Построение инициализирующих последовательностей синхронных цифровых схем с помощью генетических алгоритмов

Иванов Д.Е., Скобцов Ю.А., Эль-Хатиб А.И.

В настоящее время для решения многих «классических» задач технической диагностики применяются нетрадиционные подходы, в частности - генетические алгоритмы. Одной из таких задач является построение инициализирующих последовательностей для синхронных последовательностных схем. В статье предлагается генетический алгоритм построения таких последовательностей. В данном алгоритме вычисление оценочной функции основано на моделировании работы исправной схемы. Для программы, реализующей предложенный алгоритм, приведены результаты проведённых машинных экспериментов для схем из международного каталога ISCAS-89.

1. Введение

Основной тенденцией производства цифровых устройств в настоящее время является повышение их сложности. Это отражается на всех этапах производственного цикла, включая проектирование и тестирование цифровых устройств (ЦУ). Также в связи с резким скачком сложности проектируемых цифровых устройств применение уже существующих подходов существенно увеличивает стоимость процесса диагностирования, требуя чрезмерных затрат временных ресурсов и памяти. Поэтому требуется поиск новых методов решения традиционных задач в технической диагностике. Следует заметить, что указанные методы, в первую очередь, должны быть направлены на преодоление недостатков традиционных точных методов решения задач, главным из которых является невозможность работы со схемами большой размерности. Это порождает направление в технической диагностике, которое использует приближённые методы решения практических задач. Приближённые методы не всегда дают точный ответ при решении поставленной задачи. Однако они позволяют находить компромисс между приемлемыми в практическом смысле и реальными затратами при решении той или иной задачи, например ресурсов времени или требуемой памяти.

Одно из важнейших свойств синхронных последовательностных схем – их инициируемость, т.е. их способность быть переведёнными из начального неопределённого состояния в некоторое определённое. Такая задача, например, встаёт всякий раз при включении устройства. Часто задача решается введением в аппаратурную часть сигнала «сброс», который подключается ко всем элементам состояния схемы. С другой стороны, разработчики цифровых схем могут обеспечить инициализирующие последовательности. Инициализирующей называется последовательность, которая переводит все элементы состояний последовательностной схемы в некоторые известные значения. В связи с этим, возникает необходимость построения автоматизированного инструментария, который позволяет определить существует ли для заданного устройства инициализирующая последовательность, и в случае положительного ответа строящий такую последовательность.

В данной работе рассматривается логическая инициализируемость. В данном случае требуется, чтобы программа моделирования, работающая в трёхзначном алфавите (0, 1, X) могла перевести схему в некоторое известное состояние при условии, что она начинает работу из полностью неизвестного состояния, когда всем линиям схемы присвоено значение X. Известно также, что не все функционально инициализируемые схемы являются логически инициализируемыми.

Предлагаемый подход к решению проблемы существенно отличается от существующих. Поэтому далее мы очень кратко опишем уже известные подходы к решению задачи.

К точным методам построения инициализирующих последовательностей относятся методы, предложенные в [1]. Они основаны на построении дерева решений и технике символьных операций. Данные методы хорошо зарекомендовали себя и позволяют найти решение для схем малой и средней размерности. Однако они не применимы для больших схем из-за требований ресурсов памяти. Алгоритм данного класса, имеющий те же ограничения, предложен также в [2]

Развитием данных методов является работа [3], в которой предлагается декомпозиция схемы, что упрощает построение заданных последовательностей. В работе [4] исследуется проблема построения инициирующих последовательностей применительно к задаче автоматизированного построения тестов.

Подход, который объединяет метод ветвей и границ и «жадные» алгоритмы, предложен в [5]. Он оказался применим в практическом смысле для широкого класса схем, однако иногда требует чрезмерных ресурсов памяти.

В последнее время в связи с ростом сложности проектируемых цифровых схем получило развитие направление, которое при решении задач технической диагностики использует информацию, полученную на основе моделирования (исправного или с неисправностями) заданной схемы. К данным методам относятся также и генетические алгоритмы [6-7].

В данной статье будет предложен алгоритм построения инициализирующих последовательностей для синхронных последовательностных схем, основанный на генетических алгоритмах.

Статья имеет следующую структуру. Во введении обосновывается актуальность задачи, а также кратко рассматриваются существующие подходы к её решению. В следующем разделе кратко описывается применение генетического алгоритма к решению поставленной задачи. В третьем разделе описываются реализация алгоритма и проведённые машинные эксперименты, обосновывается выбор параметров генетического алгоритма. В заключении сделаны выводы и предложено направление дальнейших исследований.

2. Генетический алгоритм построения инициализирующих последовательностей.

К приближённым методам, успешно применяемым на практике, относятся, в частности, генетические алгоритмы. В широком смысле термин «генетический алгоритм» применяют для обозначения любой основанной на популяциях математической модели, которая использует операции селекции и рекомбинации для построения новой выборки точек в пространстве поиска. Его суть заключается в попытке повторить природный путь улучшения характеристик живых организмов. Генетический алгоритм часто рассматривают как алгоритм поиска максимума и минимума некоторой функции, хотя спектр проблем, для решения которых применяется генетический алгоритм, очень широк.

Авторы уже неоднократно применяли генетический алгоритм к решению задача технической диагностики, в частности, для построения тестовых последовательностей [7]. Там же подробно описана техника работы с входными двоичными последовательностями в генетических алгоритмах. Поэтому здесь мы не будем подробно останавливаться на данной технике, а кратко опишем основные элементы такого алгоритма.

Чтобы задать генетический алгоритм построения инициализирующих последовательностей, необходимо определить следующие понятия: особь, популяция, операции скрещивания и мутации, оценочная функция.

В генетических алгоритмах, работающих с входными двоичными последовательностями, особь представляется как одна такая последовательность, которая состоит из некоторого числа двоичных наборов. Число бит в каждом двоичном наборе (ширина последовательности) соответствует числу входов схемы и является неизменной величиной в алгоритме. Каждый входной набор соответствует одному такту работы схемы, а число таких наборов, входящих в последовательность, определяет длину особи. Эта величина может изменяться в процессе работы алгоритма под воздействием генетических

операторов. В качестве популяции выступает набор особей (входных последовательностей). В нашем алгоритме число особей в популяции (её размер) является величиной неизменной.

Генетические операции применяются к уже имеющимся особям в популяции для получения новых особей, желательно с улучшенными свойствами. Мы применяем два вида операции скрещивания и три вида оператора мутации. Операция скрещивания бывает вертикальная и горизонтальная. При вертикальном скрещивании разрезание особей идёт по входам схемы. Для каждого столбца входной последовательности особи-потомка случайным образом определяется столбец, какого из родителей необходимо скопировать. При горизонтальном скрещивании, которое выполняется по входным наборам (тактам времени) в каждой особи-родителе выбирается одна точка скрещивания. Особь-потомок формируется из начала последовательности одного родителя и конца последовательности второго родителя. Легко видеть, что при вертикальном скрещивании длина особи-потомка равна длине большего из его родителей, тогда как при горизонтальном скрещивании она может как увеличиваться, так и уменьшаться.

Механизм мутации обеспечивает появление новых генов в особях. Он применяется с заранее заданной вероятностью P_m к выходу операции скрещивания. В алгоритме применяется три стандартных для данного типа алгоритмов вида мутации, выбор между которыми происходит случайно:

- мутация одиночного бита (каждый бит в последовательностях-потомках может изменить своё значение с вероятностью 0.5%);
- добавление вектора (в случайную позицию в последовательности вставляется случайным образом сгенерированный вектор, что позволяет вводить и рассматривать тестовые последовательности с большей длиной);
- удаление вектора (из случайно выбранной позиции удаляется вектор, если он был

```
Генетический_алгоритм(РазмерПопуляции, Число итераций)
{
  // подготовка начальной популяции
  ПостроитьНачальнуюПопуляцию();
  ОценитьОсобей(НачальнаяПопуляция);
  НомерПопуляции=0;
  Пока( не_достигнут_критерий_остановки )
  {
    НоваяПозиция=0;
    Для( i=0 ; i<РазмерПопуляции ; i++)
    {
      ОперацияВыбора(РодительА, РодительБ);
      Если( rand() < Pc )
        ОперацияСкрещивания(РодительА, РодительБ, Потомок);
      Если( rand() < Pm )
        ОперацияМутации(Потомок);
      ДобавитьВПромежуточнуюПопуляцию(Потомок, НоваяПозиция);
      НоваяПозиция++;
    }
    ОценитьОсобей(ПромежуточнаяПопуляция);
    ПостроитьНовуюПопуляцию(РазмерПопуляции);
    НомерПопуляции++;
  }
  СоздатьОтчёт();
}
```

Рис.1 Генетический алгоритм.

несущественным, то оценочная функция всей последовательности не должна уменьшится);

Принимая во внимание вышесказанное, разрабатывается псевдокод генетического алгоритма построения инициализирующих последовательностей, который приведён на рис.1.

В приведённом алгоритме требует уточнения процедура *«ОценитьОсобей()»*. В ней вычисляется оценочная функция каждой особи в популяции. Ключевым здесь является момент использования готовых процедур моделирования работы исправных цифровых схем. Свойства последовательностей, которые строятся генетическим алгоритмом, определяются способом построения оценочной функции. Остановимся подробнее на её вычислении. Поскольку наш алгоритм строит инициализирующие последовательности, оценочную функцию необходимо задавать с помощью следующих параметров:

- отношение числа инициированных триггеров к их общему числу n_1; чем больше триггеров схемы перешло из неопределённого состояния в определённое, тем выше качество заданной последовательности;
- активность схемы или число событий моделирования n_2; чем выше число событий при моделировании схемы на заданной последовательности, тем выше вероятность, что ещё неустановившиеся триггеры перейдут в определённое состояние.
- длина последовательности n_3; из двух последовательностей при прочих равных условиях необходимо предпочесть более короткую;

Таким образом, оценочная функция имеет следующий вид:

$$f(s) = f(n_1, n_2 n_3) = (c_1 * n_1 + c_2 * n_2) * c_3^{n3}$$

где n_1, n_2, n_3 – описанные выше параметры, c_1, c_2, c_3 – нормализующие константы, которые имеют следующий смысл:

- c_1 и c_2 уравновешивают влияние на фитнесс-функцию числа триггеров, перешедших в известное состояние, и активность вентилей в схеме; очевидно, что для построения эффективной фитнесс-функции недостаточно одного из параметров n_1 или n_2, поскольку это может завести алгоритм в тупик, например в случае, когда число установленных триггеров не изменяется, а активность схемы близка к нулю;
- c_3 позволяет искать более короткие входные последовательности; если данную константу выбрать близкой к единице c_3<1, то из двух входных последовательностей при прочих равных условиях (число активизированных триггеров и активность схемы) большую оценочную функцию будет иметь та последовательность, длина которой меньше; заметим здесь также, что выбор константы c_3 даже очень близкой к единице существенно сказывается на временных затратах при вычислении оценочной функции, когда необходимо достичь её некоторого фиксированного значения; в нашем случае введение данного компонента фитнесс-функции оправдано, поскольку её вычисление относительно нетрудоёмкая задача, например в сравнении с вычислением фитнесс-функции при

```
Вычисление_оценочной_функции(входная последовательность, длина)
{
  МоделированиеРаботыИсправнойСхемы(входная последовательность)
  n1=ЧислоАктивированыхТриггеров;
  n2= АктивностьСхемыПриМоделировании;
  n3= Длина;
  Фитнесс=(c1*n1+c2*n2)* c3^ Длина;
}
```

Рис. 2 Псевдокод процедуры вычисления оценочной функции.

построении тестов: в нашем алгоритме используется моделирование работы исправной схемы, тогда как в упомянутом алгоритме используется моделирование с неисправностями.

Псевдокод процедуры вычисления оценочной функции одной особи (входной последовательности) приведён на рис.2.

3. Экспериментальные данные.

Проведение машинных экспериментов при построении генетических алгоритмов играет особенную роль: они позволяют принять/отвергнуть ту или иную эвристику, на их основании выбираются значения числовых констант в алгоритме.

При программной реализации и проведении машинных экспериментов необходимо установить оптимальные значения следующих числовых констант:

- начальная длина последовательности L;
- вероятность применения оператора мутации P_m;
- вероятности вертикального и горизонтального скрещивания P_c;
- веса компонентов оценочной функции c_1, c_2, c_3.

Значения следует выбирать таким образом, чтобы алгоритм показывал приемлемые результаты для всех контрольных схем [8].

Предложенный алгоритм был программно реализован в среде программирования C++ Builder и составил около 1200 строк кода. Эксперименты проводились на персональном компьютере с процессором Celeron 1800Мгц, оперативной памятью 512Мб, в операционной системе Windows XP.

Опишем подробнее, как производится выбор числовых параметров алгоритма на примере выбора вероятности оператора мутации P_m. Для проведения экспериментов выбрана частично инициализируемая схема s13207, требующая значительного перебора точек в пространстве поиска для инициализации наибольшего числа триггеров. Также для экспериментов выбраны следующие значения вероятности P_m: 0.05, 0.1, 0.15, 0.2. Для каждого из выбранных значений проводилась серия экспериментов, в которых фиксировался рост фитнесс-функции в зависимости от числа рассмотренных точек в пространстве поиска (здесь и далее мы будем приводить в качестве опорной величины не число поколений, пройденных генетическим алгоритмом, а число рассмотренных точек в пространстве поиска, что, на наш взгляд, даёт более точное представление о ситуации). Число поколений ограничивалось 30-ю, что соответствует перебору 14000 точек в пространстве поиска. Данные усреднялись для каждой схемы. Диаграммы, построенные на основании результатов данных экспериментов, приведены на рис.3-6. Видно, что лучший рост фитнесс-функции в начале и середине графиков обеспечивался при значениях P_m=0.1, P_m=0.15 и P_m=0.2. Однако точные данные (табл.1) показывают, что наибольшего итогового значения удаётся достигнуть при P_m=0.05 и P_m=0.1. Таким образом, в качестве значения вероятности оператора мутации выбрано значение 0.1.

Табл. 1 Средний рост фитнесс-функции для схемы s13207 в зависимости от числа рассмотренных точек в пространстве поиска.

	Число рассмотренных точек в пространстве поиска									
	500	2000	3500	5000	6500	8000	9500	11000	12500	14000
P_m=0.05	0,0275	0,0562	0,0712	0,1462	0,2206	0,2592	0,2933	0,3013	0,3046	0,3055
P_m=0.1	0,0233	0,0646	0,1034	0,1871	0,2487	0,2819	0,2960	0,3019	0,3034	0,3055
P_m=0.15	0,0254	0,0643	0,1429	0,2015	0,2568	0,2881	0,2978	0,3019	0,3028	0,3046
P_m=0.3	0,0305	0,0649	0,1570	0,2114	0,2712	0,2891	0,2993	0,3007	0,3037	0,3043

Рис.3 Рост фитнесс-функции. P_m=0.05.

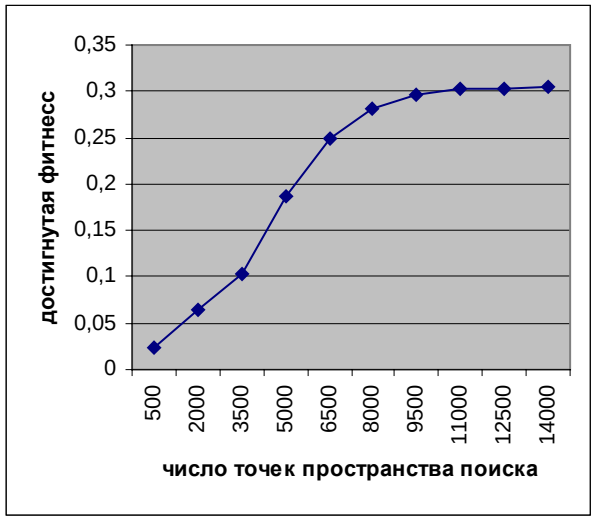

Рис.4 Рост фитнесс-функции. P_m=0.1.

Рис.5 Рост фитнесс-функции. P_m=0.15.

Рис.6 Рост фитнесс-функции. P_m=0.2.

Результаты машинных экспериментов позволили выбрать следующие значения констант алгоритма:

- P_c=0.96, P_m=0.01;
- c_1=1, а константа c_2 равна отношению числа триггеров к числу вентилей схемы, что позволяет сбалансировано учитывать активность схемы при моделировании на заданной входной последовательности,
- c_3=0.99, уменьшение данной константы позволяет искать более короткие инициализирующие последовательности, однако при этом существенно увеличивается глубина поиска, а следовательно затрачиваемое на поиск время;
- L=1, поскольку многие схемы удаётся инициализировать даже последовательностями с одним входным вектором.

С приведёнными параметрами алгоритм запускался для всех схем из каталога ISCAS-89. Результаты проведённых экспериментов приведены в табл.2. Они показывают, что все схемы, входящие в каталог либо очень легко инициализируются, либо, наоборот, построить для них последовательность практически невозможно даже при очень глубоком поиске.

Табл. 2 Результаты машинных экспериментов для схем ISCAS-89.

Имя схемы	Число триггеров в схеме	Число инициированных триггеров	Длина инициализирующей последовательности	Время работы генератора, сек	Рассмотрено вариантов при поиске
s344	15	15	2	0	1000
s349	15	15	2	0	1000
s382	21	21	1	0	1000
s386	6	6	2	0	1000
s400	21	21	1	0	1000
s444	21	21	1	0	1000
s499	22	0	-	4	5500
s510	22	0	-	4	5500
s526	21	21	2	0	1000
s526n	21	21	2	0	1000
s635	32	12	1	5	6500
s641	19	19	1	0	1000
s713	19	19	1	0	1000
s820	5	5	1	0	1000
s832	5	5	1	0	1000
s838.1	32	0	-	5	5500
s938	32	0	-	4	5500
s953	29	10	1	5	6500
s967	29	10	1	5	6500
s991	19	0	-	4	5500
s1196	18	18	1	0	1000
s1238	18	18	1	1	1000
s1423	74	74	3	2	2500
s1488	6	6	1	0	1000
s1494	6	6	1	0	1000
s1512	57	53	2	8	7500
s3271	116	116	14	11	6000
s3330	131	131	3	6	4000
s3384	183	183	8	7	5000
s4863	104	104	2	3	2000
s5378	179	179	40	61	11500
s6669	239	239	5	19	7000
s9234	228	53	2	16	6500
s13207	669	207	42	191	15500
s15850	597	308	20	207	15000
s35932	1728	1728	1	8	1000
s38417	1636	372	15	459	12500
s38584	1452	1398	36	791	15500

4. Выводы

В работе предложен генетический алгоритм построения инициализирующих последовательностей для синхронных последовательностных схем. Рассмотрена постановка задачи, кратко описаны основные детерминированные подходы к решению данной задачи. На основе разработанного авторами ранее алгоритма генерации тестовых последовательностей предложен генетический алгоритм построения инициализирующих последовательностей для синхронных последовательностных схем. Для программной реализации предложенного алгоритма приведены результаты машинных экспериментов.

Также на основании этих экспериментов обоснован выбор параметров генетического алгоритма. В качестве дальнейшего направления исследования можно отметить изучение возможности применения генетического алгоритма для построения последовательностей, переводящих схему из одного заданного состояния в другое, что необходимо в структурных методах генерации тестов.

Литература.

1. C. Pixley, S. Jeong, G. Hatchel, “Exact Calculation of Synchronization Sequences based on Binary Decision Diagrams”, IEEE Trans. on CAD, Vol.13, pp.1024-1034, 1994.
2. J.-K. Rho, F. Somenzi, C. Pixley, “Minimum Length Synchronizing Sequences of Finite State Mashine”, Proc. ACM Design Automation Conf., 1993, pp.463-468.
3. J.A. Wehbeh, D.G. Saab, “On the Initialization of Sequential Circuits,” Proc. IEEE Int. Test Conf., 1994, pp. 233-239.
4. J.A. Wehbeh, D.G. Saab, “Initialization of Sequential Circuits and its Application to ATPG,”, Proc. IEEE VLSI Test Symp., 1996, pp. 246-251.
5. M. Keim, B. Becker, B. Stenner, “On the (Non-)Resettability of Synchronous Sequential Circuits,” Proc. IEEE VLSI Test Symp., 1996, pp. 240-245.
6. P. Prinetto, M. Rebaudengo, M. Sonza Reorda, "An Automatic Test Pattern Generator for Large Sequential Circuits based on Genetic Algorithms" In Proc. Int. Test Conf., 1994, pp. 240-249.
7. Иванов Д.Е., Скобцов Ю.А. Генерация тестов цифровых устройств с использованием генетических алгоритмов // Труды института прикладной математики и механики НАН Украины. – Т.4. – Донецк, ИПММ. – 1999. – С.82-88.
8. Brgles F., Bryan D., Kozminski K. Combinational profiles of sequential benchmark circuits // International symposium of circuits and systems, ISCAS-89. – 1989. – P.1929-1934.