Theorem 6 *Let $S(k)$ be a $2^k \times 2^k$ Sylvester-Hadamard matrix, $h_k : \{0,1\}^k \times \{0,1\}^k \to \{-1,1\}$ be defined as $h_k(u,v) = S(k)_{u,v}$. Then h_k can be represented by a network with one linear output and two hidden layers with k Heaviside perceptrons in each hidden layer.*

The proof of this theorem, given in the Appendix, utilizes the well-known and easily verifiable by induction) equivalence of $2^k \times 2^k$ Sylvester-Hadamard matrix to the matrix with rows formed by generalized parities $p_u(v) : \{0,1\}^k \to \{-1,1\}$ defined as

$$p_u(v) = -1^{u \cdot v}$$

(see, e.g., [30]). Thus functions generated by Sylvester-Hadamard matrices are compositional functions. Inner products $u \cdot v$ can be computed by the first hidden layer and the classification of odd and even numbers by the second hidden layer.

Proof of Theorem 6
Any $2^k \times 2^k$ Sylvester-Hadamard matrix $S(k)$ is equivalent to the matrix $M(k)$ with rows and columns indexed by vectors $u, v \in \{0,1\}^k$ and entries

$$M(k)_{u,v} = -1^{u \cdot v}$$

(see, e.g., [30]). Thus without loss of generality we can assume that $S(k)_{u,v} = -1^{u \cdot v}$ (otherwise we permute rows and columns).

To represent the function $h_k : \{0,1\}^k \times \{0,1\}^k \to \{-1,1\}$ by a two-hidden-layer network, we first define k Heaviside perceptrons from the first hidden layer. Choose any bias $b \in (1,2)$ and define input weights $c^i = (c^{i,l}, c^{i,r}) \in \mathbb{R}^k \times \mathbb{R}^d$, $i = 1, \ldots, k$, as $c^{il}_j = 1$ and $c^{ir}_j = 1$ when $j = i$, otherwise $c^{il}_j = 0$ and $c^{ir}_j = 0$. So for an input vector $x = (u,v) \in \{0,1\}^k \times \{0,1\}^k$, the output $y_i(x)$ of the i-th perceptron in the first hidden layer satisfies $y_i(x) = \vartheta(c^i \cdot x - b) = 1$ if and only if both $u_i = 1$ and $v_i = 1$, otherwise $y_i(x)$ is equal to zero.

Let $w = (w_1, \ldots, w_k)$ be such that $w_j = 1$ for all $j = 1, \ldots, k$. In the second hidden layer, define k perceptrons by $z_j(y) := \vartheta(w \cdot y - j + 1/2)$. Finally, for all $j = 1, \ldots, k$ let the j-th unit from the second hidden layer be connected with one linear output unit with the weight $(-1)^j$.

The two-hidden-layer network obtained in this way computes the function $\sum_{j=1}^k (-1)^j \vartheta(w \cdot y(x) - j + 1/2)$, where $y_i(x) = \vartheta(c^i \cdot x - b)$, i.e., it computes the function $\sum_{j=1}^k (-1)^j \vartheta(\sum_{i=1}^{d/2} \vartheta(c^i \cdot x - b) - j + 1/2) = h_k(x) = h_k(u,v) = -1^{u \cdot v}$. □

Corollary 4 *Let $S(k)$ be a $2^k \times 2^k$ Sylvester-Hadamard matrix, $h_k : \{0,1\}^k \times \{0,1\}^k \to \{-1,1\}$ be defined as $h_k(u,v) = S(k)_{u,v}$. Then h_k can be represented by a two-hidden-layer network with k Heaviside perceptrons in each hidden layer, but every representation of h_k by one-hidden-layer Heaviside perceptron network has at least $\frac{2^k}{k}$ units or some of absolute values of output weights are greater or equal to $\frac{2^k}{k}$.*

Matlab – program na nastavenie 2-vrstvovej NS, k=4

```
fk=4;
S2=[1 1; 1 -1];
S4=[S2 S2; S2 -S2];
S8=[S4 S4; S4 -S4];
S16=[S8 S8; S8 -S8];
```

```
U16=[0 0 0 0; 0 0 0 1; 0 0 1 0; 0 0 1 1;
0 1 0 0; 0 1 0 1; 0 1 1 0; 0 1 1 1;
1 0 0 0; 1 0 0 1; 1 0 1 0; 1 0 1 1;
1 1 0 0; 1 1 0 1; 1 1 1 0; 1 1 1 1];

V16=[0 0 0 0; 0 0 0 1; 0 0 1 0; 0 0 1 1;
1 0 0 0; 1 0 0 1; 1 0 1 0; 1 0 1 1;
0 1 0 0; 0 1 0 1; 0 1 1 0; 0 1 1 1;
1 1 0 0; 1 1 0 1; 1 1 1 0; 1 1 1 1];
% Nastavenie vah pre vstup
c=[1 0 0 0 1 0 0 0; 0 1 0 0 0 1 0 0; 0 0 1 0 0 0 1 0; 0 0 0 1 0 0 0 1];
% Nastavenie vah vo vrstvach
w=[1 1 1 1];
wf=[-1 1 -1 1];

yy=NaN (1,4);

% Vypocet vystupov prvej skrytej vrstvy
y=NaN(1,4);
z=NaN(1,4);
b=1.5;
for ii=1:16
for jj=1:16
U=[U16(ii,1) U16(ii,2) U16(ii,3) U16(ii,4)];
V=[V16(jj,1) V16(jj,2) V16(jj,3) V16(jj,4)];
 for k=1:4
   pom1=[U V];
   cc=[c(k,1) c(k,2) c(k,3) c(k,4) c(k,5) c(k,6) c(k,7) c(k,8)];
   pom=cc*pom1';
   if pom-b>=0 pom2=1; else pom2=0; end;
   y(k)=pom2;
 end;

% Vypocet vystupov druhej skrytej vrstvy
 for j=1:4
    pom4=w*y';
    if pom4-j+0.5>=0 pom5=1; else pom5=0; end;
    z(j)=pom5;
 end;

% Vypocet vysledku vystupneho neuronu
    hh=wf*z';
    if hh>=0 pom6=1; else pom6=-1; end;
    val(ii,jj)=pom6;
    end;
end;
val

%% Ortogonalita
for ii=1:16
   h1=val(ii,1:16);
for jj=1:16
   h2=val(jj,1:16);
   vv(ii,jj)=h1*h2';
end
end
vv
```