Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»

Институт Радиотехники и электроники им. В.А. Котельникова

Кафедра Электроники и наноэлектроники

КОНТРОЛЬНОЕ МЕРОПРИЯТИЕ № 1

по дисциплине Синтез цифровых интегральных схем

Тема: Сумматоры. Временной анализ

Студент
гр. ЭР-05м-24 _____ Волчков Д.Н.
(подпись)

Преподаватель,
доцент _____ Баринов А.Д.
(оценка/зачёт, дата, подпись)

Москва

2024

Оглавление

1. Полный сумматор 3
2. Последовательный сумматор на 8 бит 3
3. Сумматор с параллельно-последовательным переносом 4
4. Параметризируемый последовательный сумматор 5
5. Создание регистров. Временной анализ 5
Приложение 1 9
Приложение 2 10
Приложение 3 11
Приложение 4 12
Приложение 5 13
Приложение 6 14
Приложение 7 15
Приложение 8 16
Приложение 9 17
Приложение 10 18
Приложение 11 19
Приложение 12 20
Приложение 13 21
Приложение 14 22

1. Полный сумматор

Для начала создадим полный сумматор для сложения одного бита данных. Он будет принимать три сигнала: A, B – сигналы чисел; C_in – сигнал переноса с предыдущего бита данных. Сумматор будет передавать два сигнала на выход: S – сигнал сложения чисел; C_out – сигнал переноса на следующий бит (разряд). Листинг сумматора приведен в приложении 1. RTL-представление изображено на рисунке 1:

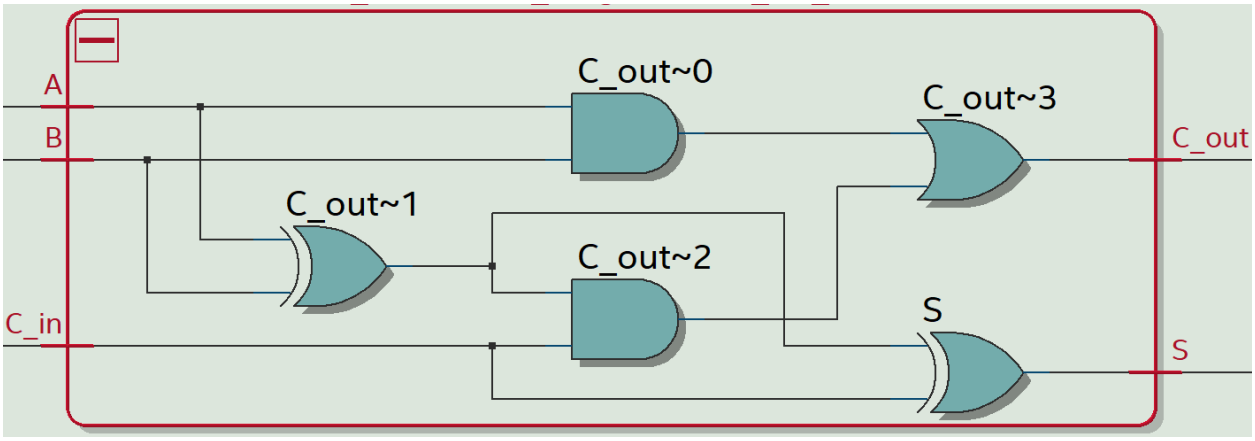


Рисунок 1 – RTL-представление полного сумматора

2. Последовательный сумматор на 8 бит

Сделаем последовательный сумматор на 8 бит. Для этого воспользуемся полным сумматором, объединим их последовательно. Последовательный сумматор обладает теми-же сигналами на входе и выходе за исключением того, что сигналы A, B и S принимают сигнал в 8 бит. Листинг сумматора приведен в приложении 2. RTL-представление изображено на рисунке 2:

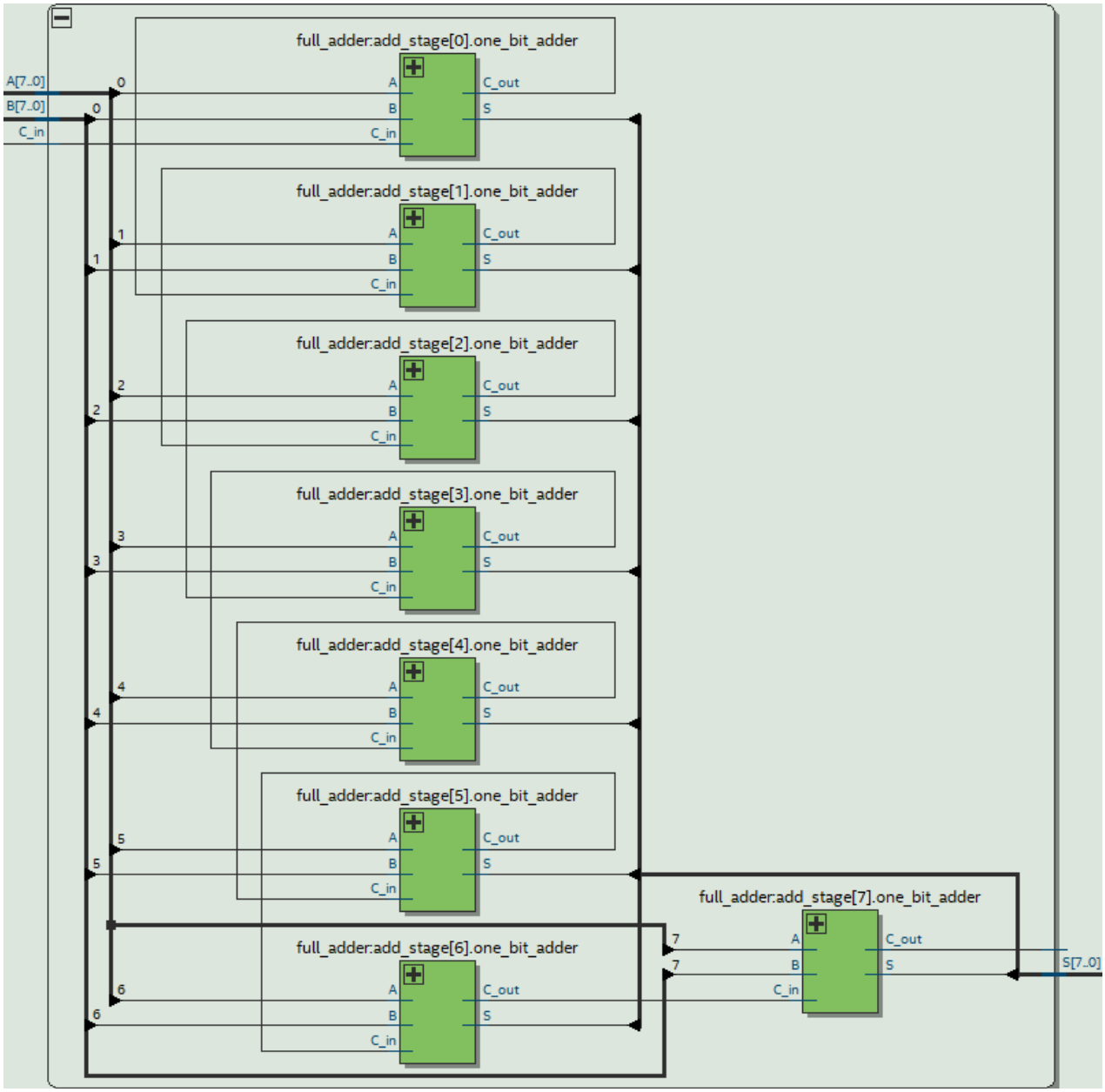


Рисунок 2 – RTL-представление последовательного сумматора на 8 бит

3. Сумматор с параллельно-последовательным переносом

Теперь напишем описание для сумматора с параллельно-последовательным переносом, для этого воспользуемся сумматором на 8 бит. Он обладает все теми же входными и выходными сигналами, за исключением того что их разрядность напрямую зависит от разрядности данного параметризируемого сумматора. Листинг сумматора представлен в приложении 3. В приложении 4 изображен сумматор с параллельно-последовательным переносом.

4. Параметризируемый последовательный сумматор

Для сравнения параллельно-последовательного сумматора с последовательным создадим таковой сумматор. Его входные и выходные сигналы ничем не отличаются от других сумматоров. Аналогично с параллельно-последовательным сумматором разрядность сигналов A, B, S зависит от разрядности сумматора, передаваемой в виде параметра. Его описание представлено в приложении 5. RTL-представление представлено в приложении 6.

5. Создание регистров. Временной анализ

Для того, чтобы сравнить работу двух сумматоров, необходимо будет превратить комбинационную схему в последовательную, для этого облачим наши входные и выходные сигналы в регистры, а также сделаем файл временных ограничений для временного анализа. Листинги регистров и файл временных - ограничений представлены в приложениях 7-9. RTL-представления регистров представлены на рисунках 3 и 4:

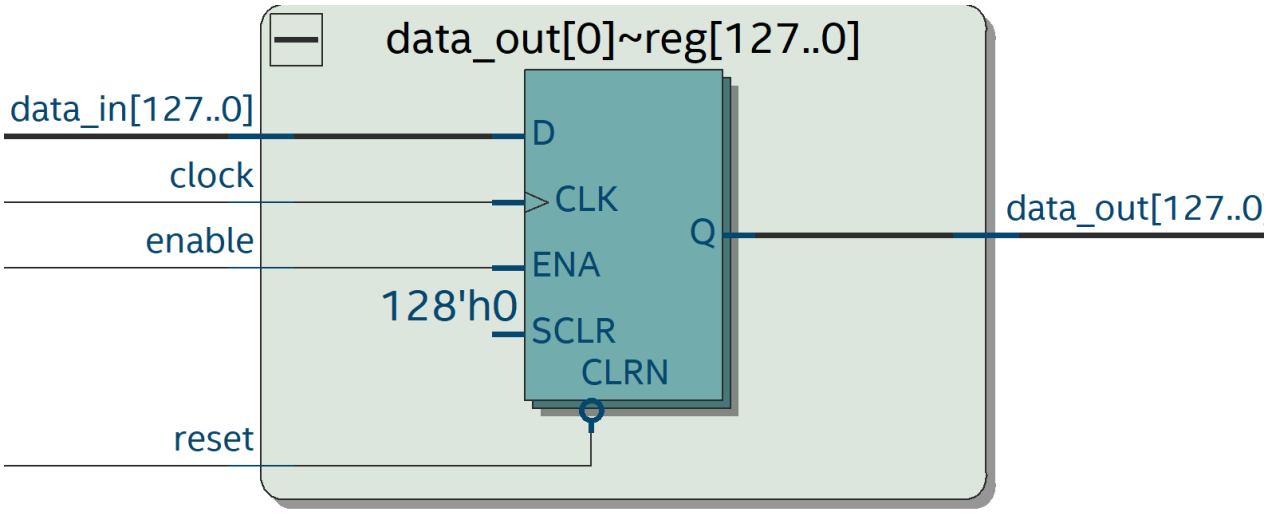


Рисунок 3 – RTL-представление входного регистра

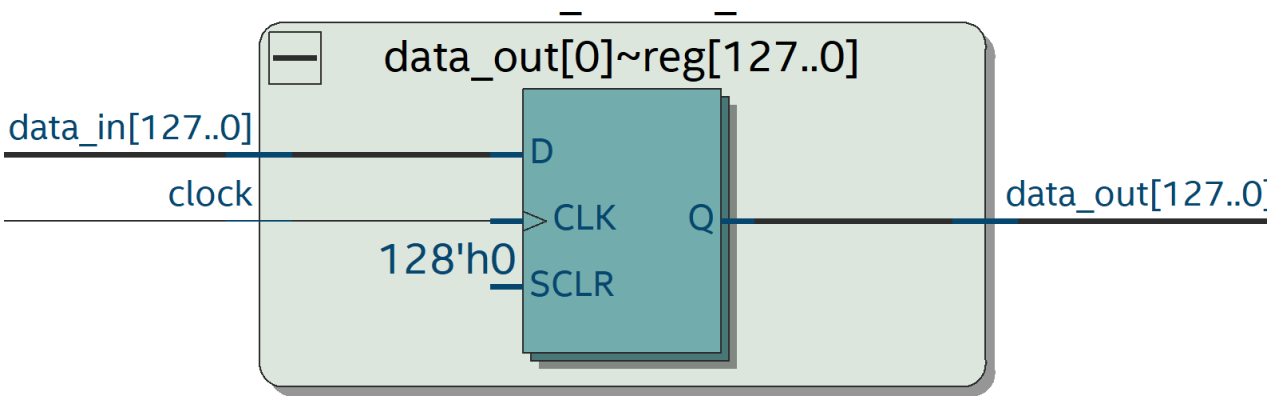


Рисунок 4 – RTL-преставление выходного регистра

Теперь окружим наши входные и выходные регистры для описаний сумматоров. Листинги приведены в приложениях 10 и 11. RTL-описания приведены в приложениях 12 и 13.

Можно проводить исследования, для этого будем изменять параметр разрядности сумматоров и смотреть как изменения разрядов повлияют на изменения максимальной рабочей частоты и количества логических элементов. Результаты представлены в таблицах 1 и 2 и на рисунках 5 и 6:

Таблица 1. Данные для параллельно-последовательного сумматора

Разрядность	Количество логических элементов	Частота, МГц
8	29	121,12
16	53	88,4
32	101	86,03
64	203	72,07
128	409	64,04

Таблица 2. Данные для последовательного сумматора

Разрядность	Количество логических элементов	Частота, МГц
8	31	152,28
16	53	79.37
32	113	61,69
64	211	41,46
128	436	21,61

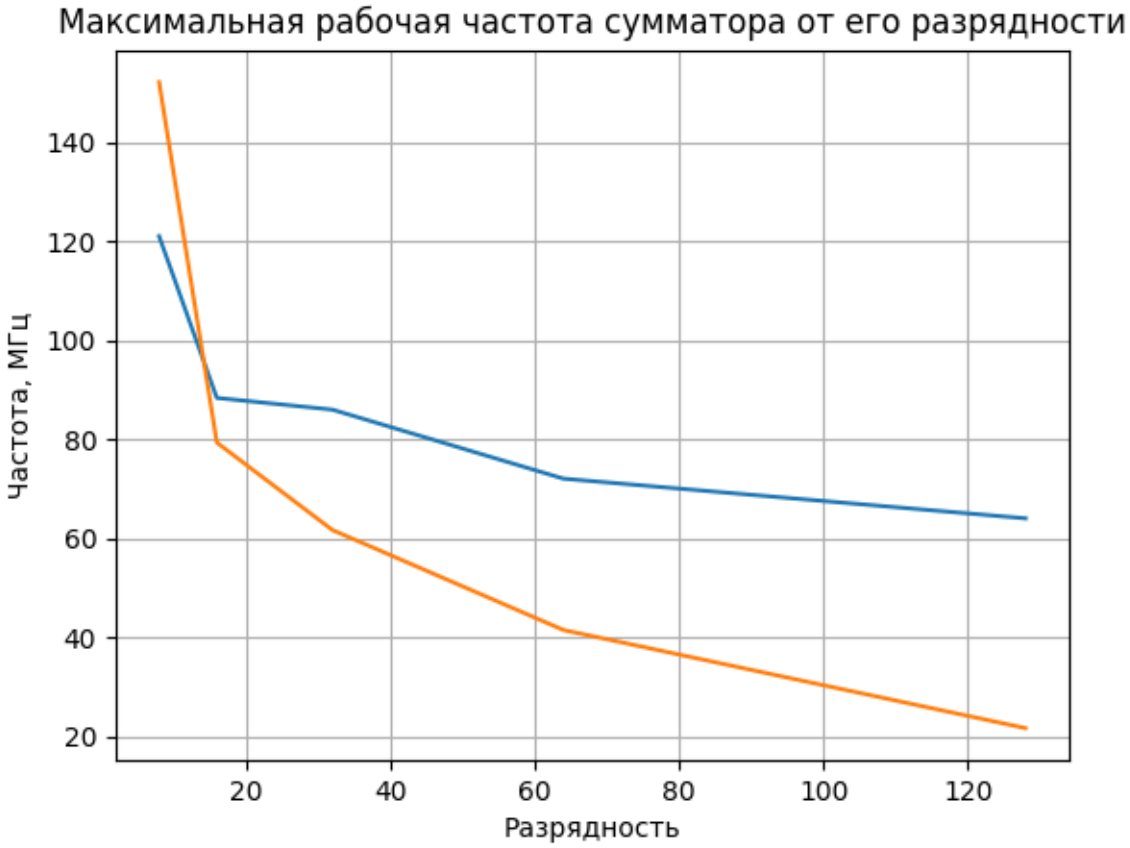


Рисунок 1 – Максимальная рабочая частота сумматора от его разрядности

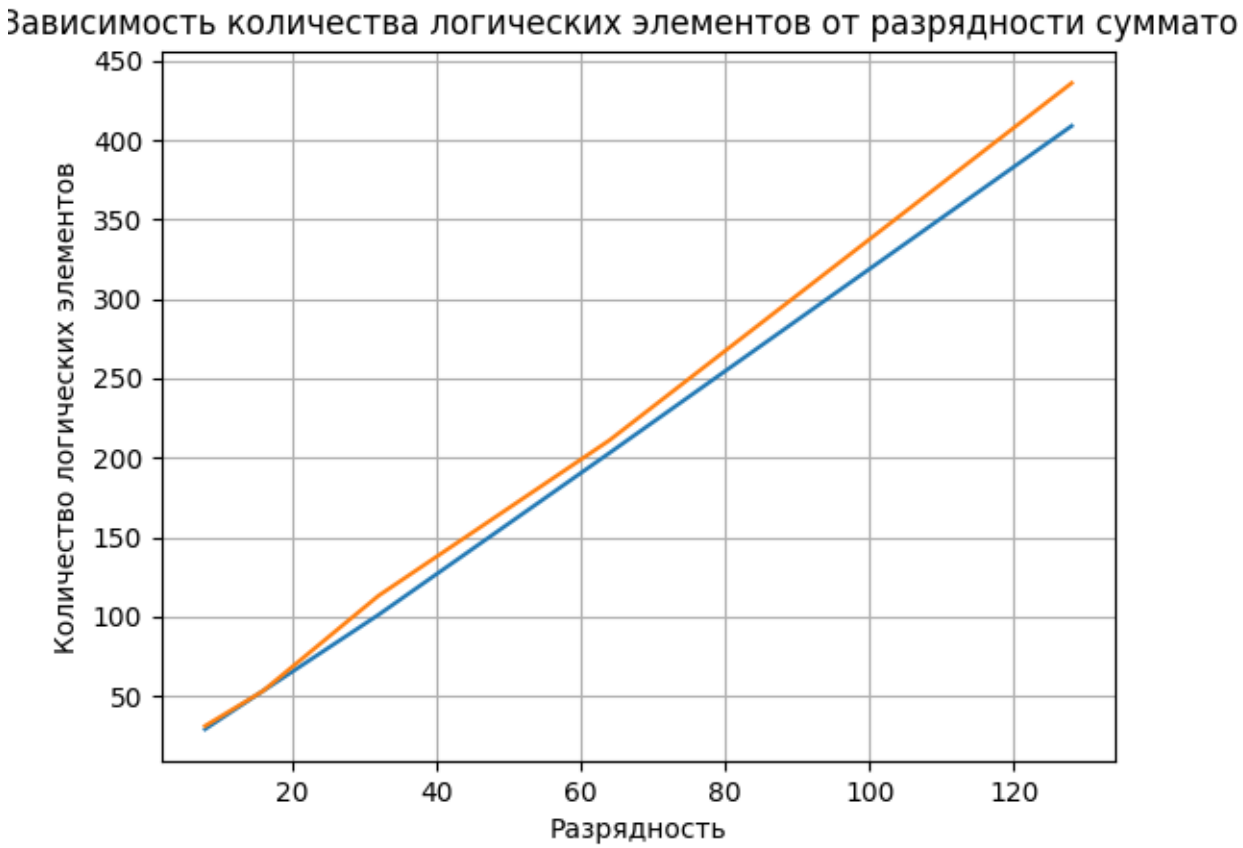


Рисунок 2 – Зависимость количества логических элементов от разрядности сумматора

Приложение 1

Листинг полного сумматора

```
module full_adder (A, B, C_in, C_out, S);
input logic A, B, C_in;
output logic S, C_out;
assign S = A ^ B ^ C_in;
assign C_out = (A & B) | (C_in & (A ^ B));
endmodule
```

Приложение 2

Листинг последовательного сумматора на 8 бит

```
module bit_8_adder (A, B, C_in, C_out, S);

input logic [7:0] A, B;
input logic C_in;
output logic [7:0] S;
output logic C_out;
logic [8:0] C;
genvar i;

generate
	for (i = 0; i < 8; i++) begin :
		add_stage full_adder one_bit_adder(.A(A[i]), .B(B[i]), .C_in(C[i]), .C_out(C[i+1]), .S(S[i]));
	end
endgenerate

assign C[0] = C_in;
assign C_out = C[8];

endmodule
```

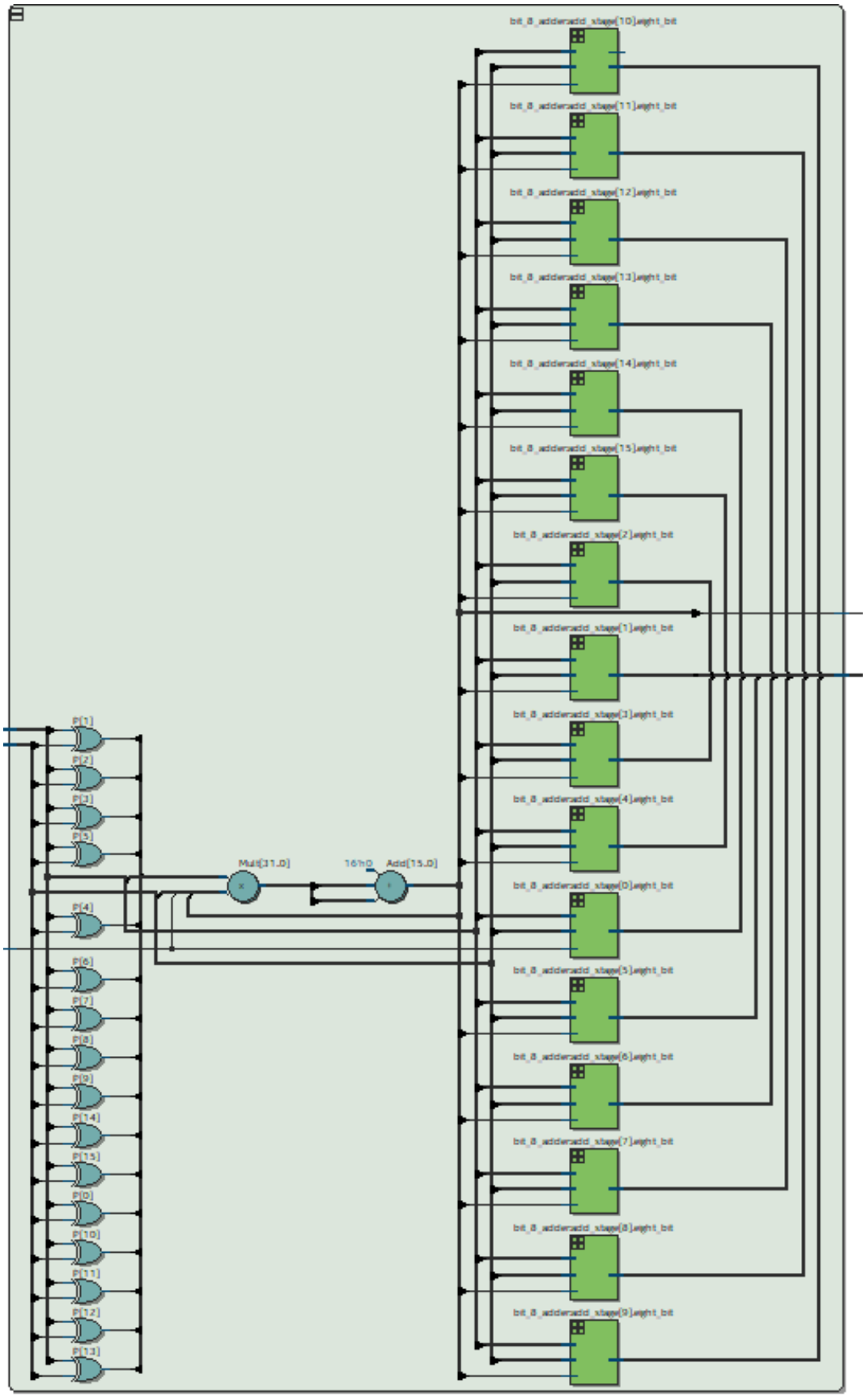
Приложение 3

Листинг параллельно-последовательного сумматора

```
module par_posl_adder_param #(parameter W = 128) (A, B, C_in, C_out, S);

input logic [W-1:0] A, B;
input logic C_in;
output logic [W-1:0] S;
output logic C_out;
logic [W/8:0] C,G,P;
genvar i;

generate
	for (i = 0; i < W/8; i++) begin:
		add_stage bit_8_adder eight_bit (.A(A[(i+1)*8-1:i*8]), .B(B[(i+1)*8-1:i*8]), .C_in(C[i]), .C_out(), .S(S[(i+1)*8-1:i*8]));
		if (i > 0) assign C[i] = G[i-1] + P[i-1]*C[i-1];
		assign P[i] = A[(i+1)*8-1] ^ B[(i+1)*8-1];
		assign G[i] = A[(i+1)*8-1] * B[(i+1)*8-1];
	end
endgenerate

assign C[W/8] = G[W/8-1] + P[W/8-1]*C[W/8-1];
assign C[0] = C_in;
assign C_out = C[W/8];

endmodule
```

Приложение 4

RTL-представление параллельно-последовательного сумматора



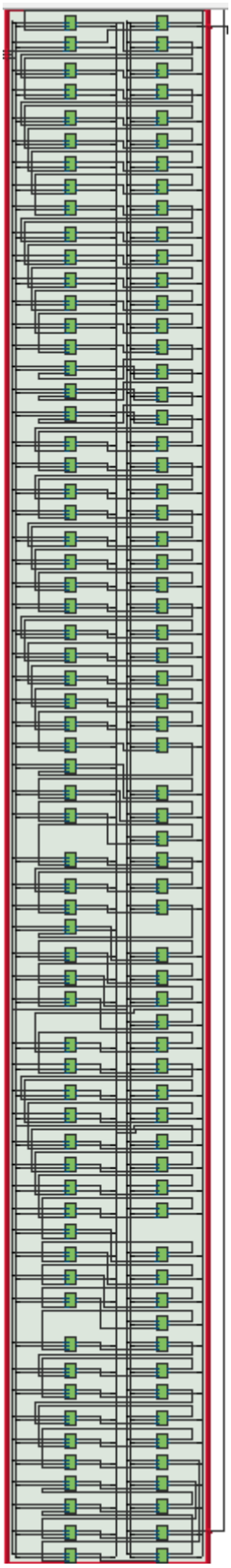
Приложение 5

Листинг последовательного сумматора

```
module posl_adder_param #(parameter W = 128) (A, B, C_in, C_out, S);

input logic [W-1:0] A, B;
input logic C_in;
output logic [W-1:0] S;
output logic C_out;
logic [W:0] C;
genvar i;

generate
	for (i = 0; i < W; i++) begin :
		add_stage full_adder one_bit_adder(.A(A[i]), .B(B[i]), .C_in(C[i]),
.C_out(C[i+1]), .S(S[i]));
	end
endgenerate

assign C[0] = C_in;
assign C_out = C[W];

endmodule
```

Приложение 6

RTL-представление последовательного сумматора



Приложение 7

Листинг входного регистра

```
module register #(parameter W = 8)
(input logic clock, reset, enable,
input logic [W-1:0] data_in,
output logic [W-1:0] data_out);
always_ff @(posedge clock or negedge reset)
if(!reset)
data_out <= {W{1'b0}};
else if(enable)
data_out <= data_in;
endmodule
```

Приложение 8

Листинг выходного регистра

```
module unit_latch #(parameter W = 1)
(input logic clock,
input logic [W-1:0] data_in,
output logic [W-1:0] data_out);
always_ff @(posedge clock)
data_out <= data_in;
endmodule
```

Приложение 9

Листинг файла временных ограничений

```
#объявляем, что порт CLK_50 является
#генератором синхросигнала с частотой 50 МГц
create_clock -period 50MHz -name {CLK_50} [get_ports {CLK_50}]
#генерируем все неопределённости, связанные
#с заданными синхросигналами
derive_clock_uncertainty
#задаём все ложные критические пути, которые можно
#исключить из временного анализа
set_false_path -from [get_clocks {CLK_50}] -to [get_ports {S}]
```

Приложение 10

Листинг верхнего уровня параллельно-последовательного сумматора

```
module par_posl_top #(parameter W = 128) (A, B, C_in, C_out, S, CLK_50, reset,
enable);

input logic [W-1:0] A, B;
input logic C_in;
output logic [W-1:0] S;
output logic C_out;
logic [W-1:0] Q_A, Q_B, Q_S;
input logic CLK_50, reset, enable;
logic Q_C_in, Q_C_out;

par_posl_adder_param #(.W(W)) DUT (.A(Q_A), .B(Q_B), .C_in(Q_C_in),
.C_out(Q_C_out), .S(Q_S));

register #(.W(W)) A_i (.clock(CLK_50), .reset(reset), .enable(enable), .data_in(A),
.data_out(Q_A));
register #(.W(W)) B_i (.clock(CLK_50), .reset(reset), .enable(enable), .data_in(B),
.data_out(Q_B));
register #(.W(1)) C_i (.clock(CLK_50), .reset(reset), .enable(enable), .data_in(C_in),
.data_out(Q_C_in));

unit_latch #(.W(W)) S_o (.clock(CLK_50), .data_in(Q_S), .data_out(S));
unit_latch #(.W(1)) C_o (.clock(CLK_50), .data_in(Q_C_out), .data_out(C_out));

endmodule
```

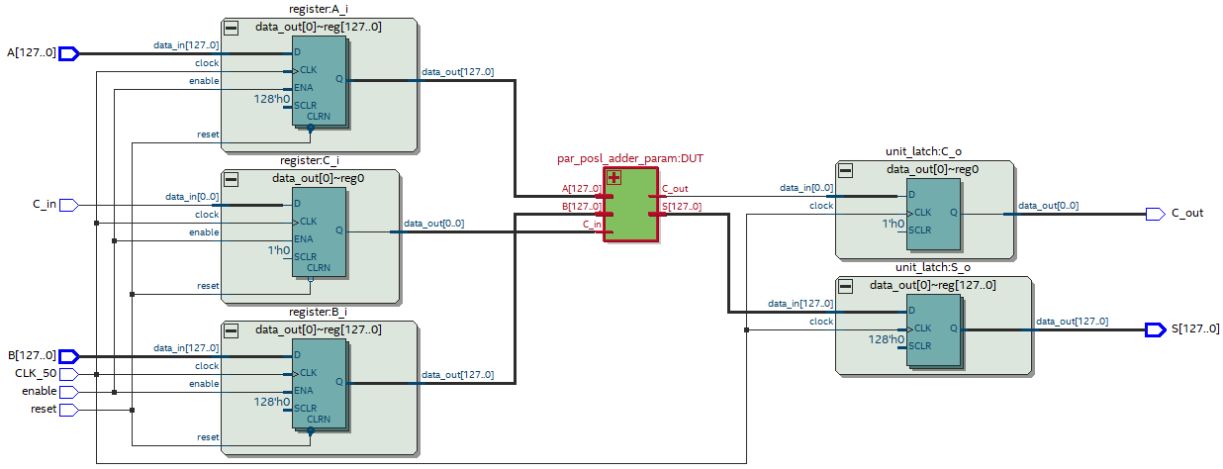
Приложение 11

Листинг верхнего уровня для последовательного сумматора

```
module posl_top #(parameter W = 128) (A, B, C_in, C_out, S, CLK_50, reset,
enable);

input logic [W-1:0] A, B;
input logic C_in;
output logic [W-1:0] S;
output logic C_out;
logic [W-1:0] Q_A, Q_B, Q_S;
input logic CLK_50, reset, enable;
logic Q_C_in, Q_C_out;

posl_adder_param #(.W(W)) DUT (.A(Q_A), .B(Q_B), .C_in(Q_C_in),
.C_out(Q_C_out), .S(Q_S));

register #(.W(W)) A_i (.clock(CLK_50), .reset(reset), .enable(enable), .data_in(A),
.data_out(Q_A));
register #(.W(W)) B_i (.clock(CLK_50), .reset(reset), .enable(enable), .data_in(B),
.data_out(Q_B));
register #(.W(1)) C_i (.clock(CLK_50), .reset(reset), .enable(enable), .data_in(C_in),
.data_out(Q_C_in));

unit_latch #(.W(W)) S_o (.clock(CLK_50), .data_in(Q_S), .data_out(S));
unit_latch #(.W(1)) C_o (.clock(CLK_50), .data_in(Q_C_out), .data_out(C_out));

endmodule
```

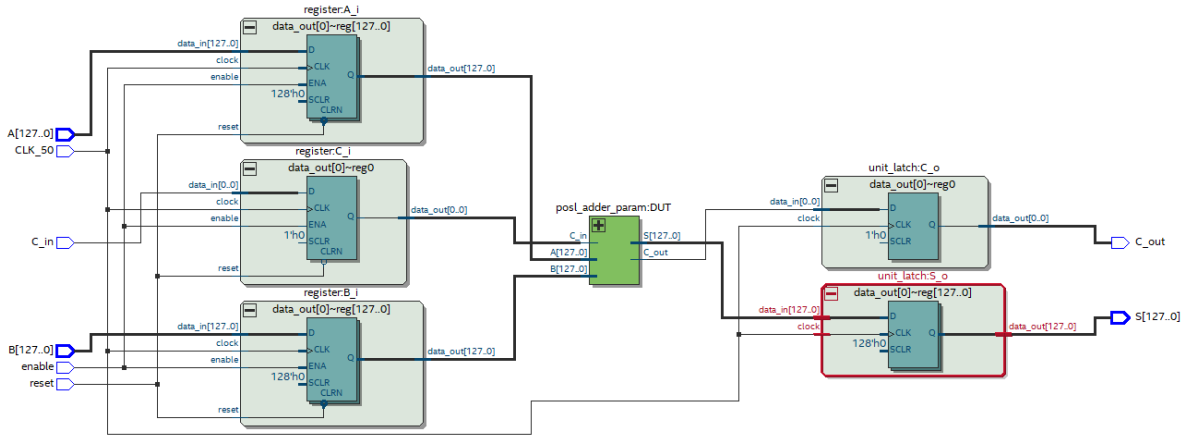
Приложение 12

RTL-представление верхнего уровня параллельно-последовательного сумматора



Приложение 13

RTL-представление верхнего уровня для последовательного сумматора



Приложение 14

Листинг кода для создания графиков

```
import numpy as np
import matplotlib.pyplot as plt

plt.grid(True)
plt.xlabel('Разрядность')
plt.ylabel('Частота, МГц')
plt.title('Максимальная рабочая частота сумматора от его разрядности')
plt.plot([8,16,32,64,128],[121.12,88.4,86.03,72.07,64.04])
plt.plot([8,16,32,64,128],[152.28,79.37,61.69,41.46,21.61])
plt.show()

plt.grid(True)
plt.xlabel('Разрядность')
plt.ylabel('Количество логических элементов')
plt.title('Зависимость количества логических элементов от разрядности
сумматора')
plt.plot([8,16,32,64,128],[29,53,101,203,409])
plt.plot([8,16,32,64,128],[31,53,113,211,436])
plt.show()
```