УДК 004.04:519.6

Гранкин Виталий Владимирович

РАЗРАБОТКА ТАБЛИЧНОГО МЕТОДА ВЫЧИСЛЕНИЯ ЗНАЧЕНИЙ ЭЛЕМЕНТАРНЫХ ФУНКЦИЙ В МОДУЛЯРНОМ КОДЕ

В стстье рассмотрено решение задачи эффективного табличного вычисления элементарных функций в модулярном коде как со стороны аппаратных затрат так и со стороны быстродействия.

Ключевые слова: *табличные вычисления, элементарные функции, модулярный код, система остаточных классов, кольцо вычетов, полиномиальная аппроксимация.*

Grankin Vitaly V.

DEVELOPMENT OF TABULAR METHOD FOR CALCULATION OF PRIMITIVE FUNCTION IN MODULAR CODE

The item dwells on the solution for tabular calculation of primitive function in modular code, both in view of hardware costs and in view of the processing speed.

Key words: *tabular calculation, primitive functions, modular code, system of residual classes, residue ring, polynomial approximation.*

На современном этапе развития информационных технологий особо значимой становиться проблема проведения параллельных вычислений. Данная проблема возникла в связи с физическими ограничениями роста быстродействия вычислительных устройств, упирающегося в невозможность эффективного повышения тактовой частоты вычислительных процессоров [1]. Производительность вычислительных узлов процессора главным образом увеличивается за счет совершенствования технологического процесса, что в настоящее время, трудноосуществимо. Вычислительная мощность системы может быть увеличена благодаря переходу к многопроцессорным и параллельным вычислениям. Параллельные алгоритмы важны ввиду постоянного совершенствования таких систем и значительного увеличения числа ядер в современных процессорах. В связи с остротой проблемы возникает необходимость в разработке математических методов, поддающихся распараллеливанию.

Часто во многих научных задачах требуется выполнить расчеты или произвести моделирование процесса с высокой точностью. Аппаратные средства ЭВМ позволяют непосредственно выполнять вычисления ограниченной точности. Для преодоления обозначенной трудности обычно используются длинная арифметика, точность при этом превышает размер разрядной сетки и ограничена только объемом доступной памяти и временем вычислений [2]. При этом время вычислений существенно зависит от размера чисел, над которыми производятся операции.

Приложение модулярной арифметики может быть одним из эффективных методов решения данной проблемы [3]. В работе [4] предложен метод вычисления элементарных функций в СОК путем полиномиальной аппроксимации.

Данный метод был реализован в среде National Instruments LabView и опробован на плате National Instruments Digital Electronics FPGA BOARD с ПЛИС Xilinx S3E500.

На рис. 1 изображен модулярный сумматор, примененный в вычислителях. Так же значительное количество ресурсов было израсходовано на реализацию внутренних устройств LabView. В ходе реализации метода было замечено достаточно большое потребление ресурсов ПЛИС, расходуемых, прежде всего на модулярный умножитель (рис. 2) и состоящий из нескольких умножителей вычислитель значения полинома (рис. 3).

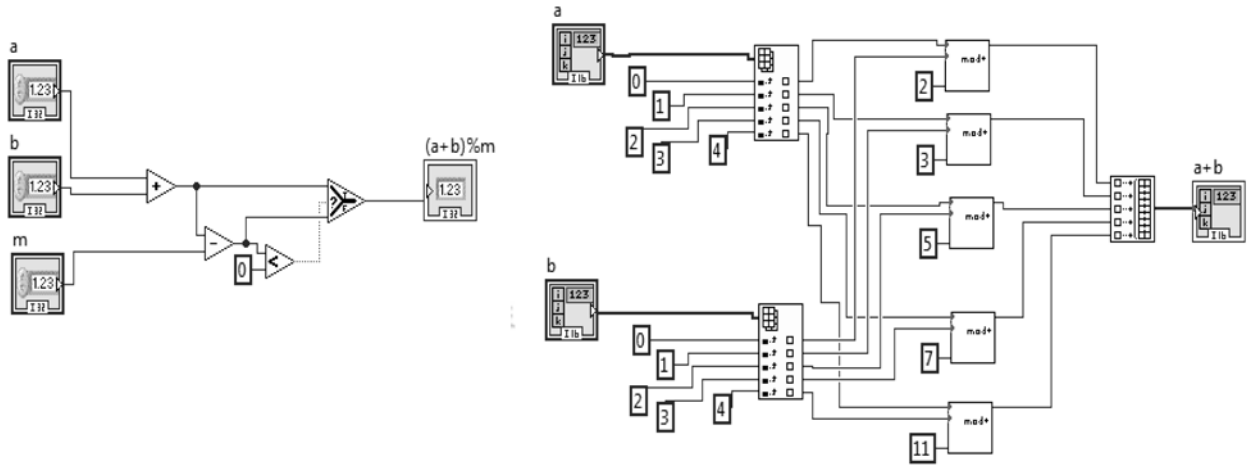

Рис. 1. Модулярный сумматор, слева для одного основания,
справа по основаниям (2, 3, 5, 7, 11)

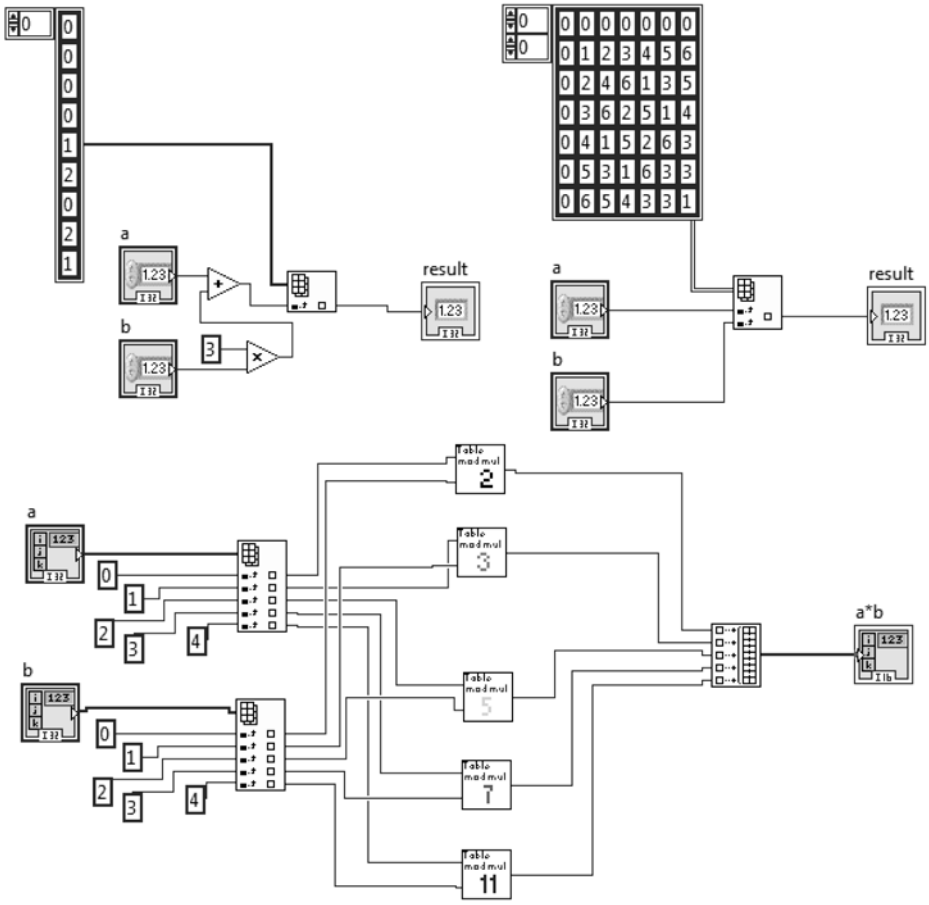

Рис. 2. Модулярный умножитель по основаниям (2, 3, 5, 7, 11)

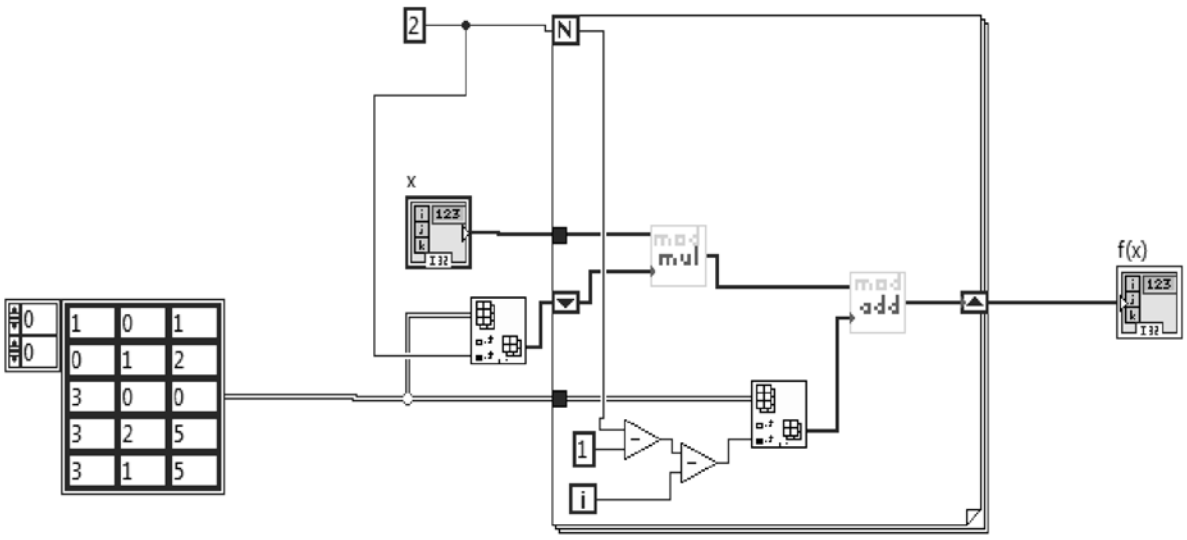

Рис. 3. Модулярный вычислитель полинома по основаниям (2, 3, 5, 7,11)

На рис. 3 изображен модулярный вычислитель полинома с тремя коэффициентами (высшая степень полинома – 2). Данный вычислитель производит вычисления за один тактирующий цикл, однако синтезируется в 3 модулярных умножителя и 3 модулярных сумматора, что увеличивает аппаратные затраты. Так же такая реализация накладывает ограничения на увеличение тактовой частоты. На рис. 4 изображен Verilog-код модулярного вычислителя полинома. Данный вычислитель многотактный (вычисление с накоплением) и требует для вычисления 3 такта.

```
451 module mod_exp_2357_11(clk, reset, x, out);
452    input clk, reset;
453    input [12:0]   x;
454    output reg [12:0] out;
455    reg [12:0] coef[0:2];
456    reg [1:0] count;
457    initial begin
458       coef[0] = {4'd5, 3'd5, 3'd0, 2'd2, 1'd1};
459       coef[1] = {4'd1, 3'd2, 3'd0, 2'd1, 1'd0};
460       coef[2] = {4'd10, 3'd6, 3'd0, 2'd1, 1'd0};
461    end
462
463    wire [12:0] mul_res, add_res;
464    mod_add_2357_11 modadd(coef[count], mul_res, add_res);
465    mod_mul_2357_11 modmul(out, x, mul_res);
466
467    always @(posedge clk)
468    begin
469       if(reset)
470       begin
471          count <= 1;
472          out <= coef[0];
473       end
474       else begin
475          if(count != 3)
476          begin
477             count <= count + 1;
478             out <= add_res;
479          end
480       end
481    end
482 endmodule
```

Рис. 4. Модулярный вычислитель значения полинома по основаниям (2, 3, 5, 7, 11) на языке Verilog

Анализ вычисления элементарных функций в СОК посредством табличных вычислений. Табличный способ вычисления элементарных функций зачастую является простым и достаточно удобным для реализации «на кристалле», однако, это верно для позиционной системы счисления, для системы остаточных классов прямой подход к табличным вычислениям недостаточно удобен. Суть табличных вычислений состоит в построении вычислителя, состоящего из компаратора и таблицы значений. При поступлении аргумента на вход табличного вычислителя производится поиск: линей-

ный, бинарный, на основе комбинационных схем дешифратор-шифратор-мультиплексор или другой, на основе результата поиска производится выборка значения функции из пары аргумент значение, с наиболее подходящим аргументом.

Табличное вычисление требует некоторый объем памяти для хранения, обычно $2n$ элементов (n – таблица поиска, n – таблица значений). Поиск нужного значения происходит достаточно быстро, например бинарный поиск по упорядоченной таблице осуществляется за логарифм от числа элементов $\log_2 n$.

В кольце вычетов не задана операция сравнения и, следовательно, не имеется прямой возможности проведения бинарного поиска [5]. Проведение линейного поиска возможно только в случае, когда таблица имеет число строк равное величине представления наибольшего числа в СОК $M = m_1 \cdot m_2 \cdot \ldots \cdot m_n$ с данными основаниями $(m_1, m_2, \ldots, m_n)$, тем самым накладывается значительное ограничение на модули и величину наибольшего числа СОК, так как требует таблицу с $\widetilde{i_{\leq n}}$ элементами. Так же линейный поиск в СОК можно производить только сравнением на равенство значения остатков аргумента со значениями остатков в таблице. Сравнение на равенство не является модулярной операцией и требует одновременного поступления всех остатков числа на вычислитель, что означает усложнение схем и возникновение задержек, связанных с ожиданием поступления сигналов со всех линий на вычислитель.

Разработка эффективного табличного метода вычисления элементарных функций в СОК. Как было сказано, традиционный способ построения таблицы в СОК требует 2n элементов для реализации биективного отображения, что является избыточным. В свою очередь полиномиальная аппроксимация функций в системе остаточных классов представляет собой сюръективное отображение множества аргументов функций на значительно меньшее множество значений функций. Можно заметить, что отображение остатков СОК полиномиальной аппроксимации, аргумент значение является биективным и взаимно однозначным. Что позволяет построить отдельные таблицы для каждого основания, причем для каждого основания таблица будет полной и отображать все возможные значения. Поскольку отображаются все возможные значения входа выхода остатка СОК, таблицы могут быть синтезированы, в том числе и в виде комбинационных схем. Аппаратные затраты для каждого модуля СОК составят O (mi) и для всего вычислителя O (Σmi). Имеется возможность применения бинарного поиска по основанию СОК. Бинарный поиск осуществляется по основаниям независимо и такие таблицы можно хранить во внешней памяти, например в оперативной, что уменьшает аппаратные затраты вычислителя.

На рис. 5 показана комбинационная схема табличного вычислителя элементарной функции $f(x) = 5 \cdot x^2$, областью значений которой является множество $\{0,1,2\}$, для реализации которого необходимо 2 дешифратора-шифратора, асимптотика аппаратных затрат составляет $O(n)$. В табл. 1 указаны значения входа-выхода модулярного и традиционного вычислителя и на рис. 6 его схема.

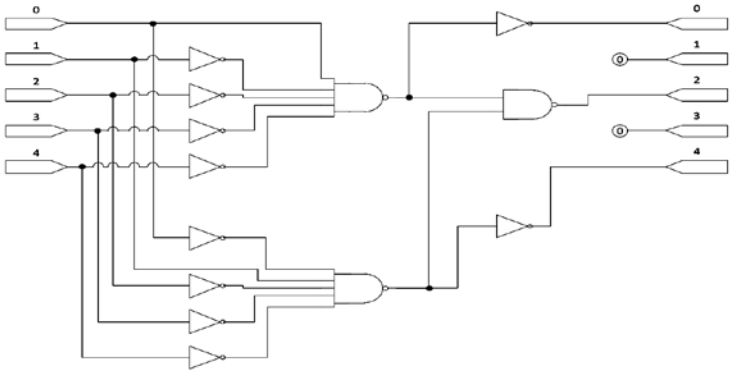

Рис. 5. Комбинационная схема табличного вычисления функции $f(x) = 5 \cdot x^2$ в десятичной системе счисления с $x \in \{0,1,2\}$

Таблица 1

Значения входа-выхода модулярного табличного вычислителя функции $f(x)=5\cdot x^2$, по основаниям $M=2\cdot 3\cdot 5=30$

Десятичное на входе	Основания СОК на входе			Основания СОК на выходе			Десятичное на выходе
	2	3	5	2	3	5	
0	0	0	0	0	0	0	0
1	1	1	1	1	2	0	5
2	0	2	2	0	2	0	20
3	1	0	3	1	0	0	15
4	0	1	4	0	2	0	20
5	1	2	0	1	2	0	5

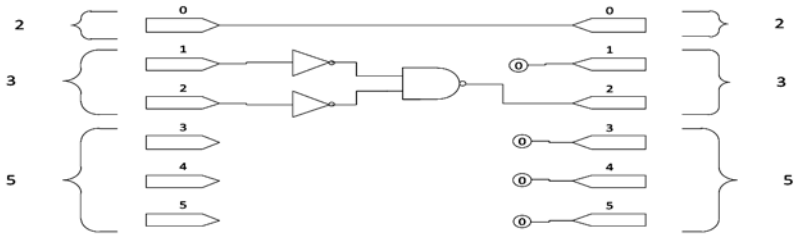

Рис. 6. комбинационная схема табличного вычисления функции $f(x)=5\cdot x^2$ в СОК с основаниями $m=(2,3,5), M=30$

На рис. 7 показана реализация модулярной функции $f(x)=e^x$ с основаниями $m=(2,3,5,7,11),\mathrm{M}=2310$ на языке Verilog, данная реализация выполнена в поведенческом стиле. Благодаря тому, что основания СОК биективны, блок может быть описан, в том числе и комбинационной схемой, без тактирующего сигнала.

```
485 module mod_exp_tabled_2357_11(x, out);
486    input [12:0] x;
487    output reg [12:0] out;
488
489    always @*
490    begin
491       case (x[0:0])//2
492          0: out[0] <= 0;
493          default: out[0] <= 1;
494       endcase
495
496       case (x[2:1])//3
497          0: out[2:1] <= 1;
498          1: out[2:1] <= 1;
499          default: out[2:1] <= 2;
500       endcase
501
502       case (x[5:3])//5
503          0: out[5:3] <= 0;
504          1: out[5:3] <= 0;
505          2: out[5:3] <= 0;
506          3: out[5:3] <= 0;
507          default: out[5:3] <= 0;
508       endcase
509
510       case (x[8:6])//7
511          0: out[8:6] <= 6;
512          1: out[8:6] <= 6;
513          2: out[8:6] <= 2;
514          3: out[8:6] <= 1;
515          4: out[8:6] <= 3;
516          5: out[8:6] <= 1;
517          default: out[8:6] <= 2;
518       endcase
519
520       case (x[12:9])//11
521          0: out[12:9] <= 10;
522          1: out[12:9] <= 5;
523          2: out[12:9] <= 10;
524          3: out[12:9] <= 3;
525          4: out[12:9] <= 6;
526          5: out[12:9] <= 8;
527          6: out[12:9] <= 9;
528          7: out[12:9] <= 9;
529          8: out[12:9] <= 8;
530          9: out[12:9] <= 6;
531          default: out[12:9] <= 3;
532       endcase
533    end
534 endmodule
```

Рис. 7. Пример реализации табличного модулярного вычислителя функции $f(x)=e^x$ с основаниями $m=(2,3,5,7,11),\mathrm{M}=2310$ на языке Verilog

В табл. 2 указаны значения потребления ресурсов ПЛИС, в случае «без вычисления функции» (колонка 3) ресурсов задействовано немного больше, чем в случае табличного вычисления, это видимо, связано с моделью оптимизации среды Xilinx ISE 14.6.

Таблица 2

Сравнение потребления ресурсов ПЛИС Xilinx S3E500 при реализации вычислителя модулярной функции $f(x)=e^x$

Показатель потребления ресурсов	Вычисление полинома	Табличное вычисление	Без вычисления функции
Триггеров (Slice Flip Flops)	393	366	369
Таблицы поиска (4 input LUTs)	999	866	867
Number of occupied Slices	541	453	454

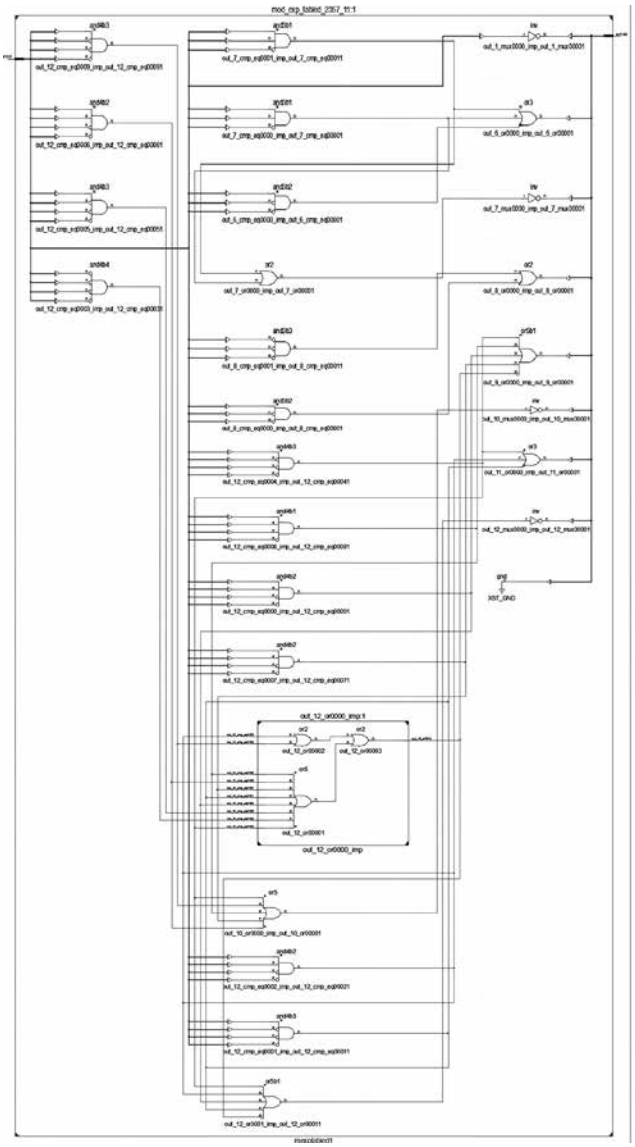

Рис. 8. Результат синтезированной реализации табличного модулярного вычислителя функции $f(x)=e^x$ с основаниями $m=(2,3,5,7,11),\mathrm{M}=2310$

Достоинством разработанного табличного метода вычисления элементарных функций в системе остаточных классов является как высокое быстродействие, так и асимптотика $O\left(\sum m_i\right)$ роста необходимых аппаратных ресурсов вычислителя. Так вычислитель, реализованный комбинационными схемами, способен выдавать результат за период одного тактирующего сигнала, при этом длина комбинационного пути может не превышать трех элементов.

Таблицы также могут храниться во внешней памяти, поиск осуществляется по основаниям независимо, причем может быть применен бинарный поиск с логарифмической асимптотикой, что решает проблему значительных аппаратных затрат при вычислениях с модулями большой разрядности.

Литература

1. Zhirnov V. V. Limits to Binary Logic Switch Scaling–A Gedanken Model [Текст] / Victor V. Zhirnov, Ralph K. Cavin, III, James A. Hutchby, George I. Bourianoff. Proceedings Of The IEEE, VOL. 91, NO. 11, november 2003. P.1.

2. Оцоков Ш. А. Диссертация кандидата технических наук. Алгоритмическая и структурная организация высокопроизводительных ЭВМ с использованием модели безошибочных вычислений. 2003 г. / дис. … канд. техн. наук: 05.13.15, 05.13.05. Москва, 2003. С.13, 264.

3. Червяков Н. И. Реализация высокоэффективной модулярной цифровой обработки сигналов на основе программируемых логических интегральных схем. 2006 г. // Нейрокомпьютеры: разработка и применение. №10. 2006. С. 24–36.

4. Гранкин В. В., Мезенцева О. С. К вопросу о разработке полиномиального метода вычисления элементарных функций в модулярном коде. 2013 г. // Материалы II Международной научно-практической конференции «Актуальные проблемы современной науки». Выпуск 2, том 2. Тенденции развития информационных технологий. Ставрополь: СевКавГТИ, 2013 г. С. 165–169.

5. Грегори Р, Кришнамурти Е. Безошибочные вычисления. Методы и приложения. 1992 г. / пер. с англ. М.:«Мир», 1992. С. 21.

УДК 004.9

Жарких Андрей Анатольевич, Шагрова Галина Вячеславовна

СИСТЕМА ФОРМИРОВАНИЯ И КОНТРОЛЯ ЛАТЕНТНЫХ ИЗОБРАЖЕНИЙ

В статье представлены результаты проведенного исследования методов, используемых для формирования латентных изображений. Предложено определение системы формирования и контроля латентных изображений объединяющей свойства всех латентных изображений.

Ключевые слова: *латентное изображение, скрытое изображение, скрываемое изображение, выявленное изображение.*

Zharkikh Andrey A., Shagrova Galina V.

SYSTEM FOR DEVELOPMENT AND CONTROL OF LATENT IMAGES

This item presents the results of research into the methods employed for latent image development. The author proposes a definition for a system of latent image development and control, which comprises the features of all latent images.

Key words: *latent image, hidden image, uncovered image.*

В настоящее время существует множество различных способов формирования изображений обладающих свойством изменения видимости своих элементов при изменении условий наблюдения или способа его регистрации, которые принято называть латентными изображениями. При этом зачастую, при разработке таких алгоритмов, не учитываются свойства общие для всех латентных изобра-