A Comparison of Timed State Space Analysis Methods

A Comparison of Timed State Space Analysis Methods

Scott R. Little
Advisor: Chris J. Myers

June 16, 2003

Contents

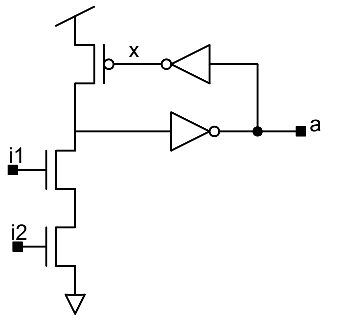
1 Introduction

Circuit designers continue to push the limits of high speed circuit design. This desire for speed motivates the creation of new and innovative design styles. One of these design styles is timed circuits. Timed circuits take advantage of timing information to increase performance. This style has been applied in industrial research to designs like IBM's guTS microprocessor [1], the Intel RAPPID project [2], and Sun's GasP circuits [3]. These experimental designs were successful at increasing performance, but they are only experimental designs. Timed circuits have not yet been used in a commercial design.

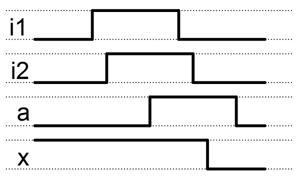
One problem that plagues timed circuit design is the difficulty of understanding the complex timing interactions between circuit components. In response to this problem, researchers have created several methods and tools to help verify timed circuit designs [4, 5, 6, 7, 8, 9, 10]. One of the most critical aspects of these methods is state space exploration of both the timed and untimed state space. This work concentrates on two leading methods to do timed state space exploration using Petri nets. These methods were previously implemented in two different CAD tools, ATACS [8, 9, 10] and VINAS-P [11, 12]. However, a clear comparison of the methods has not been done due to the fact that the methods were implemented in different tools. We extended the ATACS framework by adding the methods previously only used by VINAS-P. Having both methods implemented in the same tool allows us to do an accurate comparison of the methods to better understand the strengths and weaknesses of each method.

2 Timed Circuits

Timed circuits are a class of digital circuits that typically operate without the aid of a global clock. To compensate for the loss of the global clock, timing information and relationships



(a)



(b)

Figure 1: (a) Self-resetting AND gate. (b) Correct timing for the self-resetting AND gate.

are used to correctly order events in the circuit and increase performance. The running example of a timed circuit used in this thesis is the self-resetting AND gate shown in Figure 1(a). One of the unique characteristics of this circuit is that instead of producing logic levels on the output the circuit produces pulses. The length of the output pulse is determined by the delay through the inverter feedback path. We demonstrate the correct operation of this circuit using the waveform in Figure 1(b). The inputs, *i1* and *i2*, go high one right after the other. This action turns on both NMOS transistors creating a path between the inverter feeding node a and ground. This conduction path eventually causes the output a to go high. Shortly after a goes high the inputs are brought low. The high signal produced by the lower inverter connected to a feeds back around through the upper inverter which turns on the PMOS transistor connected to node x and brings a low.

There are a number of potential problems with this circuit if incorrect timing is permitted to occur. The waveform in Figure 2(a) represents a possible error condition that could result from a glitch on one of the inputs. In synchronous designs, glitching is allowed if the signals are stable a hold time before the arrival of the clock signal. Timed circuits do not have the luxury of allowing glitches because a global clock is not used to create snapshots of the circuit state. A glitch at any time may result in a full output pulse when no output pulse is desired. Hold violations can also

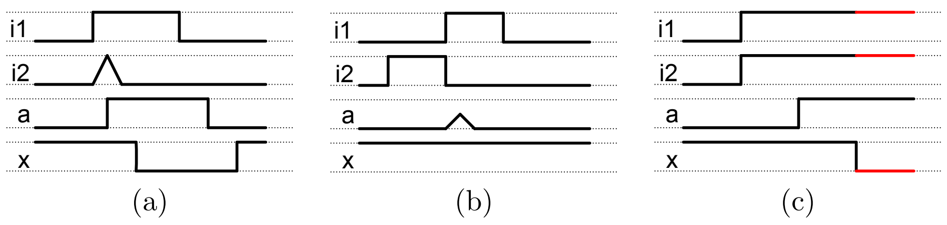


Figure 2: (a) Timing of a glitch for a self-resetting AND gate. (b) Timing of a hold violation for a self-resetting AND gate. (c) Timing of a short for a self-resetting AND gate.

plague timed circuits. The waveform in Figure 2(b) shows the results of incorrect timing between the input pulses. If the input pulses are not correctly aligned, a runt pulse could be produced instead of the desired full pulse. Another possible problem is shown in Figure 2(c). If the designer is not careful in designing the timing constraints for the falling of the inputs, a short circuit may result. The short circuit occurs when the inputs are not brought low before the high output from a is fed back around through the inverter to node z. This can easily be prevented by guaranteeing that the inputs will be low before z goes low. Understanding the correct operation and possible points of failure of timed circuits is a complex problem even for a simple circuit such as the self-resetting AND gate. For large, industrial size circuits this problem becomes much too complex for paper and pencil methods. A CAD tool is required to verify the operation of these large designs.

3 Verification

Verification is a method to exhaustively examine a design and check to make sure certain predefined properties are met. Verification could be thought of as a form of exhaustive validation, but for industrial size designs exhaustive validation is simply not possible while verification may be possible. It is for this reason that verification is being used more and more in industry. It is especially important in areas where failures may have catastrophic consequences. Verification must be carefully understood though. It is not a silver bullet to solve all validation problems. It only checks very specific properties and even then may not give complete coverage. Verification is a powerful tool, but must be used with understanding and care.

In the verification of a timed circuit the entire timed and untimed state space of the circuit is explored in what is called a reachability analysis. The reachability analysis allows the tool to check various properties in every single possible state of the circuit. Finding all of the possible states in a circuit is referred to as state space exploration. State space exploration is quite problematic because the state space grows exponentially with respect to the number of input signals in a design. This fact makes the state space exploration of industrial size circuits impossible without using clever techniques to reduce the state space or abstracting out portions of the design and operating on these smaller abstracted pieces individually.

4 Petri Nets

As circuit designers, we like to think about how the circuit is created with transistors and wires, but to do state space exploration a more formal model must be used. The model that we have chosen to use is a variant of traditional Petri nets called a Time Petri net (TPN). A traditional Petri net is a bipartite graph consisting of places and transitions. Our model simply adds timing bounds on the transitions that further constrains their firing.

An example TPN is shown in Figure 3. A TPN is composed of places, tokens, transitions, events, and timing bounds. The places are the circles and tokens are the black dots that can be contained in the places. A place containing a token is referred to as a marked place. The transitions are the thick black lines. Each transition has an event and timing bound associated with it. The event indicates how the signal changes when the transition fires. A "+" following a signal name indicates that the signal goes from a low logic level to a high logic level when that transition fires while a "−" indicates that the signal will change from a high logic level to a low logic level upon

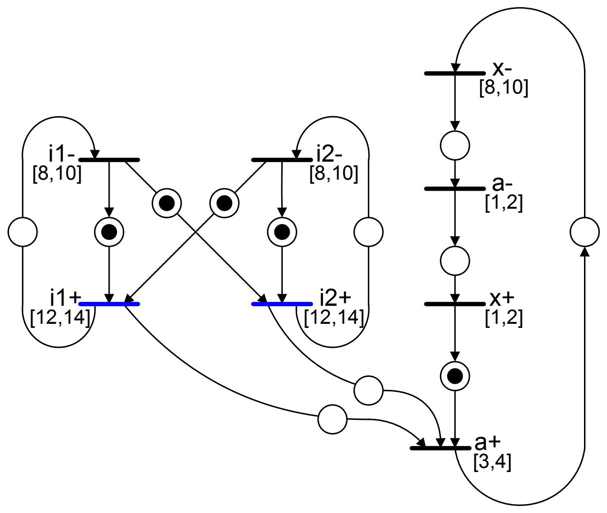


Figure 3: A Time Petri net.

firing. The timing bound is a pair of integers. The first integer represents the lower bound for the firing time and the second integer represents the upper timing bound for the firing time.

A transition is said to be enabled when all of the places in all of its incoming transitions, the preset, are marked. For instance, in Figure 3 only transitions *i1+* and *i2+* are enabled. Even though the transition has become enabled it may not be firable. To be firable a transition must be continuously enabled for at least as long as the lower timing bound, but no longer than the upper timing bound. At any time when a transition is firable it may fire. Upon firing, the tokens are removed from the preset of the transition and placed in the output places or postset of the transition. This can be seen as transition *i1+* fires from Figure 3 to create the Petri net in Figure 4. Using these rules, the entire state space of a design is explored via the Petri net model.

5 Zones

Zones are a mechanism used to track the time separation between different events in the circuit. It is by means of the zone that we determine which events are firable. Zones are simply

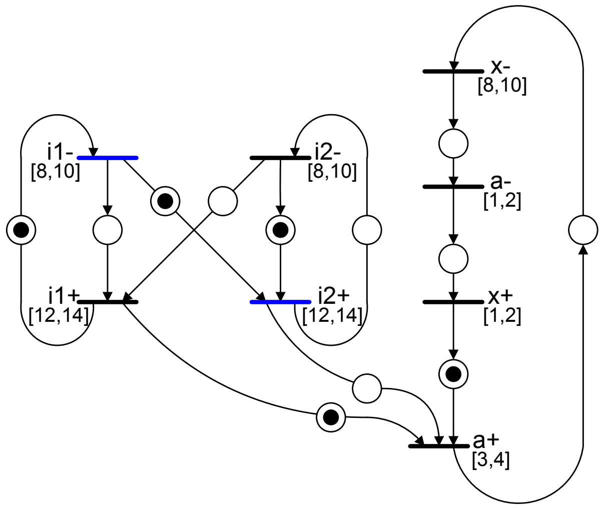


Figure 4: Step 2 in the Petri net example.

collections of inequalities and can be represented by a difference bound matrix. We can examine the zone and get a better understanding of how it works by applying the VINAS-P methodology to the self-resetting AND gate example. The initial zone for the Petri net in Figure 3 is shown in Equation 1. The zone is a set of inequalities representing the currently enabled events. The value τ_{i1+} represents the firing time of *i1*+ and τ_{i2+} represents the firing time of *i2*+. In other words, Equation 1 gives the possible differences in firing times between *i1*+ and *i2*+. By examining the zone a bit more carefully we can see that based on the inequalities *i1*+ and *i2*+ can fire in either order up to two time units apart. Therefore, *i1*+ and *i2*+ are both firable.

$$-2 \leq \tau_{i1+} - \tau_{i2+} \leq 2 \tag{1}$$

To move the example forward, we fire *i1*+ which results in the Petri net shown in Figure 4. In this net, we see that *i1*+ has fired which enables *i1*−. *i2*+ is still enabled. By looking at the zone shown in Equation 2, we can see that the only firable event is *i2*+. This can easily be seen by

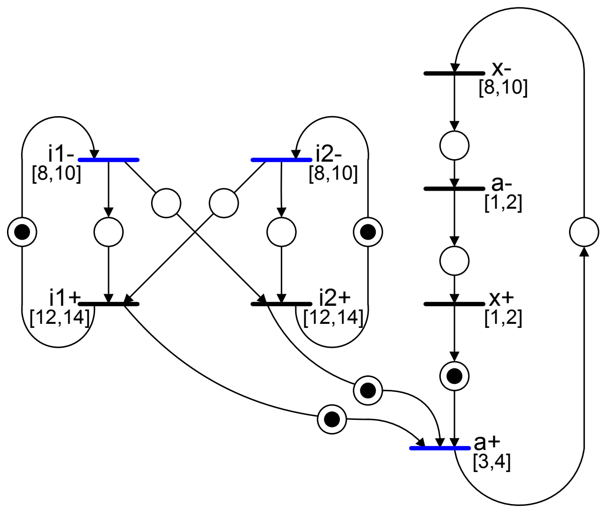


Figure 5: Step 3 in the Petri net example.

noting that both bounds in the zone are positive indicating that the firing time of *i12+* is always less than the firing time of *i1−*. If *i2+* must fire first, then it is the only firable event. We fire *i2+* which results in the Petri net show in Figure 5.

$$6 \leq \tau_{i1-} - \tau_{i2+} \leq 10 \tag{2}$$

In this state, *i1−*, *i2−*, and *a+* are enabled. Looking at the zone again, we can see that *a+* must fire before both *i1−* and *i2−*. The rest of the timed state space exploration proceeds in this manner.

$$\begin{aligned} -4 \leq \tau_{i1-} - \tau_{i2-} \leq 2 \\ 2 \leq \tau_{i1-} - \tau_{a+} \leq 7 \\ 4 \leq \tau_{i2-} - \tau_{a+} \leq 7 \end{aligned} \tag{3}$$

6 Timed State Space Analysis Methods

VINAS-P and ATACS each employ different methods to explore the timed state space. VINAS-P uses a future variable sequence based timing [7] approach to state space exploration while ATACS uses a past variable POSET based timing [10] approach. In reality these tools use a two step approach to timed state space analysis. The first step is determining which type of events to put in the zone and the second step is deciding on how to handle timing within the zone. The methods for determining which type of events are contained within the zone are the future variable method and past variable method. The methods for handling timing within the zone are sequence based timing and POSET timing. By implementing both approaches in ATACS we were able to compare all four of the possible interleavings instead of just the two approaches previously examined.

6.1 VINAS-P

The future variable method uses zones composed of transitions that are enabled to fire as demonstrated by the example in the previous section. This method is very straightforward to implement. The determination of causality is rather simple because the event that just fired and enabled the new event is certainly causal. This leads to a simpler algorithm that tends to be more robust and easier to debug. Those are important characteristics, but the algorithm must also be efficient.

Sequence based timing forces strict firing orders to be imposed in the zone. This creates a new zone for each different possible firing order. This can be thought of as a total order exploration of the timed state space.

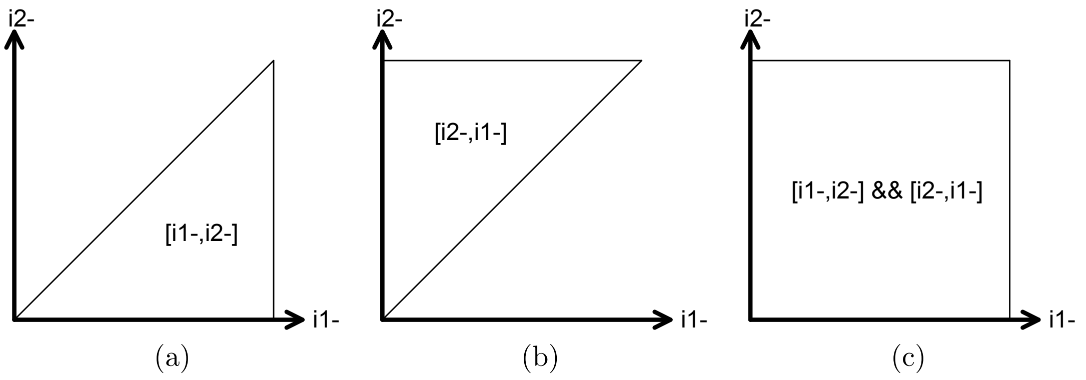


Figure 6: (a) Sequence based timing where *i1−* fires first. (b) Sequence based timing where *i2−* fires first. (c) POSET timing.

6.2 ATACS

The past variable method uses zones composed of transitions that have previously fired. This method tends to be the more complicated of the two methods because causality must be computed to determine which event should fire next. It is possible that the causality may be uncertain and therefore this method may lead to more zones because some events may be kept in the zone when they are not required.

POSET timing does not require strict ordering to be imposed in the zone. The idea behind the removal of the strict ordering is to allow the zone to represent more concurrency. This can result in fewer zones because a single zone can possibly contain multiple behaviors. Exploring fewer zones leads to a faster run time. The differences between these two approaches can best be understood by looking at Figure 6. The zones in Figure 6(a) and 6(b) show the zones that would result from a sequence based method exploring both possible firing orders of *i1−* and *i2−*. The zone found by POSET timing shown in Figure 6(c) actually includes both interleavings in a single zone. This zone represents the same amount of state space that is represented by two zones in the sequence based approach. This is due to the fact that strict timing is not required in the zone.

To provide a bit more intuition about the approaches used by VINAS-P and ATACS we can

ATACS	VINAS-P
$0 \leq \tau_{i1-} - \tau_{i2-} \leq 0$	$-2 \leq \tau_{i1+} - \tau_{i2+} \leq 2$
$0 \leq \tau_{i1-} - \tau_{x+} \leq 0$	
$0 \leq \tau_{i2-} - \tau_{x+} \leq 0$	

Table 1: Initial zones.

ATACS	VINAS-P
$-14 \leq \tau_{i1-} - \tau_{i1+} \leq -12$	$-2 \leq \tau_{i1+} - \tau_{i2+} \leq 2$
$-14 \leq \tau_{i2-} - \tau_{i1+} \leq -12$	
$0 \leq \tau_{i1-} - \tau_{i2-} \leq 0$	

Table 2: Zones after *i1+* fires.

examine the first few zones explored by both tools for the self-resetting AND gate example. Table 1 shows the initial zones found by both tools. Immediately we observe that ATACS generates a more complex zone. The zone found by ATACS includes all of the possible events that would have fired just before arriving in the initial state of the Petri net shown in Figure 3. VINAS-P only requires the enabled events to be in the zone. As we move forward firing *i1+* the zones shown in Table 2 are found. Again VINAS-P finds a simpler zone because fewer variables are required to represent what is currently enabled than are required to represent what could have already fired. Finally as we move forward by firing *i2+*, the zones are of equal complexity as shown in Table 3. This gives us an indication that the method used by VINAS-P results in zones that are the same size or smaller than the method used by ATACS.

ATACS	VINAS-P
$-6 \leq \tau_{i1+} - \tau_{a+} \leq -3$	$-4 \leq \tau_{i1-} - \tau_{i2-} \leq 2$
$-2 \leq \tau_{i1+} - \tau_{i2+} \leq 2$	$2 \leq \tau_{i1-} - \tau_{a+} \leq 7$
$-6 \leq \tau_{i2+} - \tau_{a+} \leq -3$	$4 \leq \tau_{i2-} - \tau_{a+} \leq 7$

Table 3: Zones after *i2+* fires.

7 Results

To compare the different methods, an example suite consisting of 121 examples is run on each of the different methods and critical performance information is output and collected. We will use zone count as our metric of comparison. We choose this metric because zone count is a good indicator of the ability of the method to efficiently verify a given example. To help simplify the analysis we normalize the zone counts to the past variable sequence based approach. Table 4 shows the results of ten examples that are representative of the differences found in the entire example suite.

	PV Sequence (New)	FV Sequence (VINAS-P)	PV POSET (ATACS)	FV POSET (New)
scsiSVN4	1.00	0.11	0.74	1.53
choice2	1.00	0.72	0.72	0.83
SEL	1.00	0.87	0.86	0.91
ifreq1	1.00	0.71	0.57	0.60
slu	1.00	0.59	0.59	0.59
etlatch	1.00	0.62	0.62	0.62
fib	1.00	0.62	0.63	0.43
lapbN	1.00	0.50	0.61	0.48
srdaoi	1.00	0.62	1.00	0.62
mul2c	1.00	0.51	0.50	0.47

Table 4: Normalized zone counts.

By examining these examples, we can understand some trends that will hold true throughout the entire example suite. If we compare the past variable sequence based approach to the future variable sequence based approach we can see that the future variable approach is always better and in some cases significantly better. When comparing the past variable sequence based approach to the past variable POSET based approach we determine that the POSET based approach is as good or better than the sequence based approach. When we start comparing the future variable POSET based approach a slightly different behavior is observed. It is observed that combining the future variable approach with POSET timing yields good results in many cases. There are a few notable cases where this is not true. The future variable method is superior, but when combined with POSET timing unexpected results can occur.

These claims can also be verified by looking at the results in a different way as shown in Table 5. In each table entry, the numerator in the fraction is the number of times when the method in the row finds fewer zones than the method in the column. The denominator represents the total number of zone differences between the methods in the row and column when compared over all 121 examples. When looking at the data in this way it is clear that the past variable sequence based method performs quite poorly. We also see the power of the future variable method as the future variable sequence based method stands up well against the past variable POSET based method. Only examining the number of examples with lower zone counts the future variable POSET based method seems to stand up well against the competition. Simply looking at the number of examples with a lower zone count may be misleading because it does not indicate how much better or worse the method performs on the examples. To help us understand this behavior, each entry contains a percentage value. This percentage represents the change in zone count for the method in the row compared against the method in the column. Looking at this data we can see that indeed the gains really do come from the future variable method. The future variable method combined with POSET timing does not fair well against the future variable sequence based method because even though the future variable POSET based method has lower zone counts on more examples when it has a zone count that is higher it is often much higher.

8 Conclusion

The goal of this work is to compare the various methods for timed state space analysis. The results show some expected as well as unexpected results. It is now clear that the future variable

	PV Sequence	FV Sequence	PV POSET	FV POSET
PV Sequence	-	0/62 (48.13%)	5/62 (22.61%)	4/61 (27.05%)
FV Sequence	62/62 (-22.32%)	-	25/41 (-3.51%)	12/33 (-2.73%)
PV POSET	57/62 (-15.10%)	16/41 (34.39%)	-	12/42 (10.38%)
FV POSET	57/61 (-15.61%)	21/33 (65.82%)	30/42 (-2.55%)	-

Table 5: Comparison of zone differences across all 121 examples and methods.

method is superior to the past variable method. It is also clear that it is useful to apply POSET timing to past variable methods. The area that does require more exploration is when the future variable method is combined with POSET timing. In many cases this does yield some gains, but it also yields some big losses. This happens because the future variable method by nature yields very small zones. When POSET timing is introduced more variables are required in the zone to track the concurrency. These added variables occasionally result in more zones to be explored. Further refinements or optimizations to the POSET timing method should be explored when POSET timing is applied to the future variable method.

References

[1] H. P. Hofstee, S. H. Dhong, D. Meltzer, K. J. Nowka, J. A. Silbermna, J. L. Burns, S. D. Posluszny, and O. Takahashi, "Designing for a gigahertz," *IEEE Micro*, vol. 18, no. 3, pp. 66–74, 1998.

[2] S. Rotem, K. Stevens, R. Ginosar, P. Beerel, C. Myers, K. Yun, R. Kol, C. Dike, M. Roncken, and B. Agapiev, "RAPPID: An asynchronous instruction length decoder," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 60–70, Apr. 1999.

[3] I. Sutherland and S. Fairbanks, "GasP: A minimal FIFO control," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 46–53, IEEE Computer Society Press, Mar. 2001.

[4] C. Daws, A. Olivero, S. Tripakis, and S. Yovine, "The tool kronos," in *Proceedings of Hybrid Systems III, Verification and Control*, vol. 1066 of *Lecture Notes in Computer Science*, pp. 403–415, Springer-Verlag, 1996.

[5] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL — a Tool Suite for Automatic Verification of Real–Time Systems," in *Proc. of Workshop on Verification and*

Control of Hybrid Systems III, no. 1066 in Lecture Notes in Computer Science, pp. 232–243, Springer–Verlag, Oct. 1995.

[6] T. G. Rokicki, *Representing and modeling digital circuits.* PhD thesis, Stanford University, 1994.

[7] T. Yoneda and H. Ryu, "Timed trace theoretic verification using partial order reduction," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 108–121, Apr. 1999.

[8] W. J. Belluomini and C. J. Myers, "Timed state space exploration using posets," *IEEE Transactions on Computer-Aided Design*, vol. 19, pp. 501–520, May 2000.

[9] W. J. Belluomini and C. J. Myers, "Timed circuit verification using tel structures," *IEEE Transactions on Computer-Aided Design*, vol. 20, pp. 129–146, Jan. 2001.

[10] E. Mercer, *Correctness and Reduction in Timed Circuit Analysis.* PhD thesis, University of Utah, 2002.

[11] T. Yoneda, "VINAS-P: A Tool for Trace Theoretic Verification of Timed Asynchronous Circuits," in *Proc. International Workshop on Computer Aided Verification*, vol. 1855 of *Lecture Notes in Computer Science*, pp. 572–575, Springer-Verlag, 2000.

[12] T. Kitai, Y. Oguro, T. Yoneda, E. G. Mercer, and C. J. Myers, "Level oriented formal model for asynchronous circuit verification and its efficient analysis method," in *Proceedings of Pacific Rim International Symposium on Dependable Computing*, pp. 210–219, Dec. 2002.