УДК 004.715

С. И. Похилько, А. В. Оводенко

Донецкий национальный технический университет

САМООРГАНИЗАЦИЯ УПРАВЛЕНИЯ ОБРАБОТКОЙ ДИАГНОСТИЧЕСКОЙ ИНФОРМАЦИИ ОТ НЕРАВНОВЕСНЫХ ОБЪЕКТОВ

Динамічне управління чергами обробки діагностичної інформації від нерівн... важних об'єктів у сенсорних інформаційно-обчислювальних системах, що функціо... нують у реальному масштабі часу, забезпечить відмовостійкість і живучість сист...

ми завдяки своєчасному виявленню передаварійного стану контрольованих об'єктів і реконфігоруванню системи.

Ключові слова: сортування, динамічне управління, реконфігурована система, пріоритет заявки на обслуговування.

Динамическое управление очередями обработки диагностической информации от неравновесных объектов в сенсорных информационно-вычислительных системах, функционирующих в реальном масштабе времени, обеспечит отказоустойчивость и живучесть системы благодаря своевременному обнаружению предаварийного состояния контролируемых объектов и реконфигурированию системы.

Ключевые слова: сортировка, динамическое управление, реконфигурируемая система, приоритет заявки на обслуживание

Dynamic queue management processing of diagnostic information from non equilibrium sites in sensory information and computing systems operating in real time, provide fault tolerance and survivability of the system due to early detection of pre-emergency state-controlled facilities and reconfiguring the system.

Key words: sorting, dynamic control, reconfigurable systems, priority service requests.

Постановка задачи. На основе анализа традиционных методов управления обслуживанием информационных потоков, алгоритмов аппаратного и программного упорядочивания данных необходимо представить оптимальный по времени метод динамического управления потоками информации и осуществить синтез системы со следующими функциями: формирование управляющей информации для перестройки дисциплины обслуживания на основании параметров состояния входного потока (статического приоритета, величины штрафа за потерю заявки, скорости «старения» заявки в очереди на обслуживание); выбор оптимального алгоритма управления в реальном масштабе времени в зависимости от загрузки системы и критериев значимости контролируемых, диагностических параметров неравновесных объектов в нештатных режимах. Диагностика сложных неравновесных технических объектов (ТО) предполагает необходимость учета таких его свойств, как малоинерционность, наличие большого количества взаимокоррелированных контролируемых параметров различной диагностической значимости, иерархичность структуры, многорежимность, значительная неопределенность в поведении ТО и эмпирических измерений случайных процессов с выбросами значений контролируемых параметров за пределы допустимых зон, а также случайный характер воздействия на него со стороны внешней среды [1].

В сенсорных информационных вычислительных системах, отличительной особенностью которых является наличие большого количества периферийных устройств – объектов автоматизированной контрольной диагностики, как правило однородных и одноранговых по приоритетам, например нескольких двигателей самолета, возникает проблема адаптивного управления потоком заявок на обслуживание центральным управляющим процессором.

Присваивать приоритеты заранее невозможно. Состояние объектов контроля определяется по опросу. Микроконтроллеры, обрабатывающие первичную информацию от сенсоров, определяют, есть ли превышение допустимых порогов контролируемых параметров, и если такое отклонение имеет место, то передают информацию через узел коммутации на центральный процессор управления системой.

Может сложиться такая ситуация, что на некоторых объектах контроля будет иметь место отклонение параметров от допусков. Но величина этих отклонений у разных объектов может быть различна.

Для обеспечения живучести системы управления неравновесными объектами часто бывает необходимым и достаточным выполнить реконфигурирование системы [1, 2]. Таким образом, снижая некоторые характеристики объектов управ-

ления, например своевременно отключив один двигатель, находящийся в предаварийном состоянии, в котором имеет место отклонение параметров от допустимой нормы, и снизив тягу, тем самым снизив скорость движения объекта, можно исключить аварийную ситуацию и выполнить поставленную задачу, тем самым обеспечив живучесть.

Таким образом, узел коммутации является одним из тех компонентов, с помощью которых можно улучшить качество функционирования системы управления за счет адаптивного управления очередью заявок на обслуживание центральным процессором путем анализа полей значений контролируемых параметров. Необходимо отдавать предпочтение той заявке в буфере узла коммутации, где хранятся заявки на обслуживание, и выбирать, независимо от времени поступления, ту, где значение контролируемого диагностируемого параметра наиболее критично.

Обслуживание ранее выделенной заявки не прерывается, а выделенная становится первой на очереди после нее. За время цикла обслуживания заявки необходимо успеть выявить и выбрать в буфере очереди следующую заявку с наибольшим критическим значением диагностируемого параметра. При этом, если у одноранговых, однородных объектов контроля и диагностики, например, 4 авиационных двигателя, контролируются давление смазки, температура, вибрации и другие параметры, можно для каждого из них использовать схемы выбора критических значений, работающих параллельно во времени, то есть одновременно в виде пирамиды перепрограммируемого модуля сортировки и поиска для каждого из параметров.

Таким образом, аппаратная реализация процесса сортировки буфера очереди заявок на обслуживание центральным процессором информационно-управляющей системы (ИУС) на ПЛИС, на наш взгляд, является целесообразной, так как позволяет распараллелить процесс анализа и для динамических систем с неравновесными объектами, где имеется множество однородных, одноранговых точек контроля, где невозможно предварительно присвоить приоритеты обслуживания.

Обработка диагностической информации в реальном времени со скоростью большей, чем скорость протекания технологических процессов, или соизмеримой с ней, является одним из первоочередных требований в системах диагностики для обеспечения отказоустойчивости и живучести неравновесных объектов [2, САП].

Схема сортировки и поиска объекта контроля диагностики с наиболее критическим значением диагностического параметра на примере четырех однородных, одноранговых объектов приведена на рисунке.

Здесь IBUF16 – входные буферы, SR16CLE – регистры хранения контролируемых параметров, COMPMC16 – 16-битовые компараторы, M2_1 – мультиплексоры для выбора одной из двух шин с критическим значением, OR и NOR – логические элементы для управления маршрутом выбранного критического значения контролируемого параметра, OBUF16 – выходной 16-битовый буфер выдачи критического значения.

Реализован вариант сортировки и поиска максимального (или минимального) значения контролируемого параметра на ПЛИС (FPGA).

Код программы настройки перепрограммируемого аппаратного модуля Spartan3E в программной среде Xilinx ISE 11.1 для анализа 16 регистров:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity comp is
    generic (data_size: positive:=16);
    port (    P0: in STD_LOGIC_VECTOR(data_size-1 downto 0);
```

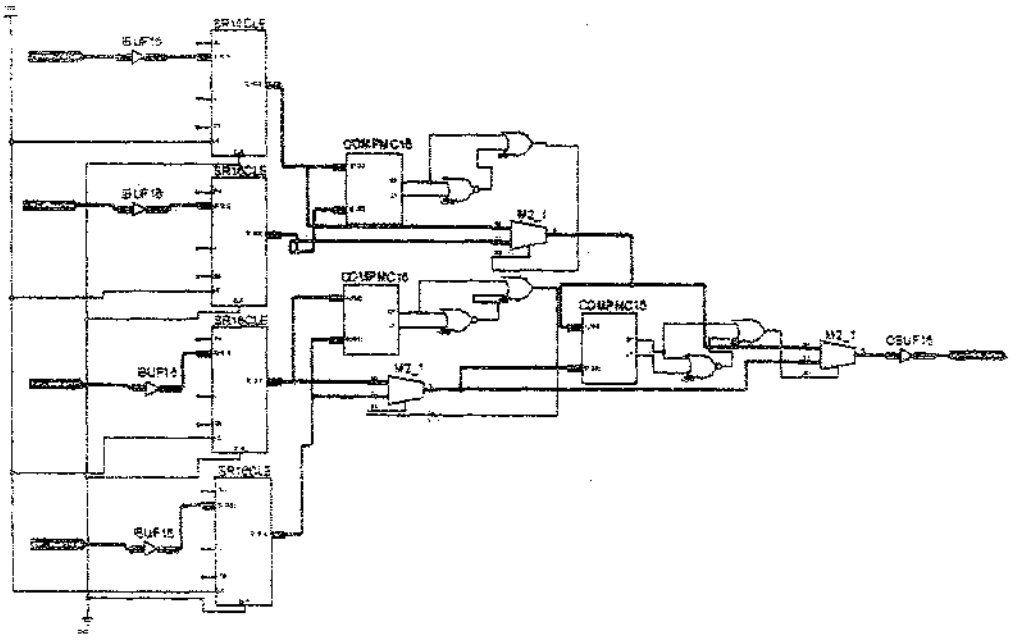

Рис. Схема сортировки и поиска контроля диагностики для четырех регистров (для четырех ТО)

```
                P1: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P2: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P3: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P4: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P5: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P6: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P7: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P8: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P9: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P10: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P11: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P12: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P13: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P14: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                P15: in STD_LOGIC_VECTOR(data_size-1 downto 0);
                O: out STD_LOGIC_VECTOR(data_size-1 downto 0));
    end comp;
    architecture Behavioral of comp is
    begin
        comparing: process (P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13,
P14, P15) is
                -- переменные для хранения промежуточных вариантов сравнения
                variable C1, C2, C3, C4, C5, C6, C7, C8: STD_LOGIC_VECTOR(data_
size-1 downto 0);
        begin
                -- первый каскад сравнения
                -- сравнение 1-й пары входных значений
                if (P0(data_size/2-1 downto 0) > P1(data_size/2-1 downto 0)) then
                        C1 := P0;
```

```
else
        C1(data_size-1 downto 0) := P1(data_size-1 downto 0);
end if;
-- сравнение 2-й пары входных значений
if (P2(data_size/2-1 downto 0) > P3(data_size/2-1 downto 0)) then
        C2 := P2;
else
        C2 := P3;
end if;
-- сравнение 3-й пары входных значений
if (P4(data_size/2-1 downto 0) > P5(data_size/2-1 downto 0)) then
        C3 := P4;
else
        C3 := P5;
end if;
-- сравнение 4-й пары входных значений
if (P6(data_size/2-1 downto 0) > P7(data_size/2-1 downto 0)) then
        C4 := P6;
else
        C4 := P7;
end if;
-- сравнение 5-й пары входных значений
if (P8(data_size/2-1 downto 0) > P9(data_size/2-1 downto 0)) then
        C5 := P8;
else
        C5 := P9;
end if;
-- сравнение 6-й пары входных значений
if (P10(data_size/2-1 downto 0) > P11(data_size/2-1 downto 0)) then
        C6 := P10;
else
        C6 := P11;
end if;
-- сравнение 7-й пары входных значений
if (P12(data_size/2-1 downto 0) > P13(data_size/2-1 downto 0)) then
        C7 := P12;
else
        C7 := P13;
end if;
-- сравнение 8-й пары входных значений
if (P14(data_size/2-1 downto 0) > P15(data_size/2-1 downto 0)) then
        C8 := P14;
else
        C8 := P15;
end if;
-- второй каскад сравнения
if (C1(data_size/2-1 downto 0) > C2(data_size/2-1 downto 0)) then
        C9 := C1;
else
        C9 := C2;
end if;
if (C3(data_size/2-1 downto 0) > C4(data_size/2-1 downto 0)) then
        C10 := C3;
```

```
        else
                C10 := C4;
        end if;
        if (C5(data_size/2-1 downto 0) > C6(data_size/2-1 downto 0)) then
                C11 := C5;
        else
                C11 := C6;
        end if;
        if (C7(data_size/2-1 downto 0) > C8(data_size/2-1 downto 0)) then
                C12 := C7;
        else
                C12 := C8;
        end if;
        -- третий каскад сравнения
        if (C9(data_size/2-1 downto 0) > C10(data_size/2-1 downto 0)) then
                C13 := C9;
        else
                C13 := C10;
        end if;
        if (C11(data_size/2-1 downto 0) > C12(data_size/2-1 downto 0)) then
                C14 := C11;
        else
                C14 := C12;;
        end if;
        -- четвертый каскад сравнения
        if (C13(data_size/2-1 downto 0) > C14(data_size/2-1 downto 0)) then
                O <= C13;
        else
                O <= C14;
        end if;
    end process comparing;
end Behavioral.
```

Проведены имитационное моделирование и тестирование алгоритма и структуры, реализующей этот алгоритм. Определены затраты времени на цикл сортировки содержимого 4, 16 и 32 16-битовых регистров с возможностью независимого анализа байтовых полей каждой из заявок в буфере очереди на обслуживание в узле коммутации и автоматического определения адреса объекта контроля с самым критическим значением контролируемого параметра. Затраты времени соответственно равны 13 наносекунд; 21 наносекунда; 29 наносекунд.

Причем программно-аппаратная реализация на микропроцессорных структурах потребует значительно больше затрат времени не только для сортировки по одному параметру, но и дополнительно на определение адреса контролируемого объекта, а также на сортировку других параметров, например температуры, давления, скорости и т.д. каждого из контролируемых объектов.

При аппаратной реализации на перепрограммируемых модулях (ПЛИС) процессов сортировки и поиска наиболее критических значений множества контролируемых однородных объектов и множества однородных параметров эти процессы реализуются параллельно во времени. Для каждого параметра выделен отдельный модуль с адекватной перепрограммируемой настройкой. Последовательно выполняется только загрузка n-регистров каждого из модулей из соответствующих полей формата кадра заявки на обслуживание в буфере очереди коммутирующего узла. Время анализа контролируемого параметра определяется выражением

$$T = \tau_{cл} \times \log_2 n,$$

где n – количество регистров (объектов контроля); $\tau_{cл}$ – затраты времени на попарное сравнение двух элементов, селекцию одного из двух значений, передачу выбранного значения через мультиплексор. Кроме того, в первом слое модуля должно учитываться время на запись в регистры.

Количество слоев в модуле определяется как

$$K_{сл} = \log_2 n,$$

где n – количество регистров (объектов контроля).

Выводы. Самоорганизация управления обработкой диагностической информации от неравновесных объектов заключается в том, что при аппаратной реализации на перепрограммируемых модулях (ПЛИС) процессов сортировки и поиска наиболее критических значений множества контролируемых однородных объектов и множества однородных параметров реализуются параллельно во времени. При этом временные затраты существенно меньше, чем при аппаратно-программной реализации данной задачи на микропроцессорных структурах.

Актуальность задачи построения модулей сортировки в базисе современных ПЛИС FPGA подтверждается современными публикациями [3], где внимание уделяется разработке аппаратного модуля, который обеспечивает близкую к максимально возможной скорости с учетом использования ПЛИС FPGA, которые имеют низкую стоимость. В рассмотрение брались ПЛИС FPGA наиболее известных на данный момент производителей: ПЛИС фирм Altera и Xilinx.

Библиографические ссылки

1. Гузик Ф. В. Статистическая диагностика неравновесных объектов / Ф. В. Гузик, В. И. Кидалов, А. П. Самойленко. – СПб.: Судостроение, 2009. – 278 с.

2. Палагин А. В. Реконфигурируемые вычислительные системы: основы и приложения / А. В. Палагин, В. Н. Опанасенко. – К.: Просвіта, 2006. – 280 с.

3. Сероштан С. Ю. Разработка аппаратного модуля сортировки с последовательным вводом данных и минимальным временем обработки / С. Ю. Сероштан, А. А. Гриценко, Ю. Г. Зинченко // Наук. праці ДонНТУ. – 2011. – Вип. 13.

Надійшла до редколегії 1.06.2012 р.