Design and Simulation of a pipeline structured Floating Point Unit for high performance general purpose processors

Eduardo Balliriain, Martín Ignacio Falcón Faya, Pablo Slavkin
Tutor: Norberto M. Lerendegui, MEE

Abstract

This paper presents the complete design and simulation results of a floating point unit that complies IEEE 754 standard for single precision operations. The architecture of the proposed FPU, which has been named PFPU, uses a pipelined core capable of enhancing its operations flow up to 600 %. The simulation results show that processing speed presents at least a 50% improvement compared to top-of-the-line commercial systems, such as Pentium IV.

The versatility of the proposed PFPU makes it perfectly suitable for a great range of applications, like common PCs, DSPs and portable equipment (this design includes a “Power-Save” module that allows it to reduce its power dissipation from 42% up to 82%).

IEEE 754 standard

In both scientific investigations and engineering it is very common to work with magnitudes that present numerical values either extremely small or incredibly high. To be able to deal with such magnitudes, developers and engineers use the web known scientific notation, which represents a real number as a mantissa (fraction) and an exponent, which is associated to a base-10 power. This way, an electron mass can be written down as $9{,}109 * 10^{-31}$ Kg. and Avogadro's number as $6{,}022 * 10^{23}$, which makes it possible to express both numbers avoiding the need to use 35 digits for the first one and 24 for the second one. This numerical representation system covers a huge range using very few digits, mostly because all physical magnitudes can be measured with limited precision.

First digital processors where developed with a very limited mathematical capacity, based mainly in the addition of single and double precision numbers. As the evolution of processors begun to take place, enhancements in arithmetical-logical units (ALUs) leaded to the implementation of multiplication and division instructions for integer numbers as well as multiple precision operations. Floating point operations became everyday's bread with the arrival of the personal computer concept, and where successfully implemented in a hardware module separated from the main processor, called coprocessor (e.g. Intel 8087), which would work on request. Due to both floating point operations requirement and IC integration technology, coprocessors disappeared to became internal modules in all PC processors (with the INTEL 486 as the pioneer), called Floating Point Units (FPU) [P4_2].

With independence of any particular developer FPU architecture, it is white clear that it is very convenient to establish a binary coding system that can offer an important dynamic range spending few bits. This system must also be compatible with the standard processor data busses size, generally 32 or 64 bits, and provide an industry standard to allow that different companies produce interchangeable hardware. This important items where covered by a Berkeley professor named William Kahan, who's work was summarized in the IEEE 754 standard ([NRM1], [NRM2], [NRM3]).

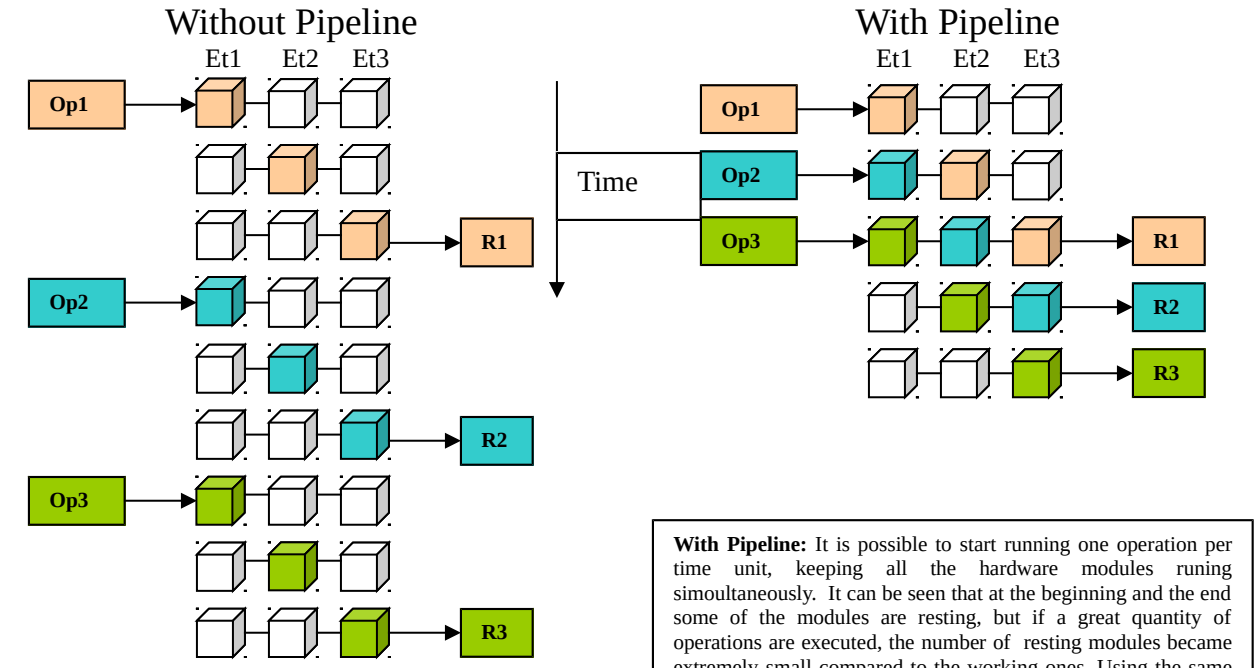
IEEE 754 standardizes two floating point numerical formats; single precision (32 bits) and double precision (64 bits). It also states that a number is organized in three sectors; Sign, Mantissa (or Fraction, which is always positive) and exponent, with 1, 8 and 23 bits assigned for single precision and 1, 11 and 52 bits for double precision. The formal expression is: $N = S.M.2^{EXP}$

The exponent must be used in a modality 'Excess 128' for single precision and 'Excess 1023' for double precision. This means that 128 or 1023 must be subtracted from the number which is stored in the exponent in order to know the real number it represents. This apparent limitation helps making simpler and faster adders, since it makes the hardware sign immune.

IEEE 754 also states that all numbers are normalized, meaning that the first number in the Mantissa is always a 1, then it is not necessary to keep it explicit, hence all hardware developers must keep in mind that all floating point numbers have the first 1 implicit, and this number corresponds to the first digit after the fraction mark.

Pipeline

One of the greatest advances in processor architecture was the incorporation of pipelining inside the stages. This concept is shown in the following figures, wich asume that 3 operations are carried in a system composed of 3 serial stages.



Without Pipeline: One operation can not start being executed until the preceding operation is completely finished, hence most of the hardware stays useless all the time (white cubes).
In this example, running 3 operations involves using 9 time

With Pipeline: It is possible to start running one operation per time unit, keeping all the hardware modules runing simoultaneously. It can be seen that at the beginning and the end some of the modules are resting, but if a great quantity of operations are executed, the number of resting modules became extremely small compared to the working ones. Using the same example as for the non-pipelined case the results are gotten in 5 time units, wich is almost half the time. When lots of operations are executed, almost all the modules will be working all the time, hence ´three times´ the work of the non pipelined system will be done.

Pipelining technique includes detecting serial stages related to the execution of an instruction and organizing the hardware in order to get all stages busy during the process, hence taking the biggest advantege possible from the system hardware.

3.1. Optimum pipeline stage number selection

The original idea of this design was to maximize the processing speed of the PFPU. The linear relationship between speed and number of stages presented so far should lead to the conclusion that an infinite number of stages should give infinite speed. However two important factors should be taken into consideration:

1) Each time a stage is added to the pipeline, a number of memory cells (latches) must be added to the hardware so the results of one stage can be stored untill the next stage has finished processing the preceding instruction. This memory cells take some time to store the results, degrading the relationship between speed and stage number worst as more stages are inserted.

2)Depending of the place chosen to ´cut´ the system while inserting pipeline stages, more or less data lines will have to be stored, giving a strong variation in the asociated hardware cost.

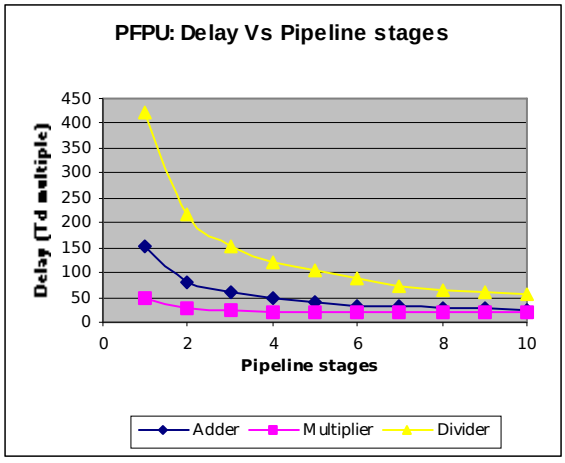
The following analysis determines optimum stage number for this design. The system performance was studied by separating it from 1 to 10 pipeline stages.

A.- Information was gathered concerning the possible places where ´cuts´ to introduce latches could be made for the ten possibilities. As this ´cuts´ would leave asimetric stages (e.g. it is not a good idea to insert latches inside a full adder, since too much hardware would be involved), the time taken by each stage differs from others, therefore the ´critical path´ concept gets very useful, defining it as the longest time taken for any stage to fullfill its task. This time, being the worst case, is then asigned to all stages equally, and is measured using transistor delay times (Td) as normalized units in order to compare the design with any comercial system desired.
B.- The quantity of gates required for each of the ten possibilities was estimated in order to get an idea of the ´hardware cost´ involved in each case.
C.- The memory cells used in this design have a time response delay of 10 Td.
D.- All collected data was normalized (forced to be between 0 and 1) so it could be used in an optimization formula (defined below).

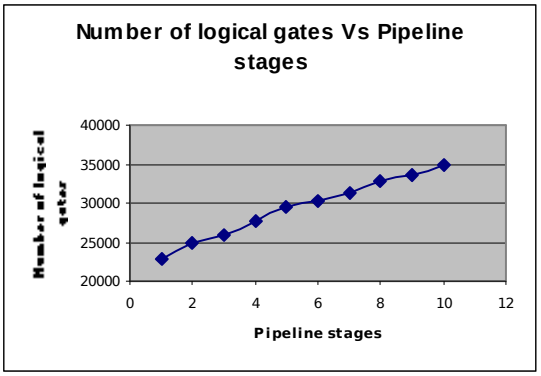
All data was sumarized in the following table:

Pipeline Stages	Operation	Critical stage Delay	Latch delay *(Td multiples)*	Total Critical delay	Gates *(without latches)*	Gates *(latches)*	Gates per operation	Total amount of gates
1	ADD / SUBST	142	10	152	2840	272	3112	
	MULT	38	10	48	6175	272	6447	22710
	DIV	412	10	422	12879	272	13151	
2	ADD / SUBST	72	10	82	2840	640	3480	
	MULT	19	10	29	6175	936	7111	24879
	DIV	207	10	217	12879	784	13663	
3	ADD / SUBST	49	10	59	2840	1265	4105	
	MULT	15	10	25	6175	1408	7583	25863
	DIV	144	10	154	12879	1296	14175	
4	ADD / SUBST	37	10	47	2840	1504	4344	
	MULT	12	10	22	6175	2456	8631	27662
	DIV	109	10	119	12879	1808	14687	
5	ADD / SUBST	30	10	40	2840	2584	5424	
	MULT	12	10	22	6175	2728	8903	29526
	DIV	96	10	106	12879	2320	15199	
6	ADD / SUBST	24	10	34	2840	2408	5248	
	MULT	12	10	22	6175	3000	9175	30134
	DIV	77	10	87	12879	2832	15711	
7	ADD / SUBST	22	10	32	2840	2832	5672	
	MULT	12	10	22	6175	3272	9447	31342
	DIV	61	10	71	12879	3344	16223	
8	ADD / SUBST	19	10	29	2840	3488	6328	
	MULT	12	10	22	6175	3544	9719	32782
	DIV	55	10	65	12879	3856	16735	
9	ADD / SUBST	18	10	28	2840	3550	6390	
	MULT	12	10	22	6175	3816	9991	33628
	DIV	49	10	59	12879	4368	17247	
10	ADD / SUBST	15	10	25	2840	4088	6928	
	MULT	12	10	22	6175	4088	10263	34950
	DIV	45	10	55	12879	4880	17759	

From this table many conclusions arise:



1.- Independently of the number of stages taken, the time used por all stages is completely defined by de divider circuit due to the fact that this circuit is much slower than the multiplier and adder. As a consecuence of this both the adder and multiplier will have idle time.
2.- The way that stage time falls as the number of stages is increased is **not linear**. This is due to the fact that the shorter the stages become (as the number of them is increased) the stronger the effect of the constant delay of the latches gets (10 Td per latch). It is seen in the table that for 10 stages 22 % of the delay is fixed by the latches.



3.- The number of logical gates used grows linearly with the number of pipeline stages to implement (because it was chosen very carefully the places where the circuit was cut). This conclusion is quite important, since it says a lot about the circuit scalebility.

The main conclusion of the analysis done before is that the botleneck of the system is the Divider block. Since there are many aplications in which this block is not required, tipically DSPs, a special mode of the design is taken into account; this mode, named DSP, has no divider block and its botleneck, which can be found by calculating again its critical path, is the adder block (as can be seen in the table above).

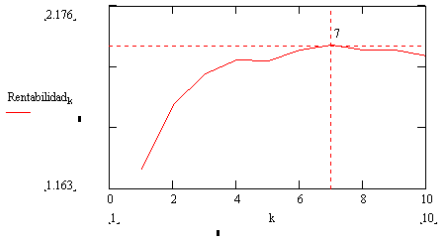
E.- Since a decision criterion had to be chosen in order to get a good relationship between speed and hardware cost, and the graphs shown before have a linear and a non linear relationship between the variables, the idea of having a rentability function which could be searched for a maximum appeared. The general function defined was:

$$\text{Rentability}_k := \frac{1}{(1-P)\cdot \text{NormGates}_k + P\cdot \text{NormDelay}_k}$$

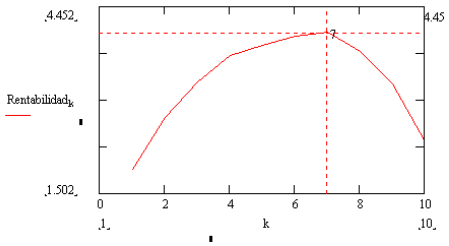
In which the k parameter is the number of pipeline stages [1, 2 .. 10] and the P parameter [0 .. 1] is the normalized importance that is given to the speed of the system (and 1-P is the importance given to the hardware cost). In this design it was decided to give the speed of the system a 60 % of importance, hence giving the hardware cost the 40 % remaining.

$$\text{Rentability}_k := \frac{1}{0.4\cdot \text{NormGates}_k + 0.6\cdot \text{NormDelay}_k}$$

When the formula is ploted as a function of the number of stages:



PFPU Rentability



DSP mode PFPU Rentability

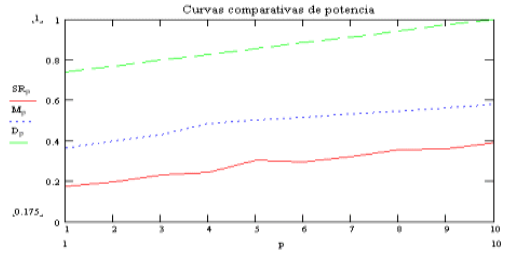
It is evident for both normal and DSP mode that the maximum rentability appears for a 7 stages pipeline, being this number of stages the chosen for this design.

From this data it is possible to calculate the maximum speed for the PFPU, wich will be the inverse of the time taken by the critical stage plus the time taken by the storing cells. If a technology of 130 nm is used (same as Pentium 4), which has a delay time of 1.5ps = Td, the speed that can be obtained is:

1/(number of transistos in the critical path * Td) = 9.009 GHz for the PFPU and 26.975 GHz for the PFPU in DSP mode.

6. Dissipated Power

The dissipated power is a very important charasteristic of the PFPU, since it can be the discriminating factor when it comes to deciding if the PFPU can be implemented for portable systems. When the number of pipeline stages is increased, more logical gates are inserted in the circuit and more power is dissipated each time this gates commute at the clock frecuency. The dissipated power as a function of the pipeline stages number is:



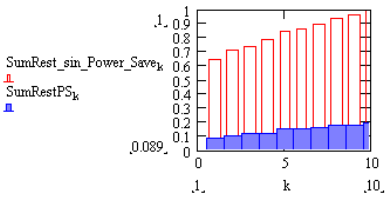
Normalized (compared with the divider) power dissipation graph

Green (- -): Divider
Blue (. .) multiplier
Red(__) : Adder

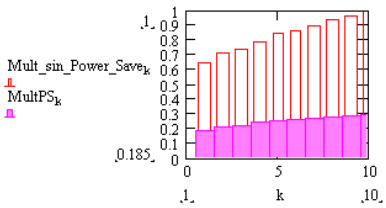
It can be seen in the figure that this functions are practically straight lines with almost the same (small) slope, which makes them be a very poor limit for the growth in pipeline stages number.

It also seems that if a large cuantity of the same operations appeared in the hardware, then only one of the operations block would be doing a useful task, while the others would be idle wasting power (e.g for 7 stages if lots of additions are being runned then 0.3 normalized power is being usefully used out of 0.3+0.5+0.9=1.7 normalized power !). So it would be a great idea to turn off this idle blocks in order to save power (batteries in portable systems).

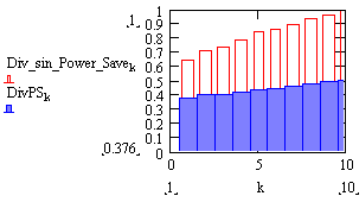
If any operation is executed in bursts while the modules iddle are ´turned off´, which will be accomplished by the addition of a new hardware module named ´Power-Save´ , then the improvement over the PFPU without this module is shown next.



Enhancement in Adder



Enhancement in Multiplier

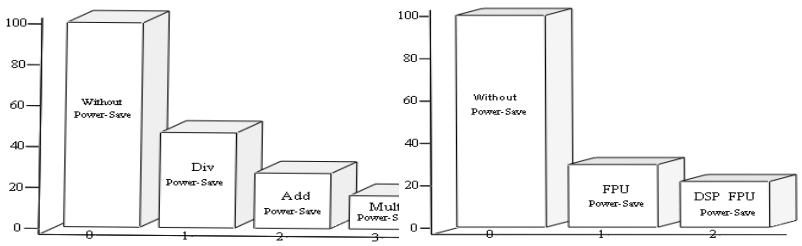


Enhancement in Divider

For this concrete design, where a 7 stages pipeline is required, the power saving rate for each instruction is:

1.- PFPU power dissipation with Power-Save for single operation executed in burst compared to PFPU without Power-Save module.

2.- PFPU power dissipation while running burst operations supposing that each operation runs 33 % of the time for the PFPU and 50 % of the time for the DSP mode PFPU



The dissipated power depends mostly on the technology used for the PFPU (parasit capacity of the MOS transistor), the supply voltage, the frecuency and the number of transistors that are currently working. All this factors are related as follows ([PW1], [PW2]):

$$\text{Dissipated\_Power} := \text{Transistor\_number} \cdot \text{Frecuency} \cdot \text{C\_Trans} \cdot \text{Supply\_V}^2$$

If this design where to be phisically implemented today, the technologie which would be used would be 13nm, same as for Pentium IV and Athlon XP processors, which are the top of the line processors for the PC family. In order to get reliable data for this technology, a simple aproximation was chosen, based on the actual Pentium IV datasheet.

Pentium IV: Frecuency: 2.5 Ghz; Power: 54.7 W; Supply Voltage: 1.5V; number of transistors: 55 million [P41]

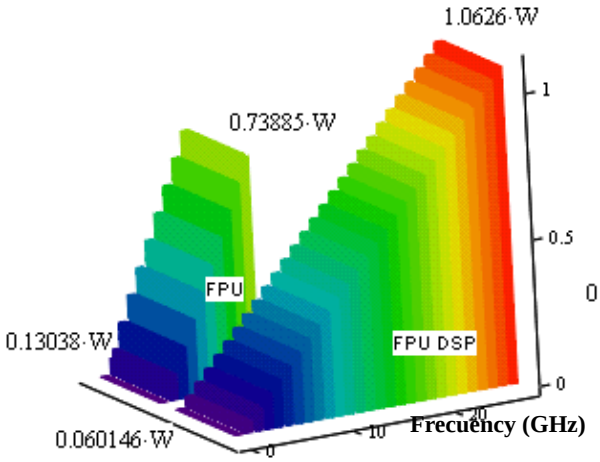
From this data it is easy to get the relationship between power per GHz per transistor (asuming the same supply voltage)

$$P_{P4} = \frac{\text{Potencia}}{\text{Frec} \cdot \text{Ntrans} \cdot V^2} = 1.778 \times 10^{-7} \cdot \frac{\text{Watt}}{\text{Ghz} \cdot \text{Transistor} \cdot \text{Volt}^2}$$

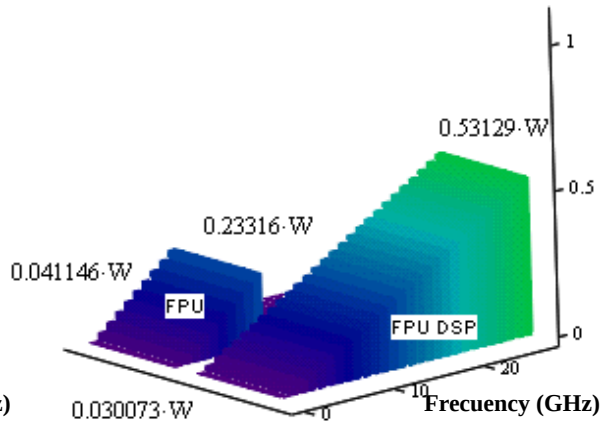
wich will be used to calculate the worst case power dissipation for the PFPU (all transistors of the corresponding stage working)

PFPU: Frecuency:1.5 GHz; number of transistors: 130 Thousand

Then the dissipated power as a function of frecuency is:



Dissipated power for non burst operation
Power-Save Deactivated



Dissipated Power for burst operation
Power-Save Activated

From this data it can be seen that if a Pentium IV 2400 MHz processor has a 2.933 GFLOP floating point operations throughput, dissipating 54.700 W and the PFPU has a 2.5 GFLOP floating point operation throughput at a frecuency of 2.5 GHz (due to the pipeline) dissipating 41mW then the rate calculation power / dissipated power of the PFPU is aproximately 680 times higher than the one for Pentium IV.

7. Simulations

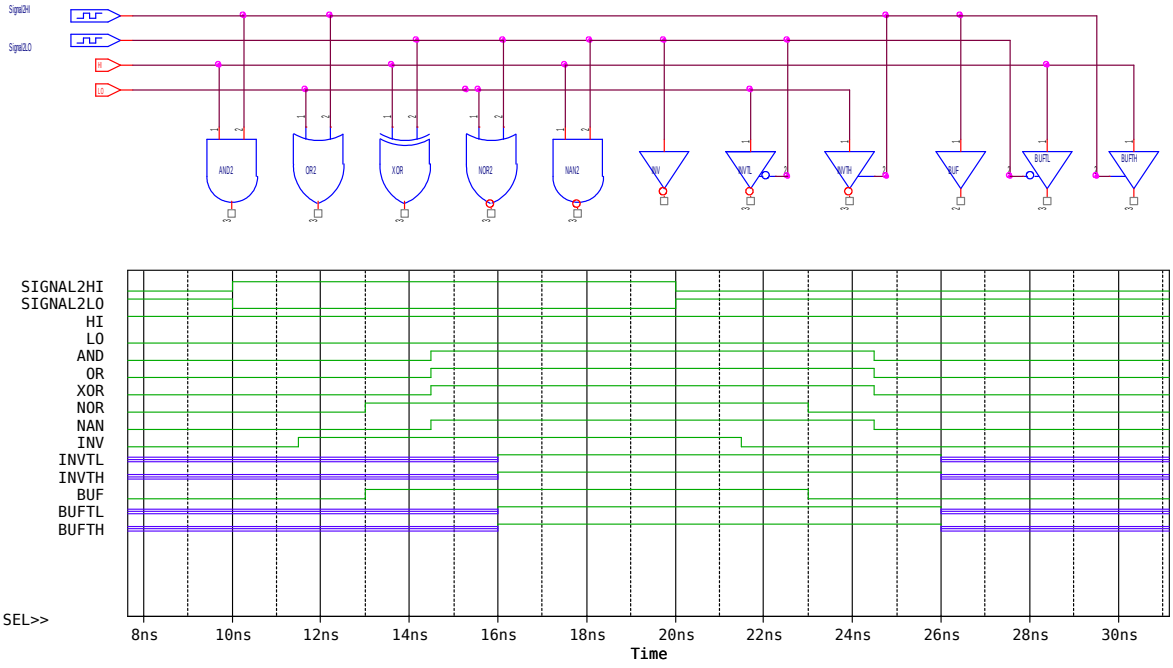
While simulating the system, several efects concerning the physical structure of the logical gates were taken into account, including them into ORCAD´s PSPICE parameters in order to gurantee a better aproximation to real-world designs. This efects will be descrived and shown in a brief Pspice simulation in order to make them explicit.

Delay time

It´s the time it takes the signal to propagate through the logical gate and present a stable value at it´s output.

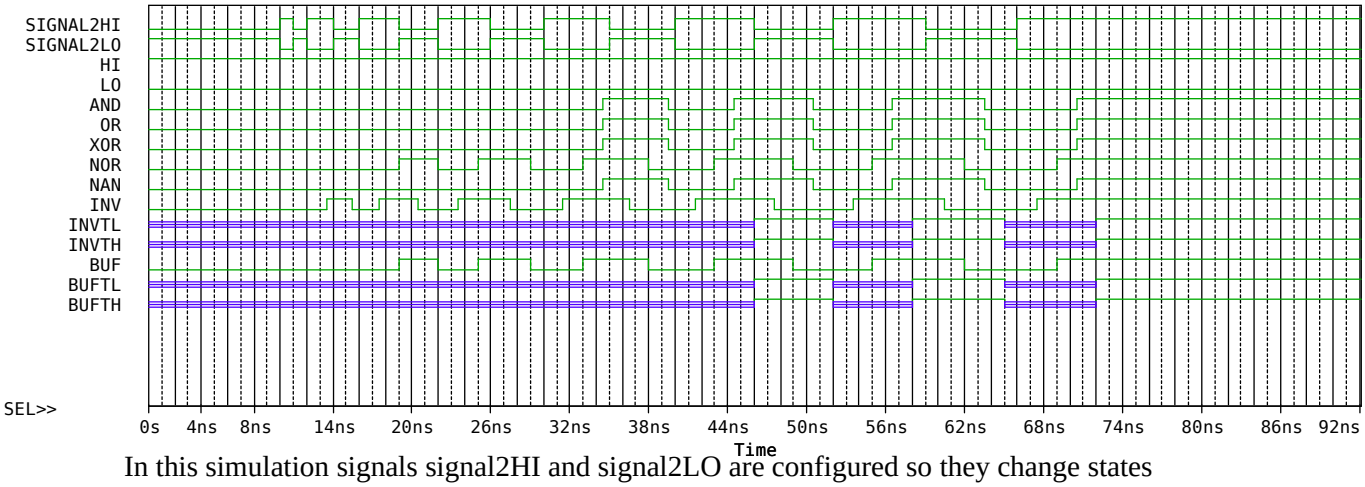
Logical Gate	Maximum Time [ps]
AND	4,5
OR	4,5
XOR	4,5
NOR	3
NAND	4,5
INV	1,5
INVTL	6
INVTH	6
BUF	3
BUFTL	6
BUFTH	6

Due to a limitation in Pspice time scale resolution, pseconds are not correctly managed, hence it was decided to scale down from pseconds to nseconds. This consideration only affects the way the scales are read, since the person reading it must remember that each time nanoseconds appear it really refers to picoseconds. Taking this into account it is possible to understand the following simulation, in wich different gates were exited at time=To in order to make explicit their delay times.



Energy

All Digital gates need a minimum quantity of energy applied to their inputs in order to get a stable change of state in their ouputs. As this energy depends of the voltage and duration of the input signal, and the voltage is fixed by the technology used, the controlled parameter is the time the signal is present at the gate´s input. If this time isn´t large enough then the gate´s will not produce any change in their output. This simulation shows how as the energy in the input of logical gates grows (the input signal lasts longer) more gates start working as their minimum requirements of energy are fulfilled.



In this simulation signals signal2HI and signal2LO are configured so they change states progressibly from 1 ps to 7 ps. As this time grows (and the signal energy grows) the gates that require less energy start ´following´ the changes in their inputs. When the input signal times are larger than 6 ps the gates that need more energy (Tristate gates) start following their inputs as well.

The following simulation tkes the following facts into account:

- The most time-restrictive stage in the PFPU is the divider, which uses 61 Td. This value must be added to the 10 Td it takes the latches to ´remember´ the output data. Using 130 nm technology, the critical time is found as Tc=(61+10) *1.5ps=106.5ps. In order to produce a more robust system, it was decided to add a bit more time to Tc, finally choosing 120 ps, wich gives 8.3 GFLOPS of calculation power.

Input Data

To generate the input data for this simulation, an ANSI C program was designed, which generates floating point random numbers and operations that change every 10 clock cycles. This program´s output is then feeded into another program´s input. This new program´s purpouse is formatting the data so that ORCAD can understand it and also execute the operations listed by the first program so the results obtained in the ORCAD simulation can be compared to know if they are correct. Since the data is given in both decimal and hexadecimal format, this second program also implements a conversion algorithm.

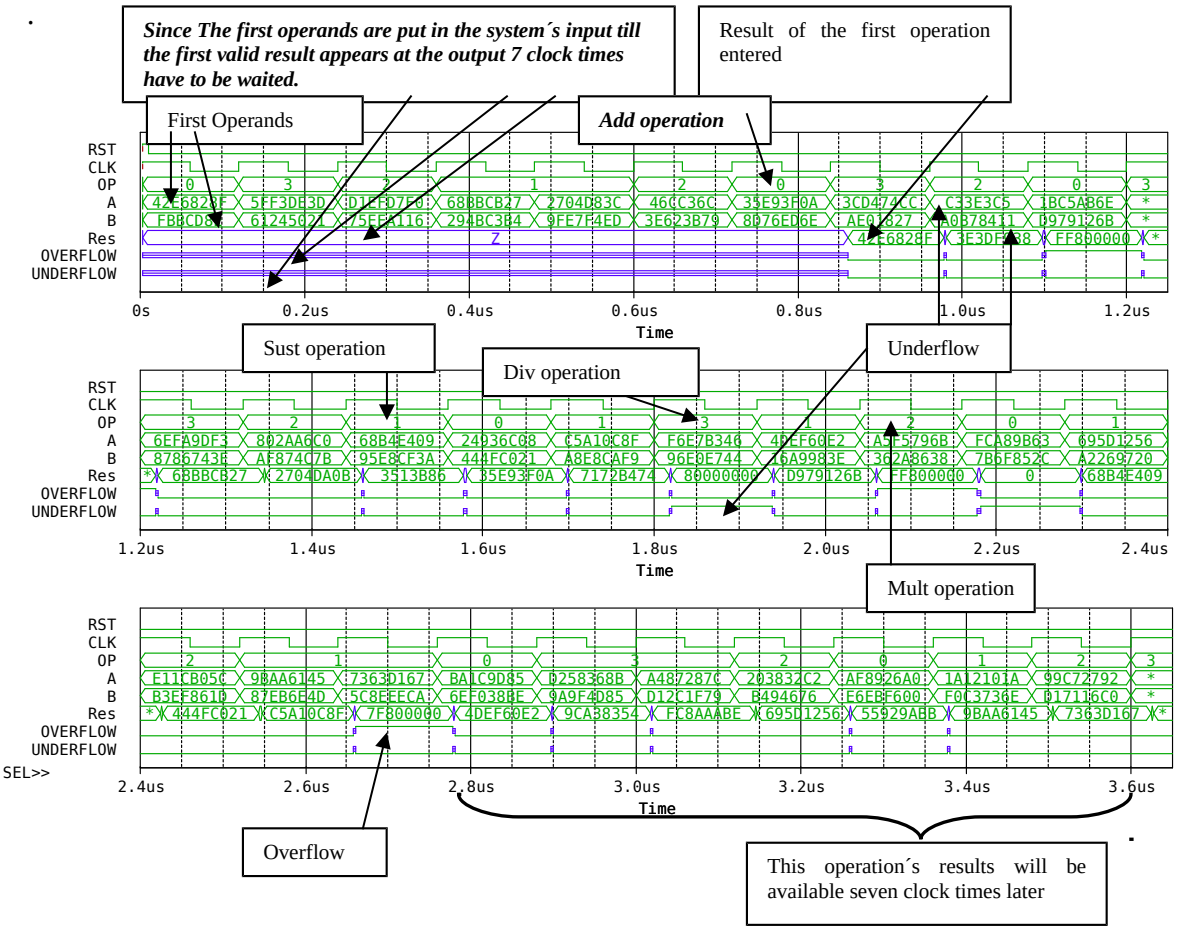
Output Data

Pspice simulation´s results are stored in a .DAT format, in wich all signals are stored. Since this format is coded Pspice interface was used in order to study this signals and export them to an hexadecimal format file. Using this two methods it was possible to verify that the output of the designed hardware was correct compared to the output of the program writtenin ANSI C.

Simulation

There are several input signals that can be identified, this are:

- **Rst:** Asincronic reset signal: it erases all registers on user´s demand
- **Clk:** Master system clock
- **Operador A:** Operator A: In case of sustraction it behaves as the minuend and in case of division it behaves as divisor.
- **Operador B:** Operand B: In case of sustraction it behaves as the sustraend and in case of division it behaves as dividend.
- Operation Identifier (**OP):** formed by two bits, which produce four different operations when demultiplexed.
- Result (**RES):** This is a 32 bit bus wich outpus the operation resut that complies IEEE 754 standard.
- Underflow (**UNDER):** Single bit that goes to 1 in case of underflow. It can be ignored by the following hardware if wished.
- Overflow (**OVER):** Single bit that goes to 1 in case an exponent overflow appears.



At the begining of the analisis it can be observed that the result signal appears in blue, meaning that the output is in tristate mode. This happens because a useful result will appear 7 clock Cycles later, because of the pipeline structure natural latency. From those seven cycles on it can be seen that the result stabilizes in a value that corresponds to the 1rt operation inserted. This can be also seen in the following table, which is another way of presenting the simulation results.

Underflow and overflow signals are also quite important, since they announce that some operation´s result has gone out of representation range of the system either because it´s too small or big respectively.

The following table expoundes a comparison between the ANSI C program results and those found at the simulation output. It can be noticed that some of them don´t match exactly, and that difference arises in the least significative digit (LSD) of the mantissa. This difference is produced by the normalization circuit, which has to shift the results several times to comply with IEEE 754 standard, and fails to recall the last bit in some cases due to the lack of preservation bits.

Computadora								PSpice		
N° op.	Tiempo	A	Op	B		Resultado		Tiempo	Resultado	Flags
0	0,00E+00	1.15255e+02	+	1.85188e-29	=	1.15255e+02		1ns	ZZZZZZZZZ	
0	0,00E+00	42e6828f	+	0fbbcd8e	=	42E6828F		861ns	42E6828F	
.								978ns	ZZZZZZZZZ	
1	1,20E-07	3.51451e+19	/	1.89440e+20	=	1.85521e-01				
1	1,20E-07	5ff3de3d	/	61245021	=	3e3df938		981ns	3E3DF938	
.								1.098us	ZZZZZZZZZ	
2	2,40E-07	-1.28765e+11	*	6.04997e+32	=	Overflow				
2	2,40E-07	d1efd7f0	*	7feeea116	=	Overflow		1.101us	FF800000	Overflow
.								1.218us	ZZZZZZZZZ	
3	3,60E-07	7.09464e+24	-	4.52448e-14	=	7.09464e+24				
3	3,60E-07	68bbcb27	-	294bc3b4	=	68bbcb27		1.221us	68BBCB27	
.								1.344us	2000CA03	
4	4,80E-07	1.84359e-15	-	-9.82375e-20	=	1.84369e-15				
4	4,80E-07	2704d83c	-	9fe7f4ed	=	2704da0c		1.347us	2704DA0B	
.								1.458us	ZZZZZZZZZ	
5	6,00E-07	2.78314e-36	*	2.20930e-01	=	6.14879e-37				
5	6,00E-07	045cc36c	*	3e623b79	=	03513b86		1.461us	03513B86	
.								1.578us	ZZZZZZZZZ	
6	7,20E-07	1.73782e-06	+	-7.60904e-31	=	-1.59348e-33				
6	7,20E-07	35e93f0a	+	8d76ed6e	=	35e93f0a		1.581us	35E93F0A	
.								1.698us	ZZZZZZZZZ	
7	8,40E-07	2.59346e-02	/	2.15795e-32	=	1.20182e+30				
7	8,40E-07	3cd474cc	/	0ae01827	=	7172b475		1.701us	7172B474	
.								1.818us	ZZZZZZZZZ	
8	9,60E-07	1.38582e-31	*	-3.10888e-19	=	-0.00000e+00				
8	9,60E-07	0c33e3c5	*	a0b78411	=	80000000		1.821us	80000000	Underflow
.								1.938us	ZZZZZZZZZ	
9	1,08E-06	3.27017e-22	+	-4.38172e+15	=	-4.38172e+15				
9	1,08E-06	1bc5ab6e	+	d979126b	=	d979126b		1.941us	D979126B	
.								2.058us	ZZZZZZZZZ	
10	1,20E-06	3.87811e+28	/	-2.02304e-34	=	Overflow				
10	1,20E-06	6efa9df3	/	8786743e	=	Overflow		2.061us	FF800000	Overflow
.								2.178us	ZZZZZZZZZ	
11	1,32E-06	-3.91691e-39	*	-2.46107e-10	=	0.00000e+00				
11	1,32E-06	802aa6c0	*	af874c7b	=	00000000		2.181us	00000000	Underflow
.								2.298us	ZZZZZZZZZ	
12	1,44E-06	6.83386e+24	-	-9.40310e-26	=	6.83386e+24				
12	1,44E-06	68b4e409	-	95e8cf3a	=	68b4e409		2.301us	68B4E409	
.								2.424us	4004C001	
13	1,56E-06	6.39341e-17	+	8.31002e+02	=	8.31002e+02				
13	1,56E-06	24936c08	+	444fc021	=	444fc021		2.427us	444FC021	
.								2.544us	44010001	
14	1,68E-06	-5.15357e+03	-	-2.58452e-14	=	-5.15357e+03				
14	1,68E-06	c5a10c8f	-	a8e8caf9	=	c5a10c8f		2.547us	C5A10C8F	
.								2.658us	ZZZZZZZZZ	
15	1,80E-06	-2.34972e+33	/	-3.63351e-25	=	Overflow				
15	1,80E-06	f6e7b346	/	96e0e744	=	Overflow		2.661us	7F800000	Overflow
.								2.661us	ZZZZZZZZZ	
16	1,92E-06	5.02013e+08	-	2.73995e-25	=	5.02013e+08				
16	1,92E-06	4def60e2	-	16a9983e	=	4def60e2		2.781us	4DEF60E2	
.								2.898us	ZZZZZZZZZ	
17	2,04E-06	-4.25830e-16	*	2.54101e-06	=	-1.08204e-21				
17	2,04E-06	a5f5796b	*	362a8638	=	9ca38354		2.901us	9CA38354	
.								3.018us	ZZZZZZZZZ	
18	2,16E-06	-7.00366e+36	+	1.24366e+36	=	-5.76000e+36				
18	2,16E-06	fca89b63	+	7b6f852c	=	fc8aaabe		3.021us	FC8AAABE	
.								3.144us	68080216	
19	2,28E-06	1.67037e+25	-	-2.25772e-18	=	1.67037e+25				
19	2,28E-06	695d1256	-	a2269720	=	695d1256		3.147us	695D1256	
.								3.258us	ZZZZZZZZZ	
20	2,40E-06	-1.80650e+20	*	-1.11537e-07	=	2.01492e+13				
20	2,40E-06	e11cb05c	*	b3ef861d	=	55929abc		3.261us	55929ABB	
.								3.378us	ZZZZZZZZZ	
21	2,52E-06	-2.81870e-22	-	-3.54237e-34	=	-2.81870e-22				
21	2,52E-06	9baa6145	-	87eb6e4d	=	9baa6145		3.381us	9BAA6145	
.								3.504us	13224145	
22	2,64E-06	1.80496e+31	-	3.21856e+17	=	1.80496e+31				
22	2,64E-06	7363d167	-	5c8eeeca	=	7363d167		3.507us	7363D167	

8. Comparisons

Compairing a single PFPU against a full processor isn´t presicelly fair, since the processor itself has many more instructions wich generally lowers it´s performance. Unfortunally it was impossible to find specific information concerning comercial processor ´s ALUs, hence this comparisons must be considered as rough aproximations made to get a general idea of the performances achieved. In order to get this aproximations as reallistic as possible, the following criteriums where taken into account:

1st Criterium
All comparisons are done following the same path as the processor manufacturer

E.g. 1: If a processor with an X clock frecuency internally duplicates that frecuency, then the comparison is done considering a PFPU running at a 2X clock frecuency.

E.g. 2: If a processor uses superscalar technology that implements 2 pipelines with 2 ALUs in them, then the comparison is done using 2 parallel PFPUs at the same clock frecuency.

2nd Criterium
3 plots are presented, showing:

a.-Processing speed of the comercial processor and processing speed of the PFPU following the manufacturer´s criterium

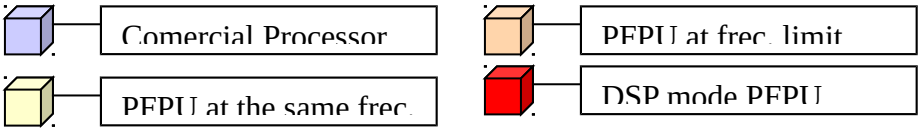
b.- Maximum processing speed of the PFPU alowwed by the technology used.

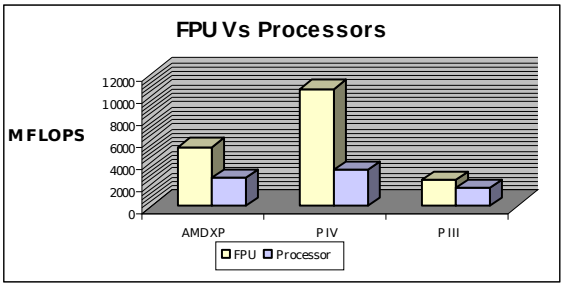
c.- Maximum processing speed of the DSP mode PFPU alowed by the technology used

3rd Criterium
All speeds are considered for burst operation while the circuit has been running for a long time (initial and final delays are very small compared to the operation time).

In order to make many comparisons available the following data was collected, though only 130 nm comparisons are shown:

Processor	Technology / Transistor Gate Delay	Internal Frec.[MHz]	FPU	Speed [MFLOP]	PFPU Speed [MFLOP]	PFPU Technology limit[MFLOP]	DSP mode [MFLOP]
Pentium 120	350 nm / 8 ps		1	150		1689	5058
Pentium Pro 200	350 nm / 8 ps		2	247		1689	10116
Pentium II 400	250 nm / 3.5ps			536		3861	11561
Pentium III 1200	130 nm / 1.5ps	2400	1	1620	2400	9009	26975
AMD Athlon XP 2200+	130 nm / 1.5ps	1800	3	2505	5400	27027	80925
Pentium IV-B 2660	130 nm / 1.5ps	5320	2	1385 / 3250	10640	18018	53950





a.The rate of calculation power at the same frecuency between the PFPU and commercial processors is:

Pentium III: **148 %**

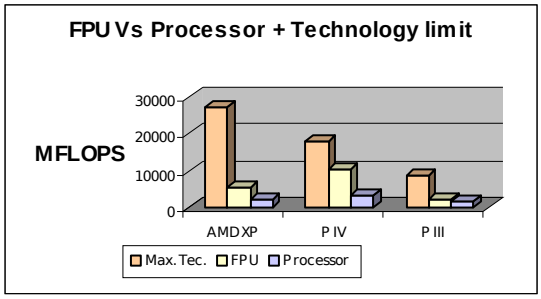
Pentium IV: **227 %**

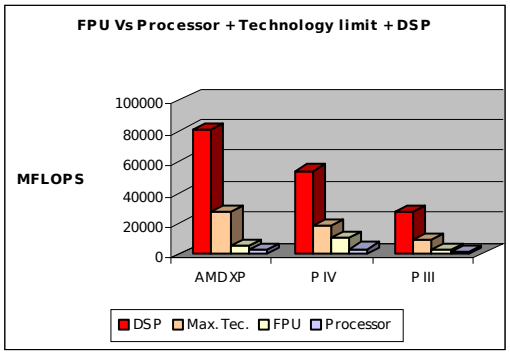
AMD XP: **115 %**

b:If the clock frecuency is taken to the maximum value allowed by the technology (obtained from the transistor delay time) then the calculation power compared to the ones found abobe is:
Pentium III: **556 %**

Pentium IV: **554 %**

AMD XP: **1789 %**
improvement is necesary to get this performance, only a litography process improvement.





If the divider is taken away from the circuit (DSP mode PFPU), the new calculation power rate is:

Pentium III: **1660 %**

Pentium IV: **1660 %**

AMD XP: **3230 %**

This means that 8 Pentium IV processors must be connected in parallel to match the calculation power of a single PFPU.

Note:
***** Calculation power for comertial processors was meassured using ´Sisoft Sandra´ software ([SW1]).
****** Calculation power for the PFPU was obtained by SIMULATING the design with ORCAD, MATLAB and MATHCAD softwares. This simulations asumed that operands where available 100% of the times, hence being best case considerations, while not taking into account real world facts, such as jump prediction, interrups,etc.

10. Conclusions

This work has presented both the complete design and results of simulations for a seven stage pipelined floating point unit (PFPU) that complies IEEE 754 standard for single presition operations. To reach this objective, basic operation algorithms were used to solve mathematical operations. Since this algorithms weren´t absolutely efficient to reach a high operation throughput, some choices were tried that modified the proposed architecture in order to enhance this efficiency by the use of a pipeline. At the beginning this way of focusing the problem seamed uncertain, but due to a structure comparative analisis it was possible to obtain an astonishing high floating point operations rate (compared to top-of-the-line comercial systems) with a low hardware cost, taking as indicator the number of logical gates involved.

The simulations runned cofirmed that the system behaves correctly under all circumstances and determined that the efficiency obtained was the expected one.

The pipeline enhacement for the proposed PFPU is capable of increasing the operation flow up to 600%. This fact allowed a system made of non-optimum blocks to reach operating processing speeds of at least 50% superior to top-of-the-line PC products (All Pentium IV processors).

This PFPU design can cover a wide range of applications, from integrated portable systems ,due to it´s advanced Power-Save feature (which can diminish the dissipated power from 48% to 82% while running operation bursts), to high calculation power aplications, where eaven dissipating a larger ammount of power the rate calculation power / energy consumption is much higher than comercial systems. This comparison though musn´t be considered strict, but taken only as an illustrative one, since a single PFPU has been compared to complete procesors due to the luck of information available regarding internal processor modules for comercial systems.

It has been proved that the divider block limits the PFPU performance, hence a special mode for the unit is introduced, named DSP (Digital Signal Processor) mode, which lacks this

*module (since DSPs are optimized to run operations in the form A*B+C*D). With this enhancement aplied, the results of the simulations indicate that the calculation power (measured in MFLOPS) of the PFPU could reach unmatched values, getting to be equivalent to that of 8 Pentium IV 2600 MHz processors parallel-working.*

The following table lists briefly the main characteristics of this design:

Calculation Power: up to 8 times (DSP mode) the one obtained from a Pentium IV 2600MHz with the same technology. **Power Consumtion PFPU:**From 41 mW (@ 1,5 GHz) up to 233 mW (@ 9 GHz) **DSP:** From 30 mW (@ 1,5 GHz) up to 531 mW (@ 30 GHz) (Reference: Pentium IV dissipates 54700 mW @ 2,4 GHz) **Power-Save system:** From 48% up to 82% energy saving for burst operation. **Pipeline :** 600% increase in operation throughput.

11. References

[P4_1] Iván Martínez, Pentium 4, **Una nueva tecnología**
http://www.itq.edu.mx/vidatec/espacio/aisc/ARTICULOS/PENTIUM4/Pentium4.html

[P4_2] Intel, **Pentium architecture FPU**
http://www.intel.com/design/intarch/techinfo/Pentium/PDF/fpu.pdf

Siguiente Página

Sumador/Restador:

[SM1]:Peter M.Seidel and Guy Even , ***An IEEE floating Point Adder Design Optimized For Speed***, September 1, 1999.

[SM2]: Yariv Levin, ***Supporting De-normalized Numbers in an IEEE Compliant Floating-Point Adder Optimized For Speed***, July 2001

[SM3]: J.R. Hoff and G.W. Foster, ***A Full Custom, High Speed Floating Point Adder, Fermi National Accelerator Laboratory***, Fermilab-Pub-92, 1992

[SM4]: Martin Horauer, Dietmar Loy, ***ADDER SYNTHESIS***, University of Technology Vienna Insitute of Computer Technology

[SM5]: Stuart F. Oberman and Michael J. Flynn, ***A variable latency pipelined floating point adder***, Technical report, Fbruary 1996.

[SM6]:Nhon T. Quach and Michael Flynn, ***An Improved Algorithm For High-Speed Floating Point Addition***, Technical report, August 1990.

[SM7]: Nhon T. Quach and Michael Flynn, ***Design and Implementation of the SNAP Floating Point Adder***, Technical report, December 1991.

[SM8]: Reto Zimmermann, ***Binary Adder Architecture for Cell-Based VLSI and their Syntesis***, A dissertation submitted to the SWISS FEDERAL INSTITUTE OF TECHNOLOGY, Zurich, 1997.

[SM9]: Montek Singh and Steven M. Nowick, ***Fine-Grain Pipelined Asynchronous Adders for High-Speed DSP Applications***, In Proceedings of the IEEE Computer Society Annual Workshop on VLSI, April 27–28, 2000, Orlando, Florida.

Multiplicador:

[ML1]: Gary W. Bewick, ***Fast Multiplication: Algorithms and implementatio***n, A dissertation submitted to the department of electrical engineering and the committee of graduate study of Stanford University, February 1994.

[ML2]: ***ECEN 6 2 6 3 Advanced VLSI Design***, Carry Save Adder Trees in Multipliers

[ML3]:Hesham and Flynn, ***Technology Scaling Effects on Multipliers***, Stanford University.

[ML4]: Guy Even, Peter M. Seidel, ***A comparisson of three rounding algorithm for IEEE Floating Point Multiplication***, August 29, 1998

Divisor:

[DV1]: Stuart F. Oberman, Student Member, IEEE, and Michael J. Flynn, Fellow, IEEE, ***Division Algorithms and Implementations***, IEEE TRANSACTIONS ON COMPUTERS, VOL. 46, NO. 8, AUGUST 1997.

[DV2]: Stuart F. Oberman, Member, IEEE, and Michael J. Flynn, Life Fellow, IEEE, **Minimizing the Complexity of SRT Tables**, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 6, NO. 1, MARCH 1998.

[DV3]:Stuart Oberman, Nhon Quach and Michael Flynn, ***The design and implementation of a high performance floating point divider***, Technical report, January 1994.

[DV4]: Stuart Oberman, and Michael Flynn, ***An Analysis of division algorithm and implementation***, Technical report, July 1995.

FPU:

[FP2]: Stuart Franklin Oberman, D***esign Issues In High Performance Floating Point Units***, Technical Report, December 1996.

[FP3]: Oberman, Hesham and flynn, ***The SNAP Proyect: Design of Floating Point Arithmetic Units***, Stanford University.

Norma IEEE:

[NRM1]: "***IEEE Standard for Binary Floating-Point Arithmetic***", ANSI/IEEE Standard 754, 1985.

[NRM2]: DAVID GOLDBERG, ***What Every Computer Scientist Should Know About Floating-Point Arithmetic***, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto.

[NRM3]: W. Kahan, ***IEEE Standar 754 for Binary Floating Point Arithmetic, Lecture notes on status of IEEE 754***, May 31, 1996.

Low Power:

[PW1]:Etienne Jacobs, ***Using Gate Sizing to Reduce Glitch Power.***

[PW2]: Luke Seed, ***Power Consumption in Circuits***, University of Sheffield E.E.E. Department Electronic Systems Group, Low Power Circuit Design Course 25 May 2001.

Multi-Thread:

[MT1]: Burton Smith, ***"A pipelined, shared resource MIMD computer",*** Proc. 1978 Int. Conf. Parallel Processing, Aug. 1978, pp. 6-8

General:

[VR1]: William Stalling, ***Organización y Arquitectura de computadoras: Principios y aplicaciones***, Ed. Megabyte, 1995.

Software:

[SW1]: Sisoft Sandra, ***Benchmarking Software***, www.sisoftware.demon.com.uk