Abstract

Controller circuits synthesized from high-level languages often have many more latches than the minimum, with a resulting sparse reachable state space that has a particular structure. We propose an algorithm that exploits this structure to remove latches. The reachable state set (RSS) is much easier to compute for the new, smaller circuit and can be used to efficiently compute the RSS of the original. Thus we provide a method for obtaining the RSS, and two different initial implementations from which to begin logic optimization.

1 Introduction

The computation of the reachable state set (RSS) of a sequential circuit is important for verification, logic optimization and test generation. The RSS computation is typically done incrementally, by looping over a computation of the next states as the image of the current states by a vector of Boolean functions [3]. When BDD-based algorithms are used, the variables are the latches and the circuit inputs. Therefore, the number of latches greatly affects the efficiency, both in memory and computation time.

In this paper, we propose a new latch removal algorithm that greatly increases the efficiency of the RSS computation while optimizing the circuit (by eliminating latches). We derive an over-approximation of the RSS (called ORSS) directly from the initial high-level specification. We then use the ORSS to compute the latches to be removed. We remove them, and compute the precise RSS of the new circuit from which we can compute the precise RSS of the original circuit. We note that the ORSS, while only an approximation of the RSS, can be used for:

1. verification, where an ORSS may be sufficient for checking a particular property [9, 10].
2. logic optimization, where the complement of the ORSS is a set of don’t care conditions [7, 8].

We concentrate on control or glue logic circuits programmed in the ESTEREL synchronous language [1, 2] and on a particular kind of ORSS derived from exclusive threads of control that occur naturally in the specifications. From this ORSS based on exclusive threads, we compute precisely sets of exclusive latches in the generated circuit. Two sets of latches $L_1$ and $L_2$ are exclusive if all registers in $L_1$ have value 0 if at least one register in $L_1$ has value 1 and conversely. In an ESTEREL sequence $p; q$, the sets of latches generated by $p$ and $q$ are exclusive. Since sequencing and concurrency can be mixed at any level, the exclusive sets structure can be quite rich; consider for example $(p [q; r] ; s) [t [u]]$. The richer the structure is, the better the ORSS is. Furthermore, the ORSS formula is very easy to compute, unlike the exact RSS.

Exclusive sets of latches can be easily and efficiently re-encoded in such a way that the total number of latches is decreased. Let $M$ be the initial circuit and $M'$ be the re-encoded one. Since $M'$ has less latches, its RSS $R'$ is easier to compute than that of $M$. Furthermore, the original RSS $R$ can be easily recovered from $R'$. Optimization and verification can then proceed either from $M$ and $R$ or $M'$ and $R'$, whichever is best.

Exclusive sets can also be computed from the exact RSS. They can be used for latch removal in complement to the techniques formerly presented in [7].

In Section 2, we give an overview of the algorithm, the circuit transformation it employs and its important properties. The re-encoding by exclusive sets is given in Section 3 and its properties are presented in Section 4. The results of our implementation are presented in Section 5. Conclusions of our work and directions for future developments are presented in Section 6.

2 Overview of the Algorithm

The algorithm for exclusive-set latch removal proceeds as follows:

1. Derive the ORSS from the structure of the high-level language description.
2. Compute the exclusive latches and determine two maximal exclusive sets of latches using the ORSS. These two sets of latches can be multiplexed through a single set, with an additional latch for the multiplexer.
3. Remove one of the latch sets, add the multiplexing logic and update the ORSS.
4. Iterate, removing as many latches as possible, or as many latches as desired given constraints on the size of the combinatorial logic.
5. Compute the exact RSS on the new circuit, and use it to compute the exact RSS for the original circuit.

2.1 General Circuit Transformation

A framework for state re-encoding was described in [7] and is illustrated in Figure 1. We use the same framework for exclusive sets.

![Figure 1: General Circuit Transformation](image)

The original FSM is M and the transformed one is $M'$. We call $L = \{i | 1 \leq i \leq n\}$ the set of latches of $M$, and $L'$ the set of latches of $M'$. Our transformation computes two sets of latches $L_1 = \{i | i \in I_1\} \subseteq L$ and $L_2 = \{i | i \in I_2\} \subseteq L$ such that $|L_1| = |L_2| = k \geq 2$; it re-encodes them with a new set of latches $L'_1 \subseteq L'$ such that $|L'_1| = k$ and a new register denoted $L'_{k+1}$. For convenience, the latches in the set $L'_1$ will have the same indices as those in $L_1$. The set $L'$ is then $L' = \{i' | 1 \leq j \leq n+1, j \notin I_2\}$.

The size of $L'$ is $n' = n + 1$.

The next-state vector for $L$ in $M$ is generated by $C$ and called $Y$. In $M'$, the encoding function $E$ has type $B^m \rightarrow B^n$, with components $E_i : B^m \rightarrow B$, $1 \leq i \leq n+1, i \notin I_2$. The decoding function $D$ has type $B^n \rightarrow B^m$, with components $D_i : B^n \rightarrow B$, $1 \leq i \leq n$.

Let $R \subseteq B^m$ (resp. $R' \subseteq B^n$) be the set of reachable states in $M$ (resp. $M'$), and let $R$ and $R'$ denote the characteristic function of $R$ and $R'$ (i.e., $R(r) = 1$ if $r \in R$). Let $r_0$ and $r'_0$ denote the initial states of $M$ and $M'$. Clearly, $M$ is equivalent to $M'$ if $r_0 = D(r'_0)$ and $D(E(r')) = r$ for all $r' \in R$. This is the equivalence property our transformation satisfies.

Functions will be represented by polynomials or BDDs. The input variables will be consistently called $y_i$, $E_i$, $l_i$ for $D$, $l_i$ for $R$, and $l'_i$ for $R'$. If $F$ denotes a polynomial or BDD over a set of variables $X$ with a set of associated indices $I_x$, for $Y$ another set of variables with a set of associated indices $I_y$ where $|I_x| = |I_y|$, and for $map: I_x \rightarrow I_y$, we denote by $F[X/Y]_{map}$ the result of the substitution of the $x_i$ by the $y_{map(i)}$, and $F[Y/X]_{map}$ the result of the substitution of the $x_i$ by the complements $\neg y_{map(i)}$. We denote by $F_+$ ($F_-$) the positive (negative) cofactor of $F$ with respect to $z$. If $X$ is a set of variables, we denote by $F_X$ ($F_X$) the result of successive positive (negative) cofactoring of $F$ with respect to each $x \in X$.

2.2 Properties of the Transformation

The exclusive-set latch removal implies a re-encoding that is reversible: it is a one-to-one function that is easily computed. As a result, the RSS of the initial circuit can be recovered by a simple BDD operation, even after multiple iterations (see Section 4.1). This is an important feature when the algorithm is used as part of an optimization strategy.

The re-encoding is also localized to the Boolean sub-space defined by the exclusive sets; the remaining latches are unchanged. This implies commutativity properties when applied iteratively (Section 4.3). The transformation also commutes with our other techniques of latch removal (Section 4.4 and [7]) so it can be applied in conjunction with them without affecting their efficacy.

The logic added by the transformation is strictly controlled, see Section 4.2, and the initial logic structure is preserved as much as possible as it contains valuable information about the natural structure of the circuit. Thus the computations required remain tractable for very large circuits, as does the additional combinational logic, which is usually the most limiting factor in incremental re-encoding algorithms.

Finally, the transformation preserves behavior on the RSS and remains correct for every other over-approximation of it, see Section 3.2. Therefore it can be used in conjunction with other efficient techniques for approximate reachability analysis [4].

3 The Exclusive Sets Transformation

3.1 Exclusive Sets of Latches

Definition 1 Two latches $l_1$ and $l_2$ are said to be exclusive, denoted $l_1 \# l_2$, if for each reachable state:

\[
\begin{align*}
& l_1 = 1 \Rightarrow l_2 = 0 \\
& l_2 = 1 \Rightarrow l_1 = 0
\end{align*}
\]

The equations (1) are equivalent to:

\[R_{l_1,l_2} = 0.\]

Definition 2 Two disjoint sets of latches $L_1 \subseteq L$ and $L_2 \subseteq L$ are said to be exclusive if

\[\forall l_1 \in L_1, \forall l_2 \in L_2, l_1 \# l_2\]

In the sequel, we assume that $|L_1| \leq |L_2|$. If $|L_1| \leq |L_2|$ we simply choose a subset $L_1' \subseteq L_2$ such that $|L_1'| = |L_1|$.

![Figure 2: Partition of the RSS by exclusive sets $L_1$ and $L_2$](image)

A geometrical interpretation of the partition of the RSS by $L_1$ and $L_2$ is given in Figure 2a. The axis labeled $L_1$ contains the circular Gray code of Boolean encodings of the integers in the interval $[0, 2^{L_1} - 1]$ (similarly for the axes labeled $L_2$ and $L \setminus (L_1 \cup L_2)$). A point in the cube represents a state, and its encoding is retrieved by concatenation of the projections on the axes. If $L_1 \# L_2$, the reachable states are points in the cube only on the hyper planes $P_1 = L_1 \times L \setminus (L_1 \cup L_2)$ and $P_2 = L_2 \times L \setminus (L_1 \cup L_2)$, and the rest of the hypercube $B^n$ is empty.

3.2 Transformation

The hyper planes $P_1$ and $P_2$ in Figure 2a are each Boolean spaces of dimension $|L_1| + |L_2|$, so we can represent their union using only $|L| - |L_1| + 1$ state variables instead of the $|L|$ state variables of
balanced complete bipartite graph problem \[5\], where each node en-
tees a global optimum. This problem is equivalent to the NP-hard
purpose of the exclusive sets transformation is to replace the latches
k have adopted a greedy bounding heuristic.

For E STEREL-produced circuits, we efficiently generate exclusive
sets from the RSS by using equation (2). We search for a pair
of exclusive sets \(L_1\) and \(L_2\) of maximum size (which does not
guarantee a global optimum). This problem is equivalent to the NP-hard
balanced complete bipartite graph problem \[5\], where each node

4 Properties

4.1 Recovering the original RSS

One can recover \(R\) from \(R'\) knowing only the \(map\) latch assignment
function:

**Proposition 1** Let \(R'\) be the characteristic function of the RSS
of the circuit \(M'\) obtained by performing the exclusive set transforma-
tion using \(L_1\) and \(L_2\) of \(M\). Then the RSS characteristic function
\(R\) of the initial circuit can be recovered as follows:

\[
\mathcal{R} = \mathcal{R}'_{n+1}[L_1/L_1], \prod_{L_2} I' \mathcal{R}'_{n+1}[L_2/L_1]_{\text{map}}, \prod_{L_2} I' \mathcal{R}_i \quad (7)
\]

4.2 Logic Cost

The circuit transformation replaces \(k - 1\) latches with 2\(k\) binary
AND gates, \(k\) binary OR gates and a \(k\)-input OR gate. The ad-
ditional logic is not critical if the transformation is performed with
the aim of computing the reachable states. In this case, any penalty
in BDD size increase caused by the additional logic is easily compen-
sated by the removal of \(k - 1\) latches. If the transformation is
performed for optimization purposes, the most critical point is the
size of the \(k\)-input OR gate which may need to be implemented by
several levels of logic depending on the technology library. In that
case, it may be better to optimize the original circuit now knowing
its RSS \(R\).

4.3 Commutativity

To understand the behavior of the algorithm over several iterations,
we have to examine how the removal of latches for a pair of exclusive
sets affects the conditions for other pairs of exclusive sets.

**Proposition 2** Let \(L_1, L_2, L_3\) and \(L_4\) be sets of latches such
that \(L_1 \neq L_2, L_3 \neq L_4\). Then \(L_3 \neq L_4\) will remain true after the exclusive
sets transformation over \(L_1\) and \(L_2\) if \((L_1 \cup L_2) \cap (L_3 \cup L_4) = \emptyset\).

When the commutativity property holds, it simplifies the updat-
ing of the list of exclusive sets corresponding to each latch. We
simply remove from them the latches implied in the previous trans-
formation, without recomputing the incompatibilities over the new
reachable state space.

4.4 Exclusive Sets Transformation Versus

**single-latch Removal**

The single-latch removal [3, 6, 7] algorithm is useful in logic
optimization because of its incremental nature and tight control of
the size of the additional logic. The condition for single-latch
removal,

\[
\mathcal{R}_i \cdot \mathcal{R}_i' = 0 \quad (8)
\]

is in general not preserved by state space re-encoding.

![Figure 3: Representation of the single-latch removal of \(i_b\) in the
context of exclusive sets \(L_1\) and \(L_2\).](image)

The single-latch removal of a latch \(i_b\) is intuitively a pair-
ning of states in \(R_i\) and \(R_i'\). Figure 3 shows the reachable state
space before and after removal of latch \(i_b \in L_1 \cup L_2\). Semi-
planes \(P_1\) and \(P_2\) contain all reachable states where \(i_b = 1\);
\(P_1\) and \(P_2\) all reachable states where \(i_b = 0\). The pairing is
graphically equivalent to folding \(P_1\) onto \(P_2\) and \(P_2\) onto \(P_1\)
as shown in Figure 3b).

By cutting the planes in Figure 2b, we can make the same
reasoning about the fact that the condition necessary for
single-latch removal (8) remains true after an exclusive
sets transformation for \(i_b \in L_1 \cup L_2\).

Thus, for \(i_b \notin L_1 \cup L_2\) the exclusive latch condition is
preserved during single-latch removal and the single-latch
removal condition is preserved during the exclusive set transfor-
mation.
We have experimentally demonstrated that the exclusive sets transformation is very useful for circuits with sparse state encodings. Our techniques are particularly useful for controllers generated from ESTEREL, for which the basic synthesis process usually generates too many latches. We are now including the algorithms in the ESTEREL v5 compiling and optimization chain.

A nice feature of the exclusive sets transformation is that the original RSS can be cheaply recomputed from the RSS of the transformed circuit. It will also be interesting to explore the possibility of performing a similar backwards RSS computation for other latch removal techniques.

### Table 1: Results of the exclusive sets in latch removal

<table>
<thead>
<tr>
<th>Circuit</th>
<th>#in</th>
<th>#out</th>
<th>#reg</th>
<th>#lit</th>
<th>ORSS</th>
<th>RSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>abc</td>
<td>4</td>
<td>12</td>
<td>13</td>
<td>16</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>abcdef</td>
<td>7</td>
<td>24</td>
<td>25</td>
<td>128</td>
<td>19</td>
<td>14</td>
</tr>
<tr>
<td>controle</td>
<td>10</td>
<td>8</td>
<td>24</td>
<td>20</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>controlecount</td>
<td>10</td>
<td>8</td>
<td>23</td>
<td>211</td>
<td>19</td>
<td>17</td>
</tr>
<tr>
<td>habault</td>
<td>4</td>
<td>19</td>
<td>90</td>
<td>66</td>
<td>67</td>
<td>40</td>
</tr>
<tr>
<td>renault</td>
<td>23</td>
<td>179</td>
<td>66</td>
<td>257</td>
<td>37</td>
<td>19</td>
</tr>
<tr>
<td>runner</td>
<td>6</td>
<td>5</td>
<td>30</td>
<td>5182</td>
<td>29</td>
<td>22</td>
</tr>
<tr>
<td>sequencer</td>
<td>55</td>
<td>98</td>
<td>121</td>
<td>109415</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>sneeca</td>
<td>23</td>
<td>7</td>
<td>102</td>
<td>407</td>
<td>54</td>
<td>40</td>
</tr>
<tr>
<td>tcnt</td>
<td>19</td>
<td>20</td>
<td>96</td>
<td>2826</td>
<td>50</td>
<td>46</td>
</tr>
<tr>
<td>tcntnocount</td>
<td>13</td>
<td>22</td>
<td>90</td>
<td>310</td>
<td>38</td>
<td>42</td>
</tr>
<tr>
<td>tcount</td>
<td>3</td>
<td>12</td>
<td>32</td>
<td>358</td>
<td>28</td>
<td>22</td>
</tr>
<tr>
<td>t-controller</td>
<td>3</td>
<td>11</td>
<td>68</td>
<td>540</td>
<td>61</td>
<td>38</td>
</tr>
<tr>
<td>t-mul-control</td>
<td>2</td>
<td>6</td>
<td>21</td>
<td>32</td>
<td>19</td>
<td>13</td>
</tr>
<tr>
<td>t-simple-control</td>
<td>3</td>
<td>11</td>
<td>67</td>
<td>451</td>
<td>60</td>
<td>39</td>
</tr>
</tbody>
</table>

### Table 2: Remlatch [7] followed by Exclusive sets

<table>
<thead>
<tr>
<th>Circuit</th>
<th>#in</th>
<th>#out</th>
<th>#reg</th>
<th>#lit</th>
<th>remlatch</th>
<th>excl</th>
</tr>
</thead>
<tbody>
<tr>
<td>abc</td>
<td>4</td>
<td>12</td>
<td>13</td>
<td>16</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>abcdef</td>
<td>7</td>
<td>24</td>
<td>25</td>
<td>128</td>
<td>19</td>
<td>14</td>
</tr>
<tr>
<td>controle</td>
<td>10</td>
<td>8</td>
<td>24</td>
<td>20</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>controlecount</td>
<td>10</td>
<td>8</td>
<td>23</td>
<td>211</td>
<td>19</td>
<td>17</td>
</tr>
<tr>
<td>habault</td>
<td>4</td>
<td>19</td>
<td>90</td>
<td>66</td>
<td>67</td>
<td>40</td>
</tr>
<tr>
<td>renault</td>
<td>23</td>
<td>179</td>
<td>66</td>
<td>257</td>
<td>37</td>
<td>19</td>
</tr>
<tr>
<td>runner</td>
<td>6</td>
<td>5</td>
<td>30</td>
<td>5182</td>
<td>29</td>
<td>22</td>
</tr>
<tr>
<td>sequencer</td>
<td>55</td>
<td>98</td>
<td>121</td>
<td>109415</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>sneeca</td>
<td>23</td>
<td>7</td>
<td>102</td>
<td>407</td>
<td>54</td>
<td>40</td>
</tr>
<tr>
<td>tcnt</td>
<td>19</td>
<td>20</td>
<td>96</td>
<td>2826</td>
<td>50</td>
<td>46</td>
</tr>
<tr>
<td>tcntnocount</td>
<td>13</td>
<td>22</td>
<td>90</td>
<td>310</td>
<td>38</td>
<td>42</td>
</tr>
<tr>
<td>tcount</td>
<td>3</td>
<td>12</td>
<td>32</td>
<td>358</td>
<td>28</td>
<td>22</td>
</tr>
<tr>
<td>t-controller</td>
<td>3</td>
<td>11</td>
<td>68</td>
<td>540</td>
<td>61</td>
<td>38</td>
</tr>
<tr>
<td>t-mul-control</td>
<td>2</td>
<td>6</td>
<td>21</td>
<td>32</td>
<td>19</td>
<td>13</td>
</tr>
<tr>
<td>t-simple-control</td>
<td>3</td>
<td>11</td>
<td>67</td>
<td>451</td>
<td>60</td>
<td>39</td>
</tr>
</tbody>
</table>

### Proposition 3

If $L_1$ and $L_2$ are two exclusive sets, and if $l_k \notin L_1 \cup L_2$ is single-latch removable, then the exclusive latch transformation for $L_1$, $L_2$ and the single-latch transformation for $l_k$ commute.

### Observation

The property remains true for generalizations of the single-latch technique [7].

## 5 Results

The benchmarks we used are circuits synthesized by the ESTEREL v5 compiler, and the results are summarized in Tables 1 and 2. Some of them are simple programs, such as abc, abcdef. The remaining files come from industrial designs, and some of them are quite large (sequencer, tcnt, renault, sneeca). They are all control or glue logic circuits.

In Table 1 we compare the number of latches removed by the exclusive sets transformation using ORSS structural information from the ESTEREL description (column ORSS) with the number obtained using exclusive sets computed from the exact RSS (column RSS). Since the ORSS is easy to compute, the ORSS-based algorithm is orders of magnitude faster than the RSS-based one at the expensive of accuracy: fewer exclusive latches are found on the ORSS. As mentioned in Section 3.3, the algorithms compute the exclusive condition exactly, but remove latches greedily. This explains the fact that for file tcntnocount the result with ORSS is better than with RSS.

In Table 2 we present the results of applying first the latch removal algorithms presented in [7] (column remlatch) and then the RSS-based exclusive sets algorithm (column excl). The exclusive sets algorithm removes latches that cannot be removed by the previous techniques, but the combinational logic tends to increase.

### References


