ЛЕКЦИЯ № 6
КОНТРОЛЬ РАБОТЫ ЦИФРОВОГО АВТОМАТА

1. Информационные основы контроля работы цифрового автомата
2. Методы контроля цифрового автомата

1. Информационные основы контроля работы цифрового автомата

Алгоритмы выполнения арифметических операций обеспечат правильный результат только в случае, если машина работает без нарушений. При возникновении какого-либо нарушения нормального функционирования результат будет неверным, однако пользователь об этом не узнает, если не будут предусмотрены меры для создания системы обнаружения возможной ошибки, а с другой стороны, должны быть проработаны меры, позволяющие исправить ошибки. Эти функции следует возложить на систему контроля работы цифрового автомата.

Определение 1 : Система контроля - совокупность методов и средств, обеспечивающих определение правильности работы автомата в целом или его отдельных узлов, а также автоматическое исправление ошибки.

Ошибки в работе цифрового автомата могут быть вызваны либо выходом из строя какой-то детали, либо отклонением от нормы параметров (например, изменение напряжения питания) или воздействием внешних помех. Вызванные этими нарушениями ошибки могут принять постоянный или случайный характер. Постоянные ошибки легче обнаружить и выявить. Случайные ошибки, обусловленные кратковременными изменениями параметров, наиболее опасны и их труднее обнаружить.

Поэтому система контроля должна строиться с таким расчетом, чтобы она позволяла обнаружить и по возможности исправить любые нарушения. При этом надо различать следующие виды ошибок результата:

1) возникающие из-за погрешностей в исходных данных;
2) обусловленные методическими погрешностями;
3) появляющиеся из-за возникновения неисправностей в работе машины.

Первые два вида ошибок не являются объектом для работы системы контроля.

Помехоустойчивые коды – одно из наиболее эффективных средств обеспечения высокой верности передачи дискретной информации.

Создана специальная теория помехоустойчивого кодирования, быстро развивающаяся в последнее время.

Бурное развитие теории помехоустойчивого кодирования связано с внедрением автоматизированных систем, у которых обработка принимаемой информации осуществляется без участия человека. Использование для обра-

ботки информации электронных цифровых вычислительных машин предъявляет очень высокие требования к верности передачи информации.

Рассмотрим сущность помехоустойчивого кодирования.

Определение 2: Под помехоустойчивыми или корректирующими кодами понимают коды, позволяющие обнаружить и устранить ошибки, происходящие при передаче информации из-за влияния помех.

Введём следующие обозначения.

Количество разрядов n в кодовой комбинации будем называть длиной или значностью кода. Каждый разряд может принимать значение 0 или 1. Количество единиц в кодовой комбинации назовём весом кодовой комбинации и обозначим ω.

Например, кодовая комбинация 100101100 характеризуется значностью n=9 и весом ω=4.

Степень отличия любых двух кодовых комбинаций данного кода характеризуется так называемым расстоянием между кодами d. Оно выражается числом позиций или символов, в которых комбинации отличаются одна от другой. Кодовое расстояние есть минимальное расстояние между кодовым комбинациями данного кода, оно определяется как вес суммы по модулю два кодовых комбинаций. Например, для определения расстояния между комбинациями 100101100 и 110110101 необходимо просуммировать их по модулю два.

ВНИМАНИЕ! Суммирование по модулю 2 выполняется следующим образом : при поразрядном суммировании совпадающие по значению разряды в сумме дают 0, не совпадающие в сумме дают 1.

Например:

$$\begin{array}{r} 100101100 \\ \oplus\ \underline{110110101} \\ 010011001 \end{array}$$

Полученная в результате суммирования новая кодовая комбинация характеризуется весом ω=4. Следовательно, расстояние между исходными кодовыми комбинациями d=4.

Ошибки, вследствие воздействия помех, проявляются в том, что в одном или нескольких разрядах кодовой комбинации нули переходят в единицы и, наоборот, единицы переходят в нули. В результате создается новая ложная кодовая комбинация.

Если ошибки происходят только в одном разряде кодовой комбинации, то такие ошибки называются однократными. При наличии ошибок в двух, трех и т.д. разрядах ошибки называются двукратными, трехкратными и т.д.

Помехоустойчивость кодирования обеспечивается за счет введения избыточности (т.е. добавления дополнительных контрольных битов) в кодовые

комбинации. Это значит, что из *n* символов кодовой комбинации для передачи информации используется *k<n* символов.

Определение 3: Систематический код – код, содержащий в себе кроме информационных контрольные разряды.

В контрольные разряды записывается некоторая информация об исходном числе.

2. Методы контроля цифрового автомата

2.1 Кодирование по методу четности-нечетности

Если в математическом коде выделен один контрольный разряд (k=1), то к каждому двоичному числу добавляется один избыточный разряд и в него записывается 1 или 0 с таким условием, чтобы сумма цифр в каждом числе была по модулю 2 равна 0 для случая четности. Появление ошибки в кодировании обнаружится по нарушению четности (нечетности). При этом допускается, что может возникнуть только одна ошибка.

Пример реализации метода четности представлен в таблице 1.

Таблица 1

Переданное сообщение		
Информационные разряды	Контрольный разряд	Наличие ошибки
10101011	1	0
11001010	0	0
10010001	1	0
11001011	0	1-нарушение

Такое кодирование имеет минимальное кодовое расстояние d = 2.

При таком способе избыточного кодирования локализовать ошибку (т.е. обнаружить место возникновения ошибки) невозможно.

Представим теперь видоизмененный способ контроля по методу четности – нечетности. Длинное число (информационное сообщение) разбивается на группы по *l* разрядов. Контрольные разряды выделяются всем группам по строкам и по столбцам согласно следующей схеме:

a_1	a_2	a_3	a_4	a_5	k_1
a_6	a_7	a_8	a_9	a_{10}	k_2
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	K_3
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	k_4
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	k_5
k_6	k_7	k_8	k_9	k_{10}	

Где a_i – информационные биты, k_j – контрольные биты (или контрольные разряды. Они же избыточные разряды).

Увеличение избыточности информации приводит к тому, что появляется возможность не только обнаружить ошибку, но и исправить ее. Пусть произошла неисправность в каком-то из разрядов этого числа (представим, что разряд a_{18} изменил состояние, т. е. a_{18}=1). Это приведет к тому, что при проверке на четность сумма $\sum_i (a_i + k_i)$ по соответствующим строкам и столбцам изменится для значений, которые содержат элемент $a18$, т. е. это будет четвертая сверху строка и третий слева столбец. Следовательно, нарушение четности по этой строке и столбцу можно зафиксировать, что в конечном счете означает обнаружение не только самой ошибки, но и места, где возникла ошибка. Изменив содержимое отмеченного разряда (в данном случае $a18$) на противоположное, можно исправить ошибку.

Пример . Определить и исправить ошибку в передаваемой информации вида

1001110	0
1110101	0
0101101	0
1010110	0
1101011	1
0001011	

Для контроля использовать метод четности по строкам и столбцам (контрольный столбец 8, контрольная строка 6).

Решение. Прежде всего осуществим проверку на четность по каждой строке: $k_1 = 0$; $k_2 = 1$; $k_3 = 0$; $k_4 = 0$; $k_5 = 0$.

Затем проверим на четность информацию по столбцам: $k_6 = 0$; $k_7 = 1$; $k_8 = 0$; $k_9 = 0$; $k_{10} = 0$; $k_{11} = 0$; $k_{12} = 0$.

Проверка показывает, что ошибка возникла в разряде второй строки и второго слева столбца. Следовательно, разряд, содержащий ошибочную информацию, находится на пересечении второй строки и второго столбца.

Ответ:

1001110	0
1010101	0
0101101	0
1010110	0
1101011	1
0001011	

Контроль по методу четности-нечетности широко используется в ЭВМ для контроля записи, считывания информации в запоминающих устройствах на магнитных носителях, а также при выполнении арифметических операций.

2.2 Контроль по модулю

Разнообразные задачи можно решать с помощью метода контроля, основанного на свойствах сравнений. Развитые на этой основе методы контроля арифметических и логических операций называют *контролем по модулю*.

Существуют два метода получения контрольного кода: числовой и цифровой.

2.2.1 Числовой метод контроля.

При числовом методе контроля код заданного числа определяется как наименьший положительный остаток от деления числа на выбранный модуль p:

$$r_A = A - \{A/p\}\, p, \qquad (1)$$

где в фигурных скобках {} – целая часть от деления числа; A – контролируемое число.

Величина модуля p существенно влияет на качество контроля; если $p = q$ (q – основание системы счисления, в которой выражено число) и имеет место числовой контроль, то контролируется только младший разряд числа и контроль как таковой не имеет смысла; для $p = q$m справедливы аналогичные соображения, так как если $m < n$, то опять не все разряды числа участвуют в контроле и ошибки в разрядах старше m вообще не воспринимаются.

При числовом методе контроля по модулю p для определения остатка используют операцию деления, требующую больших затрат машинного времени. Для числового метода контроля справедливы основные свойства сравнений (сложение, умножение сравнений и т. д.). Поэтому, если $A \equiv r_A (\mathrm{mod}\, p)$; $B \equiv r_B (\mathrm{mod}\, p)$, где $0 \le r_A \le p-1$; $0 \le r_B \le p-1$, то $A + B \equiv r_A + r_B (\mathrm{mod}\, p)$. Отсюда

$$r_{A+B} \equiv r_A + r_B (\mathrm{mod}\, p) \qquad (2)$$

Аналогичным образом доказывается справедливость и следующих соотношений:

$$r_{A-B} \equiv r_A - r_B (\mathrm{mod}\, p); \quad (3)$$
$$r_{AB} \equiv r_A r_B (\mathrm{mod}\, p) \quad (4)$$

Пример. Для заданных чисел A=125 и B=89 определить контрольные коды самих чисел, их суммы и разности, если модуль p=11.

Решение. Контрольные коды чисел определяем по (1):

$$r_A = 125 - \{125/11\}11 = 4;\ \ r_B = 89 - \{89/11\}11 = 1.$$

Аналогично находим контрольные коды для суммы и разности:

$$A + B = 214,\ r_{A+B} = 214 - \{214/11\}11 = 5;$$
$$A - B = 36,\ r_{A-B} = 36 - \{36/11\}11 = 3.$$

Проверку правильности определения контрольных кодов суммы и разности можно провести на основании (2) и (3):

$$r_{A+B} = 4 + 1 \equiv 5(\mathrm{mod}11);\ \ r_{A-B} = 4 - 1 \equiv 3(\mathrm{mod}11).$$

Ответ: $r_A = 4;\ r_B = 1;\ r_{A+B} = 5;\ r_{A-B} = 3.$

2.2.2 Цифровой метод контроля.

При цифровом методе контроля контрольный код числа образуется делением суммы цифр числа на выбранный модуль:

$$r_A^{'} = \sum_i a_i - \left\{ \frac{\sum_i a_i}{p} \right\} p$$

или

$$r^{'} \equiv \sum_i a_i (\mathrm{mod}\, p). \quad (5)$$

Возможны два пути получения контрольного кода: 1) непосредственное деление суммы цифр на модуль p; 2) суммирование цифр по модулю p.

Второй путь проще реализуется, так как если $a_i < p$, то контрольный код получается только операцией суммирования.

Пример. Определить контрольные коды чисел A = 153 и B = 41, их суммы и разности, если p = 11.

Решение. Контрольные коды исходных чисел определяем по (5). Для этого находим суммы цифр и делим их на модуль: $\sum_i a_i = 9;\ \sum_i b_i = 5.$

Следовательно, $r_A^{'} = 9;\ r_B^{'} = 5.$

Аналогично определяем контрольные коды суммы и разности:

$$C = A + B = 194;\ \ \sum c_i = 14;\ \ r_C^{'} \equiv 3(\mathrm{mod}11);$$
$$D = A - B = 112;\ \ \sum d_i = 4;\ \ r_D^{'} \equiv 4(\mathrm{mod}11).$$

Ответ: $r_A^{'} = 9,\ r_B^{'} = 5,\ r_{A+B}^{'} = 3,\ r_{A-B}^{'} = 4.$

Однако при цифровом методе свойства сравнений не всегда справедливы, и происходит это из-за наличия переносов (заемов) при выполнении арифметических действий над числами. Поэтому нахождение контрольного кода результата операции происходит обязательно с коррекцией.

Пусть заданы числа *A* и *B* и соответственно их контрольные коды

$$r_A^{'} \equiv \sum_i a_i (\mathrm{mod}\, p);\ \ r_B^{'} \equiv \sum_i b_i (\mathrm{mod}\, p);\ \ C = A + B.$$

Найдем контрольный код $r_C^{'}$.

Видимо, когда есть результат операции, тогда найти $r_C^{'}$ методом суммирования цифр по модулю не сложно. Какова будет возможность получения $r_C^{'}$ через контрольные коды слагаемых?

Сумму цифр *ci* числа можно найти, зная цифры *ai* и *bi* и количество переносов в каждом разряде. Каждый перенос уносит из данного разряда *q* единиц и добавляет одну единицу в следующий разряд, т. е. сумма цифр уменьшится на величину *q* – 1 на каждый перенос. Тогда

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i - l(q-1), \qquad (6)$$

где *l* – количество переносов, возникших при сложении.

Так как $r_A^{'} \equiv \sum_i a_i (\mathrm{mod}\, p);\ \ r_B^{'} \equiv \sum_i b_i (\mathrm{mod}\, p),$ то $r_C^{'} = \sum_i c_i (\mathrm{mod}\, p).$

Подставив эти значения в (12), получим

$$r_C^{'} = [r_A^{'} + r_B^{'} - l(q-1)](\mathrm{mod}\, p). \qquad (7)$$

Аналогичными рассуждениями можно показать, что для разности чисел *C = A – B*

$$r_C^{'} \equiv [r_A^{'} - r_B^{'} + s(q-1)](\mathrm{mod}\, p), \qquad (8)$$

где *s* – количество заемов при выполнении операции.

Пример. Определить контрольные коды чисел *A* = 589 и *B* = 195, их суммы и разности, если *p* = 11.

Решение. Контрольные коды исходных чисел определяем по (11). При этом используем второй путь, т. е. нахождение контрольного кода суммированием цифр по модулю:

$$\sum a_i = 5 \oplus 8 \oplus 9 \equiv 0(\mathrm{mod}11);\ \ r_A^{'} \equiv 0(\mathrm{mod}11);$$

$$\sum b_i = 1 \oplus 9 \oplus 5 \equiv 4(\mathrm{mod}11);\ \ r_B^{'} \equiv 4(\mathrm{mod}11).$$

Контрольный код суммы определяем по (6) – в этом случае *l* =2:

$$A + B = 784;\ \ r_{A+B}^{'} \equiv 0 + 4 - 2(10-1) \equiv 8(\mathrm{mod}11).$$

В случае, когда имеет место отрицательный остаток, к сравнению надо добавить модуль *p* столько раз, сколько необходимо для получения ближайшего положительного остатка.

Контрольный код разности получим по (8) – в этом случае *s* = 1:

$$A - B = 394;\ \ r_{A-B}^{'} \equiv 0 - 4 + 1(10-1) \equiv 5(\mathrm{mod}11).$$

Ответ: $r'_A = 0,\ r'_B = 4,\ r'_{A+B} = 8,\ r'_{A-B} = 5.$

2.3 Коды Хэмминга

Коды, предложенные американским ученым Р. Хеммингом, способны не только обнаружить, но и исправить одиночные ошибки. Эти коды – систематические.

Коды Хэмминга — наиболее известные и, вероятно, первые из самоконтролирующихся и самокорректирующихся кодов. Построены они применительно к двоичной системе счисления.

Другими словами, это алгоритм, который позволяет закодировать какое-либо информационное сообщение определённым образом и после передачи (например по сети) определить появилась ли какая-то ошибка в этом сообщении (к примеру из-за помех) и, при возможности, восстановить это сообщение. Сегодня, я опишу самый простой алгоритм Хемминга, который может исправлять лишь одну ошибку.

Также стоит отметить, что существуют более совершенные модификации данного алгоритма, которые позволяют обнаруживать (и если возможно исправлять) большее количество ошибок.

Сразу стоит сказать, что Код Хэмминга состоит из двух частей. Первая часть кодирует исходное сообщение, вставляя в него в определённых местах контрольные биты (вычисленные особым образом). Вторая часть получает входящее сообщение и заново вычисляет контрольные биты (по тому же алгоритму, что и первая часть). Если все вновь вычисленные контрольные биты совпадают с полученными, то сообщение получено без ошибок. В противном случае, выводится сообщение об ошибке и при возможности ошибка исправляется.

Как это работает.

Для того, чтобы понять работу данного алгоритма, рассмотрим пример.

Подготовка

Допустим, у нас есть сообщение «habr», которое необходимо передать без ошибок. Для этого сначала нужно наше сообщение закодировать при помощи Кода Хэмминга. Нам необходимо представить его в бинарном виде.

Символ	ACSII код	Бинарное представление
h	68	01000100
a	61	00111101
b	62	00111110
r	72	01001000

На этом этапе стоит определиться с, так называемой, длиной информационного слова, то есть длиной строки из нулей и единиц, которые мы будем кодировать. Допустим, у нас длина слова будет равна 16. Таким образом, нам

необходимо разделить наше исходное сообщение («habr») на блоки по 16 бит, которые мы будем потом кодировать отдельно друг от друга. Так как один символ занимает в памяти 8 бит, то в одно кодируемое слово помещается ровно два ASCII символа. Итак, мы получили две бинарные строки по 16 бит:

h	a
01000100	00111101

и

b	r
00111110	01001000

После этого процесс кодирования распараллеливается, и две части сообщения («ha» и «br») кодируются независимо друг от друга. Рассмотрим, как это делается на примере первой части.

Прежде всего, необходимо вставить контрольные биты. Они вставляются в строго определённых местах — это позиции с номерами, равными степеням двойки. В нашем случае (при длине информационного слова в 16 бит) это будут позиции 1, 2, 4, 8, 16. Соответственно, у нас получилось 5 контрольных бит (выделены красным цветом):

Было:

h	a
01000100	00111101

Стало:

h	a
000010000100	001011101

Таким образом, длина всего сообщения увеличилась на 5 бит. До вычисления самих контрольных бит, мы присвоили им значение «0».
Вычисление контрольных бит.

Теперь необходимо вычислить значение каждого контрольного бита. Значение каждого контрольного бита зависит от значений информационных бит (как неожиданно), но не от всех, а только от тех, которые этот контрольных бит контролирует. Для того, чтобы понять, за какие биты отвечает каждых контрольный бит необходимо понять очень простую закономерность: контрольный бит с номером N контролирует все последующие N бит через каждые N бит, начиная с позиции N. Не очень понятно, но по картинке, думаю, станет яснее:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
X		X		X		X		X		X		X		X		X		X		X	1
	X	X			X	X			X	X			X	X			X	X			2
			X	X	X	X					X	X	X	X					X	X	4
							X	X	X	X	X	X	X	X							8
															X	X	X	X	X	X	16

Здесь знаком «X» обозначены те биты, которые контролирует контрольный бит, номер которого справа. То есть, к примеру, бит номер 12 контролируется битами с номерами 4 и 8. Ясно, что чтобы узнать какими битами контролируется бит с номером N надо просто разложить N по степеням двойки.

Но как же вычислить значение каждого контрольного бита? Делается это очень просто: берём каждый контрольный бит и смотрим сколько среди контролируемых им битов единиц, получаем некоторое целое число и, если оно чётное, то ставим ноль, в противном случае ставим единицу. Вот и всё! Можно конечно и наоборот, если число чётное, то ставим единицу, в противном случае, ставим 0. Главное, чтобы в «кодирующей» и «декодирующей» частях алгоритм был одинаков. (Мы будем применять первый вариант).

Высчитав контрольные биты для нашего информационного слова получаем следующее:

h	a
100110000100	001011101

и для второй части:

b	r
100101101110	010101000

Вот и всё! Первая часть алгоритма завершена.

Декодирование и исправление ошибок.

Теперь, допустим, мы получили закодированное первой частью алгоритма сообщение, но оно пришло к нас с ошибкой. К примеру мы получили такое (11-ый бит передался неправильно):

h	a
100110000110	001011101

Вся вторая часть алгоритма заключается в том, что необходимо заново вычислить все контрольные биты (так же как и в первой части) и сравнить их с контрольными битами, которые мы получили. Так, посчитав контрольные биты с неправильным 11-ым битом мы получим такую картину:

h	a
010110010110	001011101

Как мы видим, контрольные биты под номерами: 1, 2, 8 не совпадают с такими же контрольными битами, которые мы получили. Теперь просто сложив номера позиций неправильных контрольных битов (1 + 2 + 8 = 11) мы

получаем позицию ошибочного бита. Теперь просто инвертировав его и отбросив контрольные биты, мы получим исходное сообщение в первозданном виде! Абсолютно аналогично поступаем со второй частью сообщения.