УДК 004.383.3

РАЗРАБОТКА МОДЕЛИ СЛОЖНОЙ НЕЙРОПРОЦЕССОРНОЙ СИСТЕМЫ

Романчук В.А., к.т.н., ст.преп. Рязанского государственного университета имени С.А.Есенина, e-mail: virom11@yandex.ru, тел.: +7 (920) 635-11-56

Ручкин В.Н., д.т.н., профессор Рязанского государственного университета имени С.А.Есенина

Фулин В.А., ст.преп. Рязанского государственного университета имени С.А.Есенина

В статье рассматриваются вопросы моделирования нейропроцессора семейства NM640x, разрабатываемых НТЦ «Модуль», а также вопросы моделирования сложных вычислительных систем на базе нейропроцессоров. Предложены графические представления моделей скалярного и векторного блока нейропроцессора, представленного в виде высокопараллельной системы. Предложена модель вычислительной системы на базе нейропроцессора в виде конечного автомата, показаны аналитические выражения результатов моделирования. Показаны результаты практических исследований.

Ключевые слова: нейропроцессор, моделирование, вычислительная система

Введение

В настоящее время для процессоров наступил так называемый ”технологический предел”, означающий что они достигли максимального уровня повышения быстродействия. Все разработки в данное время направлены на повышение числа процессоров на кристалле. Одним из выходов из данной ситуации является новая элементная база, например, использование нейрокомпьютеров [1].

Но для дальнейшего развития в области нейропроцессорных технологий существует ряд проблем, основными из которых являются: небольшая частота нейрочипов (30-300 МГц) и отсутствие программного обеспечения для нейропроцессоров.

Одним из способов решения перечисленных проблем является организация многопроцессорных систем. В настоящее время в области нейропроцессорных технологий ведутся исследования в части многопроцессорности, уже разработаны модули, включающие несколько процессоров с различными связями: плата МЦ4.13 (мезонин), МЦ9.01, разработанные в НТЦ ”Модуль”; вычислительные модули SMT302, SMT344, SMT313, SMT315, SMT316 на базе 1,2 и 4 модулей семейства TMS320C4x.

Но, имеются проблемы, мешающие созданию эффективных микропроцессорных структур на базе нейропроцессоров:

1. Нейропроцессоры являются пока дорогим и штучным товаром и не каждая организация может их приобрести в нужном количестве. Кроме того необходимо отметить что для реализации какой либо задачи необходимы эксперименты с различным количество процессорных модулей, что также может позволить себе лишь крупная организация.

2. Проектирование и анализ специализированных многопроцессорных систем на базе нейропроцессоров является очень трудоемким и сложным процессом, так как, в отличие от обычных процессоров, для нейропроцессоров нет необходимой теории, методов и алгоритмов и программных средств моделирования и анализа.

Цель

В данной работе, выполняемой в рамках гранта РФФИ №12-07-97516 р-центр-а «Моделирование вычислительных систем на базе нейропроцессоров», была поставлена задача разработки моделей нейропроцессоров и вычислительных систем на базе нейропроцессора - нейропроцессорных систем (НПС).

Теоретические исследования

Разработка модели нейропроцессора семейства NM640x

Составим список величин, от которых зависит поведение объекта или ход процесса, а также тех величин, которые желательно получить в результате моделирования. Обозначим входные сигналы через $X = \{x_1, x_2, ..., x_n\}$; выходные через $Y = \{y_1, y_2, ..., y_n\}$. Поведение объекта или процесса можно представить в виде: $Y = F(x_1, x_2, ..., x_n)$, где F - те действия, которые следует произвести над входными параметрами, чтобы получить результаты.

Объект моделирования можно представить в виде ”черного ящика”, преобразующего входные сигналы в выходные. В данном случае размерности вектора X и вектора Y совпадают.

Входными данными будут являться начальные значения, регистров, памяти, стека и других элементов процессора $X_d = \{x_{d1}, x_{d2}, ..., x_{dn}\}$; команда процессора X_c

Выходными данными будут значения регистров, памяти и других элементов после выполнения операции и/или ошибки процессора $Y_d = \{y_{d1}, y_{d2}, ..., y_{dn}\}$.

Разрабатывается статическая модель, так как выходные данные появляются на выходе модели всегда за определенное количество времени, зависящее от команды. Данными о том, какие состояния имеет система в тот или иной момент времени являются избыточными.

Проведем анализ процессора для выделения блоков модели и более точного моделирования. В процессоре семейства NM640x возможно два варианта команд: скалярные команды – используются для подготовки данных к выполнению операции на векторном процессоре и векторные команды – основные операции процессора (рисунок 1) [2].

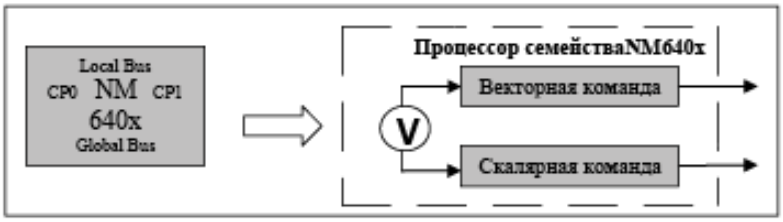


Рисунок 1 - Команды нейропроцессора семейства NM640x

Каждая из команд может выполнять несколько операций параллельно. Тогда каждый процессорный модуль (ПМ) можно представить в виде системы параллельной обработки данных.

Скалярная команда нейропроцессора семейства NM640x имеет левую и правую часть, операции каждой части выполняются параллельно. Возможна пустая (nul) команда в левой или правой части.

Детализированная схема выполнения скалярной операции представлена на рисунке 2.

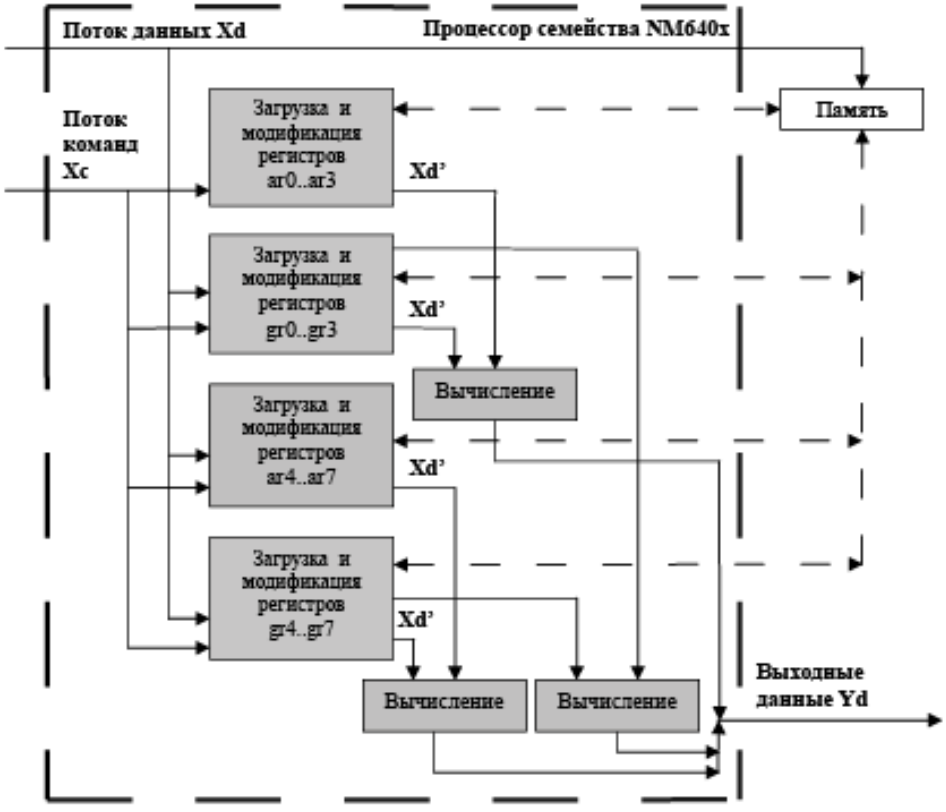


Рисунок 2 - Детализированная схема выполнения скалярной операции

Команды для реализации блоков загрузки и модификации находятся в левой части команды. Блоки загрузки и модификации для различных регистров можно объединить в один или 2 отдельных модуля: загрузки и модификации, т.к. операция выполняется только для какого либо одного регистра (пары регистров). Блоки вычисления находятся в правой части команды и также могут быть объединены в один модуль. Тогда, исходя из приведенных выше схем, можно представить процессор при выполнении скалярной операции в виде параллельной системы (рисунок 4).

Векторная команда нейропроцессора семейства NM640x также имеет левую и правую часть, возможна пустая команда в левой или правой части (рисунок 3).

Левая часть содержит команды: команды загрузки данных в векторный процессор; команды выгрузки данных из векторного процессора; специальные векторные команды.

Правая часть содержит команды: взвешенное суммирование (матричное умножение); маскирование; арифметические операции; логические операции; операция циклического сдвига; операции активации операндов; выгрузка управляющих векторных регистров.

Рисунок 3 – Выполнение скалярной команды процессора, представленное в виде параллельной системы

Кроме этого, нейропроцессор семейства NM640x работает с ”упакованными” данными – блоками по 64 бита каждый. Таким образом, в правой части за 1 такт может выполняться 2 операции по 32 бита, 8 операций по 8 бит и т.д. Максимальное количество выполняемых за один такт операций в векторном процессоре в правой части команды составляет 64 однобитных операции [2].

Сложность состоит в том, что трудно оценить число операций в той или иной команде, следовательно, необходимо рассматривать каждую операцию отдельно. Кроме того, операции имеют разное время выполнения.

Детализируем выполнение левой части команды (рисунок 4).

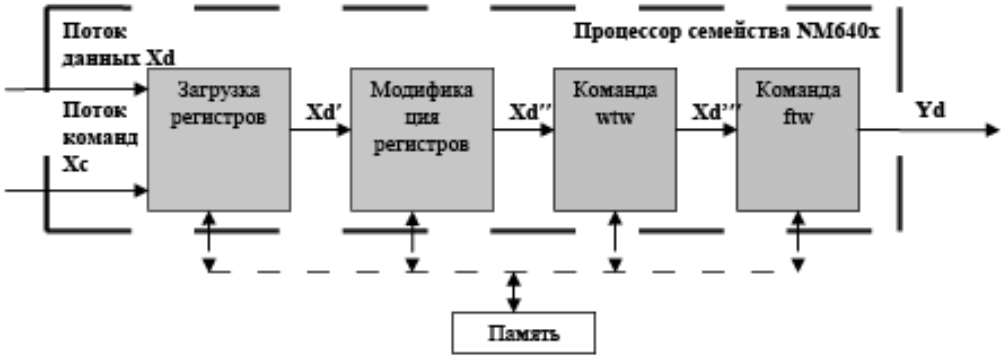


Рисунок 4 - Детализированная схема выполнения левой части векторной команды

Детализируем выполнение правой части команды (рисунок 5).

Исходя из приведенных схем, ПМ при выполнении векторной команды, можно представить в виде параллельной системы (рисунок 6).

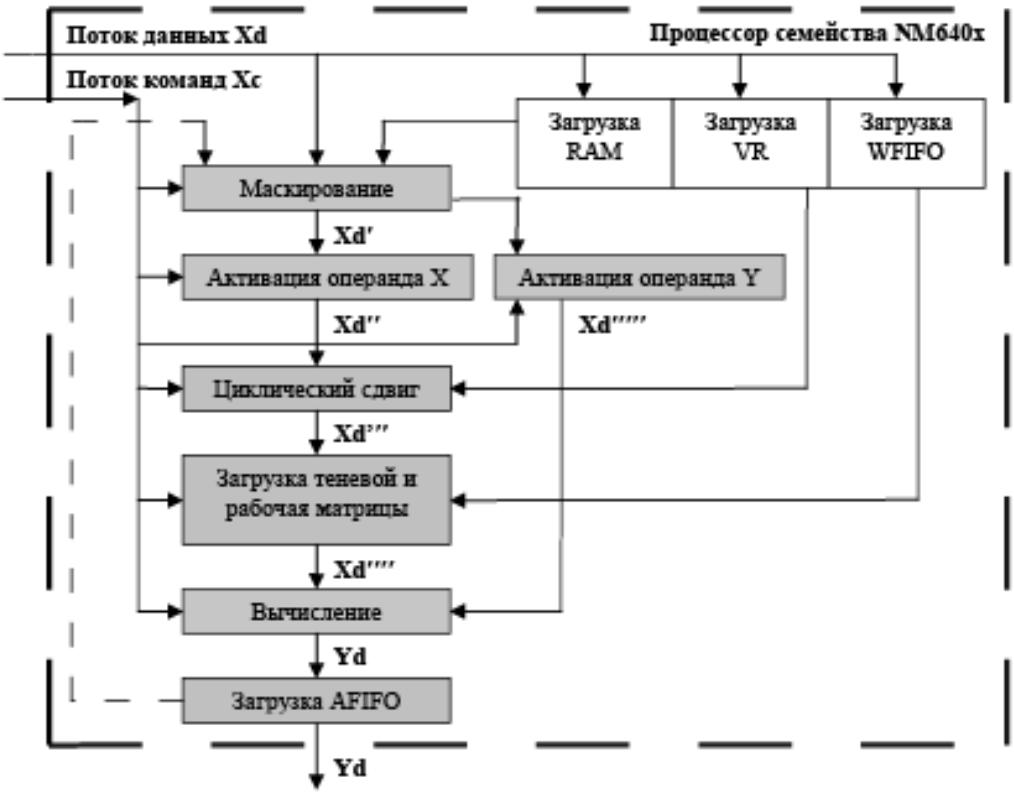


Рисунок 5 - Детализированная схема выполнения правой части векторной команды

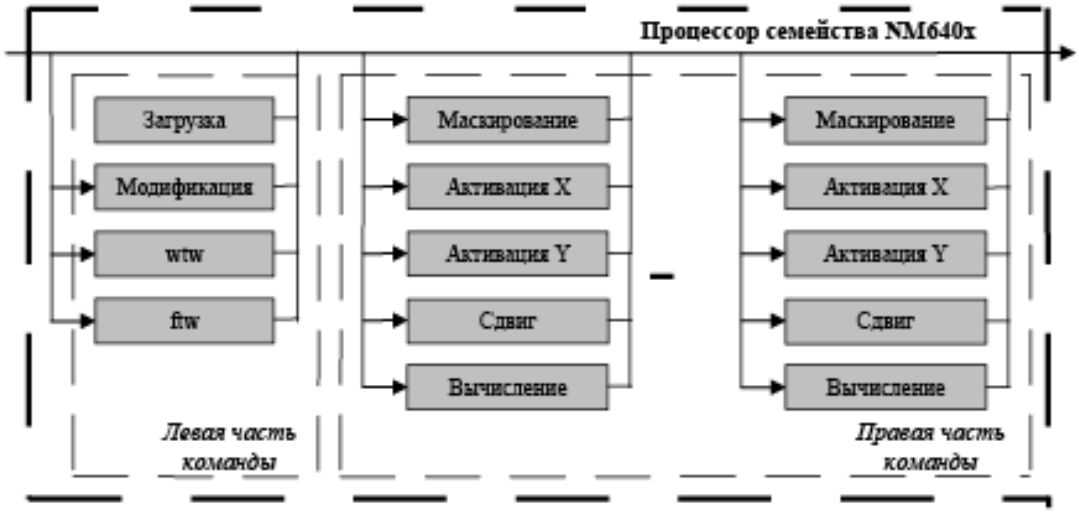


Рисунок 6 – Выполнение векторной команды процессора, представленного в виде параллельной системы

Разработка модели нейропроцессорной системы

В работе рассматривалось несколько видов структур нейропроцессорных систем: конвейерного типа, векторного типа, конвейерно-векторного типа и векторно-конвейерного типа, произвольная структура.

Для построения модели функционирования НПС будем использовать матрицу загруженности процессора: $MZ = \lfloor MZ_{ij} \rfloor$, где i - номер такта, j - номер процессора, MZ_{ij} - порядковый номер обрабатываемого потока.

Используя данную матрицу, для любого такта и периода времени можно определить какие ПМ простаивают (номер обрабатываемого потока равен 0) и какие ПМ обрабатывают данные.

На практике возможны ситуации, связанные с ошибками синхронизации кода. Такие ошибки можно отследить по матрице загруженности:

1. $MZ_{ij} < MZ_{ij+1}$. Данная ситуация означает, что ПМ №i еще не закончил обрабатывать поток данных, а ПМ №$i+1$ уже закончил обрабатывать предыдущий поток данных. В таком случае, если не предусмотрено исправление ошибки синхронизации, ПМ №$i+1$ выйдет из системы из-за неполучения данных для обработки.

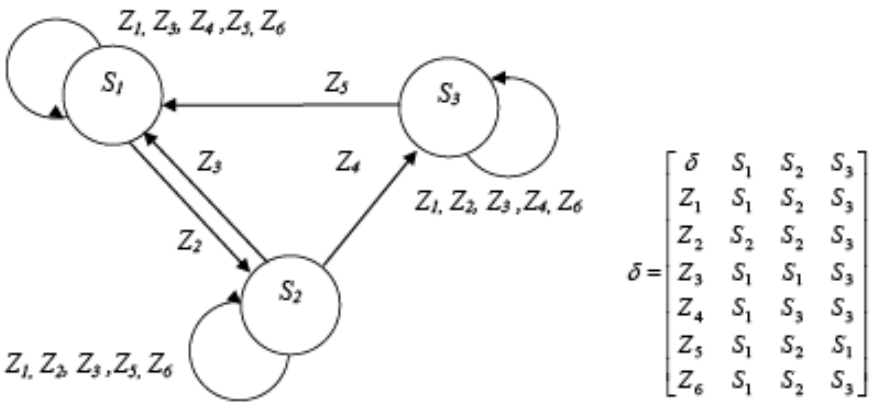
2. $MZ_{ij} - MZ_{ij+1} \geq 2$. Данная ситуация означает, что текущий ПМ №i не дождался пока ПМ №$i+1$ примет данные и начал обработку следующего потока данных. В этом случае данные следующего потока теряются, на следующем шаге обрабатывается поток, являющийся третьим после текущего.

В итоге имеем матрицу $MZ_{q \times l}$, где l - время работы системы в тактах. Также имеем кортеж $TO = <TO_1, TO_2, ..., TO_k>$, где TO_i - время обработки данных i-м процессором. Этот массив заполняется в результате функционирования НПС и расчета времени обработки одного потока данных на i-м процессоре. Пусть кортеж $T_0 = <T_{01}, T_{02}, ..., T_{0k}>$ содержит число нулевых элементов матрицы для каждого ПМ, а $T_1 = <T_{11}, T_{12}, ..., T_{1k}>$ - число ненулевых элементов матрицы для каждого ПМ. Тогда можно получить оценки системы на каждом шаге.

Рассмотрим выходные данные для некоторого i-го процессора: число микрокоманд в подпрограмме: TO_i, время проигрыша процессора: T_{nsvi}, время простоев процессора: T_{npsvi}, время обработки процессора: T_{psvi}, время обработки процессора НПС – число тактов, которые процессор работает при обработке программы: $T_{pi} = T_{1i}$, время простоев процессора НПС – число тактов, которое процессор простаивает в ожидании данных: $T_{npi} = T_{oi}$, коэффициент загруженности - коэффициент, показывающий загрузку процессора относительно времени работы системы: $K_{Ri} = \frac{TO_i - T_{oi}}{TO_i}$, пустые левые и правые части микрокоманды: T_{Lefti}, T_{Righti}, простои декодирования, выполнения, перехода, загрузки: T_{VPi}, время передачи данных: T_{ei}.

Рассмотрим выходные данные для системы: время работы системы – число тактов обработки программы: $T_o = l$, время работы на одном процессоре - рассчитывается путем реализации программы PR на системе, состоящей из одного процессора: T_{1o}, цикл конвейера НПС: $T_c = \max TO_l, l = \overline{1,k}$, количество используемых процессоров: $P = k$, время выигрыша: $T_в = T_{ps} - \sum_{i=1}^{k} TO_i$, время проигрыша для системы: $T_{ns} = l - \sum_{i=1}^{k} TO_i$, время простоя для системы: $T_{nps} = \sum_{i=1}^{k} T_{0i} = \sum_{i=1}^{k} (TO_i - T_{1i})$, время обработки для системы - суммарное число тактов, за которое каждый процессор обрабатывает данные: $T_{ps} = \sum_{i=1}^{k} T_{1i} = \sum_{i=1}^{k} (TO_i - T_{oi})$, суммарное время проигрыша для ПМ: $T_{nsv} = \sum_{i=1}^{k} T_{nsvi}$, суммарное время простоев для ПМ: $T_{npsv} = \sum_{i=1}^{k} T_{npsvi}$, суммарное время обработки для ПМ: $T_{psv} = \sum_{i=1}^{k} T_{psvi}$, коэффициент производительности – коэффициент, показывающий насколько выгодно использовать систему, а не один процессор $K_E = \frac{T_{ps}}{T_{1o}}$, коэффициент загруженности системы – коэффициент, показывающий процент загрузки системы $K_R = \frac{T_o * k - T_{nps}}{T_o * k}$.

Для получения матрицы MZ были разработаны специальные алгоритмы с использованием математического аппарата конечных автоматов. Конечный автомат $KNPS = (S, Z, W, \delta, \lambda, S_1)$ функционирования НПС показан на рисунке 7.



$$\delta = \begin{bmatrix} \delta & S_1 & S_2 & S_3 \\ Z_1 & S_1 & S_2 & S_3 \\ Z_2 & S_2 & S_2 & S_3 \\ Z_3 & S_1 & S_1 & S_3 \\ Z_4 & S_1 & S_3 & S_3 \\ Z_5 & S_1 & S_2 & S_1 \\ Z_6 & S_1 & S_2 & S_3 \end{bmatrix}$$

Рисунок 7 – Конечный автомат *KNPS* модели функционирования НПС

Пусть *MSost* - кортеж состояний, хранящий данные о текущих на момент времени t_i состояниях всех ПМ системы, а кортеж *MPotok* хранит текущие для каждого ПМ на момент времени t_i порядковые номера обрабатываемых потоков данных.

Рассмотрим множество состояний S автомата *KNPS*:

S_1 - состояние, в котором ПМ ожидает данные (*MSost*[i]=0).

S_2 - состояние, в котором ПМ ожидает данные (*MSost*[i]$\in (1, TO_i)$).

S_3 - состояние, в котором ПМ обработал данные и ждет передачи данных следующему ПМ (*MSost*[i]=-1).

Рассмотрим входной алфавит Z автомата *KNPS*:

Z_1 - нет данных для обработки.

Z_2 - есть данные, *MSost*[i]=0, состояния всех зависимых ПМ *MSost*[k]=-1 (k - индексы элементов i-ой строки, которые равны 1), t_i mod T_c равен 0.

Z_3 - есть данные, *MSost*[i]=0 и данные текущего ПМ не нужны остальным ПМ (по i-му столбцу нет значений 1).

Z_4 - есть данные, *MSost*[i]=0 и данные текущего ПМ нужны хотя бы одному ПМ (по i-му столбцу есть хотя бы одно значение 1).

Z_5 - есть данные, *MSost*[i]=-1.

Z_6 - остальные случаи.

Переход $S_1 \to S_2$ возможен только тогда, когда состояния всех ПМ, данные от которых принимает текущий ПМ, равны -1. То есть данные были обработаны и ждут передачи на текущий ПМ. Переход осуществляется в том случае, если остаток от деления текущего такта и времени цикла конвейера T_c равен 0. На практике это означает, что контроллер системы подал очередной тактовый импульс. В этом случае *MSost*[i] становится равным *TO*[i].

Переход $S_2 \to S_3$ и $S_2 \to S_1$ возможен только тогда, когда состояние i-го ПМ в кортеже *MSost*[i]=0, т.е. i-й ПМ закончил обработку данных и может ее передать другому ПМ. Если данные i-го ПМ нужны какому либо еще ПМ, то происходит переход $S_2 \to S_3$ (в этом случае *MSost*[i] становится равным -1), в противном случае $S_2 \to S_1$ (в этом случае *MSost*[i] становится равным 0).

Переход $S_3 \to S_1$ возможен тогда, когда данные ПМ были приняты для обработки другим ПМ. В этом случае *MSost*[i] становится равным ‘0’.

Рассмотрим выходной алфавит W автомата *KNPS*:

$W_1 = \lambda(S_1)$ - ПМ находится в ожидании данных;

$W_2 = \lambda(S_2)$ - ПМ в текущий момент времени t_i обрабатывает данные;

$W_3 = \lambda(S_3)$ - ПМ закончил обработку данных и ждет передачи данных.

Практические исследования

Результатом работы можно считать доработку программного комплекса «НейроКС» путем добавления новых подсистем «Моделирование нейропроцессора» и «Моделирование вычислительных систем на базе нейропроцессоров».

Функциональными возможностями модуля «Моделирование нейропроцессора» являются: стандартные функции (сохранение, загрузка файла); выбор команды путем активирования элементов интерфейса; моделирование выполнения команды и множества команд на процессоре; представление справочной информации о команде; генерация шаблона кода на основе команды; аналитический и графический вид представления данных; анализ команды процессора.

Рассмотрим функциональные возможности модуля «Моделирование нейропроцессорных систем». Исходными данными для являются: исходный код программы, дополнительный C++ код (необязателен) и архитектура системы. После загрузки исходных кодов, можно получить трассу любой подпрограммы на вкладке “Программа".

Функциональными возможностями модуля являются: стандартные функции (сохранение, загрузка файла); моделирование функционирования ПМ при выполнении программы; анализ выполнения подпрограммы на процессоре и на НПС; аналитическое и графическое представление результатов.

Результатами работы модуля являются:

- Для каждой подпрограммы:
  - Общие оценки: число команд в подпрограмме; число пустых команд.
  - Результаты моделирования и анализа: время выполнения подпрограммы; время проигрыша процессора; время простоев процессора; время обработки процессора, время обработки процессора как части НПС; время простоев процессора как части НПС; время передачи данных,
  - Вычисляемые вспомогательные оценки: коэффициент загруженности; среднее число тактов на команду,
  - Результаты отладчика: процент задержек; процент nul команд; процент nul левых частей; процент nul правых частей команды; задержки декодирования, выполнения, перехода, загрузки; сигналы на входной шине, выходной, локальной, глобальной, весовой.
- Для НПС:
  - Общие оценки: количество процессоров; объем передаваемых данных; коэффициент производительности, коэффициент загруженности.
  - Результаты моделирования и анализа: время работы системы; время работы на одном процессоре; цикл конвейера; время проигрыша НПС; время выигрыша НПС; время простоев НПС; время обработки НПС; суммарное время проигрыша для ПМ; суммарное время простоев для ПМ; суммарное время обработки для ПМ; общее время проигрыша; общее время простоев; общее время обработки; общая оценка НПС без учета ПМ; общая оценка НПС.

Более подробное описание программного комплекса представлено в [3].

Таким образом, поставленная задача разработки моделей нейропроцессоров и вычислительных систем на базе нейропроцессора была полностью выполнена.

Литература

1. Галушкин А.И. Нейронные ЭВМ - перспективное направление развития вычислительной техники – М: Препринт, 1991. - 615 с.

2. НТЦ ”Модуль”, 2009. [Электронный ресурс]. - URL: http://www.module.ru.

3. Романчук В.А., Ручкин В.Н. Разработка программного комплекса для моделирования и анализа нейропроцессорных систем обработки изображений //Цифровая обработка сигналов. 2010. - №1. - С.53-58.

MODELLING COMPLEX NEUROPROCESSOR SYSTEMS

V.A. Romanchuk, V.N. Ruchkin, V.A.Fulin

The article deals with modeling family NM640x neuroprocessors, developed by STC "Module", as well as modeling complex computing systems based on neuroprocessors. Proposed graphical representations of models of scalar and vector unit neuroprocessor represented as a highly parallel systems. A model of computer-based systems neuroprocessor as a state machine, analytical expressions are shown the simulation results. Shows the results of case studies.