AT6000 IP Generator Guide

June 2002

Macro Generator

Thank you for your interest in Atmel's AT6000 reconfigurable co-processor FPGAs with QuickChange^{TM}, and welcome to Atmel’s very powerful and unique design tool, Automatic Component Generators.

Designed to exploit Atmel's register rich, dynamically reconfigurable AT6000 series FPGAs, high speed custom logic functions can be created and implemented, resulting in significantly improved performance for compute intensive applications.

This software, when used with Atmel's AT6000 FPGA family, facilitates quick and easy implementation of over 50 compute intensive logic functions, such as multipliers, adders, and accumulators. Users can specify individual DSP parameters, including width, pipelining, and other function specific options. Within minutes the design tool will automatically create both a "hard layout" with worst case speed, area, and power consumption information, as well as generate a schematic symbol, VHDL or Verilog data for the function. The end result of this process is a fully specified, reusable circuit that can be used for any size AT6000 FPGA array, in any location or orientation, without affecting its speed or function. The use of these fully deterministic functions in synthesis results in significantly higher silicon efficiency, as well as faster speed and design time.

The following pages describe the various generators available in the IDS software. For each one there is an explanation of its function, the various parameters available, a list of its input and output pins, the statistics for 16 and 8 bit versions of the macro, and a screen capture of the graphical user interface.

The statistics for the macro contain a number of figures ranging from speed to power dissipation. The second and third columns in the table are about the speed of the component. It should be noted that the speed figures for macros which contain sequential components (registers) exclude the delay time for getting the global clock onto the chip and to the register element. This is roughly
3.3 ns plus a fanout factor.

The next two columns deal with the size of the macro. The Cells column is the number of core cells used to build the component. The Size is the width and height of the macro in core cells. The Gates column is the number of TTL equivalent gates that the circuit represents. Power consumption is provided for the macro. The figure given is assuming that 80% of the devices are switching at one time. The final figure, Gates/Cell, is basically the result of dividing the gate count by the number of cells. It provides a relative figure of density for the macro.

AT6000 Cache Logic™ FPGAs

The Atmel AT6000 is a family of SRAM-based FPGAs with thousands of registers, [1]Cache Logic™ ability (partially or fully reconfigurable without loss of data), automatic component generators, and ranges in size from 2,000 to 30,000 usable gates.

The AT6000 is designed to act as a co-processor for digital signal processors (DSPs) by off-loading a variety of functions that might otherwise slow down the DSP. These include adaptive finite impulse response (FIR) filters, fast Fourier transforms (FFT), interpolators and discrete-cosine transforms (DCT) that are required for video compression and decompression, encryption, convolution and other multimedia applications.

Atmel offers the only FPGAs with *Cache Logic* capability, providing the means to build adaptive hardware. Other unique features of the AT6000 Series devices include:

- Thousands of registers for pipelining
- 350 MHz register toggle rates for high speed data flow
- Very low power consumption (<200uA standby current)
- 1mm high PCMCIA (type 1) packages
- System speeds in excess of 90 MHz
- User-defined, reusable hard macro capability
- Hard macro generators
- QuickChange software for adaptive hardware design
- Infinitely reprogrammable in-system within <1ms
- Fully or partially reprogrammable on-the-fly without loss of data (200 ns/cell)
- Powerful simple, symmetrical, easy to use architecture

Atmel has introduced the first four members of its AT6000 FPGA family. These circuits range from 2,000 to 30,000 usable gates, and allow designers to easily achieve new levels of device integration at exceptionally high speeds. The AT6000 family features thousands of registers and XORs, as well as a new means of implementing reconfigurable hardware called *Cache Logic*.

With these features, the DSP and other logic functions can be created quickly and accurately using the software, programmed into the FPGA, and then changed as new functions or algorithms are required. Individual logic cells can be reprogrammed at a rate of 200 ns per cell, without loss of register data. In fact, part of the array can be reprogrammed while the remainder continues to operate without interruption. These features will offer significant performance gains for graphics and imaging, network, military instrumentation, and telecommunications applications.

Cache Logic makes adaptive hardware possible, and accelerates system performance by handling logic much the same way a computer's cache memory handles instructions and data. In a computer, the highest speed memory (usually SRAM) stores active data, while the bulk of the data resides in a less expensive DRAM, EPROM, or disk. In most complex applications, only a small fraction of the circuitry is active at any given time. Only active functions are loaded into the FPGA's "logic cache", while unused circuits reside in less expensive external memory. Logic can be added to the "cache" as needed to replace or complement existing functions without physical modification to the hardware.

With *Cache Logic*, a 25,000-gate application may actually only require 5,000 gates at any given cycle. By caching the extra 20,000 gates for later use, a 5,000-gate device can replace a more expensive 25,000-gate device.

If you are new to FPGAs, you'll appreciate the simple, easy to learn architecture and tools. If you are already using FPGAs, you'll find the AT6000 FPGAs especially suited for high speed, data path applications. The thousands of registers allow for designs to be extensively pipelined, enabling ultra-fast system performance.

Atmel's design software, the Integrated Development System, uses a single data base to take your work from design entry to configured circuit quickly and efficiently. With exclusive features like system level hard macro functions, and advanced timing analysis that uses physical wire lengths and loads to predict delays, our software tools let you design circuits that perform exactly to specification.

When your finalized design enters high volume production, the tools allow easy migration to Atmel's masked Gate Arrays for even more cost effective production.

Statistical Summary

The following table is a summary, for quick reference, of all of the generators covered in this manual.

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cel l	Latency (Clocks)
abs16	14.39	69.5	46	02x30	108	0.07	2.34	0
abs8	29.76	33.6	22	02x14	48	0.04	2.16	0
acc16	18.38	54.4	128	08x16	337	0.45	2.63	0
acc8	28.33	35.3	64	08x08	169	0.31	2.64	0
acp16	49.75	20.1	248	14x39	1444	1.21	5.82	7
acp8	49.75	20.1	82	08x19	413	0.66	5.03	3
acs16	26.11	38.3	179	37x06	348	0.33	1.94	0
acs8	38.61	25.9	87	19x06	167	0.16	1.91	0
arc16	21.55	46.4	95	06x16	184	0.18	1.94	0
arc8	36.63	27.3	47	06x08	92	0.09	1.96	0
arp16	73.53	13.6	454	31x47	2957	2.35	6.51	15
arp8	73.53	13.6	130	15x23	755	1.09	5.8	7
adc16	136.99	7.3	56	04x07	29	0.04	1.16	0
adc8	196.08	5.1	20	03x06	17	0.02	1.13	0
alu16	7.15	139.8	404	08x52	497	0.76	1.23	0
alu8	12.58	79.5	212	08x28	257	0.4	1.21	0
com16	11.71	85.4	115	04x35	166	0.17	1.44	0
com8	20.20	49.5	59	04x19	82	0.09	1.38	0
ceq16	51.55	19.4	38	03x16	71	0.09	1.87	0
ceq8	46.08	21.7	24	03x08	35	0.04	1.46	0
cpl16	46.73	21.4	385	18x50	2205	1.56	5.73	15
cpl8	46.08	21.7	129	10x26	613	0.82	4.75	7
cjo16	104.17	9.6	16	01x16	121	0.2	7.53	0
cjo8	136.99	7.3	8	01x08	61	0.19	7.56	0
clf16	72.46	13.8	21	01x21	133	0.21	6.31	0
clf8	84.75	11.8	13	01x13	73	0.2	5.58	0

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell	Latency (Clocks)
cps16	20.83	48	99	07x18	362	0.31	3.65	0
cps8	34.25	29.2	51	07x10	182	0.25	3.56	0
crc16	26.18	38.2	32	02x16	181	0.22	5.64	0
crc8	51.02	19.6	16	02x08	89	0.2	5.53	0
ctf16	29.94	33.4	80	4x16	328	4.1	0.18	0
ctf8	43.10	23.2	40	4x08	164	4.1	0.09	0
crp16	107.53	9.3	179	16x30	1178	1.25	6.58	14
crp8	107.53	9.3	55	08x14	336	0.66	6.11	6
ctr16	11.96	83.6	116	07x17	464	0.43	4	0
ctr8	20.37	49.1	60	07x09	232	0.34	3.87	0
crc	20.70	48.3	120	09x18	325	0.5	2.71	0
dec16	120.48	8.3	64	04x16	64	0.12	1	0
dec8	163.93	6.1	24	03x08	22	0.04	0.92	0
ded16	17.67	56.6	144	09x16	345	0.55	2.4	0
ded8	26.60	37.6	72	09x08	173	0.37	2.4	0
dep16	50.51	19.8	251	14x40	1452	1.23	5.78	7
dep8	50.51	19.8	85	08x20	417	0.67	4.91	3
div16	149.25	6.7	8	01x08	61	0.19	7.56	0
div8	149.25	6.7	4	01x04	31	0.19	7.63	0
fif16	21.41	46.7	415	19x23	1778	1.66	4.28	0
fif8	22.32	44.8	231	11x23	946	0.97	4.1	0
fir	57.14	17.5	751	27x32	3139	1.99	4.18	0
ffd16	454.55	2.2	16	01x16	120	0.19	7.5	0
ffd8	454.55	2.2	8	01x08	60	0.19	7.5	0
fdt16	344.83	2.9	16	01x16	128	0.23	8	0
fdt8	344.83	2.9	8	01x08	64	0.21	8	0
fft16	178.57	5.6	32	02x16	184	0.22	5.75	0
fft8	178.57	5.6	16	02x08	92	0.2	5.75	0
gra16	217.39	4.6	32	02x16	46	0.05	1.47	0
gra8	217.39	4.6	16	02x08	22	0.02	1.43	0

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell	Latency (Clocks)
inc16	19.38	51.6	112	07x16	314	0.41	2.8	0
inc8	30.77	32.5	56	07x08	154	0.29	2.75	0
inv16	19.38	51.6	112	07x16	314	0.41	2.8	0
inv8	30.77	32.5	56	07x08	154	0.29	2.75	0
ldt16	108.70	9.2	80	03x32	72	0.13	0.9	0
ldt8	108.70	9.2	40	03x16	36	0.06	0.9	0
lif16	74.07	13.5	425	34x13	2452	2.61	5.77	0
lif8	74.07	13.5	225	18x13	1296	1.44	5.76	0
log16	66.23	15.1	24	04x08	15	0.03	0.63	0
log8	108.70	9.2	10	03x04	7	0.01	0.7	0
mul16	4.85	206	1470	48x31	2344	2.11	1.59	0
mul8	10.33	96.8	350	24x15	532	0.49	1.52	0
mlp16	30.86	32.4	3477	79x55	13046	9.33	3.75	24
mlp8	43.86	22.8	825	38x27	3020	3.44	3.66	12
msp16	79.37	12.6	113	32x04	457	1.26	4.04	0
msp8	79.37	12.6	57	16x04	229	0.69	4.01	0
mux16	116.28	8.6	40	05x08	74	0.05	1.85	0
mux8	97.09	10.3	11	03x04	39	0.01	3.5	0
nef16	15.34	65.2	44	02x30	74	0.07	1.67	0
nef8	34.25	29.2	20	02x14	34	0.03	1.67	0
psr16	151.52	6.6	16	01x16	208	0.19	13	0
psr8	151.52	6.6	8	01x08	104	0.19	13	0
pen16	44.44	22.5	183	11x17	123	0.28	0.67	0
pen8	51.81	19.3	77	09x09	54	0.11	0.7	0
pls16	26.18	38.2	52	02x27	259	0.36	4.97	0
pls8	34.84	28.7	36	02x19	143	0.32	3.96	0
rdp16	51.28	19.5	248	16x19	976	1.59	3.94	0
rdp8	64.94	15.4	100	08x16	366	0.79	3.66	0
rsp16	21.79	45.9	280	16x21	1032	1.65	3.69	0
rsp8	25.19	39.7	116	08x18	394	0.84	3.4	0
rom16	51.02	19.6	184	16x12	144	0.3	0.78	0
rom8	57.80	17.3	76	08x10	54	0.13	0.71	0

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell	Latency (Clocks)
sre16	227.27	4.4	16	01x16	120	0.19	7.5	0
sre8	227.27	4.4	8	01x08	60	0.19	7.5	0
sba16	20.41	49	283	26x17	359	0.64	1.27	0
sba8	34.84	28.7	83	12x09	135	0.17	1.63	0
sub16	20.08	49.8	128	08x16	193	0.22	1.5	0
sub8	32.57	30.7	64	08x08	97	0.11	1.51	0
sup16	72.99	13.7	456	31x48	2965	2.36	6.5	15
sup8	72.99	13.7	132	15x24	759	1.1	5.75	7
tsb16	344.83	2.9	16	01x16	16	0.01	1	0
tsb8	344.83	2.9	8	01x08	8	0.01	1	0

Absolute Value

The function of the Absolute Value generator can be explained as follows:

```
if DATA = -2 (Width -1), then
        OVERFLOW = 1, RESULT = 0
else if DATA < 0, then
        RESULT = -DATA
else
        RESULT = DATA
```

DATA must always represent a two’s complement number and the RESULT is always a positive number.

Parameters

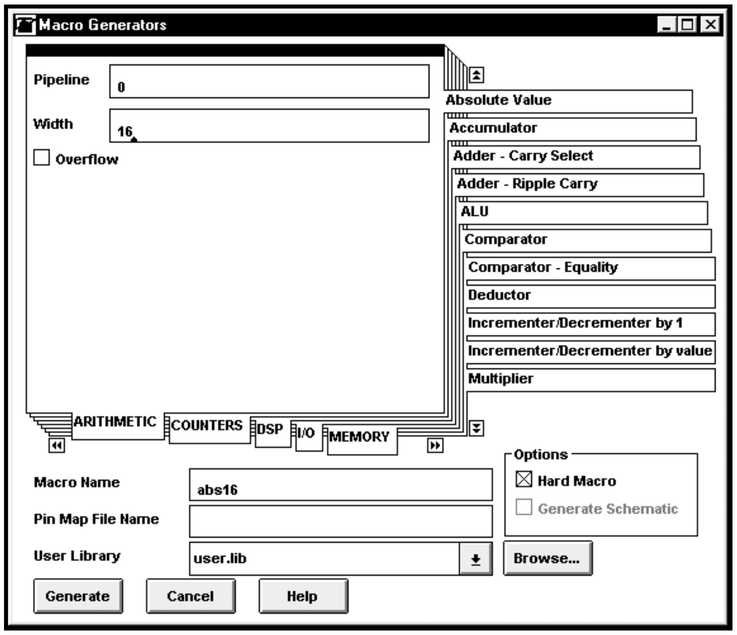
Parameter	Value	Explanation
Overflow	Boolean	Create with Overflow pin
Pipeline	Integer > 1	Create a pipeline every *n* stages
Width	Integer > 1	Width of input and output vectors

Pins

Type	Name	Option	Explanation
In	DATA[Width-1:0]	No	Input data
Out	RESULT[Width-1:0]	No	Absolute value of Data (will be Width-2 if overflow is used)
Out	OVERFLOW	Yes	True if Data = -2 $^{(Width-1)}$

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
abs16	14.39	69.5	46	2x30	108	0.07	2.34
abs8	29.76	33.6	22	2x14	48	0.04	2.16



Pipeline 0

Width 16

☐ Overflow

Macro Name abs16

Pin Map File Name

User Library user.lib

Options

☒ Hard Macro

☐ Generate Schematic

Accumulator

The Accumulator adds a given number to the register initial value. The functional description of the accumulator is as follows:

```
always(@posedge CLK or negedge R)
begin
        if(R == ‘b0)
                SUM = 0;
        else if (ACC)
                {COUT, SUM} = SUM + DATA + CIN;
end
```

Parameters

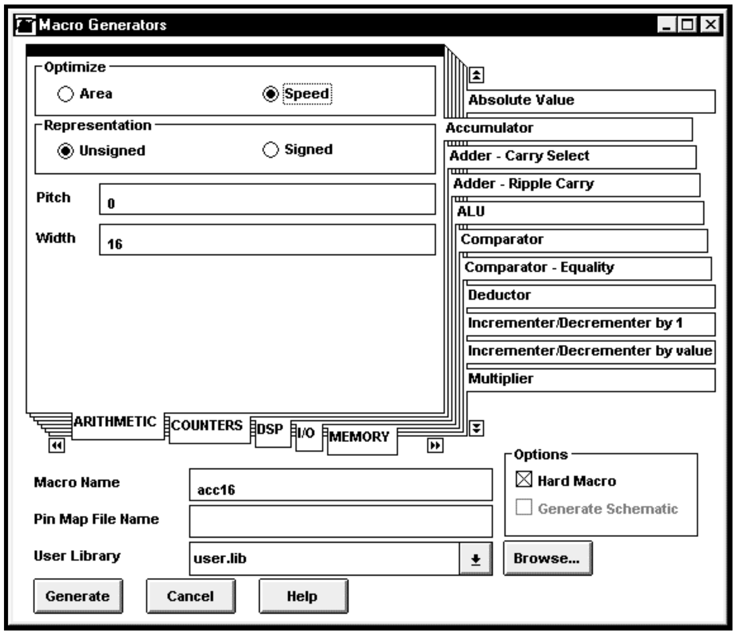
Parameter	Value	Explanation
Optimize	Area	Create an area efficient layout
	Speed	Create a fast layout
Representatio n	Unsigned	Treat inputs as unsigned
	Signed	Treat inputs as signed
Pitch	Integer > 1	Spacing between input pins, pitch of 2 means one cell between input pins (not supported for pipelining)
Pipeline	Integer > 0	Create a pipeline every *n* stages
Width	Integer > 1	Width of input and output vectors

Pins

Type	Name	Option	Explanation
In	CIN	No	Carry in
In	DATA[Width-1:0]	No	Parallel input data
In	ACC	No	Accumulate control (cannot be used with pipelining)
In	CLK	No	Clock
In	R	No	Reset (active low)
Out	SUM[Width-1:0]	No	Accumulator output
Out	COUT	No	Carry out

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cel l
acc16	18.38	54.4	128	8x16	337	0.45	2.63
acc8	28.33	35.3	64	8x8	169	0.31	2.64



Optimize: Area [✓ Speed]

Representation: [✓ Unsigned] Signed

Pitch: 0

Width: 16

Macro Name: acc16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic

Accumulator – Pipelined

This generator adds a given amount from the register initial value with the pipelining option. The functional description of the accumulator is as follows:

```
always(@posedge CLK or negedge R)
begin
        if(R == ‘b0)
                SUM = 0;
        else if (ACC)
                {COUT, SUM} = SUM + DATA + CIN;
end
```

Parameters

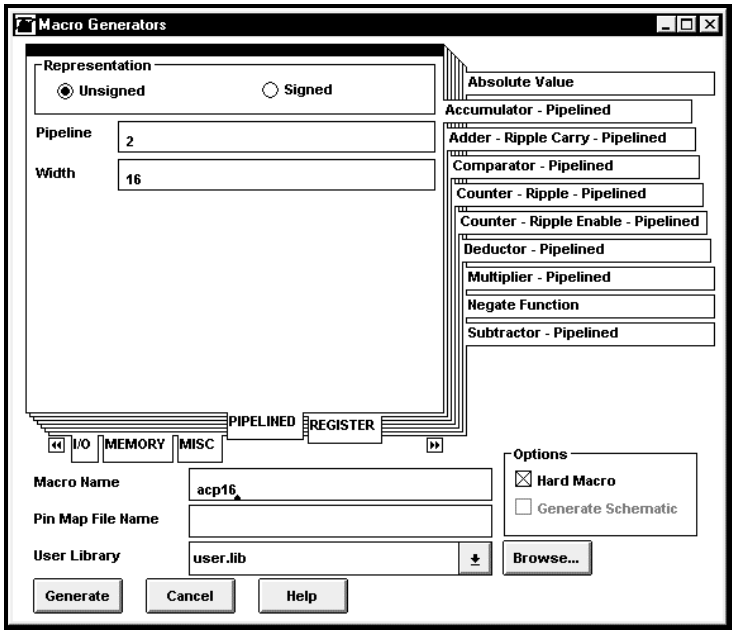
Parameter	Value	Explanation
Optimize	Area	Create an area efficient layout
	Speed	Create a fast layout
Representatio n	Unsigned	Treat inputs as unsigned
	Signed	Treat inputs as signed
Pitch	Integer > 1	Spacing between input pins, pitch of 2 means one cell between input pins (not supported for pipelining)
Pipeline	Integer > 0	Create a pipeline every *n* stages
Width	Integer > 1	Width of input and output vectors

Pins

Type	Name	Option	Explanation
In	CIN	No	Carry in
In	DATA[Width-1:0]	No	A input
In	ACC	No	Accumulate control (cannot be used with pipelining)
In	CLK	No	Clock
In	R	No	Reset (active low)
Out	SUM[Width-1:0]	No	Adder output
Out	COUT	No	Carry out

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cel l
acp16	49.75	20.1	248	14x39	1444	1.21	5.82
acp8	49.75	20.1	82	8x19	413	0.66	5.03



Representation: [✓ Unsigned] Signed

Pipeline: 2

Width: 16

Macro Name: acp16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic

Adder – Carry Select

This generator can be used to generate an *n* bit Carry Select Adder.

Parameters

Parameter	Value	Explanation
Width	Integer > 1	Width of input and output vectors

Pins

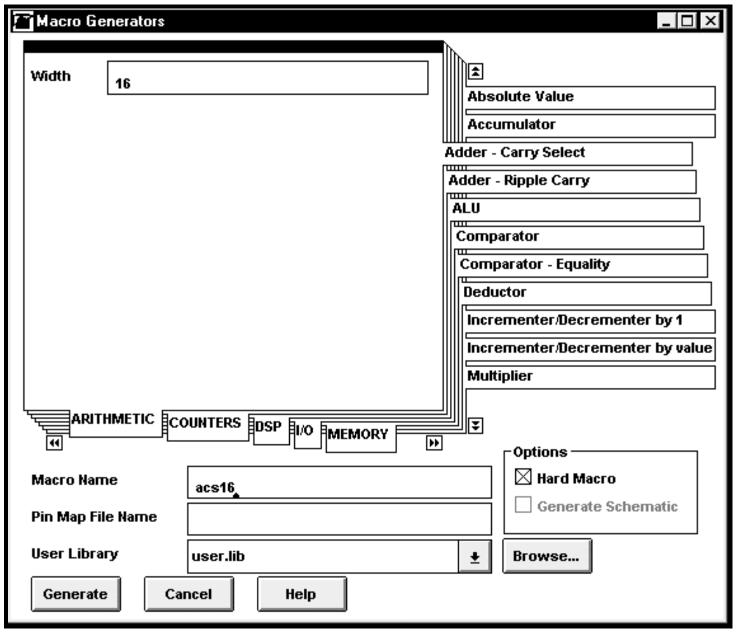
Type	Name	Option	Explanation
In	CIN	No	Carry in
In	DATAA[Width-1:0]	No	A input
In	DATAB[Width-1:0]	No	B input
Out	SUM[Width-1:0]	No	Adder output
Out	COUT	No	Carry out

Truth Table

Input			Output	
CIN	DATAA[W-1:0]	DATAB[W-1:0]	SUM[W-1:0]	COUT
C	A	B	A + B + C	1 if A+B+C > = 2^{Width}, 0 otherwise

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
acs16	26.11	38.3	179	37x6	348	0.33	1.94
acs8	38.61	25.9	87	19x6	167	0.16	1.91



Width 16

Macro Name acs16

Pin Map File Name

User Library user.lib

Options
[✓ Hard Macro]
Generate Schematic

Adder – Ripple Carry

The generator can be used to create a ripple carry adder. It has options for both an area efficient and fast layout. Regis tered inputs and/or outputs can be specified as well.

Parameters

Parameter	Value	Explanation
Representati on	Unsigned	Treat inputs as unsigned
	Signed	Treat inputs as signed
Optimize	Area	Create an area efficient layout
	Speed	Create a fast layout
Pipeline	Integer > 0	Pipeline every *n* stages
Pitch	Integer > 1	Spacing between input pins, pitch of 2 means one cell between input pins
Width	Integer > 1	Width of input and output vectors
Carryin	Noregister	Include the carry in pin on the generator but do not register it
	Register	Register carry in of the adder
	Disabled	Do not include the carry in pin on the adder
Register	None	Do not register the inputs and outputs
	Input	Register inputs on the adder, exclude carry in pin
	Output	Register outputs on the adder, exclude carry out pin
	Both	Register both inputs and outputs including the carry in and carry out pins of the adder
Carryout	Noregister	Include the carry out pin on the adder but do not register it
	Register	Register carry out of the adder
	Disabled	Do not include the carry out pin on the adder
Overflow	Boolean	Create a pin to signal overflow
Fold	Boolean	For area efficient option, fold layout in half (cannot be used with OutRegister or pipelining)

Pins

Type	Name	Option	Explanation
In	CIN	Yes	Carry in
In	DATAA[Width-1:0]	No	A input
In	DATAB[Width-1:0]	No	B input
In	CLK	No	Clock (exists only for pipelined adder)
In	R	No	Reset (active low and exists only for pipelined adder)
Out	SUM[Width-1:0]	No	Adder output
Out	COUT	Yes	Carry out (cannot be used with overflow)
Out	OVERFLOW	Yes	Overflow (cannot be used with COUT)

Carry out = $\text{DATAA} + \text{DATAB} + \text{CIN} > 2^n - 1$ or
$\text{DATAA} + \text{DATAB} + \text{CIN} < -2^n$

Overflow for Unsigned is equal to 1 only if
$\text{DATAA} + \text{DATAB} + \text{CIN} > 2^n - 1$

Overflow for Signed is equal to 1 only if
$\text{DATAA} + \text{DATAB} + \text{CIN} > 2^n - 1$ or
$\text{DATAA} + \text{DATAB} + \text{CIN} < -2^n$

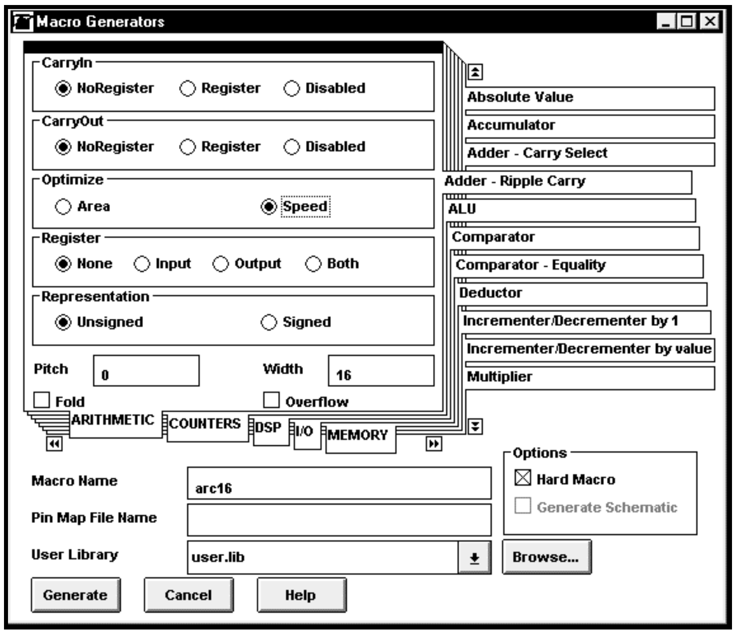
where n = Width

Truth Table

Input			Output	
CIN	DATAA[W-1:0]	DATAB[W-1:0]	SUM[W-1:0]	COUT
C	A	B	A + B + C	1 if $A+B+C >= 2^{Width}$, otherwise

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
arc16	21.55	46.4	95	6x16	184	0.18	1.94
arc8	36.63	27.3	47	6x8	92	0.09	1.96


Macro Generators

CarryIn: [✓ NoRegister] Register Disabled

CarryOut: [✓ NoRegister] Register Disabled

Optimize: Area [✓ Speed]

Register: [✓ None] Input Output Both

Representation: [✓ Unsigned] Signed

Pitch: 0

Width: 16

☐ Fold

☐ Overflow

ARITHMETIC | COUNTERS | DSP | I/O | MEMORY

Absolute Value
Accumulator
Adder - Carry Select
Adder - Ripple Carry
ALU
Comparator
Comparator - Equality
Deductor
Incrementer/Decrementer by 1
Incrementer/Decrementer by value
Multiplier

Macro Name: arc16

Pin Map File Name:

User Library: user.lib

Options:
☒ Hard Macro
☐ Generate Schematic

Browse...

Generate | Cancel | Help

Adder – Ripple Carry Pipelined

The generator can be used to make a ripple carry pipelined adder. Registered inputs and/or outputs can be specified as well.

Parameters

Parameter	Value	Explanation
Representation	Unsigned	Treat inputs as unsigned
	Signed	Treat inputs as signed
Optimize	Area	Create an area efficient layout
	Speed	Create a fast layout
Pipeline	Integer > 0	Pipeline every *n* stages
Pitch	Integer > 1	Spacing between input pins, pitch of 2 means one cell between input pins
Width	Integer > 1	Width of input and output vectors
Carryin	Noregister	Include the carry in pin on the generator but do not register it.
	Register	Register carry in of the adder
	Disabled	Do not include the carry in pin on the adder
Register	None	Do not register the inputs and outputs
	Input	Register inputs on the adder, exclude carry in pin
	Output	Register outputs on the adder, exclude carry out pin
	Both	Register both inputs and outputs including the carry in and carry out pins of the adder
Carryout	Noregister	Include the carry out pin on the adder but do not register it
	Register	Register carry out of the adder
	Disabled	Do not include the carry out pin on the adder
Overflow	Boolean	Create a pin to signal overflow
Fold	Boolean	For area efficient option, fold layout in half (cannot be used with OutRegister or pipelining)

Pins

Type	Name	Option	Explanation
In	CIN	Yes	Carry in
In	DATAA[Width-1:0]	No	A input
In	DATAB[Width-1:0]	No	B input
In	CLK	No	Clock (exists only for pipelined adder)
In	R	No	Reset (active low and exists only for pipelined adder)
Out	SUM[Width-1:0]	No	Adder output
Out	COUT	Yes	Carry out (cannot be used with overflow)
Out	OVERFLOW	Yes	Overflow (cannot be used with COUT)

Carry out = $\text{DATAA} + \text{DATAB} + \text{CIN} > 2^n - 1$ or
$\text{DATAA} + \text{DATAB} + \text{CIN} < -2^n$

Overflow for Unsigned is equal to 1 only if
$\text{DATAA} + \text{DATAB} + \text{CIN} > 2^n - 1$

Overflow for Signed is equal to 1 only if
$\text{DATAA} + \text{DATAB} + \text{CIN} > 2^n - 1$ or
$\text{DATAA} + \text{DATAB} + \text{CIN} < -2^n$

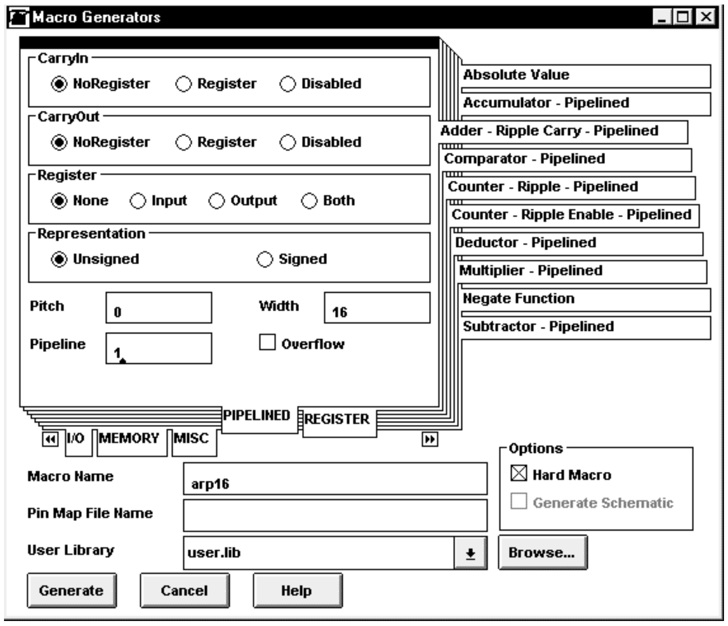
where n = Width

Truth Table

Input			Output	
CIN	DATAA[W-1:0]	DATAB[W-1:0]	SUM[W-1:0]	COUT
C	A	B	A + B + C	1 if A+B+C > 2^{Width}, 0 otherwise

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
arp16	73.53	13.6	454	31x47	2957	2.35	6.51
arp8	73.53	13.6	130	15x23	755	1.09	5.80



CarryIn: [✓ NoRegister] Register Disabled

CarryOut: [✓ NoRegister] Register Disabled

Register: [✓ None] Input Output Both

Representation: [✓ Unsigned] Signed

Pitch: 0

Width: 16

Pipeline: 1

☐ Overflow

Macro Name: arp16

Pin Map File Name:

User Library: user.lib

Options: ☒ Hard Macro ☐ Generate Schematic

Address Decoder

The Address Decoder generator is used to create a selective address decode circuit. The addresses to be decoded are specified in a file and the decoding logic for the specified addresses is generated. The addresses specified in the file must be in binary format as described below.

Parameters

Parameter	Value	Explanation
AD width	Integer > 0	Width of Address Inputs
File	File name	Name of the file containing addresses to be decoded

Pins

Type	Name	Option	Explanation
In	ADDRESS[AD-Width-1:0]	No	Input address to be decoded
Out	Q[Width-1:0]	No	Data output

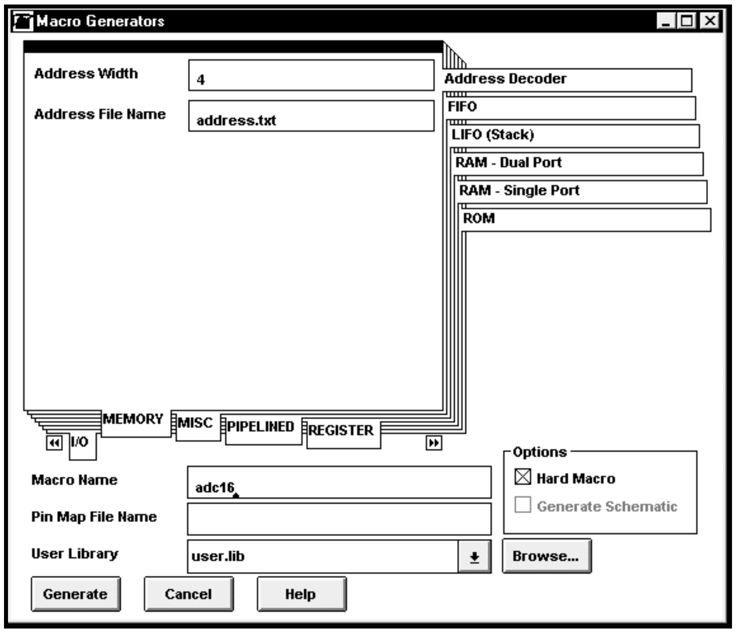
This generator reads as input a file specified by the File option (default is rom.hex) which should be located in the project directory. The format of the file is basically a header line with bin to indicate that numbers in the file are in Binary format. Each subsequent line should be of the form Address. For example

rom.hex

```
bin
0000
0100
1010
1111
1001
1100
0001
.
.
.
```

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
adc16	136.99	7.3	56	4x16	64	0.09	1.14
adc8	196.08	5.1	20	3x8	22	0.03	1.10



Address Width: 4
Address File Name: address.txt
Macro Name: adc16
Pin Map File Name:
User Library: user.lib
[✓ Hard Macro]

ALU

The ALU generator is used to create a macro which is functionally equivalent to the Arithmetic Logic Unit used in an AMD 2901 processor. The ALU generator performs three arithmetic operations and five logical operations as outlined below.

A three bit bus (S) controls the operation performed in the ALU generator. The overflow output pin (OVERFLOW) is available as an option to the generator. The overflow output is used to flag arithmetic operations that exceed the available two's complement number range. The overflow output is HIGH when overflow exists. The table below shows the ALU functions for the different select inputs (S2,S1,S0).

S2 S1 S0	ALU Function
0 0 0	DATAA + DATAB
0 0 1	DATAB - DATAA
0 1 0	DATAA - DATAB
0 1 1	DATAA OR DATAB
1 0 0	DATAA AND DATAB
1 0 1	(not DATAA) AND DATAB
1 1 0	DATAA XOR DATAB
1 1 1	DATAA XNOR DATAB

Parameters

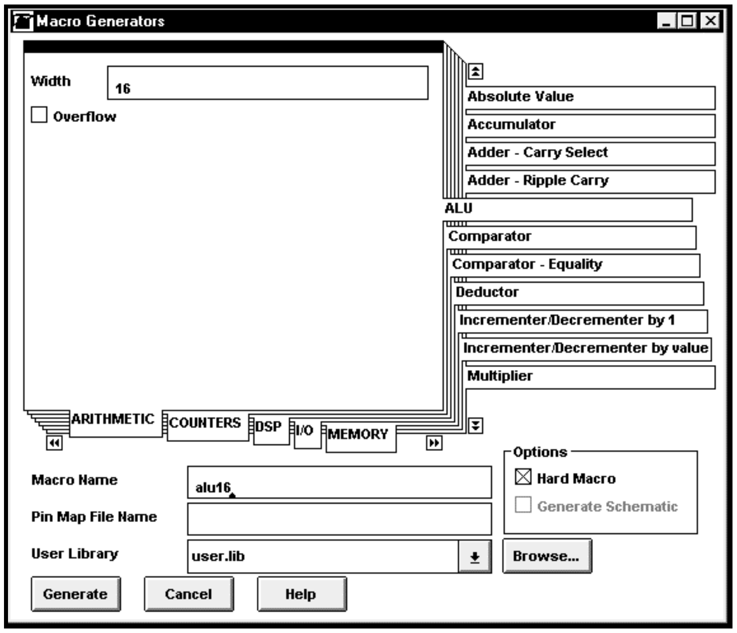
Parameter	Value	Explanation
Width	Integer > 0	Width of input data to the ALU
Overflow	Boolean	Create with Overflow pin

Pins

Type	Name	Option	Explanation
In	DATA[Width-1:0]	No	Input to ALU
In	CIN	No	Carry in
In	S[2:0]	No	Select for ALU function
Out	RESULT[Width-1:0]	No	Output of ALU
Out	OVERFLOW	Yes	Overflow
Out	COUT	No	Carry Out

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
alu16	7.15	139.8	404	8x52	497	0.76	1.23
alu8	12.58	79.5	212	8x28	257	0.40	1.21



Width 16

☐ Overflow

Macro Name alu16

Pin Map File Name

User Library user.lib

Options

☒ Hard Macro

☐ Generate Schematic

Comparator

The function of the Comparator generator can be explained as follows:

AEB = 1 if (DATAA == DATAB), 0 otherwise
ALB = 1 if (DATAA < DATAB), 0 otherwise
AGB = 1 if (DATAA > DATAB), 0 otherwise

Parameters

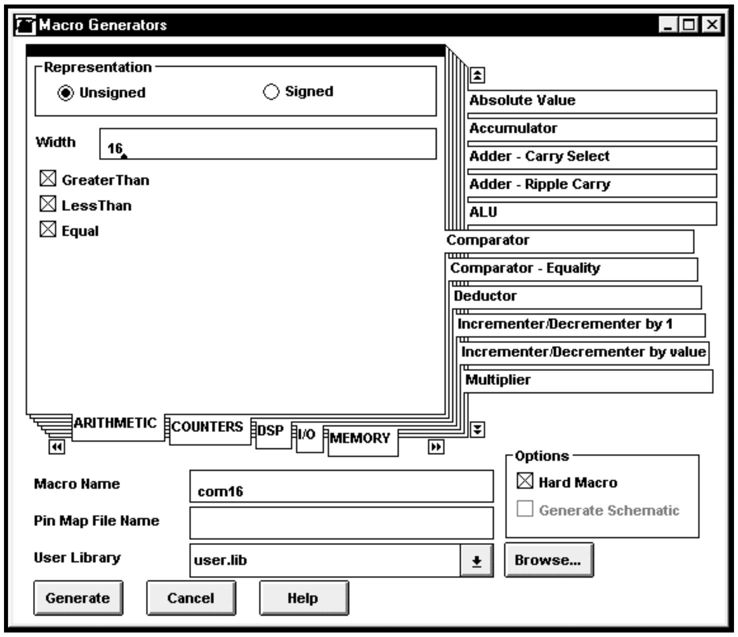
Parameter	Value	Explanation
EQ	None	Add equal pin (AEB) to full comparator
EqualOnly	None	Create Equality Only function
GT	None	Add greater than pin (AGB) to full comparator
LT	None	Add less than pin (ALB) to full comparator
Pipeline	Integer >=1	Add pipeline register every *n* cells
Representati on	Unsigned	Treat inputs as unsigned
	Signed	Treat inputs as signed
Width	Integer > 1	Width of inputs A and B

Pins

Type	Name	Option	Explanation
In	DATAA[Width-1:0]	No	Input data A
In	DATAB[[Width-1:0]	No	Input data B
Out	AEB	Yes	1 if A = B, 0 otherwise
Out	ALB	Yes	1 if A < B, 0 otherwise
Out	AGB	Yes	1 if A > B, 0 otherwise

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cel l
com16	11.71	85.4	115	4x35	166	0.17	1.44
com8	20.20	49.5	59	4x19	82	0.09	1.38



Macro Generators

Representation

[● Unsigned] ○ Signed

Width 16

☒ GreaterThan

☒ LessThan

☒ Equal

Absolute Value

Accumulator

Adder - Carry Select

Adder - Ripple Carry

ALU

Comparator

Comparator - Equality

Deductor

Incrementer/Decrementer by 1

Incrementer/Decrementer by value

Multiplier

ARITHMETIC | COUNTERS | DSP | I/O | MEMORY

Macro Name com16

Pin Map File Name

User Library user.lib

Options

☒ Hard Macro

☐ Generate Schematic

Comparator – Equality

The function of the Equality Comparator generator can be explained as follows:

AEB = 1 if (DataA == DataB), 0 otherwise

This function should be used when only Equality is needed as it is much smaller than the standard Comparator.

Parameters

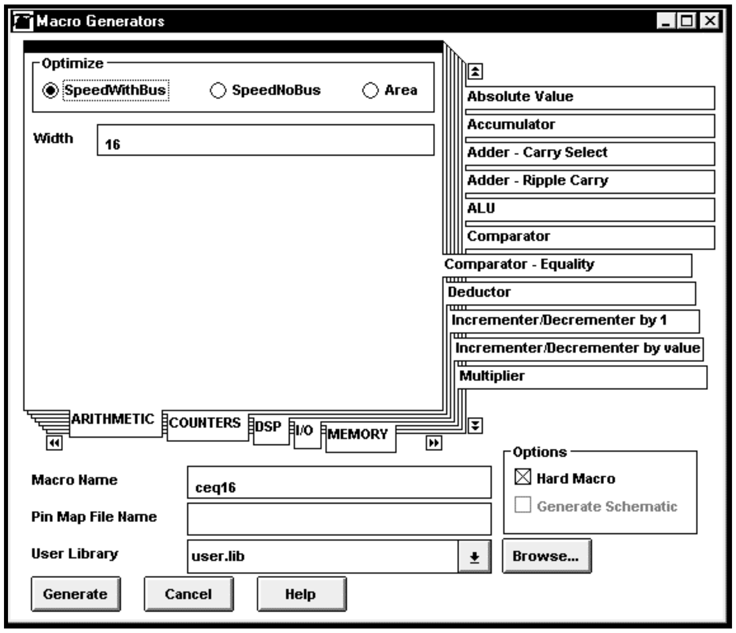
Parameter	Value	Explanation
EqualOnly	None	Create Equality Only function
Optimize	Area	Create an area efficient layout
	SpeedWithBus	Create a fast layout with local buses (fastest macro)
	SpeedNoBus	Create a fast layout without local buses
Width	Integer > 0	Width of inputs A and B

Pins

Type	Name	Option	Explanation
In	DATAA[Width-1:0]	No	Input data A
In	DATAB[Width-1:0]	No	Input data B
Out	AEB	No	1 if A = B, 0 otherwise

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
ceq16	51.55	19.4	38	3x16	71	0.09	1.87
ceq8	46.08	21.7	24	3x8	35	0.04	1.46



Optimize: [✓ SpeedWithBus] SpeedNoBus Area

Width: 16

Macro Name: ceq16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic

Comparator – Pipelined

The function of the Comparator generator can be pipelined and explained as follows:

AEB = 1, if (DataA == DataB), 0 otherwise
ALB = 1, if (DataA < DataB), 0 otherwise
AGB = 1, if (DataA > DataB), 0 otherwise

Parameters

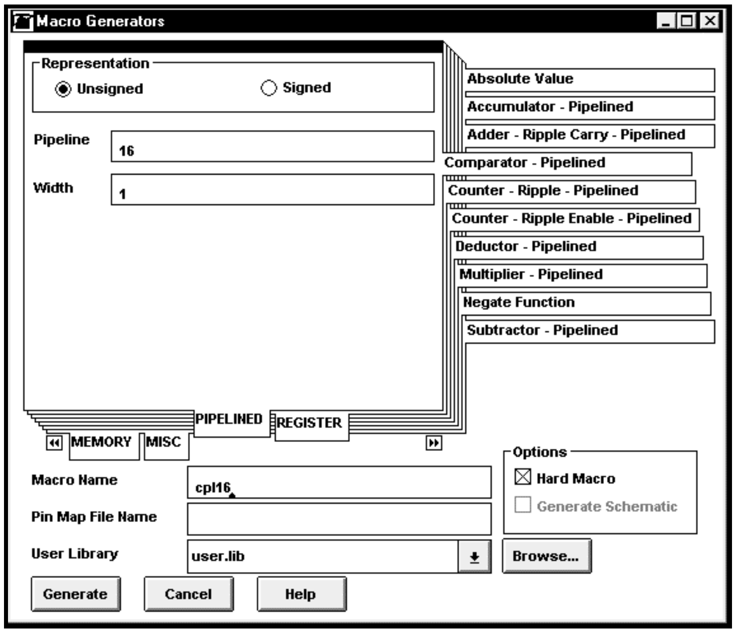
Parameter	Value	Explanation
EqualOnly	None	Create Equality Only function
Pipeline	Integer >=1	Add pipeline register every *n* cells
Representation	Unsigned	Treat inputs as unsigned
	Signed	Treat inputs as signed
Width	Integer > 0	Width of inputs A and B

Pins

Type	Name	Option	Explanation
In	DATAA[Width-1:0]	No	Input data A
In	DATAB[Width-1:0]	No	Input data B
Out	AEB	No	1 if A = B, 0 otherwise
Out	ALB	No	1 if A < B, 0 otherwise
Out	AGB	No	1 if A > B, 0 otherwise

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
cpl16	46.73	21.4	385	18x50	2205	1.56	5.73
cpl8	46.08	21.7	129	10x26	613	0.82	4.75



Representation
● Unsigned ○ Signed

Pipeline 16

Width 1

Macro Name cpl16

Pin Map File Name

User Library user.lib

Options
☒ Hard Macro
☐ Generate Schematic

Counter – Johnson

The Counter generator can be used to generate Johnson Counters. The following parameters are available.

Parameters

Parameter	Value	Explanation
Counter	Johnson	Create a Johnson counter
	Ripple	Create a Ripple Carry counter
Direction	Up	Create an Up counter
	Down	Create a Down counter (apply to Ripple only)
	UpDown	Create an Up/Down counter (apply to Ripple only)
Pipeline	Integer >= 1	Add pipeline register every *n* cells (not available for Johnson or Up/Down counter)
Width	Integer > 1	Width of input and output vectors
Enable	Boolean	Add an enable pin to component
Fold	Boolean	Fold layout in half (not available for up/down counter)
Loadable	Boolean	Add parallel load capability

Pins

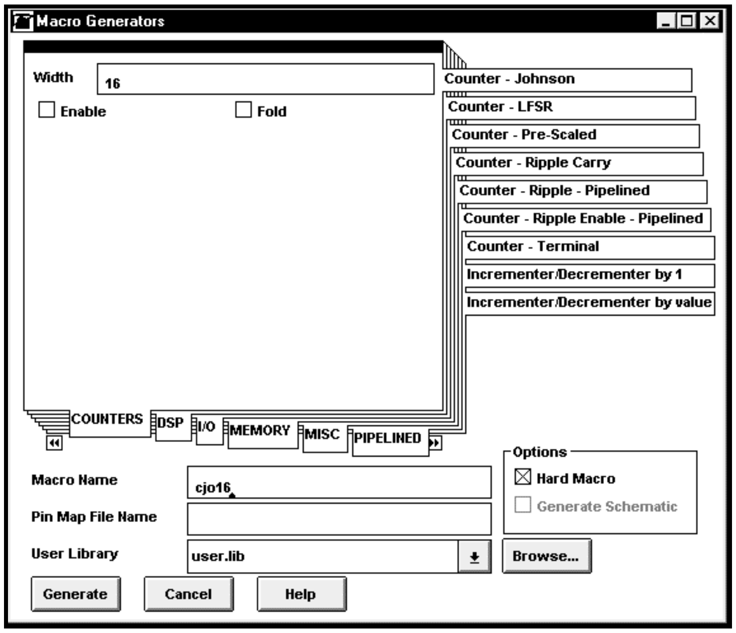
Type	Name	Option	Explanation
In	R	No	Reset, active low
In	CLK	No	Clock
In	ENABLE	Yes	Enable counter
Out	Q[Width-1:0]	No	Counter output

Truth Table

Input			Output
R	CLK	ENABLE	Q[Width-1:0]
0	X	X	0
1	X	0	Q[Width-1:0]
1	R	1	Q[Width-2:0]Q[Width-1]

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
cjo16	104.17	9.6	16	1x16	121	0.2	7.53
cjo8	136.99	7.3	8	1x8	61	0.19	7.56



Width 16

☐ Enable ☐ Fold

Counter - Johnson
Counter - LFSR
Counter - Pre-Scaled
Counter - Ripple Carry
Counter - Ripple - Pipelined
Counter - Ripple Enable - Pipelined
Counter - Terminal
Incrementer/Decrementer by 1
Incrementer/Decrementer by value

Macro Name cjo16

Pin Map File Name

User Library user.lib

Options
☒ Hard Macro
☐ Generate Schematic

Counter – LFSR

The LFSR generator can be used to create a high speed divide-by-[2**(n-1) -1] counter, where *n* is the number of bits. The count sequence can be output in a format specified by a user-supplied data file (described below).

Parameters

Parameter	Value	Explanation
ListCount	Component	Generate component only - do not output count sequence
	Count	Output count sequence only - do not generate component
	Both	Output count sequence and generate component
Width	Integer 3-64	Width of input and output data

Pins

Type	Name	Option	Explanation
In	R	No	Reset (active low)
In	CLK	No	Clock
Out	Q[Width-1:0]	No	Counter output

Truth Table

Input		Output
R	CLK	Q[W-1:0]
0	X	0
1	X	Present State
1	R	lfsr Next state

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
clf16	72.46	13.8	21	1x21	133	0.21	6.31
clf8	84.75	11.8	13	1x13	73	0.20	5.58

Count Sequence Output Specified ranges of the count sequence can be output to the file *component_name.lst*. These ranges are specified in file lfsr.dat which should be placed in the design directory. The format of the file is basically a header line with either hex or dec to indicate that the numbers in the file are either Hexa decimal or Decimal format. Each subsequent line should be in the form of Count or Range: Start_Count End_Count. For example:

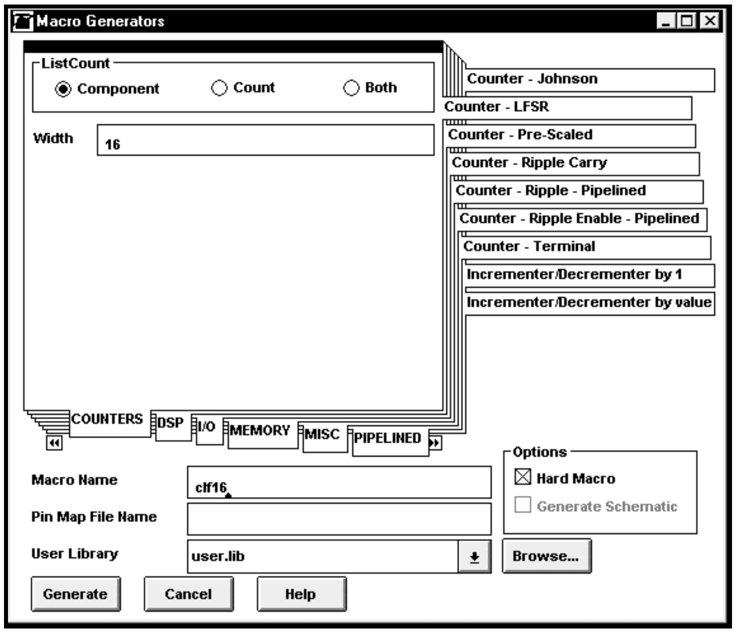
lfsr.dat

```
dec
0
4 20
```

88 127
300 844
1088

The output will have the format Count *count_number* Decode *decode_value*, where *decode_value* is the counter Q[Width-1:0] output for *count_number*.

The generator options allow the count sequence to be output without generating a component. If this option is selected, the program flow will stop right after the macro generator is run with an Error message. This error can be ignored.



ListCount: [✓ Component] Count Both

Width: 16

Macro Name: clf16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic

Counter – Pre-Scaled

The Counter generator can be used to generate a high speed Pre-Scaled Counter. This counter takes advantage of pre-scaled logic to generate the carry enable signals for each count bit which allows for faster operation than traditional carry enable generation methods. The following parameters are available for this counter.

Parameters

Parameter	Value	Explanation
Counter	Fast	Create a Pre-Scaled counter
Width	Integer (3:64)	Width of input and output vectors

Pins

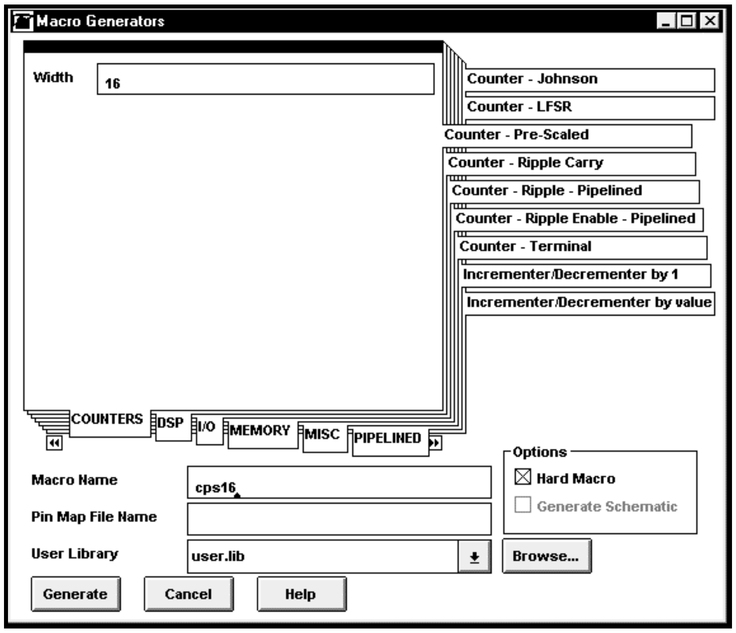
Type	Name	Option	Explanation
In	R	No	Reset (active low)
In	CLK	No	Clock
In	CI	No	Carry In (from previous stage) must be a 1 to enable counter
In	DATA[Width-1:0]	Yes	Parallel load input
In	ENABLE	Yes	Enable counter
In	SLOAD	Yes	Load signal (active low)
Out	Q[Width-1:0]	No	Counter output
Out	RCO	No	Counter carry out (will be a 1 when count has reached its max value)

Truth Table

Input						Output	
R	ENABLE	CLK	CI	SLOAD	DATA[W-1:0]	Q[W-1:0]	RCO
0	X	X	X	X	X...X	0…0	0
1	0	X	X	1	X...X	QW...Q0	0
1	X	R	0	0	DW…D0	DW…D0	0
1	1	R	1	0	DW…D0	DW…D0	0
1	1	X	1	1	X...X	Present State	W*…*Q1*Q0
1	1	R	1	1	DW…D0	DW…D0+1	QW*…*Q1*Q 0

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cel l
cps16	20.83	48	99	07x18	362	0.31	3.65
cps8	34.25	29.2	51	07x10	182	0.25	3.56



Width 16

Macro Name cps16

Pin Map File Name

User Library user.lib

[✓ Hard Macro]

Generate Schematic

Counter – Ripple Carry

The Counter generator can be used to generate Ripple Carry Counters. The following parameters are available for the counters.

Parameters

Parameter	Value	Explanation
Counter	Johnson	Create a Johnson counter
	Ripple	Create a Ripple Carry counter
Direction	Up	Create an Up counter
	Down	Create a Down counter (apply to Ripple only)
	UpDown	Create an Up/Down counter (apply to Ripple only)
Pipeline	Integer >= 1	Add pipeline register every *n* cells (not available for Johnson or Up/Down counter)
Width	Integer > 1	Width of input and output vectors
Enable	Boolean	Add an enable pin to component
Fold	Boolean	Fold layout in half (not available for up/down counter)
Loadable	Boolean	Add parallel load capability

Pins

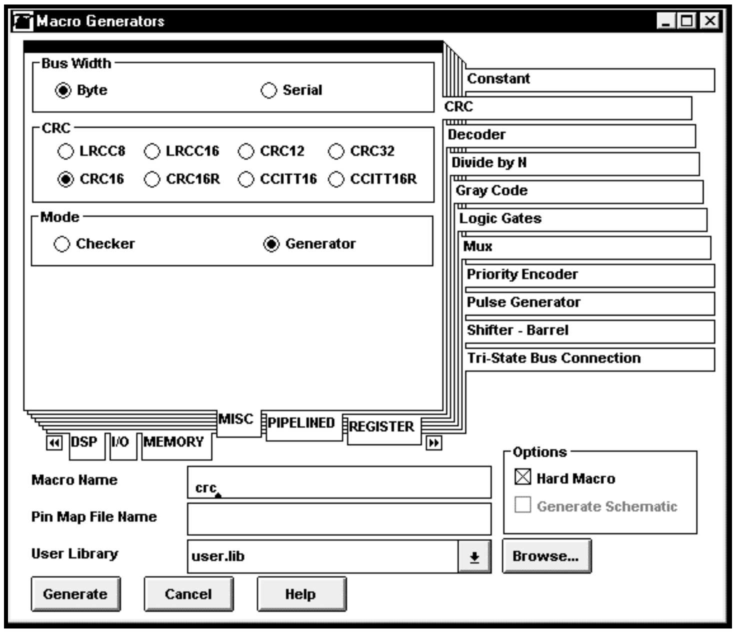
Type	Name	Option	Explanation
In	R	No	Reset (active low)
In	CLK	No	Clock
In	ENABLE	Yes	Enable counter
In	DATA[Width-1:0]	Yes	Parallel load input
In	SLOAD	Yes	Load signal (active low)
In	UP_DOWN	Yes	Up/Down control Up=1
Out	Q[Width-1:0]	No	Counter output
Out	RCO	No	Carry out

Truth Table

Input						Output	
R	ENABLE	CLK	SLOAD	DATA[W-1:0]	UP/DOWN	Q[W-1:0]	RCO
0	X	X	X	X…X	X	0…0	0
1	0	X	1	X…X	X	QW…Q0	0
1	1	R	0	DW…D0	X	DW…D0	0
1	1	X	1	X…X	X	Present State	QW*…*Q1*Q0
1	1	R	1	DW…D0	1	DW…D0+1	QW*…*Q1*Q0
1	1	R	1	DW…D0	0	DW…D0-1	QW*…*Q1*Q0

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
crc16	26.18	38.2	32	2x16	181	0.22	5.64
crc8	51.02	19.6	16	2x8	89	0.20	5.53



Bus Width: [✓ Byte] Serial

CRC: LRCC8 LRCC16 CRC12 CRC32 [✓ CRC16] CRC16R CCITT16 CCITT16R

Mode: Checker [✓ Generator]

Macro Name: crc

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic

Counter – Ripple Carry, Fast

The Counter generator can be used to generate Ripple Carry Counters. The FAST counter only supports direction UpDown. The following parameters are available for the counter.

Parameters

Parameter	Value	Explanation
Counter	Fast	Create a Ripple Carry FAST counter
Width	Integer > 1	Width of input and output vectors
Loadable	Boolean	Add parallel load capability

Pins

Type	Name	Option	Explanation
In	R	No	Reset (active low)
In	CLK	No	Clock
In	ENABLE	No	Enable counter, active high
In	RCID	No	Carry In/DOWN version, active high
In	RCIU	No	Carry In/UP version, active high
In	DATA[Width-1:0]	Yes	Parallel load data input
In	SLOAD	Yes	Load signal (active low)
In	UP_DOWN	No	UP/DOWN control, UP if UP_DOWN=1
Out	Q[Width-1:0]	No	Counter data output
Out	RCO	No	Carry out, UP or DOWN
Out	RCOD	No	Carry out, DOWN
Out	RCOU	No	Carry Out, UP

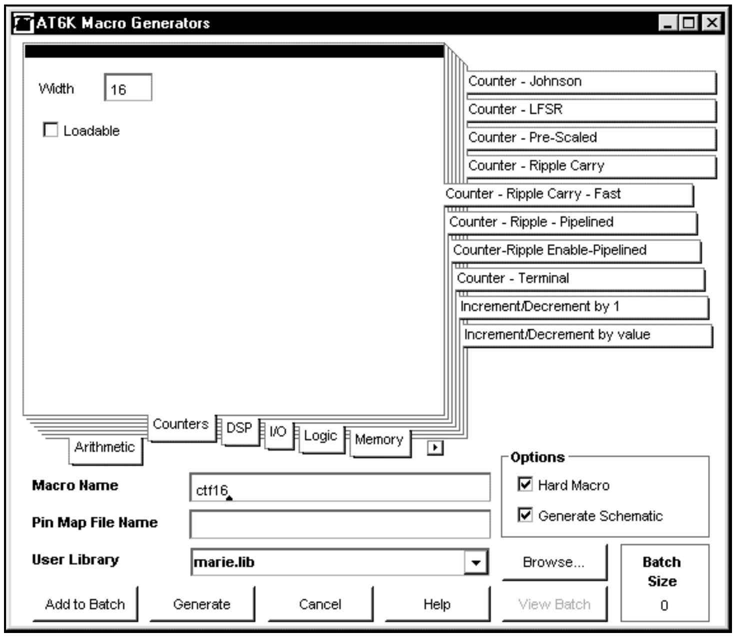
Truth Table

Input							
R	ENABLE	RCIU	RCID	CLK	SLOAD	DATA[W-1:0]	UP_DOWN
0	0	0	0	X	X	X…X	X
1	0	X	X	X	1	X…X	X
1	1	X	X	R	0	DW...D0	X
1	0	0	X	R	1	X…X	1
1	0	X	0	R	1	X…X	0
1	1	1	X	R	1	DW...D0	1
1	1	X	1	R	1	DW...D0	0

Output			
Q[W-1:0]	RCONU	RCOND	RCO
0…0	0	0	0
QW...Q0	X	X	X
DW...D0	0	0	0
Present State	QW*...*Q1*Q0	X	QW*...*Q1*Q0
Present State	X	QW*...*Q1*Q0	QW*...*Q1*Q0
DW...D0+1	QW*...*Q1*Q0	X	QW*...*Q1*Q0
DW...D0-1	X	QW*...*Q1*Q0	QW*...*Q1*Q0

Statistics

Name	**Speed** (MHz)	**Delay** (ns)	**Cells**	**Size** (x*y)	**Gates** (ttl equiv)	**Power** (mA/MHz)	**Gates/Cell**
ctf16	29.9401	33.4	80	4x16	328	4.1	0.1752
ctf8	43.1034	23.2	40	4x8	164	4.1	0.0856


AT6K Macro Generators

Width 16

☐ Loadable

Macro Name: ctf16

Pin Map File Name:

User Library: marie.lib

Options
☑ Hard Macro
☑ Generate Schematic

Batch Size 0

Counter – Ripple Carry, Pipelined

The Counter generator has an optional pipeline parameter that can be used to create a ripple carry counter with pipelining. An Up or Down Counter can be created.

Parameters

Parameter	Value	Explanation
Counter	Johnson	Create a Johnson counter
	Ripple	Create a Ripple Carry counter
Direction	Up	Create an Up counter
	Down	Create a Down counter (applies to Ripple only)
	UpDown	Create an Up/Down counter (applies to Ripple only)
Pipeline	Integer >= 1	Add pipeline register every *n* cells (not available for Johnson or Up/Down counter)
Width	Integer > 1	Width of input and output vectors
Enable	Boolean	Add an enable pin to component
Fold	Boolean	Fold layout in half (not available for up/down counter)
Loadable	Boolean	Add parallel load capability

Pins

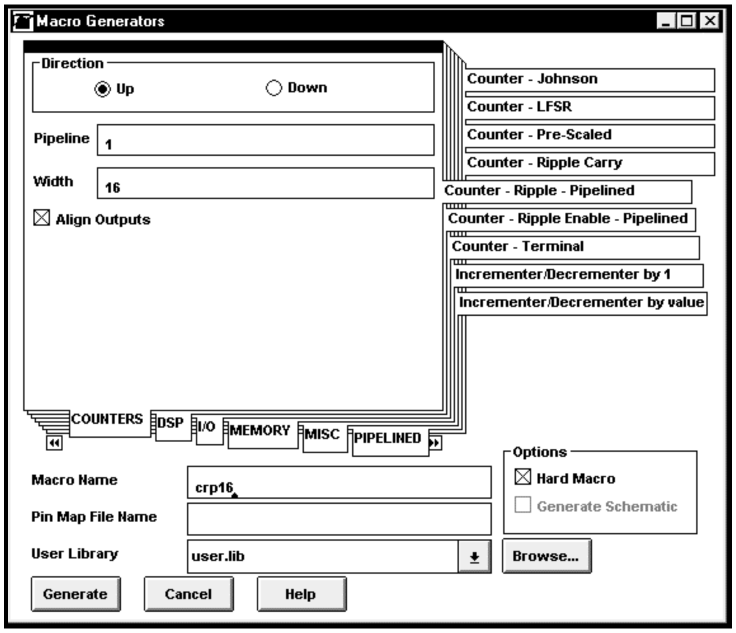
Type	Name	Option	Explanation
In	R	No	Reset (active low)
In	CLK	No	Clock
In	ENABLE	Yes	Enable counter
In	DATA[Width-1:0]	Yes	Parallel load input
In	SLOAD	Yes	Load signal (active low)
In	UP_DOWN	Yes	Up/Down control Up=1
Out	Q[Width-1:0]	No	Counter output
Out	RCO	No	Carry out

Truth Table

Input						Output	
R	ENABLE	CLK	SLOAD	DATA[W-1:0]	UP/DOWN	Q[W-1:0]	RCO
0	X	X	X	X…X	X	0…0	0
1	0	X	1	X…X	X	QW…Q0	0
1	0	R	0	DW…D0	X	DW…D0	0
1	1	X	1	X…X	X	Present State	QW*…*Q1*Q0
1	1	R	1	DW…D0	1	DW…D0+1	QW*…*Q1*Q0
1	1	R	1	DW…D0	0	DW…D0-1	QW*…*Q1*Q0

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
crp16	107.53	9.3	179	16x30	1178	1.25	6.58
crp8	107.53	9.3	55	8x14	336	0.66	6.11



Direction: [✓ Up] Down

Pipeline: 1

Width: 16

[✓ Align Outputs]

Macro Name: crp16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic

Counter – Terminal

The Counter generator can be used to generate Terminal Counters. The following parameters are available for the counters.

Parameters

Parameter	Value	Explanation
Counter	Terminal	Create a Terminal counter
Width	Integer > 1	Width of input and output vectors

Pins

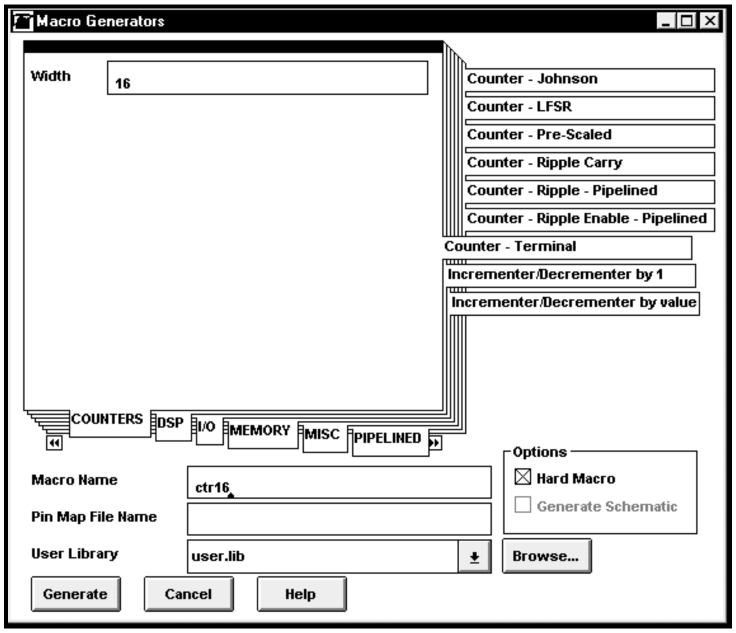
Type	Name	Option	Explanation
In	R	No	Reset (active low)
In	CLK	No	Clock
In	DATA[Width-1:0]	Yes	Parallel load input
In	SLOAD	Yes	Load signal (active low)
Out	TERMCNT	No	Terminal Signal

Truth Table

TERMCNT pin will go high after the number of clock cycles matches the value present on DATA pins after the RESET pin has been released.

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
ctr16	11.96	83.6	116	7x17	464	0.43	4.00
ctr8	20.37	49.1	60	7x9	232	0.34	3.87



Width: 16

Macro Name: ctr16

Pin Map File Name:

User Library: user.lib

Options
☒ Hard Macro
☐ Generate Schematic

CRC

The CRC (cyclic redundancy code) checker and generator macros can be used to calculate a CRC or to check for a valid CRC on bit-serial, or byte-parallel data streams.

Parameters

Parameter	Value	Explanation
CRC	LRCC8	Use cyclic redundancy code LRCC-8
	LRCC16	Use LRCC-16
	CRC12	Use CRC-12
	CRC16	Use CRC-16
	CRC16R	Use CRC-16 Reverse
	CCITT16	Use CCITT CRC-16 (SDLC)
	CCITT16R	Use CCITT CRC-16 Reverse (SDLC Reverse)
	CRC32	Use CRC-32 (Ethernet)
Mode	Checker	Create CRC checker macro
	Generator	Create CRC generator macro
Width	Serial	Create bit-serial data input/output interface
	Byte	Create byte-parallel data input/output interface

Pins

Type	Name	Option	Explanation
In	CLK	No	Bit or Byte clock
In	R	No	Reset, active low, (clears CRC register)
In	DATA_IN[Width-1:0]	No	Input data (bit serial, or byte wide)
In	ENABLE	No	Enables all operations
In	ACC	No	Enables CRC calculation (accumulates input data)
In	READ_CRC	No	Read CRC word after ac-cumulation (serial generators only)
In	SELECT	No	Select CRC output byte in 12, 16, and 32 bit parallel generators (1 or 2 bits wide)
Out	AC_IND	No	Indicates that the circuit is accumulating
Out	READ_IND	No	Indicates that the CRC register contents are being read
Out	DATA_OUT[Width-1:0]	No	Data output (bit serial, or byte wide
Out	CRC_OK	No	Indicates a valid CRC check

To calculate a CRC word for a serial data stream, the ENABLE and ACC inputs should be active (READ _CRC inactive) throughout the application of a bitstream on DATA_IN. When ENABLE and READ _CRC are active (ACC is inactive) following the application of an input bitstream, the register contents are output on DATA_OUT, least significant bit first.

Bit-Serial CRC Check

To perform a bit-serial CRC check, ENABLE and ACC should be active throughout the duration of the input bitstream on DATA_IN. A valid CRC check is indicated by CRC_OK after the bitstream has been applied.

Byte-Parallel CRC Generation

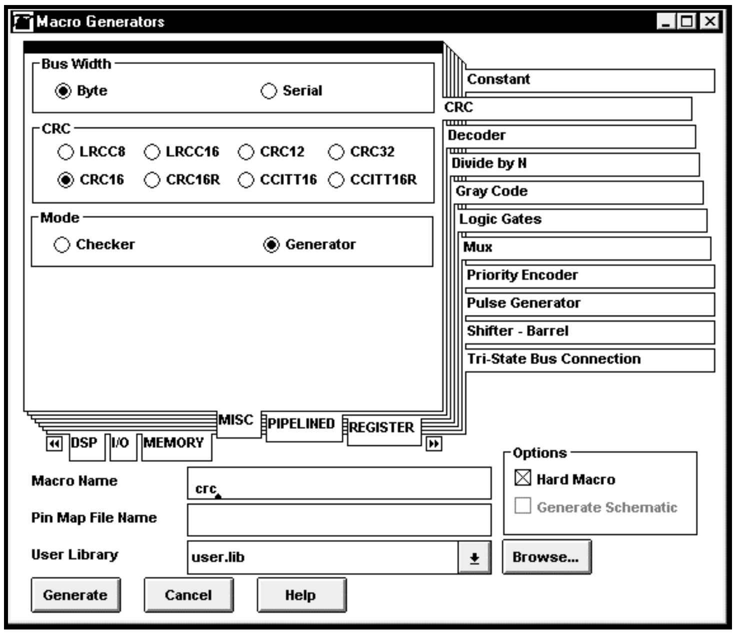
In byte-parallel CRC generators, ENABLE and ACC should be active throughout the application of the data input sequence. The SELECT input can be used to read the CRC contents following a calculation. For 12- and 16-bit generators, SELECT = 0 will output the least significant byte of the register, and SELECT = 1 will output the most significant byte. In the 32-bit generator, SELECT = 00 will select the least significant byte, and SELECT = 11, the most significant byte.

Byte-Parallel CRC Check

ENABLE and ACC should be active throughout the application of the input data stream. Following accumulation, a valid CRC check is indicated by the CRC_OK output.

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
CRC	20.70	48.3	120	09x18	325	0.5	2.71



Bus Width
[✓ Byte] Serial

CRC
LRCC8 LRCC16 CRC12 CRC32
[✓ CRC16] CRC16R CCITT16 CCITT16R

Mode
Checker [✓ Generator]

Macro Name: crc

Pin Map File Name:

User Library: user.lib

Options
[✓ Hard Macro]
Generate Schematic

Decoder

The Decoder can be used to generate a full or partial Decode of the specified number of bits of input or output.

Parameters

Parameter	Value	Explanation
No. of values to decode	Integer >= 0	Width of output vector. If this value is left at 0, the number of outputs will be determined by $2^{\text{Input Width}}$ (i.e. all values will be decoded)
Input Width	Integer >= 0	Width of input vector. If this value is left at 0, the input width will be determined by the number of outputs specified
Starting at value	Integer >= 0	First value to decode. When used in conjunction with the previous parameter, this can be used to specify exactly the range of outputs to decode. For example, setting the “No. of values to decode” parameter to 3 and “Starting at value” to 2 would decode the values 2, 3 and 4.
Output	Activehigh	Output logic level will be high for the decoded input
	Activelow	Output logic level will be low for the decoded output
InRegister	Boolean	Register inputs on the Decoder
OutRegister	Boolean	Register outputs on the Decoder

Pins

Type	Name	Option	Explanation
In	DATA[Width-1:0]	No	Decoder input pins, either Width bits wide or $\log_2$ Decodes bits wide
Out	EQ[Decodes-1:0]	No	Decoder output, either $2^{\text{Input Width}}$ bits wide or Decode bits wide. If both Width and Decodes are specified, the output will start at the lowest count and work up

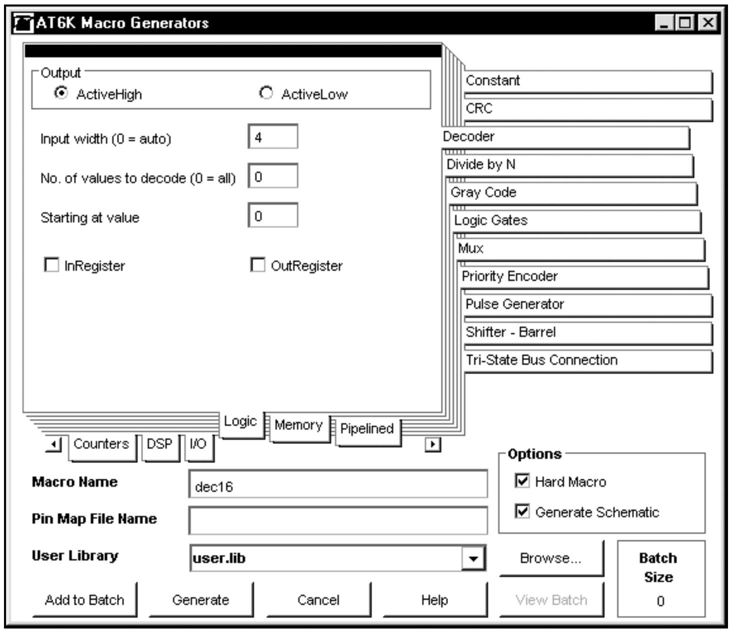
Truth Table

Option	Explanation
DATA[W-1:0]	EQ[Decodes-1:0]
0	0...001
1	0...010
2	0...100
. .	. .

.	.

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
dec16	120.48	8.3	64	4x16	64	0.12	1.00
dec8	163.93	6.1	24	3x8	22	0.04	0.92



Output: [✓ ActiveHigh] ActiveLow

Input width (0 = auto): 4

No. of values to decode (0 = all): 0

Starting at value: 0

☐ InRegister ☐ OutRegister

Macro Name: dec16

Pin Map File Name:

User Library: user.lib

Options: ☑ Hard Macro ☑ Generate Schematic

Batch Size: 0

Deductor

The Deductor subtracts a given amount from the register initial value. The functional description of the Deductor is as follows:

```
always(@posedge CLK or negedge R)
begin
        if(R == ‘b0)
                SUM = 0;
        else if (ACC)
                {COUT, SUM} = SUM - DATA - CIN;
end
```

Parameters

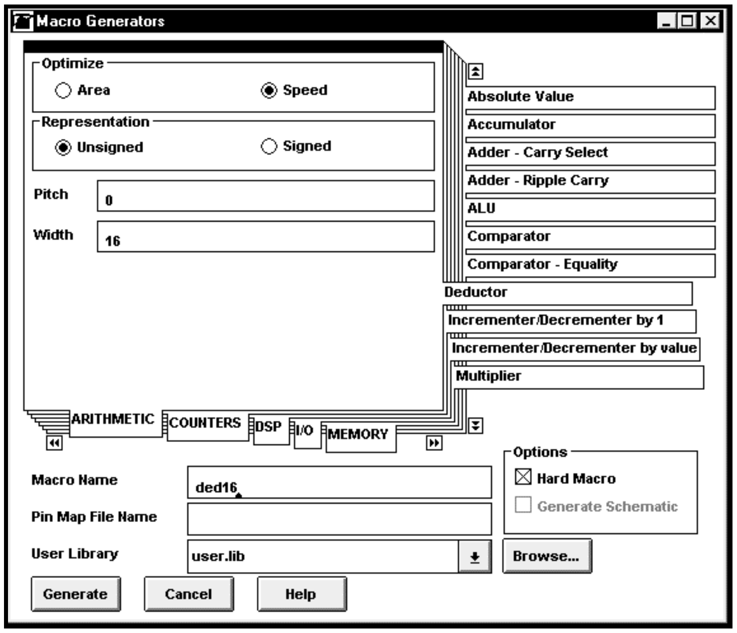
Parameter	Value	Explanation
Optimize	Area	Create an area efficient layout
	Speed	Create a fast layout
Representatio n	Unsigned	Treat inputs as unsigned
	Signed	Treat inputs as signed
Pitch	Integer > 1	Spacing between input pins, pitch of 2 means one cell between input pins (not supported for pipelining)
Pipeline	Integer > 0	Create a pipeline every *n* stages
Width	Integer > 1	Width of input and output vectors

Pins

Type	Name	Option	Explanation
In	CIN	No	Carry in
In	DATA[Width-1:0]	No	A input
In	ACC	No	Accumulate control (can-not be used with pipelining)
In	CLK	No	Clock
In	R	No	Reset (active low)
Out	SUM[Width-1:0]	No	Deductor output
Out	COUT	No	Carry out

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cel l
ded16	17.67	56.6	144	9x16	345	0.55	2.40
ded8	26.60	37.6	72	9x8	173	0.37	2.40



Optimize: Area [✓ Speed]

Representation: [✓ Unsigned] Signed

Pitch: 0

Width: 16

Macro Name: ded16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic

Deductor - Pipelined

The Deductor subtracts a given number from the register initial value. Pipelining is offered in this function. The functional description of the Deductor is as follows:

```
always(@posedge CLK or negedge R)
begin
        if(R == ‘b0)
                SUM = 0;
        else if (ACC)
                {COUT, SUM} = SUM - DATA - CIN;
end
```

Parameters

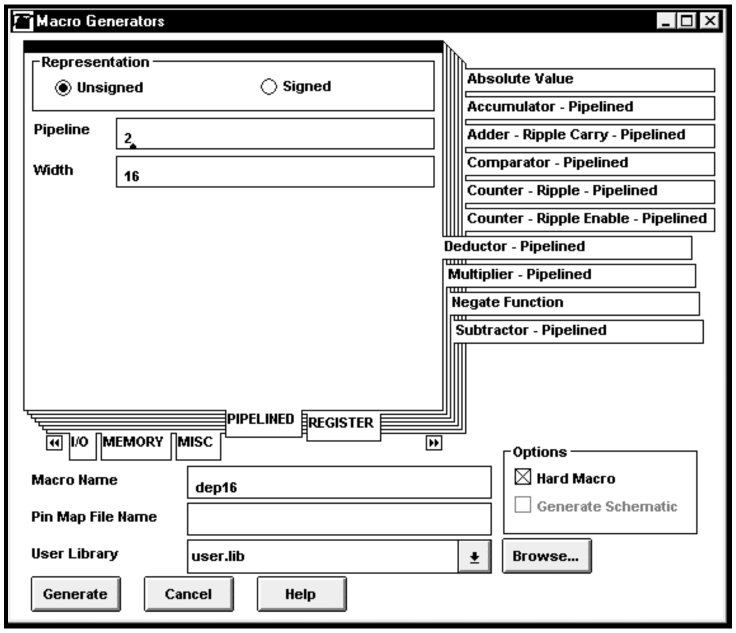
Parameter	Value	Explanation
Optimize	Area	Create an area efficient layout
	Speed	Create a fast layout
Representati on	Unsigned	Treat inputs as unsigned
	Signed	Treat inputs as signed
Pitch	Integer > 1	Spacing between input pins, pitch of 2 means one cell between input pins (not supported for pipelining)
Pipeline	Integer > 0	Create a pipeline every *n* stages
Width	Integer > 1	Width of input and output vectors

Pins

Type	Name	Option	Explanation
In	CIN	No	Carry in
In	DATA[Width-1:0]	No	A input
In	ACC	No	Accumulate control (cannot be used with pipelining)
In	CLK	No	Clock
In	R	No	Reset (active low)
Out	SUM[Width-1:0]	No	Deductor output
Out	COUT	No	Carry out

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
dep16	50.51	19.8	251	14x40	1452	1.23	5.78
dep8	50.51	19.8	85	8x20	417	0.67	4.91



Macro Generators

Representation: [✓ Unsigned] Signed

Pipeline: 2

Width: 16

Macro Name: dep16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic

Divide by N

Every time a pulse (clock pulse) is input to the Divide By N circuit, the output of this circuit produces a single pulse for every complete cycle of *n* input pulses. The number of output pulses is the number of input pulses divided by *n*.

Parameters

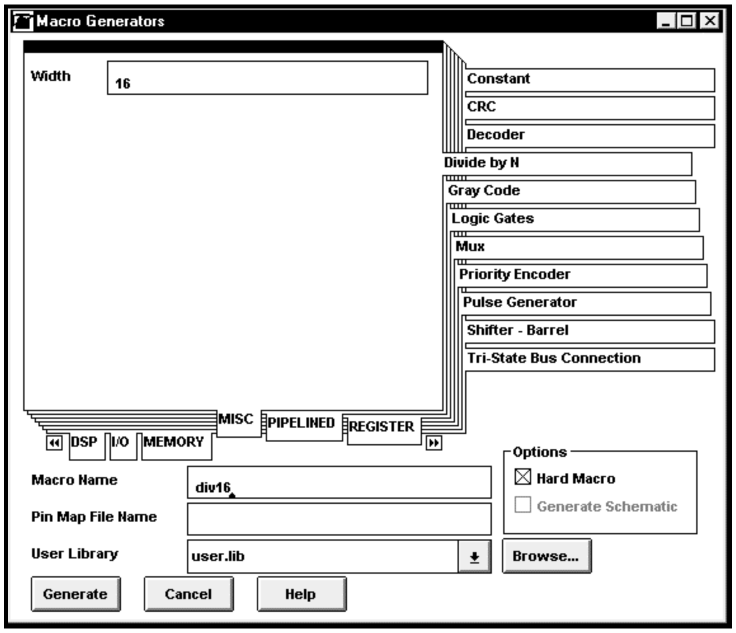
Parameter	Value	Explanation
Width	Integer > 1	How much to divide the clock by

Pins

Type	Name	Option	Explanation
In	R	No	Reset (active low)
In	CLK	No	Clock
Out	Q	No	Output

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
div16	149.25	6.7	8	1x8	61	0.19	7.56
div8	149.25	6.7	4	1x4	31	0.19	7.63



Width: 16

Macro Name: div16

Pin Map File Name:

User Library: user.lib

Options
☒ Hard Macro
☐ Generate Schematic

FIFO

The FIFO generator can be used to create a FIFO macro optimized for width or depth. This implementation is a synchronous elastic store rather than a standard FIFO, with a somewhat different operation as described below.

Write Operation

Fall-through Delay: Data written does not immediately "fall through" to the output, but is clocked through the internal registers by CLK. It will require D clocks to appear on the output pins, where D = FIFO Depth.

Data is latched by the rising edge of CLK while WEN is asserted. If appropriate, FULL will assert on this clock edge, after data has been written. Data written while FULL is asserted will be ignored.

Read Operation

When EMPTY negates, data is already present in the output register (and thus on the output pins). The next word will be clocked onto the output pins on the first rising edge of CLK on which REN is asserted.

EMPTY Latency

EMPTY is really a Data Available flag. It will be negated D clock cycles after a word is written, and will reassert when all words written have been read. It differs from a regular FIFO in that, while it responds immediately to a read, due to fall-through delay there is a D clock latency in response to a write. This means that EMPTY can be asserted even with up to (D-1) words in the FIFO.

Parameters

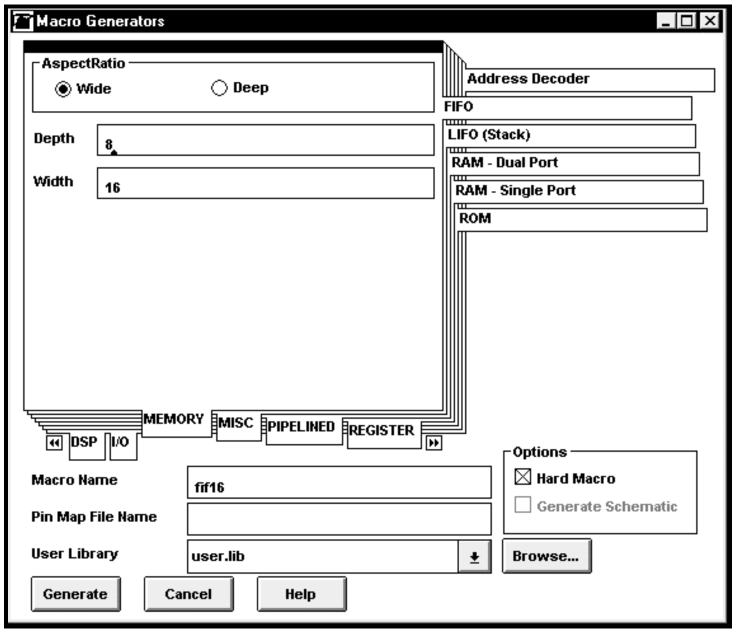
Parameter	Value	Explanation
AspectRatio	Wide	Minimize cells per bit (within each word)
	Deep	Minimize cells per word
Width	Integer 2-64	Width of parallel input and output data
Depth	Integer 3-64	Depth, must be greater than 2 for Deep

Pins

Type	Name	Option	Explanation
In	D[Width-1:0]	No	Data input
In	R	No	Reset, active low
In	CLK	No	Clock
In	REN	No	Read Enable pin (active low)
In	WEN	No	Write Enable pin (active low)
Out	Q[Width-1:0]	No	Data output
Out	FULL	No	FIFO full flag (active low)
Out	EMPTY	No	FIFO empty flag (active low)

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
fif16	21.41	46.7	415	19x23	1778	1.66	4.28
fif8	22.32	44.8	231	11x23	946	0.97	4.10



AspectRatio: [✓ Wide] Deep

Depth: 8

Width: 16

Macro Name: fif16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic

FIR

The FIR Filter generator generates a bit serial architecture for an 8-bit 8-tap FIR Filter. It contains delay modules, serial to parallel multipliers, and a column adder. Data is treated as signed or unsigned by user specification. The 8 coefficients are read in as command-line arguments. The 1-bit input signal is the serial data for the FIR Filter which inputs bit by bit. The 1-bit output is the serial output of the FIR Filter.

Parameters

Parameter	Value	Explanation
Representati on	Signed	Input and output is signed (2's complement)
	Unsigned	Input and output is unsigned (default)
Coefficient	8 Integers	Coefficients for the FIR Filter

Pins

Type	Name	Option	Explanation
In	INPUT	No	Serial input signal
In	RST1	No	Global reset, active low
In	RST2	No	Reset macro except for delay circuitry (every 20 clock cycles)
Out	OUTPUT	No	Serial output signal

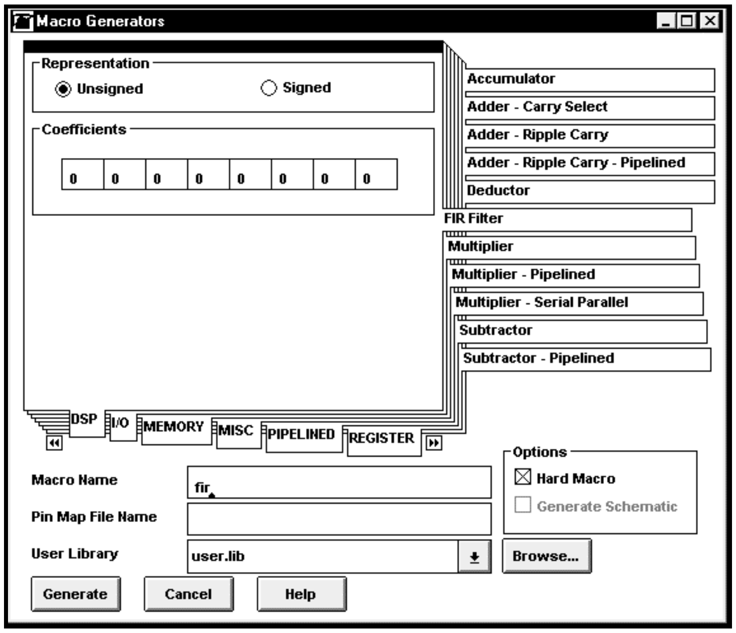
Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
fir	57.14	17.5	751	27x32	3139	1.99	4.18

The value of the coefficient is programmed as the parallel input of one of the 8 serial-parallel multipliers in the FIR Filter. The serial data is input bit by bit, the output is the discrete convolution of the input signal and specified set of coefficients. The output consists of a full 16-bit product plus an additional 4 bits for overflow. The RST2 signal resets the multipliers and the column adder every 20 clock cycles.

Mathematically, the FIR Filter calculates:

```
Y[k] =    C_(0)*X_(k-0) +
               C_(1)*X_(k-1) +
               ....
               C_(n)*X_(k-n)
```


Macro Generators

Representation
Unsigned [✓] Signed

Coefficients
0 0 0 0 0 0 0 0

Accumulator
Adder - Carry Select
Adder - Ripple Carry
Adder - Ripple Carry - Pipelined
Deductor
FIR Filter
Multiplier
Multiplier - Pipelined
Multiplier - Serial Parallel
Subtractor
Subtractor - Pipelined

DSP I/O MEMORY MISC PIPELINED REGISTER

Macro Name: fir
Pin Map File Name:
User Library: user.lib

Options
[✓] Hard Macro
[ ] Generate Schematic

Flip-Flop – D Type

The DFF generator can be used to create a D-type flip-flop.

Parameters

Parameter	Value	Explanation
Enable	Boolean	Add an Enable pin to work with both serial and parallel
Pitch	Integer > 0	Spacing between input pins. Pitch of 2 means one cell between inputs
Setlow	Boolean	Asynchronous active low set
Width	Integer > 0	Width of parallel input and output data

Pins

Type	Name	Option	Explanation
In	DATA[Width-1:0]	Yes	Data input for InputMode = Parallel
In	CLK	No	Clock Pin
In	ENABLE	Yes	Data Enable input (active low)
In	R	No	Reset Pin (active low)
Out	Q[Width-1:0]	Yes	Data output for OutputMode = Parallel or Both

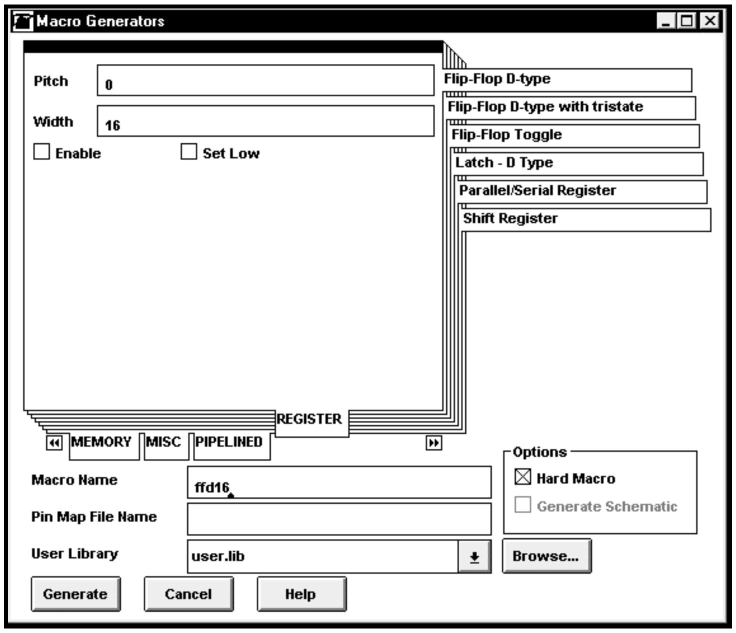
Truth Table

Input				Output
R	DATA[W-1:0]	CLK	ENABLE	Q[W-1:0]
0	X	X	X	0
1	X	0>1	X	No Change
1	0	0>1	1	0
1	1	0>1	1	1
1	X	0>1	1	Q(I) = Q-(I-1) Q(0) = SI

Note: Q- is the value of Q preceding the clock transition

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
ffd16	454.55	2.2	16	1x16	120	0.19	7.50
ffd8	454.55	2.2	8	1x8	60	0.19	7.50



Pitch: 0

Width: 16

☐ Enable ☐ Set Low

Macro Name: ffd16

Pin Map File Name:

User Library: user.lib

Options

☒ Hard Macro

☐ Generate Schematic

Flip-Flop – D Type, Tri-state

The DFF generator can be used to generate a D-type flip-flop with Tri-State output.

Parameters

Parameter	Value	Explanation
Enable	Boolean	Add an Enable pin to control the input signals
IndividualEnable	Boolean	Add an Enable pin to control each tri-state output signal separately
Pitch	Integer > 0	Spacing between input pins. Pitch of 2 means one cell between inputs.
Width	Integer > 0	Width of parallel input and output data

Pins

Type	Name	Option	Explanation
In	DATA[Width-1:0]	Yes	Data input for InputMode = Parallel
In	CLK	No	Clock Pin
In	ENABLE	Yes	Data Enable input (active low)
In	OE[Width-1:0]	Yes	Individual Enable pin
In	R	No	Reset Pin (active low)
Out	Q[Width-1:0]	Yes	Data output for OutputMode = Parallel or Both

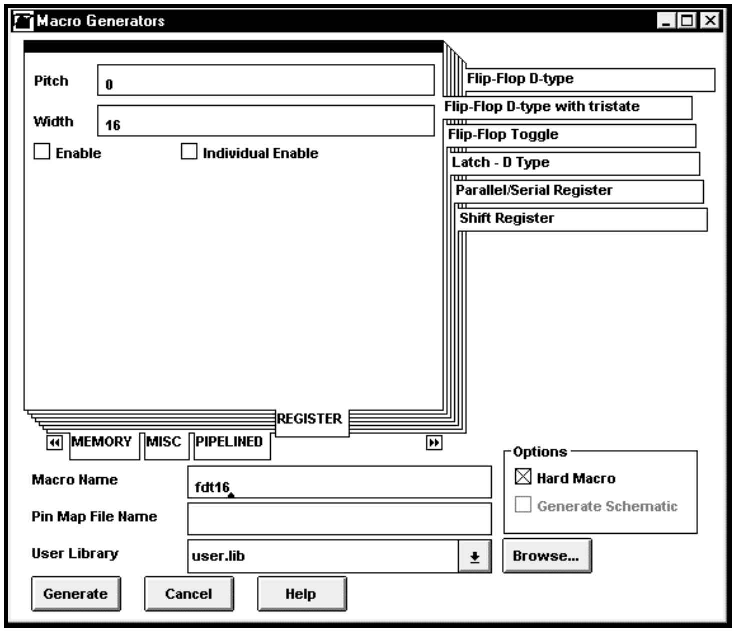
Truth Table

Input					Output
R	DATA[W-1:0]	CLK	ENABLE	OE	Q[W-1:0]
X	X	X	X	0	Z
0	X	X	X	1	0
1	DATA[W-1:0]	0>1	1	1	DATA[W-1:0]
1	DATA[W-1:0]	0>1	0	1	Q

Note: Q- is the value of Q preceding the clock transition

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
fdt16	344.83	2.9	16	1x16	128	0.23	8.00
fdt8	344.83	2.9	8	1x8	64	0.21	8.00



Pitch: 0

Width: 16

☐ Enable ☐ Individual Enable

Macro Name: fdt16

Pin Map File Name:

User Library: user.lib

Options

☒ Hard Macro

☐ Generate Schematic

Flip-Flop – Toggle

Parameters

Parameter	Value	Explanation
Enable	Boolean	Add a Data Enable pin
Width	Integer > 0	Width of parallel input and output data

Pins

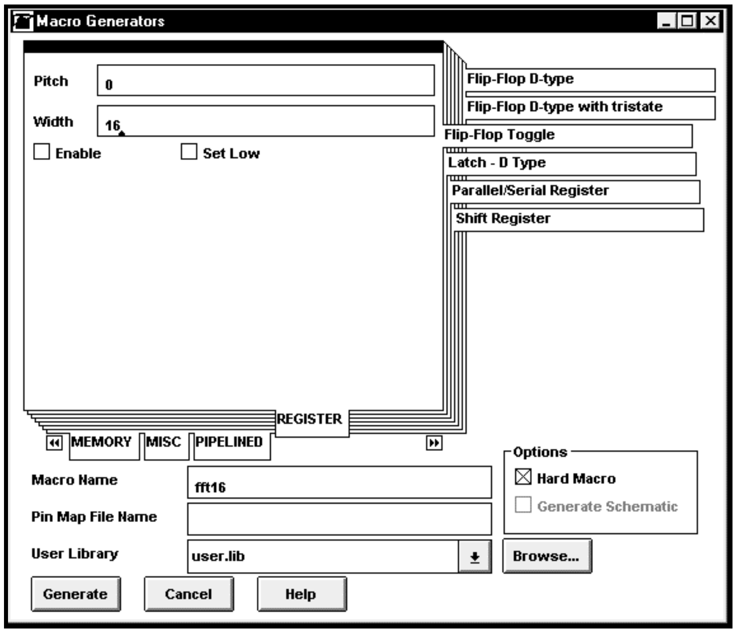
Type	Name	Option	Explanation
In	DATA[Width-1:0]	No	Data input
In	CLK	No	Clock pin
In	ENABLE	Yes	Data Enable input
In	R	No	Reset pin (active low)
Out	Q[Width-1:0]	No	Data output

Truth Table

Input				Output
R	DATA[W-1:0]	CLK	ENABLE	Q[W-1:0]
0	X	X	X	0
1	X	0>1	0	No Change
1	0	0>1	1	Q
1	1	0>1	1	~Q

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
fft16	178.57	5.6	32	2x16	184	0.22	5.75
fft8	178.57	5.6	16	2x8	92	0.20	5.75



Pitch 0

Width 16

☐ Enable ☐ Set Low

Macro Name fft16

Pin Map File Name

User Library user.lib

☒ Hard Macro

Gray Code

The Gray Code generator can be used to convert binary code to gray code or gray code to binary code.

Parameters

Parameter	Value	Explanation
Function	ToGray	Convert the binary code input to gray-code
	ToBinary	Convert the gray-code input to binary code
Pitch	Integer > 0	Spacing between input pins. Pitch of 2 means one cell between inputs
Width	Integer > 1	Width of input and output data

Pins

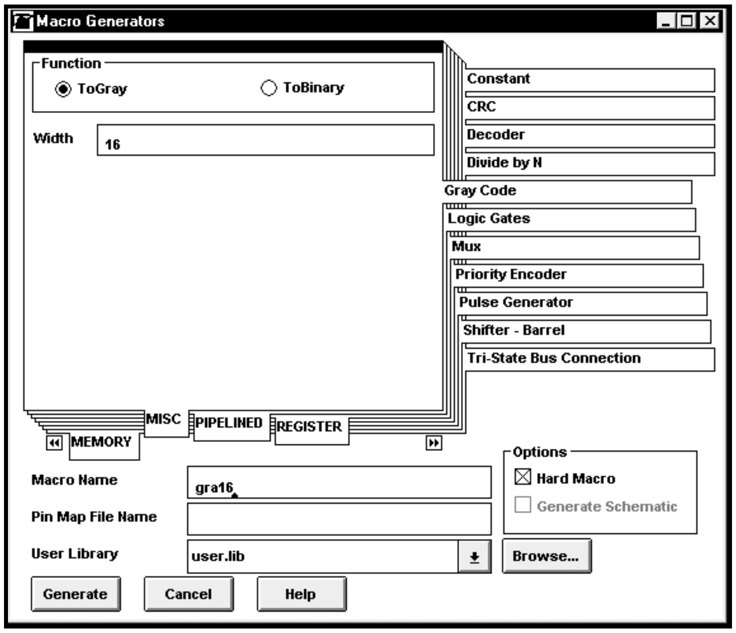
Type	Name	Option	Explanation
In	DATA[Width-1:0]	No	Input data to be converted
Out	Q [Width-1:0]	No	Output bits

Truth Table

Input		Output
Binary	DATA[Width-1:0]	DATAW, DATA[W-1] xor DATAW, DATA[W-2] xor DATA[W-1],…
Gray	DATA[Width-1:0]	DATAW, DATA[W-1] xor DATAW, DATA[W-2] xor DATA[W-1] xor DATAW, …

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
gra16	217.39	4.6	32	2x16	46	0.05	1.44
gra8	217.39	4.6	16	2x8	22	0.02	1.38



Function: [✓ ToGray] ToBinary

Width: 16

Macro Name: gra16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic

Increment/Decrement by 1

This generator can be used to create a macro which increments/ decrements a preloaded value or the data input at every rising edge of the clock by 1. The functional description of the generator is described as follows:

```
always @(posedge CLK or negedge R)
  begin
        if(R == ‘b0)
          SUM = 0;
        else if (ENABLE && !LOAD)  // Load is active low
          SUM = DATA;
        else if (ENABLE)
          begin
            if (INC_DEC)
                 {COUT, SUM} = SUM + 1;
            else
                 {COUT, SUM} = SUM - 1;
         end
  end
```

Parameters

Parameter	Value	Explanation
Function	Increment	Create an incrementer
	Decrement	Create a decrementer
	Inc_Dec	Create a generator with an Inc_Dec pin
Optimize	Area	Create an area efficient layout
	Speed	Create a fast layout
Pitch	Integer > 1	Spacing between input pins, pitch of 2 means one cell between input pins
Width	Integer > 1	Width of input and output vectors
Load	Boolean	Create a pin to load the generator with a value
Enable	Boolean	Add an enable pin to the component

Pins

Type	Name	Option	Explanation
In	ENABLE	Yes	Enable Incrementer/Decrementer
In	DATA[Width-1:0]	Yes	Parallel data input
In	INC_DEC	Yes	Increment/Decrement control. Increment when equal to 1 or Decrement otherwise
In	CLK	No	Clock
In	LOAD	Yes	Parallel load signal (active low)
In	R	No	Reset (active low)
Out	SUM[Width-1:0]	No	Incrementer/Decrementer output
Out	COUT	No	Carry out

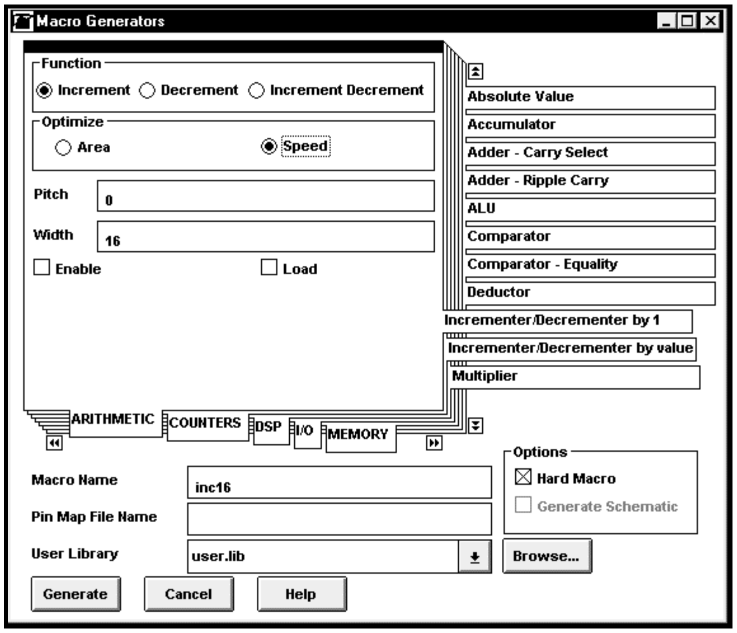
Truth Table

Input						Output	
R	ENABLE	CLK	LOAD	DATA[W-1:0]	INC_DEC	SUM[W-1:0]	COUT
0	X	X	X	X...X	X	0...0	0
1	0	X	1	X...X	X	SUMW...SUM0	0
1	1	R	0	DW...D0	X	DW...D0	0
1	1	X	1	X...X	X	Present State	SUMW*...*SUM1*SUM0
1	1	R	1	DW...D0	1	DW...D0+1	SUMW*...*SUM1*SUM0
1	1	R	1	DW...D0	0	DW...D0-1	SUMW*...*SUM1*SUM0

Note: The above truth table and the behavioral description are given for this generator when the function is set to Increment_Decrement.

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
inc16	19.38	51.6	112	7x16	314	0.41	2.80
inc8	30.77	32.5	56	7x8	154	0.29	2.75



Function: [✓ Increment] Decrement Increment Decrement

Optimize: Area [✓ Speed]

Pitch: 0

Width: 16

☐ Enable ☐ Load

Macro Name: inc16

Pin Map File Name:

User Library: user.lib

Options: ☒ Hard Macro ☐ Generate Schematic

Increment/Decrement by Value

This generator can be used to create a macro which increments/ decrements a preloaded value or the data input at every rising edge of the clock by an user defined value. The functional description of the generator is described as follows:

```
always @(posedge CLK or negedge R)
  begin
        if(R == ‘b0)
          SUM = 0;
        else if (ENABLE && !LOAD) // Load is active low
          SUM = DATA;
        else if (ENABLE)
          begin
            if (INC_DEC)
                 {COUT, SUM} = SUM + VALUE;
            else
                 {COUT, SUM} = SUM - VALUE;
          end
  end
```

Parameters

Parameter	Value	Explanation
Function	Increment	Create an incrementer
	Decrement	Create a decrementer
	Inc_Dec	Create a generator with an Inc_Dec pin
Optimize	Area	Create an area efficient layout
	Speed	Create a fast layout
Pitch	Integer > 1	Spacing between input pins, pitch of 2 means one cell between input pins
Width	Integer > 1	Width of input and output vectors
IncDecValue	Integer > 1	Value by which the data has to be incremented/decremented
Load	Boolean	Create a pin to load the generator with a value
Enable	Boolean	Add an enable pin to the component

Pins

Type	Name	Option	Explanation
In	ENABLE	Yes	Enable Incrementer/Decrementer
In	LOAD	Yes	Load Input option
In	DATA[Width-1:0]	Yes	Parallel data input
In	INC_DEC	Yes	Increment/Decrement control. Increment when equal to 1 or Decrement otherwise
In	CLK	No	Clock
In	LOAD	Yes	Parallel load signal (active low)
In	R	No	Reset (active low)
Out	SUM[Width-1:0]	No	Incrementer/Decrementer output
Out	COUT	No	Carry out

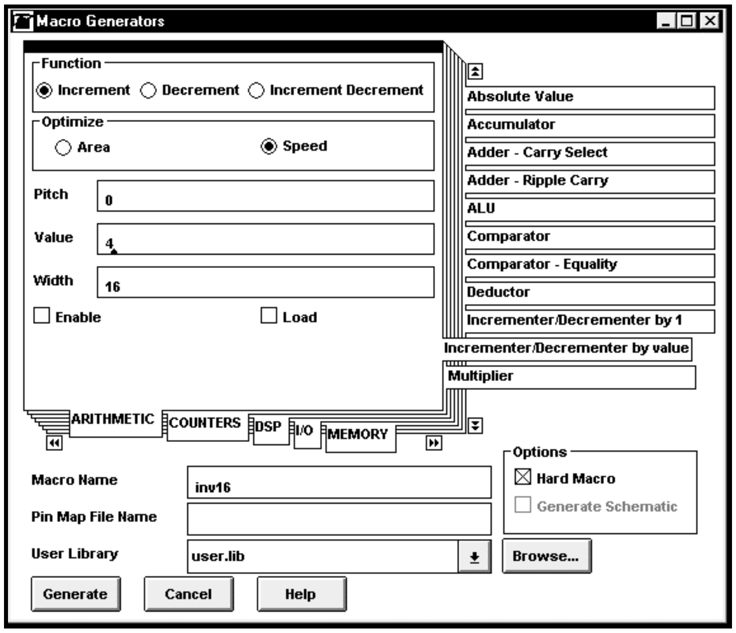
Truth Table

Input						Output	
R	ENABLE	CLK	LOAD	DATA[W-1:0]	INC_DEC	SUM[W-1:0]	COUT
0	X	X	X	X...X	X	0...0	0
1	0	X	1	X...X	X	SUMW...SUM0	0
1	1	R	0	DW...D0	X	DW...D0	0
1	1	X	1	X...X	X	Present State	SUMW*...*SUM1*SUM0
1	1	R	1	DW...D0	1	DW...D0+Value	SUMW*...*SUM1*SUM0
1	1	R	1	DW...D0	0	DW...D0- Value	SUMW*...*SUM1*SUM0

Note: The above truth table and the behavioral description are given for this generator when the function is set to Increment_Decrement.

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
inv16	19.38	51.6	112	7x16	314	0.41	2.80
inv8	30.77	32.5	56	7x8	154	0.29	2.75



Function: [✓ Increment] Decrement Increment Decrement

Optimize: Area [✓ Speed]

Pitch	0
Value	4
Width	16

☐ Enable ☐ Load

Macro Name	inv16
Pin Map File Name	
User Library	user.lib

Options

☒ Hard Macro

☐ Generate Schematic

Latch – D Type

The Latch generator can be used to generate a D-type transparent latch.

Parameters

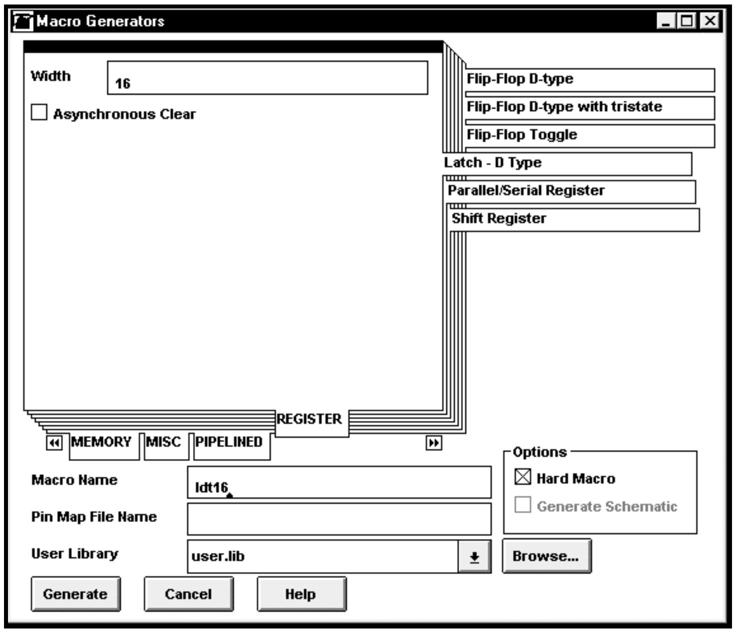
Parameter	Value	Explanation
Aclr	Boolean	Latch has an asynchronous clear
Width	Integer > 0	Width of input and output data

Pins

Type	Name	Option	Explanation
In	DATA[Width-1:0]	No	Data input to the D-type latches
In	GATE	No	Latch enable input (1 = flow-through, 0 = latch)
In	ACLR	Yes	Asynchronous clear input
Out	Q[Width-1:0]	No	Data output from D latches

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
ldt16	108.70	9.2	80	3x32	72	0.13	0.90
ldt8	108.70	9.2	40	3x16	36	0.06	0.90



Width: 16
☐ Asynchronous Clear
Macro Name: ldt16
Pin Map File Name:
User Library: user.lib
☒ Hard Macro

LIFO

The STACK/LIFO generator can be used to create a LIFO (Last In First Out) macro.

Write Operation

RW = 0. Data will be clocked into the registers on every rising CLK edge at which CKEN = 1.

EMPTY will negate with the first write; FULL will assert with the edge on which the last register is written.

Read Operation

Data will be available at the Q outputs as soon as a write is clocked. When RW = 1, data is clocked into the output registers with each rising CLK edge. Note therefore, that the first read clock after a write will clock the last-but-one byte written to the output.

Parameters

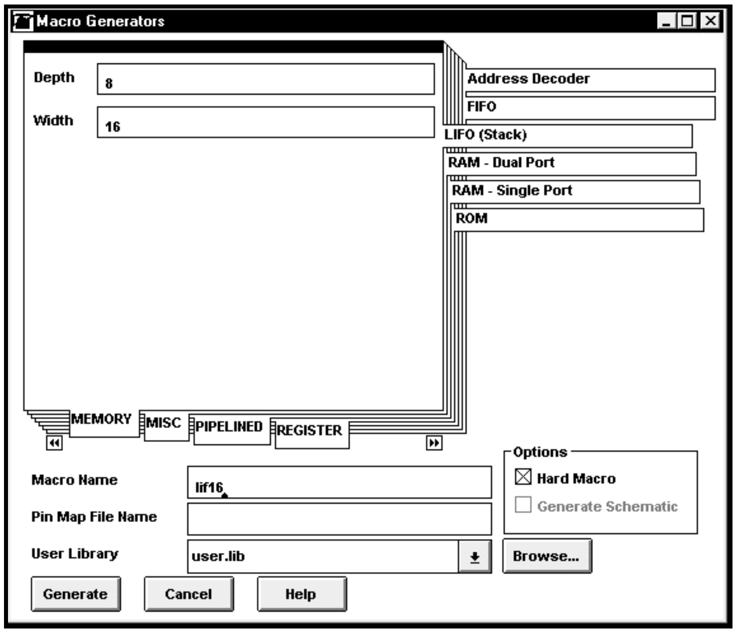
Parameter	Value	Explanation
Enable	Boolean	Inhibit write when full
Width	Integer 1-64	Width of parallel input and output data
Depth	Integer 2-64	Depth

Pins

Type	Name	Option	Explanation
In	D[Width-1:0]	No	Data input
In	R	No	Reset, active low
In	CLK	No	Clock
In	CKEN	No	Clock Enable pin (active high)
In	RW	No	Read/Write (high for read)
Out	Q[Width-1:0]	No	Data output
Out	FULL	No	LIFO full flag (active high)
Out	FULL_1	No	LIFO full - 1 flag (active high)
Out	EMPTY	No	LIFO empty flag (active low)

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
lif16	74.07	13.5	425	34x13	2452	2.61	5.77
lif8	74.07	13.5	225	18x13	1296	1.44	5.76


Macro Generators

Depth: 8

Width: 16

Macro Name: lif16

Pin Map File Name:

User Library: user.lib

Options

☒ Hard Macro

☐ Generate Schematic

Logic Gates

A variety of Logic Gates can be generated which can be optimized for either speed or area and controlled by a number of parameters.

Parameters

Parameter	Value	Explanation
CommonInput	Boolean	Select this option if a common input should be supplied to each of the Size gates
Gate	And	And
	Inv	Inverter
	Nand	Nand
	Nor	Nor
	Or	Or
	Xor	Exclusive Or
Optimize	Area	Create an area efficient layout
	Speed	Create a fast layout
Aspect	Float > 0	Aspect ratio 1 = square, 2 = 2 * wider than tall etc., applies only to area efficient layouts
Pitch	Integer > 0	Spacing between input pins. Pitch of 2 means 1 cell between inputs
Size	Integer > 0	Number of gates in component
Width	Integer > 0	Width of input vector

Pins

Type	Name	Option	Explanation
In	DATA[S-1:0]_[W-1:0]	No	Gate input pin where S goes from 0 to Size and W goes from 0 to Width
Out	RESULT[S-1:0]	No	Gate output

Note: S and W stand for Size and Width respectively.

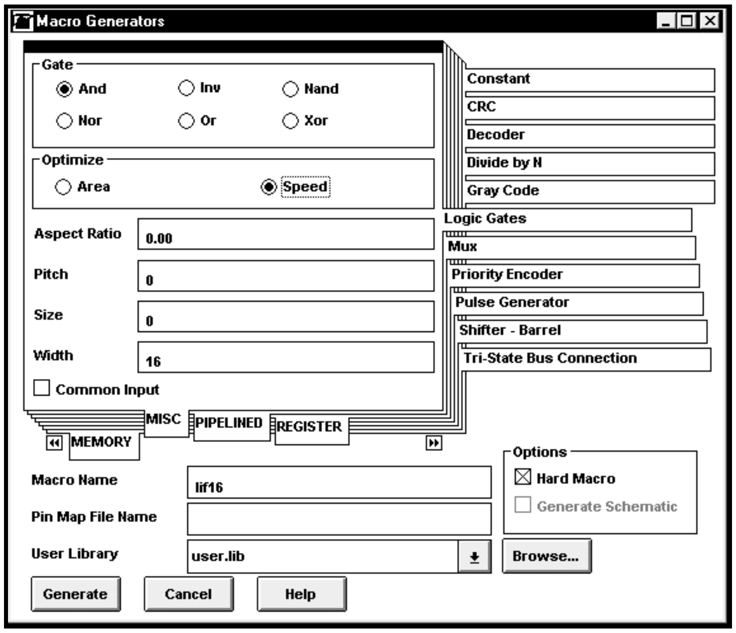
Truth Table

Input		Output
And	DATA[S-1:0]_[W-1:0]	DATA0 * DATA1 ... * DATAW-1
Inv	DATA[S-1:0]_[W-1:0]	~DATA0, ~DATA1, ... ~DATAS-1
Nand	DATA[S-1:0]_[W-1:0]	~(DATA0 * DATA1 ... * DATAW-1)
Nor	DATA[S--1:0]_[W-1:0]	~(DATA0 + DATA1 ... + DATAW-1)
Or	DATA[S-1:0]_[W-1:0]	DATA0 + DATA1 ... + DATAW-1
Xor	DATA[S-1:0]_[W-1:0]	DATA0 ^ DATA1 ... ^ DATAW-1

Note: S and W stand for Size and Width respectively.

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
log16	66.23	15.1	24	4x8	15	0.03	0.63
log8	108.70	9.2	10	3x4	7	0.01	0.70


Macro Generators

Gate: [✓ And] Inv Nand Nor Or Xor

Optimize: Area [✓ Speed]

Aspect Ratio: 0.00

Pitch: 0

Size: 0

Width: 16

☐ Common Input

Macro Name: lif16

Pin Map File Name:

User Library: user.lib

Options: ☒ Hard Macro ☐ Generate Schematic

Multiplier

The Multiplier macro can be used to generate the product of two varying width inputs. It has the option to produce an area efficient or fast (pipelined) multiplier. The function it produces is the following:

Product = DATAA * DATAB

Parameters

Parameter	Value	Explanation
Pipeline	Integer > 0	Create a pipeline every *n* stages
WidthA	Integer > 2	Width of input DataA
WidthB	Integer > 2	Width of input DataB
Truncate product to *n* least significant bits	Integer > 0	The number of output pins that have to be included in the layout starting from the LSB. This option saves the layout area by not including the unnecessary output pins
Representation	Signed	Product is signed (does not work for pipelining)
	Unsigned	Product is unsigned (default)

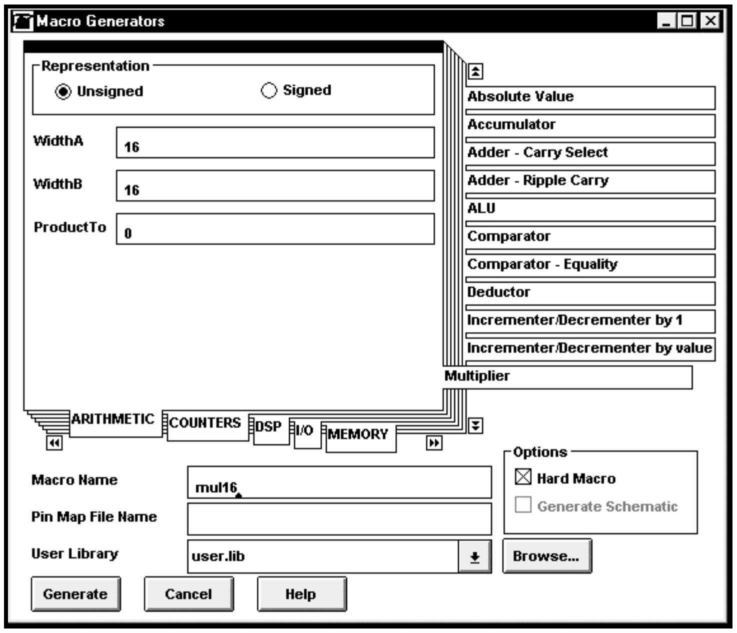
Pins

Type	Name	Option	Explanation
In	DATAA[WidthA-1:0]	No	Multiplicand
In	DATAB[WidthB-1:0]	No	Multiplier
Out	PRODUCT[Width-1:0]	No	DataA * DataB (Width is WidthA+WidthB)

Note: The pipeline function currently handles only unsigned numbers.

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
mul16	4.85	206	1470	48x31	2344	2.11	1.59
mul8	10.33	96.8	350	24x15	532	0.49	1.52


Macro Generators

Representation
[✓ Unsigned] Signed

WidthA: 16
WidthB: 16
ProductTo: 0

Options
☒ Hard Macro
☐ Generate Schematic

Macro Name: mul16
Pin Map File Name:
User Library: user.lib

Multiplier - Pipelined

The Multiplier Pipelined generator can be used to generate the product of two varying width inputs. It has the option to produce an area efficient or fast (pipelined) multiplier. The function it produces is the following:

Product = DataA * DataB

Parameters

Parameter	Value	Explanation
Pipeline	None	Create fully pipelined multiplier
WidthA	Integer > 2	Width of input DataA
WidthB	Integer > 2	Width of input DataB
ProductTo	Integer > 0	The number of the output pin that has to be included in the layout starting from the LSB. This option saves the layout area by not including the unnecessary output pins
Representation	Signed	Product is signed (does not work for pipe-lining)
	Unsigned	Product is unsigned (default)

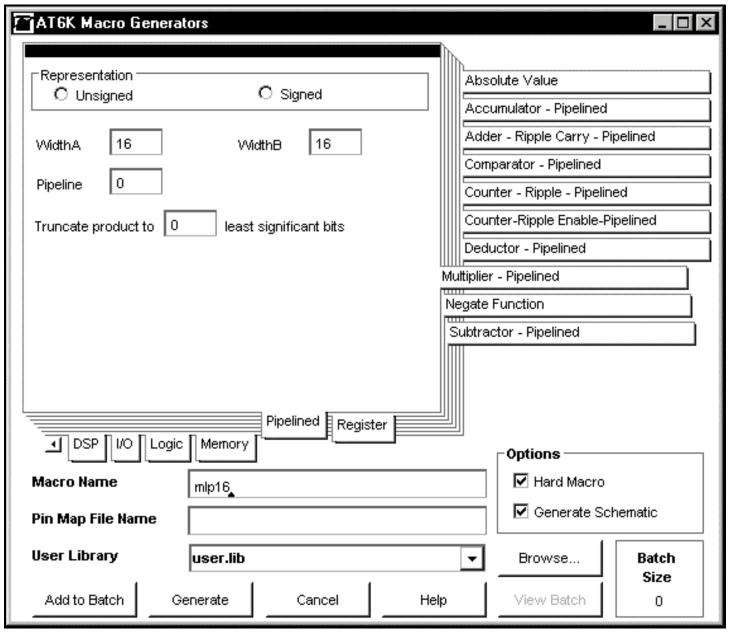
Pins

Type	Name	Option	Explanation
In	DATAA[WidthA-1:0]	No	Multiplicand
In	DATAB[WidthB-1:0]	No	Multiplier
Out	PRODUCT[Width-1:0]	No	DataA * DataB (Width is WidthA+WdithB)

Note: The pipeline function produces a fully pipelined multiplier. It currently handles only unsigned numbers.

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
mlp16	30.86	32.4	3477	79x55	13046	9.33	3.75
mlp8	43.86	22.8	825	38x27	3020	3.44	3.66



Representation
Unsigned
Signed

WidthA 16
WidthB 16
Pipeline 0
Truncate product to 0 least significant bits

Macro Name mlp16
Pin Map File Name
User Library user.lib

Options
☑ Hard Macro
☑ Generate Schematic

Batch Size 0

Multiplier – Serial Parallel

The Serial Parallel Multiplier (SPM) generator can be used to create a signed or unsigned serial by parallel multiplier with serial output. A width n unsigned multiplier is constructed in the FPGA by stringing n Carry-Save Adders. And a width n signed multiplier is constructed by stringing n-1 Carry-Save Adders and a Two's Complement together.

Let X and Y be n-bit and m-bit number respectively. The parallel input X is multiplied by each bit of serial input with the least significant bit coming first, and each of those partial products is added to the shifted accumulation of the previous product. The serial output is then taken from the output of the least significant bit adder. The output bit has the same weight as the previous serial input bit. The number of bits in the output is equal to the sum of the number of bits in each of the inputs.

Thus the product is:

```
{Y_(m-1)}*{X_(n-1)*2**(n-1)+X_(n-2)*2**(n-2)+...+X_(0)}
+ {Y_(m-2)}*{X_(n-1)*2**(n-1)+X_(n-2)*2**(n-2)+...+X_(0)}
+...
+ {Y_(0)}*{X_(n-1)*2**(n-1)+X_(n-2)*2**(n-2)+...+X_(0)}
```

Parameters

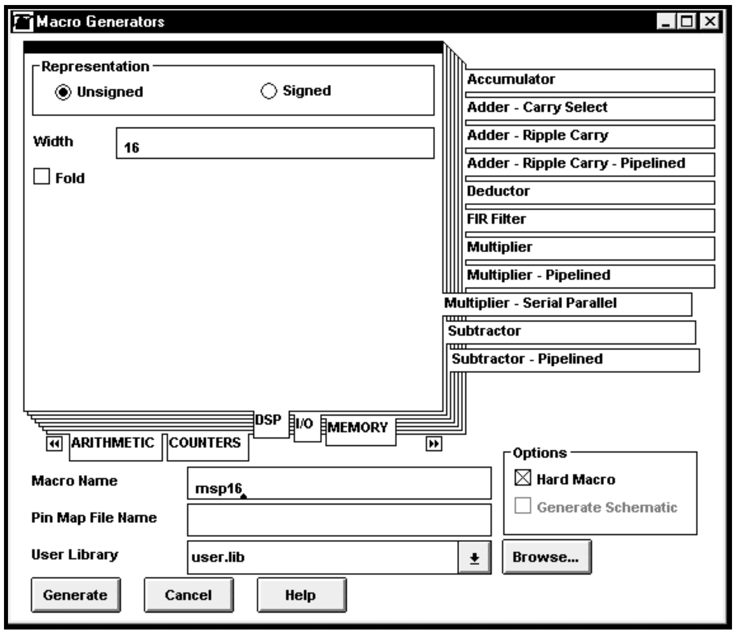
Parameter	Value	Explanation
Representation	Signed	Treat inputs as signed
	Unsigned	Treat inputs as unsigned
Width	Integer > 1	Width of parallel input data
Fold	Boolean	For area efficient option, fold layout in half

Pins

Type	Name	Option	Explanation
In	X[Width-1:0]	No	Multiplier (parallel input)
In	Y	No	Multiplicand (serial input)
Out	PROD	No	Product of X and Y (serial output)

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
msp16	79.37	12.6	113	32x4	457	1.26	4.04
msp8	79.37	12.6	57	16x4	229	0.69	4.01



Representation
[✓ Unsigned] Signed

Width 16

☐ Fold

Macro Name msp16

Pin Map File Name

User Library user.lib

☒ Hard Macro

☐ Generate Schematic

Mux

The Mux generator can be used to generate a full or partial decode of the specified number of bits of select or input.

Parameters

Parameter	Value	Explanation
Width	Integer >= 2	Width of input to the Mux
OutRegister	Boolean	Register the outputs of the Mux
Size	Integer >= 1	Number of bits in each element of input vector

Pins

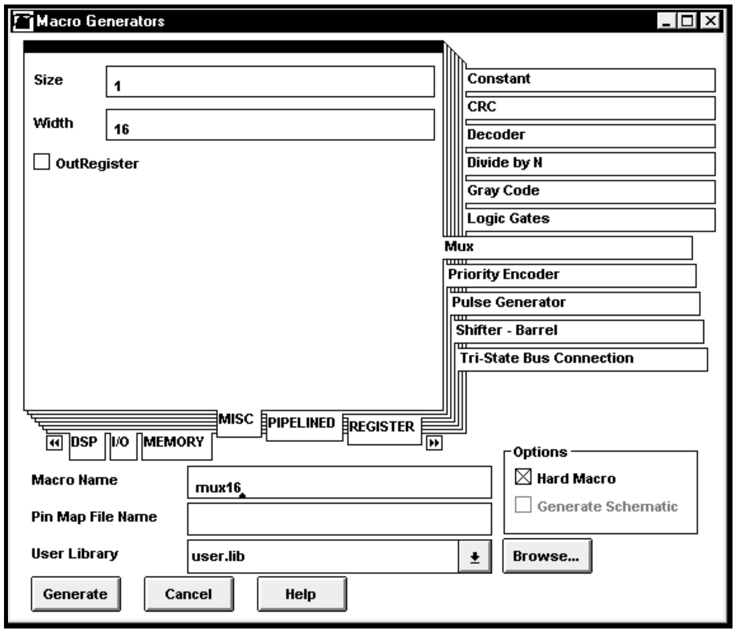
Type	Name	Option	Explanation
In	S[$\log_2$(Width) - 1:0]	No	Mux select pins, $\log_2$ Width bits wide
In	DATA[Width-1:0] X[Size-1:0]	No	Mux input pins
In	CLK	Yes	Clock for the registers
In	R	Yes	Reset for the registers (active low)
Out	RESULT[Size-1:0]	No	Mux output

Truth Table

Input		Output
S[$\log_2$(Width)-1:0]	D[Width-1:0]	Q
0	..XXX0	0
0	..XXX1	1
1	..XX0X	0
1	..XX1X	1
2	..X0XX	0
2	..X1XX	1
. . .	. . .	. . .

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
mux16	116.28	8.6	40	5x8	74	0.05	1.85
mux8	97.09	10.3	11	3x4	39	0.01	3.5



Size: 1

Width: 16

☐ OutRegister

Macro Name: mux16

Pin Map File Name:

User Library: user.lib

Options

☒ Hard Macro

☐ Generate Schematic

Negate Function

The Negate Function generator can be used to generate a two's complement function implemented in a ripple carry manner.

```
if DATA = 2 $^{Width-1}$, then
        OVERFLOW = 1, RESULT = -DATA
else
        RESULT = -DATA
```

DATA must always represent a positive number and the RESULT is always a two’s complement.

Parameters

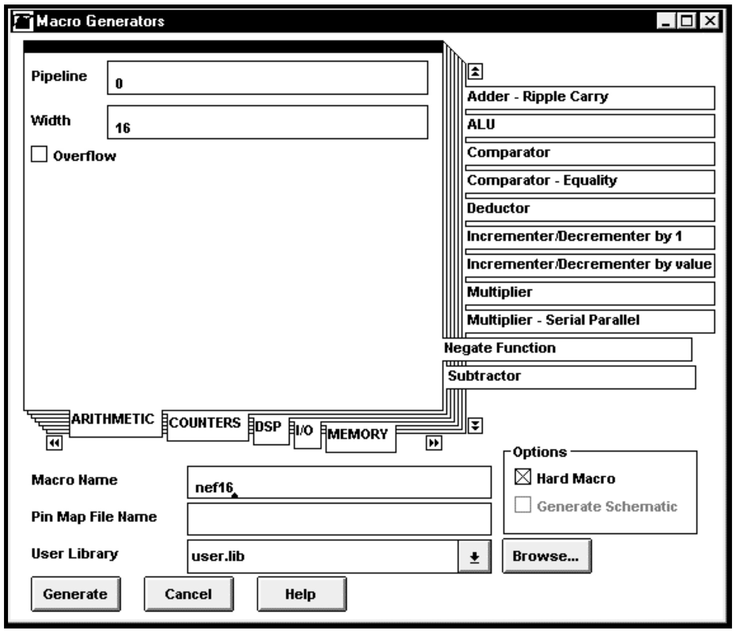
Parameter	Value	Explanation
Overflow	Boolean	Overflow output pin is present
Pipeline	Integer > 0	Pipeline every *n* stages
Width	Integer > 1	Width of input and output data

Pins

Type	Name	Option	Explanation
In	DATA[Width-1:0]	No	Data input
Out	RESULT	No	Data output = two's complement of Data
Out	OVERFLOW	Yes	1 if Data = $2^{Width-1}$, 0 otherwise

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
nef16	15.34	65.2	44	2x30	74	0.07	1.67
nef8	34.25	29.2	20	2x14	34	0.03	1.67



Pipeline: 0

Width: 16

☐ Overflow

Adder - Ripple Carry
ALU
Comparator
Comparator - Equality
Deductor
Incrementer/Decrementer by 1
Incrementer/Decrementer by value
Multiplier
Multiplier - Serial Parallel
Negate Function
Subtractor

Macro Name: nef16

Pin Map File Name:

User Library: user.lib

Options
☒ Hard Macro
☐ Generate Schematic

Parallel/Serial Register

The DFF generator can be used to create a Parallel/Serial Register.

Parameters

Parameter	Value	Explanation
Enable	Boolean	Add an Enable pin to work with both serial and parallel
InputOutput	ParallelInSerialOut	Input is parallel, output is serial
	SerialInParallelOut	Input is serial, output is parallel
Setlow	Boolean	Asynchronous active low set
Width	Integer > 0	Width of parallel input and output data

Pins

Type	Name	Option	Explanation
In	DATA[Width-1:0]	Yes	Data input for InputMode = Parallel
In	CLK	No	Clock pin
In	ENABLE	Yes	Data enable input
In	SHIFTIN	Yes	Data input for InputMode = Serial
In	SHIFTEN	Yes	1 = Function as a shift register (activates SHIFTIN), 0 = Shift
In	R	No	Reset pin (active low)
Out	Q[Width-1:0]	Yes	Data output for OutputMode = Parallel or Both
Out	SHIFTOUT	Yes	Data output for OutputMode = Serial or Both

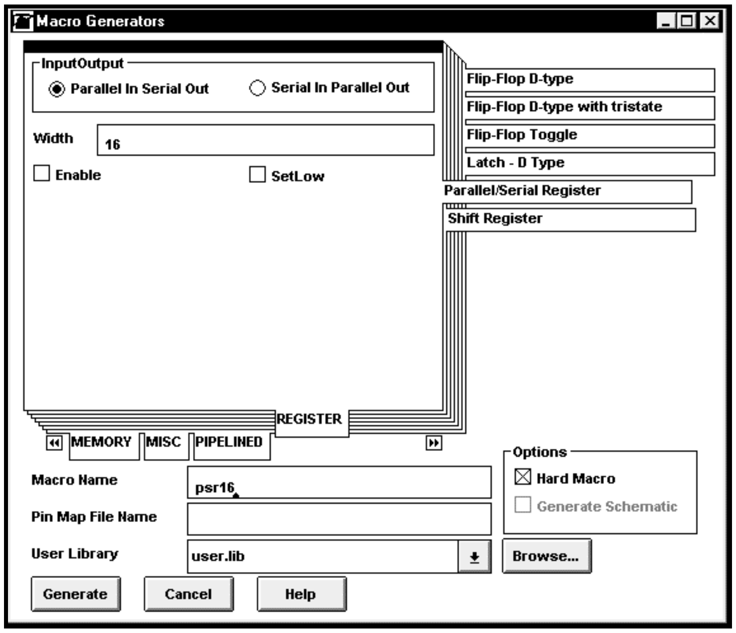
Truth Table

Input						Output	
R	DATA[W-1:0]	CLK	ENABLE	SHIFTIN	SHIFTEN	Q[W-1:0]	SHIFTOUT
0	X	X	X	X	X	0	Q(N-1)
1	X	0>1	X	X	X	No Change	Q(N-1)
1	0	0>1	1	X	0	0	Q(N-1)
1	1	0>1	1	X	0	1	Q(N-1)
1	X	0>1	1	SI	1	Q(I) = Q-(I-1) Q(0) = SI	Q(N-2)

Notes: Q- is the value of Q preceding the clock transition.

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
psr16	151.52	6.6	16	1x16	208	0.19	13.00
psr8	151.52	6.6	8	1x8	104	0.19	13.00



InputOutput: [✓ Parallel In Serial Out] Serial In Parallel Out

Width: 16

Enable ☐ SetLow ☐

Macro Name: psr16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic ☐

Priority Encoder

The Decoder generator can be used to create a Priority Encoder.

```
If DATA[0] == 1 then
        Q = Encode[Width]
else If DATA[1] == 1 then
        Q = Encode[Width-1]
        .
        .
        .
else If DATA[Width-1] == 1 then
        Q = 1
else
        Q = 0
```

Encode = the binary encoding for the width value given e.g. encode[7] = 111, encode[6]=110.

Parameters

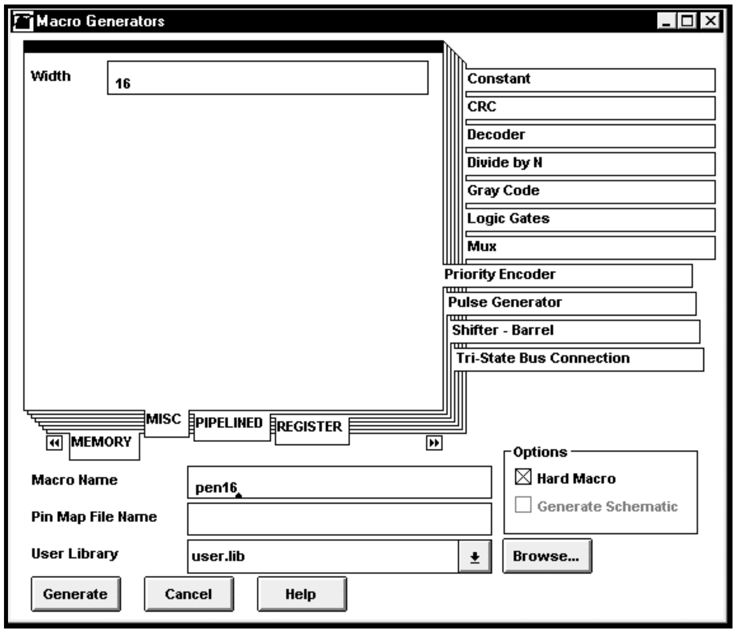
Parameter	Value	Explanation
Width	Integer > 2	Width of Inputs

Pins

Type	Name	Option	Explanation
In	DATA[Width-1:0]	No	Encoder input pins, active high
Out	Q[LOG$_2$ (Width): 0]	No	Encoder output pins

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
pen16	44.44	22.5	183	11x17	123	0.28	0.67
pen8	51.81	19.3	77	9x9	54	0.11	0.70



Width: 16

Macro Name: pen16

Pin Map File Name:

User Library: user.lib

☒ Hard Macro

☐ Generate Schematic

Pulse Generator

The Pulse Generator is created using an LFSR to provide a high speed counter. The pulse length varies from 2 to 2**(*n*-1) clocks, where *n* is the number of bits. The count sequence can be output in a format specified by a user-supplied data file (described below).

Parameters

Parameter	Value	Explanation
AccessCount	Boolean	LFSR counter Q outputs are made available
ListCount	Component	Generate component only - do not output count sequence
	Count	Output count sequence only - do not generate component
	Both	Output count sequence and generate component
Width	Integer 4-24	Width of input and output data

Pins

Type	Name	Option	Explanation
In	D[Width-1:0]	No	Load data input (decode value)
In	R	No	Reset, active low
In	CLK	No	Clock
In	START	No	Start pulse pin, active low
Out	PULSEOUT	No	Pulse output
Out	Q[Width-1:0]	Yes	Counter output (Q[Width-1] is inverted)

Truth Table

Input			Output
R	CLK	START	Q[W-1:0]
0	X	X	0
1	X	1	Present state
1	R	0	PULSEOUT starts at 0, then goes to 1 after 1 clock, then goes to 0 after pulse_length clocks

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
pls16	26.18	38.2	52	2x27	259	0.36	4.97
pls8	34.84	28.7	36	2x19	143	0.32	3.96

Count Sequence Output Specified ranges of the count sequence can be output to the file *component_name.lst*. These ranges are specified in file lfsr.dat. The format of the file is basically a header line with either hex or dec to indicate that the numbers in the file are in either Hexadecimal or Decimal format. Each subsequent line should be of the form Pulse_Length or Range: Pulse_Length_Min Pulse_Length_Max. For example:

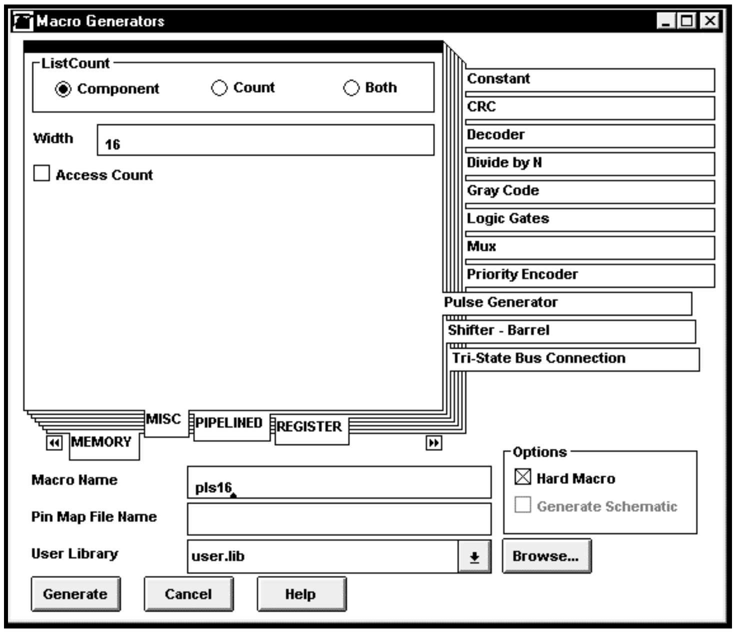
lfsr.dat

```
dec
4 20
88 127
300 844
1088
```

The output will have the format Length *pulse_length* = Count *count_number* : Decode *decode_value*, where *decode_value* is the value which should be asserted on the D[Width-1:0] inputs to produce a pulse of *pulse_length* clocks. *Count_number* is the number of clocks from the start of the LFSR count sequence starting at 0.

The generator options allow the count sequence to be output without generating a component. If this option is selected, the program flow will stop right after the macro generator is run with an Error message. This error can be ignored.

Pulse Generator Operation The pulse length is determined by the value on the D[Width-1:0] input pins (see above for determining this value). This is loaded into the counter whenever PULSEOUT is negated (low). The minimum pulse length is 3 clocks.



ListCount: [✓ Component] Count Both

Width: 16

☐ Access Count

Macro Name: pls16

Pin Map File Name:

User Library: user.lib

Options: ☒ Hard Macro ☐ Generate Schematic

RAM – Dual Port

The Dual Port RAM generator is used to create small random-access memories or register files. The read data and write data ports are brought out to separate sets of pins. Only one location can be read/written at any time. The RAM is generated using the D Flip-Flop in each core cell and is therefore synchronous with the CLK input. Data will be written into the location selected by ADDRESS at the rising-edge of the next CLK cycle. The RAM can also be asynchronously reset to zero using the R pin.

Parameters

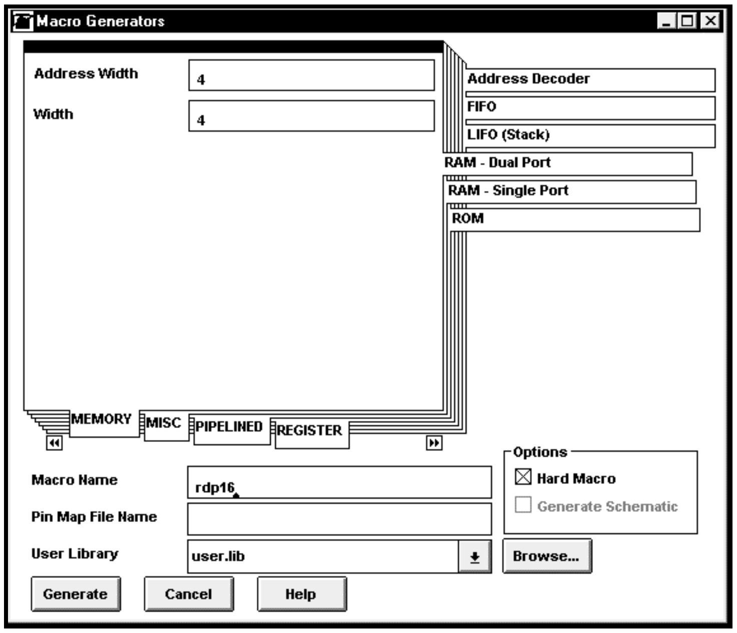
Parameter	Value	Explanation
AD width	Integer > 0	Width of Address Inputs - The number of RAM words created is determined by this parameter, i.e. ($2^{ADwidth}$)
Width	Integer > 0	Width of RAM data word created is determined by this parameter.

Pins

Type	Name	Option	Explanation
In	ADDRESS[AD-width-1:0]	No	Input address for RAM
In	DATA[Width-1:0]	No	Write Data Port
In	WE	No	Write Enable
Out	Q[Width-1:0]	No	Read Data Port
In	CLK	No	Clock input for RAM cells
In	R	No	Asynchronous Reset input

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
rdp16	51.28	19.5	248	16x19	976	1.59	3.94
rdp8	64.94	15.4	100	8x16	366	0.79	3.66



Address Width: 4

Width: 4

Macro Name: rdp16

Pin Map File Name:

User Library: user.lib

☒ Hard Macro

☐ Generate Schematic

RAM – Single Port

The Single Port RAM generator is used to create small random-access memories or register files. The read data and write data ports are combined into one set of bi-directional pins. Only one location can be read/written at any time. The RAM is generated using the D Flip-Flop in each core cell and is therefore synchronous with the CLK input. Data will be written into the location addressed
by ADDRESS at the rising-edge of the next CLK cycle. The RAM can also be asynchronously reset to zero using the R pin. When the WE (Write Enable) pin is asserted (WE='1'), the bi-directional I/O lines are tri-stated and write data can be applied to the DIO port. When WE is de-asserted (WE='0'), then read data currently selected by ADDRESS appears on the DIO port.

Parameters

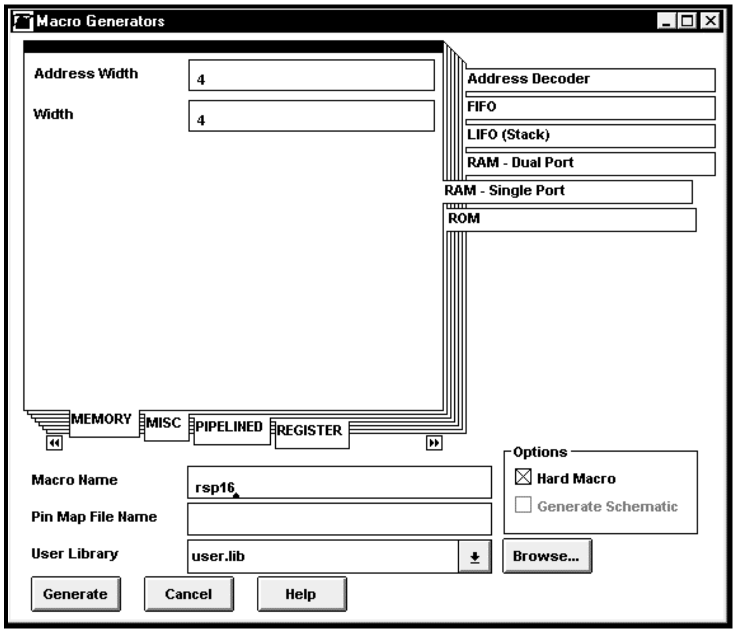
Parameter	Value	Explanation
AD width	Integer > 0	Width of Address Inputs - The number of RAM words created is determined by this parameter, i.e. ($2^{ADwidth}$)
Width	Integer > 0	Width of RAM data word created is determined by this parameter

Pins

Type	Name	Option	Explanation
In	ADDRESS[AD-Width-1:0]	No	Input address for RAM
In/Out	DIO[Width-1:0]	No	Data I/O Port
In	WE	No	Write Enable
In	CLK	No	Clock input for RAM cells
In	R	No	Asynchronous Reset input

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
rsp16	21.79	45.9	280	16x21	1032	1.65	3.69
rsp8	25.19	39.7	116	08x18	394	0.84	3.40



Address Width: 4

Width: 4

Macro Name: rsp16

Pin Map File Name:

User Library: user.lib

Options

☒ Hard Macro

☐ Generate Schematic

ROM

The ROM generator is used to create read-only data memories. The data is generated by reading a user-supplied data file (described below) and then using core cells as constant ("0" or "1") generators.

Parameters

Parameter	Value	Explanation
AD width	Integer > 0	Width of Address Inputs - the number of ROM words created is determined by this parameter, i.e. ($2^{ADwidth}$)
Optimize	Area	Create an area efficient layout
	Speed	Create a fast ROM layout
File	File name	Name of the file with data to be stored in ROM
Width	Integer > 0	Width of ROM data word created is determined by this parameter

Pins

Type	Name	Option	Explanation
In	ADDRESS[AD-Width-1:0]	No	Input address for ROM
In	MEMENAB (only available for speed option)	No	Memory Enable 1 = Output Data, 0 = Tri-State
Out	Q[Width-1:0]	No	ROM Data output

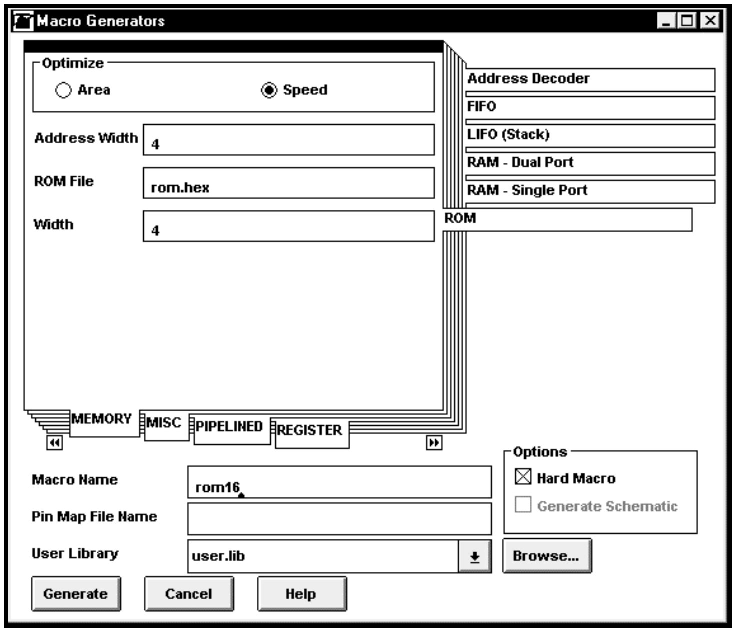
The ROM generator reads as input a file specified by the File option (default is rom.hex) which should be located in the project directory. The format of the file is basically a header line with either hex, dec, or bin to indicate that numbers in the file are in either Hexadecimal, Decimal, or Binary format respectively. Each subsequent line should be of the form Address Value. For example:

rom.hex

```
dec
0 0
1 2
2 4
3 8
4 7
5 5
6 3
.
.
.
```

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
rom16	51.02	19.6	184	16x12	144	0.30	0.78
rom8	57.80	17.3	76	8x10	54	0.13	0.71



Optimize: Area [✓ Speed]

Address Width: 4

ROM File: rom.hex

Width: 4

Macro Name: rom16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] Generate Schematic

Shift Register

The DFF generator can be used to create a shift register.

Parameters

Parameter	Value	Explanation
Enable	Boolean	Add an Enable pin to work with both serial and parallel
Setlow	Boolean	Asynchronous active low set.
Width	Integer > 0	Width of parallel input and output data

Pins

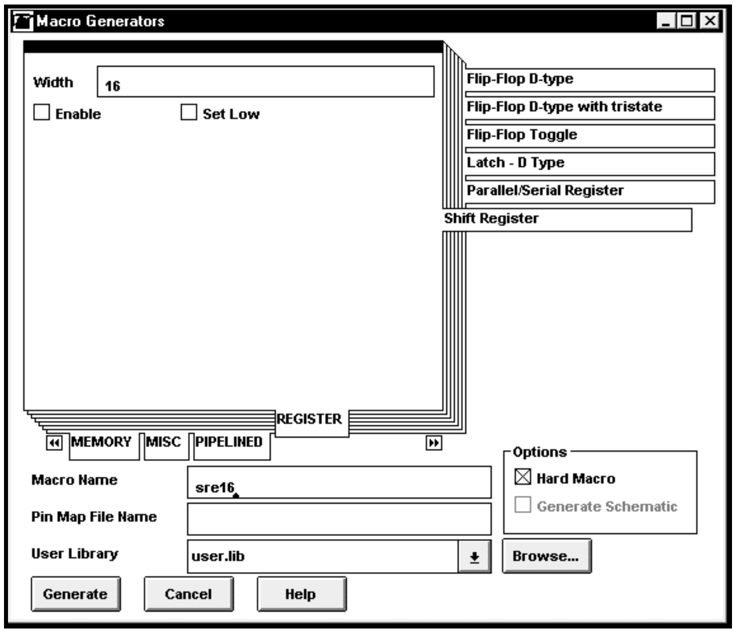
Type	Name	Option	Explanation
In	CLK	No	Clock Pin
In	ENABLE	Yes	Data Enable input
In	SHIFTIN	Yes	Data input for InputMode = Serial
In	R	No	Reset Pin (active low)
Out	SHIFTOUT	Yes	Data output

Truth Table

Input				Output
R	CLK	ENABLE	SHIFTIN	SHIFTOUT
0	X	X	X	Q(N-1)
1	0>1	X	X	Q(N-1)
1	0>1	1	X	Q(N-1)
1	0>1	1	X	Q(N-1)
1	0>1	1	SI	Q(N-2)

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
sre16	227.27	4.4	16	1x16	120	0.19	7.50
sre8	227.27	4.4	8	1x8	60	0.19	7.50



Width 16

☐ Enable ☐ Set Low

Macro Name sre16

Pin Map File Name

User Library user.lib

Options

☒ Hard Macro

☐ Generate Schematic

Shift - Barrels

The Barrel Shifter generator can be used to generate a combinatorial logic shift/rotate function with options for both an area and speed efficient layout.

Parameters

Parameter	Value	Explanation
Direction	Left	Shift/Rotate inputs to the left (lsb to msb)
	Right	Shift/Rotate inputs to the right (msb to lsb)
Function	Shift	Shift inputs in direction specified and pads with 0's
	Rotate	Rotate inputs in direction specified
Optimize	Area	Create an area efficient layout (using fully decoded distance)
	Speed	Create a speed efficient layout (using en-coded distance)
Width	Integer > 0	Width of input and output data

Pins

Type	Name	Option	Explanation
In	DATA[Width-1:0]	No	Data input to the shifter
In	DISTANCE[Width-2:0] Area DISTANCE[$\log_2$(Width)-1:0] Speed	No	Number of positions to shift/rotate (speed efficient is encoded, area efficient is fully decoded)
Out	RESULT[Width-1:0]	No	Shifted/Rotated data

Truth Table (Rotate Right)

Distance		Input	Output
AREA	SPEED	DATA	Q
0	000	D0,D1,...DW-1	D0,D1,...DW-1
1	001	D0,D1,...DW-1	DW-1,D1,...DW-2
11	010	D0,D1,...DW-1	DW-2,DW-1,D1...
111	011	.	.
.	.	.	.

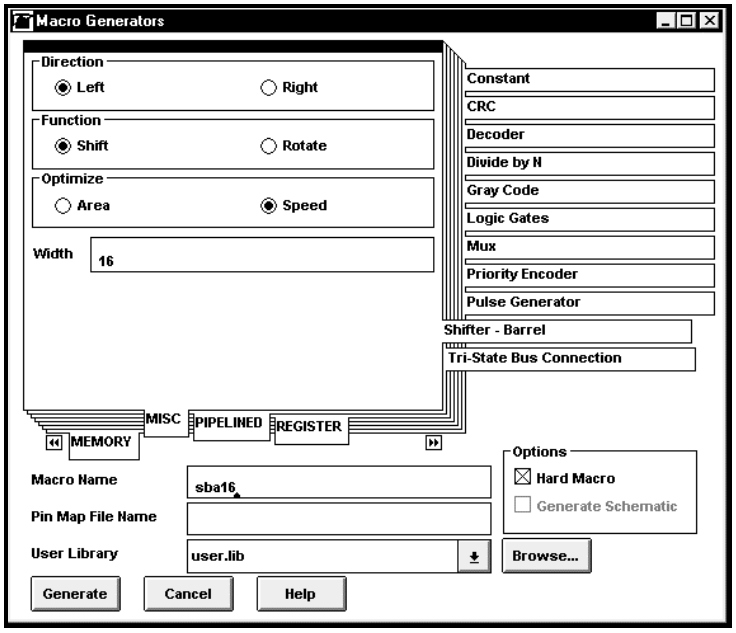
Truth Table (Shift Right)

Distance		Input	Output
AREA	SPEED	DATA	Q
0	000	D0,D1,...DW-1	D0,D1,...DW-1
1	001	D0,D1,...DW-1	0,D1,...DW-2
11	010	D0,D1,...DW-1	0,0,D1...DW-3
111	011	.	.
.	.	.	.

The speed macro Distance input is $\log_2$(Width) bits wide. The area macro Distance input is Width-1 bits wide and the data is shifted one bit position for every shift line that is asserted (i.e. for an 8 bit shifter the control would be 1 to shift 1 bit, 11 to shift 2 bits, 111 to shift 3 bits, etc.)

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
sba16	20.41	49	283	26x17	359	0.64	1.27
sba8	34.84	28.7	83	12x9	135	0.17	1.63



Direction: [✓ Left] Right
Function: [✓ Shift] Rotate
Optimize: Area [✓ Speed]
Width: 16
Macro Name: sba16
Pin Map File Name:
User Library: user.lib
Options: [✓ Hard Macro] Generate Schematic

Subtractor

The Subtractor generator options for both an area efficient and fast layout. Registered inputs and/or outputs can be specified as well.

Parameters

Parameter	Value	Explanation
Representati on	Unsigned	Treat inputs as unsigned
	Signed	Treat inputs as signed
Optimize	Area	Create an area efficient layout
	Speed	Create a fast layout
Pipeline	Integer > 0	Pipeline every *n* stages
Pitch	Integer > 1	Spacing between input pins, pitch of 2 means one cell between input pins
Width	Integer > 1	Width of input and output vectors
Carryin	Noregister	Include the carry in pin on the generator but do not register it
	Register	Register carry in of the subtractor
	Disabled	Do not include the carry in pin on the subtractor
Register	None	Do not register the inputs and outputs
	Input	Register inputs on the subtractor, exclude the carry in pin
	Output	Register outputs on the subtractor, exclude the carry out pin
	Both	Register both inputs and outputs including the carry in and carry out pins of the subtractor
Carryout	Noregister	Include the carry out pin on the subtractor but do not register it
	Register	Register carry out of the subtractor
	Disabled	Do not include the carry out pin on the subtractor
Overflow	Boolean	Create a pin to signal overflow
Fold	Boolean	For area efficient option, fold layout in half (cannot be used with Register output or pipelining)

Pins

Type	Name	Option	Explanation
In	CIN	Yes	Carry in
In	DATAA[Width-1:0]	No	A input
In	DATAB[Width-1:0]	No	B input
In	CLK	No	Clock (exists only for pipelined subtractor)
In	R	No	Reset (active low and exists only for pipe-lined subtractor)
Out	SUM[Width-1:0]	No	Subtractor output
Out	COUT	Yes	Carry out (cannot be used with overflow)
Out	OVERFLOW	Yes	Overflow (cannot be used with COUT)

Carry out = DATAA - DATAB - CIN > 2^n - 1 or
DATAA - DATAB - CIN < -2^n

Overflow for Unsigned
DATAA - DATAB - CIN > 2^n - 1 or
DATAA - DATAB - CIN < 0

Overflow for Signed
DATAA - DATAB - CIN > 2^n - 1 or
DATAA - DATAB - CIN < -2^n

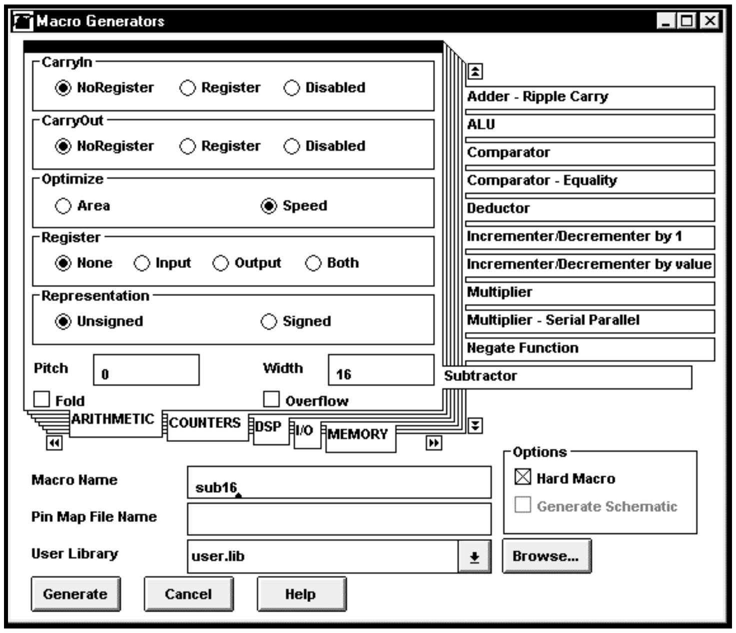
where n = Width

Truth Table

Input			Output	
CIN	DATAA[W-1:0]	DATAB[W-1:0]	SUM[W-1:0]	COUT
C	A	B	A - B - C	1 if A-B-C < 0, 0 otherwise

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
sub16	20.08	49.8	128	8x16	193	0.22	1.50
sub8	32.57	30.7	64	8x8	97	0.11	1.51



CarryIn: [✓ NoRegister] Register Disabled

CarryOut: [✓ NoRegister] Register Disabled

Optimize: Area [✓ Speed]

Register: [✓ None] Input Output Both

Representation: [✓ Unsigned] Signed

Pitch: 0

Width: 16

Fold

Overflow

Options: [✓ Hard Macro] Generate Schematic

Macro Name: sub16

Pin Map File Name:

User Library: user.lib

Subtractor - Pipelined

This Subtractor generator comes with a pipelining option. Registered inputs and/or outputs can be specified.

Parameters

Parameter	Value	Explanation
Representati on	Unsigned	Treat inputs as unsigned
	Signed	Treat inputs as signed
Optimize	Area	Create an area efficient layout
	Speed	Create a fast layout
Pipeline	Integer > 0	Pipeline every *n* stages
Pitch	Integer > 1	Spacing between input pins, pitch of 2 means one cell between input pins
Width	Integer > 1	Width of input and output vectors
Carryin	Noregister	Include the carry in pin on the generator but do not register it
	Register	Register carry in of the subtractor
	Disabled	Do not include the carry in pin on the subtractor
Register	None	Do not register the inputs and outputs
	Input	Register inputs on the subtractor, exclude the carry in pin
	Output	Register outputs on the subtractor, exclude the carry out pin
	Both	Register both inputs and outputs including the carry in and carry out pins of the subtractor
Carryout	Noregister	Include the carry out pin on the subtractor but do not register it
	Register	Register carry out of the subtractor
	Disabled	Do not include the carry out pin on the subtractor
Overflow	Boolean	Create a pin to signal overflow
Fold	Boolean	For area efficient option, fold layout in half (cannot be used with Register output or pipe-lining)

Pins

Type	Name	Option	Explanation
In	CIN	Yes	Carry in
In	DATAA[Width-1:0]	No	A input
In	DATAB[Width-1:0]	No	B input
In	CLK	No	Clock (exists only for pipelined subtractor)
In	R	No	Reset (active low and exists only for pipe-lined subtractor)
Out	SUM[Width-1:0]	No	Subtractor output
Out	COUT	Yes	Carry out (cannot be used with overflow)
Out	OVERFLOW	Yes	Overflow (cannot be used with COUT)

Carry out = DATAA - DATAB - CIN > 2^n - 1 or
DATAA - DATAB - CIN < -2^n

Overflow for Unsigned
DATAA - DATAB - CIN > 2^n - 1 or
DATAA - DATAB - CIN < 0

Overflow for Signed
DATAA - DATAB - CIN > 2^n - 1 or
DATAA - DATAB - CIN < -2^n

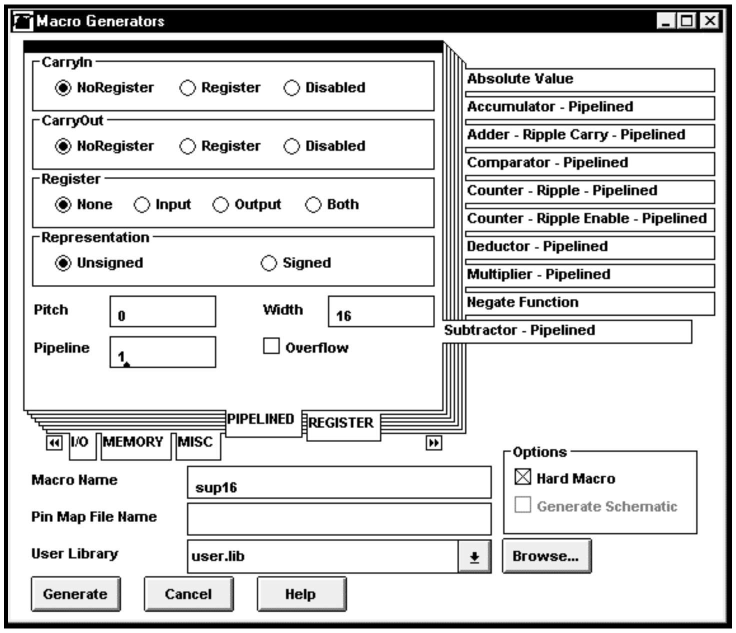
where n = Width

Truth Table

Input			Output	
CIN	DATAA[W-1:0]	DATAB[W-1:0]	SUM[W-1:0]	COUT
C	A	B	A - B - C	1 if A-B-C < 0, 0 otherwise

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
sup16	72.99	13.7	456	31x48	2965	2.36	6.5
sup8	72.99	13.7	132	15x24	759	1.10	5.75


Macro Generators

CarryIn: [✓ NoRegister] Register Disabled

CarryOut: [✓ NoRegister] Register Disabled

Register: [✓ None] Input Output Both

Representation: [✓ Unsigned] Signed

Pitch: 0

Width: 16

Pipeline: 1

Overflow: ☐

Macro Name: sup16

Pin Map File Name:

User Library: user.lib

Options: [✓ Hard Macro] ☐ Generate Schematic

Tri-State Bus Connection

The Tri-State generator can be used to create a connection to a Tri-state Bus.

Parameters

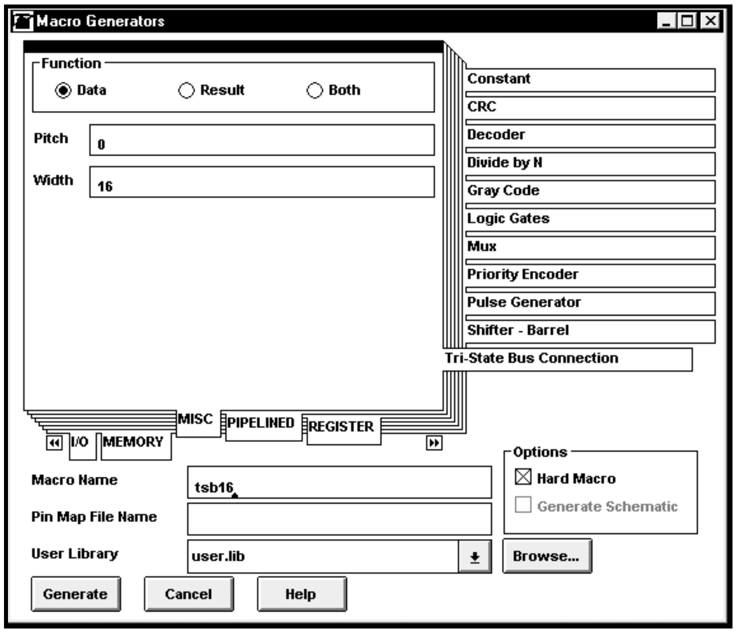
Parameter	Value	Explanation
Function	Data	Create Data pin which is the Input to the bus - TriData
	Result	Create Result pin which is the Output from the bus - TriData
	Both	Create both Data and Result pins
Pitch	Integer > 0	Spacing between input pins. Pitch of 2 means 1 cell between inputs
Width	Integer > 0	Width of input and output data

Pins

Type	Name	Option	Explanation
In	DATA[Width-1:0]	Yes	Data input to the tri-state bus
In	ENABLEDT	Yes	If high, enables Data onto the bus - TriData
In	TRIDATA[Width-1:0]	No	Bi-Directional Bus Signal
Out	RESULT[Width-1:0]	Yes	Output from the bus - TriData

Statistics

Name	Speed (MHz)	Delay (ns)	Cells	Size (x*y)	Gates (ttl equiv)	Power (mA/MHz)	Gates/Cell
tsb16	344.83	2.9	16	1x16	16	0.01	1.00
tsb8	344.83	2.9	8	1x8	8	0.01	1.00


Macro Generators

Function
(● Data) ○ Result ○ Both

Pitch: 0

Width: 16

Macro Name: tsb16

Pin Map File Name:

User Library: user.lib

Options
[✓] Hard Macro
[ ] Generate Schematic