16 Two bit adding unit

16.1 General Information

The problem is a stiff DAE of index 1, consisting of 175 differential equations and 175 algebraic equations. It has been contributed by M. Günther [Gün95, Gün98].

The software part of the problem is in the file tba.f available at [MM08].

16.2 Mathematical description of the problem

The problem is of the form

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= f(t,x), \\ 0 &= y - g(x), \end{aligned} \tag{II.16.1}$$

where

$$y, x \in \mathbb{R}^{175}, \quad f : \mathbb{R}^{351} \to \mathbb{R}^{350}, \; g : \mathbb{R}^{350} \to \mathbb{R}^{350}, \quad 0 \le t \le 320, \quad y(0) = y_0, \; x(0) = x_0.$$

Since the functions $f(t,x)$ and $g(x)$ and the (consistent) initial values y_0 and x_0 are too voluminous to be printed here, we refer to the subroutines `feval` and `init` for their definitions. The function f has discontinuities in its derivative at $t = 0, 5, 10, \ldots, 320$. The index of the components of x and y equals 1.

The function f contains several square roots. It is clear that the function can not be evaluated if one of the arguments of one of these square roots becomes negative. To prevent this situation, we set `IERR=-1` in the Fortran subroutine that defines f if this happens. See page *IV-ix* of the description of the software part of the test set for more details on `IERR`.

16.3 Origin of the problem

The two bit adding unit computes the sum of two base-2 numbers (each two digits long) and a carry bit. These numbers are fed into the circuit in the form of input signals. As a result the circuit gives their sum coded as three output signals.

The two bit adding unit circuit is a digital circuit. These circuits are used to compute boolean expressions. This is accomplished by associating voltages with boolean variables. By convention the boolean is true if the voltage exceeds $2V$, and false if it is lower than $0.8V$. In between the boolean is undefined. Using CMOS technique, however, sharper bounds are possible for the representation of booleans.

Digital circuits that compute elementary logical operations are called gates. An example of a gate is the NAND gate of test problem 9. This circuit is used to compute the logical expression $\neg(V_1 \wedge V_2)$, where V_1 and V_2 are the booleans that are fed into the circuit as input signals.

The two bit adding unit is depicted in Figure II.16.1. In this figure the symbols '&', '≥ 1' and a little white circle respectively stand for the AND, OR and NOT gate. A number of input signals and output signals enter and leave the circuit. Each signal is described by a time-dependent voltage and the boolean it represents. For these two quantities we shall use one symbol: the symbol of this boolean variable. Which one of the two quantities is meant by the symbol, is always clear from the context. With this convention, the input signals are referred to by the boolean variable they represent.

The circuit is designed to perform the addition

$$A_1 \; A_0 + B_1 \; B_0 + C_{in} = C \; S_1 \; S_0.$$

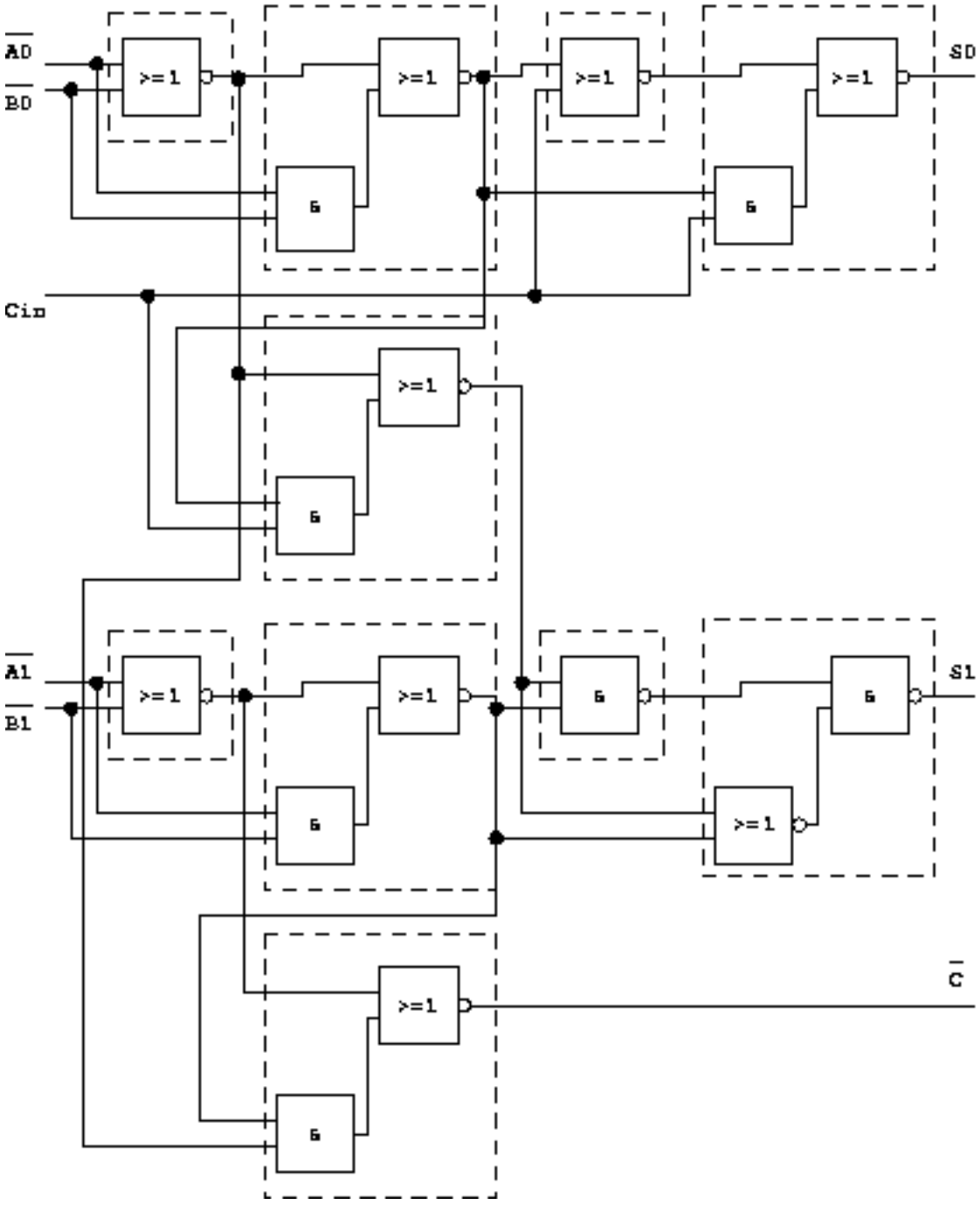

FIGURE II.16.1: *Circuit diagram of the two bit adder (taken from [Gün95]).*

TABLE II.16.1: *Characteristics of the gates that occur in the two bit adding unit.*

Name	logical expression	# nodes	# times
NOR	$\neg(V_1 \vee V_2)$	$3 \cdot 4 + 1 = 13$	3
NAND	$\neg(V_1 \wedge V_2)$	$3 \cdot 4 + 2 = 14$	1
ANDOI	$\neg(V_1 \vee (V_2 \wedge V_3))$	$4 \cdot 4 + 2 = 18$	5
ORANI	$\neg(V_1 \wedge (V_2 \vee V_3))$	$4 \cdot 4 + 2 = 18$	1

The input signals representing the two numbers and the carry bit C_{in} are fed into the circuit at the nodes indicated by $\overline{\text{A0}}$, $\overline{\text{A1}}$, $\overline{\text{B0}}$, $\overline{\text{B1}}$ and Cin. Here, a bar denotes the logical inversion. The output signals are delivered by the nodes indicated by S0, S1 and $\overline{\text{C}}$.

In Figure II.16.1, a number of boxes are drawn using dashed lines. Each of them represents one of the following gates: the NOR (first box to the left in the top-row), the ORANI gate (the box besides S_1), the NAND (the box besides the ORANI gate) and the ANDOI(the box at the bottom). The circuit diagram of the NAND-gate is given in test problem 9. For the circuit diagrams of the NOR, ANDOI and ORANI gate see Figures II.16.2, II.16.3 and II.16.4. What logical expressions they compute, is listed in Table II.16.1. The fourth column in this table lists the number of times the gate occurs in the big circuit. The third column tabulates the number of nodes in the gate. These nodes consist of two types. The first type of nodes consists of the internal nodes of the transistors due to the MOS transistor model of Shichmann and Hodges [SH68]. Each transistor has four internal nodes that are also the links between transistor and the rest of the circuit. The second type of nodes comprises the usual nodes that are used to link circuit components together. These nodes are indicated by a number placed inside a square. To prevent any misunderstanding, we remark that the big dots in Figures II.16.2–II.16.4 do not represent nodes.

The connection of a gate with the rest of the circuit consists of the input nodes and the output node of the gate. The input signals enter the gate at the nodes with symbol V_1, V_2 and V_3. The output signal leaves the gate from one of the numbered nodes. To ensure stability of the circuit, such an output node is always connected to a capacitance (we refer to the Fortran driver: `CLOAD` denoting the value of a load capacitance for the logical gates, and `COUT` for the output nodes S_0, S_1 and $\overline{C}$). Finally, three enhancement transistors are coupled with the ANDOI gate at the bottom for a correct treatment of C_{in}. This yields 12 internal nodes and two additional nodes, because the three transistors are coupled in series. Counting all nodes we have $3 \cdot 13 + 1 \cdot 14 + 5 \cdot 18 + 1 \cdot 18 + 14 = 175$ nodes.

Applying Kirchoff's law to all nodes yields a system of 175 equations. This system is an integral form DAE of the special form

$$A \cdot \dot{q}(V) = f(t, V).$$

The function q is a generally nonlinear function of node potentials V, which describes the charges stored in all charge storing elements [GDF96]. Assembling the charge flow at each node by an incidence matrix A, the dynamic part $A \cdot \dot{q}(V)$ equals the contribution of static currents denoted by $f(t, V)$. If all load capacitances at the output nodes are nonzero, then the integral form DAE has differential index 0. If only one of the load capacitances equals zero, the generalized capacitance matrix $A \cdot \partial q(V)/\partial V$ is singular, yielding a system of differential index 1. This shows the regularization effects by applying additional capacitances. Here, we use `CLOAD=0` and `COUT=2.0`.

To make this problem suitable for the solvers used in this test set, the variable $Q = A \cdot q(V)$ of assembled charges is introduced leading to

$$\begin{aligned} \dot{Q} &= f(t, V), \\ 0 &= Q - Aq(V). \end{aligned}$$

This transformation of the integral form DAE into a linearly implicit system raises the differential

index by one. However, in the case of singular load capacitances, no higher index effects are detected in the sense of an appropriate perturbation index [Gün98].

Some of the 175 variables have a special meaning. These are the voltage variables of the nodes that deliver the output signals. The output signals S_0, S_1 and $\overline{C}$ are given by the variables x_{49}, x_{130} and x_{148}, respectively. Only these variables are of interest to the engineer.

In the next section we shall see the two bit adder in operation. Every 10 units of time the addition

$$A_1\ A_0 + B_1\ B_0 + C_{in} = C\ S_1\ S_0,$$

is carried out. The numbers that are added are represented by the input signals depicted in Figure II.16.5. The outcome of the addition is represented by output signals given in Figure II.16.6. Often the output signals need time to adjust to changes in the input signal. Therefore, only during certain periods the sum is correctly represented by the output signals. The two bit adding unit has been designed in such a way that after each 10 units of time the output signal represents the sum correctly.

To see the two bit adding unit performing an addition let us see what happens at $t = 200$. Then the input signals read:

$$\overline{A}_0 = 0,\ \overline{A}_1 = 1,\ \overline{B}_0 = 0, \overline{B}_1 = 0,\ C_{in} = 1,$$

and the output signals are

$$S_0 = 1,\ S_1 = 0,\ \overline{C} = 0.$$

Recall, that a bar denotes the logical inverse. Clearly, the addition 01+11+1=101 has been carried out.

16.4 Numerical solution of the problem

M. Günther provided the source code that defines the problem.

Table II.22.2 lists the voltages of the output signals in the reference solution. For the complete reference solution at $t = 320$ we refer to subroutine `solut`. Since these components refer to the output

TABLE II.16.2: *Value at the end of the integration interval of the components of the reference solution that correspond to the output signals.*

x_{49}	0.2040419147264534
x_{130}	$0.4997238455712048 \cdot 10$
x_{148}	0.2038985905095614

signals S_0, S_1 and $\overline{C}$, they are the physically relevant quantities.

Table II.16.4 and Figures II.16.6–II.16.10 present the run characteristics, the behavior of the output signals over the integration interval and the work-precision diagram, respectively. In computing the scd values, only x_{49}, x_{130} and x_{148} were considered, since they refer to the physically important quantities.

The reference solution was computed using RADAU5 without restarts in the discontinuities in time of the derivative of the problem defining function f, with rtol = atol = 10^{-5} and h0 = $4 \cdot 10^{-5}$.

For the work-precision diagram, we used: rtol = $10^{-(2+m/8)}$, $m = 0, 1, \ldots, 32$; atol = rtol; h0 = $10 \cdot$ rtol for BIMD, GAMD, MEBDFDAE, MEBDFI, RADAU and RADAU5. The failed runs are in Table II.16.3; listed are the name of the solver that failed, for which values of m this happened, and the reason for failing.

TABLE II.16.3: *Failed runs.*

solver	m	reason
BIMD	$27, \ldots, 32$	more than nmax steps are needed
DASSL	$30, 31, 32$	corrector failed to converge repeatedly
GAMD	$25, \ldots, 29$	stepsize too small
MEBDFDAE	$0, 1$	stepsize too small
MEBDFDAE	$2, \ldots, 18$	illegal function call
PSIDE-1	$0, \ldots, 24$	stepsize too small
RADAU	$0, 1, \ldots, 17$	solver cannot handle `IERR=-1`.
RADAU5	$0, 1, \ldots, 17$	solver cannot handle `IERR=-1`.

Remark

M. Günther also wrote a special purpose solver called CHORAL, which stands for CHarge-ORiented ALgorithm [Gün95, Gün98] for integrating equations of the form

$$\begin{aligned} \frac{dy}{dt} &= f(t, x), \\ 0 &= y - q(x). \end{aligned}$$

Most equations occurring in circuit analysis are of this form. In these equations the variables y and x represent respectively (assembled) charges and voltages. CHORAL is based on Rosenbrock-Wanner methods, while the special structure of the problem is exploited. The code eliminates the y variables, reducing the linear algebra work to solving systems of order 175 instead of 350. Correspondingly, a step size prediction and error control based directly on node potentials and currents is offered. For more information see

`http://www.math.uni-wuppertal.de/~guenther`.

References

[GDF96] M. Günther, G. Denk, and U. Feldmann. Modeling and simulating charge sensitive circuits. *Math. Modelling of Systems*, 2:69–81, 1996.

TABLE II.16.4: *Run characteristics.*

solver	rtol	atol	h0	mescd	scd	steps	accept	#f	#Jac	#LU	CPU
BIMD	10^{-2}	10^{-2}	10^{-1}	2.11	3.23	836	679	12440	679	836	29.4664
	10^{-4}	10^{-4}	10^{-3}	4.44	6.58	1688	1621	24239	1621	1688	62.4786
DDASSL	10^{-2}	10^{-2}		1.55	2.40	1892	1779	3674	786		21.3276
	10^{-4}	10^{-4}		3.60	4.54	6036	5736	9380	866		27.7379
GAMD	10^{-2}	10^{-2}	10^{-1}	2.72	4.66	735	597	20213	597	735	28.1996
	10^{-4}	10^{-4}	10^{-3}	2.68	3.32	1332	1225	43250	1234	1332	61.4255
MEBDFI	10^{-2}	10^{-2}	10^{-1}	1.96	3.14	2065	1818	194700	533	533	19.1911
	10^{-4}	10^{-4}	10^{-3}	3.01	3.36	5269	4851	363601	982	982	38.9239

[Gün95] M. Günther. *Ladungsorientierte Rosenbrock–Wanner–Methoden zur numerischen Simulation digitaler Schaltungen.* Number 168 in Fortschritt-Berichte VDI Reihe 20. VDI-Verlag, Düsseldorf, 1995.

[Gün98] M. Günther. Simulating digital circuits numerically – a charge-oriented ROW approach. *Numer. Math.*, 79(2):203–212, 1998.

[MM08] F. Mazzia and C. Magherini. *Test Set for Initial Value Problem Solvers, release 2.4.* Department of Mathematics, University of Bari and INdAM, Research Unit of Bari, February 2008. Available at http://www.dm.uniba.it/~testset.

[SH68] H. Shichman and D.A. Hodges. Insulated-gate field-effect transistor switching circuits. *IEEE J. Solid State Circuits*, SC-3:285–289, 1968.

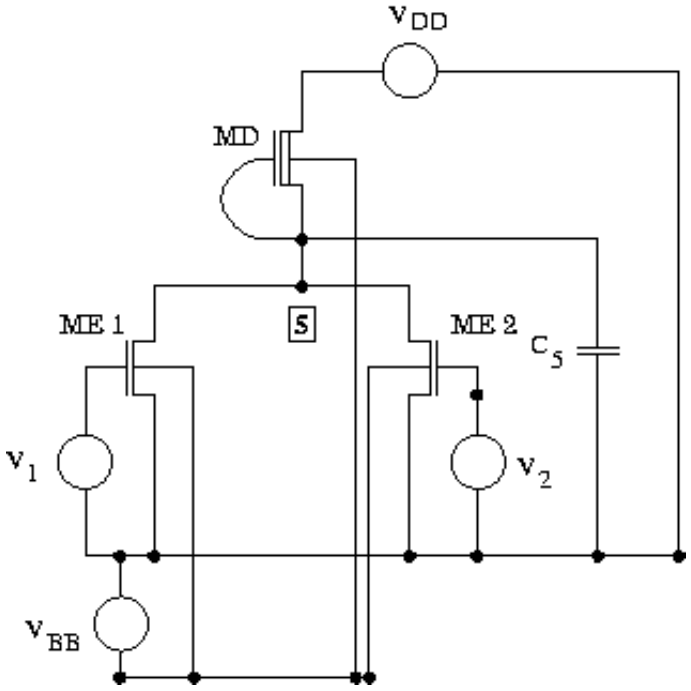

FIGURE II.16.2: *Circuit diagram of the NOR gate (taken from [Gün95]).*

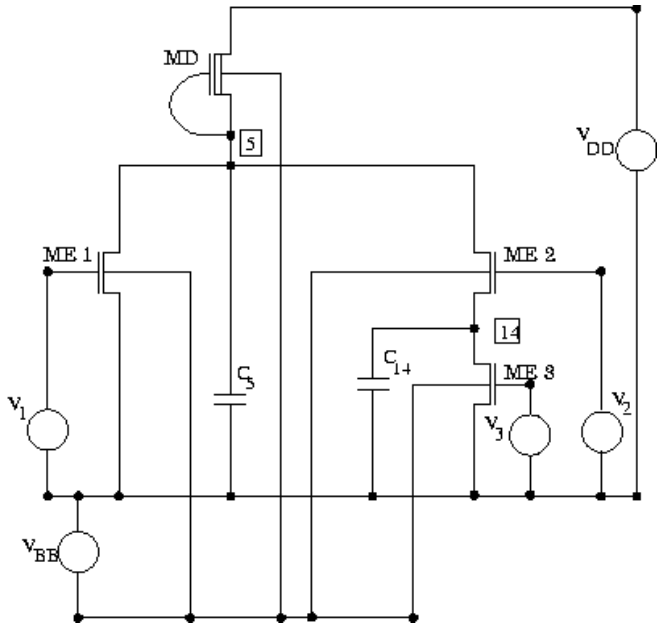

FIGURE II.16.3: *Circuit diagram of the ANDOI gate (taken from [Gün95]).*

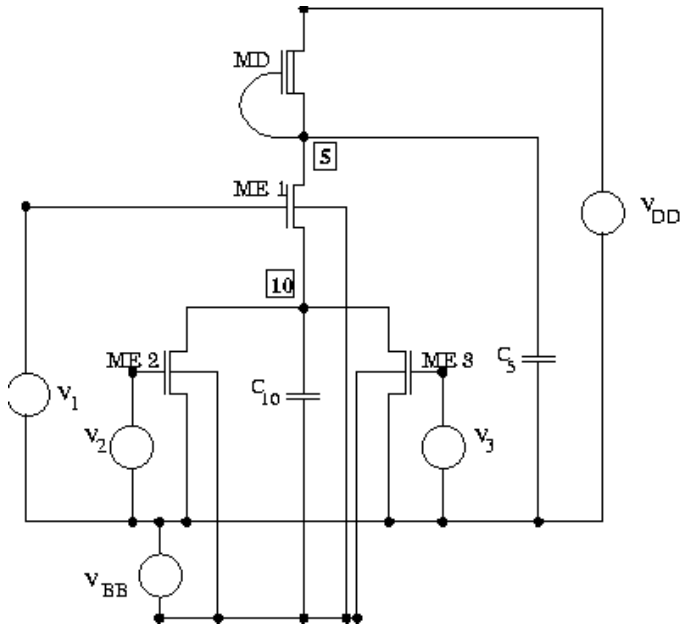

FIGURE II.16.4: *Circuit diagram of the ORANI gate (taken from [Gün95]).*

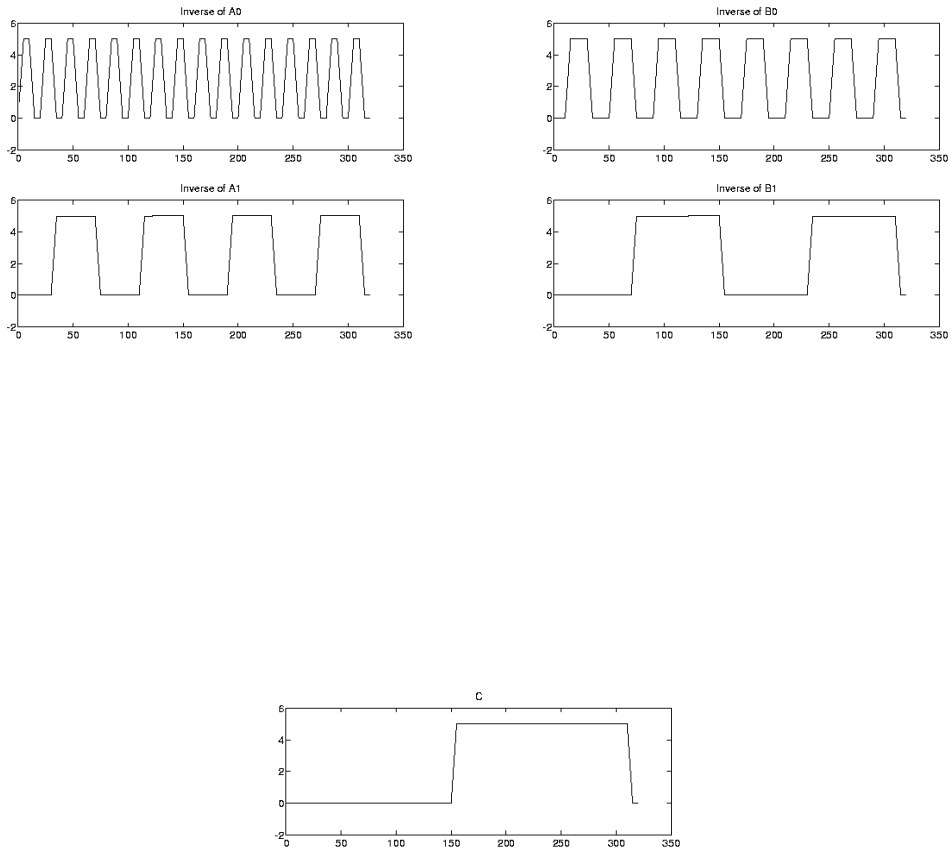

FIGURE II.16.5: *The input signals* $\overline{A}_0$, $\overline{A}_1$, $\overline{B}_0$, $\overline{B}_1$ *and* C.

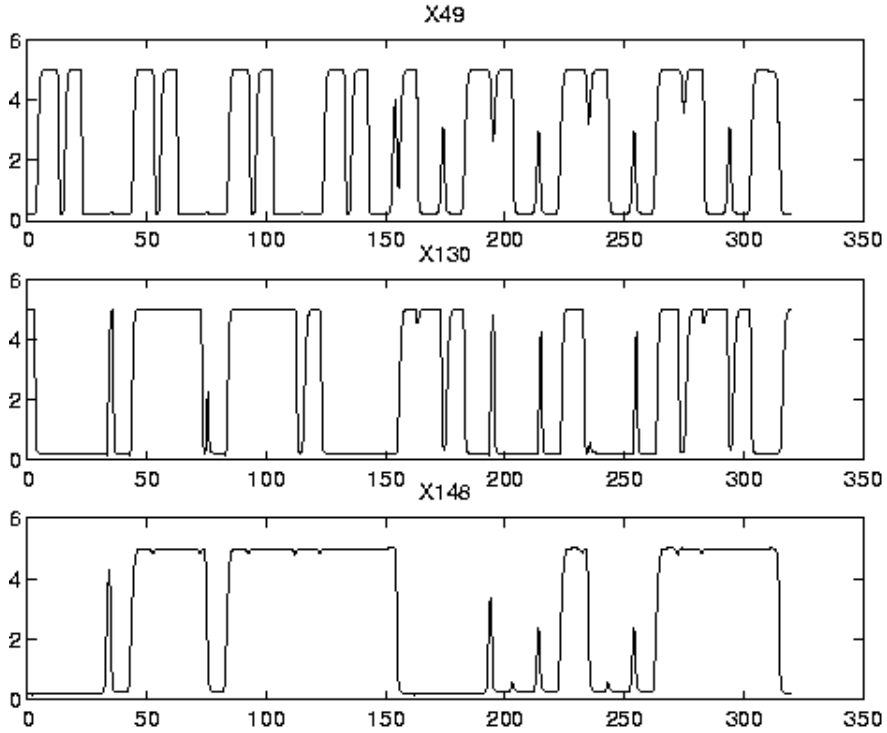

FIGURE II.16.6: *Behavior of the output signals S_0, S_1 and $\overline{C}$ over the integration interval.*

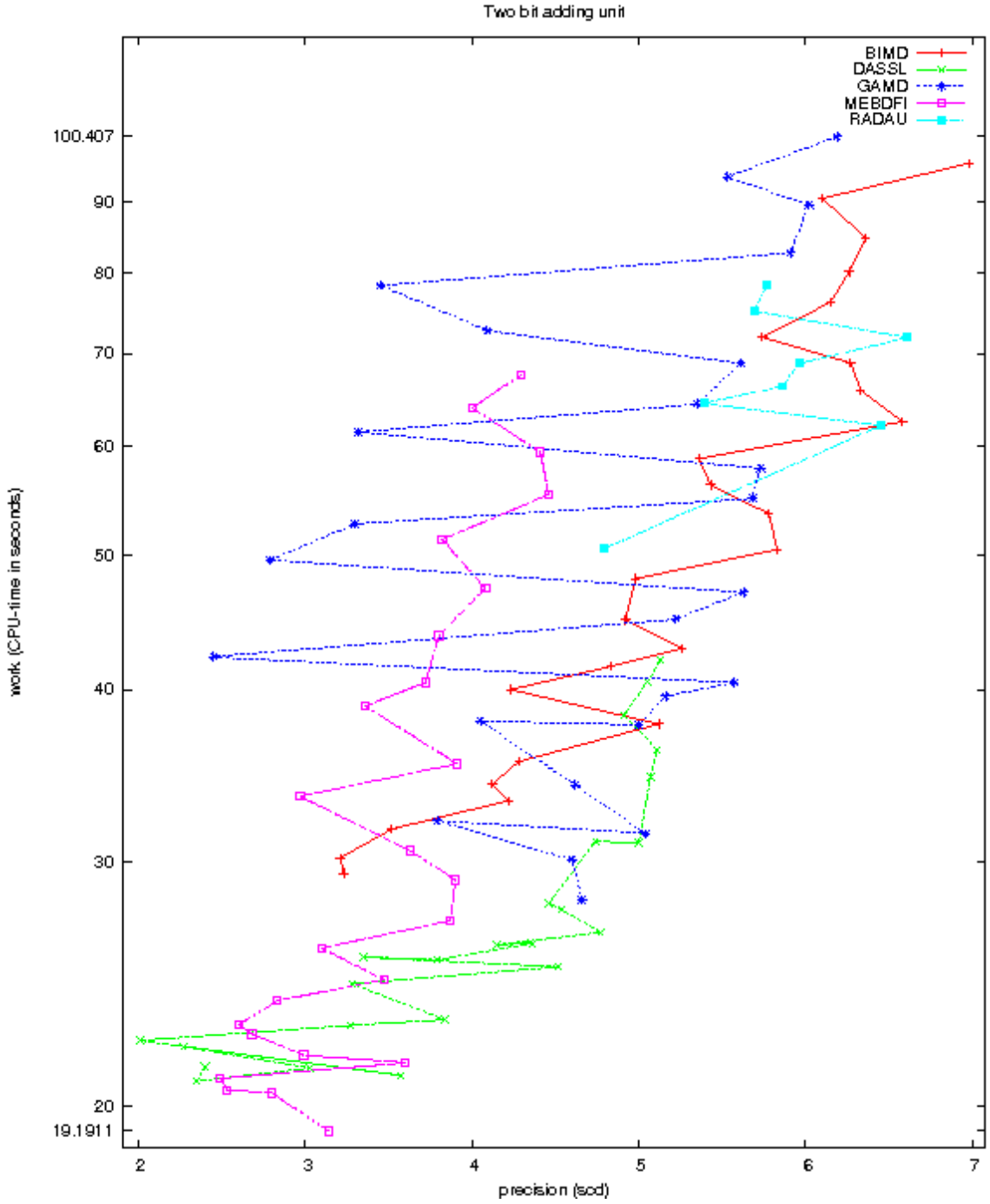

FIGURE II.16.7: *Work-precision diagram (scd versus CPU-time).*

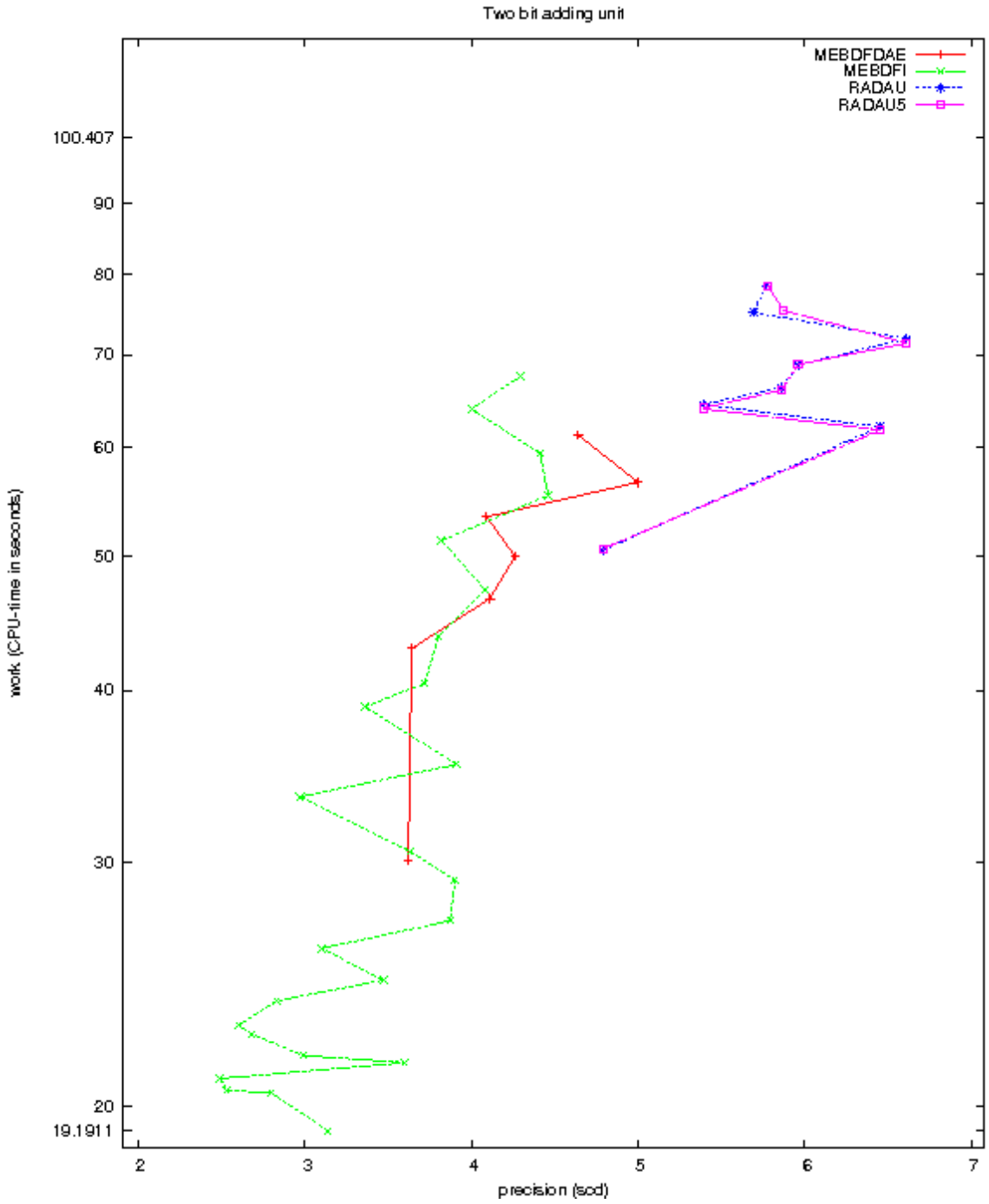

FIGURE II.16.8: *Work-precision diagram (scd versus CPU-time).*

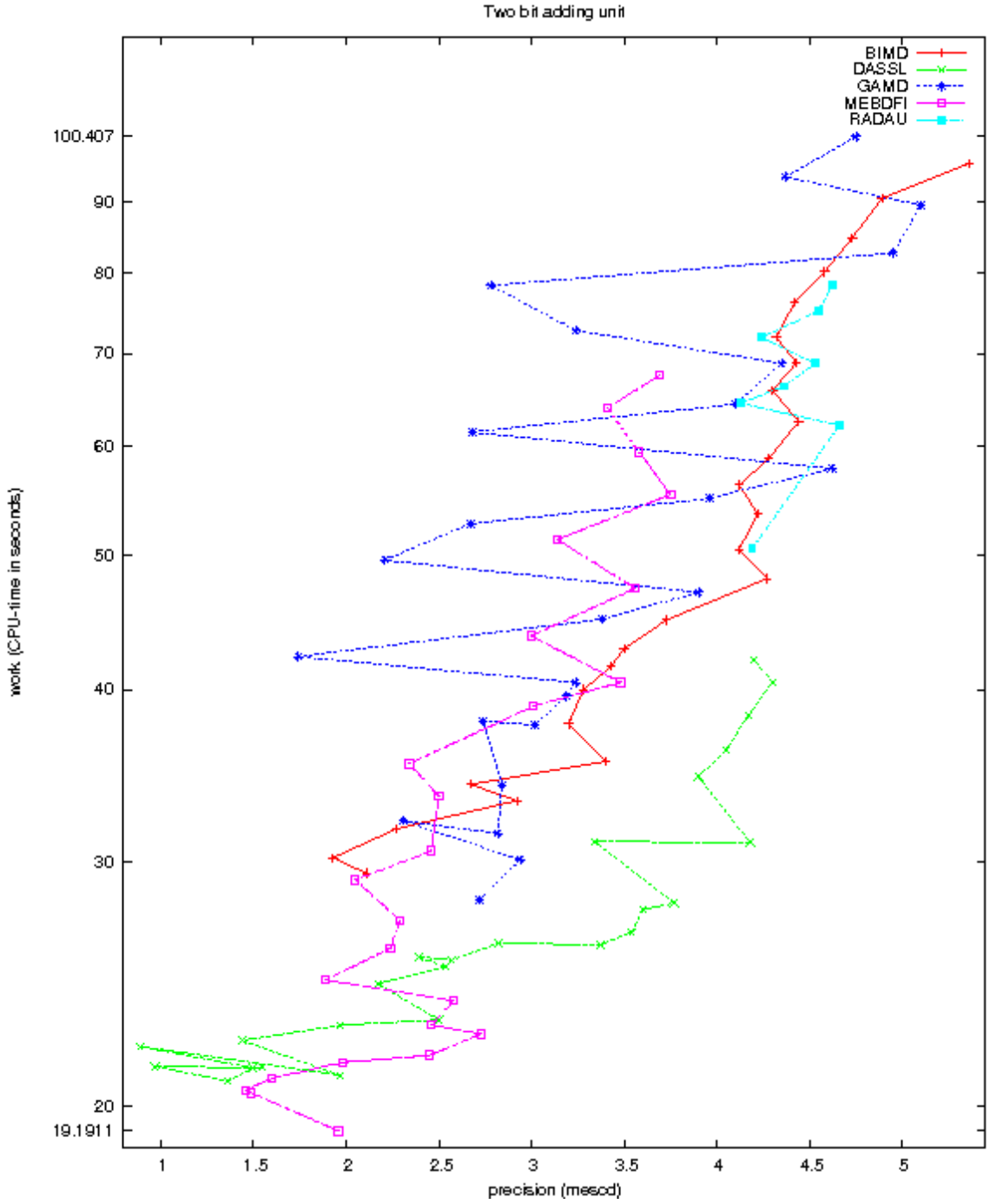

FIGURE II.16.9: *Work-precision diagram (mescd versus CPU-time).*

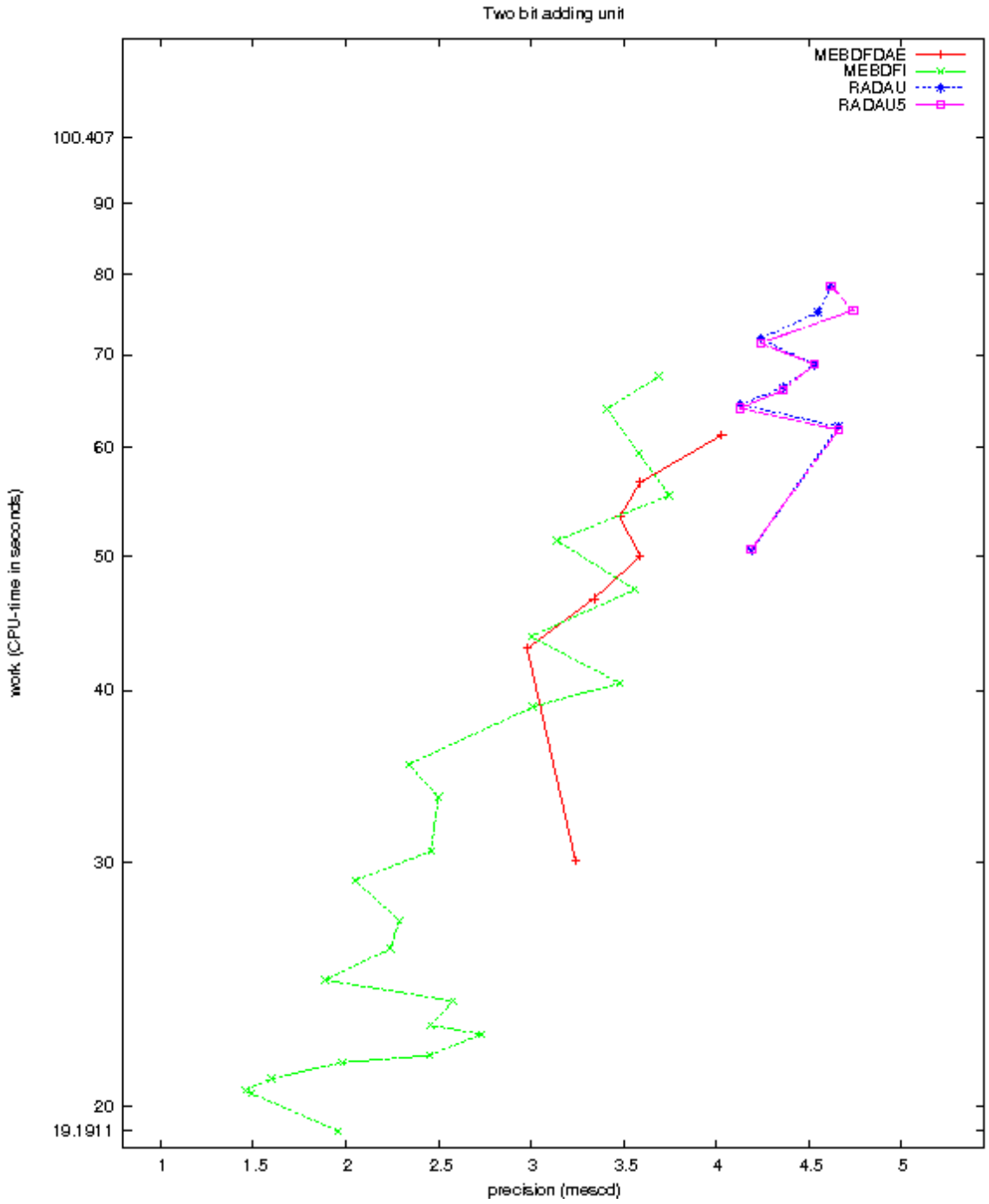

FIGURE II.16.10: *Work-precision diagram (mescd versus CPU-time).*