

DLD LAB – Experiment #4 - Accelerator and Wrappers

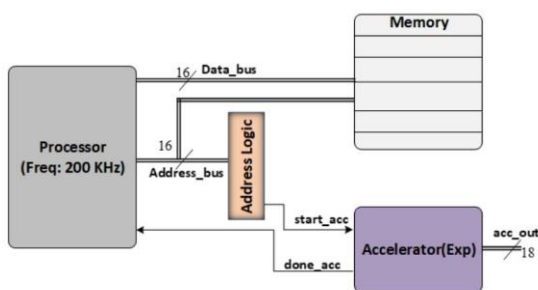
Ali Ghahari 810100201

Mohammad Sadeghi 810100175

Introduction

System on Chip is an integrated circuit that integrates multiple components. The main core of a SoC is a processor that handles different computational tasks. In addition to the processor, the system includes memory, Input/output ports, and accelerators. Accelerators are dedicated computation units that usually execute one specific task. This single task needs a smaller and less complicated datapath which leads to a high frequency. However, CPUs in which millions of operations must be executed within a fixed time interval have low frequency of operation. To increase the speed of a SOC, hardware accelerators are usually embedded in the system. The processor will dispute some of its tasks to the hardware accelerator. During this time, the accelerator performs several of the same or different operations and stores the result values in a memory. The CPU will access these results when it finishes its tasks. The focus of this experiment is on accelerators and how to integrate them into a SOC

Figure 1: Block diagram of a typical integrated circuit



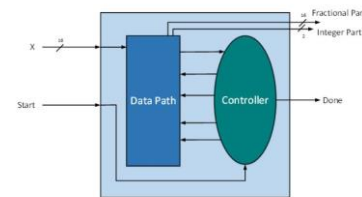
When the CPU needs to compute an exponential value, because of the higher estimation speed of the accelerator it asks the exponential hardware accelerator to complete this task. In this way, the CPU can complete other software tasks in parallel. Before starting the computation, the CPU should send a data from memory to the accelerator. This data will be stored in a buffer

inside the accelerator. When transferring is finished, the CPU initiates the accelerator for an N-round exponential estimation. CPU uses its address bus for initiating a component. By decoding the address bus through an address logic as shown in figure 1, the accelerator will have its *start* signal issued when needed.

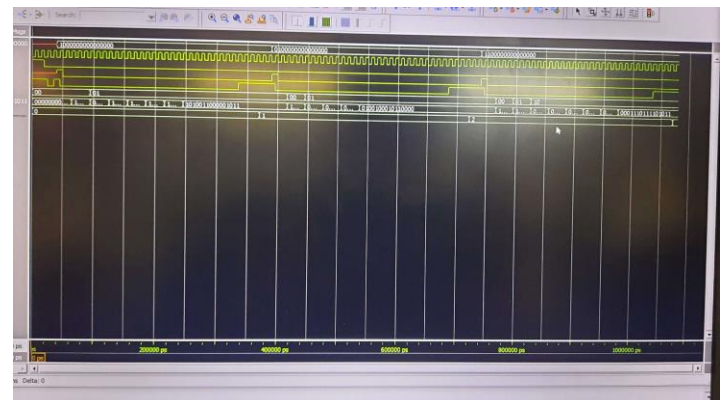
Exponential Engine

The accelerator that we are going to use is an exponential circuit. As Figure 2 shows, this module receives a 16-bit input *x* and generates a 16-bit output *Fractionalpart* and 2-bit *Integerpart*. Remember that the *x* value fits between zero and one. The accelerator starts working with a complete pulse on signal *start* and when the computation is completed signal *done* will be sent to the processor to aware it.

Figure 2: Block diagram of exponential accelerator

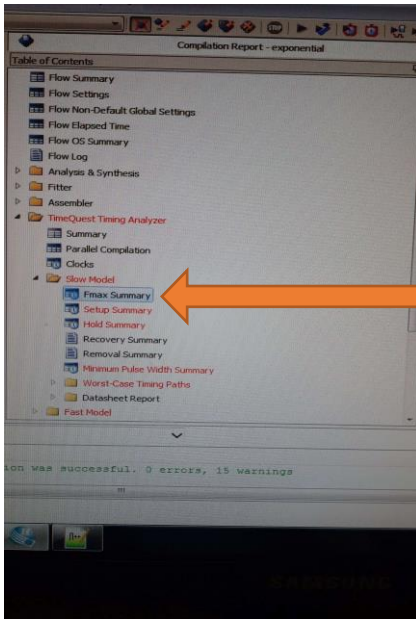


For clock generation for this module we need to be aware of the maximum frequency of this accelerator. First we running Modelsim simulation of module, write a testbench for this design with three different values for input *x*.



As you can see we have three value for x and in three times done signal become 1.

In next step we Synthesize this design in Quartus II Software. After synthesizing the design, we can find out the maximum frequency of this accelerator by referring to the Timing Analyzer reports in the Quartus synthesis tool:



As in this part we can find maximum frequency of accelerator:

	Fmax	Restricted Fmax	Clock Name	Note
1	109.39 MHz	109.39 MHz	clk	

Exponential Accelerator Wrapper

One of the applications that makes use of such multi-value exponential calculation is an activation function in Deep Neural Networks (DNN).

$$f(x_i) = e^{x_i} \{i = 1, 2, \dots, N\}$$

Multiple of these exponential values can be calculated with one accelerator instead of using one accelerator unit for each X_i . To reduce the required hardware resources for the softmax exponential function, a mathematical transformation would be useful. Each input value is split into an integer number Z_i and a fractional number V_i as below:

$$x_i = z_i + v_i$$

$$e^{x_i} = e^{z_i} * e^{v_i}$$

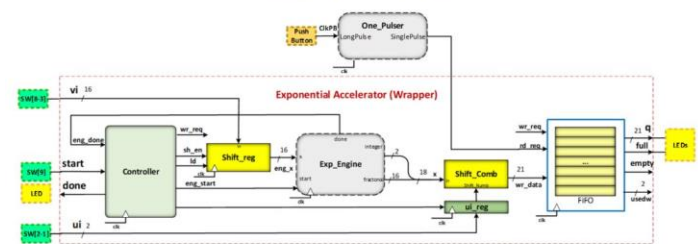
The second segment of this equation can be easily implemented with the exponential engine. To reduce the complexity of the hardware implementation of this equation, we use the base number 2 to take place of the base number e and the exponential function will be changed to:

$$e^{x_i} = 2^{u_i} * e^{v_i}$$

$$e^{x_i} = e^{v_i} \ll u_i$$

As can be seen, this equation can be implemented with a shifter that shifts e^{v_i} for u_i times. We are going to apply this transformation to the exponential in a wrapper around the exponential engine. Some other tasks are also included in the wrapper that are explained below:

Figure 4: Wrapper for exponential accelerator



- ➔ Referring to Figure 4, the wrapper receives single input in the form of a fractional value v_i , and an integer value u_i and a *start* signal from the processor.

- Considering the input values are within u_i and u_i+1 we can calculate n number of exponential in this range as follows:

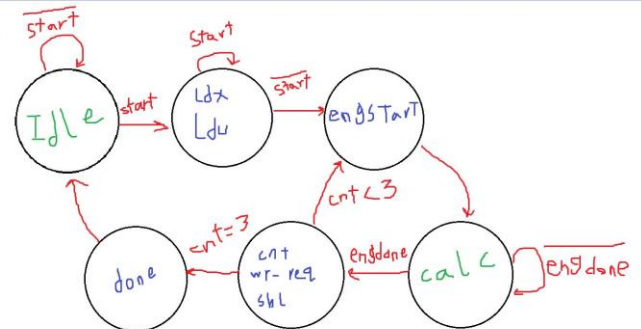
$$e^{x_i} = \begin{cases} e^{2^i v_i} \ll u_i & \text{if } v_i < 1/2^{n-1} \\ e^{2^{n-1} v_i} \ll u_i & \text{if } u_i < x_i < u_i + 1 \end{cases}$$

In this experiment $n=4$. Based on these assumptions, 4 exponential values can be calculated. Four different values can be generated with the shift register unit by first registering the value of v_i and then shifting its value one bit to the left for each exponential calculation.

- The controller is responsible for generating the *load* and *shift enable* signals for the shift register, the *start* signal for the exponential engine, and the *load* signal for the u_i -register. The exponential engine should start each calculation when the previous one is completely done. For this purpose *engdone* is fed to the controller and when done is asserted the controller generates a complete pulse on *engstart*. At the same time, the correct value of x should appear on the corresponding input of the exponential engine. For each exponential value estimation, the controller issues the *wrreq* signal for writing data to the FIFO. When all calculations are finished the controller sends a *done* signal on the wrapper output.
- For shifting the output of the exponential engine, a combinational shifter is required. The input of this shifter is the u_i value that is provided outside the wrapper. To store the value of u_i , a register called u_i -reg is used.
- When an exponential value is calculated then it should be stored in a FIFO so that when the CPU finishes its work it can retrieve all the results. As shown in the Figure 4, the FIFO has a *writereq* input and a *write data* for writing into the buffer

and a *readreq* for reading the outputs. Since we are limited to four calculations in this experiment the FIFO size would be four as well.

Here is state diagram for module controller and Huffman code for its verilog:



```
always @(*) begin
    ns = 3'b0;
    case (ps)
        IDLE: ns = start ? WAIT : IDLE;
        WAIT: ns = start ? WAIT : START;
        START: ns = CALC;
        CALC: ns = eng_done ? WRITE : CALC;
        WRITE: ns = count < 2'b11 ? START : DONE;
        DONE: ns = IDLE;
        default: ns = IDLE;
    endcase
end

always @(posedge clk, posedge rst) begin
    if(rst) count = 2'b0;
    else if(cnt) count = count + 1;
end

assign ldu = ps == WAIT;
assign ldx = ps == WAIT;
assign wr_req = ps == WRITE;
assign shl = ps == WRITE;
assign eng_start = ps == START;
assign done = ps == DONE;
assign cnt = ps == WRITE;
endmodule
```

In the next page you can see Verilog code for wrapper shown in figure 4 includes 5 component:

- 1-controller
- 2-combinational shift
- 3-Ui register
- 4-exponential unit
- 5-shift register


```

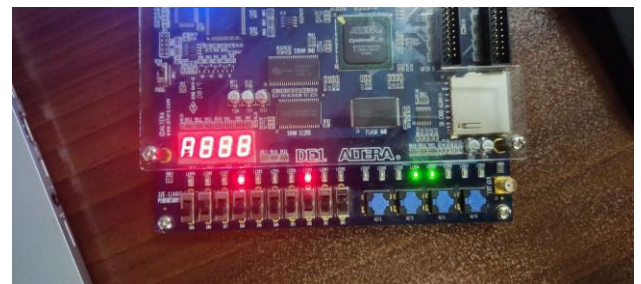
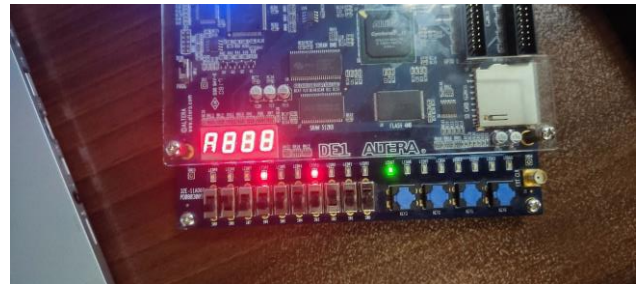
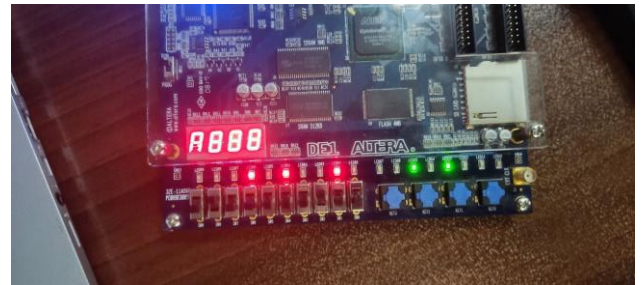
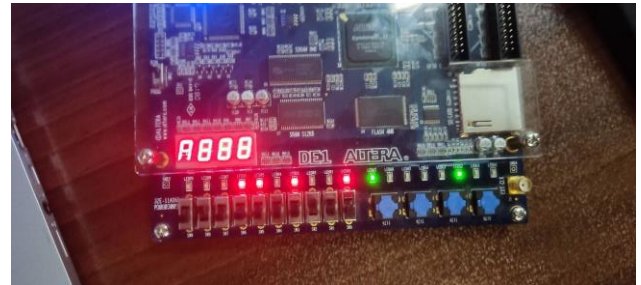
module Wrapper(clk, rst, start, v, u, done, wr_req, wr_data);
input clk, rst, start;
input[4:0] v;
input[1:0] u;
output done, wr_req;
output[20:0] wr_data;

wire ldx, ldu, shl, eng_start, eng_done;
wire[15:0] shreg_out, frac;
wire[1:0] int, ureg_out;

Controller controller(.clk(clk), .rst(rst), .wr_req(wr_req), .ldx(ldx), .ldu(ldu), .shl(shl), .eng_start(eng_start), .done(done), .
Shifting sh_reg(.clk(clk), .rst(rst), .ld(ldx), .sh(shl), .data(3'b0, v, 8'b0), .out(shreg_out));
Reg ureg(.clk(clk), .rst(rst), .ld(ldu), .data(u), .out(ureg_out));
exponential exp(.clk(clk), .rst(rst), .start(eng_start), .x(shreg_out), .done(eng_done), .intpart(int), .fracpart(frac));
ComShift comb_shift(.frac(frac), .int(int), .sh(ureg_out), .out(wr_data));
endmodule

```

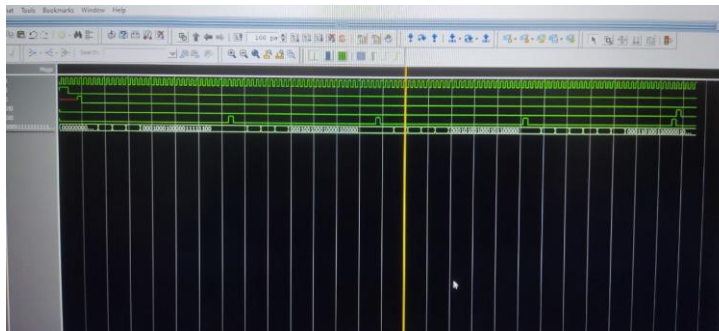
And you can see our output on board LED is same as wave we got in Modelsim:



And LEDs for stack *full* and *done* signal be 1:



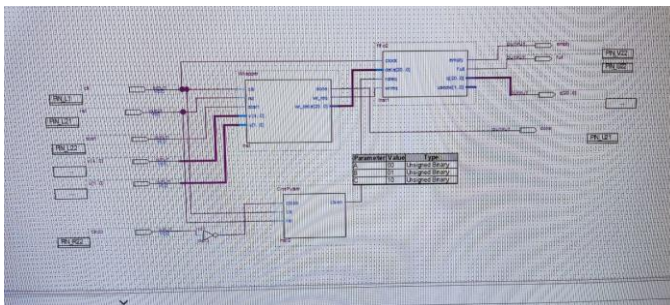
And this is our testbench result:



FPGA design

At last part we simulate this module with same input as Modelsim and see result with LED on board

First we build symbol of each component of wrapper and connect internal bus together:



And in next part we assign switch pins to our inputs and LED pins to done and FIFO full signal and other remained LEDs to MSB of our output:

Change View								
<div><div>Pin</div><div>IO Bank</div><div>WDF Group</div><div>I/O Standard</div><div>Reserved</div><div>Current Strength</div><div>Differential Pair</div></div>								
Node Name	Direction	Location	IO Bank	WDF Group	I/O Standard	Reserved	Current Strength	Differential Pair
u000	Output	PIN_301	0	RL_00	3.3V1V default	2mA (default)		
u001	Output	PIN_302	0	RL_00	3.3V1V default	2mA (default)		
u002	Output	PIN_303	0	RL_00	3.3V1V default	2mA (default)		
u003	Output	PIN_304	0	RL_00	3.3V1V default	2mA (default)		
u004	Output	PIN_305	0	RL_00	3.3V1V default	2mA (default)		
u005	Output	PIN_306	0	RL_00	3.3V1V default	2mA (default)		
u006	Output	PIN_307	0	RL_00	3.3V1V default	2mA (default)		
u007	Output	PIN_308	0	RL_00	3.3V1V default	2mA (default)		
u008	Output	PIN_309	0	RL_00	3.3V1V default	2mA (default)		
u009	Input	PIN_310	0	RL_00	3.3V1V default	2mA (default)		
u010	Input	PIN_311	0	RL_00	3.3V1V default	2mA (default)		
u011	Input	PIN_312	0	RL_00	3.3V1V default	2mA (default)		
u012	Input	PIN_313	0	RL_00	3.3V1V default	2mA (default)		
u013	Input	PIN_314	0	RL_00	3.3V1V default	2mA (default)		
u014	Input	PIN_315	0	RL_00	3.3V1V default	2mA (default)		
u015	Input	PIN_316	0	RL_00	3.3V1V default	2mA (default)		
u016	Input	PIN_317	0	RL_00	3.3V1V default	2mA (default)		
u017	Input	PIN_318	0	RL_00	3.3V1V default	2mA (default)		
u018	Input	PIN_319	0	RL_00	3.3V1V default	2mA (default)		
u019	Input	PIN_320	0	RL_00	3.3V1V default	2mA (default)		
u020	Input	PIN_321	0	RL_00	3.3V1V default	2mA (default)		
u021	Input	PIN_322	0	RL_00	3.3V1V default	2mA (default)		
u022	Input	PIN_323	0	RL_00	3.3V1V default	2mA (default)		
u023	Input	PIN_324	0	RL_00	3.3V1V default	2mA (default)		
u024	Input	PIN_325	0	RL_00	3.3V1V default	2mA (default)		
u025	Input	PIN_326	0	RL_00	3.3V1V default	2mA (default)		
u026	Input	PIN_327	0	RL_00	3.3V1V default	2mA (default)		
u027	Input	PIN_328	0	RL_00	3.3V1V default	2mA (default)		
u028	Input	PIN_329	0	RL_00	3.3V1V default	2mA (default)		
u029	Input	PIN_330	0	RL_00	3.3V1V default	2mA (default)		
u030	Input	PIN_331	0	RL_00	3.3V1V default	2mA (default)		
u031	Input	PIN_332	0	RL_00	3.3V1V default	2mA (default)		
u032	Input	PIN_333	0	RL_00	3.3V1V default	2mA (default)		
u033	Input	PIN_334	0	RL_00	3.3V1V default	2mA (default)		
u034	Input	PIN_335	0	RL_00	3.3V1V default	2mA (default)		
u035	Input	PIN_336	0	RL_00	3.3V1V default	2mA (default)		
u036	Input	PIN_337	0	RL_00	3.3V1V default	2mA (default)		
u037	Input	PIN_338	0	RL_00	3.3V1V default	2mA (default)		
u038	Input	PIN_339	0	RL_00	3.3V1V default	2mA (default)		
u039	Input	PIN_340	0	RL_00	3.3V1V default	2mA (default)		
u040	Input	PIN_341	0	RL_00	3.3V1V default	2mA (default)		
u041	Input	PIN_342	0	RL_00	3.3V1V default	2mA (default)		
u042	Input	PIN_343	0	RL_00	3.3V1V default	2mA (default)		
u043	Input	PIN_344	0	RL_00	3.3V1V default	2mA (default)		
u044	Input	PIN_345	0	RL_00	3.3V1V default	2mA (default)		
u045	Input	PIN_346	0	RL_00	3.3V1V default	2mA (default)		
u046	Input	PIN_347	0	RL_00	3.3V1V default	2mA (default)		
u047	Input	PIN_348	0	RL_00	3.3V1V default	2mA (default)		
u048	Input	PIN_349	0	RL_00	3.3V1V default	2mA (default)		
u049	Input	PIN_350	0	RL_00	3.3V1V default	2mA (default)		
u050	Input	PIN_351	0	RL_00	3.3V1V default	2mA (default)		
u051	Input	PIN_352	0	RL_00	3.3V1V default	2mA (default)		
u052	Input	PIN_353	0	RL_00	3.3V1V default	2mA (default)		
u053	Input	PIN_354	0	RL_00	3.3V1V default	2mA (default)		
u054	Input	PIN_355	0	RL_00	3.3V1V default	2mA (default)		
u055	Input	PIN_356	0	RL_00	3.3V1V default	2mA (default)		
u056	Input	PIN_357	0	RL_00	3.3V1V default	2mA (default)		
u057	Input	PIN_358	0	RL_00	3.3V1V default	2mA (default)		
u058	Input	PIN_359	0	RL_00	3.3V1V default	2mA (default)		
u059	Input	PIN_360	0	RL_00	3.3V1V default	2mA (default)		
u060	Input	PIN_361	0	RL_00	3.3V1V default	2mA (default)		
u061	Input	PIN_362	0	RL_00	3.3V1V default	2mA (default)		
u062	Input	PIN_363	0	RL_00	3.3V1V default	2mA (default)		
u063	Input	PIN_364	0	RL_00	3.3V1V default	2mA (default)		
u064	Input	PIN_365	0	RL_00	3.3V1V default	2mA (default)		
u065	Input	PIN_366	0	RL_00	3.3V1V default	2mA (default)		
u066	Input	PIN_367	0	RL_00	3.3V1V default	2mA (default)		
u067	Input	PIN_368	0	RL_00	3.3V1V default	2mA (default)		
u068	Input	PIN_369	0	RL_00	3.3V1V default	2mA (default)		
u069	Input	PIN_370	0	RL_00	3.3V1V default	2mA (default)		
u070	Input	PIN_371	0	RL_00	3.3V1V default	2mA (default)		
u071	Input	PIN_372	0	RL_00	3.3V1V default	2mA (default)		
u072	Input	PIN_373	0	RL_00	3.3V1V default	2mA (default)		
u073	Input	PIN_374	0	RL_00	3.3V1V default	2mA (default)		
u074	Input	PIN_375	0	RL_00	3.3V1V default	2mA (default)		
u075	Input	PIN_376	0	RL_00	3.3V1V default	2mA (default)		
u076	Input	PIN_377	0	RL_00	3.3V1V default	2mA (default)		
u077	Input	PIN_378	0	RL_00	3.3V1V default	2mA (default)		
u078	Input	PIN_379	0	RL_00	3.3V1V default	2mA (default)		
u079	Input	PIN_380	0	RL_00	3.3V1V default	2mA (default)		
u080	Input	PIN_381	0	RL_00	3.3V1V default	2mA (default)		
u081	Input	PIN_382	0	RL_00	3.3V1V default	2mA (default)		
u082	Input	PIN_383	0	RL_00	3.3V1V default	2mA (default)		
u083	Input	PIN_384	0	RL_00	3.3V1V default	2mA (default)		
u084	Input	PIN_385	0	RL_00	3.3V1V default	2mA (default)		
u085	Input	PIN_386	0	RL_00	3.3V1V default	2mA (default)		
u086	Input	PIN_387	0	RL_00	3.3V1V default	2mA (default)		
u087	Input	PIN_388	0	RL_00	3.3V1V default	2mA (default)		
u088	Input	PIN_389	0	RL_00	3.3V1V default	2mA (default)		
u089	Input	PIN_390	0	RL_00	3.3V1V default	2mA (default)		
u090	Input	PIN_391	0	RL_00	3.3V1V default	2mA (default)		
u091	Input	PIN_392	0	RL_00	3.3V1V default	2mA (default)		
u092	Input	PIN_393	0	RL_00	3.3V1V default	2mA (default)		
u093	Input	PIN_394	0	RL_00	3.3V1V default	2mA (default)		
u094	Input	PIN_395	0	RL_00	3.3V1V default	2mA (default)		
u095	Input	PIN_396	0	RL_00	3.3V1V default	2mA (default)		
u096	Input	PIN_397	0	RL_00	3.3V1V default	2mA (default)		
u097	Input	PIN_398	0	RL_00	3.3V1V default	2mA (default)		
u098	Input	PIN_399	0	RL_00	3.3V1V default	2mA (default)		
u099	Input	PIN_400	0	RL_00	3.3V1V default	2mA (default)		
u100	Input	PIN_401	0	RL_00	3.3V1V default	2mA (default)		
u101	Input	PIN_402	0	RL_00	3.3V1V default	2mA (default)		
u102	Input	PIN_403	0	RL_00	3.3V1V default	2mA (default)		
u103	Input	PIN_404	0	RL_00	3.3V1V default	2mA (default)		
u104	Input	PIN_405	0	RL_00	3.3V1V default	2mA (default)		
u105	Input	PIN_406	0	RL_00	3.3V1V default	2mA (default)		
u106	Input	PIN_407	0	RL_00	3.3V1V default	2mA (default)		
u107	Input	PIN_408	0	RL_00	3.3V1V default	2mA (default)		
u108	Input	PIN_409	0	RL_00	3.3V1V default	2mA (default)		
u109	Input	PIN_410	0	RL_00	3.3V1V default	2mA (default)		
u110	Input	PIN_411	0	RL_00	3.3V1V default	2mA (default)		
u111	Input	PIN_412	0	RL_00	3.3V1V default	2mA (default)		
u112	Input	PIN_413	0	RL_00	3.3V1V default	2mA (default)		
u113	Input	PIN_414	0	RL_00	3.3V1V default	2mA (default)		
u114	Input	PIN_415	0	RL_00	3.3V1V default	2mA (default)		
u115	Input	PIN_416	0	RL_00	3.3V1V default	2mA (default)		
u116	Input	PIN_417	0	RL_00	3.3V1V default	2mA (default)		
u117	Input	PIN_418	0	RL_00	3.3V1V default	2mA (default)		
u118	Input	PIN_419	0	RL_00	3.3V1V default	2mA (default)		
u119	Input	PIN_420	0	RL_00	3.3V1V default	2mA (default)		
u120	Input	PIN_421	0	RL_00	3.3V1V default	2mA (default)		
u121	Input	PIN_422	0	RL_00	3.3V1V default	2mA (default)		
u122	Input	PIN_423	0	RL_00	3.3V1V default	2mA (default)		
u123	Input	PIN_424	0	RL_00	3.3V1V default	2mA (default)		
u124	Input	PIN_425	0	RL_00	3.3V1V default	2mA (default)		
u125	Input	PIN_426	0	RL_00	3.3V1V default	2mA (default)		
u126	Input	PIN_427	0	RL_00	3.3V1V default	2mA (default)		
u127	Input	PIN_428	0	RL_00	3.3V1V default	2mA (default)		
u128	Input	PIN_429	0	RL_00	3.3V1V default	2mA (default)		
u129	Input	PIN_430	0	RL_00	3.3V1V default	2mA (default)		
u130	Input	PIN_431	0	RL_00	3.3V1V default	2mA (default)		
u131	Input	PIN_432	0	RL_00	3.3V1V default	2mA (default)		
u132	Input	PIN_433	0	RL_00	3.3V1V default	2mA (default)		
u133	Input	PIN_434	0	RL_00	3.3V1V default	2mA (default)		
u134	Input	PIN_435	0	RL_00	3.3V1V default	2mA (default)		
u135	Input	PIN_436	0	RL_00	3.3V1V default	2mA (default)		
u136	Input	PIN_437	0	RL_00	3.3V1V default	2mA (default)		
u137	Input	PIN_438	0	RL_00	3.3V1V default	2mA (default)		
u138	Input	PIN_439	0	RL_00	3.3V1V default	2mA (default)		
u139	Input	PIN_440	0	RL_00	3.3V1V default	2mA (default)		
u140	Input	PIN_441	0	RL_00	3.3V1V default	2mA (default)		
u141	Input	PIN_442	0	RL_00	3.3V1V default	2mA (default)		
u142	Input	PIN_443	0	RL_00	3.3V1V default	2mA (default)		
u143	Input	PIN_444	0	RL_00	3.3V1V default	2mA (default)		
u144	Input	PIN_445	0	RL_00	3.3V1V default	2mA (default)		
u145	Input	PIN_446	0	RL_00	3.3V1V default	2mA (default)		
u146	Input	PIN_447	0	RL_00	3.3V1V default	2mA (default)		
u147	Input	PIN_448	0	RL_00	3.3V1V default	2mA (default)		
u148	Input	PIN_449	0	RL_00	3.3V1V default	2mA (default)		
u149	Input	PIN_450	0	RL_00	3.3V1V default	2mA (default)		
u150	Input	PIN_451	0	RL_00	3.3V1V default	2mA (default)		
u151	Input	PIN_452	0	RL_00	3.3V1V default	2mA (default)		
u152	Input	PIN_453	0	RL_00	3.3V1V default	2mA (default)		
u153	Input	PIN_454	0	RL_00	3.3V1V default	2mA (default)		
u154	Input	PIN_455	0	RL_00	3.3V1V default	2mA (default)		
u155	Input	PIN_456	0	RL_00	3.3V1V default	2mA (default)		
u156	Input	PIN_457	0	RL_00	3.3V1V default	2mA (default)		
u157	Input	PIN_458	0	RL_00	3.3V1V default	2mA (default)		
u158	Input	PIN_459	0	RL_00	3.3V1V default	2mA (default)		
u159	Input	PIN_460	0	RL_00	3.3V1V default	2mA (default)		
u160	Input	PIN_461	0	RL_00	3.3V1V default	2mA (default)		
u161	Input	PIN_462	0	RL_00	3.3V1V default	2mA (default)		
u162	Input	PIN_463	0	RL_00	3.3V1V default	2mA (default)		
u163	Input	PIN_464	0	RL_00	3.3V1V default	2mA (default)		
u164	Input	PIN_465	0	RL_00	3.3V1V default	2mA (default)		
u165	Input	PIN_466	0	RL_00	3.3V1V default	2mA (default)		
u166	Input	PIN_467	0	RL_00	3.3V1V default	2mA (default)		
u167	Input	PIN_468	0	RL_00	3.3V1			