Test Generation for Sequential Networks Affected by Reconvergent Fanout: a Solution Based on a 9-Valued Algebraic Circuit Model

Enrico Macii*
University of Colorado
Dept. of Electrical and Computer Eng.
Boulder, CO 80309

Angelo R. Meo
Politecnico di Torino
Dip. di Automatica e Informatica
Torino, ITALY 10129

Abstract
Test generation for sequential networks affected by reconvergent fanout implies the simultaneous management of the good circuit and the faulty circuit. A theoretical approach to the solution of this problem, based on the definition of a 9-Valued Algebraic Circuit Model, is proposed in this paper. Results obtained on the set of scan'89 circuits show the effectiveness of the technique.

1 Introduction
Automatic Test Pattern Generation (ATPG) for sequential circuits has been recognized as a difficult problem [1]. Initial work on this topic involved the use of both random [2, 3] and deterministic [4, 5] techniques. Due to the relative ineffectiveness of these approaches, a popular way to enhancing the testability of sequential circuits has been the scan design methodology [6, 7]. Recently, there has been considerable progress made in the area of sequential ATPG. A PODEM-based [8] deterministic approach to sequential test generation (STALLION) was presented in [9]. A heuristic, simulation-based test pattern generation algorithm (CONTEST) was described in [10]. An approach based on decomposing the test generation problem into three subproblems of combinational test generation, fault-free state justification, and fault-free state differentiation (STEED) was introduced in [11]. The state of the art, concerning ATPG for sequential circuits, is based on the ESSENTIAL algorithm described in [12]. In that paper, the basic ideas of the combinational test generation Socrates [13] have been extended to the manipulation of sequential networks. Other test generation results, comparable to the ones obtained with ESSENTIAL, have been published in the last two years [14, 15, 16, 17]. Finally, [18] presents an efficient test generation system for circuits described by their state transition table, the transition fault model and its relation to the stuck-at fault model realizing an interesting bridge between functional and gate-level test generation.

A possible solution to the problem of Sequential ATPG has been proposed by the authors in [19]. A careful analysis of the findings obtained running the program with the debug option enabled showed that handling simultaneously the good circuit (i.e., the fault-free circuit), and the faulty circuit, when they have reconvergent fanout, results in the loss of some solutions, that is, some faults are classified as undetectable when a test pattern for their detection actually exists [20]. For this reason, a 9-Valued Circuit Model, based on the definition of a 9-Valued Algebra [21] has been partially introduced in [22]. This paper formally defines that model, and it shows how the original ATPG procedure has been modified to increase its performance.

The rest of this paper is organized as follows. Section 2 is dedicated to the definition of the 9-Valued Algebra, and Section 3 provides the basic theory to manipulate the 9-Valued Functions. Section 4 shows how the 9-Valued Circuit Model can be used to solve the test generation problem. Section 5 presents the experimental results obtained on the scan'89 synchronous benchmark circuits [23]. Finally, Section 6 gives conclusions, and points out the direction of our future work.

2 9-Valued Algebra
The nine values of the Algebra are derived by distinguishing between the good circuit (i.e., the fault-free circuit), and the faulty circuit.

Given a generic variable, \( \varepsilon \), the nine values that it can assume are pairs of the form:

\[ \{ \varepsilon \varepsilon ' \} \]

where \( \varepsilon , \varepsilon ' \in \{ 0, 1, \} \), and they are summarized in the first column of Table 1.

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0,0 )</td>
<td>0</td>
<td>#</td>
</tr>
<tr>
<td>( 1,1 )</td>
<td>1</td>
<td>=</td>
</tr>
<tr>
<td>( 1,0 )</td>
<td>U</td>
<td>⊽</td>
</tr>
<tr>
<td>( 0,1 )</td>
<td>G0</td>
<td>⊠</td>
</tr>
<tr>
<td>( 1, - )</td>
<td>F</td>
<td>⊤</td>
</tr>
<tr>
<td>( 0, - )</td>
<td>SF</td>
<td>⊥</td>
</tr>
<tr>
<td>( 1, + )</td>
<td>G1</td>
<td>⊡</td>
</tr>
<tr>
<td>( -3 )</td>
<td>F1</td>
<td>-</td>
</tr>
<tr>
<td>( -9 )</td>
<td>21</td>
<td>#</td>
</tr>
</tbody>
</table>

Table 1: Symbols of the 9-Valued Algebra.

The first element of the pair, \( \varepsilon \varepsilon ' \), is the value that the variable \( \varepsilon \) assumes in the good circuit; the second element of the pair, \( \varepsilon ' \), is the value that the variable \( \varepsilon \) assumes in the faulty circuit.

The second column of Table 1 reports the names assigned to the nine values by Muth in [21], while the rightmost column shows the symbology used within this paper. Notice that the symbol = represents the logic value Don't Care.

The definition of the basic boolean operators (AND, OR, NOT) is given, using the extended tabular form, in Tables 4, 5, and 6 respectively; in these tables, it is assumed that \( \varepsilon , \varepsilon ', \varepsilon '' \in \{ 0, 1, U, G0, F0, SO, G1, F1, S1 \} \).

The extension of the definition of the AND and the OR operators to the case of n input variables is straightforward, as well as the definition of the complex boolean operators NAND, NOR, EXOR, and EXNOR.

*Enrico Macii is also with the Politecnico di Torino, Dip. di Automatica e Informatica, Torino, ITALY 10129.
3 9-Valued Functions

In this section we present the most important definitions and theorems related to the 9-Valued Algebraic Circuit Model that we have developed for the modification of the ATPG procedure.

Definition 3.1 (Generalized Term)
A Generalized Term (GT) is the product of one or more 9-Valued variables.

Definition 3.2 (Elementary Implication)
Let \( x \) and \( y \) be two 9-Valued variables. We can write \( x = [\alpha, \beta] \), \( y = [\gamma, \delta] \) where \( \alpha, \beta, \gamma, \delta \in \{0,1,-\} \). Accordingly with the rules stated in Table 1, we say that \( x \) implies \( y \), and we write \( x \Rightarrow y \), if and only if \( \alpha \Rightarrow \gamma \) and \( \beta \Rightarrow \delta \).

\[
\begin{array}{ccc}
\alpha & \Rightarrow & \gamma \\
\Rightarrow & - & \\
\beta & \Rightarrow & \delta \\
\Rightarrow & - \\
\end{array}
\]

Table 2: Elementary Implication (\( \sigma \in \{0,1,-\} \)).

Definition 3.3 (Ungeneralized Function)
An Ungeneralized Function (UF) is a two-valued boolean function ascribed to the 9-Valued Algebra. We denote with \( f \) the UF that assumes the value 1, and with \( \bar{f} \) the complement of \( f \).

Definition 3.4 (Generalized Function)
A Generalized Function (GF) is a function that can assume all the values of the 9-Valued Algebra defined previously. Usually, a GF is represented as a sum of one or more GTs. Given the UF's \( f \) and \( \bar{f} \), there are eight possible GFs:

- \( F = \{f, \bar{f}\} \);
- \( F = \{f, \bar{f}\} \);
- \( F = \{f, \bar{f}\} \);
- \( F = \{f, \bar{f}\} \);
- \( F = \{f, \bar{f}\} \);
- \( F = \{f, \bar{f}\} \);
- \( F = \{f, \bar{f}\} \).

Definition 3.5 (Implication)
Let \( t_1 \) and \( t_2 \) be two GTs. We have that \( t_1 \) implies \( t_2 \), and we write \( t_1 \Rightarrow t_2 \), if and only if \( t_2 \) assumes the value \([1,1]\) when \( t_1 \) assumes the value \([1,1]\).

Definition 3.6 (Generalized Implicant)
A GT, \( t \), is a Generalized Implicant (GI) of a GF, \( F \), if and only if \( t \Rightarrow F \).

Definition 3.7 (Generalized Prime Implicant)
A GI, \( t \), of a GF, \( F \), is a Generalized Prime Implicant (GPI) of \( F \) if and only if \( t \) does not imply any other GI of \( F \).

Theorem 3.1
If \( t = abc \ldots \) is a GPI of a GF, \( F = \{f, \bar{f}\} \), with \( a = [\alpha, \alpha^*] \), \( b = [\beta, \beta^*] \), \( c = [\gamma, \gamma^*] \), then \( \theta = \alpha \beta \gamma \ldots \) and \( \theta^* = \alpha^* \beta^* \gamma^* \ldots \) are Prime Implicants (PI's) of the UF \( F \), and vice versa.

Theorem 3.2
If \( t = abc \ldots \) is a GPI of a GF, \( F = \{f, \bar{f}\} \), with \( a = [\alpha, \alpha^*] \), \( b = [\beta, \beta^*] \), \( c = [\gamma, \gamma^*] \), then \( \theta = \alpha \beta \gamma \ldots \) and \( \theta^* = \alpha^* \beta^* \gamma^* \ldots \) are Prime Implicants (PI's) of the UF \( F \), and vice versa.

Theorem 3.3
If \( t = abc \ldots \) is a GPI of a GF, \( F = \{f, \bar{f}\} \), with \( a = [\alpha, \alpha^*] \), \( b = [\beta, \beta^*] \), \( c = [\gamma, \gamma^*] \), then \( \theta = \alpha \beta \gamma \ldots \) is a PI of the UF \( F \) and \( \theta^* = \alpha^* \beta^* \gamma^* \ldots \) is a PI of the UF \( F \), and vice versa.

Theorem 3.4
If \( t = abc \ldots \) is a GPI of a GF, \( F = \{f, \bar{f}\} \), with \( a = [\alpha, \alpha^*] \), \( b = [\beta, \beta^*] \), \( c = [\gamma, \gamma^*] \), then \( \theta = \alpha \beta \gamma \ldots \) is a PI of the UF \( F \) and \( \theta^* = \alpha^* \beta^* \gamma^* \ldots \) is a PI of the UF \( F \), and vice versa.

Theorem 3.5
If \( t = abc \ldots \) is a GPI of a GF, \( F = \{f, \bar{f}\} \), with \( a = [\alpha, \alpha^*] \), \( b = [\beta, \beta^*] \), \( c = [\gamma, \gamma^*] \), then \( \theta = \alpha \beta \gamma \ldots \) is a PI of the UF \( F \) and \( \theta^* = \alpha^* \beta^* \gamma^* \ldots \) is a PI of the UF \( F \), and vice versa.

Theorem 3.6
If \( t = abc \ldots \) is a GPI of a GF, \( F = \{f, \bar{f}\} \), with \( a = [\alpha, \alpha^*] \), \( b = [\beta, \beta^*] \), \( c = [\gamma, \gamma^*] \), then \( \theta = \alpha \beta \gamma \ldots \) is a PI of the UF \( F \) and \( \theta^* = \alpha^* \beta^* \gamma^* \ldots \) is a PI of the UF \( F \), and vice versa.

4 Test Generation

At an high level of abstraction, the ATPG algorithm presented in [19, 22] can be described as a modified version of the sequential D-Algorithm [24]. In that approach, the Controllability and the Observability Functions of each node of the network are computed, working on a block-level topological description of the circuit. The test generation algorithm can be subdivided in two phases:

- observability phase;
- controllability phase.

During the observability step, the fault is propagated through the circuit until it can be observed on a primary output; the propagation is obtained with the recursive expansion of the Observability Functions of each logic block. When a primary output is reached, the observability phase ends, and it is replaced by the controllability step of the search, which carries out the justification of all the variables that have been employed during the fault propagation. The procedure terminates successfully (i.e., a test pattern is generated) if the justification operation requires a number of time frames that is smaller than a predefined threshold (i.e., the temporal length of the test pattern).

In this section we consider a generic single-output combinational block, and we define the 9-Valued Controllability Functions of the output node and the 9-Valued Observability Functions of a generic input node. These functions can be seen as a more general case of the Controllability and Observability Functions employed in the original ATPG procedure. The modularity of the software allows their direct substitution without altering the main structure of the program.

981
4.1 9-Valued Controllability Functions

The 9-Valued Controllability Functions (CF's) of the block output node are the sets of conditions to be assumed by the input lines of the block to assign to the node itself the values 0 (0-CF), 1 (1-CF), G0 (G0-CF), F0 (F0-CF), S0 (S0-CF), G1 (G1-CF), F1 (F1-CF) and S1 (S1-CF). All those functions can be computed starting from the boolean expression of the block determined during the partitioning step.

Let us consider, for example, a block having three input lines, a, b, and c. Given its UF

\[ f = ab + bc, \]

we want to compute the S1-CF (i.e., the GF \( \tilde{F} \)).

The function \( f \) can be written as a sum of PI, by applying the consensus theorem or some other methods that are commonly used in the automatic minimisation of boolean functions (see, for example, [26]). The result obtained is:

\[ f = ab + bc + ac. \]

From \( f \) we can compute \( f \) that, again, can be expressed as a sum of PI:

\[ f = ab + ac + bc. \]

Now, \( f \) and \( f \) are the UF's of the block; by applying the theorems given in the previous section, we can determine \( \tilde{F} \) as shown in Table 3.

<table>
<thead>
<tr>
<th>PI of f</th>
<th>PI of ( \tilde{f} )</th>
<th>GPI of ( \tilde{F} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a(\bar{b})</td>
<td>a(\bar{b})</td>
<td>a(\bar{b})</td>
</tr>
<tr>
<td>a(\bar{b})</td>
<td>a(\bar{b})</td>
<td>a(\bar{b})</td>
</tr>
<tr>
<td>a(\bar{b})</td>
<td>a(\bar{b})</td>
<td>a(\bar{b})</td>
</tr>
<tr>
<td>a(\bar{b})</td>
<td>a(\bar{b})</td>
<td>a(\bar{b})</td>
</tr>
<tr>
<td>a(b\bar{c})</td>
<td>a(b\bar{c})</td>
<td>a(b\bar{c})</td>
</tr>
<tr>
<td>a(b\bar{c})</td>
<td>a(b\bar{c})</td>
<td>a(b\bar{c})</td>
</tr>
<tr>
<td>a(b\bar{c})</td>
<td>a(b\bar{c})</td>
<td>a(b\bar{c})</td>
</tr>
<tr>
<td>a(b\bar{c})</td>
<td>a(b\bar{c})</td>
<td>a(b\bar{c})</td>
</tr>
</tbody>
</table>

Table 3: Computation of the S1-CF \( \tilde{F} \).

4.2 9-Valued Observability Functions

The 9-Valued Observability Functions (OF's) of a block input node are the sets of values assumed by the other input lines that let the node be observable on the block output, that is, they let the value of the output line be dependent on the signal level of the input node under consideration. We can distinguish between Direct Observability (DO), when the output node assumes exactly the value of the input node under observation, and Complemented Observability (CO), when the output node assumes the complemented value of the input node under observation.

It is well known that, if \( a \) is a generic input node of a block, the UF \( f \) of the output node can be written as

\[ f = af + af + W, \]

where \( U, V \) and \( W \) are boolean functions of the input nodes of the block, node \( a \) excluded.

From this expression of \( f \), it is possible to compute the 9-Valued OF's of the generic node \( a \).

There are eight OF's for each input node of the block, four referred to the Direct Observability and four referred to the Complemented Observability. The 9-Valued OF's for the generic node \( a \) are then GF's, and their expressions, in terms of \( U, V \) and \( W \), are listed below.

- DO-OF (a) = \( U \lor V \lor W \) (DO of the node \( a = 0 \));
- CO-OF (a) = \( U \lor V \lor W \) (CO of the node \( a = 0 \));
- D1-OF (a) = \( U \lor V \lor W \) (DO of the node \( a = 1 \));
- C1-OF (a) = \( U \lor V \lor W \) (CO of the node \( a = 1 \));
- DSO-OF (a) = \( U \lor V \lor W \) (DO of the node \( a = S0 \));
- CS0-OF (a) = \( U \lor V \lor W \) (CO of the node \( a = S0 \));
- DS1-OF (a) = \( U \lor V \lor W \) (DO of the node \( a = S1 \));
- CS1-OF (a) = \( U \lor V \lor W \) (CO of the node \( a = S1 \));

For example, let us consider again the block having three input lines, \( a, b, \) and \( c \), and having

\[ f = ab + bc + ac \]

as the UF of its output node. We want to determine the DS0-OF of the node \( b \).

If we write \( f \) as

\[ f = af + af + W, \]

we obtain that \( U = a, V = c \) and \( W = ac \). Then the expression for the DS0-OF of the node \( b \) results:

\[ DS0-OF (b) = -a c. \]

5 Experimental Results

The following assumptions are made regarding the sequential circuit to be tested.

1. The machine is assumed to not have a reset state. All test vectors are applied with the unknown state as the initial state.
2. The fault model is assumed to be single stuck-at.
3. The memory elements are considered as distinct logic primitives and faults inside the memory elements are not considered. However, all faults on present state and next state wires are considered.
4. The description of the circuit is available at the gate-level.

The ATPG procedure that we have implemented does not use completely the power provided by the 9-Valued Algebraic Circuit Model described above. In fact, the program is able to manage the cases in which the fault propagation and the lines justification operations require to set, at most, two input lines of each block to the values \( G0, F0, S0, G1, F1, S1 \). This assumption simplifies the implementation of the algorithm, because the number of GT forming each GF (i.e., 9-Valued OF's and 9-Valued OF's) is substantially reduced.

The ATPG program has been implemented using the C Programming Language on a DEC-VAX 9000 computer with VMS 5.4. The total amount of source code is about 30,000 lines.

Table 7 shows the results reached running different versions of the ATPG system on a large subset of the ITRI 30 circuits [23]. The first five columns give the circuits' characteristics: Circuit Name (Circ), Number of Gates (Gates), Number of Nodes (Nodes), Number of Flip-Flops (FF's), and Number of Stuck-at Faults considered (Faults). Columns headed by ATPG 1 and ATPG 2 give the Fault Coverage (FC (#)) and Fault Coverage (FC (%)) and the Test Generation time (TG time) presented in [19] and [22] respectively. Finally, columns headed by ATPG 3 show the Fault Coverage (FC (#)) and Fault Coverage (FC (%)) and the Test Generation time (TG time) obtained with the current release of the program.
6 Conclusions and Future Work

A 9-Valued Algebraic Circuit Model has been presented in this paper. The definitions of 9-Valued Algebra and 9-Valued Functions are applied to an existing ATPG procedure in order to allow the management of sequential circuits affected by convergent fanout. The experimental results reached running the test generator on the set of standard benchmark circuits are useful to show the benefits that can be obtained with the introduction of the 9-Valued Model.

The work is still subject to new modification; some topological techniques oriented to the reduction of the test generation time in the circuits containing special geometry in the block-level description have been already studied; the preliminary results produced by the enhanced version of our tool suggest the pursuance of our research.

The next step in the refinement process of our software will be the introduction of a random test generator as a pre-processing routine; the number of faults requiring a targeted test generation should substantially decrease, and the result should be a further reduction of the global test generation time.

Acknowledgements

The interesting discussions with Mike Lightner and Fabio Somenzi, professors at the University of Colorado, on sequential test generation are acknowledged. Moreover, the authors are very grateful to Antonio Lioy and Matteo Sonza Reorda, professors at the Politecnico di Torino, for their valuable suggestions. Finally, Enrico Macii would like to thank Allison Evans, Ph. D. student at the University of California at San Diego (UCSD), for the review of this paper.

References

Table 4: Definition of the 9-Valued AND Operator \((z = x \text{ AND } y)\).

<table>
<thead>
<tr>
<th>(x = 0)</th>
<th>(x = 1)</th>
<th>(x = U)</th>
<th>(x = G)</th>
<th>(x = F)</th>
<th>(y = 0)</th>
<th>(y = 1)</th>
<th>(y = F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>U</td>
<td>U</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>0</td>
<td>G</td>
<td>G</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5: Definition of the 9-Valued OR operator \((z = x \text{ OR } y)\).

<table>
<thead>
<tr>
<th>(x = 0)</th>
<th>(x = 1)</th>
<th>(x = U)</th>
<th>(x = G)</th>
<th>(x = F)</th>
<th>(y = 0)</th>
<th>(y = 1)</th>
<th>(y = F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>U</td>
<td>U</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>0</td>
<td>G</td>
<td>G</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6: Definition of the 9-Valued NOT Operator \((z = NOT x)\).

<table>
<thead>
<tr>
<th>(x = 0)</th>
<th>(x = 1)</th>
<th>(x = U)</th>
<th>(x = G)</th>
<th>(x = F)</th>
<th>(y = 0)</th>
<th>(y = 1)</th>
<th>(y = F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7: Statistics about some IscAS'89 Circuits (Time in HRS:MIN:SEC).

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Gates</th>
<th>Nodes</th>
<th>FFs</th>
<th>faults</th>
<th>ATPG 1 (FC (s) FC (%) TG time)</th>
<th>ATPG 2 (FC (s) FC (%) TG time)</th>
<th>ATPG 3 (FC (s) FC (%) TG time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S27</td>
<td>10</td>
<td>14</td>
<td>3</td>
<td>82</td>
<td>28</td>
<td>100.0 00:00:04</td>
<td>82</td>
</tr>
<tr>
<td>S306</td>
<td>96</td>
<td>82</td>
<td>2</td>
<td>210</td>
<td>114</td>
<td>68.0 00:10:23</td>
<td>118</td>
</tr>
<tr>
<td>S308</td>
<td>119</td>
<td>55</td>
<td>14</td>
<td>179</td>
<td>133</td>
<td>74.5 00:03:39</td>
<td>139</td>
</tr>
<tr>
<td>S344</td>
<td>160</td>
<td>73</td>
<td>18</td>
<td>324</td>
<td>267</td>
<td>82.6 00:16:01</td>
<td>267</td>
</tr>
<tr>
<td>S449</td>
<td>190</td>
<td>74</td>
<td>15</td>
<td>272</td>
<td>272</td>
<td>81.9 00:27:39</td>
<td>272</td>
</tr>
<tr>
<td>S387</td>
<td>158</td>
<td>45</td>
<td>31</td>
<td>360</td>
<td>310</td>
<td>76.9 00:05:53</td>
<td>375</td>
</tr>
<tr>
<td>S386</td>
<td>189</td>
<td>41</td>
<td>6</td>
<td>384</td>
<td>246</td>
<td>64.1 00:07:55</td>
<td>246</td>
</tr>
<tr>
<td>S449</td>
<td>148</td>
<td>38</td>
<td>21</td>
<td>434</td>
<td>340</td>
<td>80.5 00:07:41</td>
<td>340</td>
</tr>
<tr>
<td>S430</td>
<td>190</td>
<td>99</td>
<td>16</td>
<td>130</td>
<td>130</td>
<td>33.9 00:10:18</td>
<td>130</td>
</tr>
<tr>
<td>S444</td>
<td>181</td>
<td>101</td>
<td>31</td>
<td>474</td>
<td>376</td>
<td>79.3 00:01:50</td>
<td>376</td>
</tr>
<tr>
<td>S110</td>
<td>211</td>
<td>108</td>
<td>31</td>
<td>584</td>
<td>584</td>
<td>00:00:00</td>
<td>584</td>
</tr>
<tr>
<td>S226</td>
<td>194</td>
<td>85</td>
<td>21</td>
<td>555</td>
<td>555</td>
<td>00:11:48</td>
<td>555</td>
</tr>
<tr>
<td>S226m</td>
<td>193</td>
<td>86</td>
<td>31</td>
<td>553</td>
<td>553</td>
<td>00:11:81</td>
<td>553</td>
</tr>
<tr>
<td>S641</td>
<td>379</td>
<td>163</td>
<td>10</td>
<td>465</td>
<td>339</td>
<td>73.9 00:09:16</td>
<td>359</td>
</tr>
<tr>
<td>S113</td>
<td>333</td>
<td>170</td>
<td>19</td>
<td>581</td>
<td>404</td>
<td>69.6 00:12:01</td>
<td>405</td>
</tr>
<tr>
<td>S820</td>
<td>280</td>
<td>80</td>
<td>5</td>
<td>850</td>
<td>661</td>
<td>77.8 13:17:09</td>
<td>663</td>
</tr>
<tr>
<td>S832</td>
<td>287</td>
<td>80</td>
<td>5</td>
<td>870</td>
<td>686</td>
<td>78.8 23:11:50</td>
<td>689</td>
</tr>
<tr>
<td>S833</td>
<td>200</td>
<td>208</td>
<td>32</td>
<td>857</td>
<td>191</td>
<td>33.3 24:31:13</td>
<td>194</td>
</tr>
<tr>
<td>S933</td>
<td>395</td>
<td>238</td>
<td>29</td>
<td>1079</td>
<td>768</td>
<td>78.1 10:44:00</td>
<td>1091</td>
</tr>
<tr>
<td>S1068</td>
<td>529</td>
<td>206</td>
<td>18</td>
<td>1242</td>
<td>1110</td>
<td>89.4 05:22:41</td>
<td>1123</td>
</tr>
<tr>
<td>S1238</td>
<td>608</td>
<td>219</td>
<td>16</td>
<td>1265</td>
<td>1059</td>
<td>87.1 10:44:00</td>
<td>1091</td>
</tr>
<tr>
<td>S1453</td>
<td>677</td>
<td>286</td>
<td>24</td>
<td>1515</td>
<td>906</td>
<td>59.7 16:19:51</td>
<td>906</td>
</tr>
<tr>
<td>S1486</td>
<td>633</td>
<td>120</td>
<td>6</td>
<td>1486</td>
<td>1383</td>
<td>83.9 02:22:50</td>
<td>1339</td>
</tr>
<tr>
<td>S1494</td>
<td>667</td>
<td>120</td>
<td>6</td>
<td>1506</td>
<td>1382</td>
<td>81.8 02:21:12</td>
<td>1322</td>
</tr>
<tr>
<td>S378</td>
<td>379</td>
<td>118</td>
<td>179</td>
<td>4803</td>
<td>3280</td>
<td>85.1 24:38:42</td>
<td>3280</td>
</tr>
</tbody>
</table>