DW_div_seq

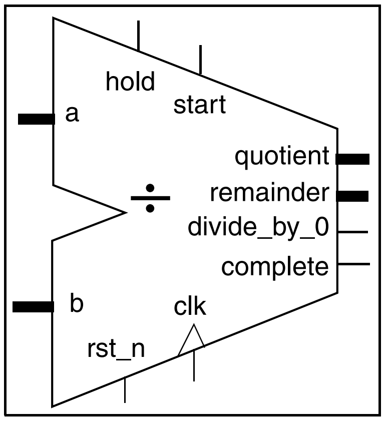
Sequential Divider

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- Parameterized word length
- Parameterized number of clock cycles
- Unsigned and signed (two's complement) data division
- Registered or un-registered inputs and outputs
- Provides minPower benefits with the DesignWare-LP license (Get the minPower version of this datasheet.)



Description

DW_div_seq is a sequential divider designed for low area, area-time trade-off, or high frequency (small cycle time) applications. DW_div_seq is an integer divider with both `quotient` and `remainder` outputs.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset, active low
hold	1 bit	Input	Hold current operation (=1)
start	1 bit	Input	Start operation (=1) A new operation is started by setting `start` =1 for one clock cycle.
a	*a_width* bit(s)	Input	Dividend
b	*b_width* bit(s)	Input	Divisor
complete	1 bit	Output	Operation completed (=1)
divide_by_0	1 bit	Output	Indicates if `b` equals 0
quotient	*a_width* bit(s)	Output	Quotient
remainder	*b_width* bit(s)	Output	Remainder

Table 1-2 Parameter Description

Parameter	Values	Description
a_width	≥ 3 Default 8	Word length of `a`
b_width	3 to *a_width* Default: 8	Word length of `b`
tc_mode	0 or 1 Default: 0	Two's complement control 0 = unsigned 1 = two's complement
num_cyc	3 to *a_width* Default: 3	User-defined number of clock cycles to produce a valid result. The real number of clock cycles depends on various parameters and is given in Table 1-6 on page 4 and the topic titled "Formula" on page 4.
rst_mode	0 or 1 Default: 0	Reset mode 0 = asynchronous reset 1 = synchronous reset
input_mode[a]	0 or 1 Default: 1	Registered inputs 0 = no 1 = yes
output_mode	0 or 1 Default: 1	Registered outputs 0 = no 1 = yes
early_start	0 or 1 Default: 0	Computation start 0 = start computation in the second cycle 1 = start computation in the first cycle See Table 1-6 on page 4 for the dependency of *early_start* on *input_mode*.

a. When configured with the parameter *input_mode* set to '0', the inputs `a` and `b` MUST be held constant from the time `start` is asserted until `complete` has gone high to signal completion of the calculation. Conversely, if a configuration with the parameter *input_mode* set to '1' is used, the `a` and `b` inputs will be captured when `start` is high and otherwise ignored.

Table 1-3 Synthesis Implementations

Implementation	Function	License Feature Required
cpa	radix-2 synthesis model	DesignWare
cpa2	radix-4 synthesis model	DesignWare

Table 1-4 Simulation Models

Model[a]	Function
DW03.DW_DIV_SEQ_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW_div_seq_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_div_seq.v	Verilog simulation model source code

a. Note that during the computation phase (after start and before complete is asserted), the simulation models output X values and therefore cannot be used as a compare for gate-level simulations.

Attention A divide by zero warning message is generated each time the `b` input changes to the value 0. This warning can be disabled for Verilog simulations by defining the Verilog macro, DW_SUPPRESS_WARN, either in the test bench or on the simulator command line (such as, `+define+DW_SUPPRESS_WARN+`).

Table 1-5 Operation Truth Table

start	hold	Operation
0	0	Idle or Running
0	1	Hold
1	X	Start

This component divides the dividend `a` by the divisor `b` to produce the `quotient` and `remainder` in a user-defined number of clock cycles (*num_cyc*). As long as `start`=1 the division operation is in the initialization state. Once `start`=0, the calculation begins followed by valid output flagged when `complete` =1 or an intervening setting of `start` to 1. The divide operation is stalled when `hold` =1. For a description of divide operation, refer to the datasheet DW_div. DW_div_seq calculates the division remainder (not modulus) on the output `remainder`, which corresponds to the "rem" operator in VHDL and the "%" operator in Verilog. This is equivalent to the `remainder` output of DW_div with *rem_mode*=*1*.

The parameter `tc_mode` determines whether the data of inputs (`a, b`) and outputs (`quotient`, `remainder`) are interpreted as unsigned (*tc_mode* = *0*) or two's complement (*tc_mode* =*1*) numbers.

The internal registers can either have an asynchronous (*rst_mode* = 0) or synchronous reset (*rst_mode* =*1*) that is connected to the reset signal `rst_n`.

After reset conditions are released (`rst_n` =1) there are no restrictions on when `start` can be set to 1 and then to 0. However, if `start` is set to 0 immediately after `rst_n` goes to 1, and `start`=0 continues through the first *num_cyc* clock cycles, then `complete` will go to 1. This first `complete` =1 when no start is initiated following reset may yield invalid results and should be disregarded.

The parameter *input_mode* determines whether the inputs are registered inside DW_div_seq (*input_mode*=*1*) or not (*input_mode*=*0*). If configured without input registers (*input_mode*=*0*), the logic that drives the inputs `a` and `b` must hold the input values constant for the entire time it takes to calculate the result (from the cycle before `start` drops until `complete` goes high). When configured with input registers (*input_mode*=1) the inputs `a` and `b` are captured when `start` is high and ignored until `start` goes high again.

Note When configured with no input registers, changes on inputs `a` and `b` while `complete` is low (calculation cycle) produces unpredictable output values. Simulation models will produce unknown output values (Xs) and post an error message indicating the instance that violated this rule and the simulation time when the violation was detected.

The parameter *output_mode* determines whether the outputs are registered.

When the parameter *early_start* is set to 1, computation starts using the value at input `b` on the first cycle (the last cycle before `start` goes to 0), and switching to the registered `b` input on the following cycles. This saves one extra cycle to store the data (*early_start*=0), but feeds the inputs directly into the components critical path.

Table 1-6 shows the *input_mode*, *output_mode*, and *early_start* parameter combinations and corresponding actual number of cycles required to perform an operation.

Table 1-6 Actual Cycles Based on *input_mode*, *output_mode*, and *early_start*

input_mode	*output_mode*	*early_start*	Actual Number of Cycles
0	0	0	*num_cyc*–2
0	0	1	Invalid parameter setting
0	1	0	*num_cyc*–1
0	1	1	Invalid parameter setting
1	0	0	*num_cyc*–1
1	0	1	*num_cyc*–2
1	1	0	*num_cyc*
1	1	1	*num_cyc*–1

Note that the *num_cyc* specification indicates the actual throughput of the device. That is, if a new input is driven before the *num_cyc* number of cycles are complete, the results are undetermined.

Formula

The following formula describes the number of dividend bits processed per cycle:

$$\text{bits processed per cycle} = \text{ceil}\,(a\_width / num\_cyc)$$

where:

a_width is the bit width of the dividend, and
num_cyc is the number of clock cycles required for division.

The actual number of clock cycles required by a computation are calculated using the following formula:

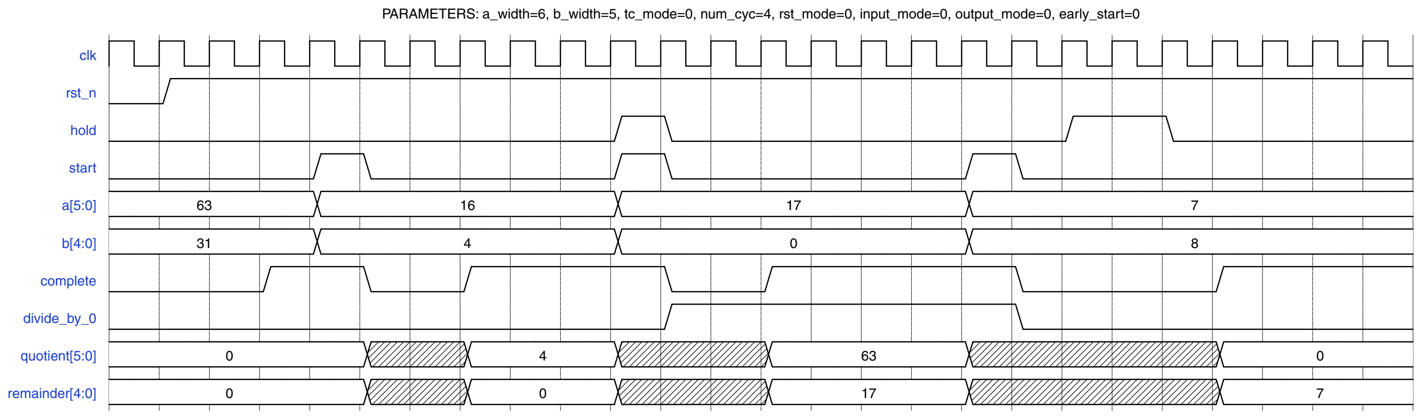
actual number of cycles = *num_cyc*-(1-*output_mode*)-(1-*input_mode*)-*early_start*

Timing Waveforms

The following timing waveforms show a 6-bit by 5-bit unsigned sequential divider for specific inputs of `hold` and `start` and their corresponding outputs. The parameter settings for each simulation are shown at the top of each figure. When `hold`=1 and `start`=0, the result is delayed by the same number of clock cycles for which `hold` is 1. For example, if `hold`=1 for two clock cycles, then the result is delayed by two clock cycles.

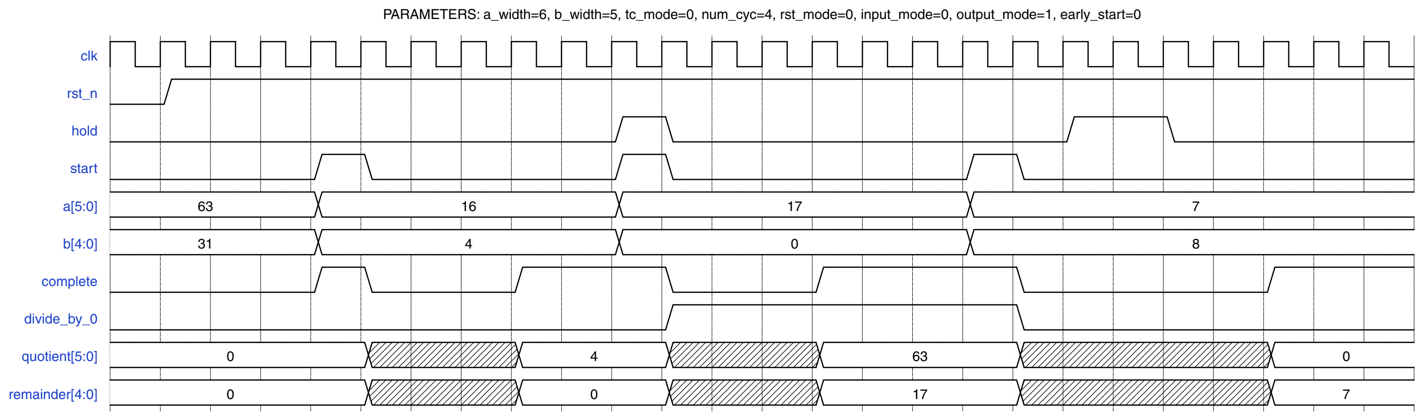
For the parameter settings shown in Figure 1-1, Table 1-6 on page 4 specifies that the result is produced 2 clock cycles after `start` is set to 0, as shown in Figure 1-1 for the operations where `hold` is kept low. For this set of parameters, the result must be read before a new input is applied, given that there is no input or output registers. Driving a `start` signal or changing the input values (dividend or divisor) before reading the output results in undetermined data.

Figure 1-1 Simulation Waveform 1


PARAMETERS: a_width=6, b_width=5, tc_mode=0, num_cyc=4, rst_mode=0, input_mode=0, output_mode=0, early_start=0
clk
rst_n
hold
start
a[5:0]
63
16
17
7
b[4:0]
31
4
0
8
complete
divide_by_0
quotient[5:0]
0
4
63
0
remainder[4:0]
0
0
17
7

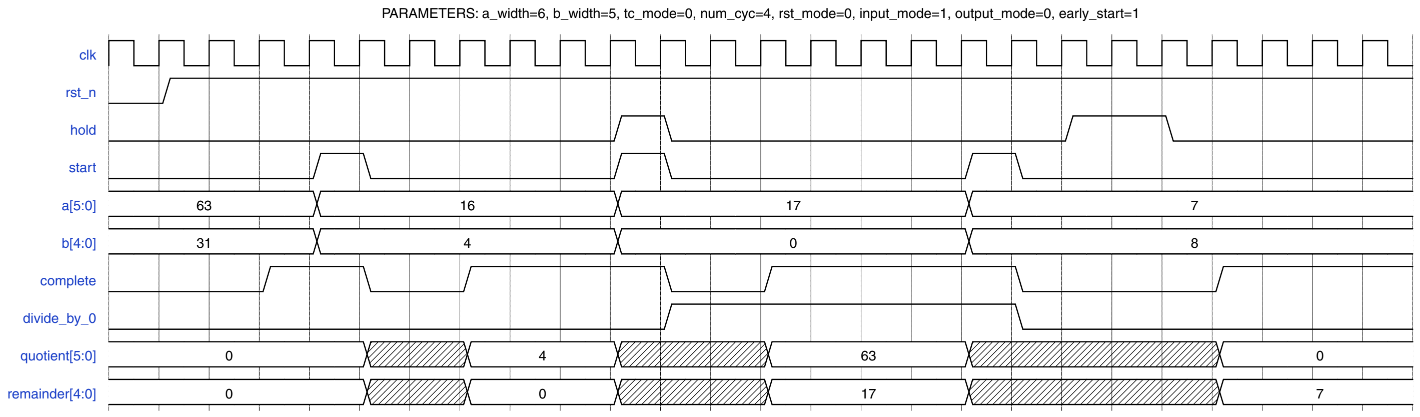
For the parameter settings shown in Figure 1-2, Table 1-6 on page 4 specifies that the result is produced 3 cycles after `start` goes to 0.

Figure 1-2 Simulation Waveform 2


PARAMETERS: a_width=6, b_width=5, tc_mode=0, num_cyc=4, rst_mode=0, input_mode=0, output_mode=1, early_start=0
clk
rst_n
hold
start
a[5:0]
63
16
17
7
b[4:0]
31
4
0
8
complete
divide_by_0
quotient[5:0]
0
4
63
0
remainder[4:0]
0
0
17
7

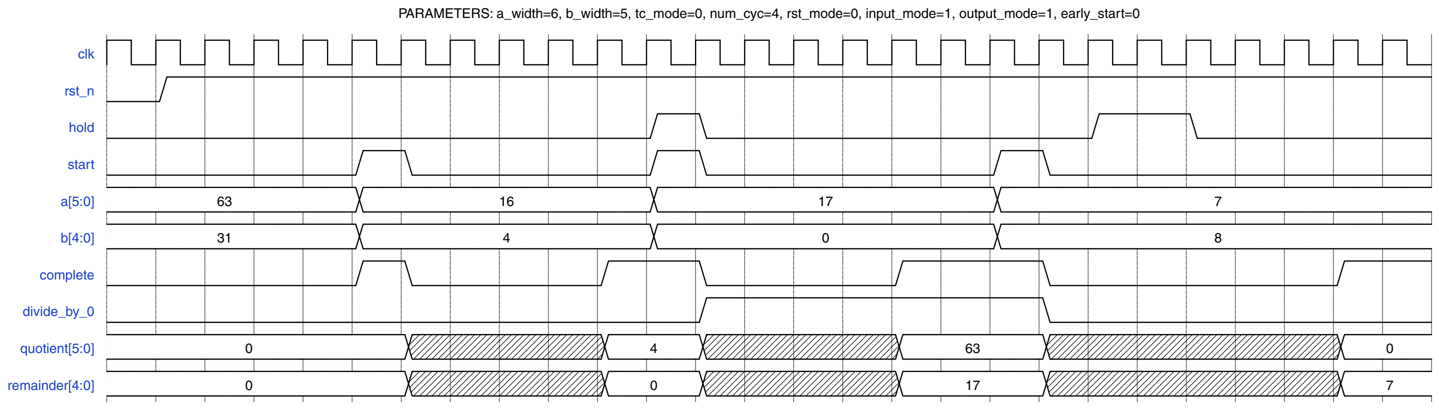
For the parameter settings shown in Figure 1-3, Table 1-6 on page 4 specifies that the result is produced 2 cycles after `start` goes to 0. With *input_mode*=1 (registered input), the input operands can be changed after the last sampled `start`=1 but no sooner, because *early_start*=1 allows for the processing to begin during the last cycle when `start`=1. This requires that input operands must be held stable during the last cycle when `start`=1.

Figure 1-3 Simulation Waveform 3



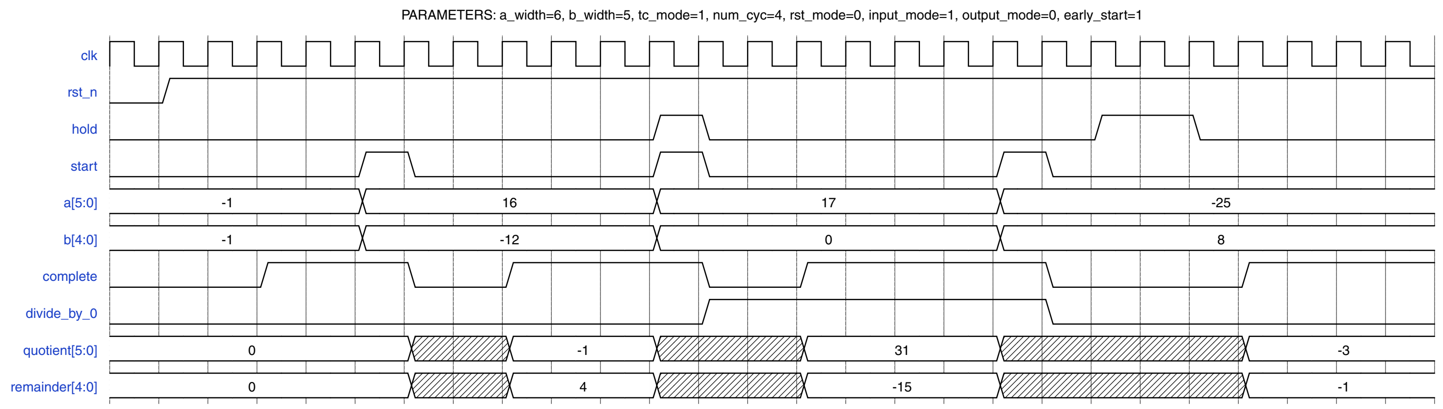
For the parameter settings shown in Figure 1-4, Table 1-6 on page 4 specifies that the result is produced 4 cycles after start goes to 0. Because *input_mode=1* (registered input), the input data can be removed after the first cycle of operation. When `hold`=1 and `start`=0, the result is delayed by the same number of clock cycles `hold`=1.

Figure 1-4 Simulation Waveform 4



For the parameter settings shown in Figure 1-5, Table 1-6 on page 4 specifies that the result is produced 2 clock cycles after start goes to 0. Because *input_mode=1* (registered input), the input data can be removed after the first cycle of operation. With `hold`=1 and `start`=0, the result is delayed by the same number of cycles for which `hold`=1. Note that the data is registered in the clock cycles when `start`=1.

Figure 1-5 Simulation Waveform 5



Related Topics

- Math – Sequential Overview
- DesignWare Building Block IP Documentation Overview

HDL Usage Through Component Instantiation - VHDL

```
library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_div_seq_inst is
  generic (inst_a_width     : POSITIVE := 8; inst_b_width    : POSITIVE := 8;
           inst_tc_mode     : INTEGER := 0;   inst_num_cyc    : INTEGER := 3;
           inst_rst_mode    : INTEGER := 0;   inst_input_mode : INTEGER := 1;
           inst_output_mode : INTEGER := 1;   inst_early_start : INTEGER := 0
           );
   port (inst_clk  : in std_logic;    inst_rst_n : in std_logic;
         inst_hold : in std_logic;    inst_start : in std_logic;
         inst_a    : in std_logic_vector(inst_a_width-1 downto 0);
         inst_b    : in std_logic_vector(inst_b_width-1 downto 0);
         complete_inst : out std_logic;
         divide_by_0_inst : out std_logic;
         quotient_inst    : out std_logic_vector(inst_a_width-1 downto 0);
         remainder_inst   : out std_logic_vector(inst_b_width-1 downto 0)  );
end DW_div_seq_inst;

architecture inst of DW_div_seq_inst is
begin
  -- Instance of DW_div_seq
  U1 : DW_div_seq
    generic map (a_width => inst_a_width,   b_width => inst_b_width,
                 tc_mode => inst_tc_mode,   num_cyc => inst_num_cyc,
                 rst_mode => inst_rst_mode,   input_mode => inst_input_mode,
                 output_mode => inst_output_mode,
                 early_start => inst_early_start   )
  port map (clk => inst_clk,   rst_n => inst_rst_n,
            hold => inst_hold,   start => inst_start,   a => inst_a,
            b => inst_b,   complete => complete_inst,
            divide_by_0 => divide_by_0_inst,   quotient => quotient_inst,
            remainder => remainder_inst   );
end inst;

-- pragma translate_off
configuration DW_div_seq_inst_cfg_inst of DW_div_seq_inst is
  for inst
  end for;
end DW_div_seq_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_div_seq_inst(inst_clk, inst_rst_n, inst_hold, inst_start, inst_a,
   inst_b, complete_inst, divide_by_0_inst, quotient_inst, remainder_inst);

  parameter inst_a_width = 8;
  parameter inst_b_width = 8;
  parameter inst_tc_mode = 0;
  parameter inst_num_cyc = 3;
  parameter inst_rst_mode = 0;
  parameter inst_input_mode = 1;
  parameter inst_output_mode = 1;
  parameter inst_early_start = 0;
  // Please add +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator
  // command line (for simulation).
  input inst_clk;
  input inst_rst_n;
  input inst_hold;
  input inst_start;
  input [inst_a_width-1 : 0] inst_a;
  input [inst_b_width-1 : 0] inst_b;
  output complete_inst;
  output divide_by_0_inst;
  output [inst_a_width-1 : 0] quotient_inst;
  output [inst_b_width-1 : 0] remainder_inst;
  // Instance of DW_div_seq
  DW_div_seq #(inst_a_width, inst_b_width, inst_tc_mode, inst_num_cyc,
               inst_rst_mode, inst_input_mode, inst_output_mode,
               inst_early_start)
    U1 (.clk(inst_clk),   .rst_n(inst_rst_n),   .hold(inst_hold),
        .start(inst_start),   .a(inst_a),   .b(inst_b),
        .complete(complete_inst),   .divide_by_0(divide_by_0_inst),
        .quotient(quotient_inst),   .remainder(remainder_inst) );
endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
August 2017	M-2016.12-SP5-1	■ For STARs 9001112685 and 9001226703: - Adjusted the description of computation when *early_start* = 1 on page 4 - Updated the following timing diagrams and the text that describes them: Figure 1-1 on page 5 Figure 1-2 on page 5 Figure 1-3 on page 6 Figure 1-4 on page 6 Figure 1-5 on page 7 ■ Added this Revision History table and the document links on this page

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043

www.synopsys.com