Functional Decomposition of Speed Optimized Circuits

B. Steinbach\textsuperscript{1}, F. Schumann\textsuperscript{2}, M. Stöckert\textsuperscript{1}

\textsuperscript{1}Freiberg University of Mining and Technology  
Institute for Computer Science  
D-09596 Freiberg / Germany

\textsuperscript{2}Technical University of Chemnitz-Zwickau  
Department of Electrical Engineering  
D-09009 Chemnitz / Germany

Abstract. There are two ways to design a digital circuit. Covering methods are widely used which includes several speed- or area-optimizing steps. We will show, that decomposition is a profound way to design logical circuits yielding short delay times, low power consumption and a little amount of chip area. This paper provides some new ideas and algorithms to decompose logical functions in order to achieve a high speed of the designed circuit.

1 Introduction

During the synthesis of logical circuits it is possible to optimize the result in reference to various properties, e.g. power consumption or speed. Last year at the PATMOS-Workshop we showed that functional decomposition opens a way to design and to evaluate electronic circuits for low power consumption [SLDZ92]. This year we shall continue this way and design circuits which have good timing properties.

The current way is to design the complete function straight forward and than to optimize the timing behavior. It seems to be better paying attention to some speed-increasing factors already during the synthesis.

In the field of functional decomposition we see opportunities for realizing this. There are three basic ideas:

\begin{itemize}
  \item reduce the path-length of the circuit
  \item avoid negators not necessary
  \item (only efficient for CMOS-circuits) design with negated gates (NAND, NOR, ...) because of their shorter delay time in typical CMOS technology
\end{itemize}
Our current work concerns combinatorical circuits without sequential parts only. We know, there are many factors to influence the speed of an integrated circuit (e.g. the length of the internal wires, the chip-temperature, the voltage, ...), but we are only concerned with pure logical design.

The basic principle is the functional decomposition. Decomposition means to split a complex logical function into two simple sub-functions. The disjunctive, conjunctive or the exclusive-or connection of this sub-functions produces the primary Boolean function. The number of Boolean variables is a criterion for simpler sub-functions. If one sub-function cannot be formed by one gate the decomposition is repeated.

Every step of the decomposition can have a large amount of different results because of different algorithms, different basic elements and ambiguity of the decomposition. The synthesis must be controlled in such a way that the result optimally meets the requirements. An overview off the different kinds of decomposition gives [DRE92]. In this paper only some basic facts about the theory of grouping are explained.

2 Decomposition – the Theory

Actually we distinguish between the groupability and the weak groupability.

Definition 1 (groupability)
A Boolean function $f(x_a, x_b, x_c)$ is said to be (disjunctively, conjunctively or exclusive-or) groupable with the operation $\circ \in \{\lor, \land, \oplus\}$ and the variables $x_a, x_b$ if

$$f(x_a, x_b, x_c) = g(x_a, x_c) \circ h(x_b, x_c)$$

(1)

A groupable function has generally several groupings with the circuit structure shown in Figure 1.

![Fig. 1: Circuit for grouping](image-url)
Not all Boolean functions are groupable. Le [Le89], however, proved that all Boolean functions, which are not groupable by exclusive-or, are always disjunctively or conjunctively weakly groupable.

**Definition 2 (weak groupability)**

A Boolean function \( f(x_a, x_b) \) is said to be (disjunctively, conjunctively or exclusive-or) weakly groupable with the operation \( \odot \in \{\lor, \land, \oplus\} \) and the variables \( x_a \) if

\[
f(x_a, x_b) = g(x_a, x_c) \odot h(x_c)
\]

A weakly groupable function can be implemented by several weak groupings with the circuit structure shown in Figure 2.

**Fig. 2: Circuit for weak grouping**

For a disjunctively or conjunctively weakly groupable function \( f(x_a, x_c) \) the function \( h(x_c) \) can always be selected in such a way that \( g(x_a, x_c) \) can be implemented more easily than \( f(x_a, x_c) \), although the number of variables is not altered.

The question "Is an incompletely specified function with the mark functions \( q(x) \) and \( r(x) \) for \( x_a \) and \( x_b \) groupable?" can be answered using the following existence theorems. The theoretical basis is explained in some publications included in the reference list at the end of this paper.

**Theorem 1 (disjunctively groupable)**

An incompletely specified function with the mark functions \( q(x) \) and \( r(x) \) for \( x_a \) and \( x_b \) is disjunctively groupable, if and only if

\[
q(x) \land \max_{x_a} r(x) \land \max_{x_b} r(x) = 0
\]

The differential equation (3) implies that no pattern from the ON-SET\(q(x)\) may be in the projection of the OFF-SET \((r(x))\) in the \(x_a\) as well as in the \(x_b\) direction.
Dual situations exist in conjunctively groupable functions.

**Theorem 2** (conjunctively groupable)
An incompletely specified function with the mark functions \(q(x)\) and \(r(x)\) for \(x_a\) and \(x_b\) is *conjunctively groupable*, if and only if

\[
\left( x_a \right) \land \max_{\bar{x}_a} q(x) \land \max_{\bar{x}_b} q(x) = 0
\]  

(4)

The exclusive-or groupability for \(x_a\) and \(x_b\) is determined step by step. First, the derivation of the incompletely specified function for one variable \(x_a\) is calculated using Lemma 1.

**Lemma 1** (derivation of an incompletely specified function)
The derivation of an incompletely specified function for \(x_a\) forms also an incompletely specified function with the mark functions

\[
q(x) := \max_{x_a} q(x) \land \max_{\bar{x}_a} r(x)
\]  

(5)

\[
r(x) := \min_{x_a} q(x) \lor \min_{\bar{x}_a} r(x)
\]  

(6)

**Theorem 3** (exclusive-or groupable)
An incompletely specified function with the mark functions \(q(x)\) and \(r(x)\) is *exclusive-or groupable* for one variable \(x_a\) and the variables \(x_b\) if and only if

\[
q(x) \land \max_{\bar{x}_a} r(x) = 0
\]  

(7)

The test (7) for \((x_{a1}, x_b)\) and \((x_{a2}, x_b)\) only confirms the exclusive-or groupability for \(((x_{a1}, x_{a2}), x_b)\), if there is at least one specified function within the incompletely specified function which meets the requirements for \(x_{a1}\) as well as for \(x_{a2}\). The probability of finding such specified functions is increased using theorem 4.
Theorem 4 (restrictions of an incompletely specified function)
The reduced incompletely specified function with the mark functions

\[
q(x) := q(x) \lor r(x \setminus x_a, \overline{x}_a)\max_{\overline{x}_a} q_a(x) \lor q(x \setminus x_a, \overline{x}_a)\max_{\overline{x}_a} r_a(x) \tag{8}
\]

\[
r(x) := r(x) \lor q(x \setminus x_a, \overline{x}_a)\max_{\overline{x}_a} q_a(x) \lor r(x \setminus x_a, \overline{x}_a)\max_{\overline{x}_a} r_a(x) \tag{9}
\]

contains all Boolean functions of the primary incompletely specified function that are exclusive-or groupable for \((x_a, \overline{x}_a)\).
The exclusive-or weak groupability is a property of all Boolean functions. But only Boolean functions which suffice theorem 5 or 6 enable disjunctively or conjunctively weak grouping.

Theorem 5 (disjunctively weakly groupable)
An incompletely specified function with the mark functions \(q(x)\) and \(r(x)\) is disjunctively weakly groupable for \(x_a\), if and only if

\[
q(x) \land \max_{\overline{x}_a} r(x) \neq 0 \tag{10}
\]

Unequation (10) implies that not all patterns of the ON-SET \((q(x))\) may be in the projection of the OFF-SET \((r(x))\) in the \(x_a\)-direction. The patterns of \(q(x)\) which are not covered by the “zero-projection” belong to the function \(q(x_a)\). Here, dual situations with regard to conjunctively weakly groupable functions can be observed, too.

Theorem 6 (conjunctively weakly groupable)
An incompletely specified function with the mark functions \(q(x)\) and \(r(x)\) is conjunctively weakly groupable for \(x_a\), if and only if

\[
r(x) \land \max_{\overline{x}_a} q(x) \neq 0 \tag{11}
\]

If only those properties of incompletely specified functions are used which have been mentioned above, a convergent decomposition method can be developed. This is due to the following theorem.
Theorem 7 (groupable)
Each Boolean function which is not disjunctively weakly groupable and not conjunctively weakly groupable has the property of exclusive-or groupability.

3 Timing model

To compare the results of the synthesis algorithms we have to use a timing model. It is similar to the model used in the program PCGAD (PC Gate Array Design) developed at the Technical University Chemnitz-Zwickau [POMA90].

![Timing models](image)

Fig. 3: Timing models

The timing behavior of a digital element can be described with two different models shown in Figure 3. Model 1 has the delay elements in the input-path. This allows to rate the inputs differently but it requires a great calculating expense. In the other model different kinds of signal-switching at the inputs cause the same (timing) result at the output.

Most of the combinatorial elements (all elements we use in the decomposition method) have the same parameters at all inputs. The mathematical description of the second model and its implementation are simpler, need less calculating time but are still exact enough, so we use that model.

The calculation rule for this model bases on [FS87] and is:

$$t_d = t_{d0} + n_x * t_x + n_y * t_y + k_{re} * t_{re}$$

- $t_d$: resulting delay time
- $t_{d0}$: basic delay time
- $n_x$: number of direct loads
- $t_x$: delay time for one direct load
- $n_y$: number of indirect loads
- $t_y$: delay time for one indirect load
- $k_{re}$: factor for rise-time of input-edge
- $t_{re}$: rise-time of input-edge
The direct load depends on the gate-inputs connected with the non-negated output of each element, the indirect load on the inputs connected with the negated output of the element. There is no indirect load due to the principle of decomposition, so we can set $n = 0$. Furthermore, we set factor $k_{re} = 0$ because the edges of signals are ignored. This model uses a scaling factor to include the average operating conditions (25°C, 5 V) into the calculating order. The capacitances of the inputs are approximated with 0.25 pF. For our practical work we use the average of the LH-and the HL-edge and the value for one direct load. The influence of the internal wires is totally neglected. Usable results for internal wiring delay are only available after placing and routing but that is not a task of the logic design process.

The parameters of the used gates are shown in table 1.

### Table 1: Timing parameters of used gates in [ps]

<table>
<thead>
<tr>
<th>parameter</th>
<th>INV</th>
<th>AND / NAND</th>
<th>OR / NOR</th>
<th>EXOR / EXAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{d0LH}$</td>
<td>340</td>
<td>660 / 500</td>
<td>580 / 690</td>
<td>910 / 490</td>
</tr>
<tr>
<td>$t_{d0HL}$</td>
<td>180</td>
<td>660 / 410</td>
<td>870 / 380</td>
<td>1130 / 510</td>
</tr>
<tr>
<td>$t_{xLH}$</td>
<td>830</td>
<td>870 / 820</td>
<td>810 / 1420</td>
<td>1390 / 1420</td>
</tr>
<tr>
<td>$t_{xHL}$</td>
<td>800</td>
<td>780 / 1100</td>
<td>860 / 790</td>
<td>730 / 1120</td>
</tr>
<tr>
<td>$t_{d}$</td>
<td>463</td>
<td>866 / 695</td>
<td>933 / 811</td>
<td>1285 / 817</td>
</tr>
</tbody>
</table>

The maximum delay time of the circuit was calculated with a structural method. This method has the advantage of less time expense. Because of our simplifications we have checked the deviation between the results of the functional and the structural way and found out that it was less then 10%, if the delay time is $>2000$ps.

### 4 Algorithm and Implementation

The algorithm bases on the simple decomposing algorithm described in [BDS91] and [HIL92]. Details are explained in [SCHU93]. It is working without any special selection of groupings. In the algorithm presented last year (now named "area") we have tried to minimize the power consumption and the area of the chip by selecting a decomposition with the smallest number of $x_c$ -variables. The new algorithm bases on the idea that the more symmetrical and the larger the vectors $xa$ and $xb$ are the better is the timing behavior of the design. The demand for symmetry bases on the fact that Boolean functions with equal numbers of variables cause circuits with a similar expense of gates.
A maximum size of $x_a$ and $x_b$ results in a smaller vector $x_c$ and so functions $g(x_a, x_c)$ and $h(x_b, x_c)$ are dependent on less variables. Table 2 shows some theoretically possible groupings of a function with 10 variables as an example.

### Table 2: Example of possible groupings with VMAX=16

| $|x_a|$ | $|x_b|$ | $|x_c|$ | $|f|$ | $|g|$ | $|h|$ | coeff. |
|------|------|------|------|------|------|--------|
| 1    | 7    | 2    | 10   | 3    | 9    | 23     |
| 2    | 3    | 5    | 10   | 7    | 8    | 35     |
| 2    | 6    | 2    | 10   | 4    | 8    | 38     |
| 3    | 4    | 3    | 10   | 6    | 7    | 52     |
| 4    | 5    | 1    | 10   | 5    | 6    | 69     |
| 5    | 5    | 0    | 10   | 5    | 5    | 85     |
| 7    | 3    | 0    | 10   | 7    | 3    | 55     |

We can see that the grouping where the largest vector of $g(x)$ and $h(x)$ has the smallest number of variables has the largest coefficient, too. The formula of the coefficient is:

$$|x_a| \geq |x_b| : \text{coeff} = VMAX \times |x_a| + |x_b|$$  \hspace{1cm} (13)

$$|x_a| < |x_b| : \text{coeff} = VMAX \times |x_a| + |x_b|$$  \hspace{1cm} (14)

VMAX is a scaling factor to decrease the influence of the larger vector of variables (VMAX-value in table 2 : $2^4 = 16$) if the number of variables in the smaller vector is equal to the number of variables in the larger one.

As a criterion for the best grouping the smaller one of $x_a$ and $x_b$ should be as large as possible. If more than one grouping has the same number of variables in the smaller vector, the grouping with the largest larger vector is the best. In contrast to the area-optimizing algorithm where first one vector is filled now the number of variables in the vectors $x_a$ and $x_b$ is increased alternatly. Figure 4 shows the algorithm of variable distribution and figure 5 the algorithm of decomposition.

The variables used in the algorithm of decomposition are kind (kind of grouping $\rightarrow$ disjunctive, conjunctive, exclusive-or, disjunctive weak, conjunctive weak) and the coefficients $d\_coeff$ (for disjunctive grouping), $c\_coeff$ (for conjunctive grouping), $e\_coeff$ (for exclusive-or grouping), $dw\_coeff$ (for disj. weak grouping) and $cw\_coeff$ (for conjunctive weak grouping).
calculating initial groupability

for all variables $x_i$ in $X$

<table>
<thead>
<tr>
<th>$X_a$</th>
<th>$X_b$</th>
<th>$X_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_a + x_i$</td>
<td>$X_b + x_i$</td>
<td>$X_c + x_i$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$(X_a, x_i), X_b$</th>
<th>$(X_b, x_i), X_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

$X_b + x_i$  \[  | X_a | < | X_b | \]

<table>
<thead>
<tr>
<th>$(X_a, x_i), X_b$</th>
<th>$(X_b, x_i), X_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

$X_b + x_i$  \[  | X_a | < | X_b | \]

<table>
<thead>
<tr>
<th>$(X_a, x_i), X_b$</th>
<th>$(X_b, x_i), X_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Fig. 4: The algorithm of variable distribution

Fig. 5: The algorithm of speed optimized decomposition(„time“)
5 Results

In order to validate our results we have designed some benchmark-circuits from the MCNC benchmark-suite with two different algorithms. On one hand we have the new algorithm to design speed optimized circuits ("time"), on the other hand we have a algorithm, which uses the method "vector $\bar{x}_c$ possibly small" (named "area") to design area-optimized circuits. The program called "designer" was tested on SPARC STATION 10 with SOLARIS 1.1 and on PC. The compiling process of the programs should be possible on all systems running the "GNU-C"-compiler.

<table>
<thead>
<tr>
<th>benchmarks</th>
<th>number of inputs</th>
<th>number of outputs</th>
<th>gates of algorithm &quot;time&quot;</th>
<th>gates of algorithm &quot;area&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>5xp1</td>
<td>7</td>
<td>10</td>
<td>114</td>
<td>121</td>
</tr>
<tr>
<td>9sym</td>
<td>9</td>
<td>1</td>
<td>260</td>
<td>240</td>
</tr>
<tr>
<td>Bw</td>
<td>5</td>
<td>28</td>
<td>367</td>
<td>322</td>
</tr>
<tr>
<td>con1</td>
<td>7</td>
<td>2</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>f2</td>
<td>4</td>
<td>4</td>
<td>24</td>
<td>40</td>
</tr>
<tr>
<td>misex1</td>
<td>8</td>
<td>7</td>
<td>129</td>
<td>115</td>
</tr>
<tr>
<td>misex2</td>
<td>25</td>
<td>18</td>
<td>245</td>
<td>254</td>
</tr>
<tr>
<td>rd53</td>
<td>5</td>
<td>3</td>
<td>38</td>
<td>40</td>
</tr>
<tr>
<td>rd73</td>
<td>7</td>
<td>3</td>
<td>135</td>
<td>132</td>
</tr>
<tr>
<td>rd84</td>
<td>8</td>
<td>4</td>
<td>243</td>
<td>265</td>
</tr>
<tr>
<td>sao2</td>
<td>10</td>
<td>4</td>
<td>230</td>
<td>237</td>
</tr>
<tr>
<td>duke2</td>
<td>22</td>
<td>29</td>
<td>1168</td>
<td>1365</td>
</tr>
<tr>
<td>f51m</td>
<td>8</td>
<td>8</td>
<td>90</td>
<td>111</td>
</tr>
<tr>
<td>misex3c</td>
<td>14</td>
<td>14</td>
<td>1241</td>
<td>1204</td>
</tr>
<tr>
<td>vg2</td>
<td>25</td>
<td>8</td>
<td>486</td>
<td>541</td>
</tr>
<tr>
<td>z4ml</td>
<td>7</td>
<td>4</td>
<td>68</td>
<td>50</td>
</tr>
<tr>
<td>Total</td>
<td>171</td>
<td>147</td>
<td>4862</td>
<td>5062</td>
</tr>
</tbody>
</table>
Table 4: Delay-time of the longest path of some MCNC-benchmarks in [ns]

<table>
<thead>
<tr>
<th>benchmark</th>
<th>algorithm &quot;time&quot;</th>
<th>algorithm &quot;area&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>5xp1</td>
<td>6.75</td>
<td>7.21</td>
</tr>
<tr>
<td>9sym</td>
<td>10.88</td>
<td>10.41</td>
</tr>
<tr>
<td>bw</td>
<td>6.98</td>
<td>7.05</td>
</tr>
<tr>
<td>con1</td>
<td>4.48</td>
<td>4.99</td>
</tr>
<tr>
<td>f2</td>
<td>4.06</td>
<td>4.06</td>
</tr>
<tr>
<td>misex1</td>
<td>6.70</td>
<td>6.70</td>
</tr>
<tr>
<td>misex2</td>
<td>5.73</td>
<td>12.14</td>
</tr>
<tr>
<td>rd53</td>
<td>4.83</td>
<td>4.83</td>
</tr>
<tr>
<td>rd73</td>
<td>7.92</td>
<td>7.92</td>
</tr>
<tr>
<td>rd84</td>
<td>11.51</td>
<td>11.51</td>
</tr>
<tr>
<td>sao2</td>
<td>8.73</td>
<td>9.08</td>
</tr>
<tr>
<td>duke2</td>
<td>10.26</td>
<td>12.68</td>
</tr>
<tr>
<td>f51m</td>
<td>7.97</td>
<td>8.43</td>
</tr>
<tr>
<td>misex3c</td>
<td>14.72</td>
<td>14.76</td>
</tr>
<tr>
<td>vg2</td>
<td>13.01</td>
<td>16.47</td>
</tr>
<tr>
<td>z4ml</td>
<td>6.33</td>
<td>6.33</td>
</tr>
<tr>
<td>total</td>
<td>130.86</td>
<td>144.57</td>
</tr>
</tbody>
</table>

In table 3 we have put together the number of gates of 16 benchmarks-circuits designed by the algorithms "time" and "area". You can see, that in 10 of 16 cases the algorithm "time" achieves better results (lower number of gates up to 60%) than the algorithm "area". The average area-decrease is about 4%. This is an interesting fact because we have not paid any attention to area-minimization.

We think it's a problem of the selection of the optimal kind of decomposition and of the selection of variables for grouping. Currently, we work with a fixed order of decomposition and a variable-selection by chance. In future we try to control also this feature of decomposing process.
In table 4 the results of timing analysis are listed. Only in the benchmark \(9_{sym}\) the delay of the longest path designed with the algorithm "time" is not shorter. The average speed-increase of the algorithm "time" is about 9.5% and there are further ideas to improve this algorithm.

We have not used any post layout evaluation because in our opinion it is effective only after placing and routing and so a task of a logical simulator.

6 Summary

There is a way to produce speed optimized circuits with a special kind of functional decomposition. This method is suitable for designing large incompletely specified functions with short delay times and a small amount of chip-area. In our future work we try to increase the number of available groupings in select the best of all.

Furthermore, we will improve the implementation of our algorithms in order to design larger problems in less time.

7 References

[BDS91] Bochmann, D., Dresig, F., Steinbach, B.  
A New Decomposition Method for Multilevel Circuit Design  
Proceedings of the European Conference on Design Automation  

[BOST91] Bochmann, D., Steinbach, B.  
Logikentwurf mit XBOOLE  
Verlag Technik, Berlin, 1991

[DRE92] Dresig, F.  
Gruppierung - Theorie und Anwendung in der Logiksynthese  
VDI Fortschrittsberichte 9/145, VDI Verlag, 1992

[FS87] Fischer, W. J., Schüffny, R.  
MOS-VLSI-Technik  
Akademie-Verlag, Berlin, 1987

[HIL92] Hilbert, H.  
Module und Tools zum Experimental-system  
Diplomarbeit, TU Chemnitz, Fachbereich Elektrotechnik, 1992
[LE89] Le, T.Q.
Testbarkeit kombinatorischer Schaltungen - Theorie und Entwurf
Dissertation, TU Karl-Marx-Stadt, 1989

[PATMA90] Porsch, U., Mayas, R.-D.
Konzepte, Implementierung und Optimierung von Programmen zur
zeitlichen Logiksimulation von CMOS Gate-Array-Schaltungen
Dissertation A, TU Chemnitz, 1990

[SCHU93] Schumann, F.
Synthesestrategien für laufzeitoptimale Schaltungen
Diplomarbeit, TU Chemnitz-Zwickau, Fachbereich Elektrotechnik, 1993

[SLDZ92] Steinbach, B., Lang, Ch., Dresig, F., Zühlke, R.
Analytical Power Estimation for CMOS Circuits at Logical Level
Proceedings of the European Workshop on Power and Timing
Modelling