Towards a side-channel model for simple logic functions and circuits: transistor level implementation of CMOS combinatorial logic circuits and simulation in SPICE

E. Dewitte, B. De Moor, B. Preneel
Katholieke Universiteit Leuven
Department of Electrical Engineering, ESAT-SCD-SISTA/COSIC
Kasteelpark Arenberg10, B-3001 Heverlee (Leuven), Belgium
{Evelyne.Dewitte, Bart.Demoor, Bart.Preneel}@esat.kuleuven.ac.be

1 Introduction

The past five years numerous papers have been published on side-channel analysis attacks. Since Kocher published the first paper on timing and power analysis attacks [13, 14], others have taken interest in the study of these threats [5, 6, 11, 16, 18], the discovery of new kinds of side-channels [1, 9] and the development of new analysis methods [2, 12, 17].
Our study of these attacks gave rise to a new idea. What if we tried to model these side-channel effects by means of a MIMO (Multiple Input Multiple Output) model and then applied advanced system identification techniques to extract information on the secret key? A lot of expertise on modeling and identification techniques is present but - as far as we know - has not yet been used in this field ...
For years already, the subgroup COSIC at our department is studying side-channel attacks [3, 19, 20]. Especially power analysis attacks have been investigated well. Power measurements are done with a digital oscilloscope. Ours is a Tektronix TDS714L with 8 Mbyte extra memory.
Of course, studying attacks has its aim in developing countermeasures to make algorithms and implementations more secure. As side-channel attacks are tailored at specific implementation strategies, countermeasures are included when the algorithm gets implemented. These protections should be situated as well on software as on hardware level [4, 6, 20]. Adjusting implementations costs money and time; as the field is rapidly evolving, simulations are to be preferred to test cryptographic systems before implementations are carried through. Building appropriate estimation and simulation tools for side-channel analysis are an interesting future challenge.
In this paper we build the foundations to our ambitious idea. We investigate if it makes sense to start building up models as suggested. This is done by focusing on the basic building blocks of all cryptographic implementations: the logic Boolean functions.

2 Aim

Our ultimate goal would be to construct a formal MIMO (Multiple Input Multiple Output) model that captures the power consumption characteristics of a digital CMOS system performing a cryptographic algorithm. The key is modeled as an unknown input to the system. Identification techniques [7, 8, 22] could then be applied to recover this input. In this paper, such a model will be tried out on simple logic digital functions: NAND, NOR, XOR (eXclusive-OR).

3 Background

3.1 General power consumption dependencies

As mentioned, the goal is to model the power consumption characteristics of cryptographic implementations. Roughly spoken, this power consumption is dependent on:

- the cryptographic algorithm which specifies when and how plaintexts are processed.
- the device. There is for instance a big difference between software implementations and hardware implementations. In software, operations are programmed to be performed step-by-step while in hardware operations are performed in parallel. So if we have to execute four operations in a row, a software implementation will do one operation per clock cycle and hence take four cycles for the whole computation while a hardware implementation can execute the four in just one cycle. This clearly influences the amount of power used.
  Also, differences show up when distinct processors are used. For instance, a 64-bit processor will show a different power consumption signal while running a DES algorithm than a smartcard implementation of DES because smartcards use 8-bit processors. So in order to make a good model it is advised to have a profound idea of the platform.
- the measurement equipment.

3.2 CMOS power consumption

Almost all digital circuits built nowadays are based on Complementary Metal Oxid Semiconductor (CMOS) technology. If a CMOS gate changes its state, this change can be measured at the V_{dd} or V_{ss} pin. At COSIC we use current probes for our measurements; therefore in the simulations we describe in this paper, we simulated the current flow to or from V_{dd}. As the source has a constant voltage ($1.8V$ in our setup), this current trace is completely proportional to the power consumption trace.

The power consumption of a CMOS gate can be written as

$$\begin{aligned} P &= P_{lk} + P_{sc} + P_{sw} \\ &\text{where} \\ P_{lk} &= \text{leakage or static power dissipation ;} \\ P_{sc} &= \text{short circuit power ;} \\ P_{sw} &= \text{switching or dynamic power consumption .} \end{aligned}$$

3.2.1 Leakage or static power consumption:

The static dissipation results from small current leakages in diodes and transistors. Consequently, it can be expressed as the product of the device leakage current and the supply voltage. A useful estimate is to allow a leakage current of 0.1nA to 0.5nA per gate at room temperature. Then total static power dissipation P_{lk} is obtained from

$$\begin{aligned} P_{lk} &= I_L * V_{dd} \\ P_{lk} &= \sum_{1}^{n} \text{leakage current} * \text{supply voltage} , \end{aligned}$$

where n is the total number of devices. The leakage current consists of two major contributions:

$$\begin{aligned} I_L &= I_{sub} + I_{gate} \\ &\text{where} \\ I_{sub} &= \text{Subthreshold current caused by low threshold voltage ;} \\ I_{gate} &= \text{Gate current caused by reduced thickness of gate oxide .} \end{aligned}$$

3.2.2 Short circuit power dissipation:

Short circuit power consumption refers to the amount of power that is dissipated in the momentary existence of a direct path from the power supply to ground. This happens when both NMOS and PMOS devices are conducting. For carefully designed circuits, short circuit power dissipation is negligible.

3.2.3 Switching or dynamic power consumption:

Dynamic power consumption is due to the charge and discharge of the load capacitance C_L. It describes the most important part in the power consumption of a CMOS device. It can be evaluated by observing that during the low-to-high transition, C_L is loaded with a charge $C_L V_{DD}$. This charge requires an energy from the supply equal to $C_L V_{DD}^2$. The energy stored on the capacitor equals 0.5 $C_L V_{DD}^2$: only half of the energy supplied by the power source is stored on C_L. The other half has been dissipated by the PMOS transistor. Notice that the percentage of energy dissipation is independent of the size (and hence the

resistance) of the PMOS device. During the discharge phase, the charge is removed from the capacitor, and its energy is dissipated in the NMOS device. We can now express the power consumption by:

$$P_{sw} = 0.5\ C_L\ V_{DD}^2\ f_{\text{clock}}\ E_{sw}\ ,$$

where E_{sw} is the switching activity factor. It represents the probability that the output node makes a transition at each clock cycle and models the fact that - in general - switching does not occur at f_{clock}, the clock frequency.

From the above reasoning it is now clear that the power dissipation is data dependent and is a function of the switching activity of a circuit. This point is essential for security as it is the basic assumption made by most attackers.

4 Strategy

Modeling a full cryptographic algorithm at once is an extremely hard job due to all the variables we mentioned above that have to be included in the model. In a first phase however, we can and even should restrict ourselves to more basic models. The proposed goal is very ambitious . . . How could this goal ever be achieved if it would not be applicable to simple logic functions? Basic Boolean functions are the building bricks of every cryptographic algorithm. So it is just natural to try out modeling these Boolean functions. If we fail, we don't have to deal with the bigger issues . . .
Modeling these logic functions has an advantage we will exploit: at this low level of abstraction (i.e. at the micro- electronics level) we dispose of an excellent estimation and simulation tool: SPICE. SPICE is an acronym for Simulation Program with Integrated-Circuit Emphasis. It has become a standard for computer-aided circuit analysis for microelectronic circuits [10, 15, 21].
To make the above more clear, let us consider the various levels of representation models.

1. The first level of representation consists merely of the algorithm itself;
2. We may have a software description of this algorithm;
3. (a) As software runs on a hardware processor the third level would then consist of a hardware description of this processor;
   (b) We can however also directly start with a hardware description of our algorithm. To execute AES, for instance, hardware descriptions are made in VHDL or VERILOG. VHDL-1076 (VHSIC (Very High Speed Integrated Circuits) Hardware Description Language) is an IEEE Standard since 1987;
4. These HDL descriptions are translated into gates, called standard cells. This process is usually called synthesis;

5. Gates are translated into transistors. Traditionally, standard cells are made with *complementary static CMOS*. They use NMOS and PMOS transistors in complementary trees. At this transistor level, we can make use of SPICE.

It is convenient to distinguish Boolean functions on two-valued elements by labeling them *switching functions*. A *switching variable* is then a letter which may take on either of the element values 0 or 1.
Now, the next issue is what we want to investigate with our switching functions. The answer is two-fold and is easily explained by looking at figure 1.

1. Given the inputs to an unknown switching function and the power dissipation of the circuit performing the switching function, can we distinguish between the probable function candidates?

2. Given the switching function and the power dissipation of the circuit performing the function, can we distinguish between possible input combinations?

Why are these interesting questions? Logic circuits take a fixed number of variables, n, which serve as inputs. As there are 2^n possible ways to assign values to the inputs, a switching function has 2^{2^n} possible outcomes. In other words, there are 2^{2^n} possible switching functions of n variables.
Truth tables completely specify switching functions. Thus if we would be allowed to feed all possible input combinations to the circuit and record all matching outputs, we would have completely defined the function. In our construction however, we don't have direct access to the outputs of the circuit. The question is then whether the power consumption traces reveal information on the outcome of the circuit. This would enable us to reconstruct the truth table and hence to recover the switching function.
Summarized, by answering the formulated questions, we investigate what information is leaked by the power traces as well on input sequences as on the circuit.

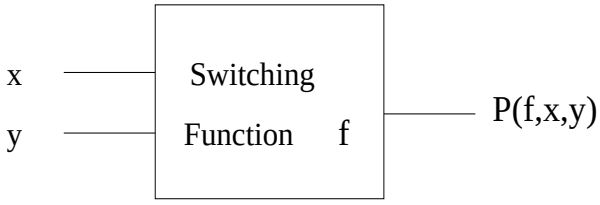

Figure 1: Simple MIMO modeling of a switching function using power consumption as an output

5 Synthesis of some simple Boolean functions

5.1 The NAND function

The truth table of the NAND function:

	X	Y	X ↑ Y
0	0	0	1
1	0	1	1
2	1	0	1
3	1	1	0

.

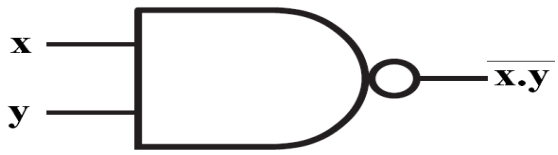

Figure 2: Standard IEEE symbol for the NAND function

5.2 The NOR function

The truth table of the NOR function:

	X	Y	X ↓ Y
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	0

.



Figure 3: Standard IEEE symbol for the NOR function

5.3 The XOR function

5.3.1 Mixed gate representation

The truth table of an XOR function, simply looks like:

	X	Y	X $\oplus$ Y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

.

The minterm expression (standard sum of products) for the XOR function is then:

$$\begin{aligned} X \oplus Y &= \sum m(1,2) \\ &= X.\bar{Y} + \bar{X}.Y \ . \end{aligned}$$

This leads us to the hardware model in figure 4.

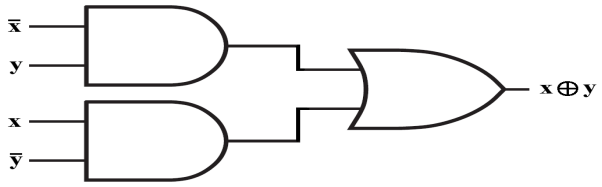

Figure 4: AND-OR representation of the XOR instruction

The maxterm expression (standard product of sums) is given by

$$\begin{aligned} X \oplus Y &= \prod M(0,3) \\ &= (X+Y).(\bar{X}+\bar{Y}) \ . \end{aligned}$$

Figure 1 shows the matching hardware model.

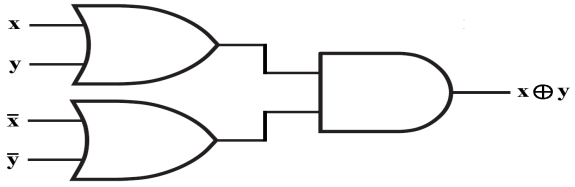

Figure 5: OR-AND representation of the XOR instruction

For the XOR function, the sum-of-products and product-of-sums expressions are equally simple. As all logic circuits can be constructed using only NOR gates or only NAND gates, we will also transform the above circuits to these forms. The sum-of-products expression was realized as a second-order AND-OR form which is equivalent to the two-level NAND circuit of figure 6.

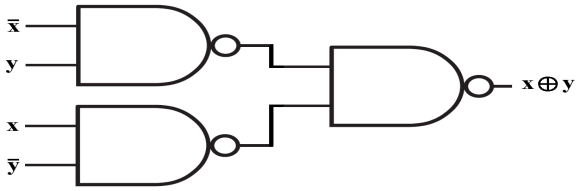

Figure 6: Two-level NAND equivalent circuit to the second-order AND-OR circuit

The product-of-sums expression on the other hand was realized as a second-order OR-AND form which is equivalent to the two-level NOR circuit of figure 7.

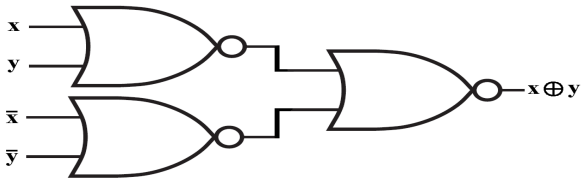

Figure 7: Two-level NOR equivalent circuit to the second-order OR-AND circuit

Finally we have to get rid of the negated inputs to our gates. Instead of putting inverters on the input, we can also choose to design the whole circuit with a single type of gate.

5.3.2 NAND representations

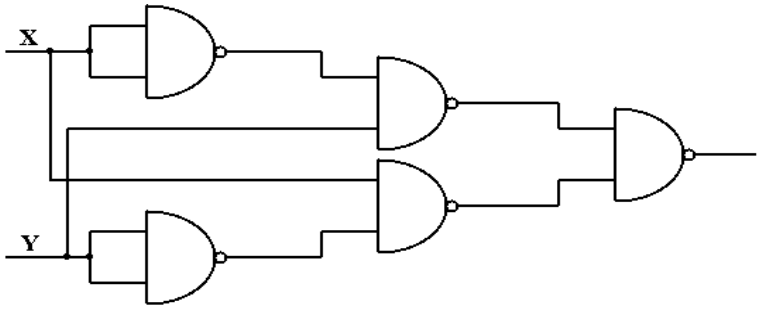

Figure 8: Exclusive-OR implementation with NAND gates

Proof
The first logic level negates the inputs as $\overline{a.a} = \bar{a}$. The second computes

$$\begin{aligned} \overline{\bar{x}.y} &= x + \bar{y} \ , \\ \overline{x.\bar{y}} &= \bar{x} + y \ . \end{aligned}$$

Finally the third level outputs

$$\begin{aligned} \overline{(x+\bar{y})(\bar{x}+y)} &= \overline{(x+\bar{y})} + \overline{(\bar{x}+y)} \\ &= \bar{x}.y + x.\bar{y} \\ &= x \oplus y \ : \text{ minterm expression} \ . \end{aligned}$$

5.3.3 NOR representations

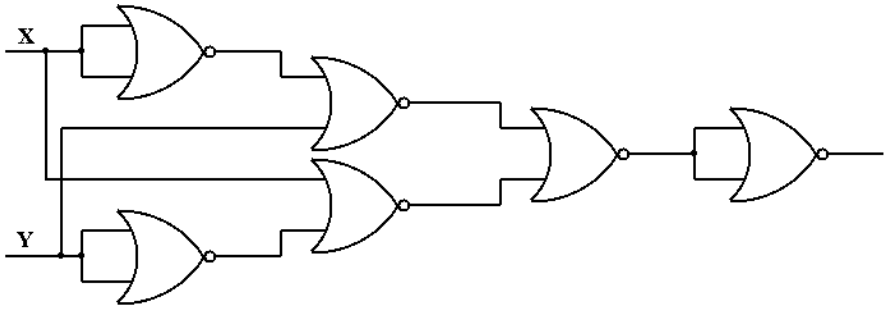

Figure 9: Exclusive-OR implementation with NOR gates

Proof
In the first level the inputs are negated: $\overline{a+a} = \bar{a}$. The second one computes

$$\begin{aligned} \overline{\bar{x}+y} &= x.\bar{y} \ , \\ \overline{x+\bar{y}} &= \bar{x}.y \ . \end{aligned}$$

The outcome of the third level is

$$\begin{aligned} \overline{x.\bar{y}+\bar{x}.y} &= \overline{(x.\bar{y})}.\overline{(\bar{x}.y)} \\ &= (\bar{x}+y).(x+\bar{y}) \ . \end{aligned}$$

This is negated by the last NOR gate:

$$\begin{aligned} \overline{(\bar{x}+y).(x+\bar{y})} &= \overline{(\bar{x}+y)} + \overline{(x+\bar{y})} \\ &= \bar{x}.y + x.\bar{y} \\ &= x \oplus y \ : \text{ minterm expression} \ . \end{aligned}$$

6 Transistor level implementation of basic CMOS switching functions using SPICE

6.1 The NAND gate

In SPICE we implemented the NAND structure of figure 10. This only uses 4 transistors: 2 PMOS and 2 NMOS transistors. To make the simulation more realistic we also implemented a capacitor (10 femtofarad) between the output node and V_{ss}.

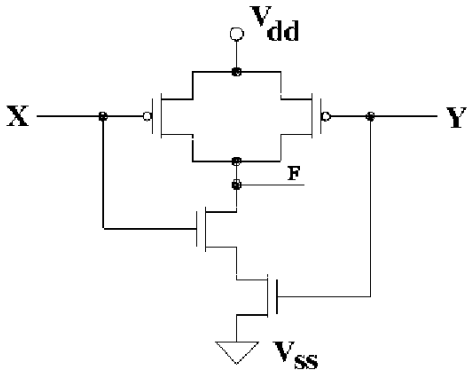

Figure 10: Complete transistor circuit for realizing $F(X,Y) = \overline{X.Y}$

When feeding a certain input to a gate, one has to make sure that the gate has enough time to cope with the transitions in its input. Some gates can react faster to changes in the input than others. It is therefore advised to do little tests on the gates and circuits: play along with the rise and fall times of the input signals and see how the gate/circuit responds. If the gate was correctly implemented but gives a wrong output for certain inputs, then the rise and fall times of the input signals should probably be raised.
A NAND gate is a very fast gate: it functions perfectly well on inputs whos rise and fall times take only 1 picosecond! Still, we will let all input waveforms have rise and fall times of 200 picoseconds. This is because our goal is rather to investigate connections of gates (as the XOR) and connections inherently induce delays: each level of logic adds to the delay in the development of a signal at the circuit output. So when we arrive at investigating the XOR circuit implemented only using NAND gates, we will have to take inputs with rise and fall times that are larger than 1 picosecond because otherwise faults will appear at outputs of certain intermediate leveled gates. These faults are carried through the next gates, causing the XOR function to be incorrect!
By simulations, we found that rise and fall times of 200 picoseconds for the input waveforms are sufficiently large to get a correctly functioning full-NAND XOR circuit. That is why we will also use such input waveforms when looking at just one NAND gate; in this manner we can investigate the power consumption characteristics per gate of the XOR circuit.

transition #	Hamming distance	possible transitions
1	-2	$11 \rightarrow 00$
2	-1	$11 \rightarrow 10$
3		$11 \rightarrow 01$
4		$10 \rightarrow 00$
5		$01 \rightarrow 00$
6	0	$10 \rightarrow 01$
7		$01 \rightarrow 10$
8	1	$00 \rightarrow 01$
9		$00 \rightarrow 10$
10		$01 \rightarrow 11$
11		$10 \rightarrow 11$
12	2	$00 \rightarrow 11$

.

The input sequences displayed in our figures have a period of 13 bits (as we want to cover all twelve transitions):

$$\begin{aligned} X &= 0011010110100 \\ Y &= 0101001100110\ . \end{aligned}$$

The resulting power patterns found in the fourth panel of the figures are thus caused by the following transitions:

$$\begin{array}{ccccccccccccccccccccc} & & 8 & & 7 & & 11 & & 1 & & 9 & & 6 & & 10 & & 2 & & \\ 00 & \rightarrow & 01 & \rightarrow & 10 & \rightarrow & 11 & \rightarrow & 00 & \rightarrow & 10 & \rightarrow & 01 & \rightarrow & 11 & \rightarrow & 10 \end{array}$$

$$\begin{array}{ccccccccc} 4 & & 12 & & 3 & & 5 & & \\ \rightarrow & 00 & \rightarrow & 11 & \rightarrow & 01 & \rightarrow & 00\ . \end{array}$$

It is also important how the input sequences are fed to the circuits. We can for instance define the input sequences as piecewise linear functions. This case is shown in figure 11, in which the first two panels graph X and Y respectively. This is however not a very realistic setup; in reality signals arrive at gates as curves instead of straight lines. In a second setup we therefore implement an inverter before the gate under consideration and feed this inverter with the (inverted) input sequences $\bar{X}$ and $\bar{Y}$, resulting again in sequences X and Y as inputs to the gate buth now these inputs are curved. Figure 12 shows these curved inputs. Both figures also display the output of the gate (third panel) and the power consumption of the gate. It is important to emphasize that the displayed power consumption is only due to the functioning of the NAND gate. For this purpose we used two sources Vdd in the second setup: one connected to the inverter and the other to the gate. This makes it possible to concentrate only on the power dissipated by the NAND gate.

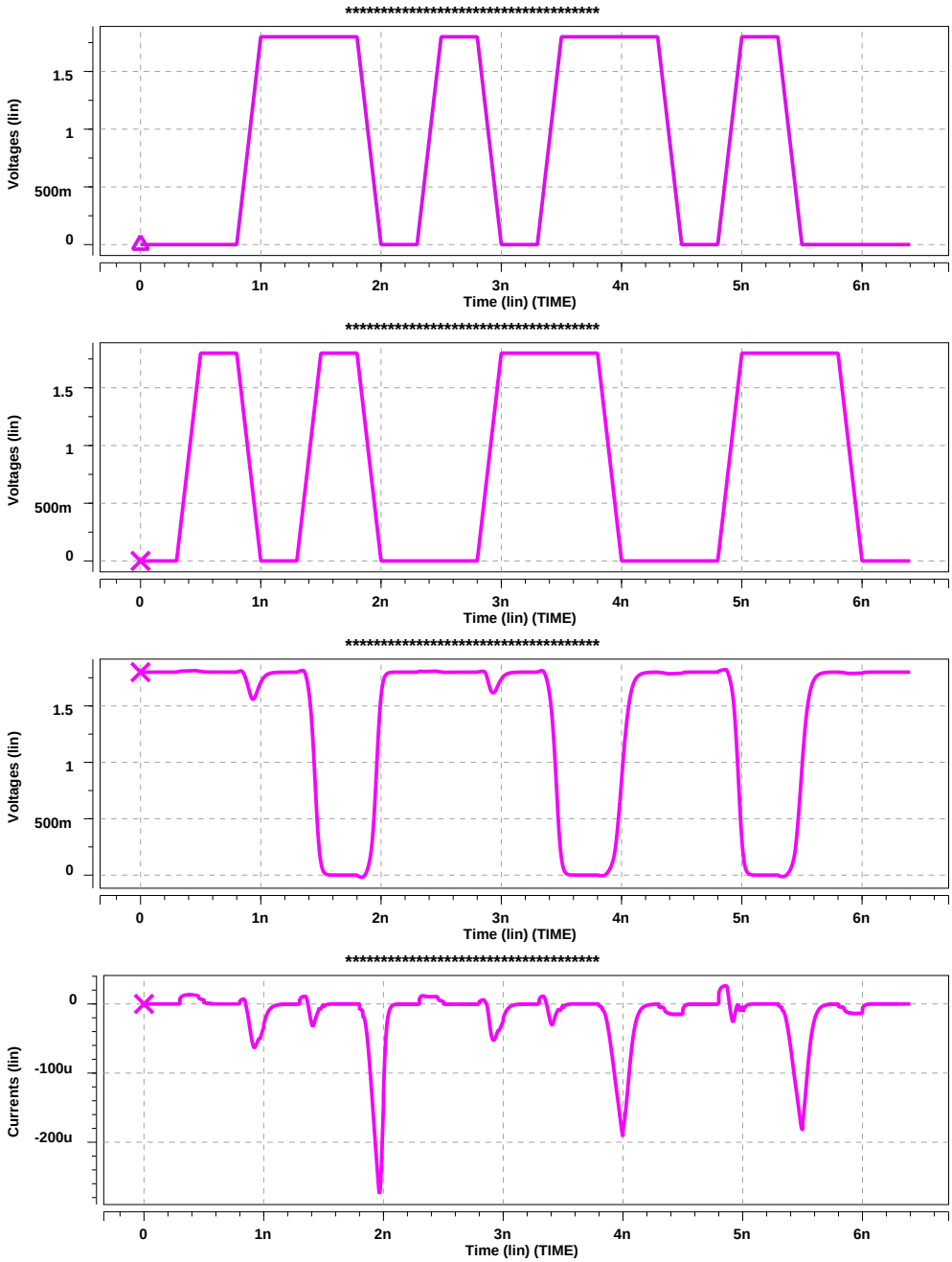

Figure 11: Ideal inputs X and Y representing all twelve transitions, output $F(X,Y) = \overline{X.Y}$ and simulated power consumption of the NAND gate

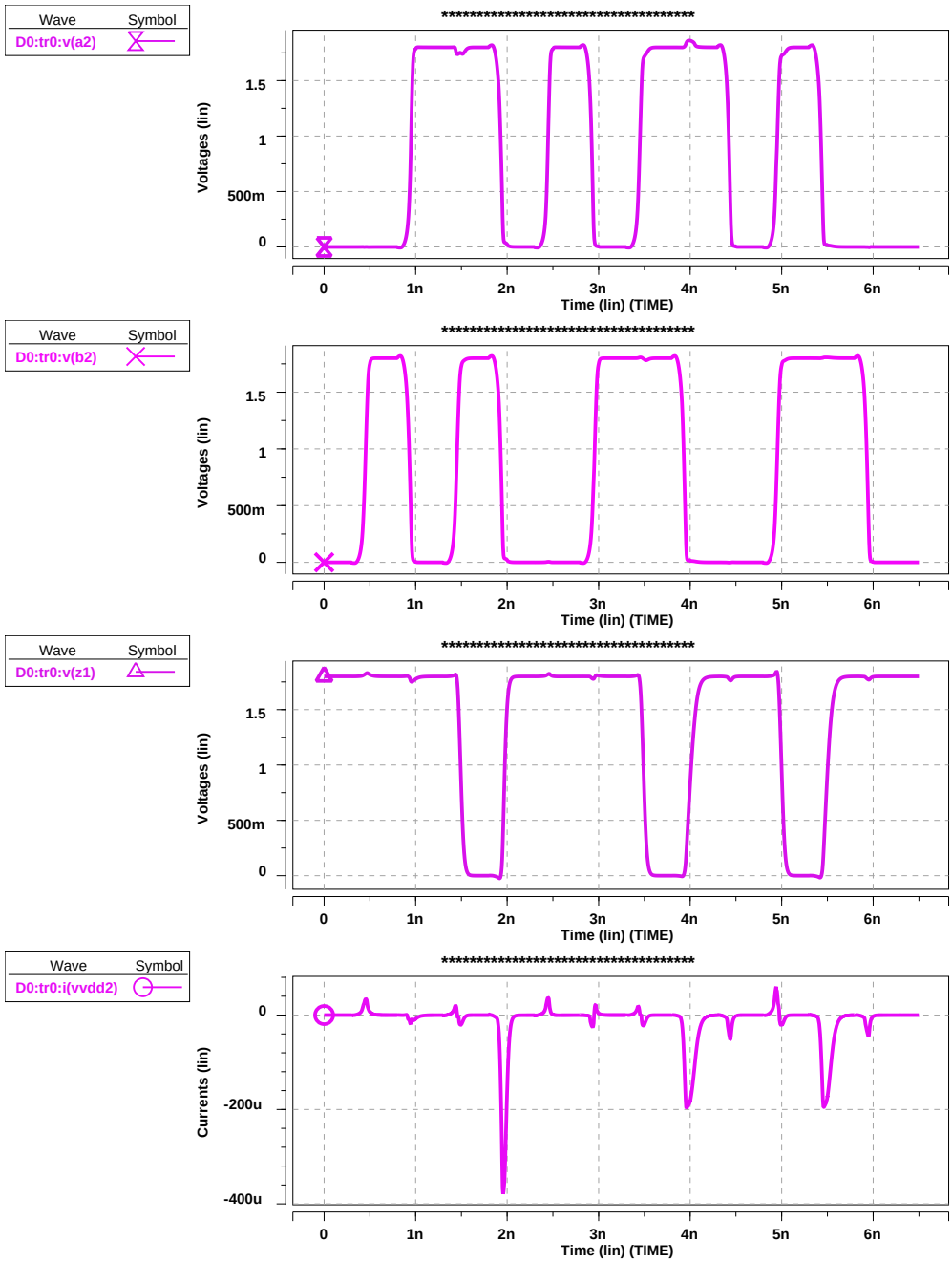

Figure 12: Real inputs X and Y representing all twelve transitions, output $F(X,Y) = \overline{X.Y}$ and simulated power consumption of the NAND gate

6.1.1 Analysis of the simulation with ideal inputs

In figure 11, 8 transitions in both inputs are visible:

- for input A at intervals

$$[800, 1000] \bigcup [1800, 2000] \bigcup [2300, 2500] \bigcup [2800, 3000] \bigcup [3300, 3500] \bigcup [4300, 4500] \bigcup [4800, 5000] \bigcup [5300, 5500] \; ; \tag{1}$$

- for input B at intervals

$$[300, 500] \bigcup [800, 1000] \bigcup [1300, 1500] \bigcup [1800, 2000] \bigcup [2800, 3000] \bigcup [3800, 4000] \bigcup [4800, 5000] \bigcup [5800, 6000] \; . \tag{2}$$

Consequently, 6 transitions occur at the output. If we now consider the resulting powertrace, we find 12 patterns in the union of transition intervals of the inputs. Table 1 gives an overview.

interval	transition at input	transition type	transition at output	transition type	pattern
[300, 500]	B	$0 \to 1$	no		positive bump
[800, 1000]	A	$0 \to 1$	no		
	B	$1 \to 0$			negative peak I
[1300, 1500]	B	$0 \to 1$	yes	$1 \to 0$	negative peak II
[1800, 2000]	A	$1 \to 0$	yes	$0 \to 1$	
	B	$1 \to 0$			negative peak III
[2300, 2500]	A	$0 \to 1$	no		positive bump
[2800, 3000]	A	$1 \to 0$	no		
	B	$0 \to 1$			negative peak I
[3300, 3500]	A	$0 \to 1$	yes	$1 \to 0$	negative peak II
[3800, 4000]	B	$1 \to 0$	yes	$0 \to 1$	negative peak IV
[4300, 4500]	A	$1 \to 0$	no		negative bump
[4800, 5000]	A	$0 \to 1$	yes	$1 \to 0$	
	B	$0 \to 1$			W-shape
[5300, 5500]	A	$1 \to 0$	yes	$0 \to 1$	negative peak IV
[5800, 6000]	B	$1 \to 0$	no		negative bump

Table 1: Patterns in the powertrace caused by transitions of in- and outputs to a NAND gate

From table 1 it is obvious that only 7 distinct patterns occur. Power analysis attacks use the powertraces to infer information on the underlying calculation and operands. Therefore we reverse the analysis: *what information is revealed by the various patterns*?

Positive bump: one of the inputs flips from 0 to 1 while the other stays constant. The output does not change.

Input combination	output transition
$00 \rightarrow 01$	no (out = 1)
$00 \rightarrow 10$	

Negative peak I: the inputs make an opposite transition while the output remains.

Input combination	output transition
$01 \rightarrow 10$	no (out = 1)
$10 \rightarrow 01$	

Negative peak II: one of the inputs flips from 0 to 1 causing a transition of the output from high to low.

Input combination	output transition
$01 \rightarrow 11$	$1 \rightarrow 0$
$10 \rightarrow 11$	

Negative peak III: both inputs go from high to low (resulting in the opposite transition at the output).

Input combination	output transition
$11 \rightarrow 00$	$0 \rightarrow 1$

Negative peak IV: one of the inputs flips from 1 to 0 causing a transition of the output from low to high.

Input combination	output transition
$11 \rightarrow 01$	$0 \rightarrow 1$
$11 \rightarrow 10$	

Negative bump: one of the inputs flips from 1 to 0 while the other stays constant. The output does not change.

Input combination	output transition
$01 \rightarrow 00$	no (out = 1)
$10 \rightarrow 00$	

W-shape: both inputs go from low to high (resulting in the opposite transition at the output).

Input combination	output transition
$00 \rightarrow 11$	$1 \rightarrow 0$

Summarizing: given a NAND gate and the powertrace of the gate processing a bivariate input sequence, we can always derive the output value and the Hamming weight of the inputs in transition phases.

6.1.2 Analysis of the simulation with real inputs

The simulation setup made sure that all transitions occured at the same intervals (1) and (2) and were of the same type. Again 12 patterns show up at the same time periods in the powertrace. Table 2 compares the patterns from the realistic setup with those from the ideal setup.

interval	pattern when using ideal inputs	pattern when using real inputs
[300, 500]	positive bump	positive peak
[800, 1000]	negative peak I	W-shape
[1300, 1500]	negative peak II	pos/neg peak
[1800, 2000]	negative peak III	negative peak I
[2300, 2500]	positive bump	positive peak
[2800, 3000]	negative peak I	neg/pos peak
[3300, 3500]	negative peak II	pos/neg peak
[3800, 4000]	negative peak IV	negative peak II
[4300, 4500]	negative bump	negative peak III
[4800, 5000]	W-shape	pos/neg peak II
[5300, 5500]	negative peak IV	negative peak II
[5800, 6000]	negative bump	negative peak III

Table 2: Patterns in the powertrace caused by transitions of in- and outputs to a NAND gate

A first dissimilarity is that bumps are replaced by peaks. Some new patterns appear where small positive and negative peaks are combined. The most important change is formed by the fact that when the inputs make opposite transitions, we can make a distinction between the two possibilities (interval $[800, 1000]$ versus $[2800, 3000]$).

6.2 The NOR gate

Next we implemented the NOR structure of figure 13 also with an added capacitor.

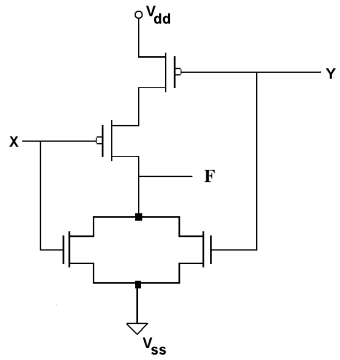

Figure 13: Complete transistor circuit for realizing $F(X,Y) = \overline{X+Y}$

6.2.1 Analysis of the simulations

We fed the same *ideal* and *realistic* inputs to the NOR circuit as were fed to the NAND circuit. Table 3 gives an overview of the resulting patterns in the powertrace.

interval	transition at input	transition type	transition at output	transition type	pattern
[300, 500]	B	$0 \rightarrow 1$	no		W-shape
[800, 1000]	A	$0 \rightarrow 1$	no		
	B	$1 \rightarrow 0$			negative peak I
[1300, 1500]	B	$0 \rightarrow 1$	yes	$1 \rightarrow 0$	positive bump
[1800, 2000]	A	$1 \rightarrow 0$	yes	$0 \rightarrow 1$	
	B	$1 \rightarrow 0$			negative peak II
[2300, 2500]	A	$0 \rightarrow 1$	no		W-shape
[2800, 3000]	A	$1 \rightarrow 0$	no		
	B	$0 \rightarrow 1$			pos/neg peak
[3300, 3500]	A	$0 \rightarrow 1$	yes	$1 \rightarrow 0$	tiny positive bump
[3800, 4000]	B	$1 \rightarrow 0$	yes	$0 \rightarrow 1$	negative peak III
[4300, 4500]	A	$1 \rightarrow 0$	no		negative peak II
[4800, 5000]	A	$0 \rightarrow 1$	yes	$1 \rightarrow 0$	
	B	$0 \rightarrow 1$			positive peak
[5300, 5500]	A	$1 \rightarrow 0$	yes	$0 \rightarrow 1$	tiny negative bump
[5800, 6000]	B	$1 \rightarrow 0$	no		negative peak II

Table 3: Patterns in the powertrace caused by transitions of in- and outputs to a NOR gate

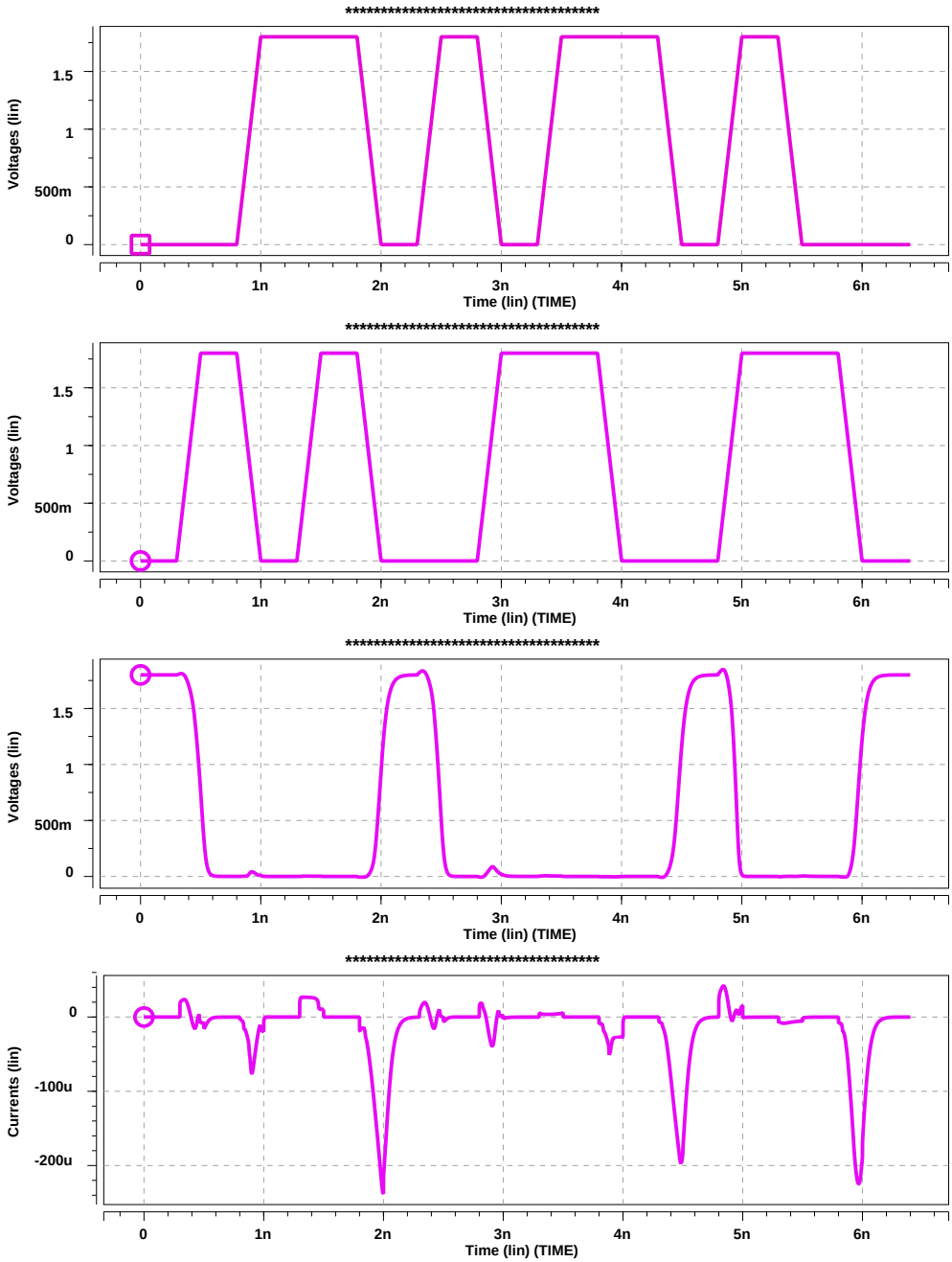

Figure 14: Inputs X and Y representing all twelve transitions, output $F(X,Y) = \overline{X+Y}$ and simulated power consumption

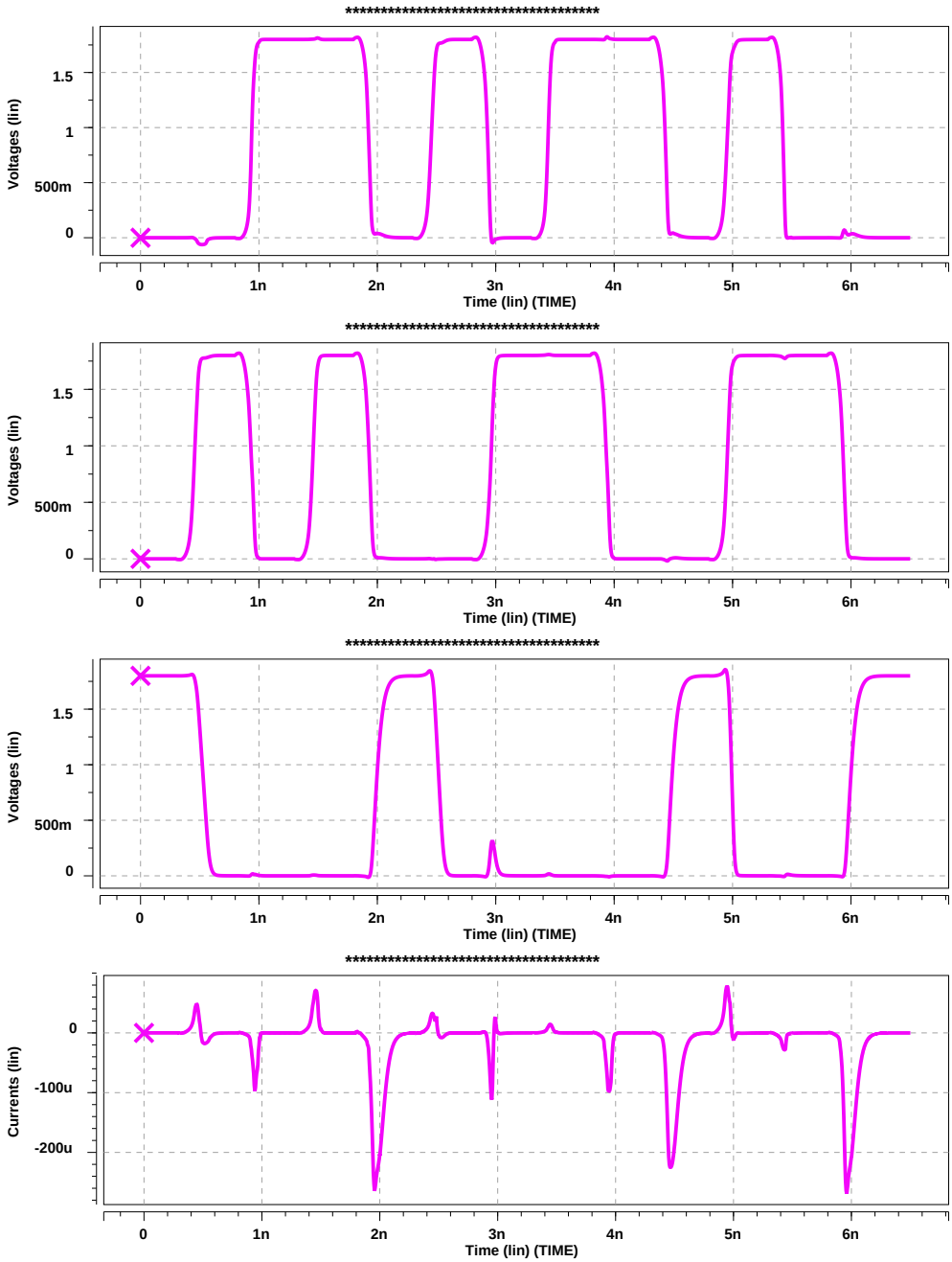

Figure 15: Inputs X and Y representing all twelve transitions, output $F(X,Y) = \overline{X+Y}$ and simulated power consumption

7 Identification of the NAND gate

Identification routines:

1. The command *impulse(data)* estimates a high order, noncausal FIR model after having prefiltered the data so that the input is *as white as possible*. The impulse response of this FIR model and its confidence region is then plotted. We hence get an idea of the delay and of the order of the multivariate model.

2. We start with multivariate ARX models. We generate a model structure matrix $NN = [na\ nb\ nk]$ which contains several combinations of the orders and delays given in row vectors na, nb and nk. The commands *arxstruc* and *selstruc* help us in choosing the appropriate orders and delay.

3. We estimate the multivariate arx models using the orders and delays suggested by BEST, AIC and MDL.

4. Validation of the models is done by using *compare* and *resid* on both the estimation data and the validation data.

7.1 Nand with ideal inputs

Using the command *impulse* returns a FIR model with orders $na = 0$, $nb = [35\ 35]$ and a delay $nk = [0\ 0]$. We only use this estimate to get an upper bound on nb and the delay values. For instance, the model structure matrix in this case is the following matrix:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 \\ \vdots & & & & \vdots \\ 35 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 3 & 2 & 2 & 0 & 0 \\ \vdots & & & & \vdots \\ 35 & 2 & 2 & 0 & 0 \\ \vdots & & & & \vdots \\ 35 & 35 & 35 & 0 & 0 \end{pmatrix}.$$

From these 630 possible structures we get the following suggestions:

- AIC: [42 34 34 0 0] ;
- MDL: [17 17 17 0 0] .

We hence build multivariate ARX models using these suggestions and validated the resulting models.

ARX model of order 42**:**

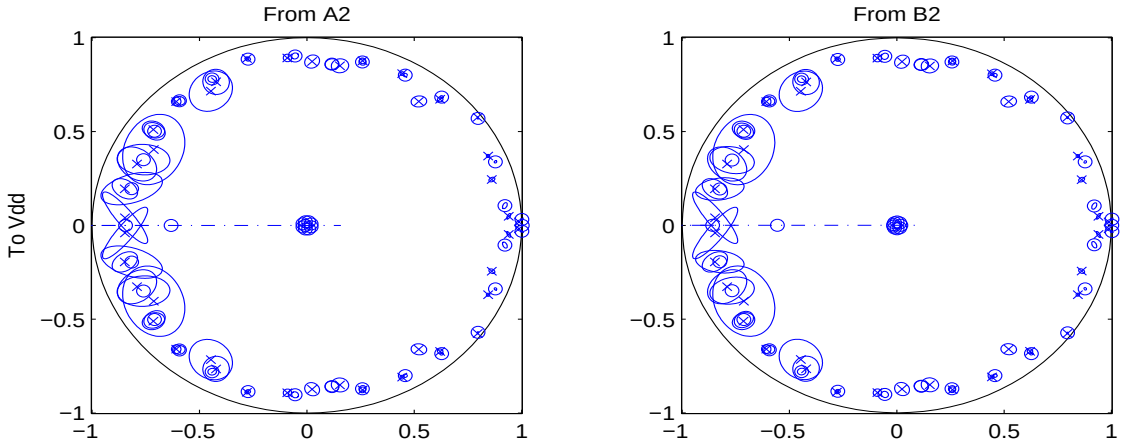

Figure 16: Pole-zero map of ARX42 with 1σ confidence region.

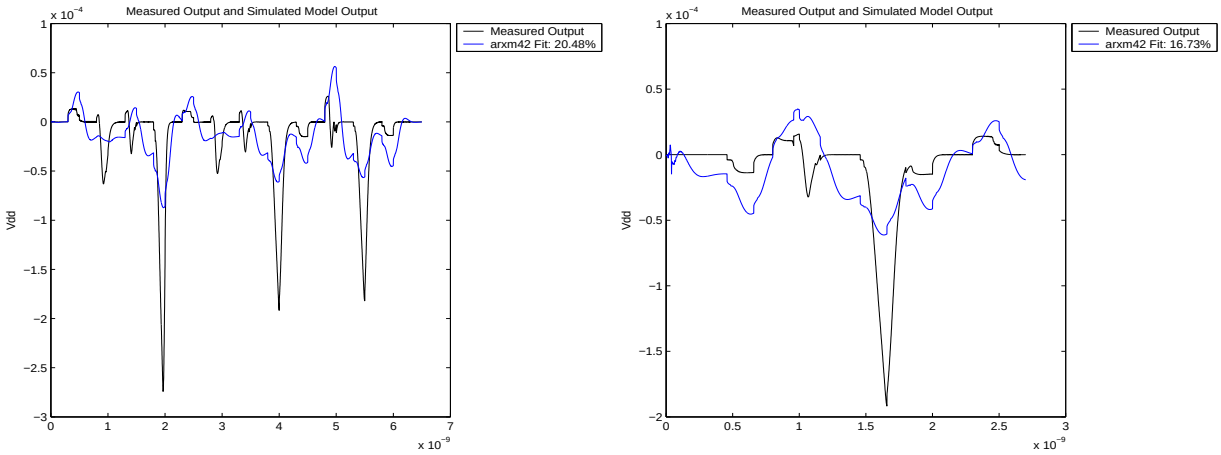

Figure 17: Simulation errors on estimation set (left) and validation set (right)

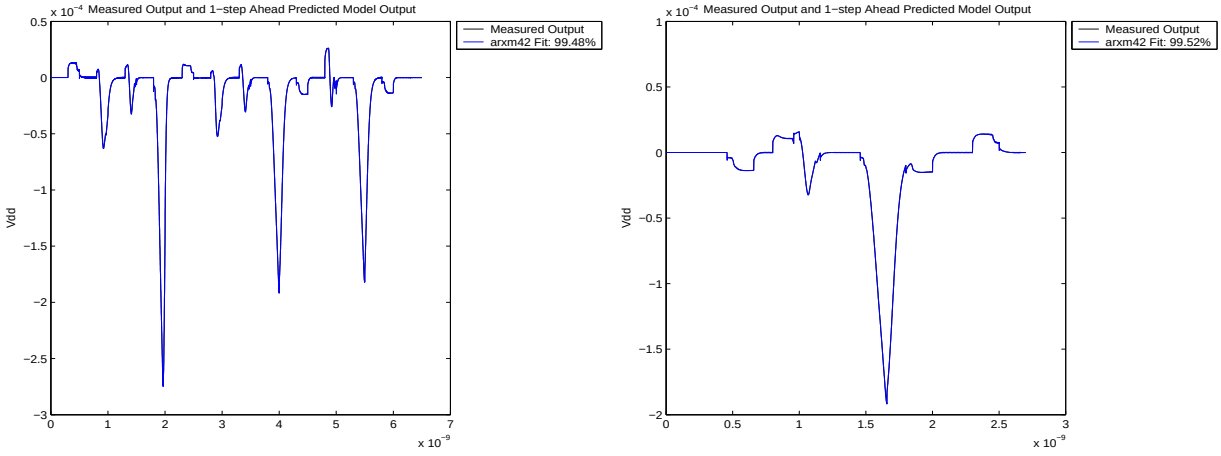

Figure 18: Prediction errors on estimation set (left) and validation set (right)

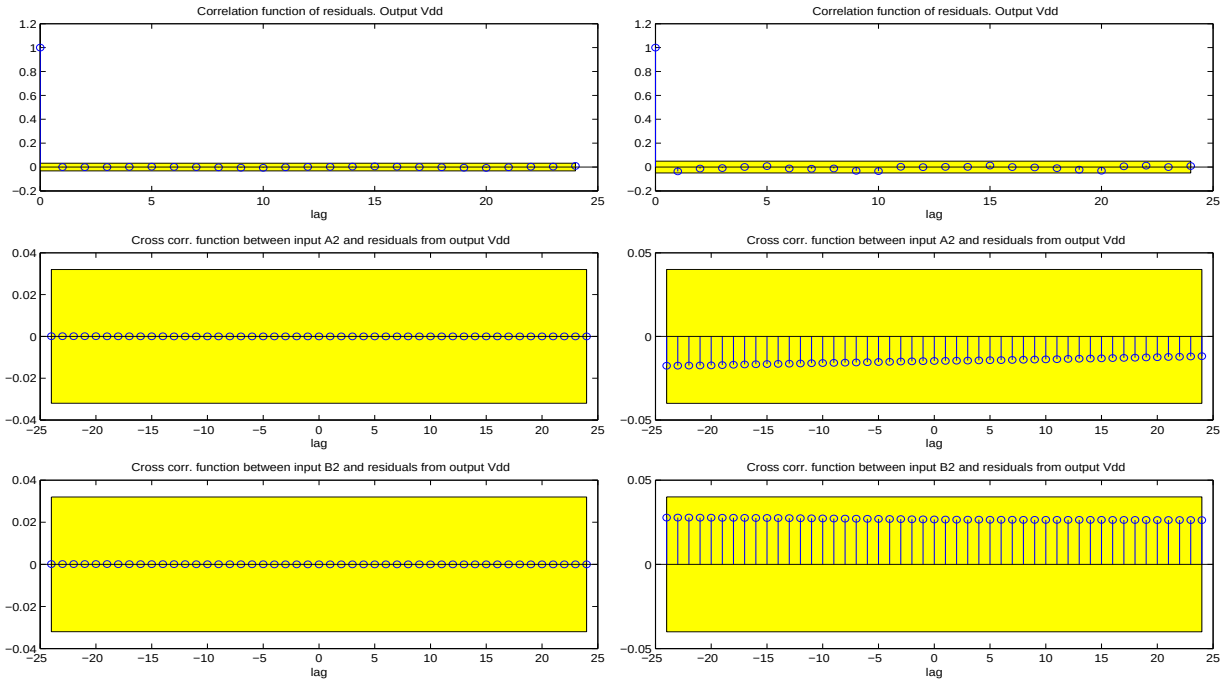

Figure 19: Residual analysis on estimation set (left) and validation set (right)

ARX model of order 17**:**

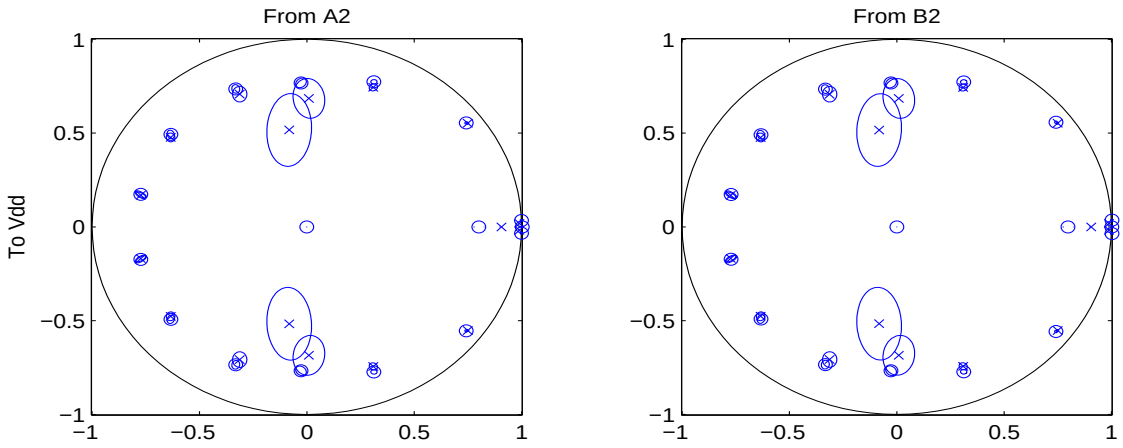

Figure 20: Pole-zero map of ARX17 with 1σ confidence region

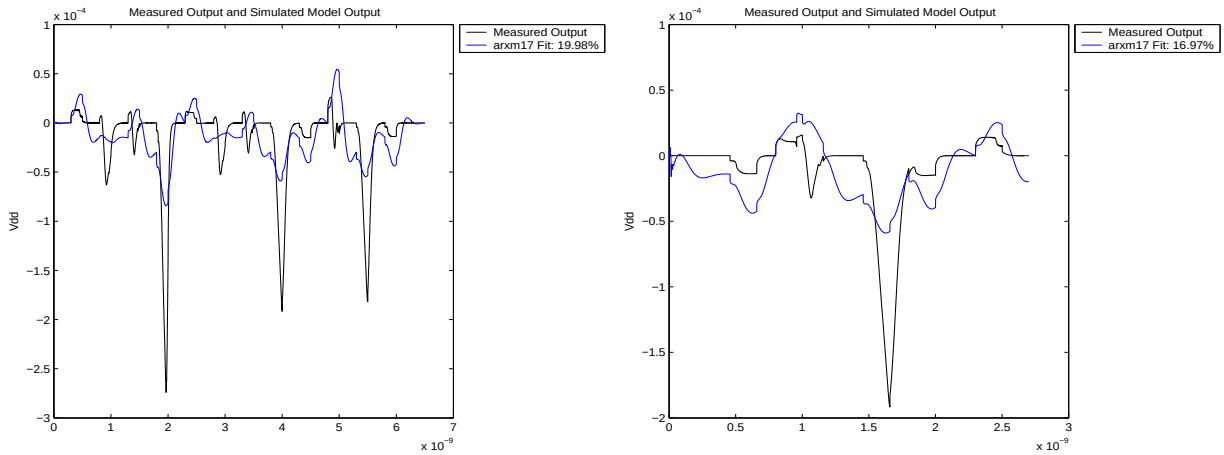

Figure 21: Simulation errors on estimation set (left) and validation set (right)

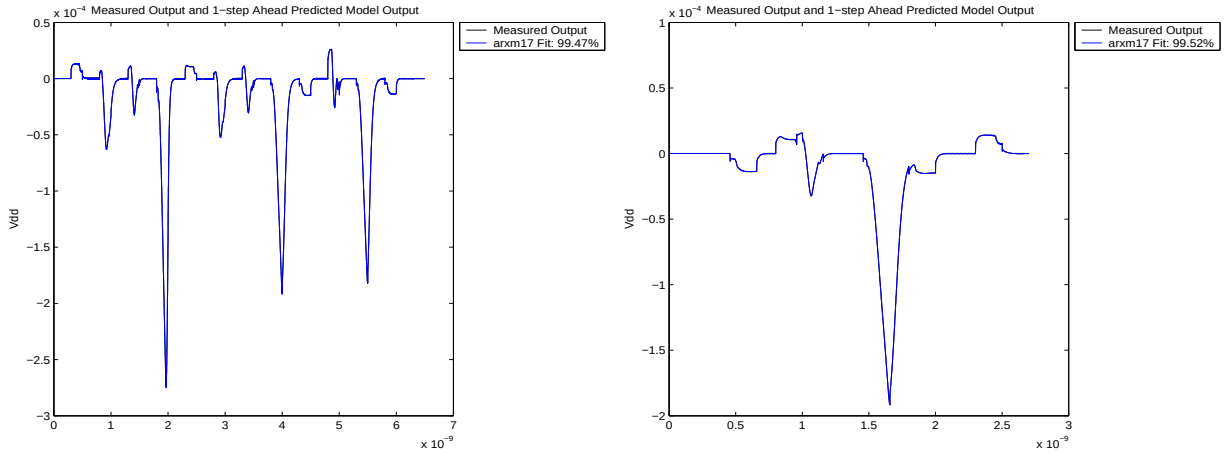

Figure 22: Prediction errors on estimation set (left) and validation set (right)

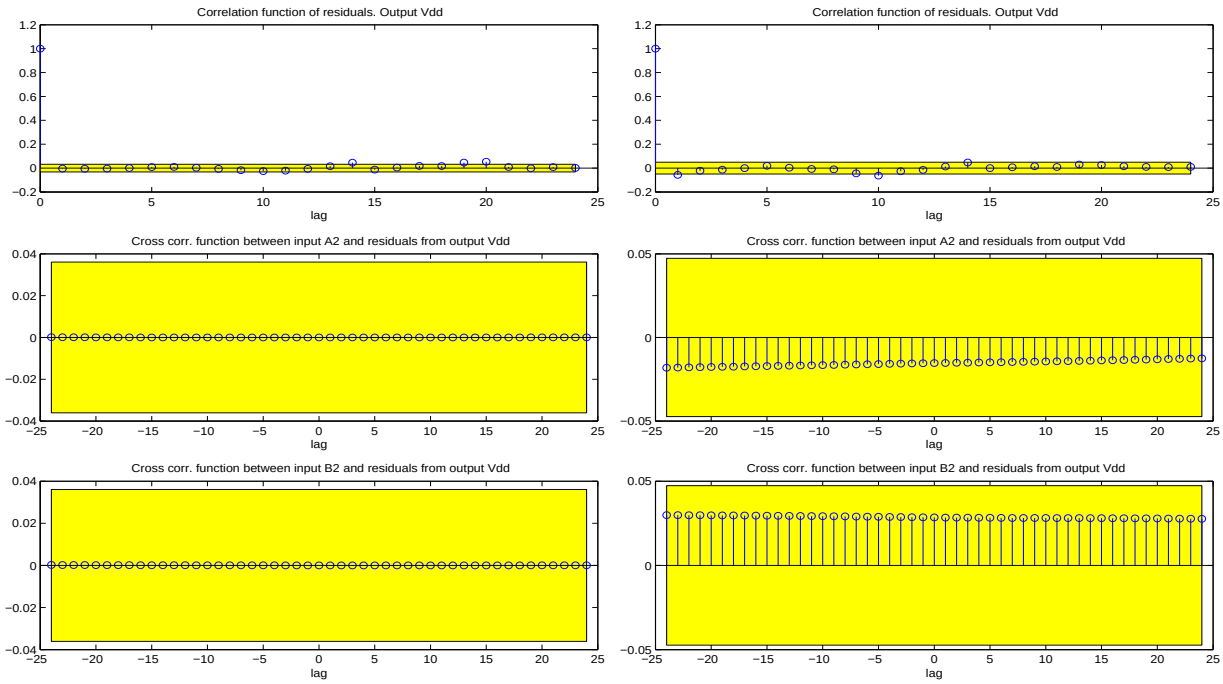

Figure 23: Residual analysis on estimation set (left) and validation set (right)

Prediction by means of the ARX models:

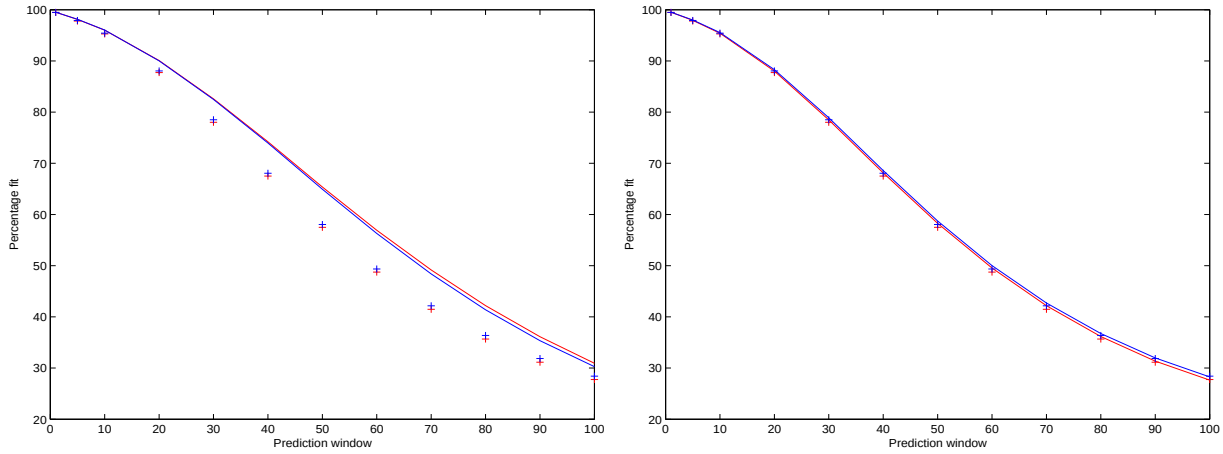

Figure 24: Prediction fit values with respect to the prediction horizon for arx17 (red) and arx42 (blue) using the identification set (solid lines) and the validation set ($'+'$)

Some ARMAX models: We next tried out several ARMAX models; these however do not seem appropriate on our system. Table 4 lists some examples.

OE models: these even perform worse than the ARMAX models. For instance: an OE model with parameters $[nb\ nb\ nf\ nf\ nk\ nk] = [35\ 35\ 42\ 42\ 0\ 0]$ gives the results displayed

orders $[na\ nb\ nb\ nc\ nk\ nk]$	simulation error on ident. set	simulation error on valid. set	prediction error on ident. set	prediction error on valid. set
[41 34 34 1 0 0]	11.68	-0.02	99.45	99.49
[41 33 33 1 0 0]	11.47	-0.65	99.45	99.49
[40 34 34 1 0 0]	11.24	-0.96	99.45	99.49
[39 34 34 1 0 0]	11.11	-1.45	99.45	99.49
[39 33 33 1 0 0]	10.94	-1.68	99.44	99.49
[14 16 16 1 0 0]	5.045	-11.95	99.41	99.45
[14 15 15 1 0 0]	12.16	0.47	99.42	99.47
[14 14 14 1 0 0]	10.03	-5.35	99.42	99.46

Table 4: ARMAX models with $nc = 1$

in the figures below.

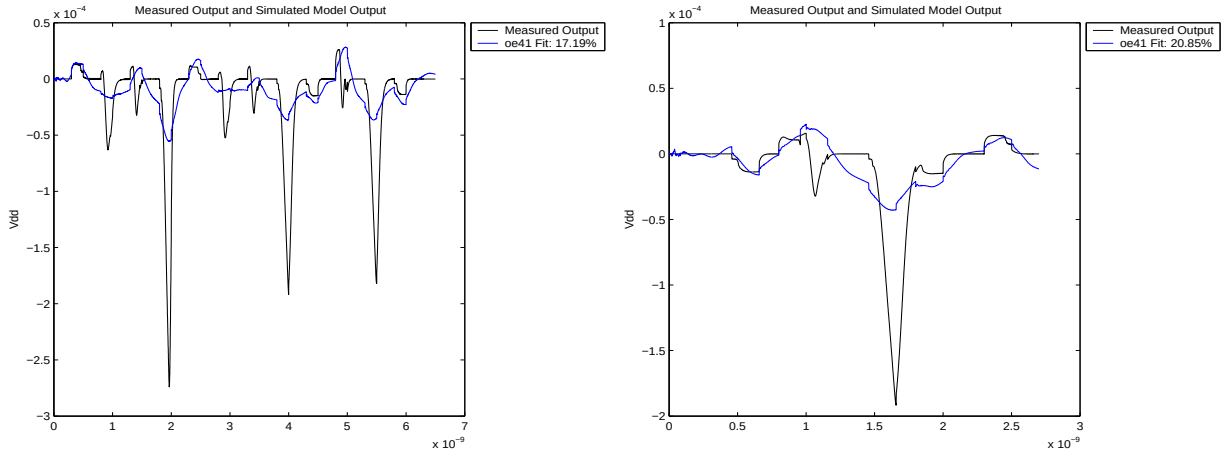

Figure 25: Simulation errors on estimation set (left) and validation set (right)

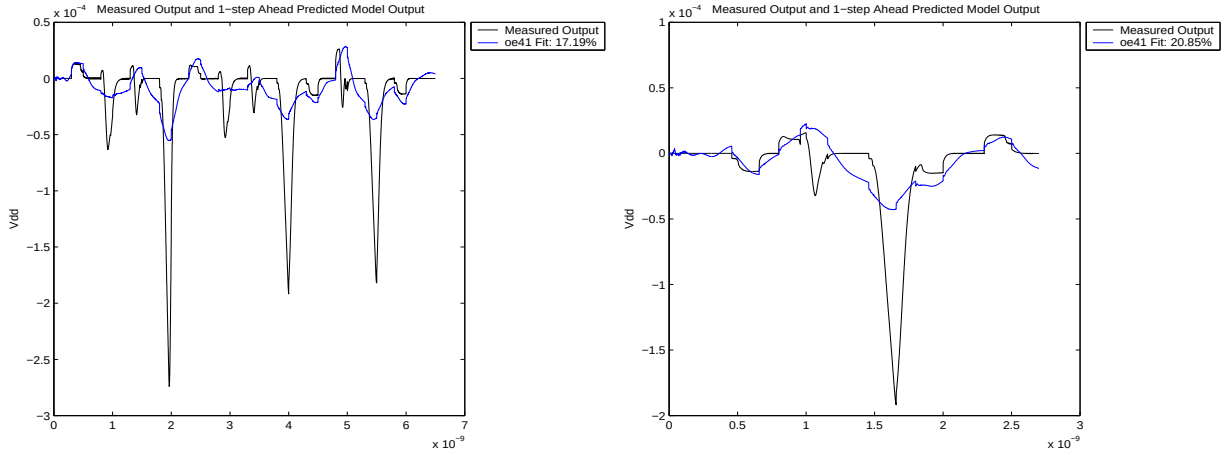

Figure 26: Prediction errors on estimation set (left) and validation set (right)

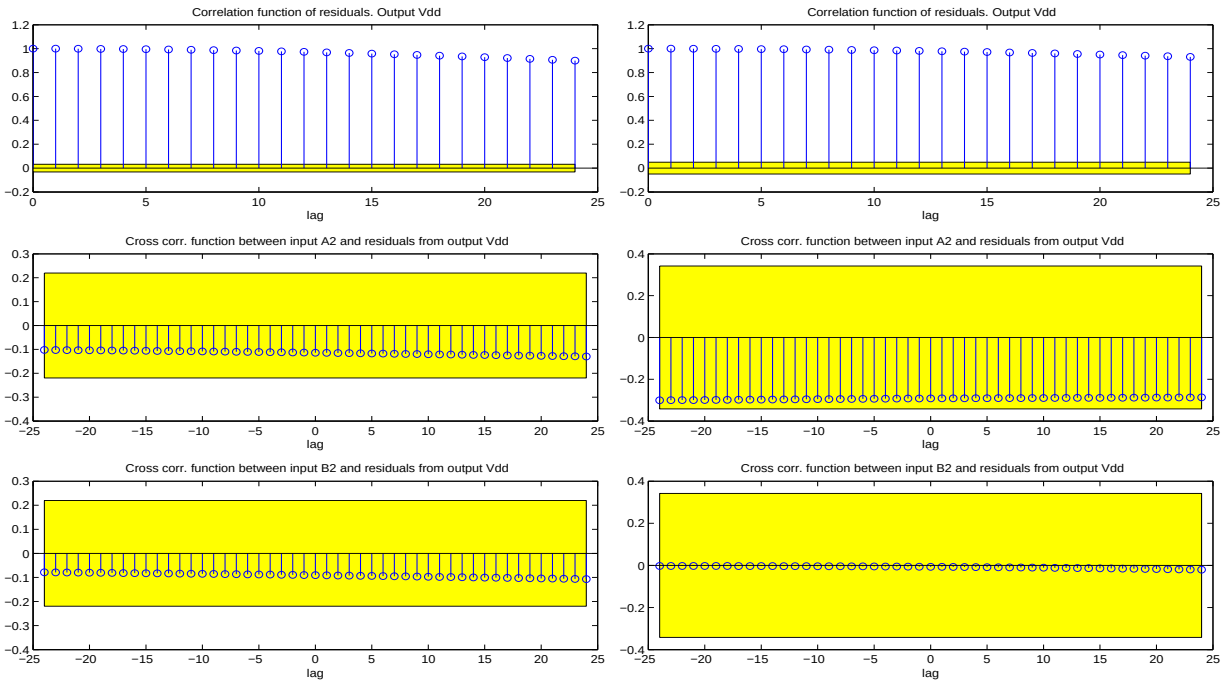

Figure 27: Residual analysis on estimation set (left) and validation set (right)

Box-Jenkins models: the first idea was to overestimate a BJ model and then use balanced reduction techniques. The overestimated systems however, turned out to be unstable and hence these techniques could not be applied. By means of trial-and-error we tested the model structures listed in table 5.

orders $[nb\ nb\ nc\ nd\ nf\ nf\ nk\ nk]$	simulation error on ident. set	simulation error on valid. set	prediction error on ident. set	prediction error on valid. set
[3 3 5 5 5 5 0 0]	17.24	12.62	99.41	99.46
[3 3 4 4 5 5 0 0]	17.10	12.53	99.41	99.46
[3 3 4 4 4 4 0 0]	17.22	12.62	99.41	99.45
[3 3 3 3 4 4 0 0]	15.59	2.91	99.38	99.43

Table 5: BJ models

Model structures [2 2 4 4 4 4 0 0], [3 3 5 5 4 4 0 0], [3 3 5 5 3 3 0 0] and [3 3 4 4 3 3 0 0] gave worse validation results.

8 Acknowledgements

Evelyne Dewitte is a research assistant with the I.W.T. (Flemish Institute for Scientific and Technological Research in Industry). Dr. Bart De Moor and Dr. Bart Preneel are full professors at the Katholieke Universiteit Leuven, Belgium. Research supported by Research Council KUL: GOA-Mefisto 666, several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects, G.0240.99 (multilinear algebra), G.0407.02 (support vector machines), G.0197.02 (power islands), G.0141.03 (identification and cryptography), G.0491.03 (control for intensive care glycemia), G.0120.03 (QIT), research communities (ICCoS, ANMMM); AWI: Bil. Int. Collaboration Hungary/ Poland; IWT: PhD Grants, Soft4s (softsensors), Belgian Federal Government: DWTC (IUAP IV-02 (1996-2001) and IUAP V-22 (2002-2006)), PODO-II (CP/40: TMS and Sustainability); EU: CAGE; ERNSI; Eureka 2063-IMPACT; Eureka 2419-FliTE; Contract Research/agreements: Data4s, Electrabel, Elia, LMS, IPCOS, VIB.

References

[1] D. Agrawal, B. Archambeault, J. R. Rao and P. Rohatgi, *The EM Side-Channel(s)*, Cryptographic Hardware and Embedded Systems, LNCS 2523, B. S. Kaliski Jr., C. K. Koc and Christof Paar, Eds., Springer-Verlag, 2002, pp 29.

[2] D. Agrawal, J. R. Rao and P. Rohatgi, *Multi-channel Attacks*, Cryptographic Hardware and Embedded Systems, LNCS 2779, C. D. Walter, C.K. Koc and C. Paar, Eds., Springer-Verlag, 2003, pp 2.

[3] S. Berna Ors, E. Oswald and B. Preneel, *Power-Analysis Attacks on an FPGA: First Experimental Results*, Cryptographic Hardware and Embedded Systems, LNCS 2779, C. D. Walter, C.K. Koc and C. Paar, Eds., Springer-Verlag, 2003, pp 35.

[4] C. Clavier, J.-S. Coron, N. Dabbous, *Differential power analysis in the presence of hardware countermeasures*, Cryptographic Hardware and Embedded Systems, LNCS 1965, C.K. Koc and C. Paar, Eds., Springer-Verlag, 2000, pp. 252-263.

[5] J.S. Coron, P. Kocher, D. Naccache, *Statistics and Secret Leakage*, in the proceedings of Financial Crypto 2000, Springer LNCS, vol 1972, pp 157-173.

[6] J. Daemen, V. Rijmen, *Resistance against implementation attacks: a comparative study of the AES proposals*, Proceedings of the 2nd AES Candidate Conference, 1999, pp. 122-132.

[7] L. De Lathauwer, B. De Moor, *From Matrix to Tensor : Multilinear Algebra and Signal Processing*, in Mathematics in Signal Processing IV , (McWhirter J., ed.), Selected papers presented at 4th IMA Int. Conf. on Mathematics in Signal Processing, Oxford University Press (Oxford, United Kingdom), 1998, pp. 1-15.

[8] B. De Moor, *An introduction to system identification*, in 1988 Integrated European Course in Mechatronics, Leuven, Belgium, May 30- Jun. 3, 1988.

[9] K. Gandolfi, C. Mourtel, F. Olivier, *Electromagnetic attacks: concrete results*, Cryptographic Hardware and Embedded Systems, LNCS 2162, C.K. Koc, D. Naccache and C. Paar, Eds., Springer-Verlag, 2001, pp. 251-261.

[10] D. Foy, *Mosfet modeling with spice, principles and practice*, Prentice Hall PTR, New Jersey, 1997.

[11] L. Goubin, J. Patarin, *DES and Differential Power Analysis*, in the proceedings of CHES 1999, Springer LNCS, vol 1717, pp 158-172.

[12] C. Karlof and D. Wagner, *Hidden Markov Model Cryptanalysis*, Cryptographic Hardware and Embedded Systems, LNCS 2779, C. D. Walter, C.K. Koc and C. Paar, Eds., Springer-Verlag, 2003, pp 17.

[13] P. Kocher, *Timing attacks on implementations of Diffie-Helman, RSA, DSS and other systems*, Advances in Cryptology, LNCS 1109, N. Koblitz, Ed., Springer-Verlag, 1996, pp. 104-113.

[14] P.Kocher, J.Jaffe, B.Jun, *Differential Power Analysis*, Advances in Cryptology, LNCS 1666, M. Wiener, Ed., Springer-Verlag, 1999, pp. 388-397.

[15] K.S. Kundert, *The designer's guide to spice & spectre*, Kluwen Academic Publishers, Boston/Dordrecht/London, 1995.

[16] R. Mayer-Sommer, *Smartly analyzing the simplicity and the power of simple power analysis on smartcards*, Cryptographic Hardware and Embedded Systems, LNCS 1965, C.K. Koc and C. Paar, Eds., Springer-Verlag, 2000, pp. 78-92.

[17] T.S.Messerges, *Using second-order power analysis to attack DPA resistant software*, Cryptographic Hardware and Embedded Systems, LNCS 1965, C.K. Koc and C. Paar, Eds., Springer-Verlag, 2000, pp. 238-251.

[18] T.S.Messerges, E.A. Dabbish, R.H. Sloan, *Examining Smart-Card Security under the Threat of Power Analysis Attacks*, IEEE transactions on computers, Vol.51, N5, May 2002.

[19] E. Oswald, *Enhancing Simple Power-Analysis Attacks on Elliptic Curve Cryptosystems*, Cryptographic Hardware and Embedded Systems, LNCS 2523, B. S. Kaliski Jr., C. K. Koc and Christof Paar, Eds., Springer-Verlag, 2002, pp 82.

[20] E.Oswald, *On Side-Channel Attacks and the Application of Algorithmic Countermeasures*, Phd Thesis, Institute for Applied Information Processing and Communications (IAIK), TU-Graz, 2003.

[21] G.W. Roberts, A.S. Sedra. *Spice, second edition*, Oxford University Press, New York/Oxford, 1997.

[22] J. Suykens, B. De Moor, *Nonlinear system identification using multilayer neural networks: some ideas for initial weights, number of hidden neurons and error criteria*, in Proc. of the 12th IFAC World Congress, Sydney, Australia, Jul. 1993, pp. 49-52.