УДК 004.315.7

Панцырев Роман Игоревич, магистрант, ФГБОУ ВО «Казанский Национальный Исследовательский Технический Университет Им. А.Н. Туполева-Каи», г. Казань

АППАРАТНАЯ РЕАЛИЗАЦИЯ ИДЕНТИФИКАЦИИ ЦЕПЕЙ МАРКОВА МОДИФИЦИРОВАННЫМ МЕТОДОМ РАБИНЕРА

Аннотация. В работе исследуется задача аппаратной реализации алгоритма идентификации скрытых цепей Маркова на основе модифицированного метода Рабинера, где особое внимание уделено разработке высокоэффективного вычислительного ядра, оптимизированного для программируемых логических интегральных схем. Проведен детальный анализ вычислительных особенностей классического алгоритма Рабинера, что позволило разработать ключевые модификации, включая переход к арифметике фиксированной точки с сохранением необходимой точности расчетов. Разработана специализированная архитектура вычислительного модуля, учитывающая специфику параллельной обработки данных в ПЛИС.

Annotation. The paper examines the problem of hardware implementation of the hidden Markov circuit identification algorithm based on the modified Rabiner method, where special attention is paid to the development of a highly efficient computing core optimized for programmable logic integrated circuits. A detailed analysis of the computational features of the classical Rabiner algorithm has been carried out, which made it possible to develop key modifications, including the transition to fixed-point arithmetic while maintaining the necessary calculation accuracy. A specialized architecture of the computing module has been developed, taking into account the specifics of parallel data processing in FPGAs.

Ключевые слова: цепи Маркова, идентификация, метод Рабинера, ПЛИС/FPGA.

Keywords: Markov chains, identification, Rabiner method, FPGA.

ВВЕДЕНИЕ

В теоретических и прикладных исследованиях в различных областях науки и техники, в частности, в области анализа состояния экономических объектов и систем важную роль играет задача идентификации автоматных марковских моделей (АММ). Задача основана на сборе последовательных измерений выходных параметров объекта, при этом наблюдаемая последовательность рассматривается как случайный процесс, описывающий изменения состояний системы в дискретные моменты времени, и расчете вероятности отнесения объекта к тому или иному априори заданному классу. Один из методов идентификации был предложен Л.Р. Рабинером в рамках решения задачи оценивания вероятности некоторой последовательности наблюдений, определяемой заданной скрытой марковской моделью, наилучшим образом соответствующей наблюдаемому измерению. В теоретических, прикладных исследованиях вопросы синтеза и анализа дискретных марковских процессов являются актуальными и на сегодняшний день [1, 2]. В работе рассматриваются алгоритмы, которые применимы к решению задачи идентификации автоматной марковской модели. Реализация модифицированного алгоритма на ПЛИС.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Конечные простые однородные Цепи Маркова (ЦМ) заданы на основе выражения [3]:

$$(S, P_{(m)}, \pi_0), \quad (1)$$

где S □ □s1 ,s2 ,...,sm □ - множество ее состояний, P(m) – стохастическая матрица (СМ) размера m×m, $\pi_0=\{\pi_i\}$, $i=\overline{1,m}$ –вектор, задающий начальное распределение вероятностей состояний. Автоматной марковской моделью (АММ) будем называть автономный вероятностный автомат вида [4, 5]:

$$A = (\hat{\mu}, S, \delta(x, s)) \quad (2)$$

Задание АММ в виде (2) эквивалентно заданию конечной простой однородной ЦМ вида (1): S - тот же объект, что и в (1), $\hat{\mu}$ – дискретная случайная величина, принимающая конечное число значений $\mu_1,\mu_2,\ldots,\mu_l$ на входе A с вероятностями $p_1,p_2,\ldots,p_l$, $\sum_{i=1}^{l} p_i = 1, 0 \le p_i \le l$, δ(x,s) – функция переходов, ставящая в соответствие паре (x,s) однозначно новое состояние $s` \in S$ и для которой множество значений $\{x_i\} = X = \{\mu_i\}$.

Последовательность состояний АММ вида (2) является простой однородной ЦМ вида (1), определяемой стохастической матрицей P, которая вычисляется по формуле [4]

$$P = \sum_{k=1}^{l} p_k M(x_k),$$

где коэффициенты p1, p2,…, pl образуют стохастический вектор $\bar{P}$, $M(x_k)$ - простая матрица, соответствующая букве x_k. Элементы матрицы $M(x_k)$, обозначим их через □ $\pi_{ij}(x_k)$, $i,j=\overline{1,m}$, определяются из соотношения [4]

$$\pi_{ij}(x_k) = \begin{cases} 1: \ \delta(x_k, s_i) = s_j \\ 0: \ \delta(x_k, s_i) \neq s_j, i, j = \overline{1,m} \end{cases}$$

Пусть задан класс AMM(P), обозначим g , g=$\overline{1,k}$, и последовательность наблюдений $\hat{S}(N) = s(1)s(2)\dots s(N)$. Требуется вычислить вероятность того, что реализации ЦМ порождены заданной автоматной марковской моделью.

Схема модели идентификации марковских последовательностей приведена на рис.1

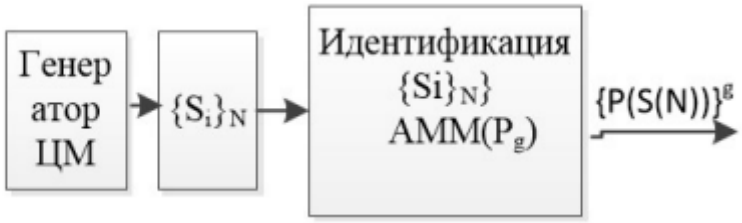

Рис. 1 – Схема модели идентификации ЦМ

В соответствии с заданной схемой идентификации последовательности, порождаемой AMM, заданной эргодический стохастической матрицей (ЭСМ) P, где P принадлежит заданному подклассу стохастических эргодических матриц, ставится следующая задача:

Вычислить P($\hat{S}(N)$|AMM(P)) - вероятность того, что $\hat{S}(N)$ сгенерирована на основе AMM(P), где ЭСМ P принадлежит заданному подклассу.

В [6] предложен алгоритм прямого-обратного хода, который применим для идентификации конечных простых однородных ЦМ, сгенерированных на основе стохастических матриц класса эргодических. Данный подход также описан в работе, как модификация модели, предложенной в [6]. Кроме того, в статье показано, что предложенный модифицированный алгоритм прямого-обратного хода позволяет решать задачу идентификации ЦМ, часть элементов которой скрыта от наблюдения. В [7] предложена реализация модифицированного метода Рабинера на нейронных сумматорах. В [8] рассматривается задача идентификации цепей Маркова и проанализирована вычислительная сложность указанных выше алгоритмов идентификации конечных простых однородных ЦМ, сгенерированных на основе AMM.

В [9] предложена модификация метода «прямого-обратного хода» для решения задачи определения вероятности того, что заданная последовательность порождаема автоматной марковской моделью.

Модифицированный алгоритм включает 3 этапа:

1) На этапе инициализации переменных вычисляем

$$a_1(i) = \pi_0(i) * z_i, i = \overline{1,m}\ , z_i = \begin{cases} 1: s(t+1) = s_j \\ 0: \text{иначе} \end{cases}$$

2) Для этапа индукции полагаем, что

$$a_{t+1}(j) = [\sum_{i-1}^{m} a_t(i) * p_{ij}] * z_j, t = \overline{1,N-1}\ , j = \overline{1,m}$$

3) Находим

$$\mathrm{P(S(N)|AMM(P))} = \mathrm{a_N(s(N))}$$

РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА

На этапе инициализации модифицированного метода Рабинера требуется операция:

$$a_1(i) = \pi_0(i) * z_i, i = \overline{1,m}\ , z_i = \begin{cases} 1: s(t+1) = s_j \\ 0: \textit{иначе} \end{cases}$$

Рассмотрим часть vhdl кода, выполняющего данную операцию:

Листинг 2.1. Этап инициализации

```
if counter = 0 then
z <= (others => 0);
if S_next(counter) = HIDDEN_STATE then
 z_temp := (others => 1);
else
 z_temp := (others => 0);
 z_temp(S_next(counter)) := 1;
end if;
z <= z_temp;
for i in 0 to MAX_STATES - 1 loop
 if i < actual_N then
  if z_temp(i) = 1 then
   a(i) <= p0(i);
  else
   a(i) <= (others => '0');
  end if;
 end if;
end loop;
counter <= counter + 1;
```

В этой части кода реализуется этап инициализации алгоритма Рабинера для первого элемента последовательности наблюдений. На нулевом шаге счетчика происходит подготовка начальных значений для дальнейших вычислений. Сначала вектор z обнуляется, после чего в зависимости от типа текущего состояния формируется временный вектор z_temp. Если состояние скрытое (HIDDEN_STATE), все элементы z_temp устанавливаются в единицу, что означает учет всех возможных состояний. Для наблюдаемого состояния z_temp обнуляется, а затем устанавливается в единицу только позиция, соответствующая конкретному наблюдаемому состоянию. Полученный z_temp сохраняется в z для использования на следующих шагах. Далее в цикле по всем возможным состояниям системы инициализируется вектор прямой переменной a(i). Если соответствующий элемент z_temp равен единице, то a(i) принимает значение из начального распределения вероятностей p0(i), в противном случае обнуляется. Это соответствует классической формуле инициализации прямой переменной $a_1(i) = \pi_i b_i(O1)$, где в данном случае $b_i(O1)$ учитывается через маскирующий вектор z_temp. После выполнения этих операций счетчик увеличивается на единицу, что позволяет перейти к обработке следующего этапа. Таким образом, данный фрагмент кода подготавливает начальные условия для работы основного алгоритма прямого-обратного хода, обеспечивая корректный учет как скрытых, так и наблюдаемых состояний на первом шаге вычислений.

Листинг 2.2. Этап индукции

```
elsif counter < actual_T then
 z <= (others => 0);
 if S_next(counter) = HIDDEN_STATE then
  z <= (others => 1);
 else
  z(S_next(counter)) <= 1;
 end if;
 for j in 0 to MAX_STATES - 1 loop
  if j < actual_N then
   temp_sum := (others => '0');
    for i in 0 to MAX_STATES - 1 loop
     if i < actual_N then
      temp_sum := temp_sum + a(i) * pij(i, j);
     end if;
    end loop;
    if S_next(counter) = HIDDEN_STATE then
     a(j) <= temp_sum(36 downto 18);
```

```
    else
     if j = S_next(counter) then
      a(j) <= temp_sum(36 downto 18);
     else
      a(j) <= (others => '0');
     end if;
    end if;
   end if;
  end loop;
```

Этот фрагмент кода реализует основной индуктивный этап алгоритма Рабинера, который выполняется для всех последующих элементов последовательности наблюдений после первого. На данном этапе происходит вычисление прямой переменной a(j) для каждого возможного состояния системы. Процесс начинается с обновления маскирующего вектора z: если текущее состояние скрытое (HIDDEN_STATE), все элементы z устанавливаются в 1, что означает учет всех возможных переходов; для наблюдаемого состояния в z устанавливается 1 только в позиции соответствующего наблюдаемого состояния.

Далее выполняется двойной цикл для расчета промежуточной суммы temp_sum, где внутренний цикл реализует ключевую операцию алгоритма - суммирование произведений предыдущих значений прямой переменной a(i) на соответствующие вероятности переходов pij(i,j). Эта операция соответствует формуле $a_{t+1}(j) = \left[\sum_{i=1}^{N} a_t(i) p_{ij}\right] * z_j$. Особенностью данной реализации является обработка скрытых и наблюдаемых состояний: для скрытых состояний результат temp_sum масштабируется и сохраняется для всех j, тогда как для наблюдаемых состояний ненулевое значение сохраняется только для j, совпадающего с текущим наблюдаемым состоянием. Масштабирование (36 downto 18) выполняется для приведения формата числа, что связано с особенностями представления чисел с фиксированной точкой в данной реализации. Завершается этап инкрементом счетчика, что обеспечивает переход к обработке финального этапа нахождения вероятности. Этот участок кода является вычислительным ядром прямого прохода алгоритма, где происходит аккумуляция вероятностей путей через СММ с учетом как наблюдаемых, так и скрытых состояний.

Листинг 2.3. Этап финализации

```
temp_sum := (others => '0');
for i in 0 to MAX_STATES - 1 loop
 if i < actual_N then
  temp_sum := temp_sum + a(i);
 end if;
end loop;
P_out <= temp_sum(18 downto 0);
ready <= '1';
computation_done <= '1';
```

В этой завершающей части кода выполняется финальный этап алгоритма Рабинера, где вычисляется итоговая вероятность наблюдения всей последовательности. После обработки всех элементов входной последовательности происходит суммирование значений прямой переменной `a(i)` по всем возможным состояниям модели (от 0 до N-1), что соответствует формуле полной вероятности $P(O|\lambda) = \sum_{i=1}^{N} a_T(i)$. Накопление суммы осуществляется в переменной `temp_sum`, которая изначально обнуляется.

Особенностью данной реализации является работа с числами в формате фиксированной точки - полученная сумма `temp_sum` усекается до 19 бит (диапазон 18 downto 0) перед записью в выходной регистр `P_out`. Это масштабирование необходимо для согласования разрядности и формата данных в системе. После завершения вычислений устанавливаются два флага: `ready` сигнализирует о готовности результата, а `computation_done` отмечает окончание всех вычислений. Таким образом, этот участок кода представляет собой завершающую фазу алгоритма, где определяется итоговая вероятность сгенерированной последовательности наблюдений для заданной модели, после чего система переходит в состояние готовности для обработки новых данных.

ЗАКЛЮЧЕНИЕ

В данной работе был разработан и успешно реализован программно-аппаратный комплекс на базе ПЛИС/FPGA, представляющий собой высокоэффективное решение для анализа марковских последовательностей с использованием модифицированного алгоритма Рабинера.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шалагин, С.В. Метод разложения стохастических матриц для синтеза конвейерных генераторов дискретных марквоских процессов /, С.В. Шалагин // Вестник технологического университета. – 2015. – № 10. – С. 160–162.

2. Эминов, Б.Ф. Об ассимптотических свойствах укрупняемых и укрупненных цепей Маркова / Б.Ф.Эминов, В.М. Захаров // Вестник технологического университета. – 2015. – № 10. – С. 167–173.

3. Кемени, Дж. Конечные цепи Маркова / Дж. Кемени, Дж. Снелл. – М.: Наука, 1970. – 272 с.

4. Поспелов, Д.А. Вероятностные автоматы / Д.А. Поспелов. – М.: Энергия, 1970. – 88 с.

5. Бухараев, Р.Г. Основы теории вероятностных автоматов/ Р.Г. Бухараев. – М.: Наука, 1985. – 287 с.

6. Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proc. IEEE, v. 77, N 2, Febr. 1989. – pp. 257-286.

7. Реализация модифицированного метода Рабинера для множества стохастических матриц на нейронных сумматорах / С. В. Шалагин, А. Р. Нурутдинова // Интеллектуальные системы. Теория и приложения. – 2022. – Т. 26. – № 1. – С. 129-132. – EDN GVESOW.(ВАК)

8. Сравнительный анализ вычислительной сложности алгоритмов идентификации конечных простых однородных цепей Маркова/ Шалагин С.В., Нурутдинова А.Р. // Вестник Казанского технологического университета. - 2016. - N 13. - С.153-156

9. Нурутдинова А.Р. Модификация модели и метода «прямого-обратного хода» для идентификации автоматных марковских моделей // Учен. зап. Казан. ун-та. Сер. Физ.-матем. науки. – 2018. – Т. 160, кн. 3. – С. 578–58