13}(\theta_{10}^{-1}(\mathbf{x}_0))\theta_{31}(\mathbf{x}_3)+g_1(\theta_{10}^{-1}(\mathbf{x}_0))+g_3(\mathbf{x}_3)} \\ \text{Then } \mathbf{H}_{\mathbf{x}_1}\mathbf{H}_{\mathbf{x}_0}F &= (-1)^{\mathbf{x}_2(\theta_{32}(\mathbf{x}_3))}\alpha \end{aligned}$$

Applying$\mathbf{H}_{\mathbf{x}_2}$, by theorem 3, gives

$$\begin{aligned}\mathbf{H}_{\mathbf{x}_2}\mathbf{H}_{\mathbf{x}_1}\mathbf{H}_{\mathbf{x}_0}F &= \prod_j((-1)^{(\mathbf{x}_2+\theta_{32}(\mathbf{x}_3))_j}+1)\alpha \\ &= \prod_j((-1)^{(\mathbf{x}_3+\theta_{32}^{-1}(\mathbf{x}_2))_j}+1)\alpha \\ &= \prod_j((-1)^{(\mathbf{x}_3+\theta_{32}^{-1}(\mathbf{x}_2))_j}+1) \\ &\quad \cdot(-1)^{\mathbf{x}_1\theta_{10}^{-1}(\mathbf{x}_0)+\theta_{13}(\theta_{10}^{-1}(\mathbf{x}_0))\theta_{31}(\theta_{32}^{-1}(\mathbf{x}_2))+g_1(\theta_{10}^{-1}(\mathbf{x}_0))+g_3(\theta_{32}^{-1}(\mathbf{x}_2))}\end{aligned}$$

Finally, we apply transform $\mathbf{H}_{\mathbf{x}_3}$, which, by theorem 4 gives

$$\mathbf{H}_{\mathbf{x}_3}\mathbf{H}_{\mathbf{x}_2}\mathbf{H}_{\mathbf{x}_1}\mathbf{H}_{\mathbf{x}_0}F = (-1)^{\mathbf{x}_3\theta_{32}^{-1}(\mathbf{x}_2)+\mathbf{x}_1\theta_{10}^{-1}(\mathbf{x}_0)+\theta_{13}(\theta_{10}^{-1}(\mathbf{x}_0))\theta_{31}(\theta_{32}^{-1}(\mathbf{x}_2))+g_1(\theta_{10}^{-1}(\mathbf{x}_0))+g_3(\theta_{32}^{-1}(\mathbf{x}_2))} \tag{2.4}$$

this being the Walsh-Hadamard transform of $F = (-1)^f$

And thus we have the final constraints on (2.3), to ensure that f is bent:

- θ_{13}, θ_{31}, g_1 and g_3 are unrestricted functions
- θ_{10} and θ_{32} are permutations
- The rest of the functions are all identity functions or zero, as appropriate

The Hadamard dual of (2.3) is given by (2.4).

Likewise, we can do this for the nega-Hadamard transform:

$$\text{Let } F = (-1)^{\theta_{01}(\mathbf{x}_0)\theta_{10}(\mathbf{x}_1)+\theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3)+\theta_{32}(\mathbf{x}_3)\theta_{23}(\mathbf{x}_2)+g_0(\mathbf{x}_0)+g_1(\mathbf{x}_1)+g_2(\mathbf{x}_2)+g_3(\mathbf{x}_3)} \tag{2.5}$$

We apply $\mathbf{N}_{\mathbf{x}_0}$, which requires θ_{01} to be the identity, and $g_0(\mathbf{x}_0)$ to be removed. Then

$$\begin{aligned}F &= (-1)^{\mathbf{x}_0(\theta_{10}(\mathbf{x}_1))}(-1)^{\theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3)+\theta_{32}(\mathbf{x}_3)\theta_{23}(\mathbf{x}_2)+g_1(\mathbf{x}_1)+g_2(\mathbf{x}_2)+g_3(\mathbf{x}_3)} \\ \text{Let } \alpha &= (-1)^{\theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3)+\theta_{32}(\mathbf{x}_3)\theta_{23}(\mathbf{x}_2)+g_1(\mathbf{x}_1)+g_2(\mathbf{x}_2)+g_3(\mathbf{x}_3)} \\ \text{Then } F &= (-1)^{\mathbf{x}_0(\theta_{10}(\mathbf{x}_1))}\alpha\end{aligned}$$

Applying $\mathbf{N}_{\mathbf{x}_0}$, we get, from theorem 5,

$$\begin{aligned}\mathbf{N}_{\mathbf{x}_0}F &= i^{3([\mathbf{x}_0+\theta_{10}(\mathbf{x}_1)].\mathbf{1})}\alpha = i^{3(\mathbf{x}_0.\mathbf{1}+\theta_{10}(\mathbf{x}_1).\mathbf{1}+2(\mathbf{x}_0\theta_{10}(\mathbf{x}_1)))}\alpha \\ &= i^{3(\mathbf{x}_0.\mathbf{1}+\theta_{10}(\mathbf{x}_1).\mathbf{1})}(-1)^{\mathbf{x}_0\theta_{10}(\mathbf{x}_1)+\theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3)+\theta_{32}(\mathbf{x}_3)\theta_{23}(\mathbf{x}_2)+g_1(\mathbf{x}_1)+g_2(\mathbf{x}_2)+g_3(\mathbf{x}_3)}\end{aligned}$$

We now choose to apply $\mathbf{N}_{\mathbf{x}_2}$, requiring θ_{23} to be constrained to the identity and g_2 to be set to zero. So,

$$\begin{aligned}
\mathbf{N}_{\mathbf{x}_0}F &= i^{3(\mathbf{x}_0.\mathbf{1}+\theta_{10}(\mathbf{x}_1).\mathbf{1})}(-1)^{\mathbf{x}_2(\theta_{32}(\mathbf{x}_3))}(-1)^{\mathbf{x}_0\theta_{10}(\mathbf{x}_1)+\theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3)+g_1(\mathbf{x}_1)+g_3(\mathbf{x}_3)} \\
\text{Let } \alpha &= i^{3(\mathbf{x}_0.\mathbf{1}+\theta_{10}(\mathbf{x}_1).\mathbf{1})}(-1)^{\mathbf{x}_0\theta_{10}(\mathbf{x}_1)+\theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3)+g_1(\mathbf{x}_1)+g_3(\mathbf{x}_3)} \\
\text{Then } \mathbf{N}_{\mathbf{x}_0}F &= (-1)^{\mathbf{x}_2(\theta_{32}(\mathbf{x}_3))}\alpha
\end{aligned}$$

Now, applying $\mathbf{N}_{\mathbf{x}_2}$, gives, from theorem 5

$$\begin{aligned}
\mathbf{N}_{\mathbf{x}_2}\mathbf{N}_{\mathbf{x}_0}F &= i^{3([\mathbf{x}_2+\theta_{32}(\mathbf{x}_3)].\mathbf{1})}\alpha = i^{3(\mathbf{x}_2.\mathbf{1}+\theta_{32}(\mathbf{x}_3).\mathbf{1}+2(\mathbf{x}_2\theta_{32}(\mathbf{x}_3)))}\alpha \\
&= i^{3(\mathbf{x}_2.\mathbf{1}+\theta_{32}(\mathbf{x}_3).\mathbf{1}+\mathbf{x}_0.\mathbf{1}+\theta_{10}(\mathbf{x}_1).\mathbf{1})}(-1)^{\mathbf{x}_2\theta_{32}(\mathbf{x}_3)+\mathbf{x}_0\theta_{10}(\mathbf{x}_1)+\theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3)+g_1(\mathbf{x}_1)+g_3(\mathbf{x}_3)}
\end{aligned}$$

We now chose to apply $\mathbf{N}_{\mathbf{x}_1}$, requiring θ_{10} and θ_{13} to be constrained to the identity and g_1 to be set to zero. Thus,

$$\begin{aligned}
\mathbf{N}_{\mathbf{x}_1}\mathbf{N}_{\mathbf{x}_2}\mathbf{N}_{\mathbf{x}_0}F &= \mathbf{H}_{\mathbf{x}_1}i^{\mathbf{x}_1.\mathbf{1}}N_{\mathbf{x}_2}N_{\mathbf{x}_0}f \\
&= \mathbf{H}_{\mathbf{x}_1}i^{\mathbf{x}_1.\mathbf{1}}i^{3(\mathbf{x}_2.\mathbf{1}+\theta_{32}(\mathbf{x}_3).\mathbf{1}+\mathbf{x}_0.\mathbf{1}+\theta_{10}(\mathbf{x}_1).\mathbf{1})} \\
&\quad \cdot(-1)^{\mathbf{x}_2\theta_{32}(\mathbf{x}_3)+\mathbf{x}_0\theta_{10}(\mathbf{x}_1)+\theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3)+g_1(\mathbf{x}_1)+g_3(\mathbf{x}_3)} \\
&= \mathbf{H}_{\mathbf{x}_1}i^{3(\mathbf{x}_2.\mathbf{1}+\theta_{32}(\mathbf{x}_3).\mathbf{1}+\mathbf{x}_0.\mathbf{1})}(-1)^{\mathbf{x}_2\theta_{32}(\mathbf{x}_3)+\mathbf{x}_0\mathbf{x}_1+\theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3)+g_1(\mathbf{x}_1)+g_3(\mathbf{x}_3)} \\
&= \mathbf{H}_{\mathbf{x}_1}(-1)^{\mathbf{x}_1(\mathbf{x}_0+\theta_{31}(\mathbf{x}_3))}i^{3(\mathbf{x}_2.\mathbf{1}+\theta_{32}(\mathbf{x}_3).\mathbf{1}+\mathbf{x}_0.\mathbf{1})}(-1)^{\mathbf{x}_2\theta_{32}(\mathbf{x}_3)+g_3(\mathbf{x}_3)} \\
\text{Let } \alpha &= i^{3(\mathbf{x}_2.\mathbf{1}+\theta_{32}(\mathbf{x}_3).\mathbf{1}+\mathbf{x}_0.\mathbf{1})}(-1)^{\mathbf{x}_2\theta_{32}(\mathbf{x}_3)+g_3(\mathbf{x}_3)} \\
\text{Then } \mathbf{N}_{\mathbf{x}_1}\mathbf{N}_{\mathbf{x}_2}\mathbf{N}_{\mathbf{x}_0}F &= \mathbf{H}_{\mathbf{x}_1}(-1)^{\mathbf{x}_1(\mathbf{x}_0+\theta_{31}(\mathbf{x}_3))}\alpha \\
&= \prod_j((-1)^{(\mathbf{x}_1+\mathbf{x}_0+\theta_{31}(\mathbf{x}_3))_j}+1)\alpha
\end{aligned}$$

To apply $\mathbf{N}_{\mathbf{x}_3}$, we constrain θ_{31} to be a permutation:

$$\mathbf{N}_{\mathbf{x}_1}\mathbf{N}_{\mathbf{x}_2}\mathbf{N}_{\mathbf{x}_0}F = \prod_j((-1)^{(\mathbf{x}_3+\theta_{31}^{-1}(\mathbf{x}_0+\mathbf{x}_1))_j}+1)\alpha$$

So,

$$\begin{aligned}
\mathbf{N}_{\mathbf{x}_3}\mathbf{N}_{\mathbf{x}_1}\mathbf{N}_{\mathbf{x}_2}\mathbf{N}_{\mathbf{x}_0}F &= (-1)^{\mathbf{x}_3\theta_{31}^{-1}(\mathbf{x}_0+\mathbf{x}_1)}i^{\theta_{31}^{-1}(\mathbf{x}_0+\mathbf{x}_1).\mathbf{1}}\alpha \\
&= i^{\theta_{31}^{-1}(\mathbf{x}_0+\mathbf{x}_1).\mathbf{1}}i^{3(\mathbf{x}_2.\mathbf{1}+\theta_{32}(\theta_{31}^{-1}(\mathbf{x}_0+\mathbf{x}_1)).\mathbf{1}+\mathbf{x}_0.\mathbf{1})} \\
&\quad \cdot(-1)^{\mathbf{x}_3\theta_{31}^{-1}(\mathbf{x}_0+\mathbf{x}_1)+\mathbf{x}_2\theta_{32}(\theta_{31}^{-1}(\mathbf{x}_0+\mathbf{x}_1))+g_3(\theta_{31}^{-1}(\mathbf{x}_0+\mathbf{x}_1))} \qquad (2.6)
\end{aligned}$$

We have applied the nega-Hadamard transform to F and obtained a phase-only function, which means that f is negabent. The final constraints in (2.5)

are:

- θ_{32} and g_3 are unrestricted functions
- θ_{31} is a permutation
- The remaining functions are constrained to the identity or zero as appropriate

As before, the nega-Hadamard dual of (2.5) is given by (2.6).

By combining the bent and negabent constraints, we can also generate the final bent-negabent construction:

$f(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \mathbf{x}_0 \cdot \mathbf{x}_1 + \mathbf{x}_1 \cdot \theta_{31}(\mathbf{x}_3) + \mathbf{x}_2 \cdot \theta_{32}(\mathbf{x}_3) + g_3(\mathbf{x}_3)$

- g_3 is an unrestricted function
- θ_{32} and θ_{31} are permutations
- The remaining functions are identity or zero

One important thing to note here is that the order we apply the transforms affects the constraints we end up with - so $\mathbf{N}_{\mathbf{x}_3}\mathbf{N}_{\mathbf{x}_1}\mathbf{N}_{\mathbf{x}_2}\mathbf{N}_{\mathbf{x}_0}$ can have quite different constraints compared to $\mathbf{N}_{\mathbf{x}_2}\mathbf{N}_{\mathbf{x}_0}\mathbf{N}_{\mathbf{x}_3}\mathbf{N}_{\mathbf{x}_1}$. It is therefore important to try different transform sequences.

2.11 Example of construction

We now show how to generate a 12-variable bent-negabent function from the $n = 4$ construction shown above. First, we have that each vector has $m = 3$ variables, so $\mathbf{x}_0 = (a_0, a_1, a_2), \mathbf{x}_1 = (b_0, b_1, b_2), \mathbf{x}_2 = (c_0, c_1, c_2), \mathbf{x}_3 = (d_0, d_1, d_2)$. We then need two permutations, and we can for example pick $\theta_{31}(d_0, d_1, d_2) = (d_1d_2 + d_0, d_1, d_0d_1 + d_1d_2 + d_2)$ and $\theta_{32}(d_0, d_1, d_2) = (d_0d_2 + d_0 + d_1 + d_2 + 1, d_0d_2 + d_1, d_0)$. We also select $g_3(d_0, d_1, d_2) = d_0d_1 + d_0d_2 + d_0$. We now place

these functions into $\mathbf{x}_0 \cdot \mathbf{x}_1 + \mathbf{x}_1 \cdot \theta_{31}(\mathbf{x}_3) + \mathbf{x}_2 \cdot \theta_{32}(\mathbf{x}_3) + g_3(\mathbf{x}_3)$ and we get

$$\begin{aligned}
& (a_0, a_1, a_2) \cdot (b_0, b_1, b_2) \\
& + (b_0, b_1, b_2) \cdot \theta_{31}(d_0, d_1, d_2) \\
& + (c_0, c_1, c_2) \cdot \theta_{32}(d_0, d_1, d_2) \\
& + g_3(d_0, d_1, d_2) \\
= \quad & (a_0, a_1, a_2) \cdot (b_0, b_1, b_2) \\
& + (b_0, b_1, b_2) \cdot (d_1 d_2 + d_0, d_1, d_0 d_1 + d_1 d_2 + d_2) \\
& + (c_0, c_1, c_2) \cdot (d_0 d_2 + d_0 + d_1 + d_2 + 1, d_0 d_2 + d_1, d_0) \\
& + d_0 d_1 + d_0 d_2 + d_0
\end{aligned}$$

Expanding out, we get

$$\begin{aligned}
& a_0 b_0 + a_1 b_1 + a_2 b_2 \\
& + b_0 d_1 d_2 + b_0 d_0 + b_1 d_1 + b_2 d_0 d_1 + b_2 d_1 d_2 + b_2 d_2 \\
& + c_0 d_0 d_2 + c_0 d_0 + c_0 d_1 + c_0 d_2 + c_0 + c_1 d_0 d_2 + c_1 d_1 + c_2 d_0 \\
& + d_0 d_1 + d_0 d_2 + d_0
\end{aligned}$$

which is a 12-variable bent-negabent function.

3 Methodology

In the previous chapter we ended with showing how one can turn a bent and/or negabent template function on n variables into a bent and/or negabent construction with certain constraints, which then can be used to generate functions with mn variables, where m is the number of variables in each input vector to the construction. This way, we have an infinite construction where we can pick m arbitrarily.

Doing the previously discussed process manually is a time consuming task, so we have developed two independent pieces of software that can automatically find these constraints when given a bent and/or negabent template function.

The first software program, hereafter referred to as program A, uses symbolic computation to implement the previously discussed transforms. Its advantage is that it's provably correct, and relatively fast, so it works on functions with a moderate number of variables – in this thesis we present the result from using the software on template functions with up to $n = 8$ variables. Its disadvantage is that it only finds constraints that are permutations or functions, but cannot find constraints that are linear. It also doesn't currently handle template functions with a degree higher than 2, and can't provably find the optimal constraints for a given template function – there might exist looser and thus better constraints for a given template function that this software can't find.

The second program, referred to as program B, uses randomly generated subfunctions to repeatedly check the constraints of a function. Its advantage is that it is able to find constraints that can be linear, and it handles functions of any degree. The disadvantages are that it's much slower than the first program, it currently can't find the assumed optimal constraints for a function beyond $n = 4$, and it doesn't guarantee that the constructions found are infinite, since the constructions are only tested with $m = 3$ and we have no proof that the constructions are valid for $m \geq 4$.

Another difference between the programs is that program A can only search for constraints on bent or negabent functions, while program B can search on bent, negabent and bent-negabent. Therefore, we have to separately combine the results from the bent and negabent searches from program A into bent-negabent constraints, by comparing all constraints found and picking the ones that are allowed in both bent and negabent functions.

3.1 Generating Boolean functions

As input for our programs, we want to have a large set of bent and/or negabent template functions. The naive way of finding these functions is to check all possible functions by iterating over all possible ANF tables. This works up to $n = 4$, but as the number of possible tables is 2^{2^n}, it becomes computationally infeasible to search by brute force. However, we know that both bent and negabent functions stay bent/negabent with the addition of a constant value. This means we can exclude all functions with a $+1$ term. This is also true for addition and/or removal of single-degree variables, reducing the search space further. Finally, bent functions only exist when n is even, and have a maximum degree of $\frac{n}{2}$, and negabent functions have a maximum degree of $\lceil \frac{n}{2} \rceil$. When we combine these facts, we get a significant reduction in search space. For example, for $n = 5$ we only need to search all functions where all terms are either quadratic or cubic and check them for negabentness. This reduces the search to $2^{\binom{5}{2}\binom{5}{3}} = 2^{10+10} = 2^{20}$ functions, which can easily be done. Likewise, for $n = 6$ we get a search space of $2^{\binom{6}{2}+\binom{6}{3}} = 2^{15+20} = 2^{35}$.

While still doable, a naive thorough search of the $n = 6$ space takes a considerable amount of time. We wanted a quicker search, and came up with this method: First, notice that as previously stated, all Boolean functions can be represented as graphs, or if required, hypergraphs. Second, we notice that if two functions are the same with function labels swapped, that is – they are isomorphic functions, this also means the graphs representing those functions are isomorphic. This allows us to use the *nauty* software[11] to remove any isomorphic functions.

We based our search on this idea: Given a set of $n = 6$ functions of degree two, and a set of degree three, both of them with all isomorphisms removed, we can create any function of degree two plus three by picking one function from the first set and another from the second set, together with a permutation between the variable labels of the first and second functions. The number of non-isomorphic quadratic functions in $n = 6$ is 156, and the number of non-isomorphic cubic functions in $n = 6$ is 2136. We generated all possible $156 \times 2136 \times 6!$ functions, and then used *nauty* to remove any isomorphisms. After pruning, we were left with 48566080 functions. Of all those, 60203 were found to be bent, and 60203 were also found to be negabent. This hints at another part of the clique property – even if a function has degree 3 or more and is a hypergraph, one is able to use the clique property on the part of the graph that

contains the quadratic terms. So, for each bent function in $n = 6$, we are able to toggle all edges corresponding to quadratic terms, and will end up with a negabent function. Likewise, we can do the same for each negabent function in $n = 6$ and end up with a bent function.

Finally, there were 84 functions that were both bent and negabent.

For $n \geq 7$, the search space again grows too large. Therefore, we've decided to only examine quadratic template functions for these larger n.

3.2 Program A – finding constraints through symbolic computation

Our main software for finding the constraints of a bent/negabent function is based on *symbolic computation* – the template function is parsed and translated into an *expression tree*, where variables and constants are the leaves of the tree, and subfunctions and operators are the internal vertices of the tree. This representation makes it easier to translate and manipulate the function, which we need to do in order to apply function transforms.

As mentioned earlier, the advantage of this program is that it's provably correct. By applying the previously proven transforms it does the (nega-)Hadamard transform on the template function, and the resulting constraints are guaranteed to be correct.

The software can operate in two different modes. In mode 1, the software will exhaustively search through all possible sequences of transformations to find the necessary constraints. In mode 2, it randomly samples a few sequences of transformations. The former is useful for functions with few variables, while the latter can be used on larger functions where a full search would be too computationally expensive.

For the exhaustive search, a recursive algorithm is applied. It's shown here as Algorithm 2. Note that the *tryTransforms()* procedure call actually returns a list of valid transformations over the variable v – there are instances where there are several equally valid transformations, and the exhaustive search checks them all. Note that even though the mode is called exhaustive, it is not guaranteed to find the most optimal constraints for a given template function. It just means the software tries all possible variable orderings, which we've mentioned earlier can affect the constraints we find. There's no proof, however, that the results are optimal.

The random sampling search, in contrast, simply selects a random ordering

Algorithm 2 searchRecursive(f, $vars$)

```
Input f: function to work on
Input vars: list of free variables

if v = ∅ then
    if f is an all phase function then
        [record constraints of f as a valid solution]
    end if
else
    for all v in vars do
        F ← tryTransforms(f, v)
        for all f′ in F do
            searchRecursive(f′, vars \ v)
        end for
    end for
end if
```

of the function's variables, which is then successively used with the *tryTransforms()* procedure call. If several functions are returned, one is chosen at random. The process is then repeated with a new random ordering of variables until we've reached some predetermined threshold of time spent or a number of orderings tested.

In both modes, the *tryTransforms()* procedure call attempts all relevant transforms of the function over the variable $\mathbf{x}$, and gathers any valid transforms found. Each transform is described briefly below.

- Hadamard phase to magnitude-phase transform (Theorem 3): Can only be applied when $\mathbf{x}$ is in the (-1)-part of the phase and nowhere else. If this is the case, the procedure sets any subfunctions containing $\mathbf{x}$ to identity or zero, then applies the transform.

- Hadamard magnitude-phase to phase transform (Theorem 4): Can only be applied when $\mathbf{x}$ is in the magnitude part of the function. The process extracts the variable by re-expressing any subfunctions containing the variable $\mathbf{x}$, setting them all to be permutations. It then substitutes any other occurrences of $\mathbf{x}$ in the function and applies the transform.

- Nega-Hadamard phase to phase transform (Theorem 5): Can only be applied when $\mathbf{x}$ is in the (-1)-part of the phase and nowhere else. If true, the procedure set the constraints of those functions to identity or zero, and applies the transform.

- Nega-Hadamard phase to magnitude-phase transform (Theorem 6): First multiplies function by $i^{\mathbf{x}\cdot\mathbf{1}}$. After that, it attempts to do a Hadamard phase to magnitude-phase transform.

- Nega-Hadamard magnitude-phase to phase transform (Theorem 7): Can only be applied when $\mathbf{x}$ is in the magnitude part of the function. Extracts the variable by re-expressing subfunctions, where some θ are constrained to permutations, substitutes for $\mathbf{x}$ elsewhere, and applies the transform.

While the idea at first seems straightforward, we encountered several situations that were hard to handle. We will discuss these situations below. (Note that the examples are purposefully made simple and the functions shown might not be bent or negabent)

Even though a function already has a magnitude part, it might be considered part of the constant if the variable we transform over doesn't exist in the magnitude part. This might lead to this situation:

$$\mathbf{H}_{x_2} \prod_j ((-1)^{(\mathbf{x}_0+\mathbf{x}_1)_j} + 1)(-1)^{\mathbf{x}_1\mathbf{x}_2+\mathbf{x}_1\mathbf{x}_3} \\ = \prod_j ((-1)^{(\mathbf{x}_0+\mathbf{x}_1)_j} + 1) \prod_k ((-1)^{(\mathbf{x}_2+\mathbf{x}_1)_k} + 1)(-1)^{\mathbf{x}_1\mathbf{x}_3}$$

But what if we want to do a magnitude/phase to phase transform over $\mathbf{x}_1$, as it exists in both phase parts? The solution is astonishingly simple. We know, due to the first magnitude part, that $\mathbf{x}_1 = \mathbf{x}_0$, so we can therefore replace $\mathbf{x}_1$ in the second magnitude part with $\mathbf{x}_0$ at the same time as we replace it in the phase part. So we end up with

$$\mathbf{H}_{\mathbf{x}_1} \prod_j ((-1)^{(\mathbf{x}_1+\mathbf{x}_0)_j} + 1) \prod_k ((-1)^{(\mathbf{x}_2+\mathbf{x}_0)_k} + 1)(-1)^{\mathbf{x}_0\mathbf{x}_3} \\ = \prod_k ((-1)^{(\mathbf{x}_2+\mathbf{x}_0)_k} + 1)(-1)^{\mathbf{x}_1\mathbf{x}_0+\mathbf{x}_0\mathbf{x}_3}$$

Of course, we could just as easily have decided to focus on the second mag-

nitude part, replacing $\mathbf{x}_1$ with $\mathbf{x}_2$, which would lead to

$$\mathbf{H}_{\mathbf{x}_1}\prod_j((-1)^{(\mathbf{x}_0+\mathbf{x}_2)_j}+1)\prod_k((-1)^{(\mathbf{x}_1+\mathbf{x}_2)_k}+1)(-1)^{\mathbf{x}_2\mathbf{x}_3}$$
$$=\prod_j((-1)^{(\mathbf{x}_0+\mathbf{x}_2)_j}+1)(-1)^{\mathbf{x}_1\mathbf{x}_2+\mathbf{x}_2\mathbf{x}_3}$$

Therefore, we decide that both transforms are valid, and return them both back to the search procedure for consideration.

Another situation occurs when we want to transform from magnitude-phase to phase over $\mathbf{x}_1$, and are confronted with this function

$$\mathbf{H}_{\mathbf{x}_0}\prod_j((-1)^{(\mathbf{x}_0+\theta_{10}(\mathbf{x}_1)+\theta_{12}(\mathbf{x}_1))_j}+1)(-1)^{\mathbf{x}_1\mathbf{x}_3}$$

We cannot simply invert one of the theta subfunctions, we need to eliminate both of them. The easiest way to do this is to force both subfunctions to be the identity. We can then sum up $\pmod 2$, and we get

$$\mathbf{H}_{\mathbf{x}_0}\prod_j((-1)^{(\mathbf{x}_0+\mathbf{x}_1+\mathbf{x}_1)_j}+1)(-1)^{\mathbf{x}_1\mathbf{x}_3}$$
$$=\mathbf{H}_{\mathbf{x}_0}\prod_j((-1)^{(\mathbf{x}_0)_j}+1)(-1)^{\mathbf{x}_1\mathbf{x}_3}$$

or, alternatively

$$\mathbf{H}_{\mathbf{x}_0}\prod_j((-1)^{(\mathbf{x}_0\mathbf{x}_0+\theta_{10}(\mathbf{x}_1)+\theta_{12}(\mathbf{x}_1)+\theta_{13}(\mathbf{x}_1))_j}+1)(-1)^{\mathbf{x}_1\mathbf{x}_3}$$
$$=\mathbf{H}_{\mathbf{x}_0}\prod_j((-1)^{(\mathbf{x}_0+\mathbf{x}_1)_j}+1)(-1)^{\mathbf{x}_1\mathbf{x}_3}$$

We know this gives a sub-optimal result, because in the latter case one might have decided to leave one of the theta functions as a permutation. But due to software considerations we have chosen to simply force all functions to identity when this situation occurs.

As mentioned, we can't use this software to directly search for bent-negabent constructions, and we have to combine the results of the bent and negabent searches. As an example, we have this template function in $n = 6$ that was used

by both the bent and negabent search

$$\mathbf{x}_0\theta_{10}(\mathbf{x}_1) + \mathbf{x}_0\mathbf{x}_2 + \theta_{12}(\mathbf{x}_1)\mathbf{x}_2 + \mathbf{x}_0\theta_{30}(\mathbf{x}_3) + \theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3) \\ + \mathbf{x}_0\mathbf{x}_4 + \mathbf{x}_2\mathbf{x}_4 + \theta_{15}(\mathbf{x}_1)\mathbf{x}_5 + g_1(\mathbf{x}_1) + g_3(\mathbf{x}_3)$$

The bent search ended up with θ_{10}, θ_{12}, θ_{13}, θ_{31}, g_1 and g_3 being full functions, and θ_{30} and θ_{15} being permutations. In contrast, the negabent search ended up with θ_{10}, θ_{13}, θ_{31}, θ_{15}, g_1 and g_3 being full functions, and θ_{12} and θ_{30} being permutations. For those subfunctions where the results are different constraints, we have to choose the most narrow of them. So we constrain θ_{12} and θ_{15} to be permutations too. With these new constraints, any function constructed with this generator is both bent and negabent, no matter the size of the subfunctions, m, and we have an infinite construction for bent-negabent functions.

3.3 Program B – Finding constraints through random sampling search

Our alternative program for finding constraints exploits the fact previously discussed for the subfunctions $V_m \to V_m$ that $\mathcal{I}_m \subset \mathcal{L}_m \subset \mathcal{P}_m \subset \mathcal{F}_m$ when $m \geq 3$.

Let us start with an example, a bent template function over $n = 2$ vector variables:

$$f(\mathbf{x}_0, \mathbf{x}_1) = \theta_{01}(\mathbf{x}_0) \cdot \theta_{10}(\mathbf{x}_1) + g_0(\mathbf{x}_0) + g_1(\mathbf{x}_1)$$

It can easily be seen that if the subfunctions θ_{01}, θ_{10}, g_0 and g_1 are all identity functions, then f is bent, due to the fact that the underlying function x_0x_1 is bent. For the next step, our software guesses that θ_{01} is in fact a linear function in $\mathcal{L}_m$. Since the identity function is part of $\mathcal{L}_m$, this guess doesn't violate what it already knows about the function. So it sets out to check this guess. It generates random subfunctions obeying the desired constraints – in this case, θ_{01} gets set to a random linear function in $\mathcal{L}_3 \backslash \mathcal{I}_3$ (note that the lesser class is removed from the choices to ensure that the function created is indeed linear) and the rest are still constrained to the identity function in $\mathcal{I}_3$. Then the function is evaluated over all inputs to get the truth table, which then enables the function to be checked for bentness. If any of the subfunction constraints the software picked were too wide, it should eventually find a function that's *not* bent. However, there's a chance that the chosen subfunctions interact in a way

that still makes the evaluated function bent, so the software repeats the process again a few times, each time with new randomly chosen functions, until the function has passed a certain number of checks. In our case, we've settled on 5 checks as a reasonable compromise. Note that even if the set of constraints are wrongly accepted here, they will still be rejected by the more thorough check later.

Assuming the software gets a bent result on all checks, it continues: The next step is to check if θ_{01} might be a permutation in $\mathcal{P}_3$ instead. Again random functions are generated, this time with θ_{01} in $\mathcal{P}_3 \backslash \mathcal{L}_3$, and the process is related. Yet again, if it finds that all tests result in bent functions, the constraints are widened – this time letting θ_{01} be a function in $\mathcal{F}_3 \backslash \mathcal{P}_3$. With the example given, the software *will* get a not-bent result with large probability with this constraint, which means it has to scale back. The software notes that θ_{01} is at most a permutation, and continue with checking the constraints of θ_{10}, g_0 and so on.

Repeating this process successively for each subfunction, the software eventually finds a result where θ_{01} is a permutation in $\mathcal{P}_m$, θ_{10} is a linear function in $\mathcal{L}_m$, g_0 is any function in $\mathcal{F}_m$ and finally, g_1 is a linear function in $\mathcal{L}_m$. The only thing remaining is to verify the bentness properly – to do this, it repeats the same random selection of subfunctions as before, but this time it is done 250 times.

The actual implementation of this algorithm is a depth-first recursive search, however, due to a late-discovered bug that lead to aggressive search tree pruning it can't do an exhaustive search, and might miss some other local maxima – for example in the above function, it might miss this set of constraints: $\theta_{01} \in \mathcal{L}_m, \theta_{10} \in \mathcal{P}_m, g_0 \in \mathcal{L}_m, g_1 \in \mathcal{F}_m$. Most often, as in this case, several local maxima are caused by the function staying the same under some variable relabeling, but in other cases there might be widely different sets of constraints. In contrast with software A, there is no ordering on the input variables of the template function – even though the software starts going down a specific branch of the search tree, if the program works correctly, the full tree will be checked no matter where it starts.

As the random sampling search works with subfunctions of size $m = 3$, this means that the resulting function has mn variables, and the bent function check works on truth tables of size 2^{mn}. This means that for larger numbers of variables, the search is infeasible. In these cases, one would use our first software program instead.

One major disadvantage to this program is that it is not provably correct. Due to the amount of tests done, we can with some confidence say the given constraints are valid for the subfunction size we use for testing, $m = 3$, and though we guess the results are valid for $m > 3$, this is not proven.

4 Results

4.1 Enumeration of bent and negabent functions

As described in the previous chapter, we've generated all template Boolean functions of certain types and tested them for bentness/negabentness. We've put the count of the various functions with n variables and containing terms of the specified degrees in this table, with all isomorphisms up to variable renaming removed. We've also listed the number of bent, negabent and bent-negabent functions for each type. The full lists of functions can be found at [12].

n	degrees	# total	# bent	# negabent	# bent-nega
3	2	4	-	2	-
4	2	11	4	4	1
5	2	34	-	13	-
5	3	34	-	3	-
5	2,3	10688	-	152	-
6	2	156	47	47	10
6	3	2136	1	5	0
6	2,3	48566080	60203	60203	84
7	2	1044	-	339	-
8	2	12346	4043	4043	1272
9	2	274668	-	98375	-
10	2	12005168	4553432	4553432	1727780

There has been other work done in enumerating bent functions – Rothaus discussed exhaustively searching $n = 6$, though he doesn't mention the exact number of functions found[1]. The exact number of bent functions for $n = 6$ is given as 42386176 in[13]. We compared this with our result by enumerating all isomorphs for each of our 60203 non-isomorphic bent functions, and we found our result agreed with Preenel's result. Both these enumerations exclude any affine terms. This number was also confirmed via various other methods in [14][15]. The number of bent functions in $n = 8$ was first described in [16] as 99270589265934370305785861242880, including affine terms. For $n \geq 10$, the exact number is not known, though a general upper bound is found in [17].

For negabent and bent-negabent functions, no enumerations have been previously done, though we know due to the clique property that when n is even, the number of bent and negabent functions are equal.

4.2 Negabent functions over less than n variables

We discovered a curious fact when enumerating the negabent functions – there existed negabent functions that contained less than n variables, but were still negabent when the nega-Hadamard transform were applied as if there were n variables the function. This fact can be seen from how the nega-Hadamard matrix is built up. Consider the negabent function $f(x_0, x_1, x_2) = x_0x_1 + x_0x_2$ – if we instead evaluate the negabentness of the function with one more variable, $f(x_0, x_1, x_2, x_3) = x_0x_1 + x_0x_2$, we will find that for each row in the upper half of the matrix, the sum of leftmost half becomes the value α, while the rightmost half sums up to $i\alpha$. Adding both, we get $(1+i)\alpha$, which extends the magnitude of α with a constant factor. Likewise, for the lower half, each row sums up to $(1-i)\alpha$. Since the magnitude of α is always the same, all the rows also have the same magnitude, and the function is shown to be negabent.

4.3 Rediscovering Maiorana-McFarland

While searching for constraints on various functions, we found a class of bent functions that looked promising. The first functions in this class, for $n = 4, 6, 8$ are shown below. The graphs can be constructed with the edge set $E = \{(i, j), i \in (0, 2, 4, 6...), j > i\}$. The graph is a semi-bipartite graph with no edges between the odd-labeled vertices, and a clique over the even-labeled vertices. The graphs have these constraints:

- All edges in the left hand side clique can have full functions on both sides
- Edges of the form $(i, i+1)$ where i is even has a permutation at the even side and identity on the other side
- All other edges have a full function on the even side and identity on the odd side

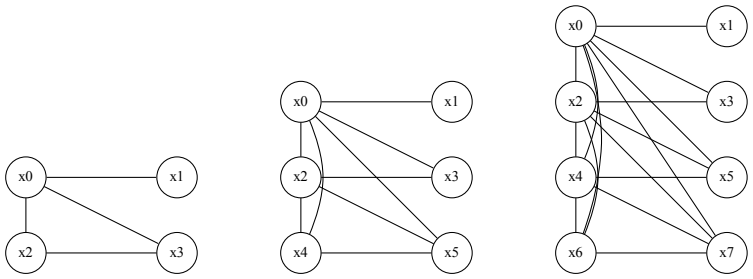

Figure 4.1: Graphs with the best constraints for bent functions

This looked like a really good class of generators at first, but we soon realized we'd just found another way of representing a subclass of Maiorana-McFarland. For example, the $n = 4$ graph

$$\mathbf{x}_1 P(\mathbf{x}_0)+\mathbf{x}_3 P(\mathbf{x}_2)+\mathbf{x}_3 F(\mathbf{x}_0)+F(\mathbf{x}_0)F(\mathbf{x}_2) = (\mathbf{x}_1, \mathbf{x}_3)(P(\mathbf{x}_0), P(\mathbf{x}_2)+F(\mathbf{x}_0))+F(\mathbf{x}_0)F(\mathbf{x}_2)$$

which is a special case of $f(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot g(\mathbf{y}) + h(\mathbf{y})$ with $\mathbf{x}$ being the concatenation of $\mathbf{x}_1$and $\mathbf{x}_3$, and $\mathbf{y}$ being the concatenation of $\mathbf{x}_0$ and $\mathbf{x}_2$.

We conjecture that any bent function which can be partitioned into a semi-bipartite graph is part of the Maiorana-McFarland class, but have yet to prove it. The converse, however, is easy to see – if one can't split the graph into a semi-bipartite graph, we can't isolate the $\mathbf{x}$ vector of the MM construction, and the function is therefore not in the *non-extended* MM class – it might still be in the extended MM class, but this is harder to verify. For the following lists we've tested this property when applicable by checking for presence of a set of $\frac{n}{2}$ vertices which only have linear or identity constraints, and where there exists no edge between any vertices in the set. The absence of such a set means the function is certainly not in the non-extended MM class.

4.4 List of interesting template functions

Here we have added the results from our programs, lists of functions with interesting properties for smaller values of n.

The function descriptions use a special, but fairly straightforward syntax. Each group of letters correspond to one term in the function, with digits meaning $0 = \mathbf{x}_0, 1 = \mathbf{x}_1$ and so on. If a letter prepends a digit, it means a function class

can be applied at that variable point. L means a linear function, P means a permutation and F is any possible function. For example, the description F0F1 P0L2 P1L3 F0 F1 corresponds to the function

$$f = \theta_{01}(\mathbf{x}_0)\theta_{10}(\mathbf{x}_1) + \theta_{02}(\mathbf{x}_0)\theta_{20}(\mathbf{x}_2) + \theta_{13}(\mathbf{x}_1)\theta_{31}(\mathbf{x}_3) + g_0(\mathbf{x}_0) + g_1(\mathbf{x}_1)$$

where $\theta_{01}, \theta_{10}, g_0$ and $g_1$can be any function, θ_{02} and θ_{13} can be any permutation, and finally θ_{20} and θ_{31} can be linear functions. In those tables where n is even, we've also included a mark on the functions that are certain to be outside the non-extended MM class. We've also marked each table with the program used to generate the table, and if applicable, the mode of the program.

For $n = 2$, there is one bent function when excluding linear and constant terms. This function is described with P01 F0, which is the Maiorana-McFarland function. There are also negabent functions for $n = 2$, but they only have linear and constant terms.

For $n = 3$, we don't have any bent functions. There is, however, one negabent function, described with 0P1 02 F1.

For $n \geq 4$ we will represent the results as tables, from both program A and program B for comparison when possible. When the title is "Program A combination", it means the result from the previous bent and negabent searches with Program A were combined to find the best bent-negabent constraints. The function's "score" is a two-numbered sequence, which displays the number of subfunctions that are full functions, and the number of subfunctions that are permutations, in that order. Note that, for $n \geq 6$, we will only list the most "interesting" functions, that is, the functions that give the best constraints, and remove less interesting functions to keep the lists at a reasonable size.

For $n = 4$ we have listed all bent, negabent and bent-negabent constraints generated with exhaustive runs of both programs, with only those functions containing only linear and constant terms removed. Note that Program B finds some of the same constraints as Program A, only with linear constraints added.

	Program A, mode 1: Bent functions, $n = 4$	
Score	Description	not MM
5, 2	F0F1 F02 P03 P12 F0 F1	
4, 2	F0F1 P02 P13 F0 F1	
3, 2	F01 P02 1P3 F0 F3	
2, 3	0P1 0P2 P13 F1 F2	
2, 1	F01 P02 12 13 F0	
1, 1	01 02 0P3 12 F3	*
1, 0	01 02 03 12 13 23 F0	*

	Program B: Bent functions, $n = 4$	
Score	Description	not MM
5, 2	F0F1 F0L2 P0L3 P1L2 F0 F1	
4, 2	F0F1 P0L2 P1L3 F0 F1	
3, 2	F0L1 P0L2 L1P3 F0 F3	
2, 2	L0P1 L2P3 F1 F3	
1, 1	01 02 L0P3 12 F3	*
1, 1	01 02 03 12 13 L2P3 F3	*

	Program A, mode 1: Negabent functions, $n = 4$
Score	Description
2, 1	F01 P02 23 F0

	Program B: Negabent functions, $n = 4$
Score	Description
2, 1	F0L1 P0L2 23 F0

	Program A combination: Bent-Negabent functions, $n = 4$
Score	Description
1, 2	P01 P02 23 F0

	Program B: Bent-Negabent functions, $n = 4$
Score	Description
1, 2	P0L1 P0L2 23 F0

For $n = 5$, we again only have negabent functions. The first table contains the result of an exhaustive search with Program A, but only on quadratic template functions. Only functions with at least one constraint which is a function or permutation are listed.

	Program A, mode 1: Negabent functions, $n = 5$
Score	Description
5, 2	F0F1 F02 P03 P12 23 24 34 F0 F1
5, 1	F0F1 F02 P03 13 14 23 24 34 F0 F1
4, 2	F0F1 P02 P13 23 24 34 F0 F1
4, 0	F0F1 02 03 12 14 23 24 34 F0 F1
3, 2	F01 P02 12 13 1P4 23 F0 F4
3, 1	F01 P02 12 13 23 24 34 F0 F4
3, 1	F01 F02 P03 34 F0
3, 1	F01 F02 P03 12 13 14 34 F0
2, 3	01 02 0P3 0P4 12 1P3 F3 F4
2, 2	01 02 0P3 12 1P4 F3 F4
2, 1	01 02 03 0P4 12 13 F2 F4
2, 0	F01 02 03 12 13 14 24 F0
2, 0	01 02 03 04 12 13 24 F3 F4
1, 1	01 02 03 1P4 F4
1, 1	01 02 03 0P4 F4
1, 0	01 02 03 14 24 34 F0
1, 0	01 02 03 12 14 34 F3
1, 0	01 02 03 04 F1
1, 0	01 02 03 04 12 13 F1
1, 0	01 02 03 04 12 13 24 F1
1, 0	01 02 03 04 12 13 24 34 F1

The second table contains the results from program B on functions with both quadratic and cubic parts in $n = 5$.

	Program B: Negabent functions, $n = 5$
Score	Description
5, 2	L0F1 02 L0P3 04 P1L2 F1F3 24 F1 F3
5, 2	F0F1 012 013 014 F0L2 P0L3 P1L2 23 24 34 F0 F1
5, 1	L0P1 02 03 04 F1F2 F1L3 24 34 F1 F2
5, 1	F0F1 012 013 014 P0L2 F0L3 12 14 23 24 34 F0 F1
4, 2	F0F1 012 013 014 P0L2 P1L3 23 24 34 F0 F1
4, 2	01 02 L0P3 12 L1P4 F3F4 F3 F4
3, 2	L0F1 012 L0P2 03 04 123 124 P1L3 34 F1 F2
3, 2	01 L0F2 03 L0P4 L1P2 13 F2 F4
3, 1	P0L1 F0L2 F0L3 14 F0
3, 1	P0L1 012 013 F0L2 024 F0L3 034 14 F0
3, 1	P0L1 012 013 F0L2 024 F0L3 034 12 14 23 24 F0
3, 1	P0L1 012 013 F0L2 024 F0L3 034 12 13 14 24 34 F0
3, 1	L0P1 012 02 03 04 123 124 F1L3 24 34 F1 F2
3, 1	L0F1 012 013 014 02 03 04 P1L2 F1L3 24 F1
3, 1	F0L1 012 013 014 P0L2 F0L3 24 F0
2, 2	P0L1 012 023 024 13 14 P2L3 34 F0 F2
2, 2	01 02 L0P3 12 L1P4 F3 F4
2, 1	P0L1 012 023 024 F0L3 14 F0
2, 1	P0L1 012 013 F0L2 024 034 14 F0
2, 1	P0L1 012 013 F0L2 024 034 13 14 23 34 F0
2, 1	P0L1 012 013 F0L2 024 034 12 13 14 24 34 F0
2, 1	L0F1 012 013 014 02 03 04 P1L2 23 F1
2, 1	F0L1 012 013 P0L2 024 034 12 13 14 23 F0
2, 1	F0L1 012 013 014 P0L2 23 F0
2, 1	01 012 L0P2 03 123 124 13 14 F2L4 F2
1, 1	P0L1 02 03 14 24 34 F0
1, 1	P0L1 012 023 024 13 F0
1, 1	P0L1 012 013 024 034 14 F0
1, 1	P0L1 012 013 024 034 14 24 34 F0
1, 1	P0L1 012 013 02 024 03 034 14 24 34 F0
1, 1	P0L1 012 013 02 024 03 034 12 23 24 F0
1, 1	P0L1 012 013 02 024 03 034 12 13 14 F0

1, 1	L0P1 02 03 04 F1
1, 1	L0P1 02 03 04 12 13 F1
1, 1	L0P1 012 013 02 124 134 23 24 F1
1, 1	L0P1 012 013 02 04 124 134 23 24 F1
1, 1	L0P1 012 013 02 03 04 124 134 F1
1, 1	L0P1 012 013 02 03 04 124 134 24 34 F1
1, 1	01 02 03 L1P4 F4
1, 1	01 012 L0P2 03 123 124 13 14 F2
1, 1	01 012 013 P0L2 024 03 034 14 24 34 F0
1, 1	01 012 013 P0L2 024 03 034 12 23 24 F0
1, 1	01 012 013 P0L2 024 03 034 12 13 14 F0
1, 1	01 012 013 02 024 P0L3 034 12 23 24 F0

For $n = 6$, we decided for Program B to only test bent and negabent template functions of degree 2, due to the number of functions of degree 3 being too large. We also tested bent-negabent functions of degree 3 with Program B. While the results from Program A are exhaustive, the results from Program B are not. Due to the number of bent and negabent constraints found by Program A, the lists have been shortened. The full tables can be found at [12].

	Program A, mode 1: Bent functions, $n = 6$	
Score	Description	not MM
12, 3	F0F1 F0F2 F03 F04 P05 F1F2 F13 P14 P23 F0 F1 F2	
11, 3	F0F1 F0F2 F03 P04 F1F2 P13 F15 P25 F0 F1 F2	
11, 3	F0F1 F0F2 F03 P04 F1F2 F13 P15 P23 F0 F1 F2	
11, 3	F0F1 F0F2 F03 F04 P05 F1F2 P13 P24 F0 F1 F2	
10, 4	F0F1 F0F2 F03 P04 F1F2 P13 P24 P25 F0 F1 F2	
10, 3	F0F1 F0F2 P03 F13 F14 P15 F23 P24 F0 F1 F2	
10, 3	F0F1 F0F2 F03 P04 F1F2 P13 P25 F0 F1 F2	
10, 3	F0F1 F0F2 F03 P04 F13 F14 P15 P23 F0 F1 F2	
10, 3	F0F1 F0F2 F03 F04 P05 F13 P14 P23 F0 F1 F2	
10, 2	F0F1 F0F2 F03 P04 F1F2 P13 14 15 24 25 F0 F1 F2	
9, 5	F0F1 F0F2 P03 P04 F1F2 P13 P15 P24 F0 F1 F2	
9, 3	F0F1 F0F2 P03 F1F2 P14 P25 F0 F1 F2	
9, 3	F0F1 F0F2 P03 F13 P14 F24 P25 F0 F1 F2	
9, 3	F0F1 F0F2 P03 F13 P14 F23 P25 F0 F1 F2	
9, 3	F0F1 F0F2 P03 F13 F14 P15 P24 F0 F1 F2	
9, 3	F0F1 F0F2 F03 P04 P13 F24 P25 F0 F1 F2	
9, 3	F0F1 F0F2 F03 P04 P13 F15 P25 F0 F1 F2	
9, 3	F0F1 F0F2 F03 P04 F13 P15 P23 F0 F1 F2	
9, 3	F0F1 F0F2 F03 F04 P05 P13 P24 F0 F1 F2	
9, 0	F0F1 F0F2 03 04 05 F1F2 13 14 23 25 F0 F1 F2	
8, 4	F0F1 F0F2 P03 P04 F13 P15 P24 F0 F1 F2	
8, 3	F0F1 F0F2 P03 F14 P15 P24 F0 F1 F2	
8, 3	F0F1 F0F2 P03 F13 P14 P25 F0 F1 F2	
8, 3	F0F1 F0F2 F03 P04 P13 P25 F0 F1 F2	
8, 3	F0F1 F02 P03 P12 2F4 3F4 P45 F0 F1 F4	
8, 3	F0F1 F02 F03 P04 P12 2F5 3P5 F0 F1 F5	
8, 3	F0F1 F02 F03 P04 F12 P13 2P5 F0 F1 F5	
8, 2	F0F1 F0F2 F03 P04 P13 14 15 24 25 F0 F1 F2	

8, 2	F0F1 F0F2 03 04 P05 13 14 P23 F25 F0 F1 F2	
8, 2	F0F1 F0F2 03 04 13 14 P15 P23 F25 F0 F1 F2	
8, 2	F0F1 F02 F03 F04 P05 F12 P13 23 24 F0 F1	
7, 3	F0F1 P02 P13 2F4 3F4 P45 F0 F1 F4	
7, 3	F0F1 F0F2 P03 P14 P25 F0 F1 F2	
7, 3	F0F1 F02 P03 P14 2P5 4F5 F0 F1 F5	
7, 3	F0F1 F02 P03 P12 F14 4P5 F0 F1 F5	
7, 3	F0F1 F02 P03 P12 3F4 P45 F0 F1 F4	
7, 3	F0F1 F02 P03 P12 2F4 P45 F0 F1 F4	
7, 3	F0F1 F02 P03 F12 P14 2P5 F0 F1 F5	
7, 3	F0F1 F02 F03 P04 P12 3P5 F0 F1 F5	
7, 2	F0F1 F02 F03 P04 P12 F15 25 35 F0 F1	
7, 2	F0F1 F02 F03 P04 F12 P15 23 25 F0 F1	
7, 2	F0F1 F02 F03 P04 F12 P13 23 25 F0 F1	
7, 2	F0F1 F02 F03 P04 F12 P13 23 24 25 F0 F1	
7, 2	F0F1 F02 F03 F04 P05 P12 23 34 F0 F1	
7, 2	F0F1 F02 F03 F04 P05 P12 23 24 34 F0 F1	*
7, 1	F0F1 F02 F03 P04 F12 14 15 23 24 25 F0 F1	
7, 1	F0F1 F02 F03 F04 P05 12 13 14 23 24 34 F0 F1	*
6, 4	F0F1 F02 P03 P13 P14 2P5 F0 F1 F5	
6, 4	F0F1 F02 P03 P04 P13 2P5 F0 F1 F5	
6, 3	F0F1 P02 P13 2F4 P45 F0 F1 F4	
6, 3	F0F1 F02 P03 P14 2P5 F0 F1 F5	
6, 3	F01 F02 P03 1F4 1P5 2P4 F0 F4 F5	
6, 2	F0F1 F02 P03 P12 F14 34 45 F0 F1	
6, 2	F0F1 F02 P03 P12 F14 24 45 F0 F1	
6, 2	F0F1 F02 P03 P12 F14 24 34 45 F0 F1	
6, 2	F0F1 F02 P03 P12 13 14 35 45 F0 F1 F5	
6, 2	F0F1 F02 P03 F14 P15 23 24 F0 F1	
6, 2	F0F1 F02 P03 F12 P14 23 25 F0 F1	
6, 2	F0F1 F02 P03 F12 P14 23 24 25 F0 F1	
6, 2	F0F1 F02 P03 13 14 2P5 35 45 F0 F1 F5	
6, 2	F0F1 F02 F03 P04 P15 23 25 F0 F1	
6, 2	F0F1 F02 F03 P04 P15 23 25 35 F0 F1	*
6, 2	F0F1 F02 F03 P04 P12 34 35 F0 F1	
6, 2	F0F1 F02 F03 P04 P12 24 25 34 35 F0 F1	*

6, 2	F0F1 F02 F03 P04 P12 23 35 F0 F1	
6, 2	F0F1 F02 F03 P04 P12 23 34 35 F0 F1	
6, 2	F0F1 F02 F03 P04 P12 23 25 35 F0 F1	*
6, 2	F0F1 F02 F03 P04 P12 23 24 25 34 35 F0 F1	*
6, 2	F0F1 F02 F03 P04 P12 14 15 34 35 F0 F1	
6, 2	F0F1 F02 F03 P04 P12 14 15 24 25 34 35 F0 F1	*
6, 2	F0F1 F02 03 04 P05 P13 F15 23 24 F0 F1	
6, 2	F0F1 02 03 P04 12 13 2P5 4F5 F0 F1 F5	
6, 2	F01 F02 F03 P04 12 13 1F5 2P5 F0 F5	
6, 1	F0F1 F02 F03 P04 14 15 23 24 25 F0 F1	
6, 1	F0F1 F02 F03 P04 14 15 23 24 25 34 35 F0 F1	*
6, 1	F0F1 F02 F03 P04 12 13 15 23 25 35 F0 F1	*
6, 1	F0F1 F02 F03 P04 12 13 14 15 23 24 25 34 35 F0 F1	*
6, 1	F0F1 F02 03 04 P13 F15 23 24 25 F0 F1	
6, 0	F0F1 F02 03 04 F12 13 15 23 24 25 F0 F1	

	Program B: Bent functions, $n = 6$	
Score	Description	not MM
12, 3	F0F1 F0F2 F0L3 F0L4 P0L5 F1F2 F1L3 P1L4 P2L3 F0 F1 F2	
11, 3	F0F1 F0F2 F0L3 P0L4 F1F2 P1L3 F1L5 P2L5 F0 F1 F2	
11, 3	F0F1 F0F2 F0L3 P0L4 F1F2 F1L3 P1L5 P2L3 F0 F1 F2	
11, 3	F0F1 F0F2 F0L3 F0L4 P0L5 F1F2 P1L3 P2L4 F0 F1 F2	
10, 3	F0F1 F0F2 P0L3 F1L3 F1L4 P1L5 F2L3 P2L4 F0 F1 F2	
10, 3	F0F1 F0F2 F0L3 P0L4 F1F2 P1L3 P2L5 F0 F1 F2	
10, 2	F0F1 F0F2 F0L3 P0L4 F1F2 P1L3 14 15 24 25 F0 F1 F2	
9, 3	F0F1 F0F2 P0L3 F1L3 P1L4 F2L4 P2L5 F0 F1 F2	
9, 3	F0F1 F0F2 P0L3 F1L3 P1L4 F2L3 P2L5 F0 F1 F2	
9, 3	F0F1 F0F2 P0L3 F1F2 P1L4 P2L5 F0 F1 F2	
9, 3	F0F1 F0F2 F0L3 P0L4 P1L3 F2L4 P2L5 F0 F1 F2	
9, 3	F0F1 F0F2 F0L3 P0L4 P1L3 F1L5 P2L5 F0 F1 F2	
9, 3	F0F1 F0F2 F0L3 F0L4 P0L5 P1L3 P2L4 F0 F1 F2	
9, 1	F0F1 F0F2 03 04 P0L5 F1F2 13 15 24 25 F0 F1 F2	
8, 3	F0F1 F0F2 P0L3 F1L4 P1L5 P2L4 F0 F1 F2	
8, 3	F0F1 F0F2 P0L3 F1L3 P1L4 P2L5 F0 F1 F2	
8, 3	F0F1 F0F2 F0L3 P0L4 P1L3 P2L5 F0 F1 F2	
8, 2	F0F1 F0F2 F0L3 P0L4 P1L3 14 15 24 25 F0 F1 F2	
8, 2	F0F1 F0F2 03 04 P0L5 13 14 P2L3 F2L5 F0 F1 F2	
8, 2	F0F1 F0F2 03 04 13 14 P1L5 P2L3 F2L5 F0 F1 F2	
7, 3	F0F1 F0L2 P0L3 P1L4 L2P5 L4F5 F0 F1 F5	
7, 3	F0F1 F0L2 P0L3 P1L2 L3F4 P4L5 F0 F1 F4	
7, 3	F0F1 F0F2 P0L3 P1L4 P2L5 F0 F1 F2	
7, 2	F0F1 F0L2 F0L3 F0L4 P0L5 12 13 P1L4 23 24 34 F0 F1	*
7, 1	F0F1 F0F2 03 04 P0L5 13 15 24 25 F0 F1 F2	
7, 1	F0F1 F0F2 03 04 13 P1L4 15 24 25 F0 F1 F2	
6, 3	F0F1 P0L2 P1L3 L2F4 P4L5 F0 F1 F4	
6, 2	F0F1 F0L2 P0L3 P1L2 13 14 35 45 F0 F1 F5	
6, 2	F0F1 F0L2 P0L3 13 14 L2P5 35 45 F0 F1 F5	
6, 2	F0F1 F0L2 F0L3 P0L4 P1L2 14 15 24 25 34 35 F0 F1	*
6, 2	F0F1 F0L2 F0L3 P0L4 12 P1L3 15 23 25 35 F0 F1	*
6, 2	F0F1 F0L2 F0L3 P0L4 12 P1L3 14 15 23 24 25 34 35 F0 F1	*
6, 2	F0F1 F0L2 F0L3 P0L4 12 13 P1L5 23 25 35 F0 F1	*
6, 2	F0F1 02 03 P0L4 12 13 L2P5 L4F5 F0 F1 F5	

5, 2	F0F1 F0L2 P0L3 P1L2 34 35 45 F0 F1	*
5, 2	F0F1 F0L2 P0L3 P1L2 14 15 34 35 45 F0 F1	*
5, 2	F0F1 F0L2 P0L3 P1L2 14 15 24 25 45 F0 F1	*
5, 2	F0F1 F0L2 P0L3 P1L2 14 15 24 25 34 35 45 F0 F1	*
5, 2	F0F1 F0L2 P0L3 P1L2 13 14 34 35 45 F0 F1	*
5, 2	F0F1 F0L2 P0L3 P1L2 13 14 23 24 35 45 F0 F1	*
5, 2	F0F1 F0L2 P0L3 P1L2 13 14 23 24 34 35 45 F0 F1	*
5, 2	F0F1 F0L2 P0L3 P1L2 13 14 15 23 24 25 34 45 F0 F1	*
5, 2	F0F1 F0L2 P0L3 13 14 P1L5 23 24 35 45 F0 F1	*
5, 2	F0F1 F0L2 P0L3 12 14 P1L5 24 25 45 F0 F1	*
5, 2	F0F1 F0L2 P0L3 12 13 14 P1L5 23 24 25 35 45 F0 F1	*
5, 2	F0F1 F0L2 03 04 P0L5 P1L2 34 35 45 F0 F1	*
5, 2	F0F1 F0L2 03 04 P0L5 P1L2 13 15 34 35 45 F0 F1	*
5, 2	F0F1 F0L2 03 04 P0L5 P1L2 13 15 23 25 34 35 45 F0 F1	*
5, 2	F0F1 F0L2 03 04 P0L5 P1L2 13 14 34 35 45 F0 F1	*
5, 2	F0F1 F0L2 03 04 P0L5 P1L2 13 14 23 24 34 35 45 F0 F1	*
5, 1	F0F1 F0L2 P0L3 13 14 24 25 34 35 45 F0 F1	*
5, 1	F0F1 F0L2 P0L3 13 14 23 25 34 35 45 F0 F1	*
5, 1	F0F1 F0L2 03 04 P0L5 13 15 24 25 34 35 45 F0 F1	*
5, 1	F0F1 F0L2 03 04 P0L5 13 15 23 24 34 35 45 F0 F1	*
5, 1	F0F1 F0L2 03 04 P0L5 13 14 23 25 34 35 45 F0 F1	*
5, 1	F0F1 02 03 P0L4 12 14 35 45 F0 F1 F5	
5, 1	F0F1 02 03 12 14 L2P5 35 45 F0 F1 F5	
4, 2	F0L1 P0L2 L1P3 23 24 35 45 F0 F3 F4	
4, 2	F0L1 F0L2 P0L3 12 14 15 24 25 L4P5 F0 F5	*
4, 2	F0F1 02 03 P0L4 P1L5 23 24 34 F0 F1	*
4, 2	F0F1 02 03 P0L4 P1L5 23 24 25 34 45 F0 F1	*
4, 2	F0F1 02 03 P0L4 12 P1L3 15 23 24 25 34 35 F0 F1	*
4, 1	F0F1 02 03 P0L4 14 15 23 24 35 45 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 15 34 35 45 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 15 23 25 34 45 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 14 34 35 45 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 14 23 24 35 45 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 14 23 24 34 35 45 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 13 24 25 45 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 13 23 24 35 45 F0 F1	*

4, 1	F0F1 02 03 P0L4 12 13 23 24 25 34 45 F0 F1	*
4, 1	F0F1 02 03 04 05 12 13 P1L4 24 25 45 F0 F1	*
3, 2	F0L1 P0L2 L1P3 24 25 34 35 45 F0 F3	*
3, 2	F0L1 P0L2 13 14 15 34 35 L4P5 F0 F5	*
3, 2	F0L1 P0L2 12 13 24 25 34 35 L4P5 F0 F5	*
3, 1	01 02 L0P3 14 25 34 35 F1 F2 F3	
2, 2	01 02 03 04 05 12 13 14 L1P5 L2P3 45 F3 F5	*
2, 1	01 02 03 L0P4 12 15 34 35 45 F2 F4	*
2, 1	01 02 03 04 12 15 P2L5 34 35 F2 F4	*
1, 1	01 02 03 L1P2 14 25 35 45 F2	*
1, 1	01 02 03 L1P2 14 24 35 45 F2	*
1, 1	01 02 03 12 P1L3 24 35 45 F1	*
1, 1	01 02 03 12 14 P2L5 35 45 F2	*
1, 1	01 02 03 04 P0L5 12 13 24 35 45 F0	*
1, 1	01 02 03 04 L0P5 12 15 25 34 35 45 F5	*
1, 1	01 02 03 04 12 15 25 34 L3P5 F5	*
1, 1	01 02 03 04 05 12 13 L2P3 45 F3	*
1, 1	01 02 03 04 05 12 13 14 P1L5 23 45 F1	*
1, 1	01 02 03 04 05 12 13 14 15 23 24 25 34 35 L4P5 F5	*

	Program A mode 1: Negabent functions, $n = 6$	
Score	Description	not MM
7, 2	F0F1 F02 F03 P04 F12 P13 34 35 45 F0 F1	*
7, 1	F0F1 F02 F03 P04 F12 14 15 34 35 45 F0 F1	*
6, 2	F0F1 F02 P03 P12 F14 23 25 35 F0 F1	*
6, 2	F0F1 F02 P03 F12 P14 34 35 45 F0 F1	*
6, 2	F0F1 F02 F03 P04 P12 24 25 45 F0 F1	*
6, 2	F0F1 F02 F03 P04 P12 23 45 F0 F1	*
6, 2	F0F1 F02 F03 P04 P12 14 15 23 45 F0 F1	*
6, 1	F0F1 F02 F03 P04 14 15 24 25 45 F0 F1	*
6, 1	F0F1 F02 03 04 P13 F15 34 35 45 F0 F1	*
6, 0	F0F1 F02 03 04 F12 13 15 34 35 45 F0 F1	*
5, 3	F0F1 F02 P03 P13 P14 34 35 45 F0 F1	*
5, 3	F0F1 F02 P03 P04 P13 34 35 45 F0 F1	*
5, 2	F0F1 F02 P03 P14 34 35 45 F0 F1	*
5, 2	F0F1 F02 P03 P14 24 35 F0 F1	*
5, 2	F0F1 F02 P03 P12 24 35 F0 F1	*
5, 2	F0F1 F02 P03 P12 13 14 25 34 F0 F1	*
5, 2	F0F1 F02 P03 13 14 P15 25 34 F0 F1	*
5, 2	F01 F02 P03 13 14 1P5 2F5 34 F0 F5	*
5, 1	F01 F02 P03 13 14 2F5 34 35 45 F0 F5	*
4, 3	F01 P02 P03 1F4 23 2P4 25 35 F0 F4	*
4, 2	F0F1 P02 P13 24 35 F0 F1	*
4, 2	F01 P02 1F3 24 25 P34 45 F0 F3	*
4, 2	F01 P02 12 13 1P4 23 F45 F0 F4	*
4, 2	F01 F02 P03 13 14 1P5 34 F0 F5	*
4, 2	F01 F02 P03 12 1P4 35 F0 F4	*
4, 2	F01 F02 P03 12 1P4 34 35 45 F0 F4	*
4, 1	F01 F02 P03 13 14 34 35 45 F0 F5	*
4, 1	F01 F02 F03 P04 45 F0	
4, 1	F01 F02 F03 P04 12 14 15 45 F0	
4, 1	F01 F02 F03 P04 12 13 45 F0	
4, 1	F01 F02 F03 P04 12 13 24 25 45 F0	*
4, 1	F01 F02 F03 P04 12 13 24 25 34 35 45 F0	*
4, 1	F01 02 03 23 24 2P5 34 4F5 F0 F5	*
4, 0	F0F1 02 03 04 12 15 23 45 F0 F1	*

4, 0	F0F1 02 03 04 12 13 24 35 F0 F1	*
4, 0	F0F1 02 03 04 12 13 15 24 35 F0 F1	*
4, 0	F01 02 03 1F4 23 24 25 35 45 F0 F4	*
3, 3	F01 P02 P03 23 24 2P5 34 F0 F5	*
3, 3	F01 P02 23 24 2P5 34 3P5 F0 F5	*
3, 2	F01 P02 23 24 34 3P5 F0 F5	*
3, 2	F01 P02 13 24 3P5 F0 F5	*
3, 2	F01 P02 13 24 25 3P4 45 F0 F4	*
3, 2	F01 P02 13 1P4 25 F0 F4	*
3, 2	F01 P02 13 1P4 24 25 45 F0 F4	*
3, 1	F01 P02 23 24 25 34 35 F0 F4	*
3, 1	F01 F02 P03 13 14 15 24 35 F0	*
3, 1	F01 F02 P03 13 14 15 23 24 34 F0	*
3, 1	F01 F02 P03 13 14 15 23 24 25 34 F0	*
3, 1	F01 F02 P03 12 13 14 34 35 45 F0	*
3, 1	F01 F02 P03 12 13 14 25 34 35 45 F0	*
3, 0	F01 F02 03 04 13 15 24 34 35 45 F0	*
3, 0	F01 F02 03 04 13 14 15 35 F0	
3, 0	F01 F02 03 04 13 14 15 23 45 F0	*
3, 0	F01 F02 03 04 13 14 15 23 25 35 F0	*
3, 0	F01 F02 03 04 13 14 15 23 24 25 35 F0	*
3, 0	F01 F02 03 04 12 13 15 24 34 35 45 F0	*
2, 3	01 0P2 1P3 P24 45 F2 F3	*
2, 3	01 0P2 0P3 P24 45 F2 F3	*
2, 2	01 02 0P3 12 P14 45 F1 F3	*
2, 1	F01 P02 23 34 35 F0	
2, 1	F01 P02 23 24 25 F0	
2, 1	F01 P02 13 23 24 25 34 45 F0	*
2, 1	F01 P02 13 14 25 35 45 F0	
2, 1	F01 P02 13 14 23 45 F0	
2, 1	F01 P02 13 14 23 25 35 F0	*
2, 1	F01 P02 13 14 23 24 25 F0	
2, 1	F01 P02 13 14 23 24 25 35 45 F0	*
2, 1	F01 P02 12 13 24 35 F0	
2, 1	F01 P02 12 13 24 34 45 F0	
2, 1	F01 P02 12 13 23 24 34 45 F0	*

2, 1	F01 P02 12 13 14 23 25 35 F0	*
2, 1	F01 P02 12 13 14 23 25 35 45 F0	*
2, 0	F01 02 03 24 25 34 45 F0	
2, 0	F01 02 03 12 24 34 35 45 F0	*
2, 0	F01 02 03 12 23 24 34 35 45 F0	*
2, 0	F01 02 03 12 14 24 25 35 45 F0	*
2, 0	F01 02 03 12 14 23 25 34 35 45 F0	*
2, 0	F01 02 03 12 14 23 24 34 35 F0	*
2, 0	F01 02 03 12 14 15 24 35 F0	*
2, 0	F01 02 03 12 14 15 23 24 34 35 F0	*
2, 0	F01 02 03 12 13 14 24 35 F0	*
2, 0	F01 02 03 12 13 14 24 25 45 F0	*
2, 0	F01 02 03 12 13 14 24 25 35 45 F0	*
2, 0	F01 02 03 04 25 35 45 F0	
2, 0	F01 02 03 04 23 24 25 F0	
2, 0	F01 02 03 04 23 24 25 35 F0	
2, 0	F01 02 03 04 23 24 25 35 45 F0	
2, 0	F01 02 03 04 12 23 34 35 45 F0	*
2, 0	F01 02 03 04 12 23 25 34 35 45 F0	*
2, 0	F01 02 03 04 12 15 23 34 35 F0	
2, 0	F01 02 03 04 12 15 23 25 34 35 F0	*
2, 0	F01 02 03 04 12 13 25 35 45 F0	
2, 0	F01 02 03 04 12 13 24 34 45 F0	
2, 0	F01 02 03 04 12 13 15 24 25 34 45 F0	*
2, 0	01 02 03 12 14 35 45 F0 F4	*
2, 0	01 02 03 12 14 34 35 F0 F4	*

	Program B: Negabent functions, $n = 6$	
Score	Description	not MM
7, 2	F0F1 F0L2 F0L3 P0L4 F1L2 P1L3 34 35 45 F0 F1	*
7, 1	F0F1 F0L2 F0L3 P0L4 F1L2 14 15 34 35 45 F0 F1	*
6, 2	F0F1 F0L2 P0L3 P1L2 F1L4 23 25 35 F0 F1	*
6, 2	F0F1 F0L2 P0L3 F1L2 P1L4 34 35 45 F0 F1	*
6, 2	F0F1 F0L2 F0L3 P0L4 P1L2 24 25 45 F0 F1	*
6, 2	F0F1 F0L2 F0L3 P0L4 P1L2 23 45 F0 F1	*
6, 2	F0F1 F0L2 F0L3 P0L4 P1L2 14 15 23 45 F0 F1	*
6, 1	F0F1 F0L2 F0L3 P0L4 14 15 24 25 45 F0 F1	*
6, 1	F0F1 F0L2 03 04 P1L3 F1L5 34 35 45 F0 F1	*
5, 2	F0L1 F0L2 P0L3 13 14 L1P5 L2F5 34 F0 F5	*
5, 2	F0F1 F0L2 P0L3 P1L4 34 35 45 F0 F1	*
5, 2	F0F1 F0L2 P0L3 P1L4 24 35 F0 F1	*
5, 2	F0F1 F0L2 P0L3 P1L2 24 35 F0 F1	*
5, 2	F0F1 F0L2 P0L3 P1L2 13 14 25 34 F0 F1	*
5, 2	F0F1 F0L2 P0L3 13 14 P1L5 25 34 F0 F1	*
5, 1	F0L1 F0L2 P0L3 13 14 L2F5 34 35 45 F0 F5	*
4, 2	F0L1 P0L2 L1F3 24 25 P3L4 45 F0 F3	*
4, 2	F0L1 P0L2 12 13 L1P4 23 F4L5 F0 F4	*
4, 2	F0L1 F0L2 P0L3 13 14 L1P5 34 F0 F5	*
4, 2	F0L1 F0L2 P0L3 12 L1P4 35 F0 F4	*
4, 2	F0L1 F0L2 P0L3 12 L1P4 34 35 45 F0 F4	*
4, 2	F0F1 P0L2 P1L3 24 35 F0 F1	*
4, 1	F0L1 F0L2 F0L3 P0L4 45 F0	
4, 1	F0L1 F0L2 F0L3 P0L4 12 13 24 25 34 35 45 F0	*
4, 1	F0L1 02 03 23 24 L2P5 34 L4F5 F0 F5	*
4, 1	F0F1 02 03 P0L4 14 15 24 35 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 15 24 35 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 14 25 34 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 13 24 35 F0 F1	*
3, 2	F0L1 P0L2 23 24 34 L3P5 F0 F5	*
3, 2	F0L1 P0L2 13 L1P4 25 F0 F4	*
3, 2	F0L1 P0L2 13 L1P4 24 25 45 F0 F4	*
3, 2	F0L1 P0L2 13 24 L3P5 F0 F5	*
3, 2	F0L1 P0L2 13 24 25 L3P4 45 F0 F4	*

3, 1	F0L1 F0L2 P0L3 13 14 15 23 24 25 34 F0	*
3, 1	F0L1 F0L2 P0L3 12 13 14 25 34 35 45 F0	*
2, 1	F0L1 P0L2 23 34 35 F0	
2, 1	F0L1 P0L2 23 24 25 F0	
2, 1	F0L1 P0L2 13 14 25 35 45 F0	
2, 1	F0L1 P0L2 13 14 23 24 25 F0	
2, 1	F0L1 P0L2 13 14 23 24 25 35 45 F0	*
2, 1	F0L1 P0L2 12 13 24 34 45 F0	
2, 1	F0L1 P0L2 12 13 14 23 25 35 45 F0	*
2, 1	F0L1 02 03 P0L4 25 35 45 F0	
2, 1	F0L1 02 03 P0L4 24 34 45 F0	
2, 1	F0L1 02 03 P0L4 23 24 25 F0	
2, 1	F0L1 02 03 P0L4 14 15 23 24 25 F0	
2, 1	F0L1 02 03 P0L4 12 15 24 34 45 F0	
2, 1	F0L1 02 03 P0L4 12 15 23 34 35 F0	
2, 1	F0L1 02 03 P0L4 12 14 25 35 45 F0	
2, 1	F0L1 02 03 P0L4 12 14 23 34 35 F0	
2, 1	F0L1 02 03 P0L4 12 13 25 35 45 F0	
2, 1	F0L1 02 03 P0L4 12 13 24 34 45 F0	
2, 1	01 02 L0P3 13 24 35 45 F2 F3	*
2, 1	01 02 03 L1P2 24 35 45 F2 F3	*
2, 1	01 02 03 L1P2 24 34 45 F2 F3	*
2, 1	01 02 03 14 P1L5 25 45 F1 F2	*
1, 1	01 02 03 L0P4 15 25 F4	
1, 1	01 02 03 L0P4 14 15 24 25 F4	
1, 1	01 02 03 L0P4 12 13 25 35 F4	*
1, 1	01 02 03 14 L1P5 24 F5	
1, 1	01 02 03 14 24 L3P5 F5	
1, 1	01 02 03 14 15 24 L2P5 35 F5	
1, 1	01 02 03 14 15 24 25 L3P4 F4	
1, 1	01 02 03 04 12 P1L3 25 45 F1	
1, 1	01 02 03 04 12 P1L3 25 35 F1	

	Program A combined:Bent-negabent functions, $n = 6$	
Score	Description	not MM
5, 3	F0F1 F02 P03 P04 P12 23 25 35 F0 F1	*
5, 2	F0F1 F02 P03 P12 13 14 34 35 45 F0 F1	*
4, 3	F0F1 P02 P03 P14 24 25 45 F0 F1	*
3, 3	F01 P02 P03 12 14 1P5 24 F0 F5	*
3, 2	F01 P02 1P3 23 24 25 34 45 F0 F3	*
2, 3	01 02 0P3 12 1P4 P35 F3 F4	*
2, 2	F01 P02 P03 12 14 35 F0	
2, 2	F01 P02 P03 12 14 23 25 34 35 45 F0	*
2, 1	F01 P02 12 13 24 34 45 F0	
2, 0	F01 02 03 04 12 15 23 34 35 F0	
1, 2	01 02 1P3 P34 45 F3	
1, 2	01 02 0P3 1P3 14 25 F3	
1, 2	01 02 03 0P4 12 13 2P4 25 35 F4	*
1, 1	01 02 13 24 3P5 F5	
1, 1	01 02 03 P04 12 13 25 45 F0	
1, 1	01 02 03 14 1P5 24 F5	
1, 1	01 02 03 12 P14 34 45 F1	
1, 0	01 02 03 04 12 13 25 45 F1	

	Program B: Bent-negabent functions, $n = 6$	
Score	Description	not MM
5, 3	F0F1 F0L2 P0L3 P0L4 P1L2 23 25 35 F0 F1	*
5, 2	F0F1 F0L2 P0L3 P1L2 13 14 34 35 45 F0 F1	*
4, 3	F0F1 P0L2 P0L3 P1L4 24 25 45 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 14 34 35 45 F0 F1	*
4, 1	F0F1 02 03 P0L4 12 13 24 25 45 F0 F1	*
4, 1	012 013 024 034 P0L1 F0F5 12 14 24 25 35 F0 F5	*
4, 1	012 013 024 034 01 P0L2 04 F0F5 12 13 15 23 45 F0 F5	*
4, 1	012 013 024 034 01 02 P0L3 F0F5 13 14 25 34 35 F0 F5	*
4, 1	012 013 024 034 01 02 P0L3 F0F5 12 14 24 25 35 F0 F5	*
4, 0	012 013 024 034 125 135 245 345 01 02 03 F0F5 12 14 15 24 F0 F5	*
3, 3	F0L1 P0L2 P0L3 12 14 L1P5 24 F0 F5	*
3, 2	F0L1 P0L2 L1P3 23 24 25 34 45 F0 F3	*
3, 2	012 013 024 034 F0L1 P0L2 F0L4 P0L5 12 13 14 23 F0	*
3, 1	012 013 024 034 F0L1 P0L2 F0L4 12 13 14 23 25 35 F0	*
2, 3	01 02 L0P3 12 L1P4 P3L5 F3 F4	*
2, 2	F0L1 P0L2 P0L3 L1L2 14 35 F0	
2, 2	F0L1 P0L2 P0L3 12 L1L4 35 F0	
2, 2	F0L1 P0L2 P0L3 12 14 23 25 34 35 45 F0	*
2, 2	012 013 024 034 P0L1 P0L2 F0L5 14 25 35 F0	*
2, 2	012 013 024 034 P0L1 P0L2 F0L5 12 13 14 24 25 34 35 F0	*
2, 2	012 013 024 034 F0L1 P0L2 P0L5 14 23 24 34 F0	*
2, 2	012 013 024 034 F0L1 P0L2 P0L5 12 13 14 23 F0	*
2, 1	F0L1 P0L2 12 13 24 34 45 F0	
2, 1	F0L1 02 03 P0L4 14 15 23 24 25 F0	
2, 1	F0L1 02 03 P0L4 12 15 24 34 45 F0	
2, 1	F0L1 02 03 P0L4 12 15 23 34 35 F0	
2, 1	012 013 024 034 P0L1 F0L5 12 15 23 24 45 F0	*
2, 1	012 013 024 034 P0L1 12 14 24 25 35 F0 F5	*
2, 1	012 013 024 034 F0L1 P0L2 14 23 24 25 34 35 F0	*
2, 1	012 013 024 034 F0L1 P0L2 12 13 14 23 25 35 F0	*
2, 1	012 013 024 034 01 P0L2 04 F0L5 12 13 14 25 35 F0	*
2, 1	012 013 024 034 01 P0L2 04 12 13 15 23 45 F0 F5	*
2, 1	012 013 024 034 01 02 P0L3 F0L5 13 15 23 34 45 F0	*
2, 1	012 013 024 034 01 02 P0L3 F0L5 12 15 23 24 45 F0	*

2, 1	012 013 024 034 01 02 P0L3 13 14 25 34 35 F0 F5	*
2, 1	012 013 024 034 01 02 P0L3 12 14 24 25 35 F0 F5	*
2, 0	012 013 024 034 125 135 245 345 01 02 03 05 12 15 24 F2 F3	*
1, 2	012 013 024 034 P0L1 P0L5 12 15 34 45 F0	
1, 2	012 013 024 034 P0L1 P0L5 12 14 23 24 F0	*
1, 2	012 013 024 034 P0L1 P0L2 14 25 35 F0	*
1, 2	012 013 024 034 P0L1 P0L2 12 13 14 24 25 34 35 F0	*
1, 2	012 013 024 034 01 P0L2 04 P0L5 12 25 34 35 F0	
1, 2	01 L0L2 L2P3 P3L4 45 F3	
1, 2	01 L0L2 L1P3 P3L4 45 F3	
1, 2	01 02 L0P3 13 L2P3 24 35 45 F3	
1, 2	01 02 03 L0P4 12 13 L2P4 25 35 F4	*
1, 1	012 013 024 034 P0L1 12 25 34 35 F0	
1, 1	012 013 024 034 P0L1 12 15 23 24 45 F0	*
1, 1	012 013 024 034 P0L1 12 14 15 23 24 45 F0	*
1, 1	012 013 024 034 125 135 245 345 01 02 L1P4 35 45 F4	*
1, 1	012 013 024 034 125 135 245 345 01 02 03 P0L5 12 23 34 45 F0	*
1, 1	012 013 024 034 125 135 245 345 01 02 03 L0P5 12 25 34 35 45 F5	*
1, 1	012 013 024 034 125 135 245 345 01 02 03 12 P1L4 34 45 F1	*
1, 1	012 013 024 034 125 135 245 345 01 02 03 05 12 P2L3 34 45 F2	*
1, 1	012 013 024 034 125 135 245 345 01 02 03 05 12 14 P2L3 24 45 F2	*
1, 1	012 013 024 034 01 P0L2 04 12 15 34 45 F0	
1, 1	012 013 024 034 01 P0L2 04 12 13 14 25 35 F0	*
1, 1	012 013 024 034 01 L0P5 12 23 34 F5	*
1, 1	012 013 024 034 01 L0P5 12 15 34 45 F5	*
1, 1	012 013 024 034 01 L0P5 12 14 15 23 34 45 F5	*
1, 1	012 013 024 034 01 02 P0L3 13 24 25 35 F0	
1, 1	012 013 024 034 01 02 P0L3 13 15 23 34 45 F0	*
1, 1	012 013 024 034 01 02 P0L3 12 25 34 35 F0	
1, 1	012 013 024 034 01 02 P0L3 12 15 23 24 45 F0	*
1, 1	012 013 024 034 01 02 P0L3 05 13 15 24 45 F0	
1, 1	012 013 024 034 01 02 P0L3 05 12 15 34 45 F0	
1, 1	012 013 024 034 01 02 L0P5 13 25 34 35 F5	*
1, 1	012 013 024 034 01 02 L0P5 13 14 24 F5	*
1, 1	012 013 024 034 01 02 L0P5 12 14 34 F5	*
1, 1	012 013 024 034 01 02 L0P5 12 13 15 45 F5	*

1, 1	012 013 024 034 01 02 03 L0P5 12 23 34 F5	*
1, 1	012 013 024 034 01 02 03 L0P5 12 14 15 23 34 45 F5	*
1, 1	01 02 L0L3 12 P1L4 L3L4 45 F1	
1, 1	01 02 03 P0L4 12 14 25 35 F0	
1, 1	01 02 03 P0L4 12 13 L2L5 45 F0	
1, 1	01 02 03 L1P2 24 34 45 F2	
1, 1	01 02 03 14 P1L5 25 45 F1	
1, 1	01 02 03 14 L1P5 24 F5	
1, 1	01 02 03 04 12 P1L3 35 45 F1	
1, 1	01 02 03 04 12 P1L3 25 45 F1	
1, 0	012 013 024 034 125 135 245 345 01 02 05 14 35 45 F2	*
1, 0	012 013 024 034 125 135 245 345 01 02 03 12 15 34 F0	*
1, 0	012 013 024 034 125 135 245 345 01 02 03 05 12 14 25 34 35 45 F2	*
1, 0	012 013 024 034 01 12 23 25 34 35 F5	*
1, 0	012 013 024 034 01 12 14 23 25 34 35 F5	*
1, 0	012 013 024 034 01 02 13 15 25 34 35 45 F5	*
1, 0	012 013 024 034 01 02 13 14 15 24 25 35 45 F5	*
1, 0	012 013 024 034 01 02 12 14 15 25 34 35 45 F5	*
1, 0	012 013 024 034 01 02 12 13 15 25 35 45 F5	*
1, 0	012 013 024 034 01 02 03 12 23 25 34 35 F5	*
1, 0	012 013 024 034 01 02 03 12 14 23 25 34 35 F5	*

We examined negabent functions for $n = 7$. These functions were created by random sampling with the symbolic computation software. Only constructions with 6 or more function constraints are listed here.

	Program A, mode 2: Negabent functions, $n = 7$
Score	Description
12, 1	F0F1 F0F2 F03 F04 P05 F1F2 F13 15 16 24 25 26 34 35 36 45 46 56 F0 F1 F2
11, 1	F0F1 F0F2 F03 F04 P05 F1F2 13 15 16 25 26 34 35 36 45 46 56 F0 F1 F2
11, 1	F0F1 F0F2 F03 F04 P05 F1F2 13 14 23 25 26 34 35 36 45 46 56 F0 F1 F2
11, 0	F0F1 F0F2 F03 04 05 F1F2 F13 14 16 24 25 26 34 35 36 45 46 56 F0 F1 F2
10, 4	F0F1 F0F2 F03 P04 F1F2 P13 P15 P23 34 35 36 45 46 56 F0 F1 F2
10, 3	F0F1 F0F2 F03 F04 P05 F13 P14 P23 34 35 36 45 46 56 F0 F1 F2
10, 2	F0F1 F0F2 F03 F04 P05 F13 P14 24 25 26 34 35 36 45 46 56 F0 F1 F2
10, 2	F0F1 F0F2 F03 04 05 F1F2 P13 P24 34 35 36 45 46 56 F0 F1 F2
10, 2	F0F1 F0F2 03 04 05 F13 P14 F16 P23 F26 34 35 36 45 46 56 F0 F1 F2
9, 1	F0F1 F0F2 F03 04 05 F13 14 16 P23 34 35 36 45 46 56 F0 F1 F2
9, 1	F0F1 F0F2 03 04 13 14 15 P23 F25 F26 34 35 36 45 46 56 F0 F1 F2
9, 1	F0F1 F02 F03 F04 P05 F12 F13 15 16 45 46 56 F0 F1
9, 0	F0F1 F0F2 03 04 05 F1F2 13 14 16 23 25 26 34 35 36 45 46 56 F0 F1 F2
8, 2	F0F1 F02 F03 P04 F12 P13 F15 34 36 46 F0 F1
8, 2	F0F1 F02 F03 P04 F12 F13 P15 45 46 56 F0 F1
8, 2	F0F1 F02 F03 P04 F12 F13 P15 23 24 25 26 45 46 56 F0 F1
8, 2	F0F1 F02 F03 F04 P05 F12 P13 35 36 56 F0 F1
8, 2	F0F1 F02 F03 F04 P05 F12 P13 34 56 F0 F1
8, 2	F0F1 F02 F03 F04 P05 F12 P13 15 16 34 56 F0 F1
8, 1	F0F1 F0F2 F03 P04 14 15 16 24 25 34 35 36 45 46 56 F0 F1 F2
8, 1	F0F1 F0F2 F03 P04 13 15 25 34 35 36 45 46 56 F0 F1 F2
8, 1	F0F1 F0F2 F03 04 05 P13 24 34 35 36 45 46 56 F0 F1 F2
8, 1	F0F1 F0F2 03 13 F14 15 P24 34 35 36 45 46 56 F0 F1 F2
8, 1	F0F1 F02 F03 F04 P05 F12 15 16 35 36 56 F0 F1
8, 1	F0F1 F02 F03 04 05 F12 P14 F16 45 46 56 F0 F1
7, 3	F0F1 P02 P13 23 2F4 25 26 3F4 35 36 P45 56 F0 F1 F4
7, 3	F0F1 F02 F03 P04 P12 23 24 25 34 35 3P6 45 F0 F1 F6
7, 2	F0F1 F02 F03 P04 P12 F15 24 26 46 F0 F1
7, 2	F0F1 F02 F03 P04 P12 F15 23 46 F0 F1
7, 2	F0F1 F02 F03 P04 F14 P15 45 46 56 F0 F1
7, 2	F0F1 F02 F03 P04 F12 P15 45 46 56 F0 F1

7, 2 F0F1 F02 F03 P04 F12 P15 35 46 F0 F1

7, 2 F0F1 F02 F03 P04 F12 P15 23 24 25 26 35 46 F0 F1

7, 2 F0F1 F02 F03 P04 F12 P13 35 46 F0 F1

7, 2 F0F1 F02 F03 F04 P05 P12 25 26 56 F0 F1

7, 2 F0F1 F02 F03 F04 P05 P12 23 56 F0 F1

7, 2 F0F1 F02 F03 F04 P05 P12 23 25 26 34 35 36 56 F0 F1

7, 2 F0F1 F02 F03 F04 P05 P12 23 24 34 35 36 56 F0 F1

7, 2 F0F1 F02 F03 F04 P05 P12 23 24 25 26 56 F0 F1

7, 2 F01 F02 F03 P04 12 13 14 1F5 16 24 2P5 26 3F5 46 F0 F5

7, 1 F0F1 F0F2 P03 13 14 24 25 34 35 36 45 46 56 F0 F1 F2

7, 1 F0F1 F0F2 03 04 13 15 P26 34 35 36 45 46 56 F0 F1 F2

7, 1 F0F1 F0F2 03 04 13 14 15 P23 34 35 36 45 46 56 F0 F1 F2

7, 1 F0F1 F02 F03 F04 P05 12 13 23 24 25 26 34 56 F0 F1

7, 1 F0F1 F02 F03 F04 P05 12 13 23 24 25 26 34 45 46 56 F0 F1

7, 1 F0F1 F02 F03 F04 P05 12 13 14 23 24 25 26 56 F0 F1

7, 0 F0F1 F0F2 03 13 14 15 24 26 34 35 36 45 46 56 F0 F1 F2

7, 0 F0F1 F0F2 03 13 14 15 23 24 26 34 35 36 45 46 56 F0 F1 F2

7, 0 F0F1 F0F2 03 04 13 15 16 23 25 34 35 36 45 46 56 F0 F1 F2

7, 0 F0F1 F0F2 03 04 13 14 15 23 25 34 35 36 45 46 56 F0 F1 F2

7, 0 F0F1 F0F2 03 04 05 13 16 24 34 35 36 45 46 56 F0 F1 F2

7, 0 F0F1 F0F2 03 04 05 13 16 24 26 34 35 36 45 46 56 F0 F1 F2

7, 0 F0F1 F0F2 03 04 05 13 14 23 25 34 35 36 45 46 56 F0 F1 F2

7, 0 F0F1 F0F2 03 04 05 13 14 16 23 25 26 34 35 36 45 46 56 F0 F1 F2

7, 0 F0F1 F02 F03 04 05 F12 14 23 34 35 36 45 46 56 F0 F1

6, 4 F0F1 F02 P03 P14 23 24 25 2P6 34 35 45 4P6 F0 F1 F6

6, 3 F0F1 F02 P03 P14 23 24 25 2P6 34 35 45 F0 F1 F6

6, 3 F01 F02 P03 12 13 1F4 15 1P6 23 2P4 25 35 F0 F4 F6

6, 2 F0F1 F02 P03 P12 F14 25 36 F0 F1

6, 2 F0F1 F02 P03 F14 P15 35 36 56 F0 F1

6, 2 F0F1 F02 P03 F14 P15 25 36 F0 F1

6, 2 F0F1 F02 P03 F12 P14 35 46 F0 F1

6, 2 F0F1 F02 P03 F12 P14 23 24 25 26 35 46 F0 F1

6, 2 F0F1 F02 F03 P04 P15 45 46 56 F0 F1

6, 2 F0F1 F02 F03 P04 P15 25 46 F0 F1

6, 2 F0F1 F02 F03 P04 P15 23 24 25 26 35 46 F0 F1

6, 2 F0F1 F02 F03 P04 P12 25 46 F0 F1

6, 2	F0F1 F02 F03 P04 P12 25 35 45 46 56 F0 F1
6, 2	F0F1 F02 F03 P04 P12 24 25 34 35 45 46 56 F0 F1
6, 2	F0F1 F02 F03 P04 P12 23 34 35 36 45 F0 F1
6, 2	F0F1 F02 F03 P04 P12 23 25 35 45 46 56 F0 F1
6, 2	F0F1 F02 F03 P04 P12 23 24 25 26 45 F0 F1
6, 2	F0F1 F02 F03 P04 P12 14 15 23 26 36 45 46 56 F0 F1
6, 2	F0F1 F02 F03 P04 14 15 P16 26 45 F0 F1
6, 2	F01 F02 P03 1F4 2F4 35 36 P45 56 F0 F4
6, 2	F01 F02 P03 13 14 1P5 2F5 34 F56 F0 F5
6, 1	F0F1 F02 P03 13 14 23 24 25 26 34 35 36 45 46 F0 F1 F5
6, 1	F0F1 F02 F03 P04 14 15 26 36 45 46 56 F0 F1
6, 1	F0F1 F02 F03 P04 14 15 16 23 24 25 26 35 45 46 56 F0 F1
6, 1	F0F1 F02 F03 P04 12 15 24 26 34 35 36 45 46 56 F0 F1
6, 1	F0F1 F02 F03 P04 12 15 23 25 35 45 46 56 F0 F1
6, 1	F0F1 F02 F03 P04 12 15 23 24 26 34 36 45 46 56 F0 F1
6, 1	F0F1 F02 F03 P04 12 14 15 16 23 25 34 35 36 45 46 56 F0 F1
6, 1	F0F1 F02 F03 P04 12 13 15 25 35 45 46 56 F0 F1
6, 1	F0F1 F02 F03 P04 12 13 14 15 23 24 25 26 45 F0 F1
6, 1	F0F1 F02 03 04 P12 23 24 25 34 35 36 45 56 F0 F1 F6
6, 1	F0F1 F02 03 04 12 13 15 23 24 25 2P6 34 35 45 F0 F1 F6
6, 1	F0F1 02 03 P14 23 24 25 26 34 35 45 4F6 56 F0 F1 F6
6, 1	F0F1 02 03 04 12 15 23 24 25 2F6 34 35 3P6 45 F0 F1 F6
6, 0	F0F1 F02 F03 04 05 14 16 45 46 56 F0 F1
6, 0	F0F1 F02 03 13 14 23 24 25 34 35 36 45 46 56 F0 F1 F6
6, 0	F0F1 F02 03 04 13 F15 16 34 36 46 F0 F1
6, 0	F0F1 F02 03 04 05 F12 13 14 35 46 F0 F1
6, 0	F0F1 F02 03 04 05 F12 13 14 23 24 25 26 35 46 F0 F1
6, 0	F0F1 F02 03 04 05 F12 13 14 16 35 46 F0 F1
6, 0	F0F1 02 03 04 12 13 23 24 25 26 34 35 45 5F6 F0 F1 F6
6, 0	F01 F02 03 04 1F5 2F5 34 35 36 46 F0 F5
6, 0	F01 F02 03 04 1F5 2F5 34 35 36 46 56 F0 F5

These functions were created by random sampling with the symbolic computation software. Since the number of bent and negabent functions starts getting large, we will only give some examples of functions, most notably those outside the MM class. We emphasize that we have not proven that these constructions are outside the extended MM class.

	Program A, mode 2: Bent functions, $n = 8$	
Score	Description	not MM
16, 3	04 05 0F6 0F7 P15 F1F6 F1F7 2P6 2F7 3P7 45 4F6 4F7 5F6 5F7 F6F7 F1 F6 F7	*
15, 3	04 05 0F6 0F7 P15 F1F6 F1F7 2P6 3P7 45 4F6 4F7 5F6 5F7 F6F7 F1 F6 F7	*
15, 3	04 05 0F6 0F7 P15 F1F6 F1F7 2F6 2P7 3P6 45 4F7 5F6 5F7 F6F7 F1 F6 F7	*
15, 3	03 04 05 F0F6 F0F7 1P6 1P7 2P7 34 35 3F6 3F7 45 4F6 4F7 5F6 5F7 F6F7 F0 F6 F7	*
15, 2	03 04 05 0F6 0F7 1F6 1P7 2P6 34 35 3F6 3F7 45 F4F6 F4F7 5F7 F6F7 F4 F6 F7	*
14, 3	F0F3 F05 P06 F0F7 14 16 1F7 2P7 P35 F3F7 46 4F7 5F7 6F7 F0 F3 F7	*
14, 3	F0F3 F05 P06 F0F7 14 15 16 1F7 2P7 P35 F3F7 46 4F7 56 5F7 6F7 F0 F3 F7	*
14, 3	0P3 0F4 05 06 07 1P4 15 16 17 25 26 27 F3F4 F3F5 F36 F37 F4F5 F46 F47 P57 F3 F4 F5	*
14, 3	04 05 0F6 0F7 P15 F1F7 2P6 2F7 3P7 45 4F6 4F7 5F6 5F7 F6F7 F1 F6 F7	*
14, 3	04 05 0F6 0F7 P15 F1F6 F1F7 2P6 3P7 45 4F6 4F7 5F7 F6F7 F1 F6 F7	*
14, 3	04 05 0F6 0F7 P15 F1F6 F1F7 2P6 2F7 3P7 45 4F6 4F7 F6F7 F1 F6 F7	*
14, 3	04 05 0F6 0F7 P15 F1F6 F1F7 2F6 2P7 3P6 45 4F7 5F7 F6F7 F1 F6 F7	*
14, 3	03 04 05 F0F6 F0F7 1P6 1P7 2P6 34 35 3F6 3F7 45 4F6 4F7 5F7 F6F7 F0 F6 F7	*
14, 3	03 04 05 06 0F7 F1F4 P15 F16 F1F7 2P7 34 35 36 3F7 P46 F4F7 5F7 6F7 F1 F4 F7	*
14, 2	P03 F0F4 F05 F06 F0F7 14 15 16 1F7 2P7 45 46 F4F7 56 5F7 6F7 F0 F4 F7	*
14, 2	P03 F04 F0F5 F06 F0F7 14 15 16 1F7 2P7 3F7 45 46 4F7 56 F5F7 F0 F5 F7	*
14, 2	03 04 05 F0F6 F0F7 1P6 1F7 2P7 34 35 3F6 3F7 45 4F6 4F7 5F6 5F7 F0 F6 F7	*
13, 4	P03 F0F5 P06 F0F7 14 16 1F7 2P7 3P5 3F7 46 4F7 F5F7 6F7 F0 F5 F7	*
13, 4	04 0P5 06 0F7 14 16 1F7 F2F5 P26 F2F7 3P7 4P5 4F7 F5F7 6F7 F2 F5 F7	*
13, 4	04 05 0F6 0F7 P15 F1F6 2P6 2P7 3P7 45 4F6 4F7 5F6 5F7 F6F7 F1 F6 F7	*
13, 3	P03 F0F6 F0F7 14 15 1P6 1F7 24 25 2F7 3F6 3F7 4P7 F6F7 F0 F6 F7	*
13, 3	0P4 05 0F6 07 F1F4 F1F6 P17 25 27 3P6 F45 F4F6 F47 F67 F1 F4 F6	*
13, 3	0P4 05 06 0F7 F1F4 P16 F1F7 25 26 2F7 3P7 F45 F46 F4F7 F1 F4 F7	*
13, 3	04 0F5 0F6 07 F1F5 F1F6 P17 2P5 3P6 4F5 4F6 47 F57 F67 F1 F5 F6	*
13, 3	04 0F5 06 0F7 F1F5 P16 F1F7 2P5 2F7 3P7 46 F56 F5F7 F1 F5 F7	*
13, 3	04 0F5 06 0F7 1P5 1F7 P26 F2F7 3P7 4F5 46 4F7 F5F7 6F7 F2 F5 F7	*
13, 3	04 06 0F7 F1F5 P16 F1F7 2P5 3P7 46 4F7 F56 F5F7 6F7 F1 F5 F7	*
13, 3	04 05 0F6 0F7 P15 F1F6 F1F7 2P6 2F7 3P7 45 4F7 F6F7 F1 F6 F7	*
13, 2	P03 F04 F0F5 F06 F0F7 14 1F5 16 17 2P5 46 47 F56 F5F7 67 F0 F5 F7	*

13, 2	P03 F04 F05 F06 F0F7 14 15 16 F1F7 2P7 45 46 4F7 56 5F7 6F7 F0 F1 F7	*
13, 2	03 05 0F6 07 P14 F1F6 F1F7 2P6 35 3F6 37 4F6 4F7 57 F6F7 F1 F6 F7	*
13, 2	03 05 06 0F7 P14 F1F6 F1F7 2P7 35 36 3F7 4F7 56 5F7 F6F7 F1 F6 F7	*
13, 2	03 05 06 0F7 P14 F15 F16 F1F7 2P7 35 36 F3F7 4F7 56 5F7 6F7 F1 F3 F7	*
13, 2	03 04 06 0F7 1F5 1P7 2P5 34 36 3F7 46 4F7 F5F6 F5F7 F6F7 F5 F6 F7	*
13, 2	03 04 05 06 0F7 1P6 1F7 2P7 34 35 F3F6 F3F7 45 46 4F7 56 5F7 F6F7 F3 F6 F7	*
13, 1	04 05 06 07 F1F4 F1F5 P17 24 26 27 35 36 37 F4F5 F46 F47 F56 F57 67 F1 F4 F5	*
13, 1	03 F0F5 06 F0F7 14 15 16 1F7 2P7 35 3F7 46 4F7 56 F5F7 6F7 F0 F5 F7	*
13, 1	03 05 06 0F7 F1F4 15 16 F1F7 2P7 35 36 3F7 46 F4F7 5F7 6F7 F1 F4 F7	*
13, 1	03 04 F0F5 06 F0F7 14 15 16 1F7 2P7 34 35 36 3F7 45 46 4F7 56 F5F7 6F7 F0 F5 F7	*
13, 1	03 04 06 0F7 14 15 16 1F7 2P7 F3F5 36 F3F7 46 4F7 56 F5F7 6F7 F3 F5 F7	*
13, 1	03 04 05 F0F6 F0F7 14 15 16 1F7 2P7 36 3F7 45 46 4F7 56 5F7 F6F7 F0 F6 F7	*
12, 4	04 0F5 06 0F7 F1F5 P16 F1F7 2P5 2P7 3P7 46 4F7 F56 6F7 F1 F5 F7	*
12, 4	04 05 0F6 0F7 P15 F1F6 F1F7 2P6 2P7 3P7 45 4F6 4F7 5F7 F1 F6 F7	*
12, 4	04 05 0F6 0F7 P15 F1F6 F1F7 2P6 2P7 3P7 45 4F6 4F7 5F6 F1 F6 F7	*
12, 3	P04 F05 F06 F0F7 15 16 1P7 25 26 P36 F3F7 4F7 5F7 6F7 F0 F3 F7	*
12, 3	P04 F05 F06 F0F7 15 16 1F7 P26 F2F7 3P7 4F7 56 5F7 F0 F2 F7	*
12, 3	P03 F0F4 05 F0F6 07 1P4 15 1F6 17 25 2P6 27 3F4 35 3F6 37 45 F4F6 47 57 F0 F4 F6	*
12, 3	F0F3 F05 P06 F0F7 14 16 1F7 2P7 P35 F3F7 46 5F7 F0 F3 F7	*
12, 3	F03 F0F5 P06 F0F7 14 16 1F7 2P7 3P5 3F7 46 F5F7 F0 F5 F7	*
12, 3	F03 F0F5 P06 F0F7 14 15 16 2P7 3P5 3F7 46 4F7 56 F5F7 F0 F5 F7	*
12, 3	F03 F0F5 P06 F0F7 14 15 16 1F7 2P7 3P5 46 4F7 56 F5F7 F0 F5 F7	*
12, 3	0P4 05 0F6 07 F1F4 F1F6 P17 25 2P6 27 35 37 F4F6 F47 F67 F1 F4 F6	*
12, 3	0P4 05 0F6 07 F1F4 F1F6 P17 25 2F6 27 3P6 F4F6 F47 57 F1 F4 F6	*
12, 3	0P4 05 06 0F7 F1F4 P16 25 26 2F7 3P7 F45 F46 F4F7 5F7 F1 F4 F7	*
12, 3	0P4 05 06 0F7 15 16 1F7 P26 F2F7 3P7 F46 F4F7 56 5F7 6F7 F2 F4 F7	*
12, 3	0P3 0F6 0F7 14 15 1P6 1F7 24 25 2P7 F3F6 F3F7 45 F6F7 F3 F6 F7	*
12, 3	0P3 04 05 0F6 07 14 15 1F6 17 2P7 34 35 F3F6 F3F7 4P6 4F7 F6F7 F3 F6 F7	*
12, 3	0P3 04 05 06 07 1F3 1F6 1P7 24 25 26 27 F35 F3F6 F3F7 47 5P6 57 F6F7 F3 F6 F7	*
12, 3	04 0P6 07 14 17 F2F5 F2F6 P27 3P5 3F6 4F5 4F6 F57 F67 F2 F5 F6	*
12, 3	04 0P6 07 14 17 F2F5 F2F6 P27 3P5 3F6 4F5 4F6 47 F57 F67 F2 F5 F6	*
12, 3	04 0P5 06 14 16 1F7 F2F5 P26 F2F7 3P7 45 46 4F7 56 F5F7 6F7 F2 F5 F7	*
12, 3	04 06 0P7 14 16 F2F5 P26 F2F7 3P5 3F7 4F5 46 F5F7 6F7 F2 F5 F7	*
12, 3	04 06 0F7 F1F5 P16 F1F7 2P5 2F7 3P7 46 4F7 F56 6F7 F1 F5 F7	*
12, 3	04 06 0F7 F1F5 P16 F1F7 2F5 2P7 3P5 46 4F7 F56 6F7 F1 F5 F7	*
12, 3	04 05 0P6 07 14 15 17 P25 F2F6 F2F7 3F6 3P7 45 4F6 5F7 F6F7 F2 F6 F7	*

12, 3	04 05 0F6 P15 F1F6 F1F7 2P6 3P7 45 4F7 5F7 F6F7 F1 F6 F7	*
12, 3	04 05 0F6 0F7 P15 F1F7 2P6 3P7 45 4F6 4F7 5F7 F6F7 F1 F6 F7	*
12, 3	04 05 0F6 0F7 P15 F1F6 F1F7 2P6 2F7 3P7 45 F6F7 F1 F6 F7	*
12, 3	04 05 0F6 0F7 P15 F1F6 F1F7 2P6 2F7 3P7 45 4F6 5F7 F1 F6 F7	*
12, 3	04 05 0F6 0F7 P15 F1F6 F1F7 2P6 2F7 3P7 45 4F6 4F7 F1 F6 F7	*
12, 3	03 0F5 06 07 1P4 1P5 2P4 36 37 F4F5 F46 F4F7 F56 F5F7 67 F4 F5 F7	*
12, 3	03 05 0P6 0F7 P14 F1F6 F1F7 2P7 35 4F6 4F7 F6F7 F1 F6 F7	*
12, 3	03 05 0P6 0F7 1P4 1F6 1F7 2P7 35 3F7 F4F6 F4F7 5F7 F4 F6 F7	*
12, 3	03 05 06 0F7 P14 F1F5 F16 F1F7 2P7 35 36 3F7 4P7 56 F5F7 F1 F5 F7	*
12, 2	P03 F0F5 F0F6 F07 14 16 17 2P5 3F5 46 47 F5F6 F57 67 F0 F5 F6	*
12, 2	P03 04 F0F5 06 F0F7 14 15 16 2P7 3F7 45 46 4F7 56 F5F7 6F7 F0 F5 F7	*
12, 2	P03 04 F0F5 06 F0F7 14 15 16 1P7 24 25 26 3F7 46 4F7 F5F7 6F7 F0 F5 F7	*
12, 2	F03 F0F4 F0F6 P07 14 15 1P6 17 24 25 27 35 3F6 37 F4F6 47 5F6 57 F0 F4 F6	*
12, 2	0P3 04 06 07 14 1F5 16 17 2P5 F3F5 F36 F3F7 46 47 F56 F5F7 67 F3 F5 F7	*
12, 2	04 05 06 0P7 14 15 16 F2F4 P26 F2F7 35 36 3F7 F4F7 5F7 6F7 F2 F4 F7	*
12, 2	03 F0F5 06 07 1F4 1P5 2P4 36 37 F4F5 F46 F47 F56 F57 67 F0 F4 F5	*
12, 2	03 05 0F6 07 P14 F1F6 F1F7 2P6 35 37 4F6 4F7 57 F6F7 F1 F6 F7	*
12, 2	03 05 0F6 07 14 1P6 17 24 27 35 3F6 37 P45 F4F6 F4F7 5F6 F6F7 F4 F6 F7	*
12, 2	03 05 06 F0F7 P14 F15 F16 F1F7 2P7 35 36 3F7 4F7 56 5F7 F0 F1 F7	*
12, 2	03 05 06 0F7 P14 F1F6 F1F7 2P7 35 36 3F7 4F7 56 F6F7 F1 F6 F7	*
12, 2	03 05 06 0F7 P14 F1F6 F1F7 2P7 35 36 3F7 4F6 4F7 56 5F7 F1 F6 F7	*
12, 2	03 05 06 0F7 1P4 1F6 1F7 2P7 35 36 3F7 F4F7 56 5F7 F6F7 F4 F6 F7	*
12, 2	03 04 05 07 F1F4 F15 F1F6 P17 2P6 35 3F6 37 F4F6 47 F67 F1 F4 F6	*
12, 1	04 05 06 07 14 15 17 24 26 27 F3F5 F3F6 P37 4F6 47 F5F6 F57 F67 F3 F5 F6	*
12, 1	03 F0F5 06 F0F7 14 15 16 2P7 35 3F7 46 4F7 56 F5F7 6F7 F0 F5 F7	*
12, 1	03 F0F5 06 F0F7 14 15 16 1F7 2P7 35 3F7 46 4F7 56 F5F7 F0 F5 F7	*
12, 1	03 F0F4 05 06 F0F7 14 15 16 1F7 2P7 35 36 3F7 46 F4F7 6F7 F0 F4 F7	*
12, 1	03 04 0F5 06 07 14 F1F5 16 F1F7 2P5 36 37 4F5 46 47 F56 F5F7 67 F1 F5 F7	*
12, 1	03 04 06 0F7 F1F4 15 16 F1F7 2P7 35 36 3F7 46 F4F7 56 5F7 F1 F4 F7	*
12, 1	03 04 05 F0F6 F0F7 14 15 16 2P7 36 3F7 45 46 4F7 56 5F7 F6F7 F0 F6 F7	*
12, 1	03 04 05 0F6 07 15 1P6 17 25 27 F3F4 35 F3F6 37 F4F6 47 5F6 57 F67 F3 F4 F6	*
12, 1	03 04 05 06 0F7 F1F4 15 16 F1F7 2P7 35 36 3F7 46 F4F7 5F7 F1 F4 F7	*
12, 1	03 04 05 06 0F7 14 F1F5 16 F1F7 2P7 36 3F7 45 46 4F7 56 F5F7 F1 F5 F7	*
12, 1	03 04 05 06 0F7 14 1F5 16 1F7 2P7 36 F3F7 45 46 4F7 56 F5F7 6F7 F3 F5 F7	*

	Program A, mode 2: Negabent functions, $n = 8$	
Score	Description	not MM
12, 3	F03 F0F5 P06 F07 P14 F1F5 F16 F17 24 26 27 3F5 46 47 P57 67 F0 F1 F5	*
12, 3	F03 F05 P06 F0F7 14 16 1P7 P25 F2F7 35 3F7 46 5F7 6F7 F0 F2 F7	*
12, 3	03 0P5 0F7 14 16 1P7 F2F5 P26 F2F7 3F7 46 F5F7 6F7 F2 F5 F7	*
12, 3	03 0P5 06 0F7 14 1P7 F2F5 P26 F2F7 36 3F7 F5F7 6F7 F2 F5 F7	*
12, 2	03 0P7 14 15 16 1F7 F24 F2F5 P26 F2F7 46 4F7 56 F5F7 F2 F5 F7	*
12, 2	03 0P6 F1F4 15 F1F6 17 2F4 25 2F6 27 F4F6 P47 57 F67 F1 F4 F6	*
11, 3	P04 F0F5 F06 F07 14 16 17 F2F5 P27 3F5 46 47 P56 F57 67 F0 F2 F5	*
11, 3	03 05 0P6 0F7 14 16 1P7 P25 F2F6 F2F7 35 3F7 46 F6F7 F2 F6 F7	*
11, 3	03 05 0P6 07 F1F4 P15 F1F6 2F6 35 37 F4F6 P47 57 F67 F1 F4 F6	*
11, 2	F03 F0F6 P07 14 15 1F6 17 24 25 F2F6 27 3F6 45 4P6 47 57 F67 F0 F2 F6	*
11, 2	03 0P6 14 15 F1F6 F1F7 24 25 2F6 2F7 45 4P7 F6F7 F1 F6 F7	*
11, 2	03 0F4 05 07 F1F4 15 F1F6 17 2F6 35 3P6 37 F4F6 P47 57 F1 F4 F6	*
10, 4	F0F3 P05 F0F6 P07 14 1P6 17 25 F3F6 P37 47 F67 F0 F3 F6	*
10, 4	F03 P05 F0F6 P07 14 15 17 P24 F2F6 3F6 45 4F6 47 57 P67 F0 F2 F6	*
10, 3	F0F3 F05 P06 F0F7 14 16 1P7 25 P35 46 5F7 6F7 F0 F3 F7	*
10, 3	0P3 05 0F7 14 16 1P7 F25 P26 F2F7 F3F7 46 5F7 F2 F3 F7	*
10, 3	04 0P7 15 16 1F7 P25 F2F7 P36 F3F7 56 5F7 6F7 F2 F3 F7	*
10, 3	04 06 0P7 F15 P16 F1F7 P25 F2F7 35 3F7 46 6F7 F1 F2 F7	*
10, 3	04 06 0P7 15 1F7 P25 F2F7 P36 F3F7 46 5F7 6F7 F2 F3 F7	*
10, 3	03 0P5 14 16 1P7 F2F5 P26 F2F7 3F7 46 F5F7 F2 F5 F7	*
10, 3	03 0P5 0F6 07 14 1P6 F2F5 P27 37 F5F6 F57 F67 F2 F5 F6	*
10, 3	03 0P5 07 14 1P6 F2F5 F2F6 P27 3F6 37 F57 F67 F2 F5 F6	*
10, 3	03 0P5 06 07 1F4 1F5 F2F5 P26 F27 36 37 P47 F56 F57 67 F2 F4 F5	*
10, 3	03 0P5 06 07 14 1P7 F2F5 P26 F2F7 36 37 F56 F5F7 F2 F5 F7	*
10, 3	03 05 0P6 07 F14 F1F6 P17 P25 F2F6 F27 35 37 4F6 57 F1 F2 F6	*
10, 2	F0F3 F05 P06 14 16 17 25 2F7 P35 F3F7 46 5F7 67 F0 F3 F7	*
10, 2	F03 04 F0F5 06 F07 14 1F5 16 17 P27 3F5 46 47 P56 F57 67 F0 F2 F5	*
10, 2	04 05 F0F6 07 14 15 1F6 17 F2F6 P27 3F6 45 4P6 47 57 F67 F0 F2 F6	*
10, 1	03 05 06 14 15 17 F2F5 P26 F2F7 36 37 47 F56 F5F7 F2 F5 F7	*
9, 4	F0F3 P05 F0F6 P07 14 1P6 17 25 F3F6 P37 47 F0 F3 F6	*
9, 4	03 0P5 0P7 14 16 1P7 F2F5 P26 F2F7 46 F5F7 F2 F5 F7	*
9, 3	P04 F0F6 14 17 P25 F2F6 F27 35 4F6 47 P67 F0 F2 F6	*
9, 3	P04 F05 F06 F07 14 16 17 F25 P26 F27 F35 P37 46 47 67 F0 F2 F3	*
9, 3	F0F3 F05 P06 14 16 1P7 25 27 P35 F3F7 46 57 6F7 F0 F3 F7	*

9, 3	F04 P05 F06 F0F7 14 15 P26 F2F7 36 37 45 4P7 67 F0 F2 F7	*
9, 3	F04 F0F6 P07 14 P25 F2F6 F27 35 37 4P6 57 F0 F2 F6	*
9, 3	0F3 05 0P6 P14 F15 F1F6 F17 24 26 27 F35 P37 46 47 67 F1 F3 F6	*
9, 3	04 0P7 15 16 1F7 P25 F26 P36 F3F7 56 5F7 6F7 F2 F3 F7	*
9, 3	04 0P6 07 P15 F1F6 F17 25 2F6 F3F6 P37 47 F1 F3 F6	*
9, 3	04 06 0P7 F15 P16 F1F7 P25 F2F7 35 46 5F7 F1 F2 F7	*
9, 3	04 06 07 P14 F1F7 F25 P26 F2F7 35 46 4F7 5P7 67 F1 F2 F7	*
9, 3	03 0P5 14 16 1P7 F2F5 P26 F2F7 46 F5F7 F2 F5 F7	*
9, 3	03 0P5 0F7 14 16 1P7 F2F5 P26 F2F7 3F7 46 F2 F5 F7	*
9, 3	03 0P5 06 07 F1F4 F1F5 P17 2F5 36 37 P46 F47 67 F1 F4 F5	*
9, 3	03 0P5 06 07 F14 P17 F2F5 P26 F27 36 37 F56 F57 67 F1 F2 F5	*
9, 3	03 0P5 06 07 1F4 1F5 F2F5 P26 36 37 P47 F56 F57 67 F2 F4 F5	*
9, 3	03 06 0P7 F1F4 F15 P16 2P4 25 36 F4F7 5F7 F1 F4 F7	*
9, 3	03 05 P06 F0F7 14 16 1P7 P25 F2F7 35 3F7 46 5F7 F0 F2 F7	*
9, 2	P04 F05 F0F6 14 25 26 27 F35 F3F6 P37 57 67 F0 F3 F6	*
9, 2	F04 F05 F0F6 P07 14 16 17 P25 F2F6 35 47 67 F0 F2 F6	*
9, 2	04 06 07 P14 F16 F1F7 P25 F2F7 35 3F7 46 47 F1 F2 F7	*
9, 2	04 05 06 14 16 F1F7 F25 P26 F2F7 3F7 45 46 47 56 5P7 67 F1 F2 F7	*
9, 2	03 0P4 F1F4 F15 F1F6 P17 25 26 27 3F6 57 67 F1 F4 F6	*
9, 2	03 0P4 05 07 F1F4 F15 P16 26 27 35 37 F46 F4F7 67 F1 F4 F7	*
9, 2	03 06 0P7 F14 F15 P16 F1F7 24 25 36 4F7 5F7 6F7 F1 F7	*
9, 2	03 05 06 07 F1F4 F1F6 P17 2F4 2F6 35 37 P45 56 57 67 F1 F4 F6	*
9, 2	03 05 06 07 F14 F1F5 P17 F2F5 P26 36 37 56 F57 67 F1 F2 F5	*
9, 1	F0F3 05 06 F0F7 14 17 25 26 27 P36 F3F7 56 57 F0 F3 F7	*
9, 1	04 F05 F0F6 07 14 15 16 17 P25 F2F6 3F6 45 47 57 67 F0 F2 F6	*
9, 1	04 06 07 P15 F1F7 25 2F7 36 F3F7 46 5F7 67 F1 F3 F7	*
9, 1	03 F0F4 F06 07 F1F4 15 F16 17 26 35 37 P46 57 F0 F1 F4	*
9, 1	03 06 07 F1F4 15 16 F1F7 2P4 25 27 36 F4F7 F1 F4 F7	*
9, 1	03 05 06 14 16 17 P25 F2F6 F2F7 35 37 47 F6F7 F2 F6 F7	*
9, 1	03 05 06 07 F1F4 F1F5 P16 2F5 36 37 47 F56 57 67 F1 F4 F5	*
9, 1	03 05 06 07 F14 F1F5 16 F17 F2F5 P27 36 37 56 67 F1 F2 F5	*
9, 1	03 05 06 07 14 16 17 P25 F2F6 F2F7 35 37 57 F6F7 F2 F6 F7	*
9, 0	03 04 06 07 F1F4 15 16 F1F7 25 27 36 46 F4F7 57 F1 F4 F7	*

	Program A combination: Bent-negabent functions, $n = 8$	
Score	Description	not MM
6, 3	F03 F0F4 F06 P07 1P4 15 16 17 25 27 36 P46 57 F0 F4	*
5, 4	P04 P06 F0F7 15 1P7 26 2F7 3P7 6F7 F0	
5, 4	P04 P06 F0F7 15 1P7 26 2F7 3P7 4F7 F0	
5, 3	04 0P6 07 1F5 1P6 2P5 27 37 47 F5F6 F67 F5	
5, 3	04 06 0P7 F15 P16 F1F7 25 2P7 35 46 5F7 F1	*
5, 3	04 05 06 14 16 1F7 F25 P26 F2F7 3P7 45 46 47 56 5P7 67 F2	*
5, 2	P03 F0F4 P06 F07 14 15 16 17 25 26 27 47 56 57 67 F0 F4	*
5, 2	03 05 06 07 F1F4 P16 F17 25 27 36 P47 57 F1 F4	*
4, 5	04 0P6 0F7 15 1P7 2P6 2P7 3P7 4F7 F6F7	
4, 4	P04 P06 F0F7 15 1P7 26 3P7 6F7 F0	
4, 4	P04 P06 F0F7 15 1P7 26 2F7 3P7 F0	
4, 4	P04 F0F6 P07 15 1P6 17 2P6 27 37 4F6 57 F0	
4, 4	04 0P6 07 P15 F1F6 P17 2P6 27 37 47 F67 F1	
4, 4	04 06 0P7 15 1P6 1F7 2P6 3P7 46 5F7 F6F7	
4, 3	0P4 06 15 1P7 26 2P7 36 F4F7 6F7 F4	
4, 3	04 0P6 07 1F5 1P6 2P5 27 37 47 F5F6 F5	
4, 3	04 0F5 0P6 07 1P5 17 2P6 37 47 F5F6 F5	
4, 3	04 06 0P7 1P5 16 1F7 26 3P7 46 F5F7 F5	
4, 3	04 05 0P6 15 17 F2F6 P27 3P6 45 57 F67 F2	*
4, 2	P03 F0F4 P06 14 15 16 17 25 26 27 47 56 57 67 F0 F4	*
4, 2	0P4 06 07 15 1P7 26 27 36 F4F7 6F7 F4	
4, 2	04 06 0P7 15 16 1F7 25 2F7 3P7 46 56 5F7 6F7	
4, 2	04 05 0P6 07 15 1F6 17 2F6 27 3P6 45 47 5F6 57 F67	
4, 2	04 05 0F6 07 15 1P6 17 2P6 37 47 F5F6 57 F5	
4, 2	04 05 06 0P7 15 1F7 26 2F7 3P7 45 46 56 5F7 6F7	
4, 2	04 05 06 07 P15 F1F7 26 2F7 3P7 45 46 56 57 67 F1	*
4, 2	03 0P5 14 16 17 F2F5 26 27 46 P57 67 F2 F5	*
4, 2	03 05 06 F07 P14 P16 F17 25 26 27 37 56 57 67 F0 F1	*
4, 2	03 05 06 07 P14 F1F5 P17 25 26 36 37 56 57 67 F1 F5	*
4, 1	P03 F0F5 06 07 14 15 17 24 26 27 45 46 47 56 57 67 F0 F5	*
4, 1	P03 04 F0F5 07 14 15 16 17 25 26 46 47 57 67 F0 F5	*
4, 1	04 06 F0F7 15 1P7 26 2F7 37 47 67 F0	
4, 1	04 05 F06 F0F7 15 16 17 26 3P7 46 47 56 57 67 F0	
4, 1	04 05 06 07 1P5 1F7 26 27 36 46 47 F5F7 67 F5	

4, 1	04 05 06 07 1F5 1P6 25 27 37 46 47 F5F6 57 F5	
4, 1	03 05 06 07 P14 15 F1F6 17 24 2F6 35 37 67 F1	*
4, 1	03 05 06 07 1F4 15 17 2P4 36 37 F45 46 F47 56 57 67 F4	*
4, 1	03 05 06 07 14 15 16 2P7 35 36 46 47 F5F7 6F7 F5	*
4, 0	F0F3 05 06 07 14 15 16 17 24 26 37 45 47 56 57 67 F0 F3	*
4, 0	04 F0F5 15 16 17 26 27 36 47 57 F0 F5	
4, 0	04 05 07 14 16 17 25 26 27 36 F4F5 46 47 57 F4 F5	
4, 0	03 05 0F7 14 16 17 25 26 F2F7 37 46 57 F2	*
4, 0	03 05 06 07 14 F1F5 16 17 25 27 36 37 56 57 67 F1 F5	*
3, 5	04 0P6 15 1P7 2P6 2P7 3P6 4F7 F6F7	
3, 5	04 0P6 0P7 15 1P7 2P6 2F7 3P7 F6F7	
3, 4	P04 P06 F07 P15 P17 26 37 F0 F1	
3, 4	P03 F04 P07 14 1P5 16 17 2P5 26 27 46 47 67 F0 F5	*
3, 4	04 0P6 07 1P5 1F6 2P6 27 37 47 P57 F67 F5	
3, 3	P04 P06 F07 1P5 17 26 37 47 F0 F5	
3, 3	P04 P06 F07 1P5 17 26 27 37 67 F0 F5	
3, 3	P04 P06 F07 15 16 17 P25 37 56 57 67 F0 F2	*
3, 3	P04 F06 15 17 2P5 36 37 47 P56 57 67 F0 F5	
3, 3	0P4 05 07 1P4 16 25 27 F36 P37 57 F3 F4	*
3, 3	04 0P6 07 1P5 16 17 2F5 2P6 36 37 47 F67 F5	
3, 3	04 0P5 06 07 1P5 16 26 27 3P7 46 47 F5F7 67 F5	
3, 2	P04 F0F6 P07 14 16 25 26 27 35 37 46 57 67 F0	*
3, 2	P03 P05 F06 F07 14 16 17 25 26 27 47 56 57 F0	*
3, 2	P03 F04 P07 14 15 16 17 25 26 27 46 47 67 F0 F2	*
3, 2	F03 P05 F06 P07 14 15 17 24 26 27 36 45 46 47 57 F0	*
3, 2	04 0P6 07 15 1F6 25 27 3P6 47 5F6 57 F67	
3, 2	04 0P6 07 15 1F6 17 2F6 27 3P6 47 F67	
3, 2	04 0P6 07 15 1F6 17 25 3P6 47 5F6 57 F67	
3, 2	04 0P6 07 15 1F6 17 25 2F6 3P6 47 57 F67	
3, 2	04 0P5 06 07 1P5 26 27 36 46 47 F5F7 67 F5	
3, 2	04 0P5 06 07 1P5 16 17 2F5 27 36 37 46 47 F57 F5	
3, 2	04 0P5 06 07 1P5 16 17 26 37 46 47 F56 F57 67 F5	
3, 2	04 0P5 06 07 1F5 16 17 2P5 37 46 47 F56 67 F5	
3, 2	04 0P5 06 07 1F5 16 17 2P5 27 37 46 47 F56 F5	
3, 2	04 0F5 06 07 1P5 16 26 27 37 47 F56 P57 67 F5	
3, 2	04 06 0P7 15 16 1F7 25 2F7 3P7 46 56 5F7	

3, 2	04 06 0P7 15 16 17 25 F2F7 3P7 46 56 57 F2	*
3, 2	04 06 0F7 15 1P7 26 2P7 36 4F7 6F7	
3, 2	04 06 07 15 16 2F6 2P7 3P6 47 57 F6F7	
3, 2	04 06 07 15 16 17 P25 F2F6 3P6 47 57 F2	*
3, 2	04 06 07 15 16 17 2P6 2F7 3P7 47 F6F7	
3, 2	04 06 07 15 16 17 2F6 2P7 3P6 47 F6F7	
3, 2	04 05 0P6 07 15 F2F6 27 3P6 45 47 57 F2	*
3, 2	04 05 0P6 07 15 1F6 2F6 27 3P6 45 47 57 F67	
3, 2	04 05 06 07 15 1P7 26 F2F7 3P7 45 46 56 57 67 F2	*
3, 2	04 05 06 07 15 16 P26 F27 P37 47 56 57 67 F2 F3	*
3, 2	03 0P5 06 07 14 1F5 16 17 2P5 36 37 46 F57 F5	*
3, 2	03 05 0P6 07 14 15 1F6 17 2P6 35 37 4F6 47 F67	*
3, 2	03 05 06 P14 P15 F16 F17 26 27 35 37 57 F1	*
3, 2	03 05 06 14 16 P25 F26 P27 35 37 57 67 F2 F4	*
3, 2	03 05 06 0P7 14 15 1F7 2P7 35 36 46 4F7 5F7	*
3, 2	03 05 06 0P7 14 15 16 1F7 2P7 35 36 46 4F7 5F7	*
3, 2	03 05 06 0P7 14 15 16 17 2P7 35 36 46 47 F5F7 F5	*
3, 2	03 05 06 07 P14 F15 P17 25 26 27 36 37 56 57 67 F1 F3	*
3, 2	03 04 07 P14 F15 F16 P17 25 26 37 46 47 F1	*
3, 1	P03 F05 06 07 14 15 17 24 26 27 45 46 47 56 57 67 F0 F1	*
3, 1	0P4 05 07 15 16 17 25 27 36 37 F46 F47 56 57 F4	
3, 1	0P4 05 06 07 15 16 17 25 27 36 37 F4F7 57 F4	
3, 1	0F4 06 07 1P4 16 25 26 27 35 37 57 F4 F5	
3, 1	04 0P5 06 07 14 16 17 2F5 27 36 37 4F5 67 F5	
3, 1	04 0F5 07 1P5 16 17 26 27 36 F57 F5	
3, 1	04 0F5 07 14 16 17 2P5 26 27 36 37 46 F57 67 F5	
3, 1	04 06 0F7 15 16 1P7 25 26 35 4F7 6F7	
3, 1	04 06 07 F15 16 25 2P7 35 46 5F7 67 F1	*
3, 1	04 05 F0F6 15 16 17 2P6 37 46 47 57 F0	
3, 1	04 05 06 07 1F5 16 2P5 27 37 46 47 56 F57 F5	
3, 1	04 05 06 07 1F5 16 17 2F5 26 3P5 46 47 57 67 F5	
3, 1	04 05 06 07 15 17 26 27 3P6 47 F5F6 67 F5	
3, 1	03 05 07 F14 F16 P17 25 26 27 46 F1	
3, 1	03 05 07 14 1P6 25 F2F6 27 36 37 57 67 F2	*
3, 1	03 05 06 07 P14 F15 16 F17 25 27 36 37 56 67 F1	*
3, 1	03 05 06 07 1F4 15 17 2P4 36 37 F45 46 56 57 67 F4	*

3, 1	03 05 06 07 14 15 16 17 2P5 36 37 46 F5F7 F5	*
3, 0	F04 06 07 14 25 26 27 35 37 46 57 F0 F2	*
3, 0	F03 F04 06 07 14 15 16 17 25 27 35 36 37 56 57 67 F0	*
3, 0	04 0F5 06 15 16 17 25 27 36 37 46 47 F57 67 F5	
3, 0	04 06 07 15 16 17 25 26 35 37 F4F7 57 F4	
3, 0	04 06 07 14 17 25 26 27 35 F45 F46 47 56 57 F4	
3, 0	04 05 15 16 26 27 36 47 F5F7 67 F5	
3, 0	04 05 15 16 17 26 27 36 47 F5F7 F5	
3, 0	04 05 07 15 16 17 26 27 36 F5F7 F5	
3, 0	04 05 06 15 16 17 25 27 36 37 46 F5F7 67 F5	
3, 0	04 05 06 07 15 16 17 2F5 27 36 37 46 F57 F5	
3, 0	04 05 06 07 14 17 25 36 37 F45 F46 47 56 57 F4	
3, 0	04 05 06 07 14 16 17 25 27 36 46 F47 56 57 F4 F5	
3, 0	03 F0F6 07 14 15 16 17 24 25 27 36 45 46 47 57 F0	*
3, 0	03 F05 06 F07 14 15 16 17 24 26 27 35 46 47 57 67 F0	*
3, 0	03 06 07 14 15 16 F1F7 24 25 27 36 47 F1	*
3, 0	03 05 14 16 17 25 26 F2F7 37 46 67 F2	*
3, 0	03 05 14 16 17 25 26 F2F7 37 46 57 F2	*
3, 0	03 05 07 14 15 F1F6 17 24 25 26 36 37 57 67 F1	*
3, 0	03 05 06 14 F1F6 17 25 27 35 46 57 67 F1	*
3, 0	03 05 06 14 15 F16 F17 26 27 35 37 46 47 57 67 F1	*
3, 0	03 05 06 07 14 F1F6 17 25 35 37 46 57 67 F1	*
3, 0	03 05 06 07 14 1F5 16 17 24 36 37 4F5 46 47 57 F5	*
3, 0	03 05 06 07 14 15 F16 F17 26 27 35 47 F1	*
3, 0	03 05 06 07 14 15 F16 F17 26 27 35 37 46 57 F1	*
3, 0	03 05 06 07 14 15 17 2F5 26 36 37 46 47 F56 57 67 F5	*
3, 0	03 04 06 07 14 15 16 17 25 27 F35 36 F37 46 57 F3	*
3, 0	03 04 05 F06 F07 14 15 16 17 26 27 36 45 46 47 57 67 F0	*

5 Conclusion

In this thesis we have studied generalizations based on the Maiorana-McFarland construction. We've found certain criteria, *constraints*, that are revealed if one splits the Hadamard or Nega-Hadamard transform into partial transforms and apply them in sequence. Using this idea, we developed two software programs using different approaches to finding these constraints. We have enumerated all bent, negabent and bent-negabent functions up to $n = 6$ and have used these functions as templates for our software, and subsequently have found numerous new promising constructions for bent, negabent and bent-negabent functions. Most interesting is the fact that some of these constructions are different from the non-extended MM construction, and can therefore be used to generate bent and/or negabent functions possibly outside the extended MM class.

5.1 Further work

One important open question is if some of the constructions we have found which are outside the non-extended MM class, will also be outside the extended MM class. If this is the case, we are also interested in studying the properties of the functions generated with these constructions.

We have briefly explored if the partial (nega-)Hadamard transforms could be represented as graph operations. While we haven't found any support for this so far, it's worth investigating further.

The symbolic software has the ability to do both Hadamard and nega-Hadamard transforms on the same function. This ability could be used to explore the properties of functions under mixed transforms.

While we have found promising results, we have been somewhat limited by the unfinished state of the software we've created. Some of the improvements and ideas for further development are listed below:

- First and foremost, make our symbolic computation software able to handle functions of cubic or higher degree
- Rewrite symbolic computation software to be faster in some edge cases
- Remove unnecessary restrictions of functions to identity in the symbolic software
- Make the symbolic software understand linear constraints and locate these

- Fix the random sampling software to evaluate the full search tree
- Evaluate feasibility of having the random sampling software use Graphic Processing Units for faster (nega)bentness checking

Bibliography

References

[1] O. S. Rothaus. On "bent" functions. *Journal of Combinatorial Theory, Series A*, 20:300–305, May 1976.

[2] Serge Mister and Carlisle Adams. Practical S-Box Design. In *Selected Areas in Cryptography*, 1996.

[3] John Olsen, Robert Scholtz, and Lloyd Welch. Bent-Function Sequences. *IEEE Transactions on Information Theory*, 28(6):858–864, 1982.

[4] Jacques Wolfmann. Bent Functions and Coding Theory. In A. "Pott, P. V. Kumar, T. Helleseth, and D." Jungnickel, editors, *Difference Sets, Sequences and Their Correlation Properties*, volume 542 of *NATO Science Series*, pages 393–418. Springer Netherlands, 1999.

[5] Claude Carlet, Lars Eirik Danielsen, Matthew G. Parker, and Patrick Solé. Self-dual bent functions. Fourth International Workshop on Boolean Functions: Cryptography and Applications (BFCA 2008). Copenhagen, Denmark.

[6] J. F. Dillon. *Elementary Hadamard Difference sets*. PhD thesis, Univ. of Maryland, 1974.

[7] M.G. Parker. The Constabent Properties of Golay-Davis-Jedwab Sequences. Int. Symp. Information Theory, Sorrento, Italy, 2000.

[8] Matthew G. Parker and Alexander Pott. On Boolean functions which are bent and negabent. In *Proceedings of the 2007 international conference on Sequences, subsequences, and consequences*, SSC'07, pages 9–23, Berlin, Heidelberg, 2007. Springer-Verlag.

[9] Kai-Uwe Schmidt, Matthew G. Parker, and Alexander Pott. Negabent Functions in the Maiorana—McFarland Class. In *Proceedings of the 5th international conference on Sequences and Their Applications*, SETA '08, pages 390–402, Berlin, Heidelberg, 2008. Springer-Verlag.

[10] P. Stanica, S. Gangopadhyay, A. Chaturvedi, A. Gangopadhyay, and S. Maitra. Investigations on bent and negabent functions via the

nega–Hadamard transform. *Information Theory, IEEE Transactions on*, PP(99):1, 2012.

[11] Brendan D. McKay. *nauty User's Guide*, 2003. http://cs.anu.edu.au/ bdm/nauty/nug.pdf.

[12] Y. Ådlandsvik. `http://ymgve.net/master/`.

[13] Bart Preneel. *Analysis and Design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.

[14] Meng Qing-shu, Yang min, Zhang huan guo, and Cui jing song. A Novel Algorithm Enumerating Bent Functions, 2004.

[15] Stuart W. Schneider. Finding Bent Functions Using Genetic Algorithms, 2009.

[16] Philippe Langevin and Gregor Leander. Counting all bent functions in dimension eight 99270589265934370305785861242880. *Des. Codes Cryptography*, 59(1-3):193–205, April 2011.

[17] Claude Carlet and Andrew Klapper. Upper Bounds on the Numbers of Resilient Functions and of Bent Functions. In *23rd Symposium on Information Theory in the Benelux*, pages 307–314, May 2002.