

POLITECHNIKA WROCŁAWSKA  
WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI



Organizacja i Architektura Komputerów

---

# Sprawozdanie z Zadania Projektowego: Sumatory modularne (1)

---

Student: Michał Zajdel 263932  
Kuba Starościak 264461

WROCŁAW, KWIECIEŃ 2024



---

## Spis treści

<b>1</b>	<b>Wstęp</b>	<b>4</b>
1.1	Cele projektu . . . . .	4
1.2	Opis implementacji . . . . .	4
<b>2</b>	<b>Założenia projektowe</b>	<b>5</b>
2.1	System resztkowy . . . . .	5
2.2	Sumator modulo $2^n - K$ . . . . .	5
2.3	Sumator modulo $2^n + K$ . . . . .	6
2.4	Fazy przetwarzania . . . . .	7
<b>3</b>	<b>Testy funkcjonalności</b>	<b>9</b>
<b>4</b>	<b>Napotkane Problemy</b>	<b>9</b>
<b>5</b>	<b>Wnioski</b>	<b>9</b>
<b>6</b>	<b>Implementacje</b>	<b>9</b>
6.1	Python . . . . .	9
6.2	Verilog . . . . .	19
6.3	Logisim Evolution . . . . .	27
6.4	Promty wykorzystane w rozmowie z chatGPT . . . . .	71
<b>7</b>	<b>Źródła</b>	<b>76</b>

## Listings

6.1	Sumator Ladnera-Fischera (Python) . . . . .	9
6.2	Wersja $2^n - k$ (Python) . . . . .	13
6.3	Wersja $2^n + k$ (Python) . . . . .	16
6.4	Sumator w języku Verilog . . . . .	19
6.5	Struktura Fischera . . . . .	27

---

# 1 Wstęp

Poniższe sprawozdanie zawiera opis implementacji sumatora modularnego Hiasata. Jest to układ wykonujący dodawanie binarne w określonym pierścieniu modularnym. Zaprojektowany układ logiczny, według badań przedstawionych w artykule wykazuje między 13% a 33% większą efektywność pod względem powierzchni, propagacji, mocy i energii względem innych sumatorów modularnych.

## 1.1 Cele projektu

Wykonanie projektu obejmowało poniżej cele:

- Zrozumienie koncepcji zaprezentowanej w artykule Ahmeda Hiasata
- Implementacja proponowanego w artykule rozwiązania w wybranym języku programowania
- Implementacja układu w języku opisu sprzętu Verilog

## 1.2 Opis implementacji

Implementacja układu została wykonana w następujących wariantach:

- **wersja wysokopoziomowa** – Symulacja działania układu w języku Python
- **wersja HDL** – implementacja układu  $2^n - k$ , w języku Verilog
- **W symulatorze układów cyfrowych** – zbudowanie układu w programie Logisim Evolution

Sumator w wersji w pythonie, oraz w verilogu działa dynamicznie –  $N$  w wersji Python jest ustawione automatycznie - w zależności od ilości bitów większej liczby dodawanej,  $k$  ustawiane są dynamicznie. W wersji verilog i  $N$  i  $k$  są ustawiane dynamicznie

---

## 2 Założenia projektowe

### 2.1 System resztkowy

System resztkowy to metoda przedstawiania liczb, oparta na wykorzystaniu reszt z dzielenia przez określony zestaw modułów. Każda liczba jest reprezentowana poprzez swoje reszty w poszczególnych modułach. Dzięki zastosowaniu tego systemu możliwe jest równoległe wykonywanie operacji arytmetycznych, co skutkuje znacznym skróceniem czasu potrzebnego na poszczególne działania, szczególnie przy operacjach na dużych liczbach. W przypadku implementacji tego systemu w sprzęcie, istotnym elementem są sumatory modułowe, które umożliwiają dodawanie liczb, zachowując jednocześnie wyniki w zakresie określonym przez zastosowane modulo. Ahmed Hiasat zaproponował w swoim artykule nową metodę implementacji ogólnego sumatora modułowego z zastosowaniem modulo  $2^n \pm K$ .

### 2.2 Sumator modulo $2^n - K$

Przyjmując założenie, że  $n$  to dodatnia liczba całkowita,  $m_1$  to moduł o postaci  $2^n - K$ , gdzie  $K$  mieści się w przedziale od 3 do  $2^{n-1} - 1$ , a  $A$  i  $B$  to  $n$ -bitowe liczby całkowite mniejsze od  $m_1$ , wzór na sumę  $S$  można zapisać w następujący sposób:

$$S = \langle A + B \rangle_{m_1} \quad (2.1)$$

$$S = \begin{cases} A + B & \text{jeżeli } A + B < m_1 \\ A + B - (2^n - K) & \text{jeżeli } A + B \geq m_1 \end{cases} \quad (2.2)$$

Po zastosowaniu modulo  $2^n$  dla obu przypadków otrzymujemy równanie:

$$S = \begin{cases} \langle A + B \rangle_{2^n} & \text{jeżeli } c_{out} = 0 \\ \langle A + B + K \rangle_{2^n} & \text{jeżeli } c_{out} = 1 \end{cases} \quad (2.3)$$

---

### 2.3 Sumator modulo $2^n + K$

Zakładając, że  $n$  jest liczbą naturalną,  $m_2$  jest modułem w postaci  $2n + K$ , gdzie  $3 \leq K \leq 2^{n-1} - 1$ , a  $A$  i  $B$  są  $n$ -bitowymi liczbami naturalnymi mniejszymi od  $m_2$  to wzór na sumę  $S$  można zapisać za pomocą następującego wzoru:

$$S = \langle A + B \rangle_{m_2} \quad (2.4)$$

$$S = \begin{cases} A + B & \text{jeżeli } A + B < m_2 \\ A + B - (2^n - K) & \text{jeżeli } A + B \geq m_2 \end{cases} \quad (2.5)$$

$$S = \begin{cases} \langle A + B \rangle_{2^{n+1}} & \text{jeżeli } c_{out} = 0 \\ \langle A + B + \tilde{K} \rangle_{2^{n+1}} & \text{jeżeli } c_{out} = 1 \end{cases} \quad (2.6)$$

$$\tilde{K} = 2^n - K \quad (2.7)$$

Po zastosowaniu modulo  $2^n$  dla obu przypadków otrzymujemy równanie: Przy obliczaniu wartości  $c_{out}$  stosuje się logiczną sumę najbardziej znaczących bitów przeniesienia, które wynikają z dwóch składowych równań w układzie. Kolejne etapy działania sumatora są identyczne jak w przypadku sumatora modulo  $2^n - K$ , z wyjątkiem tego, że dla sumatora  $2^n + K$  przy  $n = 6$  należy użyć struktury sumatora  $2^n - K$  dla  $n = 7$ , a zamiast  $K$  należy używać wektora przedstawionego poniżej.

---

## 2.4 Fazy przetwarzania

- Faza wstępnego przetwarzania:

$$A = a_{n-1}, \dots, a_1, a_0 \quad (2.8)$$

$$B = b_{n-1}, \dots, b_1, b_0 \quad (2.9)$$

$$\begin{aligned} H &= h_{n-1}, \dots, h_1 h_0 \\ h_i &= a_i \oplus b_i \end{aligned} \quad (2.10)$$

$$\begin{aligned} G &= g_{n-1}, \dots, g_1 g_0 \\ g_i &= a_i \wedge b_i \end{aligned} \quad (2.11)$$

$$\begin{aligned} P &= p_{n-1}, \dots, p_1 p_0 \\ p_i &= a_i \vee b_i \end{aligned} \quad (2.12)$$

$$\tilde{K} = 0k_{n-2}, \dots, k_1, k_0 \quad (2.13)$$

$$A' = a'_{n-1}, \dots, a'_1 a'_0 \quad (2.14)$$

$$a'_i = \begin{cases} a_i \oplus b_i & \text{jeżeli } k_i = 0 \\ a_i \odot b_i & \text{jeżeli } k_i = 1 \end{cases} \quad (2.15)$$

$$B' = b'_n b'_{n-1}, \dots, b'_1 0 \quad (2.16)$$

$$b'_{i+1} = \begin{cases} a_i \wedge b_i & \text{jeżeli } k_i = 0 \\ a_i \vee b_i & \text{jeżeli } k_i = 1 \end{cases} \quad (2.17)$$

---


$$\begin{aligned} H' &= h'_{n-1}, \dots, h'_1 h'_0 \\ h'_i &= a'_i \oplus b'_i \end{aligned} \tag{2.18}$$

$$\begin{aligned} G' &= g'_{n-1}, \dots, g'_1 g'_0 \\ g'_i &= a'_i \wedge b'_i \end{aligned} \tag{2.19}$$

$$\begin{aligned} P' &= p'_{n-1}, \dots, p'_1 p'_0 \\ p'_i &= a'_i \vee b'_i \end{aligned} \tag{2.20}$$

- Faza Parallel-Prefix

$$\begin{aligned} p_{i,i-1} &= p_i \wedge p_{i-1} \\ p'_{i,i-1} &= p'_i \wedge p'_{i-1} \end{aligned} \tag{2.21}$$

$$\begin{aligned} g_{i,i-1} &= g_i \vee (g_{i-1} \wedge p_i) \\ g'_{i,i-1} &= g'_i \vee (g'_{i-1} \wedge p'_i) \end{aligned} \tag{2.22}$$

- Faza obliczania sumy

$$c_{out} = c'_{n-1} \vee c_{n-1} \tag{2.23}$$

$$s_0 = h_0 \tag{2.24}$$

$$S = \begin{cases} h_i \oplus c_{i-1} & \text{jeżeli } c_{out} = 0 \\ h'_i \oplus c'_{i-1} & \text{jeżeli } c_{out} = 1 \end{cases} \tag{2.25}$$



---

## 3 Testy funkcjonalności

Zaprojektowane układy zostały poddane manualnemu testowaniu. Po wprowadzeniu, przez osobę testującą, wartości  $a$ ,  $b$  i  $k$ , wynik był porównywany za oczekiwanym przez testującego. Zostało wykonane około 100 testów dla różnych wartości dodawanych liczb,  $k$  oraz  $n$  (Implementacja w językach python i verilog). Wszystkie testy zwróciły oczekiwany wynik w implementacji Python. Testy dla liczb o  $n$  mniejszym niż 6 w Verilog nie okazały się poprawne.

## 4 Napotkane Problemy

Podczas implementacji układu, wykryty został błąd - w strukturze przedstawionej w artykule, wybieranie bitu  $c_{out}$  odbywa się tylko na podstawie bitu wektora przeniesienia  $prim\ c'_{n-1}$ , jednak w tej sytuacji układ nie zawsze wykaże poprawny wynik. Bit  $c_{out}$  został więc otrzymany przy pomocy bramki OR.

## 5 Wnioski

W ramach projektu zostały zrealizowane jego założenia. Implementacja wysokopoziomowa, układ w symulatorze układów cyfrowych tak jak i implementacja w języku Verilog zwracają odpowiednie wyniki.

## 6 Implementacje

### 6.1 Python

Implementacja sumatorów w języku Python

Listing 6.1: Sumator Ladnera-Fischera (Python)

```
1 from array import *
2 from math import ceil, log2
3 def run(number_1, number_2):
4
5     num_1_Bits = bitify(number_1)
6     num_2_Bits = bitify(number_2)
7
```

---

```
8     num_1_Bits.reverse()
9     num_2_Bits.reverse()
10
11     max_length = max(len(num_1_Bits), len(num_2_Bits))
12
13     num_1_Bits = num_1_Bits + [False] * (max_length - len
14                                     (num_1_Bits))
15     num_2_Bits = num_2_Bits + [False] * (max_length - len
16                                     (num_2_Bits))
17     #preprocessing stage
18     H, P, G = preProcessingStage(num_1_Bits, num_2_Bits)
19     print(H)
20     #parallel-prefix stage
21     C = parallelPrefixStage(H, P, G, max_length)
22     print(C)
23     #Sum
24     S = SumComputationStage(H, C)
25
26     printNum(num_1_Bits)
27     printNum(num_2_Bits)
28     printNum(S)
29
30 def preProcessingStage(number_1_Bit, number_2_Bit):
31
32     H = []
33     P = []
34     G = []
35
36     for bit1, bit2 in zip(number_1_Bit, number_2_Bit):
37         H.append(XOR(bit1, bit2))
38         P.append(OR(bit1, bit2))
39         G.append(AND(bit1, bit2))
40
41     return H, P, G
```

---

```

41 def parallelPrefixStage(H, P, G, n):
42     logic = []
43     for i in range (0, n):
44         tuple1 = bitify(i)
45         tuple1 = [False] * (n - len(tuple1)) + tuple1
46         tuple1.reverse()
47         logic.append(tuple1)
48     C = []
49     Pairs = [[[G[0],P[0]] ]* ceil(log2(n))]
50     for i in range (1, n):
51         pi = False
52         gi = False
53         for j in range (0, ceil(log2(n))):
54             if logic[i][j]:
55                 if j == 0:
56                     pi = P_i(P[i], P[i-1])
57                     gi = G_i(G[i], G[i - 1], P[i])
58                     Pairs.insert(i, [[gi,pi]])
59                 else:
60                     k = i
61                     while logic[k][j]:
62                         k -= 1
63                     gi = G_i(Pairs[i][j-1][0], Pairs[k][j-1][0], Pairs[i][j-1][1])
64                     pi = P_i(Pairs[i][j-1][1], Pairs[k][j-1][1])
65                     Pairs[i].append([gi,pi])
66             else:
67                 if j == 0:
68                     pi = P[i]
69                     gi = G[i]
70                     Pairs.insert(i, [[gi,pi]])
71                 else:
72                     pi = Pairs[i][j-1][1]
73                     gi = Pairs[i][j-1][0]

```

---

```
74         Pairs[i].append([gi,pi])
75
76     for list in Pairs:
77         C.append(list[ceil(log2(n))-1][0])
78
79     return C
80
81 def SumComputationStage(H,C):
82     S = [H[0]]
83     for i in range(1,len(H)):
84         S.append(XOR(H[i],C[i-1]))
85     if (C[len(C) - 1]):
86         S.append(C[len(C) - 1])
87     return S
88
89 def bitify(number):
90     inBits = []
91     numberBinary = "{0:b}".format(number)
92     for char in numberBinary:
93         if char == '1':
94             inBits.append(True)
95         else: inBits.append(False)
96     return inBits
97
98 def AND(bit1, bit2):
99     return bit1 and bit2
100
101 def OR(bit1, bit2):
102     return bit1 or bit2
103
104 def XOR(bit1, bit2):
105     return (bit1 and not bit2) or (not bit1 and bit2)
106
107 def P_i (Pi, prev_Pi):
108     return AND(Pi, prev_Pi)
```

---

```

109
110 def G_i (Gi, prev_Gi, Pi):
111     return OR(Gi, AND(prev_Gi, Pi))
112
113 def printNum(inBits):
114     numString = ""
115     for char in reversed(inBits):
116         if char:
117             numString += '1'
118         else: numString += '0'
119     print(numString)

```

Listing 6.2: Wersja  $2^n - k$  (Python)

```

1 import LadnerFischer as og
2 from array import *
3 from math import ceil, log2
4
5 def run(number_1, number_2, k):
6
7     num_1_Bits = og.bitify(number_1)
8     num_2_Bits = og.bitify(number_2)
9     k_Bits = og.bitify(k)
10
11     num_1_Bits.reverse()
12     num_2_Bits.reverse()
13     k_Bits.reverse()
14
15     max_length = max(len(num_1_Bits), len(num_2_Bits))
16     og_Length = max_length
17
18     num_1_Bits = num_1_Bits + [False] * (max_length - len
        (num_1_Bits))
19     num_2_Bits = num_2_Bits + [False] * (max_length - len
        (num_2_Bits))
20     k_Bits = k_Bits + [False] * (max_length - len(k_Bits))

```

---

```

    )
21
22     num_1_prim_Bits, num_2_prim_Bits = numsPrim(
        num_1_Bits, num_2_Bits, k_Bits)
23
24     #preprocessing stage
25     H, P, G = og.preProcessingStage(num_1_Bits,
        num_2_Bits)
26     H_prim, P_prim, G_prim = og.preProcessingStage(
        num_1_prim_Bits, num_2_prim_Bits)
27
28     #parallel prefix stage
29     C = og.parallelPrefixStage(H,P,G, max_length)
30     C_prim = og.parallelPrefixStage(H_prim, P_prim,
        G_prim, max_length)
31     C_out = og.OR(C[og_Length-1],C_prim[og_Length-1])
32
33
34     if False:
35         #print("NUM1Prim")
36         #print(num_1_prim_Bits)
37         #print("NUM2Prim")
38         #print(num_2_prim_Bits)
39         print(C)
40         print(C_prim)
41
42     #sumcompstage
43     S = SumComputationStage(H, C, H_prim, C_prim, C_out)
44
45     if True:
46         print("Liczba 1:")
47         og.printNum(num_1_Bits)
48         print("Liczba 2:")
49         og.printNum(num_2_Bits)
50         print("Modulo:")

```

---

```
51         print(2**((len(num_1_Bits)) - k))
52         print("Wynik")
53         og.printNum(S)
54
55
56
57 def numsPrim(number_1_Bit, number_2_Bit, modulo_bits):
58     num1Prim = []
59     num2Prim = [False]
60     for i in range(0, len(number_1_Bit)):
61         if modulo_bits[i]:
62             num1Prim.append(not og.XOR(number_1_Bit[i],
63                                         number_2_Bit[i]))
64             num2Prim.append(og.OR(number_1_Bit[i],
65                                   number_2_Bit[i]))
66         else:
67             num1Prim.append(og.XOR(number_1_Bit[i],
68                                   number_2_Bit[i]))
69             num2Prim.append(og.AND(number_1_Bit[i],
70                                   number_2_Bit[i]))
71     num2Prim.pop()
72     return num1Prim, num2Prim
73
74 def SumCoptutationStage(H, C, H_prim, C_prim, C_out):
75     S = []
76     if not C_out:
77         S.append(H[0])
78     else:
79         S.append(H_prim[0])
80
81     for i in range(1, len(H)):
82         if not C_out:
83             S.append(og.XOR(H[i], C[i-1]))
84         else:
85             S.append(og.XOR(H_prim[i], C_prim[i-1]))
```

```
82
83     return S
84
85 run(99, 0, 55)
```

Listing 6.3: Wersja  $2^n + k$  (Python)

```
1 import LadnerFischer as og
2 from array import *
3 from math import ceil, log2
4
5 def run(number_1, number_2, k):
6
7     num_1_Bits = og.bitify(number_1)
8     num_2_Bits = og.bitify(number_2)
9
10
11     num_1_Bits.reverse()
12     num_2_Bits.reverse()
13
14     max_length = max(len(num_1_Bits), len(num_2_Bits))
15     og_Length = max_length
16
17
18     k_Bits = og.bitify(2**((max_length+1) -k))
19     k_Bits.reverse()
20
21     num_1_Bits = num_1_Bits + [False] * (max_length - len
22                                         (num_1_Bits) + 1)
23     num_2_Bits = num_2_Bits + [False] * (max_length - len
24                                         (num_2_Bits) + 1)
25     k_Bits = k_Bits + [False] * (max_length - len(k_Bits)
26                                 + 1)
27
28     k_Bits[max_length] = False
29     num_1_prim_Bits, num_2_prim_Bits = numsPrim(
30         num_1_Bits, num_2_Bits, k_Bits)
```



---

```

26
27     #preprocessing stage
28     H, P, G = og.preProcessingStage(num_1_Bits,
29                                     num_2_Bits)
30
31     #parallelprefix stage
32     C = og.parallelPrefixStage(H,P,G, max_length)
33     C_prim = og.parallelPrefixStage(H_prim, P_prim,
34                                     G_prim, max_length)
35
36     C_out = og.OR(C_prim[og_Length-1],C_prim[og_Length
37                 -1])
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

---

```
57
58     #sumcompstage
59     S = SumCpmutationStage(H, C, H_prim, C_prim, C_out)
60
61     if True:
62         print("Liczba 1:")
63         og.printNum(num_1_Bits)
64         print("Liczba 2:")
65         og.printNum(num_2_Bits)
66         print("Wynik")
67         og.printNum(S)
68
69
70
71 def numsPrim(number_1_Bit, number_2_Bit, modulo_bits):
72     num1Prim = []
73     num2Prim = [False]
74     for i in range(0, len(number_1_Bit)):
75         if modulo_bits[i]:
76             num1Prim.append(not og.XOR(number_1_Bit[i],
77                                         number_2_Bit[i]))
78             num2Prim.append(og.OR(number_1_Bit[i],
79                                    number_2_Bit[i]))
79         else:
80             num1Prim.append(og.XOR(number_1_Bit[i],
81                                     number_2_Bit[i]))
82             num2Prim.append(og.AND(number_1_Bit[i],
83                                     number_2_Bit[i]))
84     num2Prim.pop()
85     return num1Prim, num2Prim
86
87 def SumCpmutationStage(H, C, H_prim, C_prim, C_out):
88     S = []
89     if not C_out:
90         S.append(H[0])
```

---

```

88     else:
89         S.append(H_prim[0])
90
91     for i in range(1, len(H)):
92         if not C_out:
93             S.append(og.XOR(H[i],C[i-1]))
94         else:
95             S.append(og.XOR(H_prim[i], C_prim[i-1]))
96
97     return S
98
99 run(127, 127, 120)

```

## 6.2 Verilog

Listing 6.4: Sumator w języku Verilog

```

1 module SumCoptationZero(
2     input H,
3     input H_prim,
4     input C_out,
5     output S
6 );
7     assign S = (C_out == 1'b1)? H : H_prim;
8 endmodule
9
10 module SumCoptationNormal(
11     input H,
12     input H_prim,
13     input C,
14     input C_prim,
15     input C_out,
16     output S
17 );
18     assign S = (C_out == 1'b1)? H ^ C : H_prim ^ C_prim;
19 endmodule

```

---

```

20
21 module SumComputation #(parameter N = 8) (
22     input [N : 0] C,
23     input [N : 0] C_prim,
24     input [N - 1 : 0] H,
25     input [N - 1 : 0] H_prim,
26
27     output [N - 1 : 0] sum
28 );
29     genvar i;
30     SumComputationZero zero(
31         .H(H[0]),
32         .H_prim(H_prim[0]),
33         .C_out(C[N] | C_prim[N]),
34         .S(sum[0])
35     );
36
37     for (i = 1; i < N; i = i + 1) begin
38         SumComputationNormal norm(
39             .H(H[i]),
40             .H_prim(H_prim[i]),
41             .C(C[i]),
42             .C_prim(C_prim[i]),
43             .C_out(C[N] | C_prim[N]),
44             .S(sum[i])
45         );
46     end
47 endmodule
48
49 module ParallelPrefixSingle (
50     input G,
51     input P,
52     input P_prev,
53     input G_prev,
54     output G_out,

```

---

```

55     output P_out
56 );
57     assign P_out = P & P_prev;
58     assign G_out = G | (G_prev & P);
59 endmodule
60
61
62 module ParallelPrefix #(parameter N = 8) (
63     input [N - 1 : 0] G,
64     input [N - 1 : 0] P,
65     input [N - 1 : 0] G_prim,
66     input [N - 1 : 0] P_prim,
67
68     output [N : 0] C,
69     output [N : 0] C_prim
70 );
71     parameter levels = $clog2(N);
72     genvar i;
73     genvar j;
74     wire [N-1: 0] G_W[levels: 0], P_W[levels: 0],
75         P_prim_W[levels: 0], G_prim_W[levels: 0];
76
77     for (i = 0; i < N; i = i + 1) begin
78         assign G_W[0][i] = G[i];
79         assign G_prim_W[0][i] = G_prim[i];
80         assign P_W[0][i] = P[i];
81         assign P_prim_W[0][i] = P_prim[i];
82     end
83
84     for ( i = 0; i < levels; i = i + 1) begin
85         for ( j = 0; j < N; j = j + 1) begin
86             if(j[i+1] == 1'b1) begin
87                 localparam index_of_Source = j - (j %
                    (2**i) + 1);

```

---

```

88
89     ParallelPrefixSingle prefixNormal(
90         .G(G_W[i][j]),
91         .P(P_W[i][j]),
92         .G_prev(G_W[i][index_of_Source]),
93         .P_prev(P_W[i][index_of_Source]),
94         .G_out(G_W[i + 1][j]),
95         .P_out(P_W[i + 1][j])
96     );
97
98     ParallelPrefixSingle prefixPrim(
99         .G(G_prim_W[i][j]),
100        .P(P_prim_W[i][j]),
101        .G_prev(G_prim_W[i][index_of_Source])
102        ,
103        .P_prev(P_prim_W[i][index_of_Source])
104        ,
105        .G_out(G_prim_W[i + 1][j]),
106        .P_out(P_prim_W[i + 1][j])
107    );
108    end
109
110    else begin
111        assign G_W[i+1][j] = G_W[i][j];
112        assign G_prim_W[i+1][j] = G_prim_W[i][j];
113        assign P_W[i+1][j] = P_W[i][j];
114        assign P_prim_W[i+1][j] = P_prim_W[i][j];
115    end
116    end
117
118    end
119
120    assign C[0] = 0;
121    assign C_prim[0] = 0;
122    for (i = 1; i <= N; i = i + 1) begin
123        assign C[i] = G_W[levels][i-1];

```

---

```

121         assign C_prim[i] = G_prim_W[levels][i-1];
122     end
123
124 endmodule
125
126 module PreProcessingSingle (
127     input num1,
128     input num2,
129
130     output H,
131     output G,
132     output P
133 );
134     assign H = num1 ^ num2;
135     assign G = num1 & num2;
136     assign P = num1 | num2;
137 endmodule
138
139 module PreProcessing #(parameter N = 8) (
140     input [N - 1: 0] num1,
141     input [N - 1: 0] num2,
142     input [N - 1: 0] num1_prim,
143     input [N : 0] num2_prim,
144
145     output [N - 1: 0] H,
146     output [N - 1: 0] G,
147     output [N - 1: 0] P,
148     output [N - 1: 0] H_prim,
149     output [N - 1: 0] G_prim,
150     output [N - 1: 0] P_prim
151 );
152     genvar i;
153     for (i = 0; i < N; i = i + 1) begin
154         PreProcessingSingle preprocessNorm(
155             .num1(num1[i]),

```

---

```

156         .num2(num2[i]),
157         .H(H[i]),
158         .G(G[i]),
159         .P(P[i])
160     );
161
162     PreProcessingSingle preprocessPrim(
163         .num1(num1_prim[i]),
164         .num2(num2_prim[i]),
165         .H(H_prim[i]),
166         .G(G_prim[i]),
167         .P(P_prim[i])
168     );
169 end
170 endmodule
171 module numbersPrimSingle(
172     input num1,
173     input num2,
174     input k,
175
176     output num1_Prim,
177     output num2_Prim
178 );
179     assign num1_Prim = (k == 1'b0) ? num1^num2 : ~(num1 ^
        num2);
180     assign num2_Prim = (k == 1'b0) ? (num1 & num2) : (
        num1 | num2);
181 endmodule
182
183 module numbersPrim #(parameter N = 8) (
184     input [N-1 :0] number1,
185     input [N-1 :0] number2,
186     input [N-1 :0] k,
187
188     output [N-1: 0] number1_prim,

```



---

```

189     output [N: 0] number2_prim
190 );
191     genvar i;
192     assign number2_prim[0] = 1'b0;
193     for (i = 0; i < N; i = i + 1) begin
194         numbersPrimSingle lol(
195             .num1(number1[i]),
196             .num2(number2[i]),
197             .k(k[i]),
198             .num1_Prim(number1_prim[i]),
199             .num2_Prim(number2_prim[i+1])
200         );
201     end
202 endmodule
203
204 module hello_world;
205     parameter N = 6;
206
207     reg [N - 1:0] a;
208     reg [N - 1:0] b;
209     reg [N - 1:0] k;
210
211     wire [N - 1:0] a_prim;
212     wire [N : 0] b_prim;
213
214     wire [N - 1:0] H;
215     wire [N - 1:0] G;
216     wire [N - 1:0] P;
217
218     wire [N - 1:0] H_prim;
219     wire [N - 1:0] G_prim;
220     wire [N - 1:0] P_prim;
221
222     wire [N :0] C;
223     wire [N :0] C_prim;
```

---

```
224
225     wire [N - 1 : 0] SUM;
226
227     numbersPrim #(N) numbers_Prim(
228         .number1(a),
229         .number2(b),
230         .k(k),
231         .number1_prim(a_prim),
232         .number2_prim(b_prim)
233     );
234
235     PreProcessing #(N) preprocess(
236         .num1(a),
237         .num2(b),
238         .num1_prim(a_prim),
239         .num2_prim(b_prim),
240
241         .H(H),
242         .G(G),
243         .P(P),
244         .H_prim(H_prim),
245         .G_prim(G_prim),
246         .P_prim(P_prim)
247     );
248
249     ParallelPrefix #(N) parapre(
250         .G(G),
251         .P(P),
252         .G_prim(G_prim),
253         .P_prim(P_prim),
254         .C(C),
255         .C_prim(C_prim)
256     );
257
258     SumComputation #(N) sumcomp(
```

```

259         .H(H),
260         .C(C),
261         .H_prim(H_prim),
262         .C_prim(C_prim),
263         .sum(SUM)
264     );
265
266     initial begin
267         a = 63;
268         b = 0;
269         k = 10;
270
271         $display("Liczba 1: %d, Liczba 2: %d", a, b);
272         #150;
273         $display("Suma modulo: %d", SUM);
274         #150;
275     end
276
277 endmodule

```

## 6.3 Logisim Evolution

Implementacja sumatora w programie Logisim Evolution - plik .circ

Listing 6.5: Struktura Fischera

```

1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <project source="3.8.0" version="1.0">
3   This file is intended to be loaded by Logisim-evolution
4     v3.8.0(https://github.com/logisim-evolution/).
5
6   <lib desc="#Wiring" name="0">
7     <tool name="Pin">
8       <a name="appearance" val="classic"/>
9     </tool>
10  </lib>
11
12  <lib desc="#Gates" name="1"/>

```

---

```
11 <lib desc="#Plexers" name="2"/>
12 <lib desc="#Arithmetic" name="3"/>
13 <lib desc="#Memory" name="4"/>
14 <lib desc="#I/O" name="5"/>
15 <lib desc="#TTL" name="6"/>
16 <lib desc="#TCL" name="7"/>
17 <lib desc="#Base" name="8"/>
18 <lib desc="#BFH-Praktika" name="9"/>
19 <lib desc="#Input/Output-Extra" name="10"/>
20 <lib desc="#Soc" name="11"/>
21 <main name="main"/>
22 <options>
23   <a name="gateUndefined" val="ignore"/>
24   <a name="simlimit" val="1000"/>
25   <a name="simrand" val="0"/>
26 </options>
27 <mappings>
28   <tool lib="8" map="Button2" name="Poke Tool"/>
29   <tool lib="8" map="Button3" name="Menu Tool"/>
30   <tool lib="8" map="Ctrl Button1" name="Menu Tool"/>
31 </mappings>
32 <toolbar>
33   <tool lib="8" name="Poke Tool"/>
34   <tool lib="8" name="Edit Tool"/>
35   <tool lib="8" name="Wiring Tool"/>
36   <tool lib="8" name="Text Tool"/>
37   <sep/>
38   <tool lib="0" name="Pin"/>
39   <tool lib="0" name="Pin">
40     <a name="facing" val="west"/>
41     <a name="output" val="true"/>
42   </tool>
43   <sep/>
44   <tool lib="1" name="NOT Gate"/>
45   <tool lib="1" name="AND Gate"/>
```

```
46     <tool lib="1" name="OR Gate"/>
47     <tool lib="1" name="XOR Gate"/>
48     <tool lib="1" name="NAND Gate"/>
49     <tool lib="1" name="NOR Gate"/>
50     <sep/>
51     <tool lib="4" name="D Flip-Flop"/>
52     <tool lib="4" name="Register"/>
53 </toolbar>
54 <circuit name="main">
55     <a name="appearance" val="logisim_evolution"/>
56     <a name="circuit" val="main"/>
57     <a name="circuitnamedboxfixedsize" val="true"/>
58     <a name="simulationFrequency" val="1.0"/>
59     <comp lib="0" loc="(2040,2570)" name="Constant">
60         <a name="facing" val="south"/>
61         <a name="value" val="0x0"/>
62     </comp>
63     <comp lib="0" loc="(2090,910)" name="Constant">
64         <a name="facing" val="west"/>
65         <a name="value" val="0x0"/>
66     </comp>
67     <comp lib="0" loc="(2100,330)" name="Pin">
68         <a name="appearance" val="classic"/>
69         <a name="facing" val="south"/>
70         <a name="label" val="A6"/>
71     </comp>
72     <comp lib="0" loc="(2100,460)" name="Pin">
73         <a name="appearance" val="classic"/>
74         <a name="facing" val="south"/>
75         <a name="label" val="B6"/>
76     </comp>
77     <comp lib="0" loc="(2130,330)" name="Pin">
78         <a name="appearance" val="classic"/>
79         <a name="facing" val="south"/>
80         <a name="label" val="A5"/>
```

---

```
81 </comp>
82 <comp lib="0" loc="(2130,460)" name="Pin">
83   <a name="appearance" val="classic"/>
84   <a name="facing" val="south"/>
85   <a name="label" val="B5"/>
86 </comp>
87 <comp lib="0" loc="(2160,330)" name="Pin">
88   <a name="appearance" val="classic"/>
89   <a name="facing" val="south"/>
90   <a name="label" val="A4"/>
91 </comp>
92 <comp lib="0" loc="(2160,460)" name="Pin">
93   <a name="appearance" val="classic"/>
94   <a name="facing" val="south"/>
95   <a name="label" val="B4"/>
96 </comp>
97 <comp lib="0" loc="(2190,330)" name="Pin">
98   <a name="appearance" val="classic"/>
99   <a name="facing" val="south"/>
100   <a name="label" val="A3"/>
101 </comp>
102 <comp lib="0" loc="(2190,460)" name="Pin">
103   <a name="appearance" val="classic"/>
104   <a name="facing" val="south"/>
105   <a name="label" val="B3"/>
106 </comp>
107 <comp lib="0" loc="(2220,330)" name="Pin">
108   <a name="appearance" val="classic"/>
109   <a name="facing" val="south"/>
110   <a name="label" val="A2"/>
111 </comp>
112 <comp lib="0" loc="(2220,460)" name="Pin">
113   <a name="appearance" val="classic"/>
114   <a name="facing" val="south"/>
115   <a name="label" val="B2"/>
```

---

```
116 </comp>
117 <comp lib="0" loc="(2250,330)" name="Pin">
118   <a name="appearance" val="classic"/>
119   <a name="facing" val="south"/>
120   <a name="label" val="A1"/>
121 </comp>
122 <comp lib="0" loc="(2250,460)" name="Pin">
123   <a name="appearance" val="classic"/>
124   <a name="facing" val="south"/>
125   <a name="label" val="B1"/>
126 </comp>
127 <comp lib="0" loc="(2280,330)" name="Pin">
128   <a name="appearance" val="classic"/>
129   <a name="facing" val="south"/>
130   <a name="label" val="A0"/>
131 </comp>
132 <comp lib="0" loc="(2280,460)" name="Pin">
133   <a name="appearance" val="classic"/>
134   <a name="facing" val="south"/>
135   <a name="label" val="B0"/>
136 </comp>
137 <comp lib="1" loc="(100,1590)" name="AND Gate">
138   <a name="facing" val="south"/>
139 </comp>
140 <comp lib="1" loc="(100,2000)" name="AND Gate">
141   <a name="facing" val="south"/>
142 </comp>
143 <comp lib="1" loc="(1030,2280)" name="OR Gate">
144   <a name="facing" val="south"/>
145 </comp>
146 <comp lib="1" loc="(1050,1000)" name="AND Gate">
147   <a name="facing" val="south"/>
148   <a name="label" val="G_P4"/>
149 </comp>
150 <comp lib="1" loc="(1050,2200)" name="AND Gate">
```

---

```
151     <a name="facing" val="south"/>
152 </comp>
153 <comp lib="1" loc="(1050,610)" name="AND Gate">
154     <a name="facing" val="south"/>
155     <a name="label" val="G4"/>
156 </comp>
157 <comp lib="1" loc="(1070,1060)" name="NOT Gate">
158     <a name="facing" val="south"/>
159 </comp>
160 <comp lib="1" loc="(1070,670)" name="NOT Gate">
161     <a name="facing" val="south"/>
162 </comp>
163 <comp lib="1" loc="(1090,1110)" name="AND Gate">
164     <a name="facing" val="south"/>
165     <a name="label" val="H_P4"/>
166 </comp>
167 <comp lib="1" loc="(1090,720)" name="AND Gate">
168     <a name="facing" val="south"/>
169     <a name="label" val="H4"/>
170 </comp>
171 <comp lib="1" loc="(1120,2200)" name="AND Gate">
172     <a name="facing" val="south"/>
173 </comp>
174 <comp lib="1" loc="(1140,600)" name="OR Gate">
175     <a name="facing" val="south"/>
176     <a name="label" val="P4"/>
177 </comp>
178 <comp lib="1" loc="(1140,990)" name="OR Gate">
179     <a name="facing" val="south"/>
180     <a name="label" val="P_P4"/>
181 </comp>
182 <comp lib="1" loc="(1150,1430)" name="OR Gate">
183     <a name="facing" val="south"/>
184 </comp>
185 <comp lib="1" loc="(1170,1340)" name="AND Gate">
```



---

```
186     <a name="facing" val="south"/>
187 </comp>
188 <comp lib="1" loc="(1170,1770)" name="AND Gate">
189     <a name="facing" val="south"/>
190 </comp>
191 <comp lib="1" loc="(1180,1850)" name="OR Gate">
192     <a name="facing" val="south"/>
193 </comp>
194 <comp lib="1" loc="(1210,2680)" name="XOR Gate">
195     <a name="facing" val="south"/>
196 </comp>
197 <comp lib="1" loc="(1230,1770)" name="AND Gate">
198     <a name="facing" val="south"/>
199 </comp>
200 <comp lib="1" loc="(1230,900)" name="NOT Gate">
201     <a name="facing" val="south"/>
202 </comp>
203 <comp lib="1" loc="(1240,1350)" name="AND Gate">
204     <a name="facing" val="south"/>
205 </comp>
206 <comp lib="1" loc="(1250,1000)" name="AND Gate">
207     <a name="facing" val="south"/>
208     <a name="label" val="G_P3"/>
209 </comp>
210 <comp lib="1" loc="(1250,610)" name="AND Gate">
211     <a name="facing" val="south"/>
212     <a name="label" val="G3"/>
213 </comp>
214 <comp lib="1" loc="(1260,1930)" name="OR Gate">
215     <a name="facing" val="south"/>
216 </comp>
217 <comp lib="1" loc="(1270,1060)" name="NOT Gate">
218     <a name="facing" val="south"/>
219 </comp>
220 <comp lib="1" loc="(1270,670)" name="NOT Gate">
```

---

```
221     <a name="facing" val="south"/>
222 </comp>
223 <comp lib="1" loc="(1280,1860)" name="AND Gate">
224     <a name="facing" val="south"/>
225 </comp>
226 <comp lib="1" loc="(1290,1110)" name="AND Gate">
227     <a name="facing" val="south"/>
228     <a name="label" val="H_P3"/>
229 </comp>
230 <comp lib="1" loc="(1290,720)" name="AND Gate">
231     <a name="facing" val="south"/>
232     <a name="label" val="H3"/>
233 </comp>
234 <comp lib="1" loc="(1310,1480)" name="OR Gate">
235     <a name="facing" val="south"/>
236 </comp>
237 <comp lib="1" loc="(1320,1420)" name="AND Gate">
238     <a name="facing" val="south"/>
239 </comp>
240 <comp lib="1" loc="(1340,600)" name="OR Gate">
241     <a name="facing" val="south"/>
242     <a name="label" val="P3"/>
243 </comp>
244 <comp lib="1" loc="(1340,990)" name="OR Gate">
245     <a name="facing" val="south"/>
246     <a name="label" val="P_P3"/>
247 </comp>
248 <comp lib="1" loc="(1350,1860)" name="AND Gate">
249     <a name="facing" val="south"/>
250 </comp>
251 <comp lib="1" loc="(1380,1420)" name="AND Gate">
252     <a name="facing" val="south"/>
253 </comp>
254 <comp lib="1" loc="(1440,1860)" name="OR Gate">
255     <a name="facing" val="south"/>
```

---

```
256 </comp>
257 <comp lib="1" loc="(1450,900)" name="NOT Gate">
258   <a name="facing" val="south"/>
259 </comp>
260 <comp lib="1" loc="(1460,1780)" name="AND Gate">
261   <a name="facing" val="south"/>
262 </comp>
263 <comp lib="1" loc="(1470,1000)" name="AND Gate">
264   <a name="facing" val="south"/>
265   <a name="label" val="G_P2"/>
266 </comp>
267 <comp lib="1" loc="(1470,610)" name="AND Gate">
268   <a name="facing" val="south"/>
269   <a name="label" val="G2"/>
270 </comp>
271 <comp lib="1" loc="(1490,1060)" name="NOT Gate">
272   <a name="facing" val="south"/>
273 </comp>
274 <comp lib="1" loc="(1490,2680)" name="XOR Gate">
275   <a name="facing" val="south"/>
276 </comp>
277 <comp lib="1" loc="(1490,670)" name="NOT Gate">
278   <a name="facing" val="south"/>
279 </comp>
280 <comp lib="1" loc="(1510,1110)" name="AND Gate">
281   <a name="facing" val="south"/>
282   <a name="label" val="H_P2"/>
283 </comp>
284 <comp lib="1" loc="(1510,720)" name="AND Gate">
285   <a name="facing" val="south"/>
286   <a name="label" val="H2"/>
287 </comp>
288 <comp lib="1" loc="(1530,1790)" name="AND Gate">
289   <a name="facing" val="south"/>
290 </comp>
```

---

```
291 <comp lib="1" loc="(1560,600)" name="OR Gate">
292   <a name="facing" val="south"/>
293   <a name="label" val="P2"/>
294 </comp>
295 <comp lib="1" loc="(1560,990)" name="OR Gate">
296   <a name="facing" val="south"/>
297   <a name="label" val="P_P2"/>
298 </comp>
299 <comp lib="1" loc="(1610,1940)" name="OR Gate">
300   <a name="facing" val="south"/>
301 </comp>
302 <comp lib="1" loc="(1630,1870)" name="AND Gate">
303   <a name="facing" val="south"/>
304 </comp>
305 <comp lib="1" loc="(1660,900)" name="NOT Gate">
306   <a name="facing" val="south"/>
307 </comp>
308 <comp lib="1" loc="(1680,1000)" name="AND Gate">
309   <a name="facing" val="south"/>
310   <a name="label" val="G_P1"/>
311 </comp>
312 <comp lib="1" loc="(1680,610)" name="AND Gate">
313   <a name="facing" val="south"/>
314   <a name="label" val="G1"/>
315 </comp>
316 <comp lib="1" loc="(170,1590)" name="AND Gate">
317   <a name="facing" val="south"/>
318 </comp>
319 <comp lib="1" loc="(170,2000)" name="AND Gate">
320   <a name="facing" val="south"/>
321 </comp>
322 <comp lib="1" loc="(1700,1060)" name="NOT Gate">
323   <a name="facing" val="south"/>
324 </comp>
325 <comp lib="1" loc="(1700,1870)" name="AND Gate">
```

---

```
326     <a name="facing" val="south"/>
327 </comp>
328 <comp lib="1" loc="(1700,670)" name="NOT Gate">
329     <a name="facing" val="south"/>
330 </comp>
331 <comp lib="1" loc="(1720,1110)" name="AND Gate">
332     <a name="facing" val="south"/>
333     <a name="label" val="H_P1"/>
334 </comp>
335 <comp lib="1" loc="(1720,720)" name="AND Gate">
336     <a name="facing" val="south"/>
337     <a name="label" val="H1"/>
338 </comp>
339 <comp lib="1" loc="(1770,2680)" name="XOR Gate">
340     <a name="facing" val="south"/>
341 </comp>
342 <comp lib="1" loc="(1770,600)" name="OR Gate">
343     <a name="facing" val="south"/>
344     <a name="label" val="P1"/>
345 </comp>
346 <comp lib="1" loc="(1770,990)" name="OR Gate">
347     <a name="facing" val="south"/>
348     <a name="label" val="P_P1"/>
349 </comp>
350 <comp lib="1" loc="(1780,1480)" name="OR Gate">
351     <a name="facing" val="south"/>
352 </comp>
353 <comp lib="1" loc="(1800,1390)" name="AND Gate">
354     <a name="facing" val="south"/>
355 </comp>
356 <comp lib="1" loc="(1870,1390)" name="AND Gate">
357     <a name="facing" val="south"/>
358 </comp>
359 <comp lib="1" loc="(190,2260)" name="OR Gate">
360     <a name="facing" val="south"/>
```

---

```
361     <a name="label" val="C_out"/>
362 </comp>
363 <comp lib="1" loc="(1910,900)" name="NOT Gate">
364     <a name="facing" val="south"/>
365 </comp>
366 <comp lib="1" loc="(1920,1570)" name="OR Gate">
367     <a name="facing" val="south"/>
368 </comp>
369 <comp lib="1" loc="(1930,1000)" name="AND Gate">
370     <a name="facing" val="south"/>
371     <a name="label" val="G_P0"/>
372 </comp>
373 <comp lib="1" loc="(1930,610)" name="AND Gate">
374     <a name="facing" val="south"/>
375     <a name="label" val="G0"/>
376 </comp>
377 <comp lib="1" loc="(1940,1470)" name="AND Gate">
378     <a name="facing" val="south"/>
379 </comp>
380 <comp lib="1" loc="(1950,1060)" name="NOT Gate">
381     <a name="facing" val="south"/>
382 </comp>
383 <comp lib="1" loc="(1950,670)" name="NOT Gate">
384     <a name="facing" val="south"/>
385 </comp>
386 <comp lib="1" loc="(1970,1110)" name="AND Gate">
387     <a name="facing" val="south"/>
388     <a name="label" val="H_P0"/>
389 </comp>
390 <comp lib="1" loc="(1970,720)" name="AND Gate">
391     <a name="facing" val="south"/>
392     <a name="label" val="H0"/>
393 </comp>
394 <comp lib="1" loc="(1990,2680)" name="XOR Gate">
395     <a name="facing" val="south"/>
```

---

```
396 </comp>
397 <comp lib="1" loc="(2000,1470)" name="AND Gate">
398   <a name="facing" val="south"/>
399 </comp>
400 <comp lib="1" loc="(2020,600)" name="OR Gate">
401   <a name="facing" val="south"/>
402   <a name="label" val="P0"/>
403 </comp>
404 <comp lib="1" loc="(2020,990)" name="OR Gate">
405   <a name="facing" val="south"/>
406   <a name="label" val="P_P0"/>
407 </comp>
408 <comp lib="1" loc="(290,1760)" name="OR Gate">
409   <a name="facing" val="south"/>
410 </comp>
411 <comp lib="1" loc="(290,2290)" name="OR Gate">
412   <a name="facing" val="south"/>
413 </comp>
414 <comp lib="1" loc="(310,1690)" name="AND Gate">
415   <a name="facing" val="south"/>
416 </comp>
417 <comp lib="1" loc="(310,2210)" name="AND Gate">
418   <a name="facing" val="south"/>
419 </comp>
420 <comp lib="1" loc="(380,1690)" name="AND Gate">
421   <a name="facing" val="south"/>
422 </comp>
423 <comp lib="1" loc="(380,2210)" name="AND Gate">
424   <a name="facing" val="south"/>
425 </comp>
426 <comp lib="1" loc="(410,2670)" name="XOR Gate">
427   <a name="facing" val="south"/>
428 </comp>
429 <comp lib="1" loc="(480,2080)" name="OR Gate">
430   <a name="facing" val="south"/>
```

---

```
431 </comp>
432 <comp lib="1" loc="(490,1410)" name="OR Gate">
433   <a name="facing" val="south"/>
434 </comp>
435 <comp lib="1" loc="(500,2000)" name="AND Gate">
436   <a name="facing" val="south"/>
437 </comp>
438 <comp lib="1" loc="(510,1350)" name="AND Gate">
439   <a name="facing" val="south"/>
440 </comp>
441 <comp lib="1" loc="(570,2000)" name="AND Gate">
442   <a name="facing" val="south"/>
443 </comp>
444 <comp lib="1" loc="(580,1350)" name="AND Gate">
445   <a name="facing" val="south"/>
446 </comp>
447 <comp lib="1" loc="(640,1500)" name="OR Gate">
448   <a name="facing" val="south"/>
449 </comp>
450 <comp lib="1" loc="(660,1420)" name="AND Gate">
451   <a name="facing" val="south"/>
452 </comp>
453 <comp lib="1" loc="(660,2290)" name="OR Gate">
454   <a name="facing" val="south"/>
455 </comp>
456 <comp lib="1" loc="(680,1000)" name="AND Gate">
457   <a name="facing" val="south"/>
458   <a name="label" val="G_P6"/>
459 </comp>
460 <comp lib="1" loc="(680,2210)" name="AND Gate">
461   <a name="facing" val="south"/>
462 </comp>
463 <comp lib="1" loc="(680,610)" name="AND Gate">
464   <a name="facing" val="south"/>
465   <a name="label" val="G6"/>
```



---

```
466 </comp>
467 <comp lib="1" loc="(700,1060)" name="NOT Gate">
468   <a name="facing" val="south"/>
469 </comp>
470 <comp lib="1" loc="(700,670)" name="NOT Gate">
471   <a name="facing" val="south"/>
472 </comp>
473 <comp lib="1" loc="(720,1110)" name="AND Gate">
474   <a name="facing" val="south"/>
475   <a name="label" val="H_P6"/>
476 </comp>
477 <comp lib="1" loc="(720,2670)" name="XOR Gate">
478   <a name="facing" val="south"/>
479 </comp>
480 <comp lib="1" loc="(720,720)" name="AND Gate">
481   <a name="facing" val="south"/>
482   <a name="label" val="H6"/>
483 </comp>
484 <comp lib="1" loc="(730,1420)" name="AND Gate">
485   <a name="facing" val="south"/>
486 </comp>
487 <comp lib="1" loc="(750,2210)" name="AND Gate">
488   <a name="facing" val="south"/>
489 </comp>
490 <comp lib="1" loc="(770,600)" name="OR Gate">
491   <a name="facing" val="south"/>
492   <a name="label" val="P6"/>
493 </comp>
494 <comp lib="1" loc="(770,990)" name="OR Gate">
495   <a name="facing" val="south"/>
496   <a name="label" val="P_P6"/>
497 </comp>
498 <comp lib="1" loc="(80,1670)" name="OR Gate">
499   <a name="facing" val="south"/>
500 </comp>
```

---

```
501 <comp lib="1" loc="(80,2080)" name="OR Gate">
502   <a name="facing" val="south"/>
503 </comp>
504 <comp lib="1" loc="(850,2090)" name="OR Gate">
505   <a name="facing" val="south"/>
506 </comp>
507 <comp lib="1" loc="(860,1000)" name="AND Gate">
508   <a name="facing" val="south"/>
509   <a name="label" val="G_P5"/>
510 </comp>
511 <comp lib="1" loc="(860,610)" name="AND Gate">
512   <a name="facing" val="south"/>
513   <a name="label" val="G5"/>
514 </comp>
515 <comp lib="1" loc="(880,1060)" name="NOT Gate">
516   <a name="facing" val="south"/>
517 </comp>
518 <comp lib="1" loc="(880,2010)" name="AND Gate">
519   <a name="facing" val="south"/>
520 </comp>
521 <comp lib="1" loc="(880,670)" name="NOT Gate">
522   <a name="facing" val="south"/>
523 </comp>
524 <comp lib="1" loc="(900,1110)" name="AND Gate">
525   <a name="facing" val="south"/>
526   <a name="label" val="H_P5"/>
527 </comp>
528 <comp lib="1" loc="(900,720)" name="AND Gate">
529   <a name="facing" val="south"/>
530   <a name="label" val="H5"/>
531 </comp>
532 <comp lib="1" loc="(950,2010)" name="AND Gate">
533   <a name="facing" val="south"/>
534 </comp>
535 <comp lib="1" loc="(950,600)" name="OR Gate">
```

---

```
536     <a name="facing" val="south"/>
537     <a name="label" val="P5"/>
538 </comp>
539 <comp lib="1" loc="(950,990)" name="OR Gate">
540     <a name="facing" val="south"/>
541     <a name="label" val="P_P5"/>
542 </comp>
543 <comp lib="1" loc="(990,2670)" name="XOR Gate">
544     <a name="facing" val="south"/>
545 </comp>
546 <comp lib="2" loc="(1030,2580)" name="Multiplexer">
547     <a name="facing" val="south"/>
548 </comp>
549 <comp lib="2" loc="(1180,2580)" name="Multiplexer">
550     <a name="facing" val="south"/>
551 </comp>
552 <comp lib="2" loc="(1240,2580)" name="Multiplexer">
553     <a name="facing" val="south"/>
554 </comp>
555 <comp lib="2" loc="(1460,2580)" name="Multiplexer">
556     <a name="facing" val="south"/>
557 </comp>
558 <comp lib="2" loc="(1520,2580)" name="Multiplexer">
559     <a name="facing" val="south"/>
560 </comp>
561 <comp lib="2" loc="(1740,2580)" name="Multiplexer">
562     <a name="facing" val="south"/>
563 </comp>
564 <comp lib="2" loc="(1800,2580)" name="Multiplexer">
565     <a name="facing" val="south"/>
566 </comp>
567 <comp lib="2" loc="(1960,2580)" name="Multiplexer">
568     <a name="facing" val="south"/>
569 </comp>
570 <comp lib="2" loc="(380,2580)" name="Multiplexer">
```

---

```
571     <a name="facing" val="south"/>
572 </comp>
573 <comp lib="2" loc="(440,2580)" name="Multiplexer">
574     <a name="facing" val="south"/>
575 </comp>
576 <comp lib="2" loc="(690,2580)" name="Multiplexer">
577     <a name="facing" val="south"/>
578 </comp>
579 <comp lib="2" loc="(750,2580)" name="Multiplexer">
580     <a name="facing" val="south"/>
581 </comp>
582 <comp lib="2" loc="(970,2580)" name="Multiplexer">
583     <a name="facing" val="south"/>
584 </comp>
585 <comp lib="5" loc="(2330,330)" name="LED">
586     <a name="facing" val="south"/>
587 </comp>
588 <comp lib="5" loc="(2330,460)" name="LED">
589     <a name="facing" val="south"/>
590 </comp>
591 <comp lib="5" loc="(2350,330)" name="LED">
592     <a name="facing" val="south"/>
593 </comp>
594 <comp lib="5" loc="(2350,460)" name="LED">
595     <a name="facing" val="south"/>
596 </comp>
597 <comp lib="5" loc="(2370,330)" name="LED">
598     <a name="facing" val="south"/>
599 </comp>
600 <comp lib="5" loc="(2370,460)" name="LED">
601     <a name="facing" val="south"/>
602 </comp>
603 <comp lib="5" loc="(2390,330)" name="LED">
604     <a name="facing" val="south"/>
605 </comp>
```

---

```
606 <comp lib="5" loc="(2390,460)" name="LED">
607   <a name="facing" val="south"/>
608 </comp>
609 <comp lib="5" loc="(2410,330)" name="LED">
610   <a name="facing" val="south"/>
611 </comp>
612 <comp lib="5" loc="(2410,460)" name="LED">
613   <a name="facing" val="south"/>
614 </comp>
615 <comp lib="5" loc="(2430,330)" name="LED">
616   <a name="facing" val="south"/>
617 </comp>
618 <comp lib="5" loc="(2430,460)" name="LED">
619   <a name="facing" val="south"/>
620 </comp>
621 <comp lib="5" loc="(2450,330)" name="LED">
622   <a name="facing" val="south"/>
623 </comp>
624 <comp lib="5" loc="(2450,460)" name="LED">
625   <a name="facing" val="south"/>
626 </comp>
627 <comp lib="5" loc="(2500,330)" name="LED">
628   <a name="facing" val="south"/>
629 </comp>
630 <comp lib="5" loc="(2520,330)" name="LED">
631   <a name="facing" val="south"/>
632 </comp>
633 <comp lib="5" loc="(2540,330)" name="LED">
634   <a name="facing" val="south"/>
635 </comp>
636 <comp lib="5" loc="(2560,330)" name="LED">
637   <a name="facing" val="south"/>
638 </comp>
639 <comp lib="5" loc="(2580,330)" name="LED">
640   <a name="facing" val="south"/>
```

---

```
641 </comp>
642 <comp lib="5" loc="(2600,330)" name="LED">
643   <a name="facing" val="south"/>
644 </comp>
645 <comp lib="5" loc="(2620,330)" name="LED">
646   <a name="facing" val="south"/>
647 </comp>
648 <comp lib="8" loc="(2250,593)" name="Text">
649   <a name="text" val="k = 15"/>
650 </comp>
651 <comp lib="8" loc="(2263,613)" name="Text">
652   <a name="text" val=" 0001111"/>
653 </comp>
654 <wire from="(100,1590)" to="(100,1620)"/>
655 <wire from="(100,2000)" to="(100,2030)"/>
656 <wire from="(1000,2320)" to="(1000,2350)"/>
657 <wire from="(1000,2320)" to="(1030,2320)"/>
658 <wire from="(1010,1290)" to="(1010,2230)"/>
659 <wire from="(1010,2560)" to="(1160,2560)"/>
660 <wire from="(1010,2600)" to="(1010,2610)"/>
661 <wire from="(1010,2600)" to="(1030,2600)"/>
662 <wire from="(1010,740)" to="(1010,760)"/>
663 <wire from="(1010,740)" to="(1050,740)"/>
664 <wire from="(1010,760)" to="(1010,1290)"/>
665 <wire from="(1020,2350)" to="(1020,2550)"/>
666 <wire from="(1020,2350)" to="(1210,2350)"/>
667 <wire from="(1030,2110)" to="(1030,2150)"/>
668 <wire from="(1030,2110)" to="(1260,2110)"/>
669 <wire from="(1030,2280)" to="(1030,2320)"/>
670 <wire from="(1030,2580)" to="(1030,2600)"/>
671 <wire from="(1030,360)" to="(1030,560)"/>
672 <wire from="(1030,360)" to="(1120,360)"/>
673 <wire from="(1030,800)" to="(1030,920)"/>
674 <wire from="(1030,800)" to="(1090,800)"/>
675 <wire from="(1030,920)" to="(1030,950)"/>
```

---

```
676 <wire from="(1030,920)" to="(1120,920)"/>
677 <wire from="(1040,2360)" to="(1040,2550)"/>
678 <wire from="(1040,2360)" to="(1180,2360)"/>
679 <wire from="(1050,1000)" to="(1050,1010)"/>
680 <wire from="(1050,1010)" to="(1050,1120)"/>
681 <wire from="(1050,1010)" to="(1070,1010)"/>
682 <wire from="(1050,1150)" to="(1050,1280)"/>
683 <wire from="(1050,1150)" to="(1140,1150)"/>
684 <wire from="(1050,1280)" to="(1050,1930)"/>
685 <wire from="(1050,2200)" to="(1050,2230)"/>
686 <wire from="(1050,610)" to="(1050,620)"/>
687 <wire from="(1050,620)" to="(1050,740)"/>
688 <wire from="(1050,620)" to="(1070,620)"/>
689 <wire from="(1070,1010)" to="(1070,1030)"/>
690 <wire from="(1070,1270)" to="(1070,1350)"/>
691 <wire from="(1070,1270)" to="(1180,1270)"/>
692 <wire from="(1070,1350)" to="(1070,2140)"/>
693 <wire from="(1070,2140)" to="(1070,2150)"/>
694 <wire from="(1070,2140)" to="(1100,2140)"/>
695 <wire from="(1070,490)" to="(1070,560)"/>
696 <wire from="(1070,490)" to="(1160,490)"/>
697 <wire from="(1070,620)" to="(1070,640)"/>
698 <wire from="(1070,910)" to="(1070,950)"/>
699 <wire from="(1070,910)" to="(1160,910)"/>
700 <wire from="(1090,1110)" to="(1090,1200)"/>
701 <wire from="(1090,1200)" to="(2260,1200)"/>
702 <wire from="(1090,720)" to="(1090,800)"/>
703 <wire from="(1090,800)" to="(2370,800)"/>
704 <wire from="(1100,2140)" to="(1100,2150)"/>
705 <wire from="(1110,1020)" to="(1110,1060)"/>
706 <wire from="(1110,1020)" to="(1140,1020)"/>
707 <wire from="(1110,630)" to="(1110,670)"/>
708 <wire from="(1110,630)" to="(1140,630)"/>
709 <wire from="(1120,1620)" to="(1120,1790)"/>
710 <wire from="(1120,1620)" to="(1150,1620)"/>
```

---

```
711 <wire from="(1120,1790)" to="(1160,1790)"/>
712 <wire from="(1120,2200)" to="(1120,2320)"/>
713 <wire from="(1120,360)" to="(1120,550)"/>
714 <wire from="(1120,360)" to="(2160,360)"/>
715 <wire from="(1120,920)" to="(1120,940)"/>
716 <wire from="(1130,1170)" to="(1130,1380)"/>
717 <wire from="(1130,1170)" to="(1250,1170)"/>
718 <wire from="(1140,1020)" to="(1140,1150)"/>
719 <wire from="(1140,2130)" to="(1140,2150)"/>
720 <wire from="(1140,2130)" to="(1350,2130)"/>
721 <wire from="(1140,600)" to="(1140,630)"/>
722 <wire from="(1140,630)" to="(1140,710)"/>
723 <wire from="(1140,710)" to="(1140,760)"/>
724 <wire from="(1140,710)" to="(1180,710)"/>
725 <wire from="(1140,990)" to="(1140,1020)"/>
726 <wire from="(1150,1250)" to="(1150,1290)"/>
727 <wire from="(1150,1250)" to="(1470,1250)"/>
728 <wire from="(1150,1430)" to="(1150,1620)"/>
729 <wire from="(1150,1650)" to="(1150,1720)"/>
730 <wire from="(1150,1650)" to="(1440,1650)"/>
731 <wire from="(1160,1790)" to="(1160,1800)"/>
732 <wire from="(1160,2560)" to="(1220,2560)"/>
733 <wire from="(1160,490)" to="(1160,550)"/>
734 <wire from="(1160,490)" to="(2160,490)"/>
735 <wire from="(1160,760)" to="(1160,910)"/>
736 <wire from="(1160,760)" to="(1380,760)"/>
737 <wire from="(1160,910)" to="(1160,940)"/>
738 <wire from="(1170,1340)" to="(1170,1380)"/>
739 <wire from="(1170,1770)" to="(1170,1780)"/>
740 <wire from="(1170,1780)" to="(1200,1780)"/>
741 <wire from="(1170,2500)" to="(1170,2550)"/>
742 <wire from="(1170,2500)" to="(2380,2500)"/>
743 <wire from="(1180,1850)" to="(1180,1860)"/>
744 <wire from="(1180,1860)" to="(1180,2360)"/>
745 <wire from="(1180,2580)" to="(1180,2590)"/>
```



---

```
746 <wire from="(1180,2590)" to="(1190,2590)"/>
747 <wire from="(1180,710)" to="(1180,1270)"/>
748 <wire from="(1190,1280)" to="(1190,1290)"/>
749 <wire from="(1190,1280)" to="(1210,1280)"/>
750 <wire from="(1190,1710)" to="(1190,1720)"/>
751 <wire from="(1190,1710)" to="(1200,1710)"/>
752 <wire from="(1190,2410)" to="(1190,2550)"/>
753 <wire from="(1190,2410)" to="(2270,2410)"/>
754 <wire from="(1190,2590)" to="(1190,2620)"/>
755 <wire from="(120,1510)" to="(120,1540)"/>
756 <wire from="(120,1510)" to="(150,1510)"/>
757 <wire from="(120,1940)" to="(120,1950)"/>
758 <wire from="(120,1940)" to="(150,1940)"/>
759 <wire from="(1200,1460)" to="(1200,1710)"/>
760 <wire from="(1200,1460)" to="(1240,1460)"/>
761 <wire from="(1200,1710)" to="(1210,1710)"/>
762 <wire from="(1200,1780)" to="(1200,1800)"/>
763 <wire from="(1210,1260)" to="(1210,1280)"/>
764 <wire from="(1210,1260)" to="(1340,1260)"/>
765 <wire from="(1210,1280)" to="(1220,1280)"/>
766 <wire from="(1210,1710)" to="(1210,1720)"/>
767 <wire from="(1210,1820)" to="(1210,1940)"/>
768 <wire from="(1210,1820)" to="(1230,1820)"/>
769 <wire from="(1210,2300)" to="(1210,2350)"/>
770 <wire from="(1210,2300)" to="(1260,2300)"/>
771 <wire from="(1210,2680)" to="(1210,2730)"/>
772 <wire from="(1210,2730)" to="(2530,2730)"/>
773 <wire from="(1220,1280)" to="(1220,1300)"/>
774 <wire from="(1220,2560)" to="(1440,2560)"/>
775 <wire from="(1230,1770)" to="(1230,1820)"/>
776 <wire from="(1230,2360)" to="(1230,2550)"/>
777 <wire from="(1230,2360)" to="(1480,2360)"/>
778 <wire from="(1230,2600)" to="(1230,2620)"/>
779 <wire from="(1230,2600)" to="(1240,2600)"/>
780 <wire from="(1230,370)" to="(1230,560)"/>
```

---

```
781 <wire from="(1230,370)" to="(1320,370)"/>
782 <wire from="(1230,810)" to="(1230,870)"/>
783 <wire from="(1230,810)" to="(1290,810)"/>
784 <wire from="(1230,900)" to="(1230,920)"/>
785 <wire from="(1230,920)" to="(1230,950)"/>
786 <wire from="(1230,920)" to="(1320,920)"/>
787 <wire from="(1240,1350)" to="(1240,1460)"/>
788 <wire from="(1240,1870)" to="(1240,1880)"/>
789 <wire from="(1240,1870)" to="(1250,1870)"/>
790 <wire from="(1240,2580)" to="(1240,2600)"/>
791 <wire from="(1250,1000)" to="(1250,1010)"/>
792 <wire from="(1250,1010)" to="(1250,1170)"/>
793 <wire from="(1250,1010)" to="(1270,1010)"/>
794 <wire from="(1250,1720)" to="(1260,1720)"/>
795 <wire from="(1250,1780)" to="(1250,1870)"/>
796 <wire from="(1250,1780)" to="(1300,1780)"/>
797 <wire from="(1250,2320)" to="(1250,2550)"/>
798 <wire from="(1250,2320)" to="(1440,2320)"/>
799 <wire from="(1250,610)" to="(1250,620)"/>
800 <wire from="(1250,620)" to="(1250,730)"/>
801 <wire from="(1250,620)" to="(1270,620)"/>
802 <wire from="(1250,730)" to="(1370,730)"/>
803 <wire from="(1260,1280)" to="(1260,1300)"/>
804 <wire from="(1260,1280)" to="(1560,1280)"/>
805 <wire from="(1260,1680)" to="(1260,1720)"/>
806 <wire from="(1260,1680)" to="(1550,1680)"/>
807 <wire from="(1260,1790)" to="(1260,1810)"/>
808 <wire from="(1260,1790)" to="(1280,1790)"/>
809 <wire from="(1260,1930)" to="(1260,2110)"/>
810 <wire from="(1260,2110)" to="(1260,2300)"/>
811 <wire from="(1270,1010)" to="(1270,1030)"/>
812 <wire from="(1270,500)" to="(1270,560)"/>
813 <wire from="(1270,500)" to="(1360,500)"/>
814 <wire from="(1270,620)" to="(1270,640)"/>
815 <wire from="(1270,900)" to="(1270,950)"/>
```

---

```
816 <wire from="(1270,900)" to="(1350,900)"/>
817 <wire from="(1280,1690)" to="(1280,1790)"/>
818 <wire from="(1280,1690)" to="(1610,1690)"/>
819 <wire from="(1280,1860)" to="(1280,1880)"/>
820 <wire from="(1290,1110)" to="(1290,1210)"/>
821 <wire from="(1290,1210)" to="(2270,1210)"/>
822 <wire from="(1290,1310)" to="(1290,1430)"/>
823 <wire from="(1290,1310)" to="(1370,1310)"/>
824 <wire from="(1290,720)" to="(1290,810)"/>
825 <wire from="(1290,810)" to="(2380,810)"/>
826 <wire from="(130,2190)" to="(130,2330)"/>
827 <wire from="(130,2190)" to="(170,2190)"/>
828 <wire from="(1300,1320)" to="(1300,1370)"/>
829 <wire from="(1300,1320)" to="(1390,1320)"/>
830 <wire from="(1300,1500)" to="(1300,1780)"/>
831 <wire from="(1300,1500)" to="(1310,1500)"/>
832 <wire from="(1300,1790)" to="(1300,1810)"/>
833 <wire from="(1300,1790)" to="(1320,1790)"/>
834 <wire from="(1310,1020)" to="(1310,1060)"/>
835 <wire from="(1310,1020)" to="(1340,1020)"/>
836 <wire from="(1310,1480)" to="(1310,1500)"/>
837 <wire from="(1310,630)" to="(1310,670)"/>
838 <wire from="(1310,630)" to="(1340,630)"/>
839 <wire from="(1320,1420)" to="(1320,1430)"/>
840 <wire from="(1320,1430)" to="(1330,1430)"/>
841 <wire from="(1320,1770)" to="(1320,1790)"/>
842 <wire from="(1320,1770)" to="(1350,1770)"/>
843 <wire from="(1320,1790)" to="(1330,1790)"/>
844 <wire from="(1320,370)" to="(1320,550)"/>
845 <wire from="(1320,370)" to="(2190,370)"/>
846 <wire from="(1320,920)" to="(1320,940)"/>
847 <wire from="(1330,1790)" to="(1330,1810)"/>
848 <wire from="(1340,1020)" to="(1340,1260)"/>
849 <wire from="(1340,1360)" to="(1340,1370)"/>
850 <wire from="(1340,1360)" to="(1360,1360)"/>
```

---

```
851 <wire from="(1340,600)" to="(1340,630)"/>
852 <wire from="(1340,630)" to="(1380,630)"/>
853 <wire from="(1340,990)" to="(1340,1020)"/>
854 <wire from="(1350,1430)" to="(1350,1770)"/>
855 <wire from="(1350,1430)" to="(1380,1430)"/>
856 <wire from="(1350,1860)" to="(1350,2130)"/>
857 <wire from="(1350,860)" to="(1350,900)"/>
858 <wire from="(1350,860)" to="(1540,860)"/>
859 <wire from="(1350,900)" to="(1360,900)"/>
860 <wire from="(1360,1350)" to="(1360,1360)"/>
861 <wire from="(1360,1350)" to="(1380,1350)"/>
862 <wire from="(1360,1360)" to="(1360,1370)"/>
863 <wire from="(1360,500)" to="(1360,550)"/>
864 <wire from="(1360,500)" to="(2190,500)"/>
865 <wire from="(1360,900)" to="(1360,940)"/>
866 <wire from="(1370,1700)" to="(1370,1810)"/>
867 <wire from="(1370,1700)" to="(1720,1700)"/>
868 <wire from="(1370,730)" to="(1370,1310)"/>
869 <wire from="(1380,1420)" to="(1380,1430)"/>
870 <wire from="(1380,630)" to="(1380,760)"/>
871 <wire from="(1380,760)" to="(1380,1350)"/>
872 <wire from="(1390,750)" to="(1390,1320)"/>
873 <wire from="(1390,750)" to="(1470,750)"/>
874 <wire from="(1400,1340)" to="(1400,1370)"/>
875 <wire from="(1400,1340)" to="(1610,1340)"/>
876 <wire from="(1420,1630)" to="(1420,1810)"/>
877 <wire from="(1420,1630)" to="(1470,1630)"/>
878 <wire from="(1440,1650)" to="(1440,1730)"/>
879 <wire from="(1440,1650)" to="(1780,1650)"/>
880 <wire from="(1440,1860)" to="(1440,2320)"/>
881 <wire from="(1440,2560)" to="(1500,2560)"/>
882 <wire from="(1450,2510)" to="(1450,2550)"/>
883 <wire from="(1450,2510)" to="(2390,2510)"/>
884 <wire from="(1450,380)" to="(1450,560)"/>
885 <wire from="(1450,380)" to="(1540,380)"/>
```

---

```
886 <wire from="(1450,820)" to="(1450,870)"/>
887 <wire from="(1450,820)" to="(1510,820)"/>
888 <wire from="(1450,900)" to="(1450,920)"/>
889 <wire from="(1450,920)" to="(1450,950)"/>
890 <wire from="(1450,920)" to="(1540,920)"/>
891 <wire from="(1460,1780)" to="(1460,1810)"/>
892 <wire from="(1460,2580)" to="(1460,2590)"/>
893 <wire from="(1460,2590)" to="(1470,2590)"/>
894 <wire from="(1470,1000)" to="(1470,1010)"/>
895 <wire from="(1470,1010)" to="(1470,1250)"/>
896 <wire from="(1470,1010)" to="(1490,1010)"/>
897 <wire from="(1470,1250)" to="(1470,1630)"/>
898 <wire from="(1470,2420)" to="(1470,2550)"/>
899 <wire from="(1470,2420)" to="(2280,2420)"/>
900 <wire from="(1470,2590)" to="(1470,2620)"/>
901 <wire from="(1470,610)" to="(1470,620)"/>
902 <wire from="(1470,620)" to="(1470,730)"/>
903 <wire from="(1470,620)" to="(1490,620)"/>
904 <wire from="(1470,730)" to="(1470,750)"/>
905 <wire from="(1470,730)" to="(1600,730)"/>
906 <wire from="(1480,1720)" to="(1480,1730)"/>
907 <wire from="(1480,1720)" to="(1500,1720)"/>
908 <wire from="(1480,2320)" to="(1480,2360)"/>
909 <wire from="(1480,2320)" to="(1610,2320)"/>
910 <wire from="(1490,1010)" to="(1490,1030)"/>
911 <wire from="(1490,2680)" to="(1490,2740)"/>
912 <wire from="(1490,2740)" to="(2540,2740)"/>
913 <wire from="(1490,510)" to="(1490,560)"/>
914 <wire from="(1490,510)" to="(1580,510)"/>
915 <wire from="(1490,620)" to="(1490,640)"/>
916 <wire from="(1490,900)" to="(1490,950)"/>
917 <wire from="(1490,900)" to="(1580,900)"/>
918 <wire from="(150,1150)" to="(150,1510)"/>
919 <wire from="(150,1150)" to="(770,1150)"/>
920 <wire from="(150,1510)" to="(150,1540)"/>
```

---

```
921 <wire from="(150,1940)" to="(150,1950)"/>
922 <wire from="(150,1940)" to="(170,1940)"/>
923 <wire from="(1500,1590)" to="(1500,1720)"/>
924 <wire from="(1500,1590)" to="(1560,1590)"/>
925 <wire from="(1500,1720)" to="(1510,1720)"/>
926 <wire from="(1500,2560)" to="(1720,2560)"/>
927 <wire from="(1510,1110)" to="(1510,1220)"/>
928 <wire from="(1510,1220)" to="(2280,1220)"/>
929 <wire from="(1510,1720)" to="(1510,1740)"/>
930 <wire from="(1510,2350)" to="(1510,2550)"/>
931 <wire from="(1510,2350)" to="(1670,2350)"/>
932 <wire from="(1510,2610)" to="(1510,2620)"/>
933 <wire from="(1510,2610)" to="(1520,2610)"/>
934 <wire from="(1510,720)" to="(1510,820)"/>
935 <wire from="(1510,820)" to="(2390,820)"/>
936 <wire from="(1520,2580)" to="(1520,2610)"/>
937 <wire from="(1530,1020)" to="(1530,1060)"/>
938 <wire from="(1530,1020)" to="(1560,1020)"/>
939 <wire from="(1530,2360)" to="(1530,2550)"/>
940 <wire from="(1530,2360)" to="(1710,2360)"/>
941 <wire from="(1530,630)" to="(1530,670)"/>
942 <wire from="(1530,630)" to="(1560,630)"/>
943 <wire from="(1540,380)" to="(1540,550)"/>
944 <wire from="(1540,380)" to="(2220,380)"/>
945 <wire from="(1540,700)" to="(1540,860)"/>
946 <wire from="(1540,700)" to="(1610,700)"/>
947 <wire from="(1540,860)" to="(1540,870)"/>
948 <wire from="(1540,920)" to="(1540,940)"/>
949 <wire from="(1550,1680)" to="(1550,1740)"/>
950 <wire from="(1550,1680)" to="(1870,1680)"/>
951 <wire from="(1560,1020)" to="(1560,1280)"/>
952 <wire from="(1560,1280)" to="(1560,1590)"/>
953 <wire from="(1560,600)" to="(1560,630)"/>
954 <wire from="(1560,630)" to="(1610,630)"/>
955 <wire from="(1560,990)" to="(1560,1020)"/>
```

---

```
956 <wire from="(1580,510)" to="(1580,550)"/>
957 <wire from="(1580,510)" to="(2220,510)"/>
958 <wire from="(1580,760)" to="(1580,900)"/>
959 <wire from="(1580,760)" to="(1840,760)"/>
960 <wire from="(1580,900)" to="(1580,940)"/>
961 <wire from="(1590,1860)" to="(1590,1890)"/>
962 <wire from="(1590,1860)" to="(1600,1860)"/>
963 <wire from="(1600,730)" to="(1600,1860)"/>
964 <wire from="(1610,1340)" to="(1610,1610)"/>
965 <wire from="(1610,1610)" to="(1650,1610)"/>
966 <wire from="(1610,1690)" to="(1610,1820)"/>
967 <wire from="(1610,1690)" to="(1920,1690)"/>
968 <wire from="(1610,1940)" to="(1610,2320)"/>
969 <wire from="(1610,630)" to="(1610,700)"/>
970 <wire from="(1610,700)" to="(1610,1340)"/>
971 <wire from="(1620,1420)" to="(1900,1420)"/>
972 <wire from="(1620,750)" to="(1620,1420)"/>
973 <wire from="(1620,750)" to="(1680,750)"/>
974 <wire from="(1630,1870)" to="(1630,1890)"/>
975 <wire from="(1650,1610)" to="(1650,1800)"/>
976 <wire from="(1650,1800)" to="(1650,1820)"/>
977 <wire from="(1650,1800)" to="(1680,1800)"/>
978 <wire from="(1660,390)" to="(1660,560)"/>
979 <wire from="(1660,390)" to="(1750,390)"/>
980 <wire from="(1660,830)" to="(1660,870)"/>
981 <wire from="(1660,830)" to="(1720,830)"/>
982 <wire from="(1660,900)" to="(1660,920)"/>
983 <wire from="(1660,920)" to="(1660,950)"/>
984 <wire from="(1660,920)" to="(1750,920)"/>
985 <wire from="(1670,2330)" to="(1670,2350)"/>
986 <wire from="(1670,2330)" to="(1800,2330)"/>
987 <wire from="(1680,1000)" to="(1680,1010)"/>
988 <wire from="(1680,1010)" to="(1680,1370)"/>
989 <wire from="(1680,1010)" to="(1700,1010)"/>
990 <wire from="(1680,1370)" to="(1760,1370)"/>
```

---

```
991 <wire from="(1680,1800)" to="(1680,1820)"/>
992 <wire from="(1680,610)" to="(1680,620)"/>
993 <wire from="(1680,620)" to="(1680,750)"/>
994 <wire from="(1680,620)" to="(1700,620)"/>
995 <wire from="(170,1590)" to="(170,1940)"/>
996 <wire from="(170,2190)" to="(170,2210)"/>
997 <wire from="(1700,1010)" to="(1700,1030)"/>
998 <wire from="(1700,520)" to="(1700,560)"/>
999 <wire from="(1700,520)" to="(1790,520)"/>
1000 <wire from="(1700,620)" to="(1700,640)"/>
1001 <wire from="(1700,900)" to="(1700,950)"/>
1002 <wire from="(1700,900)" to="(1780,900)"/>
1003 <wire from="(1710,2320)" to="(1710,2360)"/>
1004 <wire from="(1710,2320)" to="(1780,2320)"/>
1005 <wire from="(1720,1110)" to="(1720,1230)"/>
1006 <wire from="(1720,1230)" to="(2290,1230)"/>
1007 <wire from="(1720,1700)" to="(1720,1820)"/>
1008 <wire from="(1720,1700)" to="(2000,1700)"/>
1009 <wire from="(1720,2560)" to="(1780,2560)"/>
1010 <wire from="(1720,720)" to="(1720,830)"/>
1011 <wire from="(1720,830)" to="(2400,830)"/>
1012 <wire from="(1730,2520)" to="(1730,2550)"/>
1013 <wire from="(1730,2520)" to="(2400,2520)"/>
1014 <wire from="(1740,1020)" to="(1740,1060)"/>
1015 <wire from="(1740,1020)" to="(1770,1020)"/>
1016 <wire from="(1740,2580)" to="(1740,2600)"/>
1017 <wire from="(1740,2600)" to="(1750,2600)"/>
1018 <wire from="(1740,630)" to="(1740,670)"/>
1019 <wire from="(1740,630)" to="(1770,630)"/>
1020 <wire from="(1750,2430)" to="(1750,2550)"/>
1021 <wire from="(1750,2430)" to="(2290,2430)"/>
1022 <wire from="(1750,2600)" to="(1750,2620)"/>
1023 <wire from="(1750,390)" to="(1750,550)"/>
1024 <wire from="(1750,390)" to="(2250,390)"/>
1025 <wire from="(1750,920)" to="(1750,940)"/>
```



---

```
1026 <wire from="(1760,1370)" to="(1760,1430)"/>
1027 <wire from="(1770,1020)" to="(1770,1320)"/>
1028 <wire from="(1770,1320)" to="(1850,1320)"/>
1029 <wire from="(1770,2680)" to="(1770,2750)"/>
1030 <wire from="(1770,2750)" to="(2550,2750)"/>
1031 <wire from="(1770,600)" to="(1770,630)"/>
1032 <wire from="(1770,630)" to="(1840,630)"/>
1033 <wire from="(1770,990)" to="(1770,1020)"/>
1034 <wire from="(1780,1310)" to="(1780,1340)"/>
1035 <wire from="(1780,1310)" to="(1930,1310)"/>
1036 <wire from="(1780,1480)" to="(1780,1650)"/>
1037 <wire from="(1780,1650)" to="(1780,2320)"/>
1038 <wire from="(1780,2560)" to="(1940,2560)"/>
1039 <wire from="(1780,860)" to="(1780,900)"/>
1040 <wire from="(1780,860)" to="(2120,860)"/>
1041 <wire from="(1780,900)" to="(1790,900)"/>
1042 <wire from="(1790,2340)" to="(1790,2550)"/>
1043 <wire from="(1790,2340)" to="(2050,2340)"/>
1044 <wire from="(1790,2600)" to="(1790,2620)"/>
1045 <wire from="(1790,2600)" to="(1800,2600)"/>
1046 <wire from="(1790,520)" to="(1790,550)"/>
1047 <wire from="(1790,520)" to="(2250,520)"/>
1048 <wire from="(1790,900)" to="(1790,940)"/>
1049 <wire from="(1800,1390)" to="(1800,1430)"/>
1050 <wire from="(1800,2320)" to="(1800,2330)"/>
1051 <wire from="(1800,2320)" to="(1920,2320)"/>
1052 <wire from="(1800,2580)" to="(1800,2600)"/>
1053 <wire from="(1810,2360)" to="(1810,2550)"/>
1054 <wire from="(1810,2360)" to="(2070,2360)"/>
1055 <wire from="(1820,1330)" to="(1820,1340)"/>
1056 <wire from="(1820,1330)" to="(1850,1330)"/>
1057 <wire from="(1840,1280)" to="(1960,1280)"/>
1058 <wire from="(1840,630)" to="(1840,760)"/>
1059 <wire from="(1840,760)" to="(1840,1280)"/>
1060 <wire from="(1850,1320)" to="(1850,1330)"/>
```

---

```
1061 <wire from="(1850,1330)" to="(1850,1340)"/>
1062 <wire from="(1870,1270)" to="(1910,1270)"/>
1063 <wire from="(1870,1390)" to="(1870,1680)"/>
1064 <wire from="(1870,750)" to="(1870,1270)"/>
1065 <wire from="(1870,750)" to="(1930,750)"/>
1066 <wire from="(1890,1300)" to="(1890,1340)"/>
1067 <wire from="(1890,1300)" to="(2020,1300)"/>
1068 <wire from="(190,1500)" to="(190,1540)"/>
1069 <wire from="(190,1500)" to="(580,1500)"/>
1070 <wire from="(190,1940)" to="(190,1950)"/>
1071 <wire from="(190,1940)" to="(590,1940)"/>
1072 <wire from="(190,2260)" to="(190,2560)"/>
1073 <wire from="(190,2560)" to="(360,2560)"/>
1074 <wire from="(1900,1420)" to="(1900,1520)"/>
1075 <wire from="(1910,1270)" to="(1910,1390)"/>
1076 <wire from="(1910,1390)" to="(1910,1410)"/>
1077 <wire from="(1910,1390)" to="(2050,1390)"/>
1078 <wire from="(1910,1410)" to="(1920,1410)"/>
1079 <wire from="(1910,400)" to="(1910,560)"/>
1080 <wire from="(1910,400)" to="(2000,400)"/>
1081 <wire from="(1910,840)" to="(1910,870)"/>
1082 <wire from="(1910,840)" to="(1970,840)"/>
1083 <wire from="(1910,900)" to="(1910,920)"/>
1084 <wire from="(1910,920)" to="(1910,950)"/>
1085 <wire from="(1910,920)" to="(2000,920)"/>
1086 <wire from="(1920,1410)" to="(1920,1420)"/>
1087 <wire from="(1920,1570)" to="(1920,1690)"/>
1088 <wire from="(1920,1690)" to="(1920,2320)"/>
1089 <wire from="(1930,1000)" to="(1930,1010)"/>
1090 <wire from="(1930,1010)" to="(1930,1260)"/>
1091 <wire from="(1930,1010)" to="(1950,1010)"/>
1092 <wire from="(1930,1260)" to="(1930,1310)"/>
1093 <wire from="(1930,1260)" to="(2070,1260)"/>
1094 <wire from="(1930,610)" to="(1930,620)"/>
1095 <wire from="(1930,620)" to="(1930,750)"/>
```

---

```
1096 <wire from="(1930,620)" to="(1950,620)"/>
1097 <wire from="(1940,1470)" to="(1940,1520)"/>
1098 <wire from="(1950,1010)" to="(1950,1030)"/>
1099 <wire from="(1950,2530)" to="(1950,2550)"/>
1100 <wire from="(1950,2530)" to="(2410,2530)"/>
1101 <wire from="(1950,530)" to="(1950,560)"/>
1102 <wire from="(1950,530)" to="(2040,530)"/>
1103 <wire from="(1950,620)" to="(1950,640)"/>
1104 <wire from="(1950,910)" to="(1950,950)"/>
1105 <wire from="(1950,910)" to="(2040,910)"/>
1106 <wire from="(1960,1280)" to="(1960,1410)"/>
1107 <wire from="(1960,1410)" to="(1960,1420)"/>
1108 <wire from="(1960,1410)" to="(1980,1410)"/>
1109 <wire from="(1960,2580)" to="(1960,2590)"/>
1110 <wire from="(1960,2590)" to="(1970,2590)"/>
1111 <wire from="(1970,1110)" to="(1970,1240)"/>
1112 <wire from="(1970,1240)" to="(2300,1240)"/>
1113 <wire from="(1970,2440)" to="(1970,2550)"/>
1114 <wire from="(1970,2440)" to="(2300,2440)"/>
1115 <wire from="(1970,2590)" to="(1970,2620)"/>
1116 <wire from="(1970,720)" to="(1970,840)"/>
1117 <wire from="(1970,840)" to="(2410,840)"/>
1118 <wire from="(1980,1410)" to="(1980,1420)"/>
1119 <wire from="(1990,1020)" to="(1990,1060)"/>
1120 <wire from="(1990,1020)" to="(2020,1020)"/>
1121 <wire from="(1990,2680)" to="(1990,2760)"/>
1122 <wire from="(1990,2760)" to="(2560,2760)"/>
1123 <wire from="(1990,630)" to="(1990,670)"/>
1124 <wire from="(1990,630)" to="(2020,630)"/>
1125 <wire from="(2000,1470)" to="(2000,1700)"/>
1126 <wire from="(2000,400)" to="(2000,550)"/>
1127 <wire from="(2000,400)" to="(2280,400)"/>
1128 <wire from="(2000,920)" to="(2000,940)"/>
1129 <wire from="(2010,2600)" to="(2010,2620)"/>
1130 <wire from="(2010,2600)" to="(2040,2600)"/>
```

---

```
1131 <wire from="(2020,1020)" to="(2020,1300)"/>
1132 <wire from="(2020,1410)" to="(2020,1420)"/>
1133 <wire from="(2020,1410)" to="(2120,1410)"/>
1134 <wire from="(2020,600)" to="(2020,630)"/>
1135 <wire from="(2020,630)" to="(2120,630)"/>
1136 <wire from="(2020,990)" to="(2020,1020)"/>
1137 <wire from="(2040,2570)" to="(2040,2600)"/>
1138 <wire from="(2040,530)" to="(2040,550)"/>
1139 <wire from="(2040,530)" to="(2280,530)"/>
1140 <wire from="(2040,910)" to="(2040,940)"/>
1141 <wire from="(2040,910)" to="(2090,910)"/>
1142 <wire from="(2050,1390)" to="(2050,2340)"/>
1143 <wire from="(2070,1260)" to="(2070,2360)"/>
1144 <wire from="(210,2190)" to="(210,2210)"/>
1145 <wire from="(210,2190)" to="(240,2190)"/>
1146 <wire from="(2100,330)" to="(2100,340)"/>
1147 <wire from="(2100,340)" to="(2330,340)"/>
1148 <wire from="(2100,460)" to="(2100,470)"/>
1149 <wire from="(2100,470)" to="(2330,470)"/>
1150 <wire from="(2120,630)" to="(2120,860)"/>
1151 <wire from="(2120,860)" to="(2120,1410)"/>
1152 <wire from="(2130,330)" to="(2130,350)"/>
1153 <wire from="(2130,350)" to="(2350,350)"/>
1154 <wire from="(2130,460)" to="(2130,480)"/>
1155 <wire from="(2130,480)" to="(2350,480)"/>
1156 <wire from="(2160,330)" to="(2160,360)"/>
1157 <wire from="(2160,360)" to="(2370,360)"/>
1158 <wire from="(2160,460)" to="(2160,490)"/>
1159 <wire from="(2160,490)" to="(2370,490)"/>
1160 <wire from="(2190,330)" to="(2190,370)"/>
1161 <wire from="(2190,370)" to="(2390,370)"/>
1162 <wire from="(2190,460)" to="(2190,500)"/>
1163 <wire from="(2190,500)" to="(2390,500)"/>
1164 <wire from="(2220,330)" to="(2220,380)"/>
1165 <wire from="(2220,380)" to="(2410,380)"/>
```

---

```
1166 <wire from="(2220,460)" to="(2220,510)"/>
1167 <wire from="(2220,510)" to="(2410,510)"/>
1168 <wire from="(2240,1180)" to="(2240,2380)"/>
1169 <wire from="(2250,1190)" to="(2250,2390)"/>
1170 <wire from="(2250,330)" to="(2250,390)"/>
1171 <wire from="(2250,390)" to="(2430,390)"/>
1172 <wire from="(2250,460)" to="(2250,520)"/>
1173 <wire from="(2250,520)" to="(2430,520)"/>
1174 <wire from="(2260,1200)" to="(2260,2400)"/>
1175 <wire from="(2270,1210)" to="(2270,2410)"/>
1176 <wire from="(2280,1220)" to="(2280,2420)"/>
1177 <wire from="(2280,330)" to="(2280,400)"/>
1178 <wire from="(2280,400)" to="(2450,400)"/>
1179 <wire from="(2280,460)" to="(2280,530)"/>
1180 <wire from="(2280,530)" to="(2450,530)"/>
1181 <wire from="(2290,1230)" to="(2290,2430)"/>
1182 <wire from="(2300,1240)" to="(2300,2440)"/>
1183 <wire from="(2330,330)" to="(2330,340)"/>
1184 <wire from="(2330,460)" to="(2330,470)"/>
1185 <wire from="(2350,330)" to="(2350,350)"/>
1186 <wire from="(2350,460)" to="(2350,480)"/>
1187 <wire from="(2350,780)" to="(2350,2470)"/>
1188 <wire from="(2360,790)" to="(2360,2480)"/>
1189 <wire from="(2370,330)" to="(2370,360)"/>
1190 <wire from="(2370,460)" to="(2370,490)"/>
1191 <wire from="(2370,800)" to="(2370,2490)"/>
1192 <wire from="(2380,810)" to="(2380,2500)"/>
1193 <wire from="(2390,330)" to="(2390,370)"/>
1194 <wire from="(2390,460)" to="(2390,500)"/>
1195 <wire from="(2390,820)" to="(2390,2510)"/>
1196 <wire from="(240,2190)" to="(240,2340)"/>
1197 <wire from="(240,2340)" to="(290,2340)"/>
1198 <wire from="(2400,830)" to="(2400,2520)"/>
1199 <wire from="(2410,330)" to="(2410,380)"/>
1200 <wire from="(2410,460)" to="(2410,510)"/>
```

---

```
1201 <wire from="(2410,840)" to="(2410,2530)"/>
1202 <wire from="(2430,330)" to="(2430,390)"/>
1203 <wire from="(2430,460)" to="(2430,520)"/>
1204 <wire from="(2450,330)" to="(2450,400)"/>
1205 <wire from="(2450,460)" to="(2450,530)"/>
1206 <wire from="(2500,330)" to="(2500,2700)"/>
1207 <wire from="(2510,340)" to="(2510,2710)"/>
1208 <wire from="(2510,340)" to="(2520,340)"/>
1209 <wire from="(2520,330)" to="(2520,340)"/>
1210 <wire from="(2520,350)" to="(2520,2720)"/>
1211 <wire from="(2520,350)" to="(2540,350)"/>
1212 <wire from="(2530,360)" to="(2530,2730)"/>
1213 <wire from="(2530,360)" to="(2560,360)"/>
1214 <wire from="(2540,330)" to="(2540,350)"/>
1215 <wire from="(2540,370)" to="(2540,2740)"/>
1216 <wire from="(2540,370)" to="(2580,370)"/>
1217 <wire from="(2550,380)" to="(2550,2750)"/>
1218 <wire from="(2550,380)" to="(2600,380)"/>
1219 <wire from="(2560,330)" to="(2560,360)"/>
1220 <wire from="(2560,390)" to="(2560,2760)"/>
1221 <wire from="(2560,390)" to="(2620,390)"/>
1222 <wire from="(2580,330)" to="(2580,370)"/>
1223 <wire from="(2600,330)" to="(2600,380)"/>
1224 <wire from="(2620,330)" to="(2620,390)"/>
1225 <wire from="(270,2080)" to="(270,2240)"/>
1226 <wire from="(270,2080)" to="(290,2080)"/>
1227 <wire from="(270,730)" to="(270,1710)"/>
1228 <wire from="(270,730)" to="(680,730)"/>
1229 <wire from="(290,1550)" to="(290,1640)"/>
1230 <wire from="(290,1550)" to="(670,1550)"/>
1231 <wire from="(290,1760)" to="(290,2080)"/>
1232 <wire from="(290,2110)" to="(290,2160)"/>
1233 <wire from="(290,2110)" to="(660,2110)"/>
1234 <wire from="(290,2290)" to="(290,2340)"/>
1235 <wire from="(30,2380)" to="(390,2380)"/>
```

---

```
1236 <wire from="(30,2390)" to="(700,2390)"/>
1237 <wire from="(30,2400)" to="(980,2400)"/>
1238 <wire from="(30,2410)" to="(1190,2410)"/>
1239 <wire from="(30,2420)" to="(1470,2420)"/>
1240 <wire from="(30,2430)" to="(1750,2430)"/>
1241 <wire from="(30,2440)" to="(1970,2440)"/>
1242 <wire from="(30,2470)" to="(370,2470)"/>
1243 <wire from="(30,2480)" to="(680,2480)"/>
1244 <wire from="(30,2490)" to="(960,2490)"/>
1245 <wire from="(30,2500)" to="(1170,2500)"/>
1246 <wire from="(30,2510)" to="(1450,2510)"/>
1247 <wire from="(30,2520)" to="(1730,2520)"/>
1248 <wire from="(30,2530)" to="(1950,2530)"/>
1249 <wire from="(310,1690)" to="(310,1710)"/>
1250 <wire from="(310,2210)" to="(310,2240)"/>
1251 <wire from="(330,1620)" to="(330,1640)"/>
1252 <wire from="(330,1620)" to="(360,1620)"/>
1253 <wire from="(330,2150)" to="(330,2160)"/>
1254 <wire from="(330,2150)" to="(350,2150)"/>
1255 <wire from="(350,2130)" to="(350,2150)"/>
1256 <wire from="(350,2130)" to="(380,2130)"/>
1257 <wire from="(350,2150)" to="(360,2150)"/>
1258 <wire from="(360,1620)" to="(360,1640)"/>
1259 <wire from="(360,2150)" to="(360,2160)"/>
1260 <wire from="(360,2560)" to="(420,2560)"/>
1261 <wire from="(360,750)" to="(360,1620)"/>
1262 <wire from="(360,750)" to="(770,750)"/>
1263 <wire from="(370,2470)" to="(2350,2470)"/>
1264 <wire from="(370,2470)" to="(370,2550)"/>
1265 <wire from="(380,1690)" to="(380,2130)"/>
1266 <wire from="(380,2580)" to="(380,2600)"/>
1267 <wire from="(380,2600)" to="(390,2600)"/>
1268 <wire from="(390,2380)" to="(2240,2380)"/>
1269 <wire from="(390,2380)" to="(390,2550)"/>
1270 <wire from="(390,2600)" to="(390,2610)"/>
```

---

```
1271 <wire from="(400,1580)" to="(400,1640)"/>
1272 <wire from="(400,1580)" to="(730,1580)"/>
1273 <wire from="(400,2130)" to="(400,2160)"/>
1274 <wire from="(400,2130)" to="(770,2130)"/>
1275 <wire from="(410,2670)" to="(410,2700)"/>
1276 <wire from="(410,2700)" to="(2500,2700)"/>
1277 <wire from="(420,2560)" to="(670,2560)"/>
1278 <wire from="(430,2360)" to="(430,2550)"/>
1279 <wire from="(430,2360)" to="(660,2360)"/>
1280 <wire from="(430,2600)" to="(430,2610)"/>
1281 <wire from="(430,2600)" to="(440,2600)"/>
1282 <wire from="(440,2580)" to="(440,2600)"/>
1283 <wire from="(450,2340)" to="(450,2550)"/>
1284 <wire from="(450,2340)" to="(480,2340)"/>
1285 <wire from="(460,1830)" to="(460,2030)"/>
1286 <wire from="(460,1830)" to="(490,1830)"/>
1287 <wire from="(470,1160)" to="(470,1360)"/>
1288 <wire from="(470,1160)" to="(860,1160)"/>
1289 <wire from="(480,1860)" to="(480,1950)"/>
1290 <wire from="(480,1860)" to="(860,1860)"/>
1291 <wire from="(480,2080)" to="(480,2340)"/>
1292 <wire from="(490,1270)" to="(490,1300)"/>
1293 <wire from="(490,1270)" to="(970,1270)"/>
1294 <wire from="(490,1410)" to="(490,1460)"/>
1295 <wire from="(490,1460)" to="(490,1830)"/>
1296 <wire from="(500,2000)" to="(500,2030)"/>
1297 <wire from="(510,1350)" to="(510,1360)"/>
1298 <wire from="(520,1920)" to="(520,1950)"/>
1299 <wire from="(520,1920)" to="(550,1920)"/>
1300 <wire from="(530,1290)" to="(530,1300)"/>
1301 <wire from="(530,1290)" to="(560,1290)"/>
1302 <wire from="(550,1830)" to="(550,1920)"/>
1303 <wire from="(550,1830)" to="(580,1830)"/>
1304 <wire from="(550,1920)" to="(550,1950)"/>
1305 <wire from="(550,780)" to="(660,780)"/>
```



---

```
1306 <wire from="(550,790)" to="(840,790)"/>
1307 <wire from="(550,800)" to="(1030,800)"/>
1308 <wire from="(550,810)" to="(1230,810)"/>
1309 <wire from="(550,820)" to="(1450,820)"/>
1310 <wire from="(550,830)" to="(1660,830)"/>
1311 <wire from="(550,840)" to="(1910,840)"/>
1312 <wire from="(560,1180)" to="(720,1180)"/>
1313 <wire from="(560,1190)" to="(900,1190)"/>
1314 <wire from="(560,1200)" to="(1090,1200)"/>
1315 <wire from="(560,1210)" to="(1290,1210)"/>
1316 <wire from="(560,1220)" to="(1510,1220)"/>
1317 <wire from="(560,1230)" to="(1720,1230)"/>
1318 <wire from="(560,1240)" to="(1970,1240)"/>
1319 <wire from="(560,1260)" to="(560,1290)"/>
1320 <wire from="(560,1260)" to="(950,1260)"/>
1321 <wire from="(560,1290)" to="(560,1300)"/>
1322 <wire from="(570,2000)" to="(570,2330)"/>
1323 <wire from="(580,1350)" to="(580,1500)"/>
1324 <wire from="(580,1500)" to="(580,1830)"/>
1325 <wire from="(580,340)" to="(660,340)"/>
1326 <wire from="(590,1940)" to="(590,1950)"/>
1327 <wire from="(590,1940)" to="(970,1940)"/>
1328 <wire from="(590,350)" to="(840,350)"/>
1329 <wire from="(590,360)" to="(1030,360)"/>
1330 <wire from="(590,370)" to="(1230,370)"/>
1331 <wire from="(590,380)" to="(1450,380)"/>
1332 <wire from="(590,390)" to="(1660,390)"/>
1333 <wire from="(590,400)" to="(1910,400)"/>
1334 <wire from="(590,470)" to="(700,470)"/>
1335 <wire from="(590,480)" to="(880,480)"/>
1336 <wire from="(590,490)" to="(1070,490)"/>
1337 <wire from="(590,500)" to="(1270,500)"/>
1338 <wire from="(590,510)" to="(1490,510)"/>
1339 <wire from="(590,520)" to="(1700,520)"/>
1340 <wire from="(590,530)" to="(1950,530)"/>
```

---

```
1341 <wire from="(60,1120)" to="(60,1620)"/>
1342 <wire from="(60,1120)" to="(680,1120)"/>
1343 <wire from="(60,1810)" to="(60,2030)"/>
1344 <wire from="(60,1810)" to="(80,1810)"/>
1345 <wire from="(600,1280)" to="(1050,1280)"/>
1346 <wire from="(600,1280)" to="(600,1300)"/>
1347 <wire from="(620,1320)" to="(620,1450)"/>
1348 <wire from="(620,1320)" to="(830,1320)"/>
1349 <wire from="(640,1290)" to="(1010,1290)"/>
1350 <wire from="(640,1290)" to="(640,1370)"/>
1351 <wire from="(640,1500)" to="(640,1510)"/>
1352 <wire from="(640,1510)" to="(670,1510)"/>
1353 <wire from="(640,2080)" to="(640,2240)"/>
1354 <wire from="(640,2080)" to="(670,2080)"/>
1355 <wire from="(660,1420)" to="(660,1450)"/>
1356 <wire from="(660,2110)" to="(1030,2110)"/>
1357 <wire from="(660,2110)" to="(660,2160)"/>
1358 <wire from="(660,2290)" to="(660,2360)"/>
1359 <wire from="(660,340)" to="(660,560)"/>
1360 <wire from="(660,340)" to="(750,340)"/>
1361 <wire from="(660,780)" to="(660,920)"/>
1362 <wire from="(660,780)" to="(720,780)"/>
1363 <wire from="(660,920)" to="(660,950)"/>
1364 <wire from="(660,920)" to="(750,920)"/>
1365 <wire from="(670,1510)" to="(670,1550)"/>
1366 <wire from="(670,1550)" to="(670,2080)"/>
1367 <wire from="(670,2560)" to="(730,2560)"/>
1368 <wire from="(680,1000)" to="(680,1010)"/>
1369 <wire from="(680,1010)" to="(680,1120)"/>
1370 <wire from="(680,1010)" to="(700,1010)"/>
1371 <wire from="(680,1360)" to="(680,1370)"/>
1372 <wire from="(680,1360)" to="(710,1360)"/>
1373 <wire from="(680,2210)" to="(680,2240)"/>
1374 <wire from="(680,2480)" to="(2360,2480)"/>
1375 <wire from="(680,2480)" to="(680,2550)"/>
```

---

```
1376 <wire from="(680,610)" to="(680,620)"/>
1377 <wire from="(680,620)" to="(680,730)"/>
1378 <wire from="(680,620)" to="(700,620)"/>
1379 <wire from="(680,730)" to="(680,740)"/>
1380 <wire from="(690,2580)" to="(690,2600)"/>
1381 <wire from="(690,2600)" to="(700,2600)"/>
1382 <wire from="(700,1010)" to="(700,1030)"/>
1383 <wire from="(700,2150)" to="(700,2160)"/>
1384 <wire from="(700,2150)" to="(730,2150)"/>
1385 <wire from="(700,2390)" to="(2250,2390)"/>
1386 <wire from="(700,2390)" to="(700,2550)"/>
1387 <wire from="(700,2600)" to="(700,2610)"/>
1388 <wire from="(700,470)" to="(700,560)"/>
1389 <wire from="(700,470)" to="(790,470)"/>
1390 <wire from="(700,620)" to="(700,640)"/>
1391 <wire from="(700,910)" to="(700,950)"/>
1392 <wire from="(700,910)" to="(790,910)"/>
1393 <wire from="(710,1330)" to="(710,1360)"/>
1394 <wire from="(710,1330)" to="(810,1330)"/>
1395 <wire from="(710,1360)" to="(710,1370)"/>
1396 <wire from="(720,1110)" to="(720,1180)"/>
1397 <wire from="(720,1180)" to="(2240,1180)"/>
1398 <wire from="(720,2670)" to="(720,2710)"/>
1399 <wire from="(720,2710)" to="(2510,2710)"/>
1400 <wire from="(720,720)" to="(720,780)"/>
1401 <wire from="(720,780)" to="(2350,780)"/>
1402 <wire from="(730,1420)" to="(730,1580)"/>
1403 <wire from="(730,1580)" to="(730,2150)"/>
1404 <wire from="(730,2150)" to="(730,2160)"/>
1405 <wire from="(730,2560)" to="(950,2560)"/>
1406 <wire from="(740,1020)" to="(740,1060)"/>
1407 <wire from="(740,1020)" to="(770,1020)"/>
1408 <wire from="(740,2350)" to="(1000,2350)"/>
1409 <wire from="(740,2350)" to="(740,2550)"/>
1410 <wire from="(740,2600)" to="(740,2610)"/>
```

---

```
1411 <wire from="(740,2600)" to="(750,2600)"/>
1412 <wire from="(740,630)" to="(740,670)"/>
1413 <wire from="(740,630)" to="(770,630)"/>
1414 <wire from="(750,1350)" to="(1070,1350)"/>
1415 <wire from="(750,1350)" to="(750,1370)"/>
1416 <wire from="(750,2210)" to="(750,2330)"/>
1417 <wire from="(750,2580)" to="(750,2600)"/>
1418 <wire from="(750,340)" to="(2100,340)"/>
1419 <wire from="(750,340)" to="(750,550)"/>
1420 <wire from="(750,920)" to="(750,940)"/>
1421 <wire from="(760,2320)" to="(760,2550)"/>
1422 <wire from="(760,2320)" to="(890,2320)"/>
1423 <wire from="(770,1020)" to="(770,1150)"/>
1424 <wire from="(770,2130)" to="(1140,2130)"/>
1425 <wire from="(770,2130)" to="(770,2160)"/>
1426 <wire from="(770,600)" to="(770,630)"/>
1427 <wire from="(770,630)" to="(770,750)"/>
1428 <wire from="(770,990)" to="(770,1020)"/>
1429 <wire from="(790,470)" to="(2100,470)"/>
1430 <wire from="(790,470)" to="(790,550)"/>
1431 <wire from="(790,730)" to="(790,910)"/>
1432 <wire from="(790,730)" to="(830,730)"/>
1433 <wire from="(790,910)" to="(790,940)"/>
1434 <wire from="(80,1460)" to="(490,1460)"/>
1435 <wire from="(80,1460)" to="(80,1540)"/>
1436 <wire from="(80,1670)" to="(80,1810)"/>
1437 <wire from="(80,1860)" to="(480,1860)"/>
1438 <wire from="(80,1860)" to="(80,1950)"/>
1439 <wire from="(80,2080)" to="(80,2330)"/>
1440 <wire from="(80,2330)" to="(130,2330)"/>
1441 <wire from="(810,750)" to="(810,1330)"/>
1442 <wire from="(810,750)" to="(950,750)"/>
1443 <wire from="(830,1850)" to="(830,2040)"/>
1444 <wire from="(830,1850)" to="(970,1850)"/>
1445 <wire from="(830,730)" to="(830,1320)"/>
```

---

```
1446 <wire from="(830,730)" to="(860,730)"/>
1447 <wire from="(840,350)" to="(840,560)"/>
1448 <wire from="(840,350)" to="(930,350)"/>
1449 <wire from="(840,790)" to="(840,920)"/>
1450 <wire from="(840,790)" to="(900,790)"/>
1451 <wire from="(840,920)" to="(840,950)"/>
1452 <wire from="(840,920)" to="(930,920)"/>
1453 <wire from="(850,2090)" to="(850,2100)"/>
1454 <wire from="(850,2100)" to="(890,2100)"/>
1455 <wire from="(860,1000)" to="(860,1010)"/>
1456 <wire from="(860,1010)" to="(860,1160)"/>
1457 <wire from="(860,1010)" to="(880,1010)"/>
1458 <wire from="(860,1860)" to="(1180,1860)"/>
1459 <wire from="(860,1860)" to="(860,1960)"/>
1460 <wire from="(860,610)" to="(860,620)"/>
1461 <wire from="(860,620)" to="(860,730)"/>
1462 <wire from="(860,620)" to="(880,620)"/>
1463 <wire from="(860,730)" to="(860,740)"/>
1464 <wire from="(870,2030)" to="(870,2040)"/>
1465 <wire from="(870,2030)" to="(880,2030)"/>
1466 <wire from="(880,1010)" to="(880,1030)"/>
1467 <wire from="(880,2010)" to="(880,2030)"/>
1468 <wire from="(880,480)" to="(880,560)"/>
1469 <wire from="(880,480)" to="(970,480)"/>
1470 <wire from="(880,620)" to="(880,640)"/>
1471 <wire from="(880,910)" to="(880,950)"/>
1472 <wire from="(880,910)" to="(970,910)"/>
1473 <wire from="(890,2100)" to="(890,2320)"/>
1474 <wire from="(900,1110)" to="(900,1190)"/>
1475 <wire from="(900,1190)" to="(2250,1190)"/>
1476 <wire from="(900,1930)" to="(900,1960)"/>
1477 <wire from="(900,1930)" to="(930,1930)"/>
1478 <wire from="(900,720)" to="(900,790)"/>
1479 <wire from="(900,790)" to="(2360,790)"/>
1480 <wire from="(920,1020)" to="(920,1060)"/>
```

---

```
1481 <wire from="(920,1020)" to="(950,1020)"/>
1482 <wire from="(920,630)" to="(920,670)"/>
1483 <wire from="(920,630)" to="(950,630)"/>
1484 <wire from="(930,1930)" to="(1050,1930)"/>
1485 <wire from="(930,1930)" to="(930,1960)"/>
1486 <wire from="(930,350)" to="(2130,350)"/>
1487 <wire from="(930,350)" to="(930,550)"/>
1488 <wire from="(930,920)" to="(930,940)"/>
1489 <wire from="(950,1020)" to="(950,1260)"/>
1490 <wire from="(950,2010)" to="(950,2320)"/>
1491 <wire from="(950,2560)" to="(1010,2560)"/>
1492 <wire from="(950,600)" to="(950,630)"/>
1493 <wire from="(950,630)" to="(950,750)"/>
1494 <wire from="(950,750)" to="(950,760)"/>
1495 <wire from="(950,990)" to="(950,1020)"/>
1496 <wire from="(960,2490)" to="(2370,2490)"/>
1497 <wire from="(960,2490)" to="(960,2550)"/>
1498 <wire from="(970,1120)" to="(1050,1120)"/>
1499 <wire from="(970,1120)" to="(970,1270)"/>
1500 <wire from="(970,1270)" to="(970,1850)"/>
1501 <wire from="(970,1940)" to="(1210,1940)"/>
1502 <wire from="(970,1940)" to="(970,1960)"/>
1503 <wire from="(970,2580)" to="(970,2610)"/>
1504 <wire from="(970,480)" to="(2130,480)"/>
1505 <wire from="(970,480)" to="(970,550)"/>
1506 <wire from="(970,760)" to="(1010,760)"/>
1507 <wire from="(970,760)" to="(970,910)"/>
1508 <wire from="(970,910)" to="(970,940)"/>
1509 <wire from="(980,2400)" to="(2260,2400)"/>
1510 <wire from="(980,2400)" to="(980,2550)"/>
1511 <wire from="(990,2670)" to="(990,2720)"/>
1512 <wire from="(990,2720)" to="(2520,2720)"/>
1513 </circuit>
1514 </project>
```

---

## 6.4 Prompty wykorzystane w rozmowie z chatGPT

Rozmowa z chatGPT dotyczyła głównie działania języka Verilog. Istniała jeszcze jedna rozmowa - dotycząca błędów w verilogu; Jednak pomoc chatGPT, okazała się kompletnie niepoprawna - postanowiłem więc nie zawierać jej w sprawozdaniu.

what is the difference between wire and output in verilog ChatGPT

In the context of Verilog, there are two main types of data objects: wires and outputs. Here's the difference between them:

**Wire:** A wire is a type of signal in Verilog that can be used to connect different components within a module. It is used for interconnecting signals between different modules or within a single module. Wires can be used for both sequential and combinational logic. They are typically used for passing data between modules or for connecting internal signals within a module. Wires are continuous assignments and can be driven by multiple drivers. Wires are not recommended for driving outputs in synthesis unless you explicitly intend to create a bidirectional signal.

**Output:** An output is a type of data object that represents an output port of a Verilog module. Outputs are used to send data from a module to its parent module or the top-level design. Outputs are typically used for representing the primary outputs of a module. They are used for driving external signals or signals that need to be observed outside the module. Outputs are single driven, meaning they can have only one driver. Outputs are typically used for generating outputs in synthesizable designs.

In summary, wires are used for internal interconnections within a module or between modules, while outputs represent the primary outputs of a module and are used for driving external signals. User `K == 1'b1` what does this statement mean in verilog ChatGPT

In Verilog, the statement `K = 1'b1`; assigns the value 1 (logic high) to the variable K. Let's break it down:

K is a variable or signal that is being assigned a value. `1'b1` is a constant value in Verilog that represents a single-bit binary value of 1. The `1'b` prefix indicates that it is a single-bit value. The assignment operator `=` is used to assign the value on the right side of the operator to the variable on the left side.

So, the statement `K = 1'b1`; sets the value of K to logic high (1), indicating that the variable K is assigned the binary value of 1. User how to assign it to false ChatGPT

To assign the value 0 (logic low or false) to a variable or signal in Verilog, you can use the statement `K = 1'b0`; Let's break it down:

K is the variable or signal that you want to assign a value to. `1'b0` represents a single-bit binary value of 0. The `1'b` prefix indicates that it is a single-bit value, and the 0

---

indicates the logic low value.

So, the statement  $K = 1'b0$ ; sets the value of K to logic low (0), indicating that the variable K is assigned the binary value of 0, which represents false. User initial begin  $a = 53$ ;  $b = 60$ ;  $k = 5$ ; assign  $b_{prim}[0] = 1'b0$ ; *for*( $i = 0$ ;  $i < N_{BIT}$ ;  $i++$ )*begin*NumberPrimnumbers $_{prim}(.num1(a[i]$ 1));*end*10;display("end what appears to be the problem here ChatGPT

The code snippet you provided seems to have a problem with the usage of  $k[i]$  in the for loop.

Based on the context, it appears that k is a scalar variable, as you're accessing  $k[i]$  within the loop. However, in Verilog, you cannot directly access individual bits of a scalar variable using an index like  $k[i]$ . Scalar variables are single bits by definition and cannot be indexed.

If you intended k to be an array with multiple bits, you would need to declare it as an array in your code, like  $reg [N_{BIT} - 1 : 0]k$ ; , *where* $N_{BIT}$  *is the desired size of the k array.*

Here's a corrected version of your code snippet, assuming k is an array:

```
verilog
integer i; reg [NBIT - 1 : 0]k;
initial begin a = 53; b = 60; k = 5;
assign bprim[0] = 1'b0;
for (i = 0; i < NBIT; i = i+1)beginNumberPrimnumbers $_{prim}(.num1(a[i]), .num2(b[i]), .k(k[i]), .num1$ 
1));end
10; display("end
```

Make sure to replace  $N_{BIT}$  *with the appropriate size for your k array.* *User module hello<sub>w</sub>orld; parameter N*  
7;

```
reg [NBIT - 1 : 0]a; reg [NBIT - 1 : 0]b; reg [NBIT - 1 : 0]k;
reg [NBIT : 0]bprim; reg [NBIT - 1 : 0]aprim; genvar i;
initial begin a = 53; b = 60; k = 5; assign bprim[0] = 1'b0; for( $i = 0$ ;  $i < N_{BIT}$ ;  $i +$ 
+)beginNumberPrimnumbers $_{prim}(.num1(a[i]), .num2(b[i]), .k(k[i]), .num1_{prim}(a_{prim}[i]), .num2_{prim}(b$ 
1));end10;display("end endmodule ChatGPT
```

The code you provided seems to be incomplete as the module NumberPrim is not defined. Assuming you have defined the NumberPrim module correctly, there are a couple of issues in your code:

The genvar declaration is missing the keyword generate before genvar i. The generate keyword is required when using generate constructs in Verilog.

In the NumberPrim module instantiation, you are using  $b_{prim}[i+1]$  *as an index for num2<sub>prim</sub>, which is in*  
0], *the valid indices for b<sub>prim</sub> are from N<sub>BIT</sub> to 1. Therefore, it should be b<sub>prim</sub>[i] instead of b<sub>prim</sub>[i+*



---

1].

Here's the corrected code with the above issues addressed:

```
verilog
module hello_world; parameter N_BIT = 7;
reg [N_BIT - 1 : 0] a; reg [N_BIT - 1 : 0] b; reg [N_BIT - 1 : 0] k;
reg [N_BIT : 0] b_prim; reg [N_BIT - 1 : 0] a_prim; genvar i;
initial begin a = 53; b = 60; k = 5; assign b_prim[0] = 1'b0; for (generate i = 0; i <
N_BIT; i = i + 1) begin NumberPrim numbers_prim (.num1(a[i]), .num2(b[i]), .k(k[i]), .num1_prim(a_prim[i]), .num2_prim(b
endmodule
```

Please make sure to define the NumberPrim module correctly and provide its implementation to ensure the code functions as expected. User module hello\_world; parameter N\_BIT = 7;

```
reg [N_BIT - 1 : 0] a; reg [N_BIT - 1 : 0] b; reg [N_BIT - 1 : 0] k;
reg [N_BIT : 0] b_prim; reg [N_BIT - 1 : 0] a_prim; genvar i;
initial begin a = 53; b = 60; k = 5; assign b_prim[0] = 1'b0; for (i = 0; i < N_BIT; i =
i + 1) begin
NumberPrim numbers_prim (.num1(a[i]), .num2(b[i]), .k(k[i]), .num1_prim(a_prim[i]), .num2_prim(b
1)); end
10; display("endendmoduleWhatisthesyntaxerrorhereChatGPT
```

The syntax error in your code is that you have not defined the module NumberPrim that is being instantiated within the hello\_world module. The error occurs because Verilog expects to find

To resolve this error, you need to define the NumberPrim module before instantiating it. Here's an example of how you can define the NumberPrim module:

```
verilog
module NumberPrim (
input num1,
input num2,
input k,
output num1_prim,
output num2_prim
); // Define the internal logic of the NumberPrim module here // ...
endmodule
```

Replace the comments with the appropriate internal logic of the NumberPrim module. Once you have defined the NumberPrim module correctly, the syntax error will be resolved.

---

Note: Make sure that the module NumberPrim is defined in a separate file or within the same file before the `hello_worldmodule`. *User But it is defined, the error pops out on the loop initialization line, the*

Apologies for the confusion. I misunderstood your previous response. Upon reevaluating your code, I see that the error lies in the line where you declare the genvar variable `i` within the initial block. In Verilog, genvar declarations must be placed outside procedural blocks like `initial` or `always`. They are typically used within generate blocks.

To fix the error, move the genvar declaration outside the initial block, like this:

```
verilog
module hello_world; parameter N_BIT = 7;
reg [N_BIT - 1 : 0] a; reg [N_BIT - 1 : 0] b; reg [N_BIT - 1 : 0] k;
reg [N_BIT : 0] b_prim; reg [N_BIT - 1 : 0] a_prim; genvar i;
initial begin a = 53; b = 60; k = 5; assign b_prim[0] = 1'b0; for(i = 0; i < N_BIT; i = i +
1) begin NumberPrim numbers_prim(.num1(a[i]), .num2(b[i]), .k(k[i]), .num1_prim(a_prim[i]), .num2_prim(b_p
1)); end; end; display("end endmodule
```

By moving the genvar `i`; declaration outside the initial block, the syntax error should be resolved, and the code should compile without issues. User but the genvar was originally outside the initial block ChatGPT

Apologies for the confusion. You are correct, and I apologize for the incorrect statement in my previous response.

Upon reviewing your code again, I see that the genvar declaration is already correctly placed outside the initial block. I apologize for the incorrect information I provided earlier.

The issue you mentioned with the syntax error might be caused by a different part of your code or by the module NumberPrim. Please ensure that the NumberPrim module is defined correctly with the correct port declarations and that it appears before the instantiation in your code.

If you could provide the definition of the NumberPrim module or any specific error messages you are encountering, I would be happy to assist you further in resolving the issue. User module NumberPrim ( input num1, input num2, input k,

```
output num1_prim, output num2_prim); wire num1_PW, num2_PW; if(k == 1'b1) begin assign num1_PW =
assign num1_prim = num1_PW;
if (k == 1'b1) begin assign num2_PW = num1|num2; end else begin assign num2_PW =
num1num2; end
assign num2_prim = num2_PW;
endmodule there you go ChatGPT
```

Thank you for providing the NumberPrim module definition. The module implemen-

---

tation looks fine, and there doesn't appear to be any syntax errors in it.

Based on the provided code, the NumberPrim module takes inputs num1, num2, and k and provides outputs num1<sub>prim</sub> and num2<sub>prim</sub>. The internal logic of the module performs different operations based on the values of num1 and num2.

Please ensure that the NumberPrim module is defined correctly and appears in the same file or is included/imported into the file where the hello\_world module is defined. Also, make sure the module is correctly instantiated and used.

If you are encountering any specific error messages or issues, please provide them so that I can assist you further. User Can GenVar be causing an issue when initialized in the wrong place (for example in the initial)? ChatGPT

Yes, the placement of genvar declarations is important in Verilog. If a genvar declaration is placed within an initial block, it will result in a syntax error.

genvar declarations are typically used in generate constructs, such as generate blocks or for loops. They are used to create instances of modules or generate procedural blocks during elaboration time.

Here are some key points regarding the placement of genvar declarations:

genvar declarations should be placed outside of any procedural blocks like initial, always, or task.

genvar declarations are commonly placed before the generate block that uses them or at the module scope if they are used in multiple generate blocks.

genvar declarations are not necessary if you are not using any generate constructs in your code.

If you place a genvar declaration within an initial block or any procedural block, it will result in a syntax error. Therefore, it is important to ensure that genvar declarations are placed correctly according to the guidelines mentioned above.

---

## 7 Źródła

### Literatura

- [1] Rozmowa ze sztuczną inteligencją chatgpt.
- [2] Verilog reference guide. [https://www.hdlworks.com/hdl\\_corner/verilog\\_ref/](https://www.hdlworks.com/hdl_corner/verilog_ref/).
- [3] David Dai. Prefix adder schematic. <https://www.youtube.com/watch?v=beDjDXY6o9M&t>.
- [4] Ahmad Hiasat. General modular adder designs for residue number system applications. *IET Circuits, Devices Systems*, 12(4):424–431, 2018.
- [5] Fischer M.J Ladner, E. Parallel prefix computation. *J. ACM*, 27(4):831–838, 1980.