CHAPTER V

EXPERIMENTS AND RESULTS

We test our technique on the well-known benchmarks such as fifth-order elliptic filter and discrete cosine transformation. We compare our result with the result of the experiment by Ahmad, Dhodhi and Chen [14] since their results are also based on the genetic algorithm. Our testing has two aspects. The first aspect is to test whether our encoding scheme can produce a result as good as or better than the previous results obtained by using different encoding scheme such as that employed by Ahmad, Dhodhi and Chen [14]. The second aspect is to compare the results when the constraints on the number of checkpoints, the maximum recovery time, and the number of registers are enforced in addition to those constraints stated in [14]. Firstly, the design is fifth-order elliptic wave filter. The properties of each functional unit in the experiment are previously given in Table 4.1.

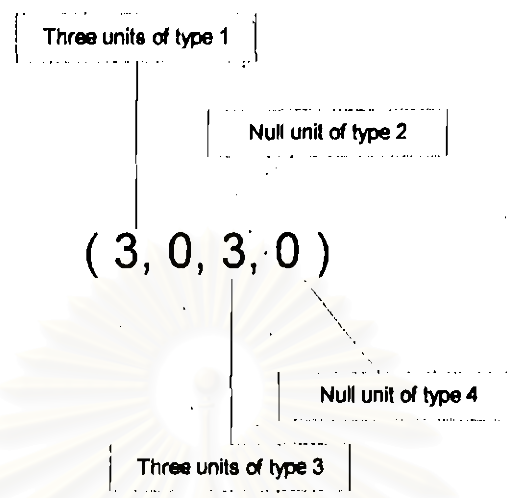

Figure 5.1 An example of the symbol defined in the experiment.

Table 5.1 shows the comparison between the number of control steps with and without constraints on the checkpoints, the number of registers, and the maximum allowable checkpoints. Each number in the parenthesis in the first column indicates the number of units of each type used. For example, as shown in Figure 5.1, (3,0,3,0) means that we use three units of type 1, zero unit of type 2, three units of type 3, and zero unit of type 4. From Table 1, types 1 and 2 are multipliers while types 3 and 4 are adders/subtracters. Our results are the same as theirs and even better when there is no checkpoint constraint. In case (1,0,1,1), our approach obtains the CDFG without checkpoint insertion within 21 control steps as shown in Figure 5.2. Moreover, if the checkpoints are inserted we still obtain the same results in most cases. Figure 5.3-5.7 show the final CDFG with checkpoint insertion in case of (3,0,3,0), (2,0,2,0), (1,1,2,1), (1,1,1,1) and (1,0,1,0) with constraints, which are composed of nine registers, four checkpoints and six control steps for the maximum allowable checkpoints.

Table 5.1 Comparison between the number of control steps with and without checkpoint constraints

Module Sets	# of CSs [14]	# of CSs no checkpoint	# of CSs # reg = 9 chkpnts = 4 max chkpnts = 6	# of CSs # reg = 9 chkpnts = 5 max chkpnts = 6
(3,0,3,0)	17	17	18	18
(2,0,2,0)	18	18	18	18
(1,1,2,1)	20	20	20	20
(1,1,1,1)	21	21	21	21
(1,0,1,1)	22	21	22	22

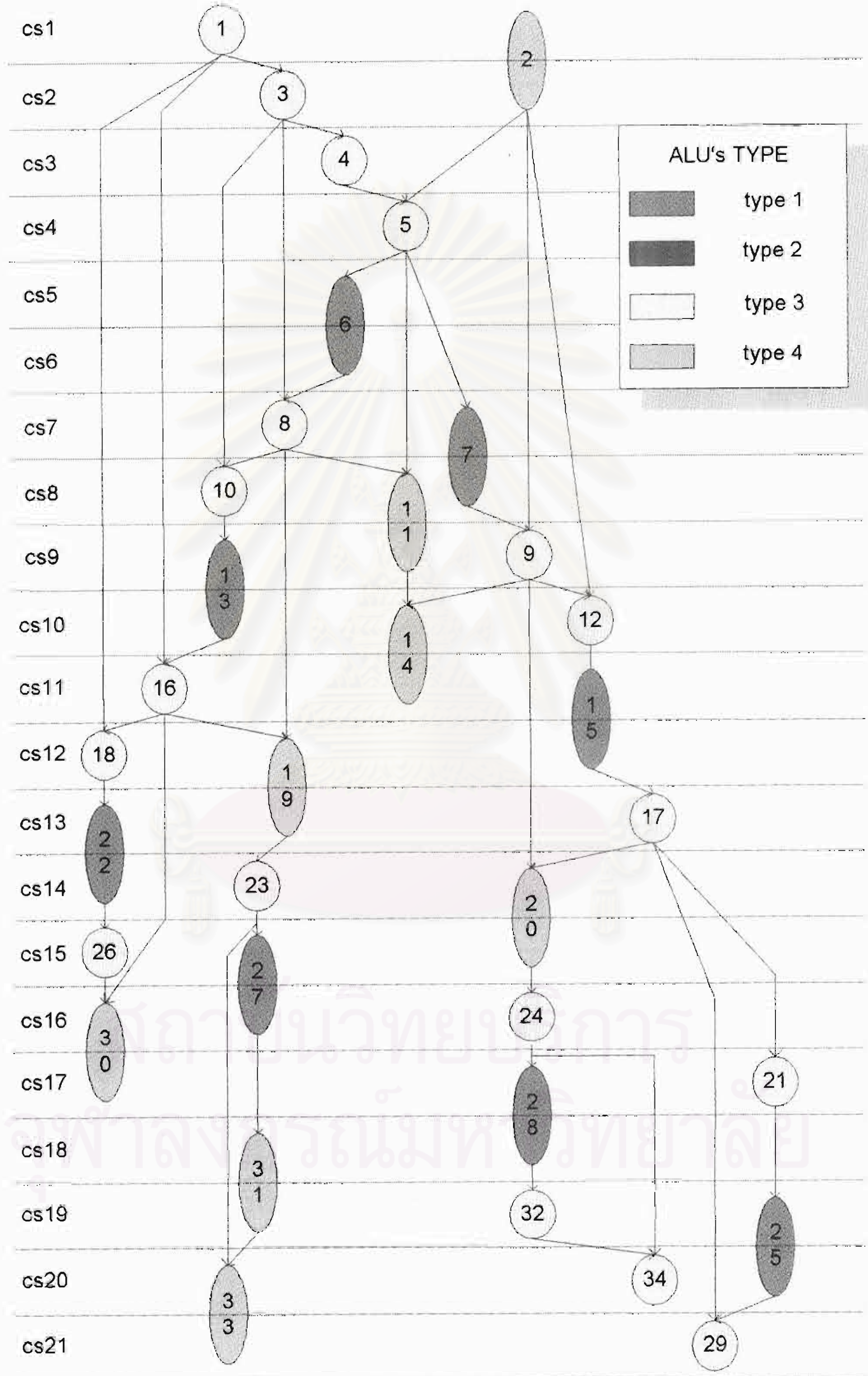

Figure 5.2 The final CDFG without checkpoint insertion in case (1,0,1,1)

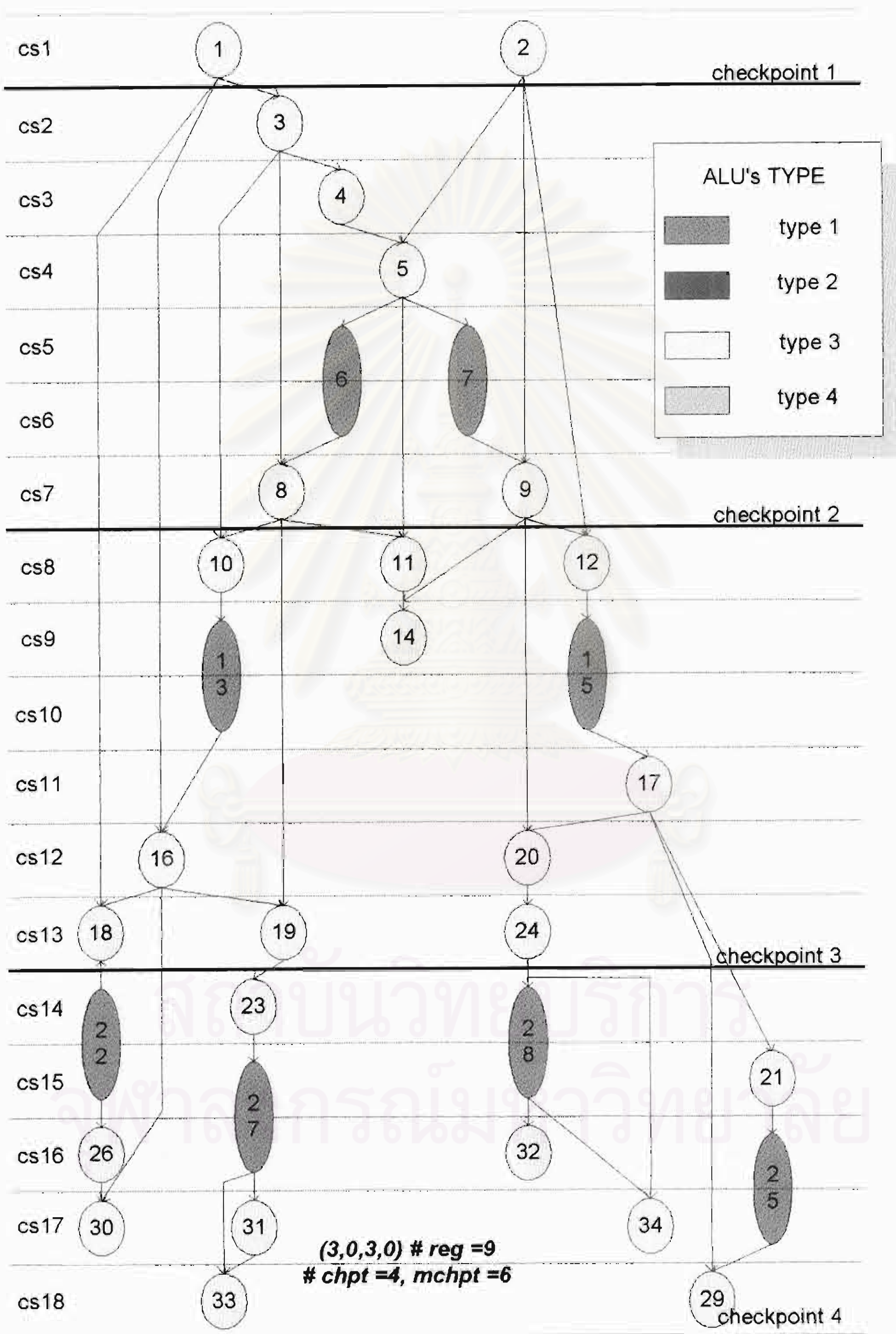

Figure 5.3 The final CDFG with checkpoint insertion in case (3,0,3,0)

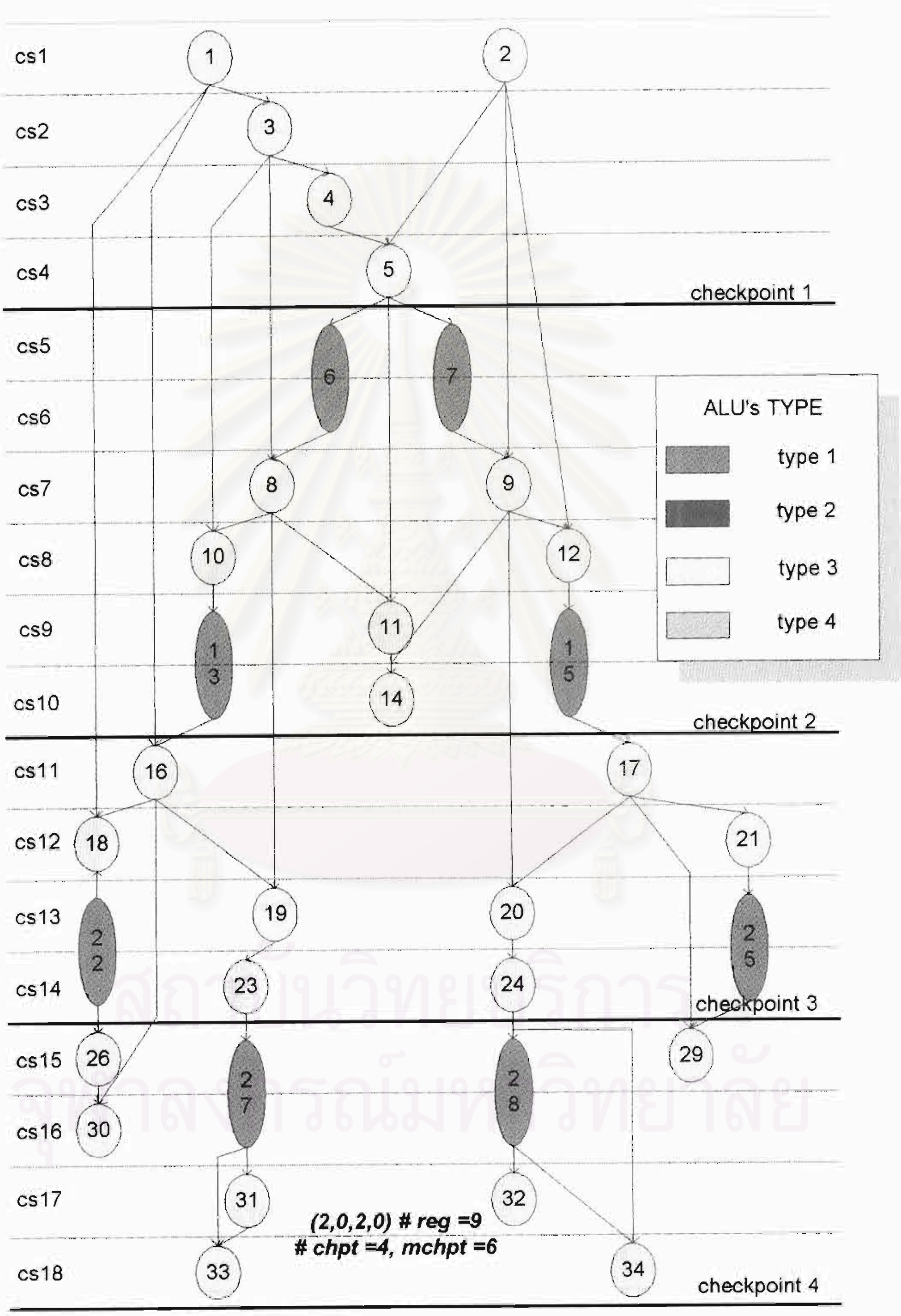

Figure 5.4 The final CDFG with checkpoint insertion in case (2,0,2,0)

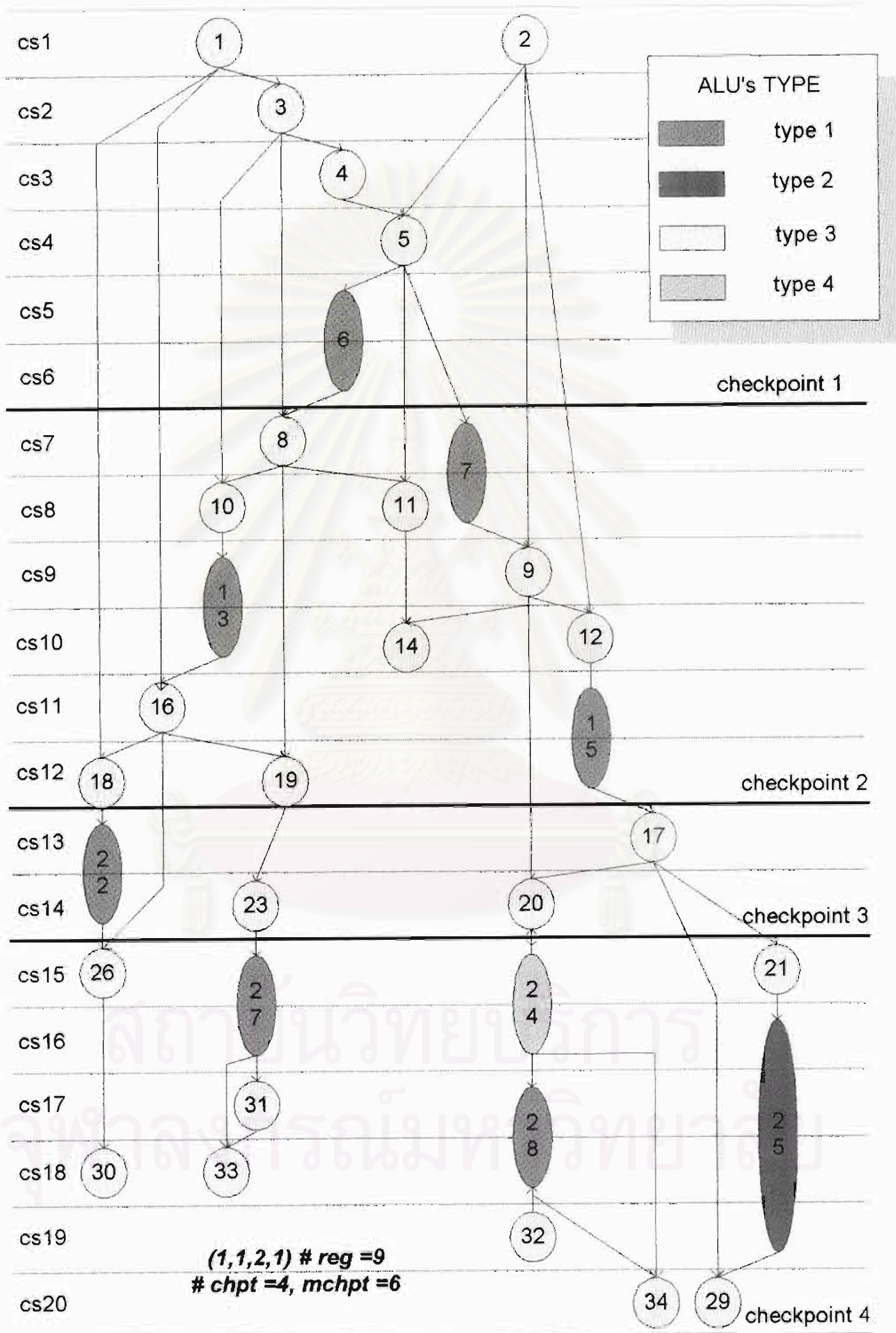

Figure 5.5 The final CDFG in case (1,1,2,1)

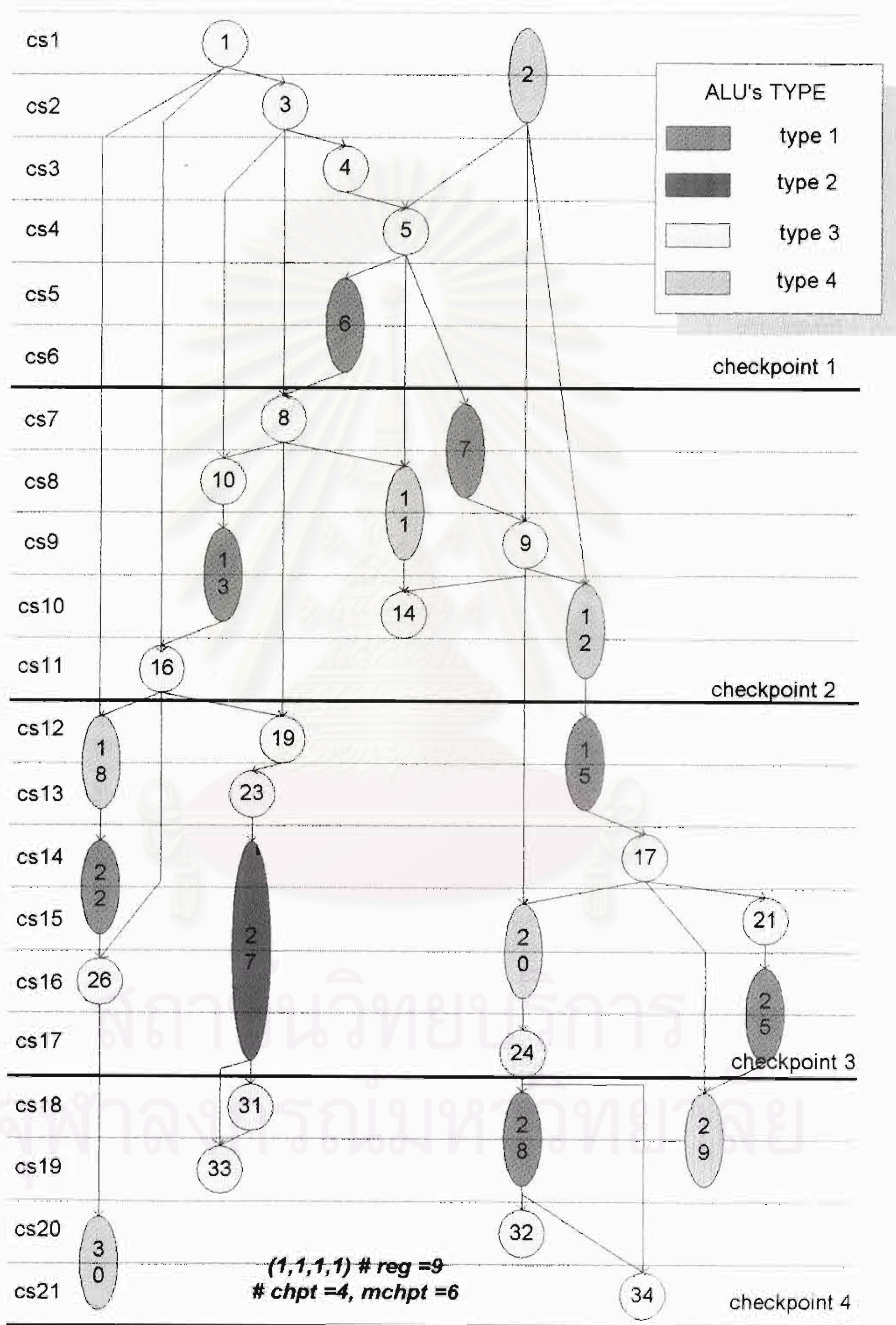

Figure 5.6 The final CDFG with checkpoint insertion in case (1,1,1,1)

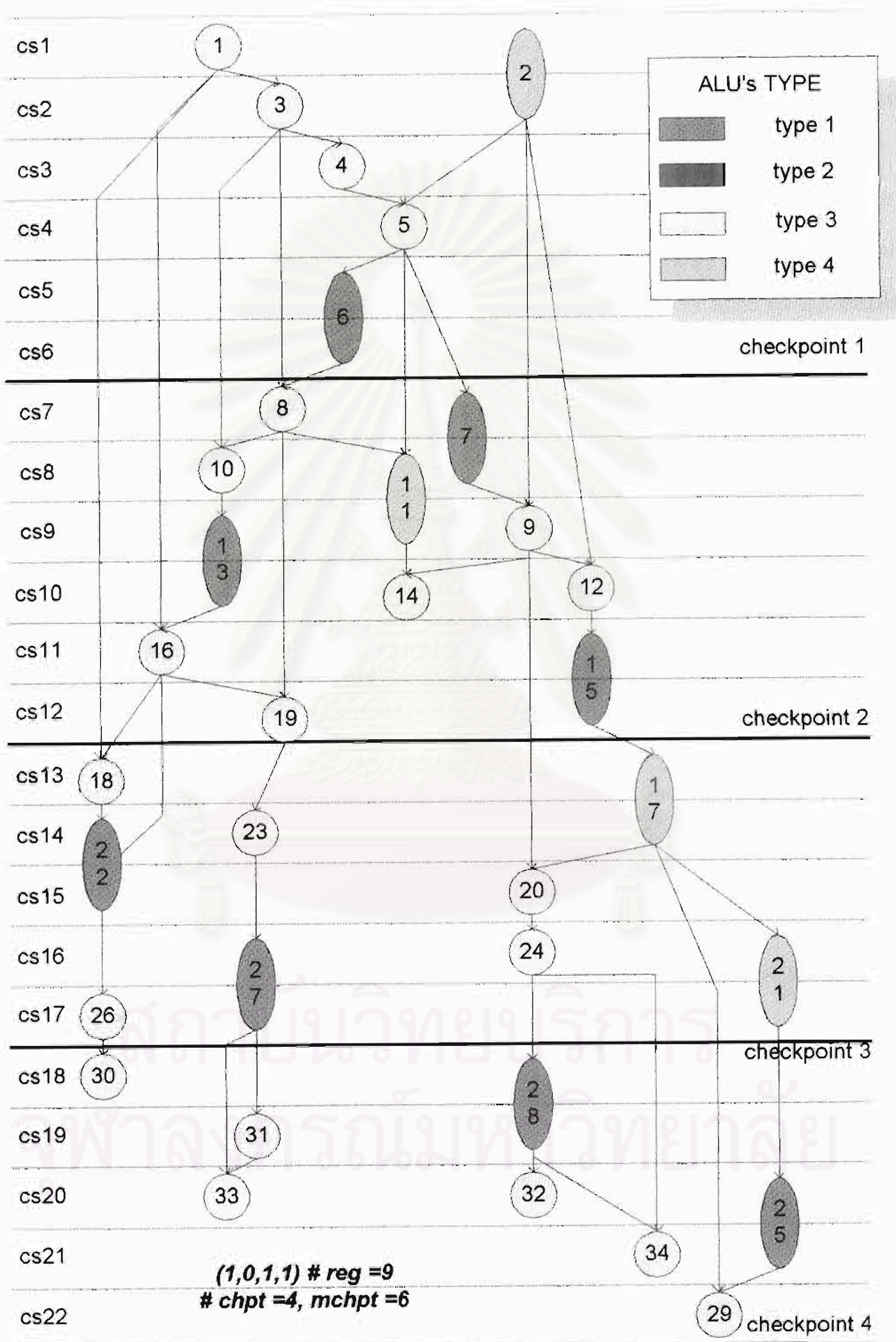

Figure 5.7 The final CDFG with checkpoint insertion in case (1,0,1,1)

Secondly, we examine an 8-point Discrete Cosine Transform compression. Data-flow graph for the DCT example is shown in Figure 5.8. We assume that no module selection and no checkpoint insertion can be compared with the result in [14]. Our system gives the schedule using two ALUs, which take one control step, and three multipliers, which take two control steps. The result shows that our system is able to schedule within 16 control steps whereas their system performs in 17 control steps by using the same resource. Table 5.2 presents the microprogram code for this scheduling.

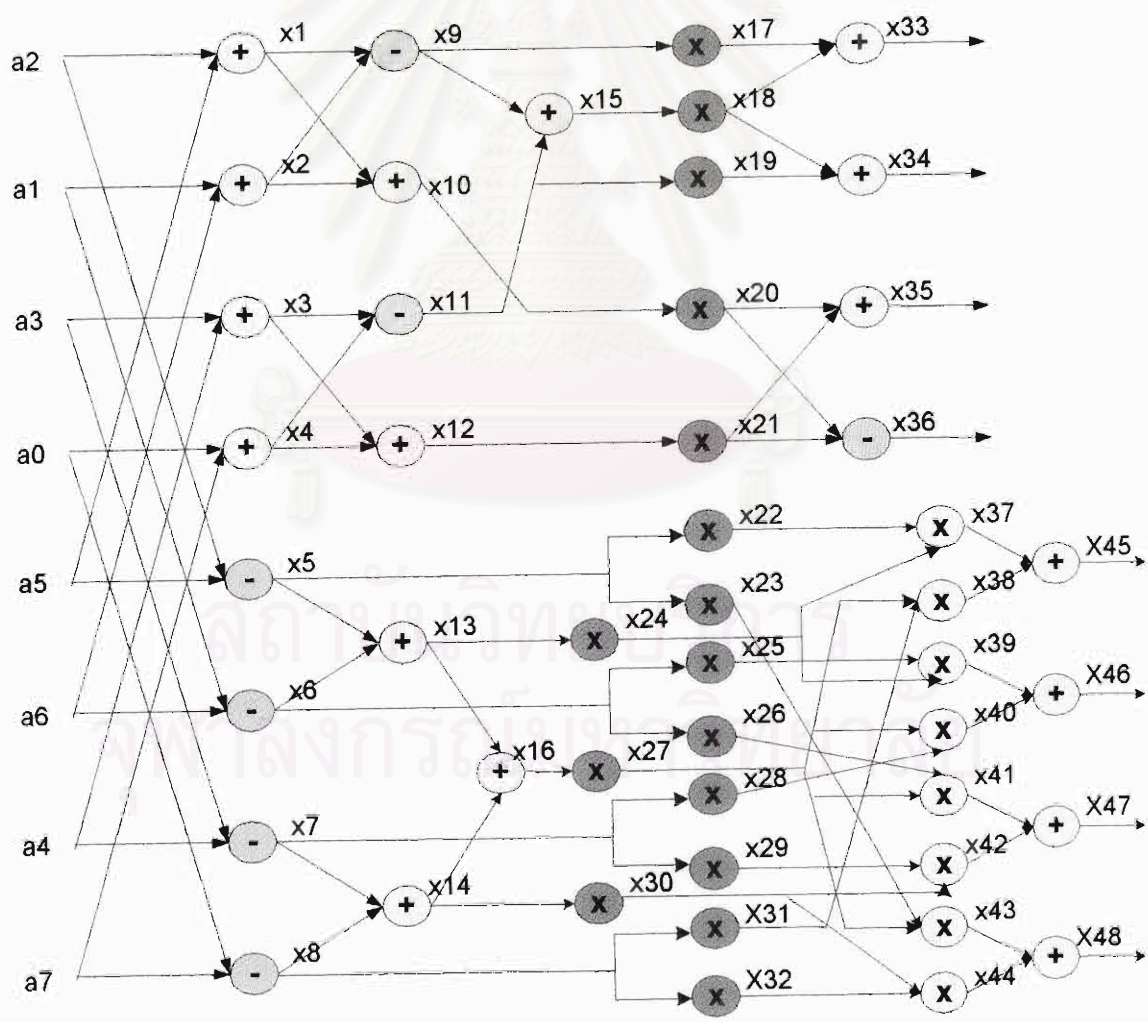

Figure 5.8 The Discrete Cosine Transform example (DCT).

Table 5.2 Microprogram of DCT example

# CS	Multiplier-1	Multiplier-2	Multiplier-3	ALU1(+,-)	ALU2(+,-)
1				x3=a3+a4	x5=a2-a5
2	x22=x28*a22	x23=x5*a23		x2=a1+a6	x8=a0-a7
3			x31=x8*a31	x4=a0+a7	x6=a1-a6
4	x32=x8*a32	x26=x6*a26		x12=x3+x4	x7=a3-a4
5			x28=x7*a28	x1=a2+a5	x14=x7+x8
6	x25=x6*a25	x29=x7*a29		x10=x1=x2	x11=x3-x4
7			x30=x14*a30	x9=x1-x2	x13=x5+x6
8	x20=x10*a20	x21=x12*a21		x15=x9+x11	x16=x13+x14
9			x24=x13*a24	x42=x29+x30	x44=x30+x32
10	x19=x11*a19	x27=x12*a27		x35=x20+x21	x36=x20-x21
11			x17=x19*a17	x37=x22+x24	x39=x24+x25
12	x18=x15*a18			x41=x26+x27	x38=x27+x31
13				x47=x41+x42	x40=x27+x28
14				x46=x39+x40	x43=x23+x27
15				x33=x17+x18	x34=x18+x19
16				x45=x37+x38	x48=x43+x44