УДК 681.3

ПРОГРАММНАЯ МОДЕЛЬ СИСТЕМЫ НА КРИСТАЛЛЕ

К. Е. Окунев,
ассистент
А. А. Ключарев,
канд. техн. наук, доцент
Санкт-Петербургский государственный университет аэрокосмического приборостроения

Предлагается формализованное описание программной модели систем на кристалле как системы реального времени. Модель строится на основе совмещения связанных автоматных, графовых и вероятностных составляющих системы и входных воздействий. Приводятся примеры использования рассмотренной модели для определения интерфейсных блоков и их анализа с помощью характеристик типа «пересечений уровня».

Ключевые слова *— системы на кристалле, совместная разработка программного и аппаратного обеспечения.*

Введение

Текущие темпы развития технологий в электронике привели к широкому использованию систем на кристалле (СнК) в бытовой и промышленной технике. Чаще всего СнК представляют собой системы реального времени. Неотъемлемой частью СнК являются микропроцессорные элементы, программное обеспечение (ПО) которых реализует функции управления системой. ПО в СнК тесно связано с работой специализированной аппаратуры и коммуникационных интерфейсов. Характеристики проектируемой системы полностью зависят от согласованности между собой всех программных и аппаратных компонентов.

При разработке СнК функционально законченная система формируется из готовых аппаратных и программных модулей, включая операционные системы реального времени (ОСРВ) [1]. Это требует разработки принципов отбора отдельных модулей, решения задач программной и аппаратной совместимости и оптимизации режимов работы компонентов системы для конкретной СнК.

Для описания аппаратного обеспечения в СнК на различных этапах проектирования используются три типа основных моделей:

— программные модели аппаратной реализации, которые описывают правила и алгоритмы взаимодействия с аппаратурой;

— функциональные модели — *behavior models* (поведенческие модели), упрощенно описывающие правила работы компонента аппаратного обеспечения;

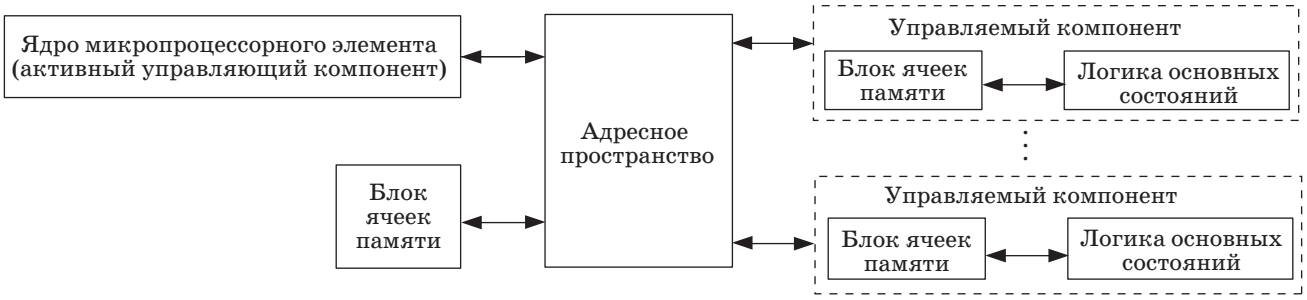
— модели аппаратной реализации СнК.

Последние два типа моделей имеют стандартизированные описания [2]. Функциональные модели описываются с помощью несинтезируемых конструкций языков *HDL* (*Hardware Description Language*), таких как *Verilog* и *VHDL*. Модели аппаратной реализации описываются только синтезируемыми конструкциями *HDL*. Обе эти модели сложны для анализа при проектировании ПО, поскольку учитывают детализированные алгоритмы реализации целевой функции аппаратного обеспечения. Поэтому для разработки ПО обычно используется программная модель аппаратной части СнК.

Однако в настоящее время отсутствуют единые методики и правила описания программных моделей, что снижает качество СнК и ограничивает возможности переноса ПО на другие системы. Целью данной работы является построение формальной программной модели СнК, разработка принципов ее анализа и практической реализации для систем реального времени.

Программная модель и ее составляющие

С точки зрения программной модели аппаратные компоненты можно разделить на активные управляющие и управляемые (рис. 1). Под активными управляющими компонентами понимаются ядра микропроцессорных элементов, осуществляющих общее управление системой посредством исполнения программного кода. Управляемые компоненты взаимодействуют с микропроцессорным элементом. При этом на аппаратном уровне они могут выполнять функции управле-



■ ***Рис. 1.*** *Пример взаимодействия составляющих системы*

ния, определяемые логикой изменения основных состояний.

Программная модель представляется формализованными описаниями основных состояний управляемых компонентов, программно доступных элементов памяти, адресных пространств и системы команд управляющих блоков системы. Программно доступные элементы памяти не только используются для хранения данных, но и участвуют в управлении и обмене данными между компонентами.

Формализованное описание состояний управляемых компонентов в терминах теории конечных автоматов [3, 4] может быть представлено как

$$Km = (Q, I, Ft, q_0, O, Fo),$$

где Km — конечный автомат; Q — множество состояний автомата; I — множество входных сигналов; Ft — функции переходов; q_0 — начальное состояние; O — множество выходных сигналов; Fo — функция выходов.

При построении программной модели необходимо разделять множества входных и выходных сигналов на несколько подмножеств из-за различия свойств и природы интерфейсных сигналов, а также особенностей их использования. Это непосредственно влияет на программную модель компонента. Выделим следующие подмножества [5]:

$I_{inp} \subset I$ — множество входных сигналов, не участвующих в управлении системой, но являющихся источником данных для внутренней логики;

$I_c \subset I$ — множество управляющих сигналов, которые участвуют при выборе различных ветвей в алгоритме функционирования компонента, причем $I = I_{inp} \cup I_c$;

$Ip \subset I$ — множество входных и управляющих сигналов, доступных для ПО;

$Ii \subset I$ — множество входных и управляющих сигналов, не имеющих прямого управления со стороны ПО;

$Op \subset O$ — множество выходных сигналов, непосредственно доступных для ПО;

$Oo \subset O$ — множество выходных сигналов, обрабатываемых другими блоками и не являющихся напрямую доступными ПО.

В общем случае на каждый вход из множества I подаются сигналы, значения которых представляют собой многомерные случайные величины, описываемые многомерными плотностями вероятностей либо функциями распределения. Соответственно, модель состояний носит вероятностный характер. Изучение вероятностных характеристик поведения аппаратного обеспечения является принципиально важным аспектом при проектировании систем как мягкого, так и жесткого реального времени. Тогда множеству I ставится в соответствие множество случайных величин $p: I \to F_d$, где F_d — множество плотностей распределения $f_d i(\mathbf{X})$, $i \epsilon [1, n]$, многомерных случайных величин:

$$F_d = \{f_d 1(\mathbf{X}), f_d 2(\mathbf{X}), ..., f_d n(\mathbf{X})\},$$

где $\mathbf{X}$ — вектор случайных величин, распределения которых зависят от реализации конкретных систем.

Основная задача ПО состоит в реализации функций настройки компонента на заданный режим и организации управления взаимодействием с другими элементами системы в реальном времени. Алгоритм управления каждым компонентом обеспечивает достижение заданных состояний q_t и функции выходов из случайного текущего состояния q_p, при этом $q_t \in Q$, $q_p \in Q$.

Совокупность правил и алгоритмов обработки входного потока данных определяет режим работы, что с точки зрения конечного автомата представляет ограничения на последовательность изменения состояний, т. е. не любой путь может быть использован для достижения заданного состояния. Конечный автомат может быть описан ориентированным графом переходов $G(Q, E)$, где состояния соответствуют вершинам графа ($q_j \in Q$). Дуги графа определяются функциями переходов $E = \{(q_1, Ft(q_1, I_1)), ..., (q_j, Ft(q_j, I_j))\}$, где $I_j \in I$. В этом случае маршруты в конечном автомате есть подмножество всех возможных путей в графе $E \subset Q \times Q$.

Следовательно, задание режима работы сводится к определению подмножества путей ($P_m \subset E$, $g : E \to Pm$) графа, допустимых к использованию для достижения цели (перевода системы из состояния q_p в q_t), которые определяются подмножеством входных управляющих сигналов ($I_c \subset I$). Тогда алгоритм управления может быть представлен совокупностью последовательных множеств входных воздействий I_1, I_2, ..., I_j, выдаваемых ПО в зависимости от состояния системы. Эти воздействия позволяют обеспечить передвижение по заданным маршрутам из P_m. Для каждого режима работы существует свой сценарий входных воздействий, который позволяет пройти указанный маршрут путем следования по определенным дугам графа.

Выбор алгоритма управления сводится к определению наилучшего маршрута по критериям длины пути и затрат ресурсов. Для этого необходимо каждому переходу сопоставить весовые коэффициенты, отражающие реальные затраты на каждый переход или его существование $c : E \to P$, где P — это множество функций, определяющих весовые коэффициенты для каждого перехода. В системах СнК основными весовыми характеристиками являются энергопотребление перехода в активной фазе; энергопотребление перехода в пассивной фазе (затраты на само существование перехода); аппаратные затраты на реализацию перехода; временная длительность перехода и т. д. Весовые коэффициенты зависят от конкретной реализации, например от существования других переходов, так как несколько переходов могут использовать одни и те же аппаратные ресурсы.

В реальных системах возникает вопрос о достижимости конечной точки заданного маршрута. Компонент является полностью управляемым ПО, если множество управляющих сигналов состоит только из программно доступных сигналов или сигналов, напрямую порожденных от них ($Ic \cap Ip = Ip$ и $Ic \cap Ii = \varnothing$). Действительно, если переходы обеспечиваются только программно доступными сигналами, то условие выполняется автоматически, если же в функции переходов участвуют результаты функции выходов, то они, так или иначе, являются производными от программно управляемых входных сигналов. В случае невыполнения условия полной программной управляемости компонента система является не полностью определенной, так как существуют пути, не зависящие от функционирования ПО. В этом случае поведение системы будет зависеть от вероятностных характеристик входного потока сигналов из множества Ii.

Маршрут является абсолютно достижимым, если во всех вершинах (состояниях) на протяжении всего маршрута переходы осуществляются только посредством программно доступных управляющих сигналов $Ic \subset Ip$ и их производных и не существует перехода, нарушающего маршрут ($f : P_m \to S$, где $S \subset Q$ и $Es = \{(q_1, Ft(q_1, I_1)), ..., (q_j, Ft(q_j, I_j))\}$, где $\{q, ..., q_j\} \subset S$ и $Es \cup P_m = P_m$ и $q_p \notin S_1$, где $l : Es \to S_1$), т. е. не существует пути, при использовании которого конечная точка оказывается недостижимой без перехода в начальную точку маршрута.

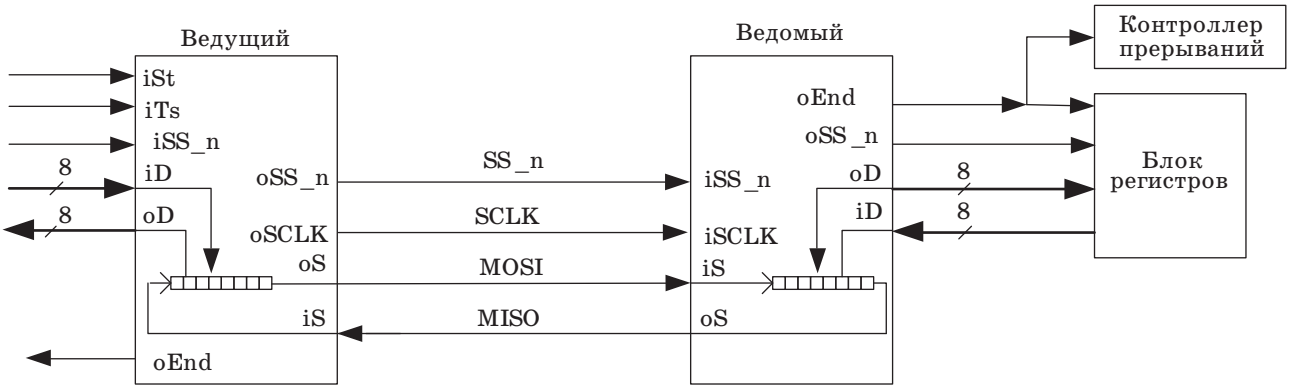
Условие абсолютной достижимости не выполняется, если в маршруте существует хотя бы один путь, который образует замкнутый контур в графе с отправной точкой маршрута (q_p) без достижения конечной точки (q_t), что приводит к существованию перехода, делающего маршрут недостижимым при определенных условиях либо условно достижимым. Маршрут является условно достижимым, если существуют условия, при которых пути в маршруте образуют замкнутый контур с участием начального состояния и без участия конечного, а также существует зависимость от программно неуправляемых сигналов.

В графе можно выделить вершины (состояния), проходя через которые система может отклониться от заданного маршрута $Ft(q_j, I_j) \notin Es$. Такие состояния определим как критичные состояния конечного автомата, в которых неправильная или несвоевременная выдача управляющих сигналов со стороны активного компонента может привести к недостижимости конечной точки маршрута либо отложить достижение на бесконечно долгий срок. Именно наличие таких состояний формирует требования к ПО систем реального времени исходя из условий сохранения заданного маршрута.

Таким образом, модель состояний — это совокупность связанных автоматной модели, модели вероятностных характеристик входных сигналов, модели весовых коэффициентов стоимости существования и реализации элементов автоматной модели, а также функций связывания этих моделей в единое целое: $<Km, Fd, P, p, c>$. Данная совокупность позволяет оценивать различные параметры системы с точки зрения возможности реализации, надежности, вероятностных и стоимостных характеристик, а также производить оценку качества алгоритмов ПО.

Описание модели состояний программной модели на примере контроллеров *SPI*

Рассмотрим примеры описания и анализа модели состояний программной модели на примере ведущего и ведомого контроллеров интерфейса *SPI* (*Serial Peripheral Interface*), представленных на рис. 2. Данный интерфейс находит широкое применение в СнК, так как он является интер-



■ ***Рис. 2.*** *Сигналы и взаимодействие ведущего и ведомого блоков шины SPI*

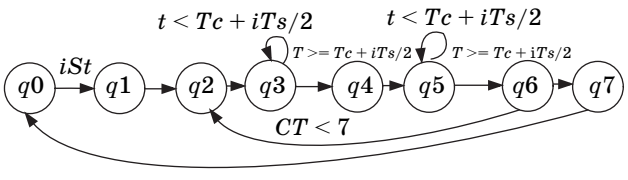
фейсом взаимодействия между компонентами в рамках одной печатной платы или между внутренними компонентами СнК. Его популярность объясняется высоким быстродействием и простотой реализации как самих контроллеров, так и линий связи за счет минимального набора сигналов.

Интерфейс является последовательным синхронным интерфейсом [6]. В ходе обмена всегда существует один ведущий контроллер (*Master device*) и ведомый контроллер (*Slave Device*). Ведущий контроллер (ВК) отвечает за выбор активного ведомого контроллера (причем только один может быть активным в один момент времени), производимый низким уровнем сигнала разрешения (*SS_n*). Сигнал *SS_n* может находиться в активном состоянии, даже если реальной передачи данных не происходит. Также ВК отвечает за формирование временной диаграммы передачи. Минимальная единица передаваемой информации на входе ВК — это байт. ВК передает этот байт последовательно, маркируя передачу синхросигналом (*SCLK*), бит за битом, на одном фронте *SCLK* бит выдается ведущим и ведомыми контроллерами, а на противоположном захватывается в сдвиговом регистре. Данная итерация повторяется 8 раз для передачи одного байта, затем *SCLK* останавливается, если нет другого байта для передачи.

Рассмотрим модель ВК *SPI*, построенную на анализе временной диаграммы (рис. 3) и спецификаций *SPI*. Модель системы представляет из себя совокупность состояний $Q = \{q0, q1, q2, q3, q4, q5, q6, q7\}$ с начальным состоянием $q0$; множеств входных и выходных сигналов $Ip = \{iTs, iSt, iSS\_n, iD[7:0]\}$; $Ii = \{iS\}$; $Op = \{oD[7:0], oEnd\}$; $Oo = \{oS, oSS\_n, oSCLK\}$; функции переходов, описанной в виде направленного графа (рис. 4); функции выходов, представленной в таблице, в виде изменений состояний сигналов в каждом из состояний. Также в модели присутствуют внутренний счетчик реального времени (*Tc*), счетчик выдаваемых битов (*CT*) и регистр данных (*D*[7:0]), что позволяет упростить логическую схему графа. Определим маршрут как путь от состояния $q0$ до состояния $q7$ с последующим возвращением в $q0$.

Анализ данной модели показывает, что ведущий компонент *SPI* является полностью программно управляемым и удовлетворяет требованию абсолютной достижимости, так как ни один из его переходов не зависит от элементов множества *Ii*. В модели отсутствуют критические состояния, так как не существует пути, нарушающего маршрут. Модель позволяет определить условие сохранения доступности для программного обеспечения состояния сигнала *oD*. Также определяются временные характеристики процесса работы компонента с точки зрения ПО, поскольку вычисляется время пути от исходной точки маршрута до конечной. Это может быть переведено в количество циклов микропроцессорного устройства, в течение которых ПО может переключиться на реализацию других задач.

■ ***Рис. 3.*** *Пример временной диаграммы интерфейса SPI*



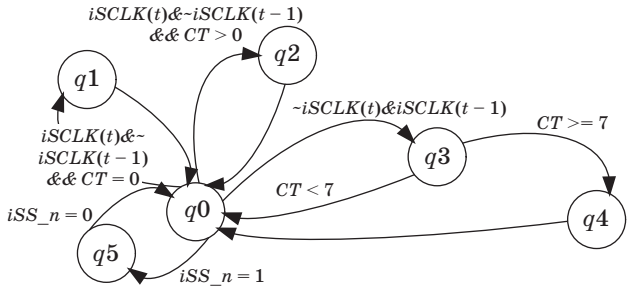
■ ***Рис. 4.*** *Граф переходов состояний ведущего устройства SPI*

■ ***Таблица***

Состояние	Изменение выходов контроллера	
	ведущего	ведомого
$q0$	—	—
$q1$	$oSS\_n = iSS\_n$ $D = iD$ $CT = 0$ $oEnd = 0$	$D = iD$ $oD = iD[7]$
$q2$	$Tc = t$ $oSCLK = 1$ $oS = D[7]$	$oD = D[7]$
$q3$	—	$D = D << 1$ $D[0] = iS$ $CT = CT + 1$
$q4$	$Tc = t$ $D = D << 1$ $D[0] = iS$ $oSCLK = 0$	$oD = D$ $CT = 0$ $oEnd = not\ oEnd$
$q5$	—	$CT = 0$
$q6$	$CT = CT + 1$	—
$q7$	$oD = D$ $oEnd = 1$	—

Модель ведомого блока интерфейса *SPI*, построенная на основании анализа рис. 3 и спецификаций, описывается совокупностью состояний $Q = \{q0, q1, q2, q3, q4, q5\}$ с начальным состоянием $q0$; множеств входных и выходных сигналов $Ip = \{iD[7:0]\}$; $Ii = \{iS, iSS\_n, iSCLK\}$; $Op = \{oD[7:0], oEnd, oSS\_n\}$; $Oo = \{oS\}$; функции переходов, описанной в виде направленного графа (рис. 5); функции выходов, представленной в таблице, в виде изменений состояний сигналов в каждом из состояний. В качестве внутренних сигналов используются регистр данных (D) и счетчик (CT). Основным маршрутом данных является путь от состояния $q0$ до состояния $q4$ с последующим возвращением в $q0$.

Как видно из графа переходов на рис. 5, ведомое устройство не является программно управляемым компонентом, а также не удовлетворяет требованиям абсолютной достижимости. Пример наглядно демонстрирует систему, которая является полностью зависимой от внешних устройств,



■ ***Рис. 5.*** *Граф переходов состояний ведомого устройства SPI*

так как на программном уровне отсутствует возможность контроля за приемом и передачей данных, что накладывает жесткие ограничения на реализацию ПО.

Рассмотрим взаимодействие данной модели с ПО с точки зрения задачи приема непрерывной последовательности байтов на временном интервале T без учета структуры передаваемой информации, что подразумевает стационарность характеристик сигналов из $Ic \subset Ii$ на $[t_i, t_i + T]$. Так как программно доступные множества сигналов отображаются на соответствующие регистры, доступные ПО, а сигнал *oEnd* поступает на блок контроллера прерываний, то взаимодействие с данным устройством может производиться в двух режимах: обмен по опросу флага *oEnd*; обмен по прерыванию, выработанному по флагу *oEnd*. Особенностью обмена является то, что данные на выходе перезаписываются с приходом каждого нового байта, а данные для передачи захватываются с началом приема следующего. Так как блок не является программно управляемым, то задача ПО — уложиться во временные рамки по приему/передаче новых данных. Для режима опроса они выражаются в периоде опроса регистра *oEnd*, а для режима обмена по прерываниям — во времени реакции на соответствующее прерывание. Для простоты дальнейших рассуждений предположим, что данное время одинаково в обоих случаях и соответствует $T_{soft}(t)$, тогда условием правильного приема является соблюдение условия $T_{soft}(t) < T_{rec}(t)$, где $T_{rec}(t)$ — время приема очередного байта.

Исходя из анализа условий достижения конечного состояния, формирующего *oEnd*, из начального по графу переходов делается вывод, что необходимо наличие отрицательного значения на входе iSS_n, а также поступление $n = 8$ сигналов *iSCLK*. Полагая, что сигнал iSS_n постоянно находится в активном состоянии на $[t_i, t_i + T]$, можно утверждать, что $T_{rec}(t)$ полностью зависит от поведения сигнала *iSCLK*. На интервале $[t_i, t_i + T]$ значения длительностей передачи одного бита можно рассматривать как непрерывные случайные последовательности. В независимости от источника формирования синхросигнала передачи существует флуктуация периода синхросигнала, особенно заметная при его формировании программными способами, а не специализированной аппаратурой. Флуктуация периода влияет на время передачи байта в соответствии с графом переходов, а следовательно, и блока в целом.

Модель периода синхросигнала *iSCLK* можно представить в виде

$$T_{sclk}(t) = T\_sclk + \xi(t),$$

где T_sclk — длительность периода синхросигнала на интервале $[t_i, t_i + T]$; $\xi(t)$ — флуктуационная

составляющая, появляющаяся за счет работы аппаратных и программных средств генерации частоты.

В случае флуктуаций фронтов сигнала в окрестности $f_{T\_SCLK}(t)$ с сохранением длительности периода битов, называемой фазовой флуктуацией, можно утверждать, что длительность приема байта будет соответствовать

$$T_{rec}(t) = nT\_sclk + \xi(t).$$

Причем, исходя из анализа источников синхросигналов фазовая флуктуационная составляющая $\xi(t)$ распределена по нормальному закону [7].

Время $T_{soft}(t)$ также может содержать случайную составляющую, отражающую возможность передачи управления более высокоприоритетным задачам или прерываниям на $[t_i,\ t_i + T]$. В этом случае можно записать

$$T_{soft}(t) = T\_sft + T_{int}k(T),$$

где T_sft — интервал опроса данных, не зависящий от внешних случайных воздействий и алгоритмов планирования; $k(T)$ — количество прерываний исполнения задачи опроса на интервале T более высокоприоритетным процессом или прерыванием; T_{int} — время обработки высокоприоритетной задачи.

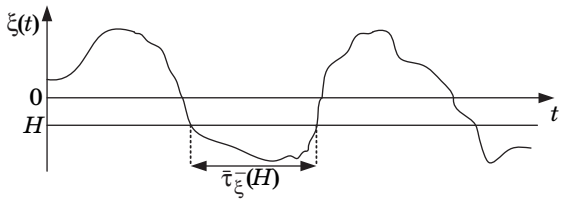
Задача анализа соотношения $T_{soft}(t) < T_{rec}(t)$ сводится к анализу применимости самого входного потока к функциональности системы исходя из программной модели. Для решения подобных задач удобно воспользоваться характеристиками типа «пересечений уровня» [8]. Выбор уровня будет определяться степенью влияния случайных составляющих и возможных моделей системы. Далее рассматриваются примеры определения уровней в различных случаях.

Если принять независимость $T_{soft}(t)$ от случайной составляющей и условие стационарности T_sclk на $[t_i,\ t_i + T]$, то может быть осуществлен переход к анализу только распределения флуктуационной составляющей. В данном случае предельный уровень H можно определить как

$$H = T\_sft - nT\_sclk.$$

Пересечение этого уровня указывает на невыполнение временных требований по допустимой частоте входного потока данных, который может обработать ПО, и классифицируется как нарушение работоспособности для систем жесткого реального времени. Для систем мягкого реального времени оценивается количество таких пересечений $N_\xi(H,\ T)$ и время нахождения в запрещенной зоне $\overline{\tau}_\xi(H)$, проиллюстрированные на рис. 6.

При принятии гипотезы распределения флуктуационной составляющей по нормальному закону применим аналитический способ анализа работоспособности. Рассматриваем флуктуацию периода синхросигнала *SCLK* в виде стационарного гауссова случайного процесса с нулевым математическим ожиданием, некоторой дисперсией σ_ξ^2 и плотностью вероятности [9]:



■ ***Рис. 6.*** *Случайный процесс флуктуации периода синхросигнала*

$$\rho_\xi(\xi) = \frac{1}{\sqrt{2\pi\sigma_\xi^2}} e^{\frac{-\xi^2}{2\sigma_\xi^2}}. \qquad (1)$$

Для такой модели распределения (1) среднее число пересечений уровня H траекторией случайного процесса на $[t_i,\ t_i + T]$ определяется как [8]

$$N_\xi(H,T) = \frac{T}{\pi}\sqrt{-r''(0)}\, e^{\frac{-H}{2\sigma_\xi^2}}, \qquad (2)$$

где $r''(0)$ — вторая производная от нормированной корреляционной функции $r(\tau)$ рассматриваемого процесса $\xi(t)$ при $\tau = 0$.

Средняя длительность несоблюдения $T_{soft}(t) < T_{rec}(t)$ при $h = \sigma_\xi^{-1}H$

$$\overline{\tau}_\xi(h) = 2\pi(-r''(0))^{-1/2}\Phi(h)e^{\frac{h^2}{2}}, \qquad (3)$$

где $\Phi(x) = (2\pi)^{-1/2}\int_{\infty}^{x} e^{-y^2/2}\,dy$ — интеграл вероятности.

Второй случай — это бесконечно малое влияние $\xi(t)$ на общее поведение системы вследствие высокой стабильности генератора и невысоких частот передачи данных и наличие существенного влияния $T_{int}k(T)$ на $T_{soft}(t)$. Возникновение событий, приводящее к прерыванию текущего процесса, может подчиняться закону распределения Пуассона:

$$\rho_i(T) = \frac{\lambda^i}{i!}e^{-\lambda},\ i = 0, 1, 2, \ldots, \qquad (4)$$

где i — количество событий на интервале наблюдения T; λ — параметр распределения, вычисляемый как $\lambda = \nu T$, где ν — интенсивность потока.

В этом случае допустимый уровень можно определить как

$$H = \left[\frac{nT\_sclk - T\_sft}{T_{int}}\right], \qquad (5)$$

а среднее нормированное количество превышений уровня H рассчитывается как

$$N^{+}(H,1) = 1 - \sum_{i=0}^{H} (\lambda^{i} \exp(-\lambda) / i!). \qquad (6)$$

Пересечение уровня H указывает на невыполнение временных требований по совместимости программного и аппаратного обеспечения и позволяет аналитически оценивать данную ситуацию на этапе проектирования.

Таким образом, с помощью выражений (1)–(6) можно рассчитать вероятности выхода за уровень H, среднее количество пересечений H, а также время нахождения случайного процесса ниже H для определения совместимости программного и аппаратного обеспечения по временным и вероятностным характеристикам с точки зрения систем реального времени.

Заключение

Комбинация автоматной, вероятностной и графовой составляющих в описании программной модели аппаратной части системы реального времени, предложенная в работе, формализует представление функций и логики работы с аппаратурой с точки зрения ПО. Использование приведенной совокупности формализованных описаний совместно с характеристиками типа «пересечений уровней» для входных воздействий позволяет разработать обобщенный комплексный подход к анализу систем реального времени и оптимизировать выбор программно-аппаратных решений аналитическим путем на стадии проектирования. Это имеет важное практическое значение для повышения качества и сокращения времени на разработку конечной продукции.

Предложенный подход проиллюстрирован простейшими примерами на основе *SPI*. Но основными его приложениями являются сложные нестандартные коммуникационные интерфейсы и устройства, проектируемые для специализированных СнК. Множество асинхронных взаимодействующих процессов и сигналов, описываемых сложными законами, характерно для таких систем, поэтому анализ без соответствующих моделей затруднителен. Предложенные решения и набор моделей позволяют решить эту задачу.

Литература

1. **Астапкович А. М.** Микрооперационные системы реального времени / Под. ред. М. Б. Сергеева. — СПб.: Политехника, 2002. — 246 с.
2. **Ciletti M. D.** Advanced Digital Design with the Verilog(TM) HDL. — Prentice Hall, 2002. — 1008 с.
3. **Gupta R. C.** Co-Synthesis of Hardware and Software for Digital Embedded Systems: The Springer International Series in Engineering and Computer Science. — Springer, 1995. — 288 с.
4. **Хопкрофт Д., Мотвани Р., Ульман Д.** Введение в теорию автоматов, языков и вычислений: Пер. с англ. — М.: Издательский дом Вильямс, 2002. — 528 с.
5. **Садовничий В. А.** Теория операторов. — М.: Высш. шк., 1999. — 368 с.
6. **AVR RISC** microcontroller: Data Book. Atmel Inc, 1999. — 930 с.
7. **Banerjee D.** PLL Performance, Simulation and Design. 4th ed. — Dog Ear Publishing, LLC, 2006. — 344 с.
8. **Тихонов В. И., Хименко В. И.** Выбросы траекторий случайных процессов. — М.: Наука, 1987. — 303 с.
9. **Ивановский Р. И.** Теория вероятностей и математическая статистика. Основы, прикладные аспекты с примерами и задачами в среде Mathcad. — СПб.: БХВ-Петербург, 2008. — 528 с.