

Динамические предсказатели переходов

В обычной конвейерной реализации предсказывалось что переход не будет выполнен, и после неправильного предсказания должны быть очищены все стадии до неправильного предсказания. Динамический предсказатель условных переходов позволяет уменьшить вероятность неправильного перехода, тем самым ускоряя выполнение программы при одной тактовой частоте. Динамические предсказатели переходов отличается от статического тем что он использует историю выполнения программы для принятия решения о переходе. Адреса переходов и решение о том будет ли выполнен переход содержится в буфере целевых адресов ветвления.

Управляющий автомат будет содержать 4 состояния:

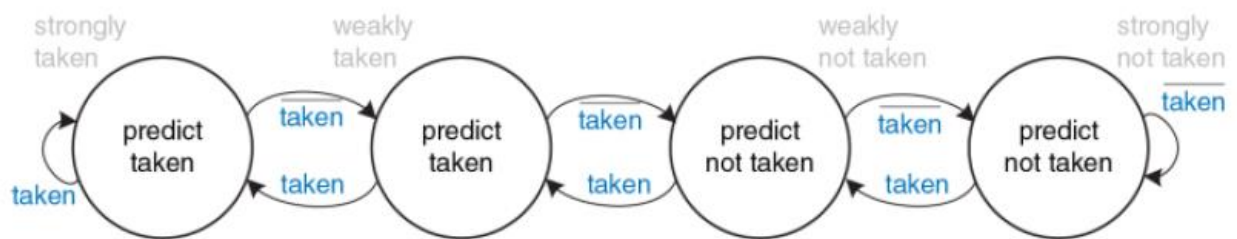


Рис 1. Диаграмма состояний двухбитного предсказателя переходов

Кодирование состояний: Наиболее правильным будет выбрать начальным состоянием то что переход скорей всего выполнится или скорей всего не выполнится, так как автомат сможет перейти сразу в одно из двух состояний.

00 – переход скорее всего не выполнится;

01 – переход точно не выполнится;

10 – переход скорее всего выполнится;

11 – переход точно выполнится;

Выходной сигнал автомата будет браться из старшего бита состояния.

Описание на Verilog в файле DynamicPredictors.v

Так же потребуется расширить мультиплексор (PC1Mux) с которого счетчик команд берет значение, и изменить управляющий сигнал.

i_mux2(outDBP[30:1]) – адрес перехода из буфера целевых адресов ветвления.

i_mux3(rNPC) – значение PC + 4 из предыдущего такта, требуется для того чтоб вернуться по правильному адресу если было предсказано что переход будет выполнен, но на следующем такте оказалось что переход не должен был быть выполнен.

Модуль DynamicPredictors.

Есть два входа для адресов:

i_addr (outIM[4:0]) – для чтения содержимого буфера, берется из стадии Fetch.

i_addrw (InstD[4:0]) – для записи данных в буфер, берется из стадии Decode.

Запись в буфер будет производиться при активном сигнале разрежения записи $.WE(nEqualBP \ \& \ (\sim StallD))$, то есть тогда когда первые две стадии не простаивают и когда значение предсказанное в стадии Fetch отличается от значения которое было получено в Decode.

$.i_next(PCSrcD)$ – сигнал который поступает на вход автомата, и является вычисленным значением будет ли состояться переход из стадии Decode.

$.i_data(outNPC)$ – данные из стадии Decode, вычисленный адрес перехода.

Прерывания.

Исключения, вызванные устройствами ввода-вывода, часто называют прерываниями.

Когда происходит прерывание в регистр EPC будет записано значение по которому процессору надо перейти для продолжению работы программы. В Cause регистр будет записана причина прерывания для того чтобы определить обработчику прерываний какую программу выполнять, адрес обработчика прерываний D0. Эти регистры являются частью сопроцессора 0. Описание на Verilog в файле EPC_Cause.v

Модуль EPC. В регистр будет записано значение которое приходит на счетчик команд по следующему такту, он же addPC, или же в него будет записано значение из стадии Decode outNPC, он же адрес условного или безусловного перехода. Второе значение требуется записать в регистр если последней инструкцией которую достали будет инструкция ветвления. Сигнал разрешения записи $EPCWrite(Interrupt \ || \ (rInterrupt \ \&\& \ PCSrcD))$,

$rInterrupt$ – значение из предыдущего такта было ли выполнено прерывание.

$rInterrupt \ \&\& \ PCSrcD$ – равен 1, когда на предыдущем такте было выполнено прерывание и поступает сигнал об том что нужно выполнить операцию ветвления.

И чтоб адрес перехода не был в данный момент в момент записан в счетчик команд, нам нужно изменить младший бит управления мультиплексором PC1Mux на такое значение ($nEqualBP \ \&\& \ \sim rInterrupt$). Адрес регистра в адресном пространстве сопроцессора0 равен 14.

Модуль Cause. В регистр будет записана причина прерывания которая поступает от мультиплексора MuxCause, разрешением записи будет сигнал Interrupt. Значение из регистра можно прочитать командой mfc0 адрес в адресном пространстве сопроцессора0 равен 13.

Модуль Status. Регистр сожержит биты разрешение прерываний и маску. Записать в регистр можно командой mfc0, адрес в адресном пространстве сопроцессора0 равен 12. Когда произошло прерывание, все прерывания должны быть запрещены, пока процессор не выйдет из прерывания, за это отвечает младший бит регистра. Поэтому нужно добавить синхронный сброс и установку. Сброс происходит по сигналу $srst(Interrupt \ \&\& \ (\sim StallF))$, установка по сигналу $sset(eret)$.

Так же нужно добавить мультиплексор PC2Mux который будет управлять сигналами, которые приходят на счетчик команд.

$.i_mux0(outMPC)$ – сигнал от PC1Mux

