Мехоношин Станислав, группа 3.2

Проектирование асинхронных схем
Крис Дж. Мейер

1. Предисловие
2. Благодарности
3. Введение
    a. Описание проблемы
    b. Каналы связи
    c. Протоколы взаимодействия
    d. Графическое представление
    e. Нечувствительные к задержке схемы
    f. Схемы Хаффмана
    g. Схемы Мюллера
    h. Синхронные схемы
    i. Проверка
    j. Применение
    k. Начиная работу
    l. Источники
    m. Задачи
4. Каналы взаимодействия
    a. Базовая структура

Введение

Вино - это разлитая по бутылкам поэзия.

- Роберт Льюис Стивенсон

Вино воодушевляет и делает человека более страстным.

- Овидий

Я сделал вино из изюма, и мне не нужно будет ждать, пока но состарится.

- Стивен Райт

В этой главе с помощью простого примера, мы сделаем неформальное введение во многие концепции и методы проектирования, которые описаны в данной книге. Каждая из затронутых тем будет рассмотрена более подробно в следующих главах.

1.1 Описание проблемы

В маленьком городке на юге штата Юта есть небольшой винный завод, а рядом с ним винный магазин. Так как жители этого городка уверены, что Сухой закон до сих пор в силе, то у магазина есть только один клиент. В магазине есть маленькая полка, на которой можно разместить только одну бутылку вина. Каждый час владелец магазина получает бутылку свежего вина и ставит ее на полку. В половине каждого часа приходит клеинт, чтобы купить вино, тем самым освобождая место для новой бутылки. Покупатель знает как важно приходить точно вовремя. Однажды он прибыл раньше времени и обнаружил полку в магазине пустой, что весьма разозлило его. В другой раз, он опоздал и обнаружил, что владелец магазина сам выпил вино, чтобы освободить место для новой бутылки. Но самый досадый момент произошел, когда покупатель вошел в то время, как владелец магазина ставил бутылку на полку. От неожиданности они столкнулись, бутылка вина, упав на пол, разбилась, в результате чего

никто не смог насладиться этим прекрасным напитком.

Такой синхронный метод продажи работал какое-то время, и обе стороны были довольны. Но потом(в середине 1980х) в городе появилась телефонная связь. Это великолепное изобретение взволновало горожан. У покупателя вина появилась замечательная идея. Он знал, что завод может работать быстрее, если он сможет быстрее покупать вино. Поэтому он предложил владельцу магазина сообщать по телефону когда новая партия вина будет доставлена в магазин. Это должно было исключить вероятность раннего появления и обманутых ожиданий. Не прошло и часа, как покупателю сообщили, что можно забирать вино. Он был так взволнован, что бежал со всех ног. Купив вино, он предложил владельцу магазина сообщать на завод, когда появляется место для новой партии вина. Именно так продавец и поступил, и через 10 минут у покупателя уже зазвонил телефон. Так продолжалось весь час. Иногда требовалось 10 минут, чтобы телефон зазвонил, иногда время ожидания растягивалось до 20 минут. Один раз покупателю позвонили через 5 минут после того, как он покинул магазин(к счастью он жил рядом). В конце часа он понял, что выпил уже 5 бутылок вина!

Немного опьянев, он решил вздремнуть. Через час он проснулся в расстроенных чувствах, поняв что не повесил трубку телефона. Огорчившись, что забыл про вино, он сорвался в магазин, ожидая, то продавец уже выпил несколько бутылок его вина. Но к его ужасу, он увидел только одну бутылку вина на полке, и ни одной пустой вокруг. На вопрос почему завод перестал поставлять вино, продавец ответил, что так как он не звонил, производители решили приостановить доставку, пока в магазине не появится место для новой бутылки вина.

С того для этот асинхронный метод торговли стал основным принципом ведения бизнеса. Производители вина были довольны, так как продавали в среднем больше вина. Жена владельца магазина была рада, что ее мужу не нужно выпивать остатки вина. Покупатель был невероятно счастлив, потому что теперь он мог получать вино намного быстрее, а при необходимости мог сделать перерыв, будучи

спокойным, что не пропустит ни одной бутылочки вина.

1.2 Каналы связи

Однажны инженер VLSI(сверхбольших интегральных схем) остановился в маленьком городке, и разговорился с владельцем винного магазинчика о его бизнесе.

Дела шли хорошо, но жена хозяина магазина продолжала приставать к нему с напоминанием о поездке на остров Мауи, которую он обещал уже на протяжении нескольких лет. Он не знал как поступить, так как не мог никому доверить магазин, пока будет в отпуске. Также он волновался, что если не будет достаточно осторожен, то производители вина и покупатель решат, что магазин им совсем ни к чему, и будут работать напрямую. Он решительно не мог этого допустить.

Инженер VLSI внимательно слушал его, и когда тот закончил, сказал, что может решить все его проблемы, разработав для него специальную схему. Сначала владелец отнесся к этому скептически, узнав что схема будет работать с помощью электроэнергии, которая была еще не до конца принята жителями города. Но инженер сообщил, что на самом деле это все очень просто, и нарисовал небольшую диаграмму на салфетке(рис. 1.1). На ней изображены два канала связи, которые должны быть синхронизированы. Один между винным заводом и магазином, другой между магазином и покупателем. Когда завод получает запрос из магазина по каналу WineryShop, он отправляет бутылку вина. Это может быть описано следующим образом:

…

Заметим, что оператор **process** подразумевает, что *winery* всегда возвращает бутылку вина. Покупатель вина, при сообщении от магазина по каналу *ShopPatron,* приходит чтобы получить вино, как описано ниже:

…

Теперь, магазин, выступающий в роли посредника, должен помимо наценки предоставить буфер для вина, позволющий заводу начать производить следующую бутылку. Это описано далее:

…

Эти три процесса вместе образуют спецификацию. Первые два описывают типы объектов, с которыми взаимодействует владелец магазина, а последний описывает поведение самого магазина.

1.3 Протоколы взаимодействия

После получения спецификации необходимо опередлить протокол взаимодействия, который реализует взаимодействие. Например, продавец отправляет “запрос” заводу для получения новой бутылки вина. Через некоторое время новая бутылка прибывает, подтверждая получение запроса. Выставив бутылку на полку, продавец отправляет покупателю “запрос” на покупку вина. Через какое-то время покупатель прибыв и купив вино, подтверждает запрос на покупку. Это может быть описано следующим образом:

…

Для построения VLSI схемы необходимо назначить провода сигналам каждой из четырех описанных операций. Два провода идут к устройству, совершающему телефонные звонки. Они называются выходами. Другой провод идет от кнопки, на которую нажимает курьер, когда доставляет вино в магазин. И последний провод идет от кнопки, которую нажимает покупатель. Эти два сигнала являются входными. Так как схема цифровая, то каналы могут находиться в двух положениях: 0(низкое напряжение), 1(высокое напряжение). Пусть описанные события обозначаются сменой состояния в положение “1”. Это описано ниже:

…

Функция **assign** используется выше для установки определенного значения сигналу. Функция **guard** ожидает, пока сигнал не достигнет этого значения. Однако в спецификации существует проблема, когда вторая бутылка вина прибывает, сигнал req_wine уже находится в состоянии “1”. Поэтому необходимо сбрасывать их значения перед началом нового цикла.

Рис. 1.2 (а) Схема сигнала двухфазного магазина. (б) Схема сигнала четырехфазного магазина.

Когда *req_wine* меняет значение с “0” на “1”, выполняется телефонный звонок, когда значение снова меняется с “0” на “1”

выполняется другой звонок. Это называется переходной передачей сигналов. Также это называется двухфазовой или двухцикловой передачей сигналов по очевидным причинам. Форма сигнала, демонстрирующая поведение двухфазного магазина изображена на Рис. 1.2(а). Другой вариант приведен ниже:

…

Данный протокол называется передачей сигналов по уровням, потому что вызов совершается когда сигнал запроса имеет значение “1”. Он также называется четырехфазовой или четырехтактной передачей сигнала, так как для завершения требуется четыре перехода. Форма сигнала, демонстрирующая поведение четырех-фазного магазина изображена на Рис. 1.2(б). Хотя этот протокол может показать более сложным, из-за вдвое большего количества переходов, чаще всего с его помощью можно построить более простую схему.

Помимо рассмотреных существуют и другие варианты проколов. В оригинальном протоколе магазин выполняет звонки на завод и клиенту, таким образом он является активным участником в обоих взаимодействиях. Завод и клиент соответственно являются пассивными участниками, просто ожидающими сигнала к действию. Другим вариантом может быть ситуация, когда завод является активным участником и оповещает магазин, когда новая бутылка вина готова, как описано ниже:

…

Аналогично покупатель также может звонить, когда он закончил последнюю бутылку вина и ему требуется новая. Но в такой случае владельцу магазина необходимо установить вторую телефонную линию.

…

К сожалению ни одна из этих спецификаций не может быть превращена в схему как есть. Давайте вернемся к первоначальному четырех-фазовому протоколу, называемому *Shop_4Phase*. Изначально, все каналы передачи сигналов имеют значение “0”, и

схема готова совершить звонок, чтобы запросить бутылку вина. После того как вино было доставлено, и сигналы *req_wine* и *ack_wine* сброшены, состояние канала снова “0”. Проблема состоит в том, что в этом случае схема должна позвонить покупателю. Другими словам, когда везде значение “0”, схема находится в неоднозначном состоянии, не зная кому звонить, заводу или покупателю. Необходимо найти решение этой ситуации. В изначальном четырехфазовом протоколе, эта проблема может быть решена путем изменения порядка в котором меняются состояния каналов передачи сигнала. Важно, чтобы вино было доставлено раньше, чем будет совершен звонок покупателю, когда сброс значения канала передачи меньше всего повлияет на работу схемы.

Рис.1.3 Схема для активного магазина

Изменения схемы, описанные ниже, позволяют всегда однозначно определить, какую операцию выполнить следующей. Также жаждущий покупатель получит звонок раньше, а результате получится довольно простая схема, изображенная ниже на Рис. 1.3

…

Можно также изменить протокол, в котором магазин пассивно ждет звонка от винного завода, но в тоже же время звоним покупателю, как показано ниже. Итоговая схема изображена на Рис. 1.4. Элемент, в середине которого изображен символ C называется C-элементом Мюллера. Когда оба входа в пребывают в состоянии “1”, его выход имеет значение “1”. Аналогично если оба входа имеют значение “0”, то на выходе получается “0”. В противном случае он возвращает свое предыдущее значение.

…

Другой интересный момент, касающийся этого протокола это, что он ожидает когда *ack_patron* примет значение “0”, хотя начинает работать он также со значением “0”. Предохранитель, в котором это условие заранее выполнено, просто пропускат дальше. Этот

предохранитель называется пустым, так как ничего не выполняет. Тем не менее, второй раз, *ack_patron* может не сброситься в этом месте. Расположение этого предохранителя позволяет сбрасывать *ack_patron* параллельно с установкой *req_wine* и *ack_wine*. Подобная параллельная работа позволит увеличить производительность системы.

1.4 Графическое представление

Перед описанием как эти схемы создаются, давайте рассмотрим альтернативные методы их изображения с помощью графов. Первый вариант представляет собой использование асинхронного конечного автомата(AFSM). Рассмотрим например протокол активный/активный, который описывался ранее(см. *Shop_AA_reshuffled*).
Он может быть представлен как AFSM, изображенный на Рис. 1.5(а), или в форме таблицы, называемой таблицей переходов Хаффмана, которая изображена на Рис. 1.5(б). В конечном автомате каждая вершина графа представляет собой состояние, а каждая дуга - переход между состояниями. Переход обозначается числом входных значений, необходимых для перехода(это числа слева от символа “/”). Числа, расположенные справа от “/” означают то, что происходит с выходными данными при смене состояний. Начиная в состоянии 0, если оба *ack_wine* и *ack_patron* равны “0”, как в начальном случае, то на выходе *req_wine* устанавливается “1”, и автомат переходит в состояние 1. В состоянии 1 автомат находится пока *ack_wine* не примет значение “1”, после этого устанавливает *req_patron* равным “1” и переходит в состояние 2. Аналогичное поведение изображено в таблице переходов Хаффмана, в которой строки - это состояния, а столбцы это входные значения(т.е. *ack_wine* и *ack_patron*). Каждая запись подписана следующим состоянием и следующий значение выходных данных(т.е. *req_wine* и *req_patron*) для заданного сочетания состояния и входного значения. Когда следующее значение равняется с текущим его обводят, чтобы обозначить что оно неизменно.

Не все протоколы могут быть описаны, с помощью AFSM. Модель конечных автоматов подразумевает, что изменения входных данных влекут за собой изменения выходных данных и соответствующего состояния. Во втором варианте(см. *Shop_PA_reshuffled*), входные и выходные данные могут меняться одновременно. Например *req_wine* может быть равен “0”, пока *req_patron* будет равен “1”. В таких случаях, чтобы изобразить поведение второго варианта схемы, изображенного на Рис. 1.6(а), можно использовать другой графический метод, называемый сеть Петри.

В сети Петри узлы графа представляют собой преобразования сигнала. Например, *req_wine+* означает, что *req_wine* меняется с “0” на “1”. Аналогично, *req_wine-* значит что, *req_wine* меняется с “1” на “0”. Дуги в этом графе представляют собой причинные отношения между преобразованиями. Например дуга между *req_wine+* и *ack_wine+* означает, что *req_wine* должен быть установлен в положение “1” перед тем как *ack_wine* сможет принять значение “1”. Маленькие кружки называются *метки,* а набор меток характеризует маркировку сети. *И*значальная маркировка представлена на рис. 1.6(а).

Для преобразования сигнала на всех подходящих к нему дугах должны быть метки. Поэтому единственным преобразованием, которое может быть совершено с изначальной маркировке, является переход *req_wine* в положение “1”. После того, как *req_wine* поменяет значение, все метки удаляются с входящих дуг, а новые устанавливаются на все исходящие. В этом случае метка на дуге между *ack_wine-* и *req_wine+* должна быть удалена, а новая метка будет расположена на дуге между *req_wine+* и *ack_wine-*, как изображено на рис. 1.6(б). При такой маркировке ack_wine может быть установлен в положение “1”, и никакие другие преобразования сигналов не возможны. После того, как *ack_wine* примет положение “1”, метки удаляются с двух входящих дуг, и ставятся на две

исходящие дуги, как показано на рис. 1.6(в). При такой разметки возможны два следующих преобразования сигнала. Либо *req_patron* примет положение “1”, либо *req_wine* примет положение “0”. Эти два преобразования могут быть выполнены в любом порядке. Остальное поведение схемы можно определить аналогичным образом.

Может потребоваться некоторое время, чтобы освоиться с сетью Петри и начать применять ее на практике. Другая графическая модель, называемая структура TEL(синхронное событие/уровень) более точно соответствует словеному описанию. TEL структура для *Shop_PA_reshuffled* изображена на рис. 1.7.