Instituto Superior Técnico

PROJETO DE SISTEMAS DIGITAIS
MEEC 2018-2019

2º Projeto

Escalonamento e Reserva de Recursos

Alunos:
João Pedro Cardoso, 84096
Micaela Moraes Serôdio, 84139
Docente:
Horácio Neto

Conteúdo

1 Introdução

O objetivo deste trabalho é desenhar e desenvolver em VHDL um circuito que realiza interpolações bilineares em que os inputs estão entre [-256,255] e os output entre [-512,511]. Para tal efeito, o circuito faz as seguintes equações:

$$R_0 = \frac{Q_{10} - Q_{00}}{x_1 - x_0} \times (x - x_0) + Q_{00} \tag{1}$$

$$R_1 = \frac{Q_{11} - Q_{01}}{x_1 - x_0} \times (x - x_0) + Q_{01} \tag{2}$$

$$P = \frac{R_1 - R_0}{y_1 - y_0} \times (y - y_0) + R_0 \tag{3}$$

Para obter o resultado, é usado uma FSMD com 6 estados de execução e um caminho de dados tendo 2 Somadores e 1 Multiplicador como Unidades Funcionais. Também tem 12 Registos, 8 de Entrada e 4 registos auxiliares. Para a reutilização de recursos, a lista de prioridade usa o caminho critico como a métrica.

2 Fluxo de Dados

Na Fig. 1 encontra-se representado o fluxo de dados correspondente a uma interpolação bilinear, de acordo com o esquema apresentado no enunciado.

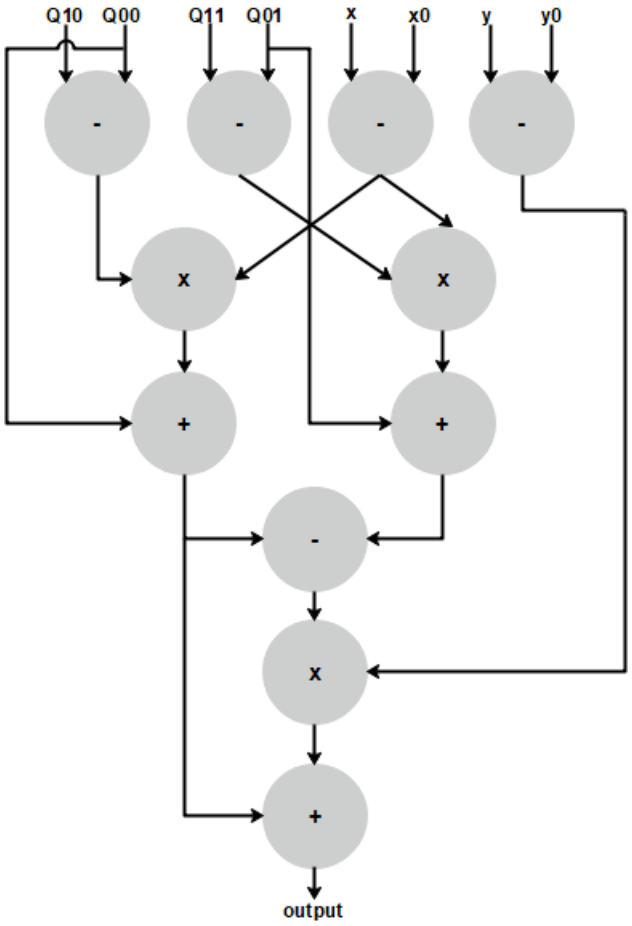

Figura 1: Fluxo de dados.

3 Lista de Prioridades

Para Executar todas as operações usando recursos limitados, foi usado o caminho critico como métrica para a Lista de Prioridades, tendo em conta o Fluxo de Dados chegamos a seguinte lista de prioridade:

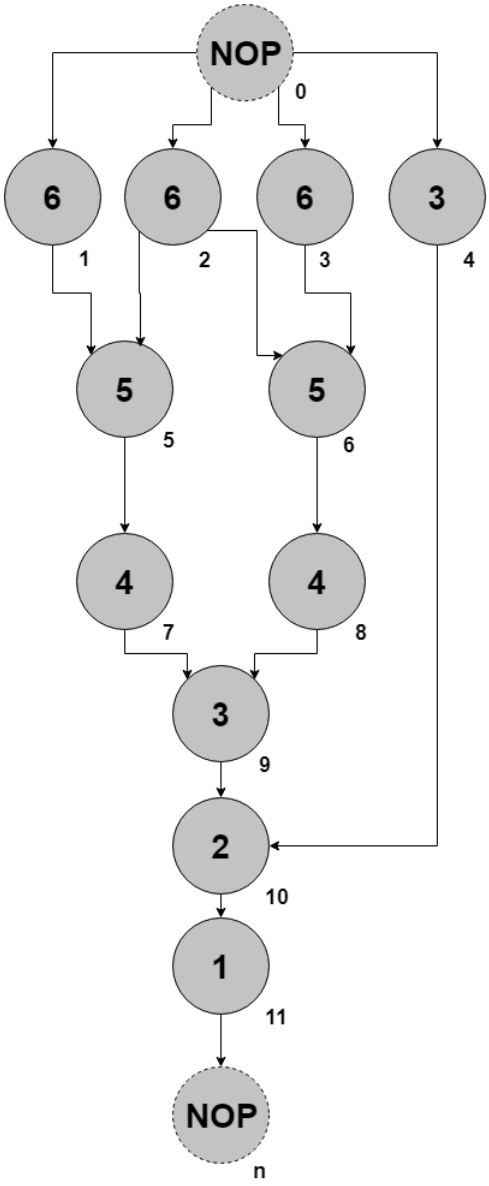

Figura 2: Lista de prioridades

Tendo em conta as unidades disponíveis, chegamos ao escalonamento com os recursos disponíveis da seguinte figura.

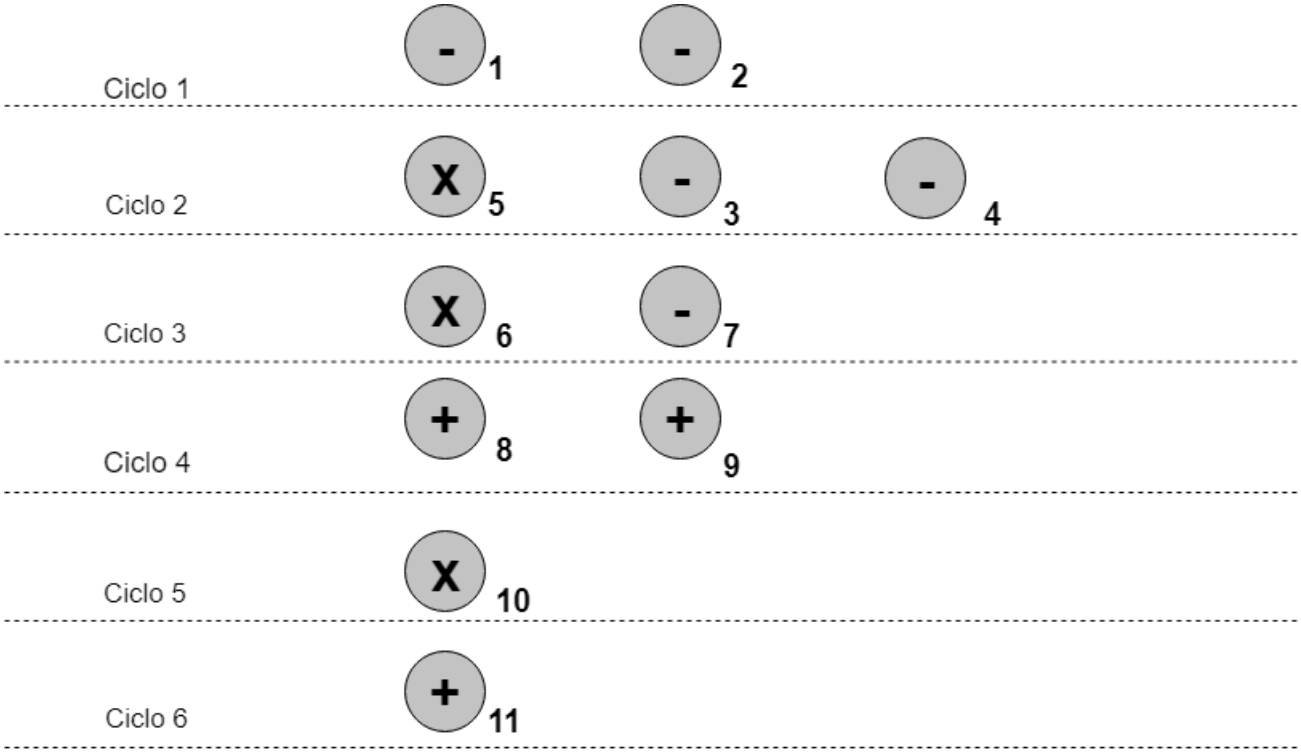

Figura 3: Escalonamento

4 Unidade de Dados

Seguindo o Fluxo de Dados e o escalonamento das secções anteriores vai ser preciso pelo menos 4 Registos Auxiliares para cálculos entre ciclos e para guardar o resultado. Como os 8 registos de entrada só são precisos para subtrações, as Entradas dos multiplexers são otimizadas para esse efeito. Para as entradas dos Adders, pode ser 4 dos registos de entrada ou 4 dos registos auxiliares. Para selecionar esses valores, usa-se $selreg1$ e $selreg2$ para o Adder1 e $selreg3$ e $selreg4$ para o Adder2. Para a entrada do Multiplicador tem 4 possibilidades dos registos auxiliares. Depois as saídas das Unidades Funcionais são as entradas dos multiplexers dos Registos Auxiliares, que escolhem o resultado a querer guardar.

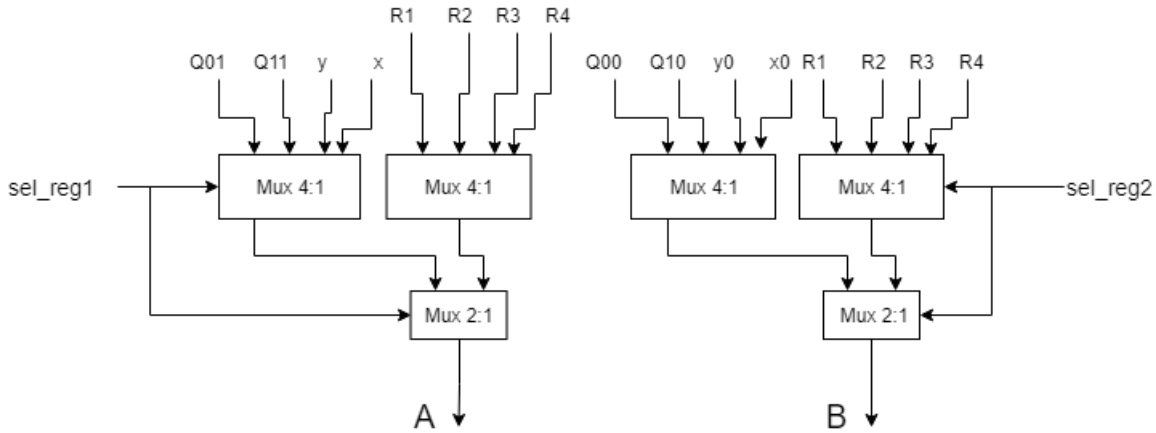

Figura 4: Multiplexer das entradas do Adder1

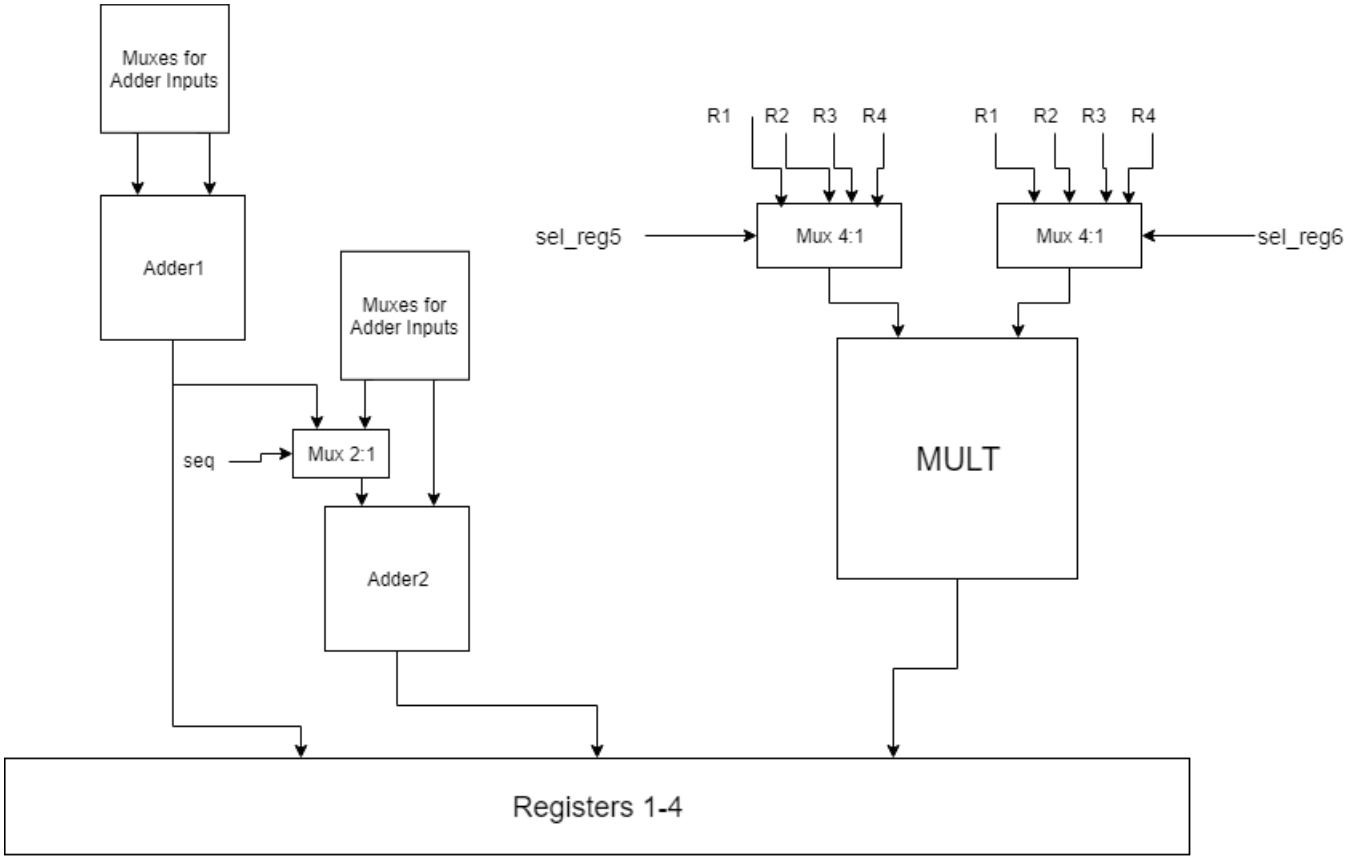

Figura 5: Unidade de Dados

5 Unidade de Controlo

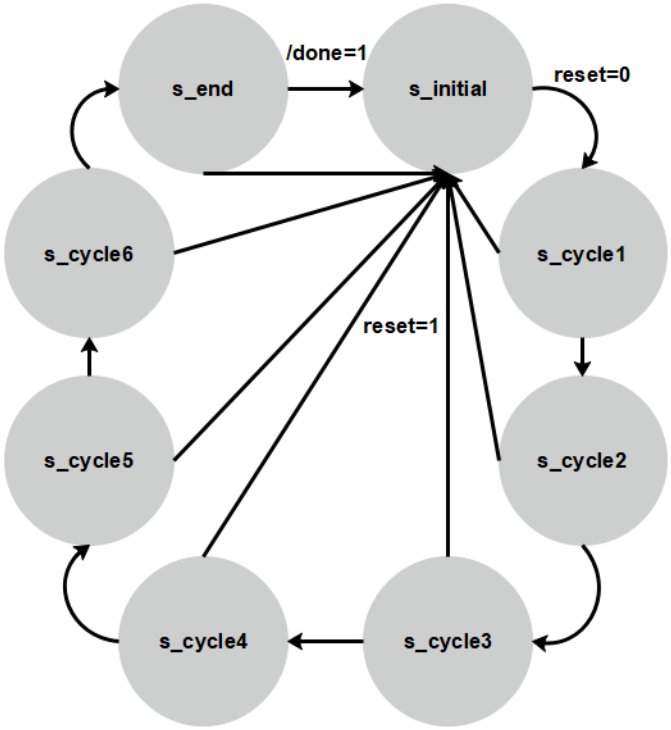

Figura 6: Diagrama de estados.

A unidade de controlo representada pelo diagrama de estados da Fig. 6, é constituída por oito estados: o estado inicial ($s\_initial$), os estados de execução (s_cycle1, s_cycle2, s_cycle3, s_cycle4, s_cycle5 e s_cycle6) e estado final (s_end).

Esta unidade é controlada por 2 inputs: o sinal de ciclo de relógio (clk) e sinal de reset (rst).

Os outputs são os selecionadores de registos para os inputs das Unidades Funcionais(Adder1, Adder2 e Mult), isto é sel_reg1, sel_reg2, sel_reg3, sel_reg4, sel_reg5, sel_reg6, os selecionadores de registos para guardar os resultados das operações sel_out1, sel_out2, sel_out3, sel_out4, os enables dos registos ($load$);, os selecionadores do tipo de operação que o somador faz (sel_add), o selecionador da truncação $trunc$, o selecionador de somas sequenciais seq e o sinal que indica que o resultado está disponível $done$.

Em seguida, uma breve descrição de cada estado de controlo:

Tabela 1: Descrição dos estados de controlo.

ESTADOS	**OPERAÇÕES**	**DESCRIÇÃO**
s_initial	————————	Estado em que se recebe os inputs, se inicializa os outputs e se aguarda por reset=0.
s_cycle1	R1 <= Q10-Q00 R2 <= x-x0	Estado em que se utilizam os 2 somadores para se efetuarem 2 subtrações paralelas.
s_cycle2	R1 <= y-y0 R3 <= Q11-Q01 R4 <= R1*R2	Estado em que se efetuam 2 subtrações paralelas, 1 multiplicação e se trunca o resultado da multiplicação.
s_cycle3	R2 <= R4+Q00 R4 <=R2*R3	Estado em que se efetua 1 soma, 1 multiplicação e se trunca o resultado da multiplicação.
s_cycle4	R3 <= R4+Q01 R4 <= R2-Adder1	Estado em que se efetua 1 soma no primeiro somador e 1 subtração no segundo somador.
s_cycle5	R3 <= R4*R1	Estado em que se efetua apenas uma multiplicação e se trunca o resultado.
s_cycle6	R4 <= R3+R2	Estado em que se efetua a soma final e se obtém o resultado da interpolação.
s_end	Output <= R4	Estado final, em que se coloca o sinal done a 1 e se obtém o resultado da interpolação no sinal de saída (output).

6 Recursos Usados, Restrições de Tempo, Latência e Frequência Máxima

Através da análise do ficheiro "Utilization Summary"do projeto implementado verificou-se que foram usados os seguintes recursos: 210 LUT's de 20800 disponíveis (1.01%); 52 FF's de 41600 (0.13%); 1 DSP de 90(1, 11%); 88 IO's de 106(83.02%); 1 BUFG de 32(3.13%). Para se definir as "Timming Contraints"criou-se um ciclo de relógio de 7.25 ns, de modo a que fosse possível efetuar o caminho crítico (controlo+mult+muxes) num ciclo de relógio e que o Worst Negative Slack fosse o menor possível de forma a otimizar a latência do circuito.

Quanto ao ficheiro "Timming Summary"verificou-se que todas as "timming contraints"foram atingidas, sendo que WNS foi de 0.019 ns, WHS 0.220 ns e WPWS 3.125 ns.

A frequência máxima é dada pelo inverso do período mínimo, que neste caso corresponde ao clock definido - Worst Negative Slack. O período mínimo é 7.25 ns-WNS, isto é, 7.231 ns. Logo, a frequência máxima é de, aproximadamente, 138 MHz.

A performance do circuito pode ser quantificada pela latência em nanosegundos. Neste caso, a latência é de 7 ciclos, isto é, são necessários sete ciclos de relógio desde que se coloca o reset a zero, até se obter o resultado válido da interpolação bilinear.

Assim, podemos concluir que a performance do circuito é dada por $7 * 7.25$ ns, ou seja, $50, 75$ ns.

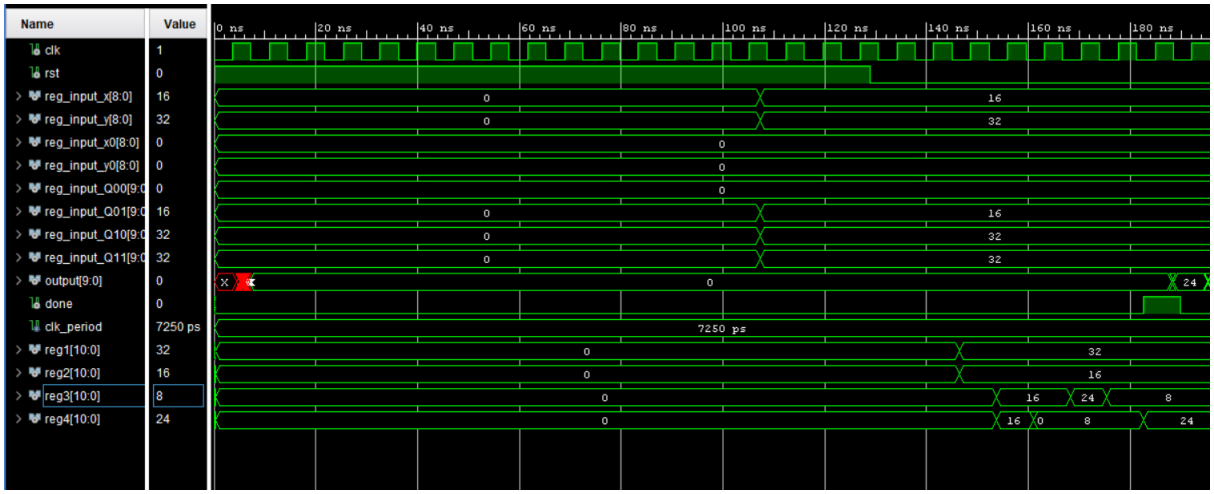

Figura 7: Simulação temporal pós-implementação.

A simulação temporal pós-implementação é a emulação mais próxima da aplicação final do "design"num dispositivo, que permite garantir que o circuito implementado respeita todos os requisitos temporais e funcionais. Através da Fig. 7 conclui-se que, para seguintes inputs Q00=0_{10}, Q01=16_{10}, Q11=32_{10}, Q10=32_{10}, x0 e y0 = 0_{10}, y=32_{10} e x =16_{10}, o output corresponde ao valor esperado (output=24_{10}). E, ainda, que o sinal que indica a validade do resultado é colocado a 1 passados 7 ciclos após o sinal reset ser colocado a 0, tal como seria de esperar.

7 Pipeline

A performance pode ser aumentada em 6x com o uso de pipelining, mas para tal seriam precisos mais recursos a nível de registos auxiliares e de Unidades Funcionais pois, haveria uma menor reutilização dos recursos.

Então considerando a arquitetura e a lista de prioridades da secção anterior:

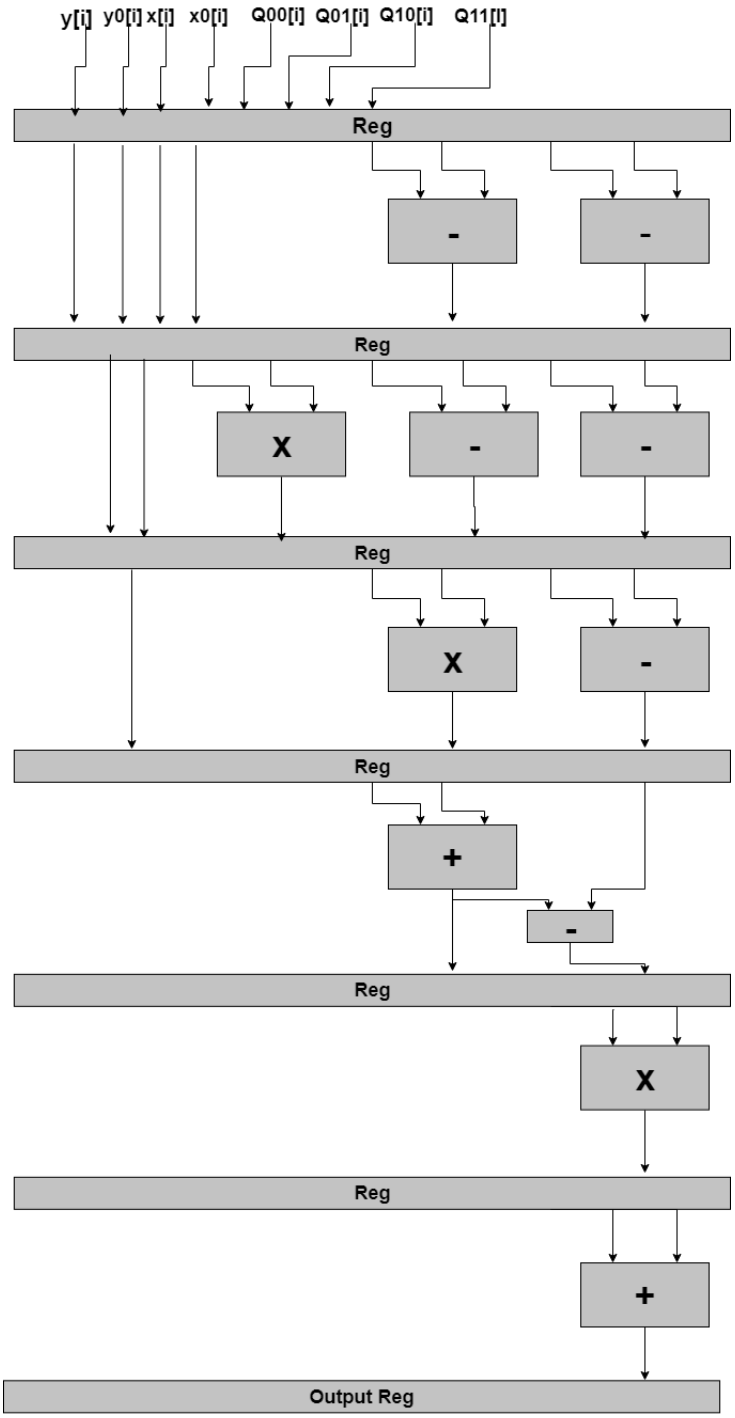

Figura 8: Dataflow em Pipeline

Para realizar tal circuito, seriam precisos: N°R=8 de Entrada+6 ciclo1+6 ciclo2+5 ciclo3 +

4 ciclo4 + 3 ciclo5 e 1 ciclo6= 33 Registos Para os cálculos no pipeline é preciso 8 somadores e 3 Multiplicadores. O clock máximo do circuito mantêm-se o mesmo mas o throughput aumenta significante-mente conseguindo 6x a rapidez do circuito com resource sharing.

8 Conclusão

Em suma, verifica-se que o circuito apresentado realiza todas as operações desejadas com recurso a apenas 2 somadores e 1 multiplicador. O circuito foi desenhado e implementado com base do fluxo de dados e na lista de prioridade utilizando como métrico. As decisões foram tomadas com o objetivo de diminuir a latência e, consequentemente, melhorar a performance do circuito. No entanto, através da análise dos recursos utilizados verifica-se que se usou quase a totalidade dos IO's disponibilizados.

Pode-se concluir também que a performance do circuito pode ser melhorada através do uso de pipeline, tal como foi analisado no capítulo 7.

Os objetivos foram atingidos, tanto no que diz respeito ao design como à simulação temporal pós-implementação, que descreve o comportamento mais próximo do comportamento real do circuito implementado num dispositivo como a FPGA.