



Международная научно-практическая
конференция
КОМПЬЮТЕРНОЕ
ПРОЕКТИРОВАНИЕ В
ЭЛЕКТРОНИКЕ

EDA
Conference

**АЛГОРИТМ ВНЕДРЕНИЯ ВОДЯНЫХ ЗНАКОВ
В VHDL-ОПИСАНИЯ ЦИФРОВЫХ УСТРОЙСТВ,
РЕАЛИЗУЕМЫХ НА FPGA**

Лавренова А.С.
Иванюк А.А.

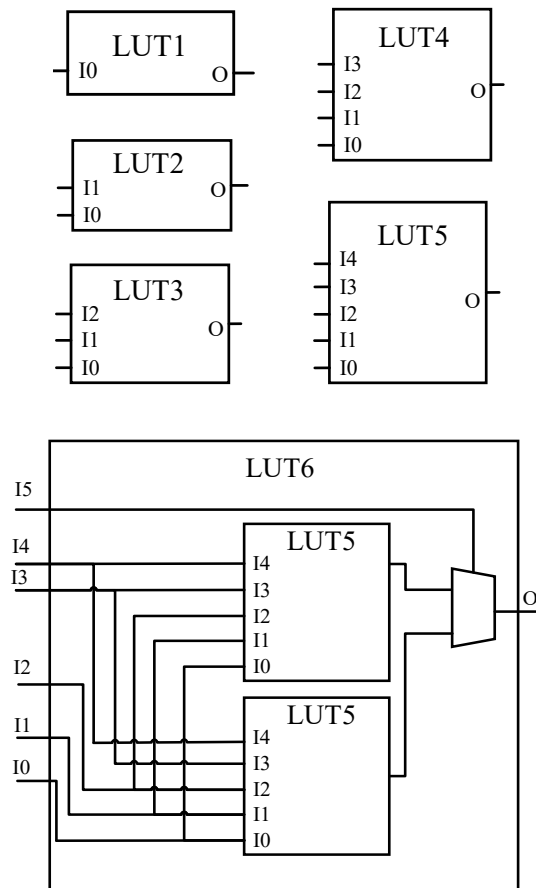
Минск 2026

Цель работы:

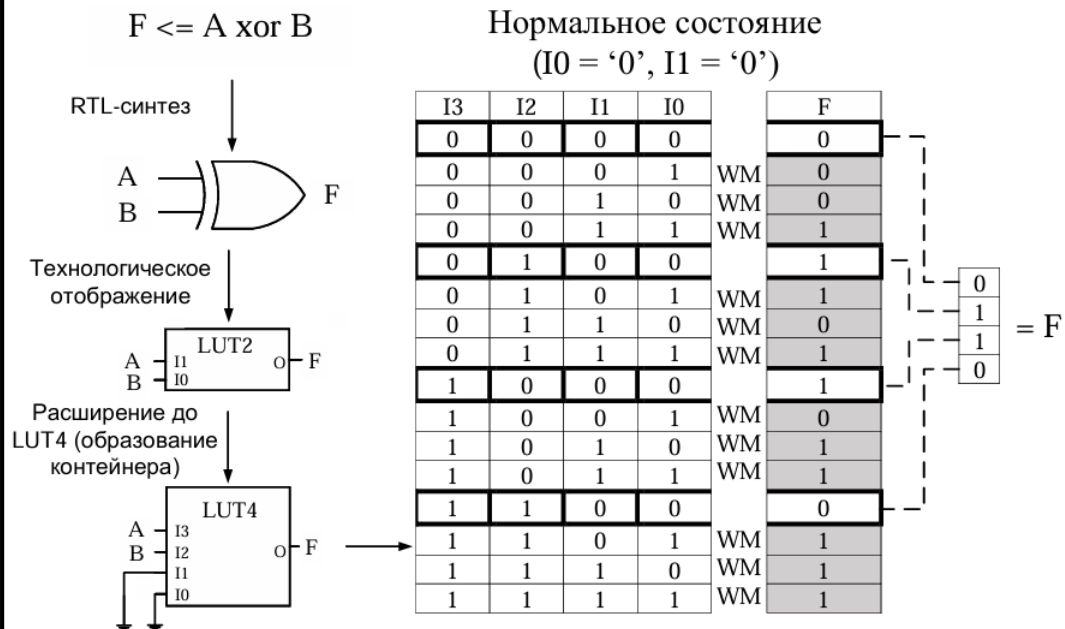
- Рассмотрение методов внедрения аппаратных водяных знаков в VHDL-описания цифровых устройств, реализуемых на FPGA.
- Анализ структуры LUT-блоков и определение возможностей использования свободных ячеек параметра INIT в качестве контейнера для водяного знака.
- Разработка алгоритма внедрения и извлечения водяных знаков с использованием псевдослучайного распределения битов на основе LFSR.
- Исследование влияния синтеза и оптимизации САПР на сохранность внедрённой информации.
- Экспериментальная оценка устойчивости водяных знаков с применением различных способов защиты от оптимизации САПР.
- Разработка программного средства для автоматизации анализа VHDL-описаний, внедрения и извлечения водяных знаков.

Конфигурации LUT-блоков FPGA и наличие в них свободных ячеек

Типовые конфигурации
LUT-блоков (от 1 до 6 входов)



Формирование свободных ячеек в памяти
конфигурации LUT-блоков



Пример для FPGA Xilinx Spartan-3: технологический LUT4, в котором при синтезе переключательных функций также остаются свободные ячейки.

Анализ произвольно выбранных проектов

1) ШИМ генератор сигналов и звука

https://github.com/DominikCaban/de1_project_sinus/tree/main

2) Блок вывода информации калькулятора

<https://github.com/lbrushman1/ENEL-373-FPGA-Calculator>

3) Простой 16-битный процессор

https://github.com/mamadouxxx/hardware_design/tree/main/tp2_ms3

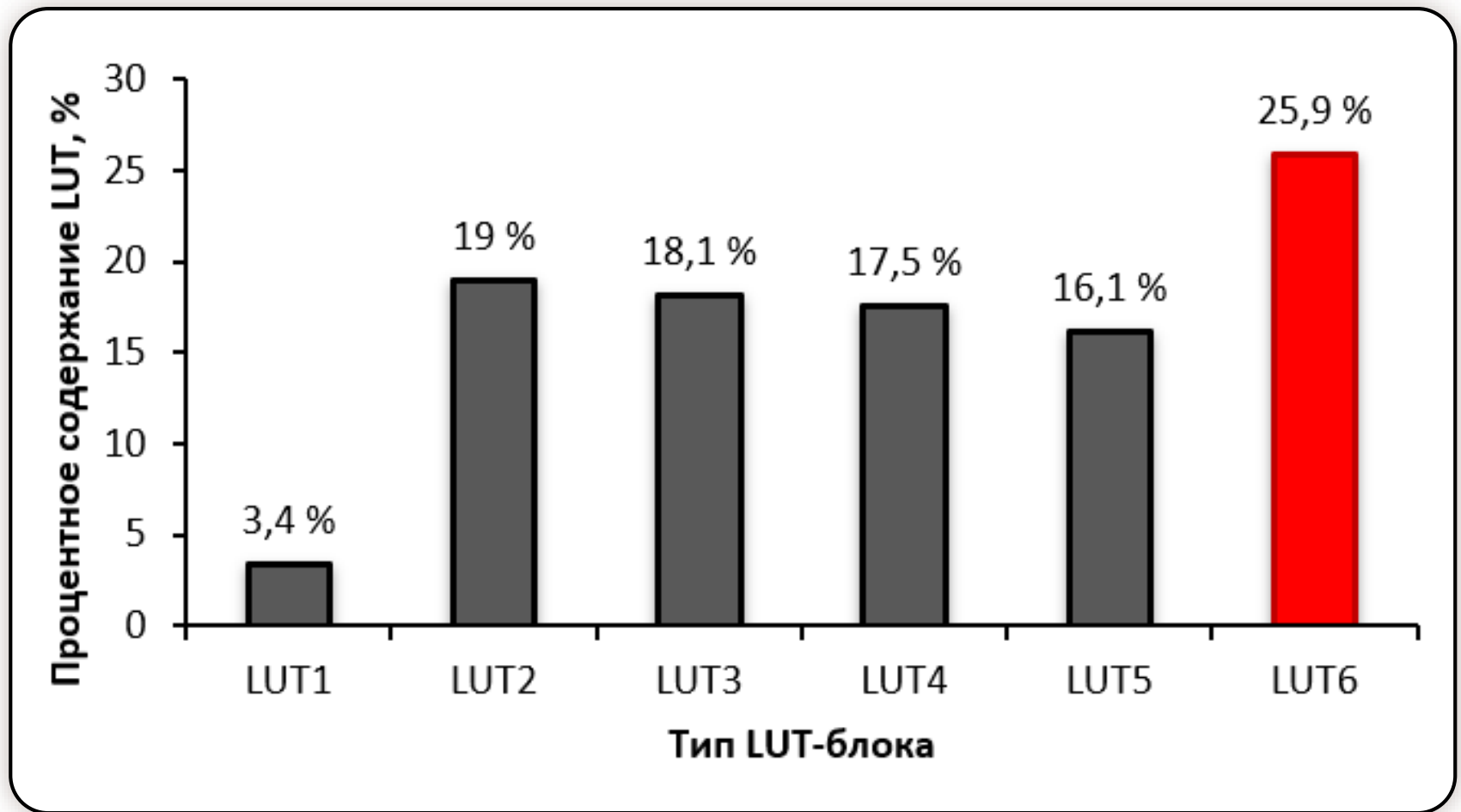
4) Вычитающий счётчик

<https://github.com/panaiteelena/8-bytes-Counter---Register-system>

5) Устройство вывода четырёх шестнадцатеричных цифр на индикатор

https://github.com/pblackcookie/FPGA_Count_Down/tree/main/assignment1_main2

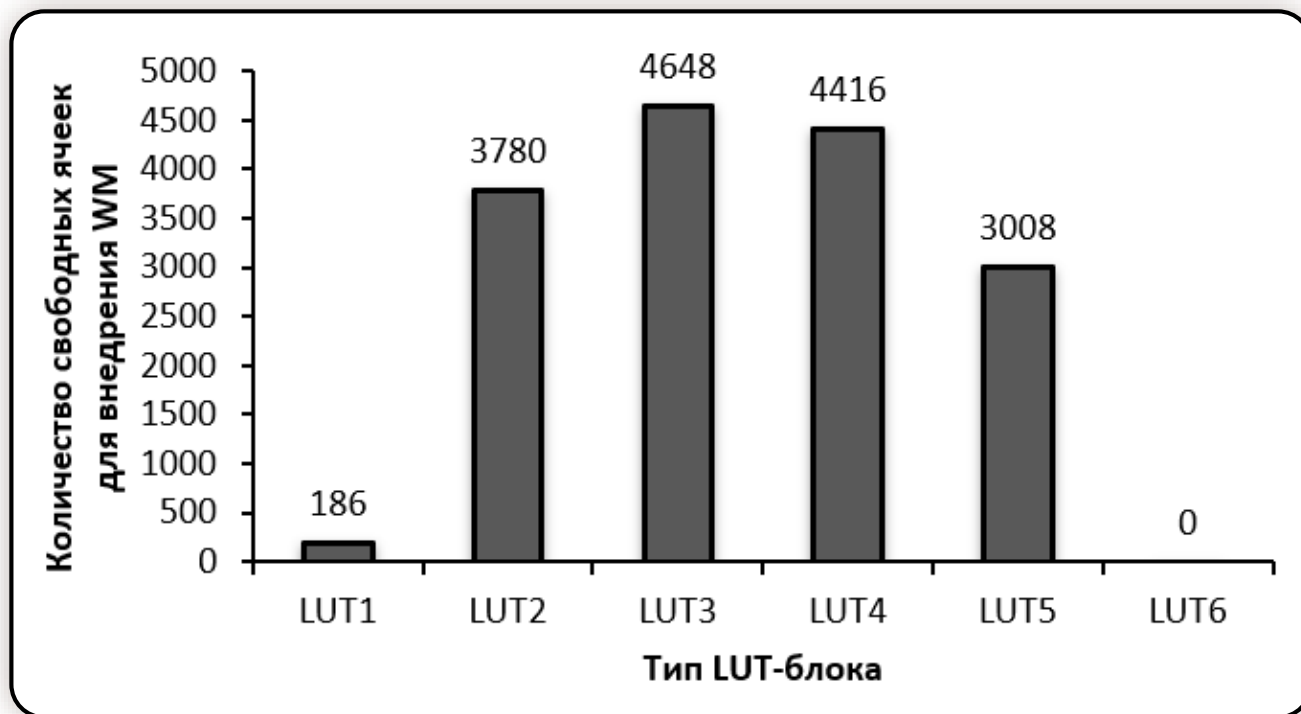
Удельный процент использования LUT-блоков для различных произвольно выбранных проектов (Vivado 2021.1)



Размер контейнера для внедрения водяного знака в произвольно выбранном проекте (Vivado 2021.1)

Простой 16-битный процессор

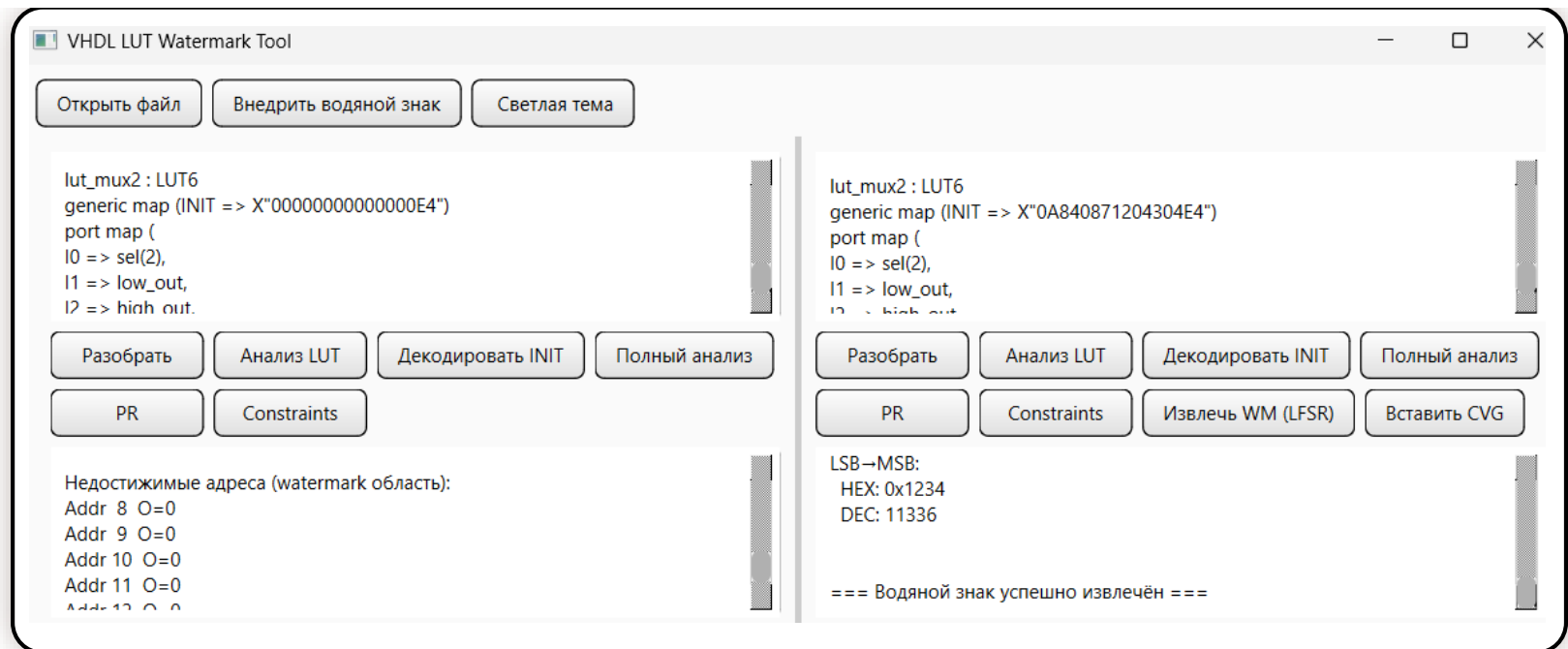
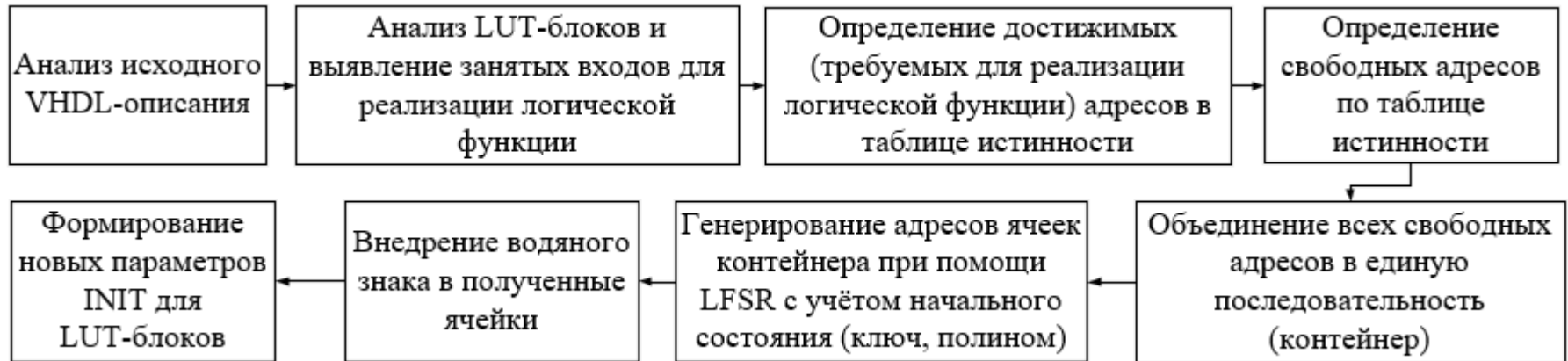
https://github.com/mamadouxxx/hardware_design/tree/main/tp2_ms3



Кристалл: Xilinx Artix-7 XC7A35T-1CPG236C (xc7a35tcpg236-1), 2 600 CLB, 5 200 Slices, 20 800 LUT.

В проекте : 519 примитивов LUT1–LUT6; суммарная свободная ёмкость памяти конфигураций LUT-блоков 16 038 бит (LUT6: 0 бит).

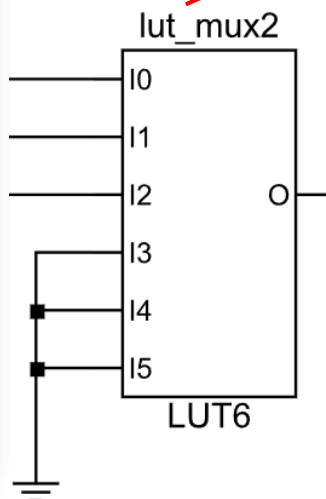
Алгоритм внедрения водяного знака с использованием LFSR



Анализ VHDL-описания и LUT-блоков

Тип	Значение	Позиция
LIBRARY	library	(1:1)
BASIC_IDENTIFIER	ieee	(1:9)
;	;	(1:13)
USE	use	(2:1)
BASIC_IDENTIFIER	ieee	(2:5)
.	.	(2:9)
BASIC_IDENTIFIER	std_logic_1164	(2:10)
.	.	(2:24)
ALL	all	(2:25)
;	;	(2:28)
LIBRARY	library	(4:1)
BASIC_IDENTIFIER	UNISIM	(4:9)
;	;	(4:15)
USE	use	(5:1)
BASIC_IDENTIFIER	UNISIM	(5:5)
.	.	(5:11)
BASIC_IDENTIFIER	VComponents	(5:12)
.	.	(5:23)
ALL	all	(5:24)

Метка	Тип	Входы	Ёмкость	Используется	Занятость, %	Свободно	Строка
lut_low	LUT6	6	64	64	100.0	0	стр. 21
lut_high	LUT6	6	64	64	100.0	0	стр. 33
lut_mux2	LUT6	6	64	8	12.5	56	стр. 45



=== Анализ с учётом зафиксированных входов ===

Зафиксированные входы:

I3 = 0

I4 = 0

I5 = 0

Используемые адреса (реальная логика):

Addr 0 O=0

Addr 1 O=0

Addr 2 O=1

Addr 3 O=0

Addr 4 O=0

Addr 5 O=1

Addr 6 O=1

Addr 7 O=1

Недостижимые адреса (watermark область):

Addr 8 O=0

Addr 9 O=0

Addr 10 O=0

...

Addr 63 O=0

Изменение параметра INIT после внедрения водяного знака

INIT-параметр до внедрения

```
lut_mux2 : LUT6
generic map (INIT => X"0000000000000000E4")
port map (
  I0 => sel(2),
  I1 => low_out,
  I2 => high_out,
  I3 => '0',
  I4 => '0',
  I5 => '0',
  O => y
);
```

WM
отсутствует

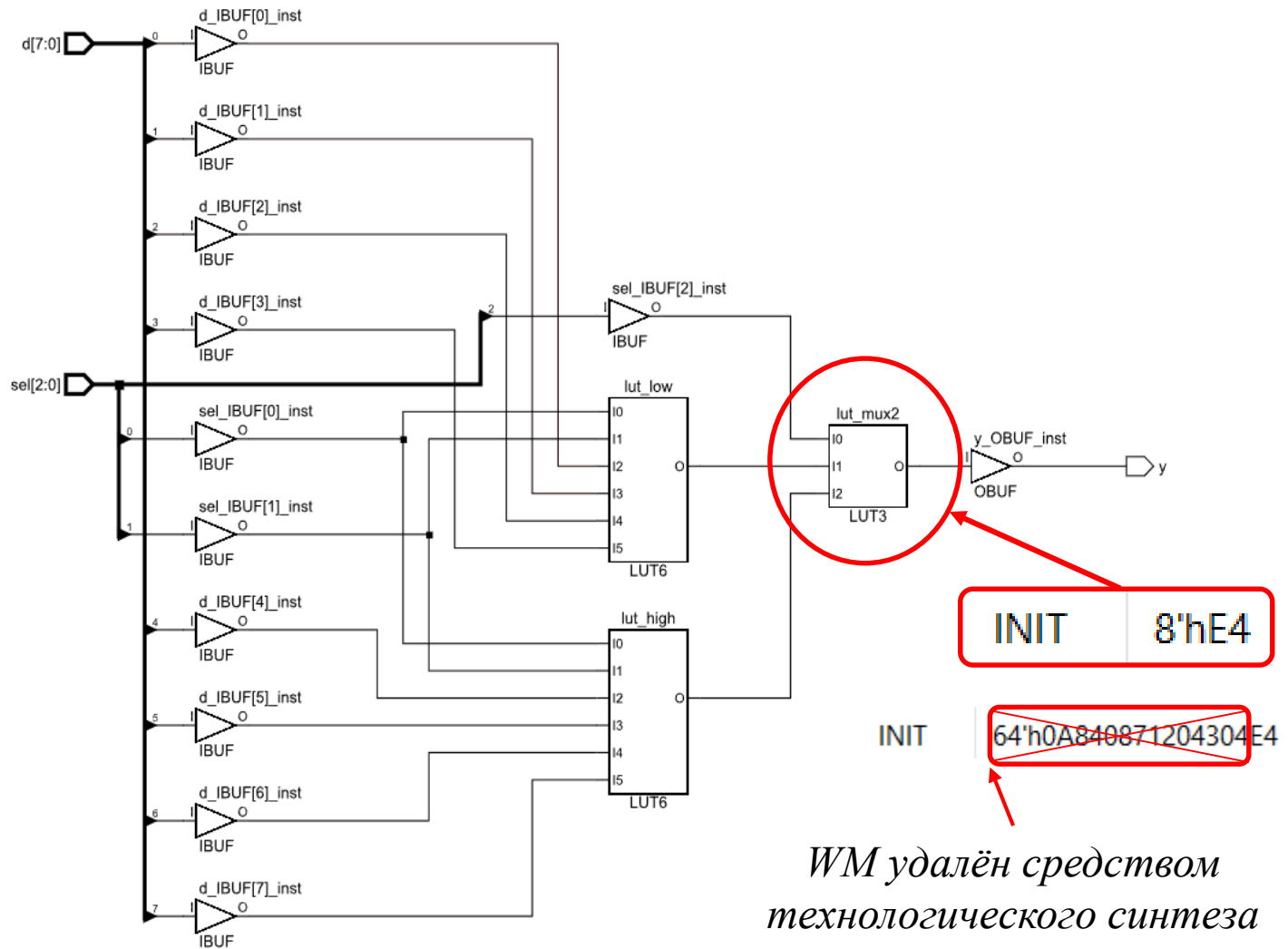
INIT-параметр после внедрения

```
lut_mux2 : LUT6
generic map (INIT => X"0A840871204304E4")
port map (
  I0 => sel(2),
  I1 => low_out,
  I2 => high_out,
  I3 => '0',
  I4 => '0',
  I5 => '0',
  O => y
);
```

WM внедрён

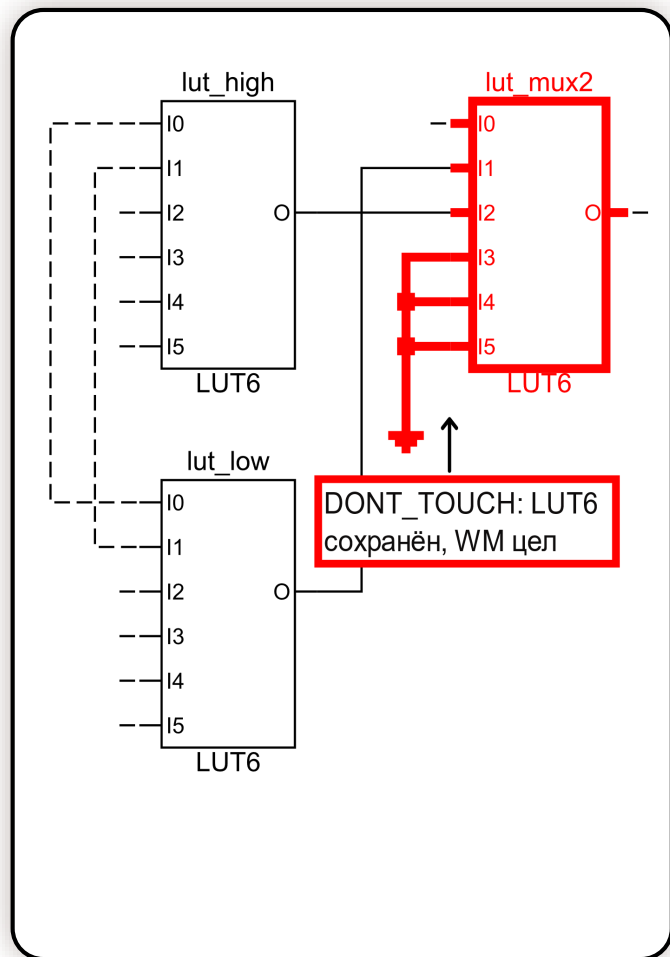
Реализуемая функция сохраняется после внедрения водяного знака

Результат технологического синтеза на ПЛИС



Методы защиты от оптимизации

Атрибут DONT_TOUCH



```
attribute DONT_TOUCH : string;
```

```
architecture structural of mux8_lut is
```

```
signal low_out, high_out : std_logic;
```

```
attribute DONT_TOUCH of lut_mux2 : label is "TRUE";
```

```
begin
```

```
lut_mux2 : LUT6
```

```
generic map (INIT => X"0A840871204304E4")
```

```
port map (
```

```
  I0 => sel(2),
```

```
  I1 => low_out,
```

```
  I2 => high_out,
```

```
  I3 => '0',
```

```
  I4 => '0',
```

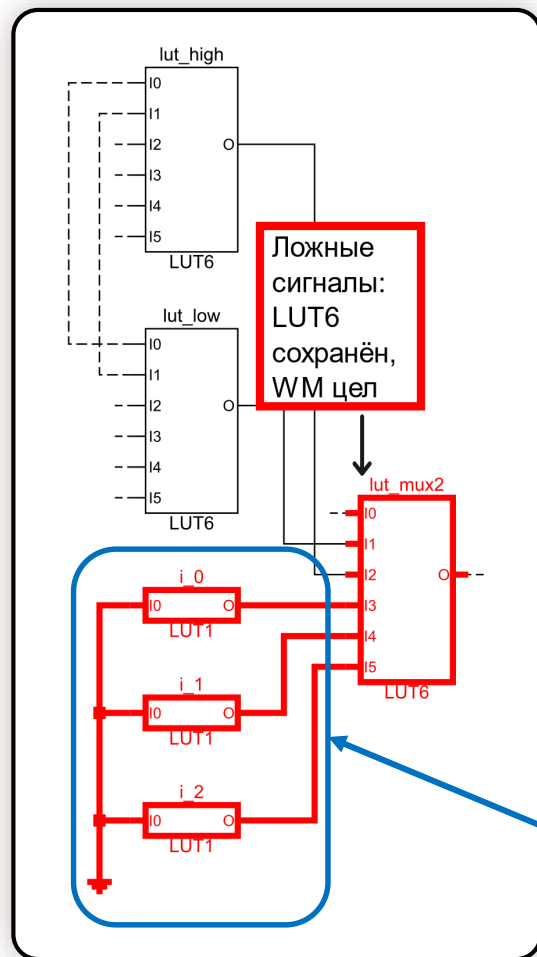
```
  I5 => '0',
```

```
  O => y
```

```
);
```

Методы защиты от оптимизации

Ложные сигналы



```
attribute DONT_TOUCH : string;
```

```
signal dep1 : std_logic;  
signal dep2 : std_logic;  
signal dep3 : std_logic;
```

*Требуется
атрибут
DONT_TOUCH*

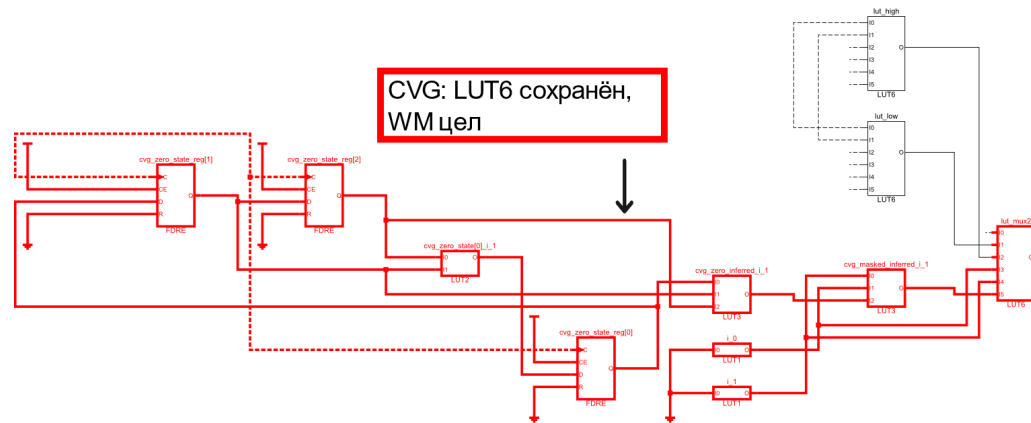
```
attribute DONT_TOUCH of dep1 : signal is "TRUE";  
attribute DONT_TOUCH of dep2 : signal is "TRUE";  
attribute DONT_TOUCH of dep3 : signal is "TRUE";
```

```
signal low_out, high_out : std_logic;  
begin
```

```
dep1 <= d(0) xor d(0);  
dep2 <= d(1) xor d(1);  
dep3 <= d(2) xor d(2);
```

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Генератор CVG (Constant Value Generator)



*Отсутствует
атрибут
DONT_TOUCH*

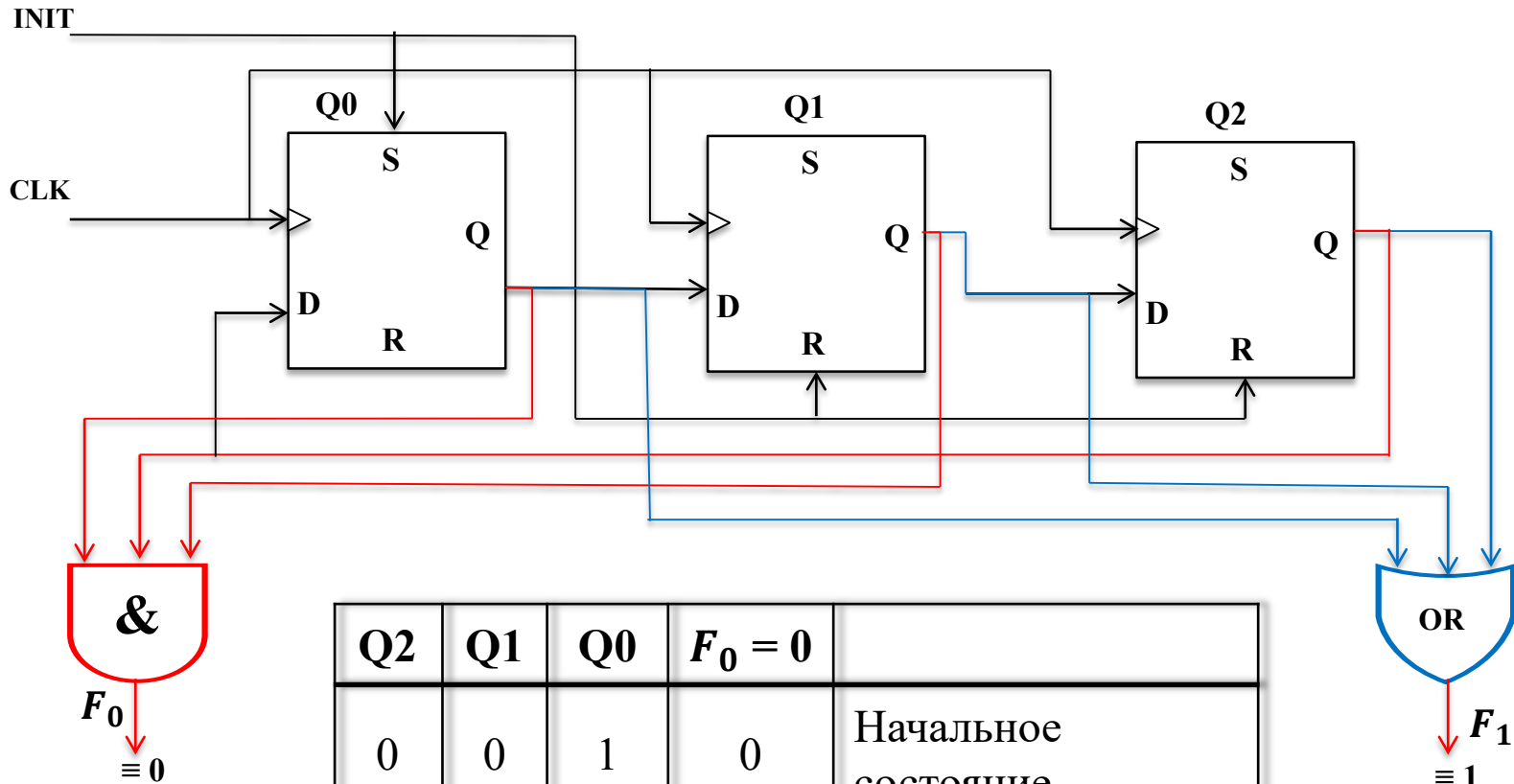
```

signal cvg_state : std_logic_vector(2 downto 0)
    := "001";

begin
cvg_register : process(clk)
begin
    if rising_edge(clk) then
        cvg_state <= cvg_state(1 downto 0) & cvg_state(2);
    end if;
end process;

```

Cxema CVG (Constant Value Generator)



Q2	Q1	Q0	$F_0 = 0$	
0	0	1	0	Начальное состояние
0	1	0	0	1-й такт (начало цикла)
1	0	0	0	2-й такт
0	0	1	0	3-й такт (повтор)

Сравнительный анализ ресурсоемкости

Вариант реализации	Кол-во LUT-блоков	Типы LUT-блоков	Триггеры	BUFGCTRL
Без защиты	3	$2 \times \text{LUT6}$, $1 \times \text{LUT3}$	0	0
DONT_TOUCH	3	$3 \times \text{LUT6}$	0	0
Ложные сигналы	6	$3 \times \text{LUT6}$, $3 \times \text{LUT1}$	0	0
CVG	3	$3 \times \text{LUT6}$	3	1

Заключение

- В работе рассмотрен метод внедрения водяных знаков и его применимость в качестве системы скрытой маркировки VHDL-описаний для защиты интеллектуальной собственности.
- Данная работа способствует повышению устойчивости внедрённой информации к оптимизациям САПР.
- Разработано программное средство, позволяющее автоматизировано внедрять и извлекать водяные знаки в реальных FPGA-проектах.
- Планируется исследование и разработка других методов внедрения водяных знаков для повышения их устойчивости к оптимизации и сохранения скрытой информации.
- Планируется экспериментальная оценка влияния настроек оптимизации Vivado 2026.1 на количество используемых LUT-блоков в произвольно выбранных FPGA-проектов.
- Предполагается расширение экспериментальных исследований для сравнения устойчивости водяных знаков, изменения аппаратных ресурсов и определения наиболее эффективных подходов к их внедрению.

Спасибо за внимание!