Двухсотпятидесятишестиричный программный табличный последовательный полный сумматор

Так как время суммирования в последовательных табличных программных сумматорах на одноядерных компьютерах и на многоядерных компьютерах без распараллеливания не зависит от количества переносов, то нет нужды и в построении симметричного сумматора, достаточно и несимметричного, который проще симметричного. Следовательно нет нужды и в нечётном основании системы счисления, в том числе и в троичной и в кратных степеням тройки, достаточно и основания системы счисления сумматора кратной степеням двойки.

В исследовании показана возможность построения программного табличного двухсотпятидесятишестиричного несимметричного полного сумматора.

Таблица значений несимметричных двухсотпятидесятишестиритов (байтов):

несимметричные двухсотпятидесятишестириты (байты)
в десятичном коде

255	254	353	252	251	250	249	248	247	246	245	244	243	242	241	240
239	238	237	236	235	234	233	232	231	230	229	228	227	226	225	224
223	222	221	220	219	218	217	216	215	214	213	212	211	210	209	208
207	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192
191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176
175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160
159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144
143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128
127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
95	94	93	92	91	90	89	88	87	86	85	84	83	82	61	80
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

в шестнадцатиричном коде (16X16=256 код исчерпан)

FF	FE	FD	FC	FB	FA	F9	F8	F7	F6	F5	F4	F3	F2	F1	F0
EF	EE	ED	EC	EB	EA	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
DF	DE	DD	DC	DB	DA	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
CF	CE	CD	CC	CB	CA	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
BF	BE	BD	BC	BB	BA	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
AF	AE	AD	AC	AB	AA	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
9F	9E	9D	9C	9B	9A	99	98	97	96	95	94	93	92	91	90
8F	8E	8D	8C	8B	8A	89	88	87	86	85	84	83	82	81	80
7F	7E	7D	7C	7B	7A	79	78	77	76	75	74	73	72	71	70
6F	6E	6D	6C	6B	6A	69	68	67	66	65	64	63	62	61	60
5F	5E	5D	5C	5B	5A	59	58	57	56	55	54	53	52	51	50
4F	4E	4D	4C	4B	4A	49	48	47	46	45	44	43	42	41	40
3F	3E	3D	3C	3B	3A	39	38	37	36	35	34	33	32	31	30
2F	2E	2D	2C	2B	2A	29	28	27	26	25	24	23	22	21	20
1F	1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10
0F	0E	0D	0C	0B	0A	09	08	07	06	05	04	03	02	01	00

в тридцатидвухричном коде (32x32=1024 есть запас для следующего сумматора, а ещё есть трёхбуквенные, четырёхбуквенные и более буквенные слоги и слова)

ЗЯ	ЗЮ	ЗЭ	ЗЪ	ЗЫ	ЗЬ	ЗЩ	ЗШ	ЗЧ	ЗЦ	ЗХ	ЗФ	ЗУ	ЗТ	ЗС	ЗР
ЗП	ЗО	ЗН	ЗМ	ЗЛ	ЗК	ЗЙ	ЗИ	ЗЗ	ЗЖ	ЗЕ	ЗД	ЗГ	ЗВ	ЗБ	ЗА
ЖЯ	ЖЮ	ЖЭ	ЖЪ	ЖЫ	ЖЬ	ЖЩ	ЖШ	ЖЧ	ЖЦ	ЖХ	ЖФ	ЖУ	ЖТ	ЖС	ЖР
ЖП	ЖО	ЖН	ЖМ	ЖЛ	ЖК	ЖЙ	ЖИ	ЖЗ	ЖЖ	ЖЕ	ЖД	ЖГ	ЖВ	ЖБ	ЖА
ЕЯ	ЕЮ	ЕЭ	ЕЪ	ЕЫ	ЕЬ	ЕЩ	ЕШ	ЕЧ	ЕЦ	ЕХ	ЕФ	ЕУ	ЕТ	ЕС	ЕР
ЕП	ЕО	ЕН	ЕМ	ЕЛ	ЕК	ЕЙ	ЕИ	ЕЗ	ЕЖ	ЕЕ	ЕД	ЕГ	ЕВ	ЕБ	ЕА
ДЯ	ДЮ	ДЭ	ДЪ	ДЫ	ДЬ	ДЩ	ДШ	ДЧ	ДЦ	ДХ	ДФ	ДУ	ДТ	ДС	ДР
ДП	ДО	ДН	ДМ	ДЛ	ДК	ДЙ	ДИ	ДЗ	ДЖ	ДЕ	ДД	ДГ	ДВ	ДБ	ДА
ГЯ	ГЮ	ГЭ	ГЪ	ГЫ	ГЬ	ГЩ	ГШ	ГЧ	ГЦ	ГХ	ГФ	ГУ	ГТ	ГС	ГР
ГП	ГО	ГН	ГМ	ГЛ	ГК	ГЙ	ГИ	ГЗ	ГЖ	ГЕ	ГД	ГГ	ГВ	ГБ	ГА

```
ВЯ ВЮ ВЭ ВЪ ВЫ ВЬ ВЩ ВШ ВЧ ВЦ ВХ ВФ ВУ ВТ ВС ВР
ВП ВО ВН ВМ ВЛ ВК ВЙ ВИ ВЗ ВЖ ВЕ ВД ВГ ВВ ВБ ВА
БЯ БЮ БЭ БЪ БЫ БЬ БЩ БШ БЧ БЦ БХ БФ БУ БТ БС БР
БП БО БН БМ БЛ БК БЙ БИ БЗ БЖ БЕ БД БГ БВ ББ БА
АЯ АЮ АЭ АЪ АЫ АЬ АЩ АШ АЧ АЦ АХ АФ АУ АТ АС АР
АП АО АН АМ АЛ АК АЙ АИ АЗ АЖ АЕ АД АГ АВ АБ АА
```

Так как в англолатинском алфавите 26 букв, то англолатинский алфавит лучше подходит для обозначения цифр в системах счисления кратных 27 (26+1=27=3^3).
Так как в русскокириллическом алфавите 33 буквы, то русскокириллический алфавит лучше подходит для обозначения цифр в системах счисления кратных 32 (33-1=32=2^5). Обычно, по традиции, выбрасывают букву **ё**.

Листинг программы последовательного сложения двух 16-ти разрядных 256-ричных несимметричных чисел на табличном 256-тиричном (8-мибитном) полном (трёхаргументном, трёхоперандном) сумматоре на C++ (Dev-C++ 4.9.9.2):

```
//256 Full Adder NonSymmetric
#include <stdio.h>   /* printf */
#include <ctime>     /* clock_t, clock, CLOCKS_PER_SEC */
#include <iostream>  /* system("pause") */
#include <Windows.h> /* SetConsoleTextAttribute */

main()
{
//HANDLE hConsoleHandle = GetStdHandle(STD_OUTPUT_HANDLE);
//    SetConsoleTextAttribute(hConsoleHandle,FOREGROUND_GREEN|
FOREGROUND_INTENSITY);
HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
SetConsoleTextAttribute(hStdOut,FOREGROUND_GREEN|FOREGROUND_INTENSITY);

//adder initialisation
short int Base=256,i,j,k,Sum;
short int F3SumModB[Base][Base][2]; //!
short int F3Carry[Base][Base][2];   //!

for(k=0;k<=1;k=k+1)
{
  for(j=0;j<Base;j=j+1)
  {
    for(i=0;i<Base;i=i+1)
    {
      Sum=i+j+k;
      if (Sum<Base)
      {
        F3SumModB[i][j][k]=Sum;
        F3Carry[i][j][k]=0;
      } else
        {
          F3SumModB[i][j][k]=Sum-Base;
          F3Carry[i][j][k]=1;
        }
    }
  }
}
//system("pause");
//adding numbers
short int Long=16,S[16],C;
//  0 255 255 255  0 255 255  0   0 255   0   0 255   0   0 255
//  0   0 255 255  0   0 255  0   0 255   0   0   2   0   0   1
short int A[16]={255,  0,  0,255,  0,  0,255,  0,  0,255,255,  0,255,255,255,
0};
short int B[16]={  1,  0,  0,  2,  0,  0,255,  0,  0,255,  0,  0,255,255,  0,
0};
for(i=Long-1;i>=0;i=i-1)
{
  printf("%4i",A[i]);
```

```
}
printf("\n");
for(i=Long-1;i>=0;i=i-1)
{
  printf("%4i",B[i]);
}
printf("\n");
printf("----------------------------------------------------------------\n");
//system("pause");
C=0;

unsigned int start_time = clock(); // начальное время
//adding
for(int i=0;i<1000000;i=i+1){
  for(j=0;j<Long;j=j+1)
  {
    S[j]=F3SumModB[A[j]][B[j]][C];
    C=F3Carry[A[j]][B[j]][C];
  }
}
unsigned int end_time = clock();     // конечное время
    unsigned int xsec_time = end_time - start_time; // искомое время [nsec]

for(i=Long-1;i>=0;i=i-1)
{
  printf("%4i",S[i]);
}
printf("\n\n");

//float times1000000000 = (float) xsec_time/CLOCKS_PER_SEC;
//CLOCKS_PER_SEC=1000[clock/sec], 1 clock = 1 msec
printf ("It took %10i clocks[msec]/1000000x16addings\n",xsec_time);
printf ("        %10i nsec/1adding\n",xsec_time/16);
printf ("        %10.3f usec/1adding\n",(float)xsec_time/(1000*16));
printf ("%10f MFPOPS (MegaFixedPointOperationsPerSecond)\n",
(float)1000*16/xsec_time);
printf ("Equivalent frequency = %8.3f MHz\n",(float)1000*16/xsec_time);
printf ("It is in 2*256/Ternary=2*256/3=170.8 times faster when 'Setun'
TernaryHalfAdder\n");
printf ("and in 256/Bin=256/2=128 times faster when Binary Full Adder\n");

getc(stdin); //For END press <Enter>
}
```

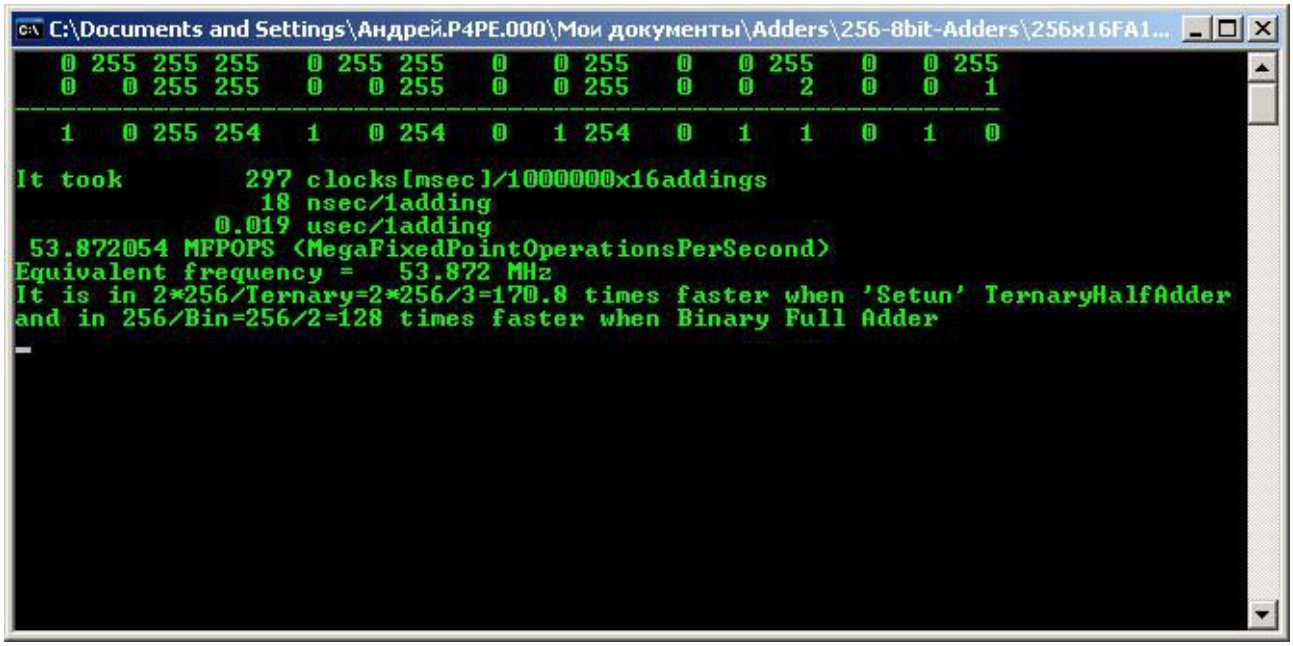

Рис.1. Снимок с экрана результата выполнения программы последовательного суммирования двух шестнадцатиразрядных двухсотпятидесятишестиричнных

(16*8=128 битных) чисел на C++.

16-ти разрядный 256-тиричный (8-мибитный) сумматор **по длине операндов** эквивалентен 16*8=128-ми битному двоичному сумматору.

Одноразрядный двухсотпятидесятишестиричный (8-мибитный) полный сумматор является одной из 256^((256^3)*2)=256^33554432

$$n^{(n^P * R)} = 256^{(256^3 * 2)} = 256^{33554432}$$

тринарных (трёхперендных, трёхаргументных) двухсотпятидесятишестиричных логических функций с бинарным выходом, где n – основание системы счисления, P – количество аргументов (операндов, входов), а R – количество выходов, что на много порядков больше, чем все большие числа Дирака вместе взятые.

Время суммирования двух двусотпятидесятишестиритов равно двум временам считывания результата из двух трёхмерных массивов двухсотпятидесятишестиричного несимметричного полного сумматора в ОЗУ (SRAM).

В программе измеряется время сложения только одного 256-тиричного (8-мибитного) разряда, поэтому сравним его с некоторыми другими одноразрядными полусумматорами и полными сумматорами.

Из-за большего основания системы счисления сумматора (256 вместо 3) и из-за того, что одноразрядный двухсотпятидесятишестиричный несимметричный полный (трёхоперандный, трёхаргументный) сумматор за одно сложение (в приведённом примере приблизительно за 20-30 наносекунд) складывает два двухсотпятидесятишестирита и бит переноса, а не два трита за два сложения, как в одноразрядном троичном симметричном полусумматоре эвм "Сетунь" и "Сетунь-70" Соболева и Брусенцова, то **одноразрядный двухсотпятидесятишестиричный несимметричный полный (трёхоперандный, трёхаргументный) сумматор теоретикологикоматематически в 2*256/3≈170,7 раза быстрее одноразрядного троичного симметричного полусумматора эвм "Сетунь" и "Сетунь-70" Соболева и Брусенцова**.

Из-за большего основания системы счисления сумматора (256 вместо 2) **одноразрядный двухсотпятидесятишестиричный несимметричный полный сумматор в 256/2=128 раз быстрее одноразрядного двоичного полного сумматора**.

Из-за большего основания системы счисления сумматора (256 вместо 4) **одноразрядный двухсотпятидесятишестиричный несимметричный полный сумматор в 256/4=64 раза быстрее и четверичных одноединичных (4-Bit BinaryCoded Quadro UnoUnary, 4B BCQ UU) квадросумматоров команды из МИФИ под руководством Хетагурова.**

Так как в двухсотпятидесятишестиричном несимметричном полном сумматоре на считывание одного значения из одного массива в ОЗУ тратится время равное 2*dt, а считать нужно два значения (сумма по модулю 256 и бит переноса), то на считывание двух значений тратится время равное 2*2*dt=4*dt, где dt – время

задержки в одном типовом логичесом элементе, что в два раза менше, чем в двухсотпятидесятишестиричном (8-мибитном) сумматоре 8-ми Когге-Стоуна, в котором на сложение двух двусотпятидесятишестиритов тратится время равное 8*dt, т. е. табличный двухсотпятидесятишестиричный несимметричный полный сумматор в 8*dt/4*dt=2 раза быстрее 8-мибитного сумматора Когге-Стоуна.

Литература:
1. Сумматор Когге-Стоуна, 8-ми битный. Куликов А.С.

Андрей Куликов, Россия-Русь, Москва, Царицыно, версия 2019.10.14.