# Multiplicação de Matrizes

VHDL, ASM, PO-PC, Pipeline e Basys2

Cindy Evelyn Peterson (219155) e Maurício Izé (273168)

### Problema

#### Entrada

- Duas matrizes 4x4
- 32 números unsigned de 4 bits

#### Saída

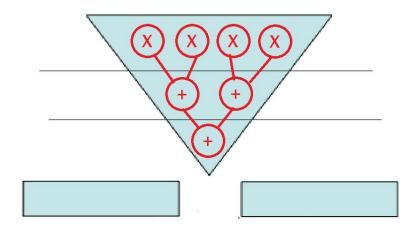
- Uma matriz 4x4 com o resultado da multiplicação das matrizes de entrada
- 16 números unsigned de 10 bits

## Implementação

Um pipeline de 3 estágios

8 entradas por ciclo de clock

- Estágio 1:
  - 4 multiplicadores de 4 bits
- Estágio 2:
  - 2 somadores de 8 bits
- Estágio 3:
  - 1 somador de 9 bits



A cada ciclo de clock é calculado um número da matriz resultante (latência = 3 ciclos)

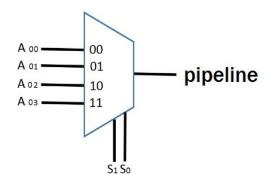
### Implementação

Os 32 números das matrizes de entrada foram armazenados em 32 signals de 4

```
bits.

SIGNAL A00 : STD_LOGIC_VECTOR(3 DOWNTO 0)
SIGNAL A01 : STD_LOGIC_VECTOR(3 DOWNTO 0)
SIGNAL A02 : STD_LOGIC_VECTOR(3 DOWNTO 0)
SIGNAL A03 : STD_LOGIC_VECTOR(3 DOWNTO 0)
```

E selecionados por 4 multiplexadores 4-1



### Métricas

#### 21 ciclos de clock

- 3 ciclos de latência
- 16 ciclos para calcular os 16 números da matriz resultante
- 2 ciclos de gerenciamento (reset, estado final)

#### Report

- Minimum period: 7.346ns
- o Maximum Frequency: 136.123MHz
- Tempo de execução: 154 ns

### Métricas

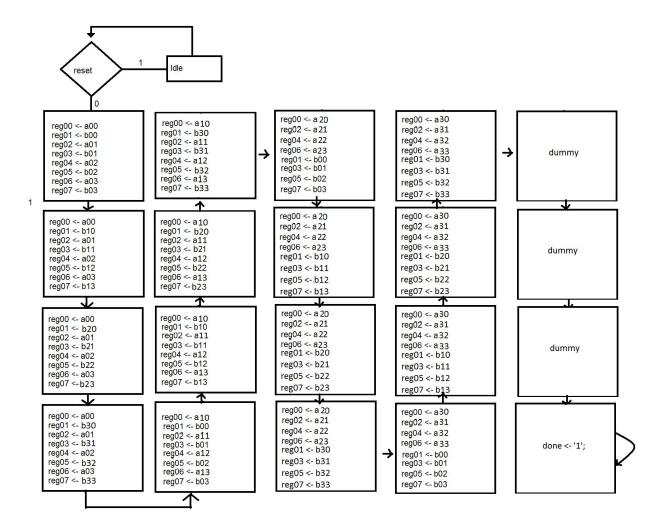
#### Área

- 4 multiplicadores (resultado em 8 bits)
- 2 somadores (resultado em 9 bits)
- 1 somador (resultado em 10 bits)
- 8 multiplexadores
- 8 registradores 4 bits (estágio de entrada)
- 4 registradores 8 bits (estágio 1)
- 2 registradores 9 bits (estágio 2)
- 1 registrador 10 bits (estágio 3)

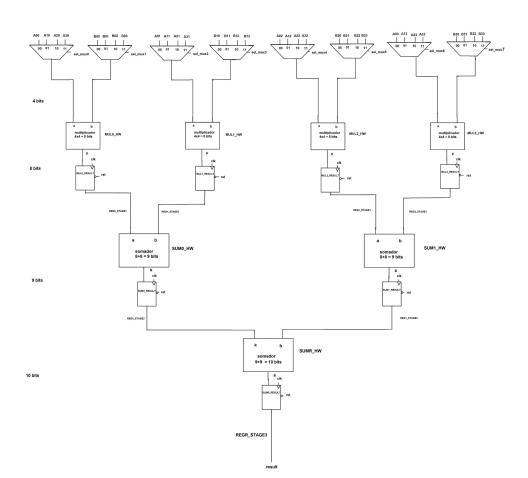
#### Report

- # Comparators: 12
- 4-bit comparator greater: 12
- o Flip-Flops: 127
- 4-bit adder: 12
- o # IOs: 33
- # of 4 input LUTs: 204

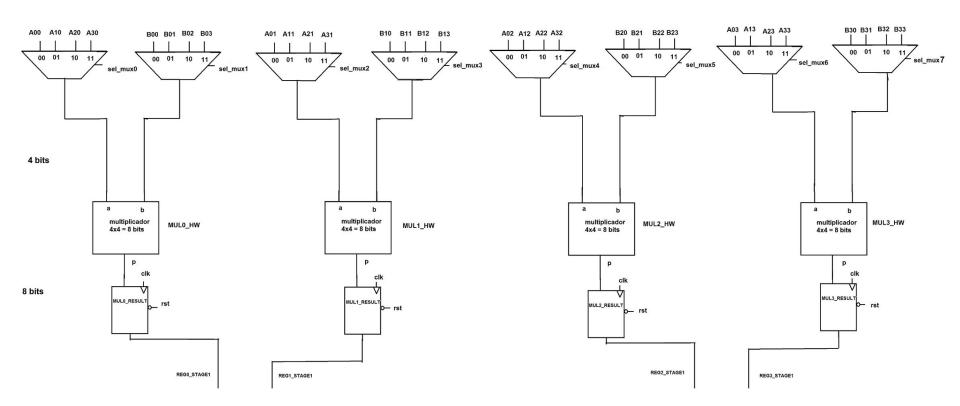
### **ASM**



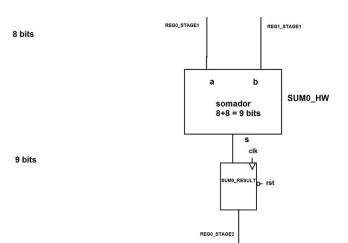
### PO

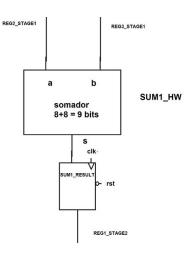


# PO - Estágio 1

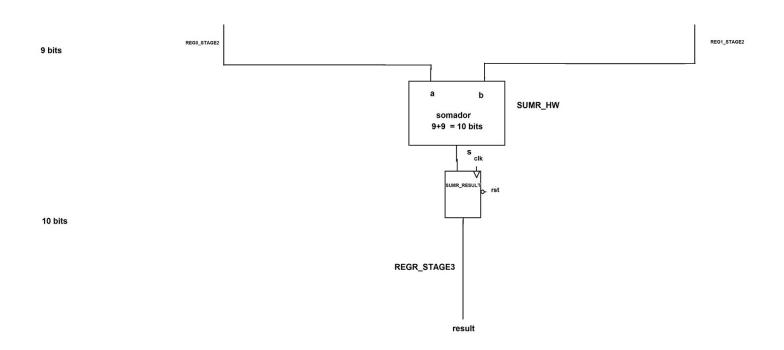


# PO - Estágio 2





# PO - Estágio 3

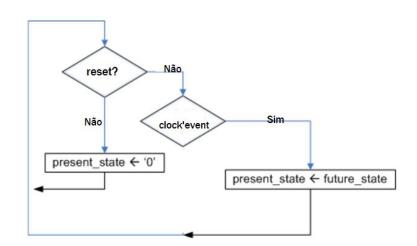


#### PC

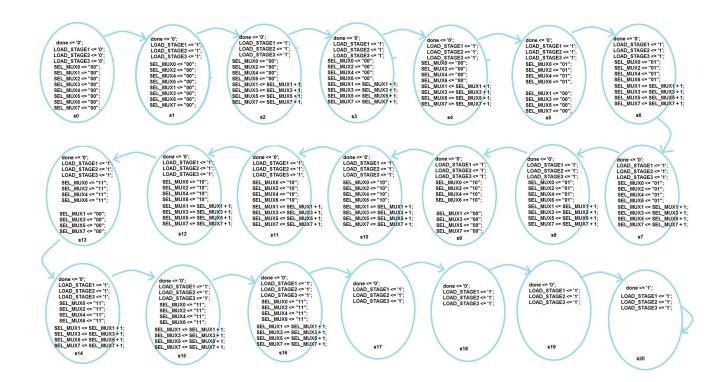
#### FSM com 21 estados

- 1 estado inicial (reset)
- 16 estados para gerenciar fluxo dos 16 números da matriz resultante (1 número por ciclo)
- o 3 estados finais 'dummy' por causa da latência inicial de 3 ciclos do pipeline
- 1 estado para armazenamento do resultado final

#### Controle da FSM

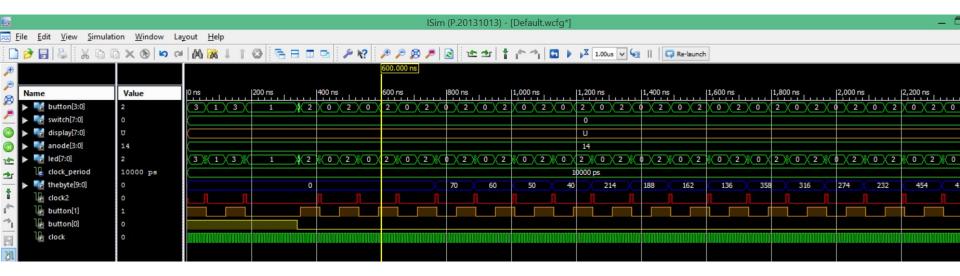


#### PC



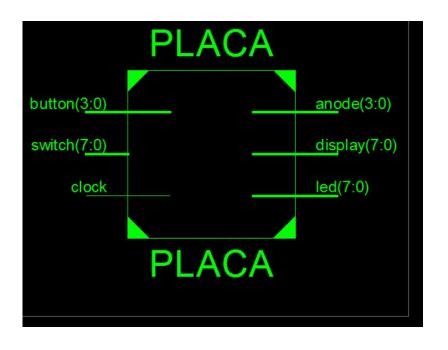
	reset é desativado																							
							s o reset	ser des	ativado, o	s multi	plicadore	s já possi	uem o re	sultado,	, mas os r	egistrado	res ainda	a não der	am o lo	ad deles				
					I	os regis	stradores	do prir	neiro está	gio cari	regam o r	esultado	da multi	plicação	0									
		os registradores do segundo estágio carregam o resultado da soma																						
								o regis	strador do	terceir	o e último	o estágio	carrega	o result	ado final	do prime	iro elem	ento da r	natriz re	esultante			matriz calc	ulada
																							19 ciclos de	clock
				16 elementos da matriz + 3 estágios do pipeline																				
Name    Name	Va -	60 ns	80 ns	100 ns	┷	120 1	S L L	140	ns X d	160	) ns	180 X 3	ns X d		) ns	X 3	ns X	240	ns ( 2	260 X	ns	280	3	300 ns
▶ <b>sel_mux6[1:0]</b> 0			(			Ŧ					1		$\equiv$		2		$\equiv$				3			
▶ sel_mux7[1:0] 0	-		0		$\Rightarrow$		2	3	X o		2	X 3	X 0		2	X 🗓	X 0		2	ХŢ			3	
mux0[3:0] 1	-					$\perp$			Х	$\perp$	5		X		9		<u>X</u>				13			
▶ ■ mux1[3:0] 1 ▶ ■ mux2[3:0] 2	-		1		_		3	4	X 1	2	X 3	X_4_	X_1_	X_2	X 3	X 4	X_1_	X	3	Х	14		4	
mux3[3:0] 5	-		5		<del>-</del>		7	8	<b>├</b>		7 7	V 8	<b>~</b>	V 4	V 7	V .	X	V 4	7	V -	14		8	
mux4[3:0] 3	-				=	Ť		Ť	$\sim$	+	7			$\stackrel{\sim}{+}$	11	^=	$\sim$	Ť		^	15		•	
mux5[3:0] 9			9		<u> </u>	10	11	12	X 9	10	X 11	X 12	9		X 11	X 12	9	(1D)	11	χ			12	
▶ ■ mux6[3:0] 4									χ		8		X		12		X				0			
mux7[3:0] 1:	3		13			14	15	0	13	14	15	X •	13	14	15	X	13	14	15	X			0	
Ū <sub>O</sub> clk 0	-																							
done o						_																		
-	0	st	)	s	:1	s2	s3	s <del>1</del>	x s5	s6	x 57	s8	× 59	s10	x11	x12	s13	s14		X s16	X s17		s19 X	s20
▶ ■ mul0_result[7:0] 1 ▶ ■ mul1_result[7:0] 1	-		1	<u> </u>	=}	2	3	4	X_5	1D	15	X 2D	X 9	1B	X 27	X 35	13	$\equiv$	39	X——			52	
■ mull_result[7:0] 1: ■ mull_result[7:0] 2:	.0		10	$\rightarrow$	₩	112	14	15	X 30	35	X 42	48	X 50	60	X 70	X 80	70	84	98	<b>∛</b> —			112	
	2		27 52	-Æ	➾	30 55	33 60	35	X 104	112	X 77 X 120	3	X 99 X 156	110	X 121	\ 132	135	150	165	^	0		180	
reg0_stage1[7:0] 0	-		0	<b>/</b> !	₹		2	#	V 4	#	V 10	)   	20	100	X 18	↑	X 36	(13)	26	X 39	v		52	
reg1_stage1[7:0] 0	-		0			1D	12	14	X 16	30	X 36	42	X 48	50	X 60	<del>\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ </del>	X 80	7	84	X 98	$\sim$		112	
▶ ■ reg2_stage1[7:0] 0	ı		0		<i>) ) 1</i>	27	30	33	X 36	63	70	X 77	84	99	X 110	X 121	132	135		X 165	ý <del> </del>		180	
▶ ■ reg3_stage1[7:0] 0			0		7/5	52	56	60	X o	104	112	120	0	156	168	X 180	X				0			
▶ ■ sum0_result(8:0) 0			0			11	14	17	20	35	46	57	68	( 59	78	97	116	83	110	137	X		164	
▶ Sum1_result[8:0]  0			0		$\longrightarrow$	79	86	93	Х 36	167	182	197	84	255	278	301	132	135	150	165	X		180	
reg0_stage2[8:0] 0			0			4	11	14	17	20	35	45	57	68	59	X 78	97	116	83	110	137	X	164	
▶ ₹ reg1_stage2[8:0] 0	-		0			4	79	85	93	35	167	182	197	84	255	278	301	132	135	150	165	X	180	
sumr_result[9:0] 0			0				90	100	110	55	202	228	254	152	314	356	398	248	218	260	302	X	344	
regr_stage3[9:0] 0			0					90	100	110	56	202	228	254	152	314	356	398	248	218	260	302		344
▶ ■ result[9:0] 0			0					90	100	110	56	202	228	254	152	314	356	398	248	218	260	302	X	344

### Simulação

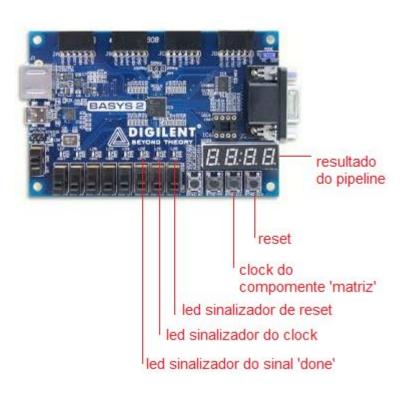


# Implementação na Basys2

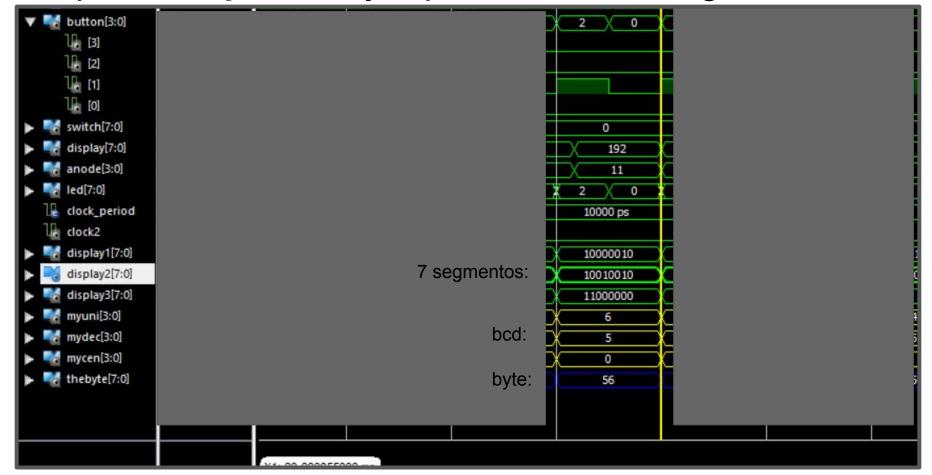


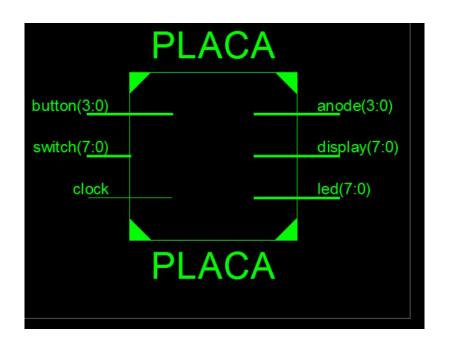


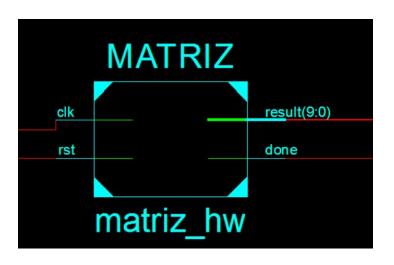
# Implementação na Basys2



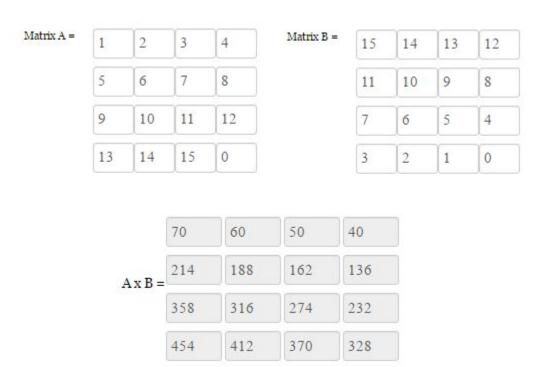
### Representação de byte para BCD e 7 seg:







# Exemplo 1



## Fim