

Instituto Superior Técnico – Universidade de Lisboa

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Ano Lectivo 2014 / 2015

Lisboa, 16 de Novembro de 2014

Projecto de Sistemas Digitais

**PROJECTO DE SISTEMAS DIGITAIS**

Projecto 2

**Escalonamento e Partilha de Recursos**

Grupo nº21

**David Romão Fialho,** nº 73530

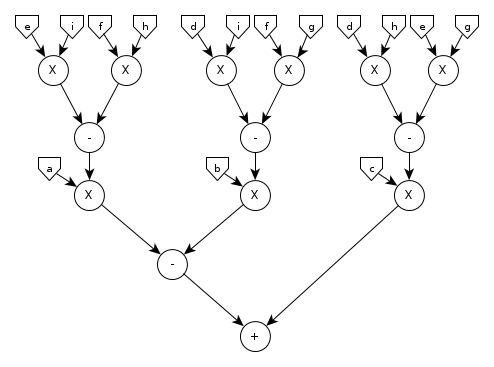
**Ricardo Filipe Amendoeira**, nº73373

**Projecto**

Pretende-se projectar um circuito digital, com duas unidades principais: Unidade de Controlo e Datapath, que permita calcular o determinante de uma matriz de 3x3 usando a seguinte relação:

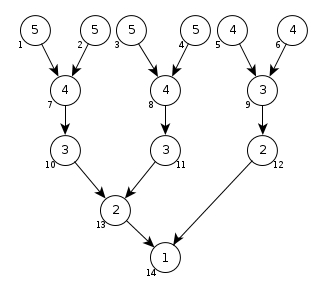
A execução deste algoritmo deve ser o mais eficiente possível, para tal é necessário que sjeam feitas o mínimo de operações aritméticas possíveis durante a execução do algoritmo. Tal pode ser conseguido colocando algumas das variáveis em evidência, minimizando-se assim o número de multiplicações:

A partir desta expressão pode contruir-se o seguinte diagrama de fluxo de dados:



Figura

Por análise deste diagrama podem ser identificadas as prioridades de cada operação, tendo como métrica a prioridade do caminho crítico. Estas são indicadas no diagrama da figura 2.

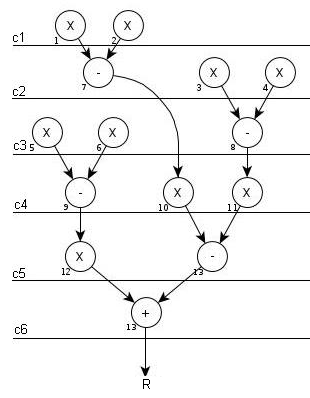
A seguinte lista de operações, ordenadas pela sua prioridade e tendo em conta as dependências de cada operação, está indicada abaixo:

|  |  |  |
| --- | --- | --- |
| Operação | | Prioridade |
| 1 | Multiplicação | 5 |
| 2 | Multiplicação | 5 |
| 3 | Multiplicação | 5 |
| 4 | Multiplicação | 5 |
| 7 | Subtracção | 4 <- 1, 2 |
| 8 | Subtracção | 4 <- 3, 4 |
| 5 | Multiplicação | 4 |
| 6 | Multiplicação | 4 |
| 9 | Subtracção | 3 <- 5, 6 |
| 10 | Multiplicação | 3 <- 7 |
| 11 | Multiplicação | 3 <- 8 |
| 12 | Multiplicação | 2 <- 9 |
| 13 | Soma | 2 <- 10, 11 |
| 14 | Soma | 1 <- 13, 12 |

Figura

Uma vez que apenas se tem disponíveis dois multiplicadores e um somador/subtractor e considerando que cada operação leva um ciclo de relógio a ser executada, isto implica que em cada ciclo, no máximo, podem ser feitas duas multiplicações independentes e uma soma/subtracção. Desta forma, seguindo a ordem definida pela lista de prioridade podem distribuir-se as operações por um mínimo de 6 ciclos de relógio, como demonstrado na figura 3:

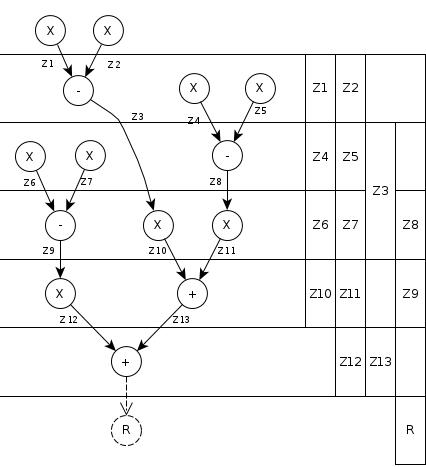
|  |  |  |
| --- | --- | --- |
| **Operação** | | **Prioridade** |
| 1 | Multiplicação | 5 |
| 2 | Multiplicação | 5 |
| 3 | Multiplicação | 5 |
| 4 | Multiplicação | 5 |
| 7 | Subtracção | 4 <- 1, 2 |
| 8 | Subtracção | 4 <- 3, 4 |
| 5 | Multiplicação | 4 |
| 6 | Multiplicação | 4 |
| 9 | Subtracção | 3 <- 5, 6 |
| 10 | Multiplicação | 3 <- 7 |
| 11 | Multiplicação | 3 <- 8 |
| 12 | Multiplicação | 2 <- 9 |
| 13 | Soma | 2 <- 10, 11 |
| 14 | Soma | 1 <- 13, 12 |



Figura

Entre cada ciclo é necessário guardar os resultados obtidos para que no ciclo seguinte se possa continuar a execução do algoritmo. Cada resultado obtido de uma operação é armazenada num dos registos disponíveis, quando um resultado deixa de ser necessário o registo fica disponível para guardar outro resultado, se necessário.

O armazenamento destes resultados está demonstrado na figura 4, onde cada resultado é identificado por um número Z’x’ e cada registo é representado por uma coluna onde, em cada ciclo, se indica o resultado que este guarda. Quando nada é indicado para um dado registo num dado ciclo isot significa que o registo está diponível.



RX1

RX2

R1

R2

Figura

A distribuição dos resultados pelos diferentes registos foi feita de forma a que fossem necessários o número minímo de registos e de forma a garantir que não fossem necessários quaisquer multiplexers nas entradas dos registos.

Cada multiplicador armazena o seu resultado num registo dedicado, o multiplicador 1 e 2 guardam o seus resultados nos registos RX1 e RX2, respectivamente. Desta forma o armazenamento destes resultados torna-se o mais simples possível. O mesmo poderia ser feito para o somador/subtractor se no ciclo 4 não fosse necessário serem armazenados dois resultados deste operador. Por essa razão, e uma vez que os registos dos mutiplicadores estão sempre ocupados, são necessários 4 registos para armazenar os resultados intermédios.

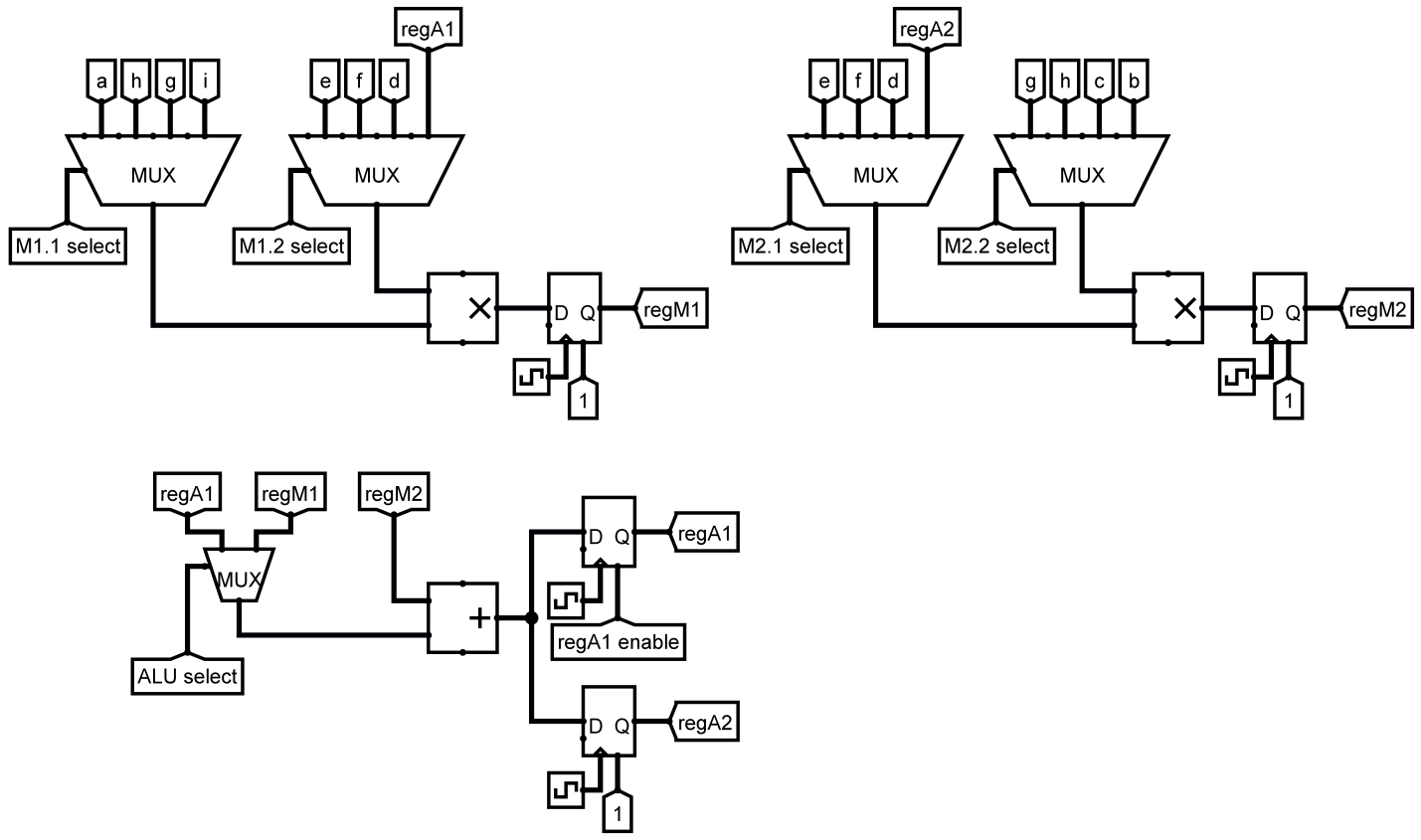
Uma vez que se tem mais do que um multiplicador é necessário identificar que multiplicador é utilizado em cada multiplicação. Esta distribuição foi feita de forma a obter o menor número possível de *multiplexers* nas entradas dos 3 operadores. Abaixo encontra-se uma tabela com as operações realizadas por cada multiplicador, estas operações são identificadas pelo número de identificação utilizado na figura 2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Multiplicador** | **Operações Realizadas** | | | | |
| Multiplicador 1 | 1 | 3 | 5 | 10 |  |
| Multiplicador 2 | 2 | 4 | 6 | 11 | 12 |

Desta forma, é possível concuir-se que cada operador tem as seguintes entradas:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Multiplicador 1** | | **Multiplicador 2** | | **Somador/Subtractor** | |
| **Entrada 1** | **Entrada 2** | **Entrada 1** | **Entrada 2** | **Entrada 1** | **Entrada 2** |
| a | e | e | g | RX1 | RX2 |
| h | f | f | h | R1 |  |
| g | d | d | d |  |  |
| i | R1 | R2 | b |  |  |

Tem-se então o seguinte diagrama de blocos para a *datapath*:



Figura

Uma vez que os registos são todos de 16 bits os valores de entrada para que possa ser garantido um resultado final correcto são limitados. Isto porque, após cada operação o resultado na saída dos operadores é sempre truncado a 16 bits, caso se verifique uma situação de *overflow* o resultado final poderá não ser o correcto.

Tendo em conta a expressão anterior:

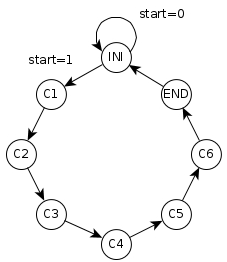
e considerando que para uma multiplicação com bits são necessários bits para armazenar o resultado e para uma soma de bits o resultado necessita de mais um bit, tem-se:

Como o sinal de saída só tem 16 bits temos a seguinte relação:

Isto quer dizer que para assegurar um resultado correcto os sinais de entrada devem ter valores dentro do intervalo [-16; 15] (5 bits com sinal). Poderam obter-se resultados correctos com entradas com maior número de bits desde que os valores introduzidos não resultem numa situação de *overflow* em nenhuam das operações realizdas.

A datapath anteriormente projectada é controlada por uma Unidade de Controlo composta por uma máquina de estados apresentada na figura 6, que define os sinais de selecção dos *multiplexers* e os sinais de *enable* dos registos:

|  |  |
| --- | --- |
| **Bits de Estado** | **Descrição** |
| RA1\_enable | *Enable* do registo A1 |
| RA2\_enable | *Enable* do registo A2 |
| RX1\_enable | *Enable* do registo X1 |
| RX2\_enable | *Enable* do registo X2 |
| X1\_select1 | Select do *multiplexer* da entrada 1 do multiplicador X1 |
| X1\_select2 | Select do *multiplexer* da entrada 2 do multiplicador X1 |
| X2\_select1 | Select do *multiplexer* da entrada 1 do multiplicador X2 |
| X2\_select2 | Select do *multiplexer* da entrada 2 do multiplicador X2 |
| A\_select | Select do *multiplexer* da entrada 1 do somador/subtractor |
| A\_control | Bit de controlo do somador/subtractor (0-soma; 1-subtracção) |
| Done | Quando activado indica que a execução terminou |



Figura

Os bits de estado para cada ciclo são indicados na tabela abaixo para que a imagem não ficasse muito confusa:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Estados:** | **INI** | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** | **END** |
| RA1\_enable | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| RA2\_enable | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| RX1\_enable | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| RX2\_enable | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| X1\_select1 | 00 | 11 | 11 | 01 | 00 | 00 | 00 | 00 |
| X1\_select2 | 11 | 00 | 10 | 10 | 11 | 11 | 11 | 11 |
| X2\_select1 | 11 | 01 | 01 | 00 | 11 | 11 | 11 | 11 |
| X2\_select2 | 10 | 01 | 00 | 00 | 11 | 10 | 10 | 10 |
| A\_select | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| A\_control | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| Done | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

A Unidade de Controlo tem como entradas o sinal de relógio, um sinal de reset e um sinal de *start*. O sinal de *start* permite iniciar a execução do algoritmo, até ser activado a máquina de estados mantém-se no estado inicial, após a sua activação a execução começa no primeiro ciclo e só termina após o 6º ciclo. O sinal de reset quando é activado interrope a execução do algoritmo fazendo com que a máquina de estados volte ao estado inicial.

Após uma modelação pós-layout o período mínimo obtido foi de 7.706 ns, o que resulta numa frequência máxima de 129.769 MHz. A latência obtida foi de 13.832 ns.

O relatório de síntese indica que foram utilizados os seguintes componentes no desenho do circuito:

|  |  |
| --- | --- |
| **Blocos** | **Número utilizado** |
| Multiplicadores 16x16-bit | 2 |
| Somadores/Subtractores 16-bit | 1 |
| Regitos 1-bit | 1 |
| Regitos 14-bit | 1 |
| Regitos 16-bit | 4 |
| Multiplexer 16-bit 4-to-1 | 4 |

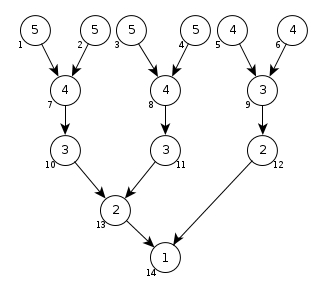
**Introdução de uma Memória independente com 2 portos de Leitura**

Considerando agora que os elementos da matriz estão armazenados numa memória com 2 portos de leitura, passa a haver uma restrição no número de variáveis que podem ser utilizadas em cada ciclo.

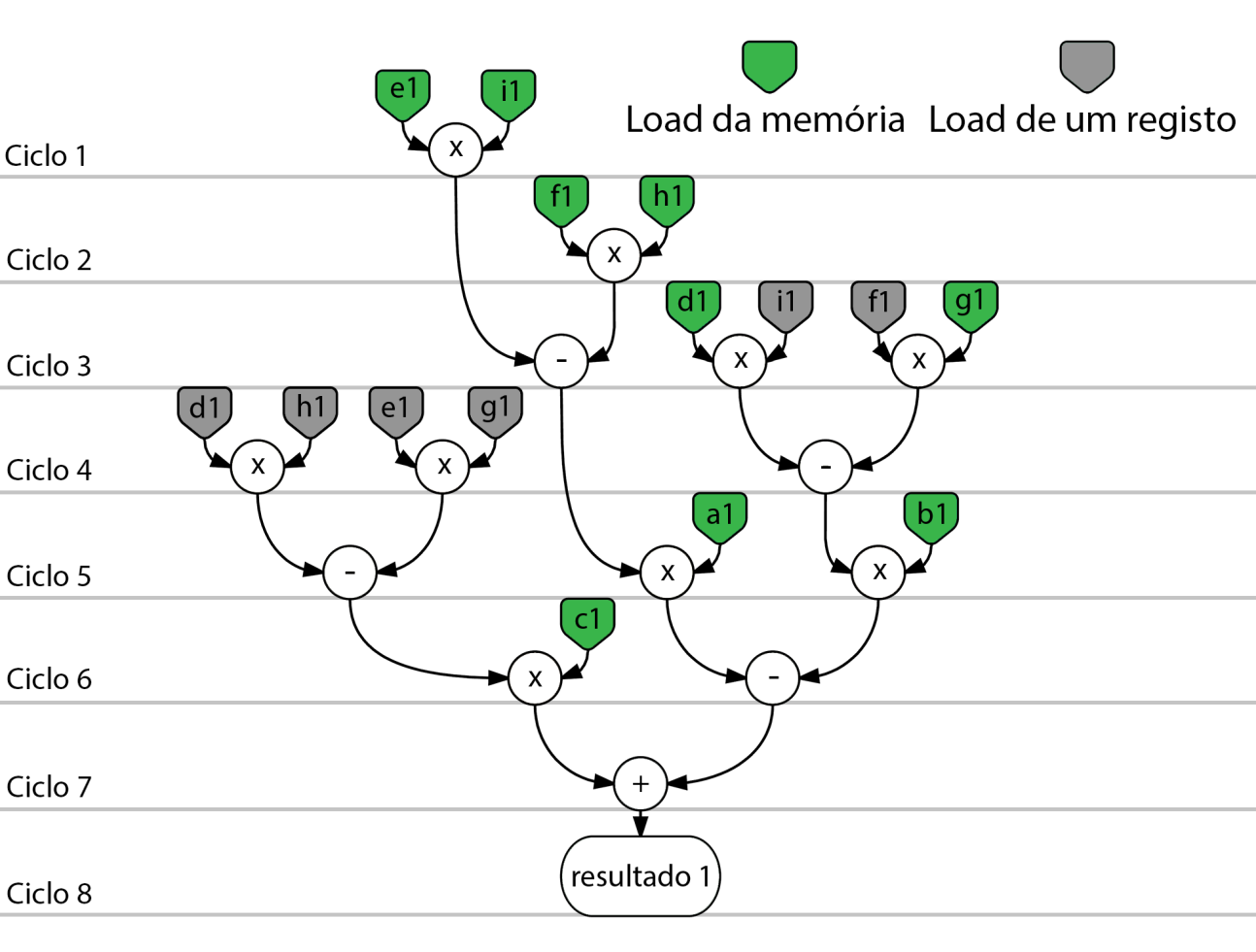
Para que a introdução desta restrição tenha o menor impacto possível na performance do circuito, pretende-se fazer o máximo de leituras da memória por ciclo (2 leituras). Para resolver este problema foram introduzidos registos adicionais que armazenam algumas das variávei em ciclos em que não são feitas as duas leituras da memória disponibilizando posteriormente esses valores sem que seja necessário recorrer à memória.

Para fazer o escalonamento deste circuito teve-se em conta a prioridade das operações e as suas dependências, que incluem resultados de operações anteriores a disponibilidade das variáveis necessárias. Desta forma, obtemos a seguinte lista de prioridades:

|  |  |  |
| --- | --- | --- |
| Operação | | Prioridade |
| 1 | Multiplicação | 5 <- e, i |
| 2 | Multiplicação | 5 <- f, h |
| 3 | Multiplicação | 5 <- d, i |
| 4 | Multiplicação | 5 <- f, g |
| 7 | Subtracção | 4 <- 1, 2 |
| 8 | Subtracção | 4 <- 3, 4 |
| 5 | Multiplicação | 4 <- d, h |
| 6 | Multiplicação | 4 <- e, g |
| 9 | Subtracção | 3 <- 5, 6 |
| 10 | Multiplicação | 3 <- 7, a |
| 11 | Multiplicação | 3 <- 8, b |
| 12 | Multiplicação | 2 <- 9, c |
| 13 | Soma | 2 <- 10, 11 |
| 14 | Soma | 1 <- 13, 12 |



A partir da lista de prioridades obtida, obtém-se o seguinte escalonamento para o circuito:



Verifica-se então que são necessários 7 ciclos de relógio para obter o resultado, mais um ciclo que sem a restrição introduzida pela memória independente. Este ciclo adicional corresponde ao facto de que o primeiro ciclo do escalonamento anterior ter que ser dividido em dois devido à limitação no acesso às variáveis. Os registos anteriormente utilizados para armazenar os resultados continuam a armazenar os mesmos valores nos mesmo ciclos, tirando o facto de que o resultado da primeira multiplicação tenha que ser guardado durante 2 ciclos.

Desta forma tem-se as seguintes entradas em cada operador, para cada ciclo:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** | **C7** | **Nº Entradas** |
| **Multiplicador 1 entrada 1** | Mem1 | X | Mem1 | Rv4 | Mem1 | X | X | 2 |
| **Multiplicador 1 entrada 2** | Mem2 | X | Rv2 | Rv2 | R1 | X | X | 3 |
| **Multiplicador 2 entrada 1** | X | Mem1 | Rv3 | Rv3 | R2 | R2 | X | 3 |
| **Multiplicador 2 entrada 2** | X | Mem2 | Mem2 | Rv1 | Mem2 | Mem2 | X | 2 |
| **Somador/Sutractor entrada 1** | X | X | RX1 | RX1 | RX1 | RX1 | R1 | 2 |
| **Somador/Sutractor entrada 2** | X | X | RX2 | RX2 | RX2 | RX2 | RX2 | 1 |

Da tabela podemos retirar que são necessários:

* 3 *multiplexers* 2-1 de 16 bits
* 2 *multiplexers* 4-1 de 16 bits

Verifica-se então que foi possível substituir dois dos *multiplexers* de 4-1 do circuito sem memória por dois *multiplexers* de 2-1.

Em resumo aumentaram-se o número de registos necessários em 4 registos e reduziu-se os *multiplexers* necessários, resultando na necessidade de 1 ciclo a mais para produzir o resultado.