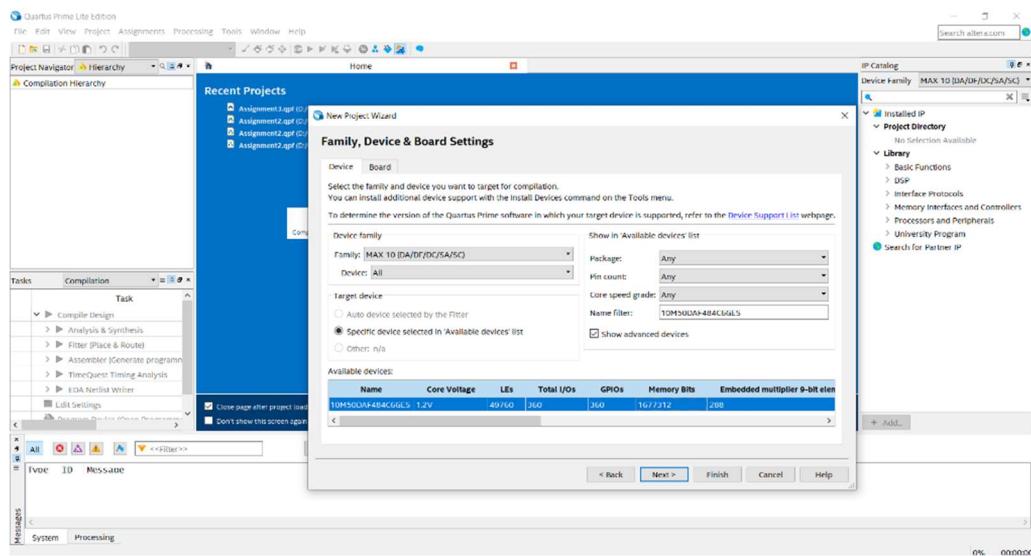


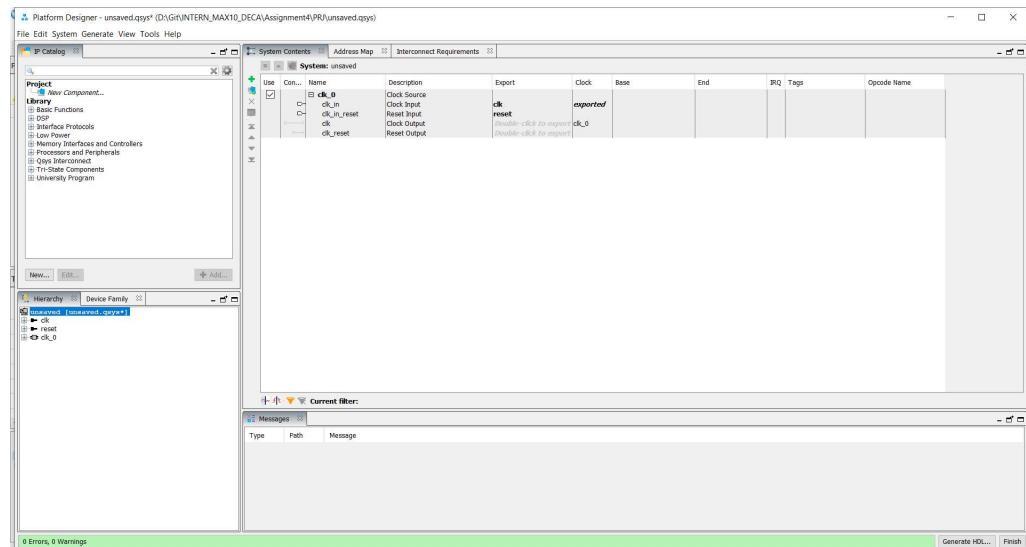
2.5 สร้าง GPIO IP -> simulation -> ลองที่บอร์ดจริง

- สร้าง Custom GPIO IP

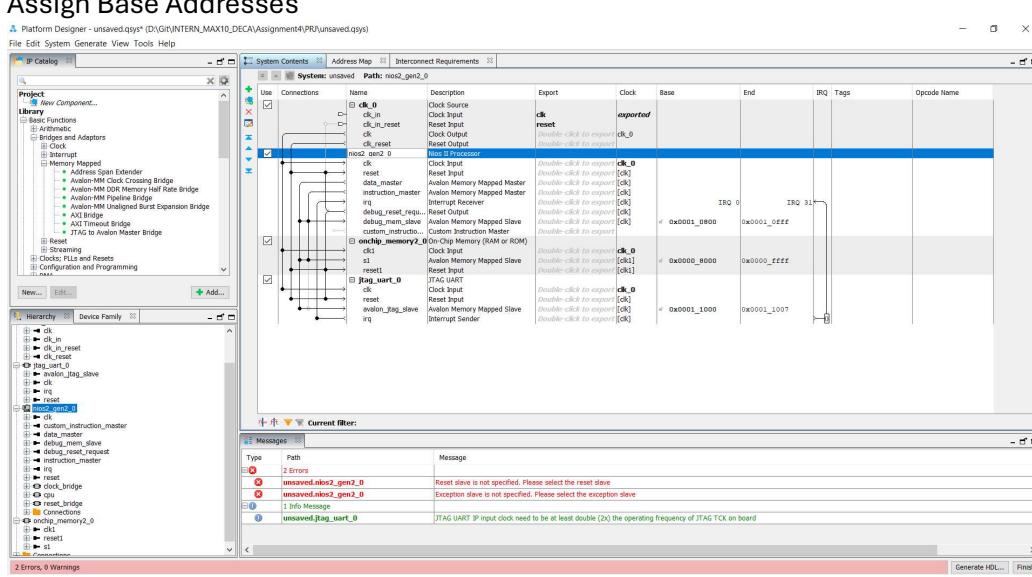
- สร้าง Project ใน Quartus II (เลือก device ที่ใช้งานเป็น 10M50DAF484C6GES สำหรับบอร์ด DECA MAX10)



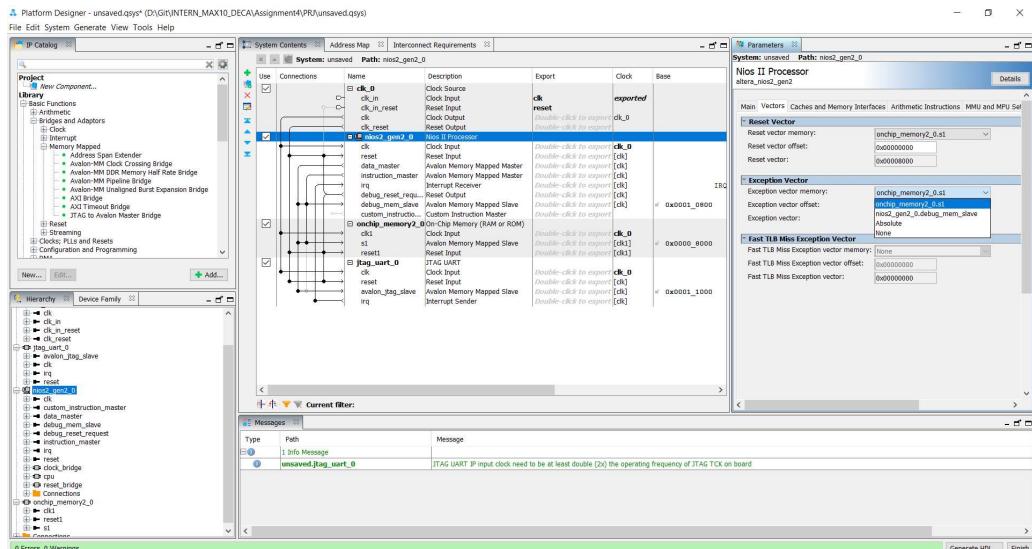
- เปิด Tools → Platform Designer ซึ่งจะมีสัญญาณ Clock ในระบบโดยอัตโนมัติ และเราจะใช้สัญญาณ Reset แบบ Active Low ในระบบ จากนั้น double click ที่ชื่อของสัญญาณเพื่อ Export



- เพิ่ม Nios II Processor (Nios II/e), On-Chip Memory (RAM) และ JTAG UART (optional) เข้ากับระบบ และเชื่อมต่อสัญญาณต่างๆ และกำหนด Address ของแต่ละ component โดยกด System → Assign Base Addresses

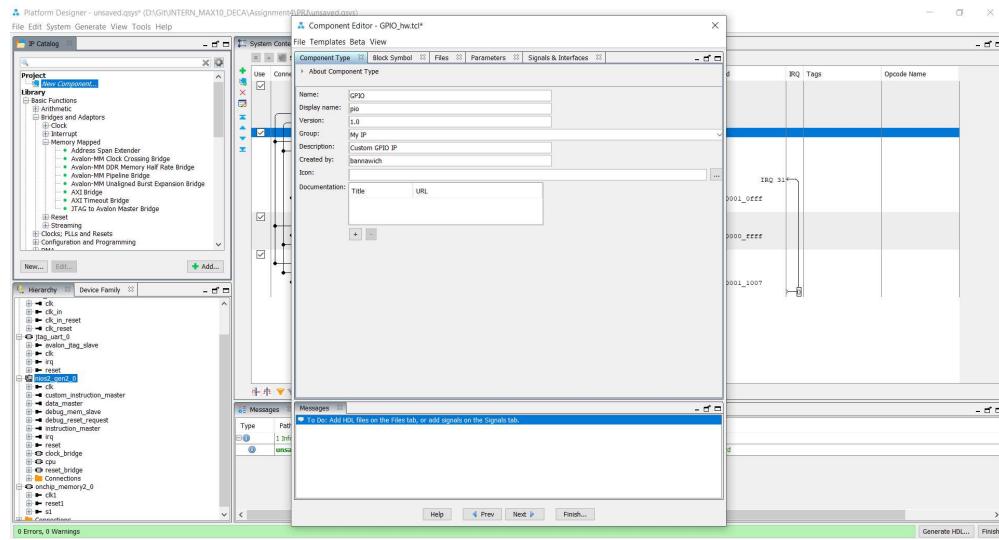


- กำหนด Reset vector memory และ Exception vector memory เป็น onchip memory

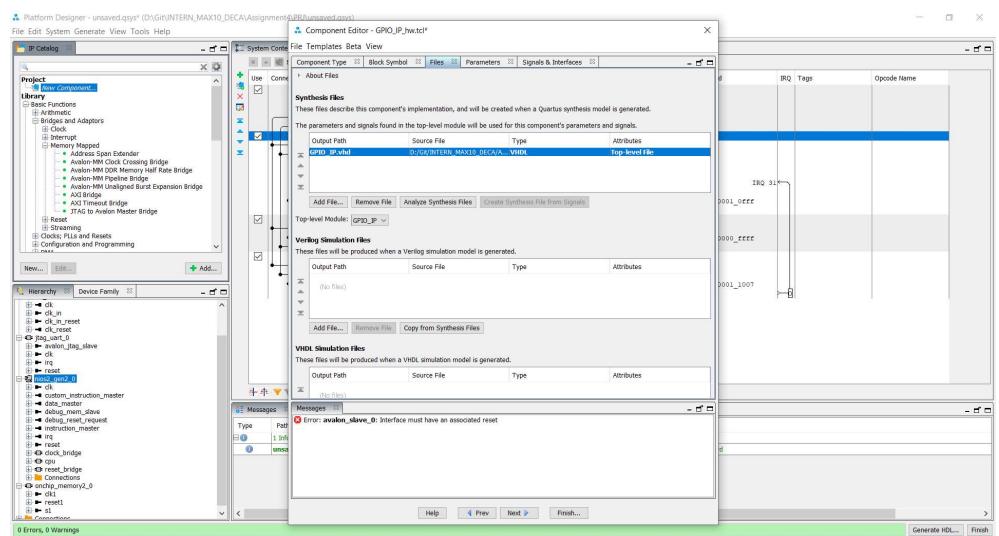


- เลือก New Component บนแท็บ IP Catalog

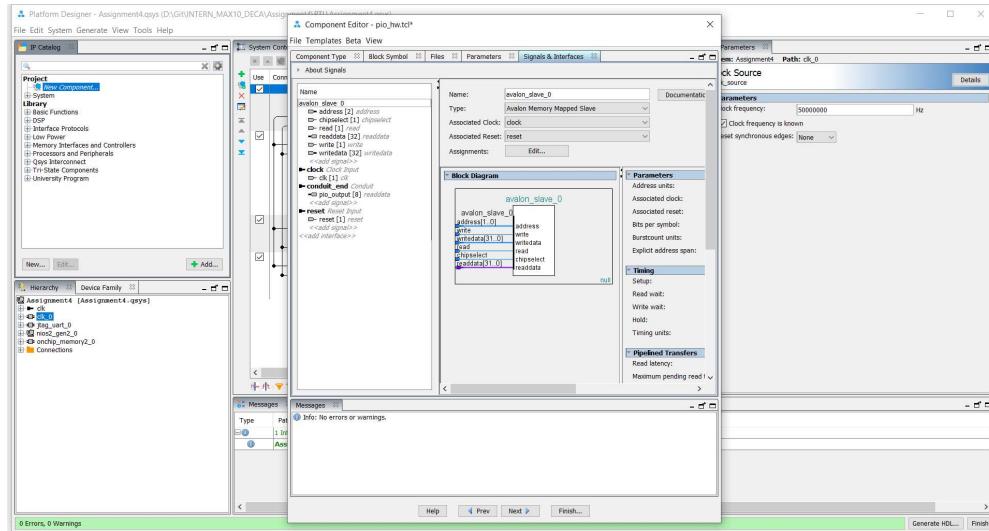
ชั้ง component ที่เราจะเพิ่มในระบบ (GPIO IP) ประกอบด้วย Avalon Memory-Mapped Interface และ Avalon Conduit Interface ขั้นตอนแรก คือการตั้งชื่อ และกำหนดชื่อที่จะแสดง และระบุ Group



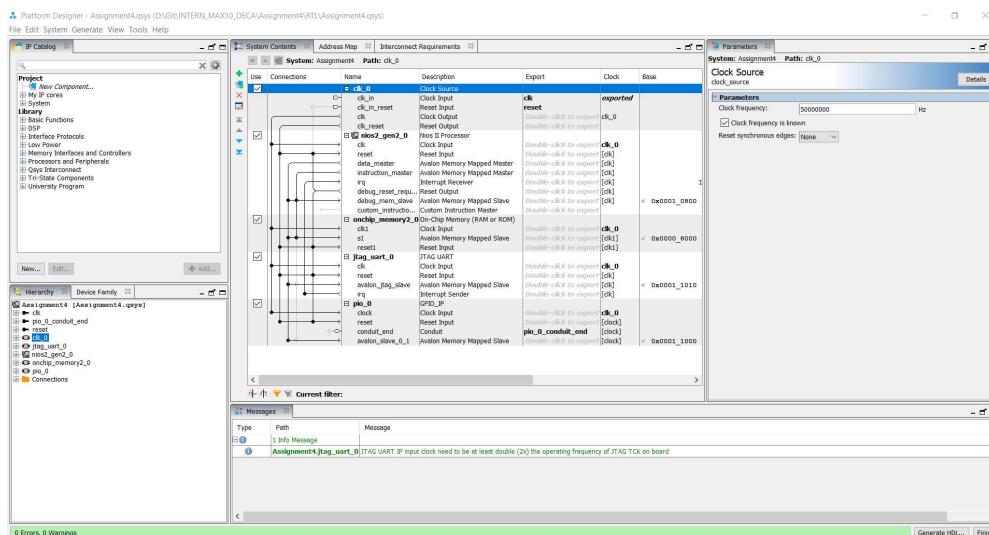
- ในหน้า Files คลิกที่ปุ่ม Add เพื่อเพิ่มไฟล์ VHDL ที่สร้าง component (IP core) ของเรา และกด Analyze Synthesis Files เพื่อให้ Qsys analyze ไฟล์นี้เพื่อกำหนดประเภทของ interfaces ที่ component ใช้



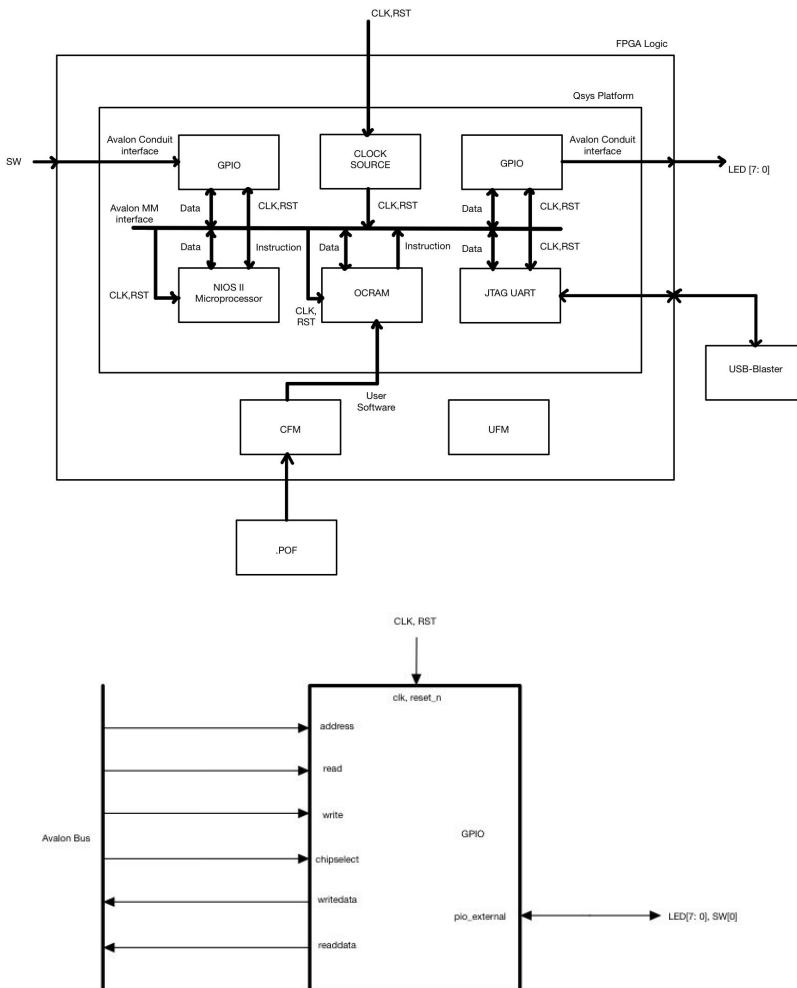
- ในหน้า Signals & Interfaces ใส่สัญญาณใน interfaces และระบุประเภทสัญญาณให้ถูกต้อง



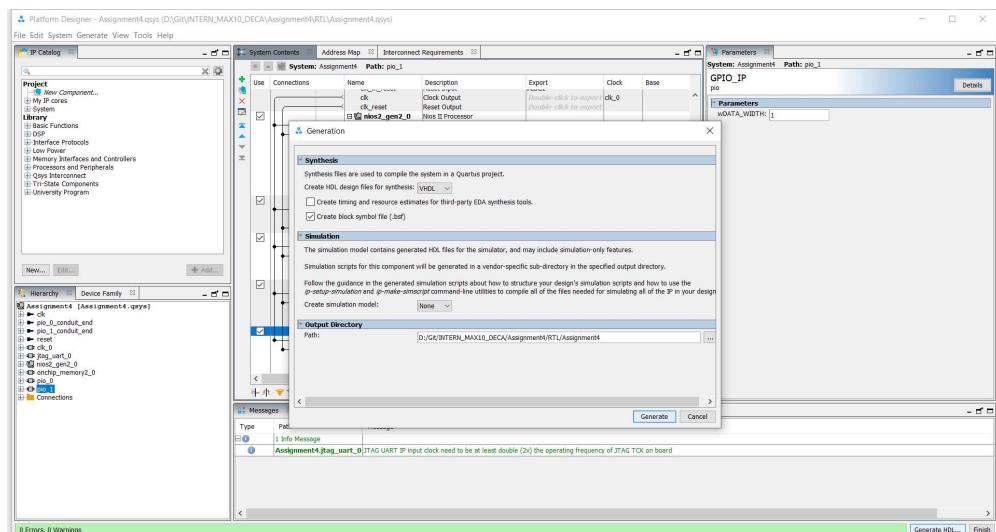
- เพิ่ม IP ที่สร้างขึ้น และเชื่อมต่อสัญญาณต่างๆ



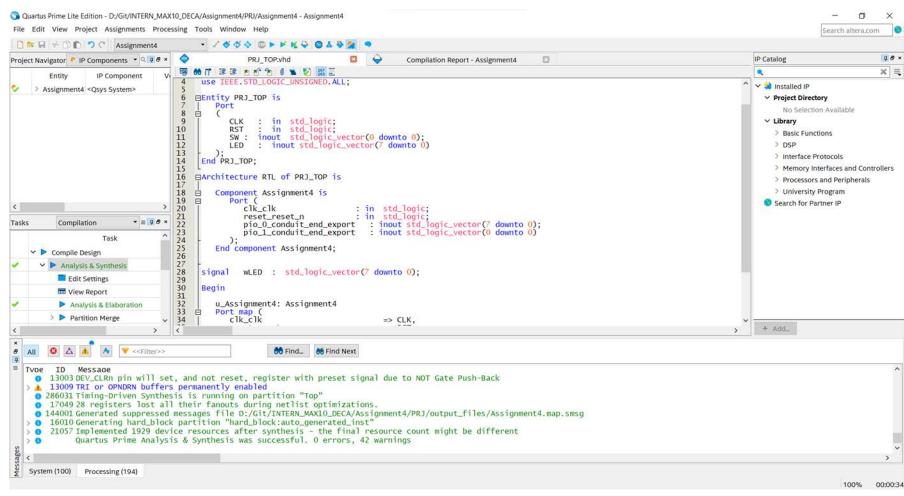
- ลักษณะการเชื่อมต่อแสดงด้วยแผนภาพได้ ดังนี้



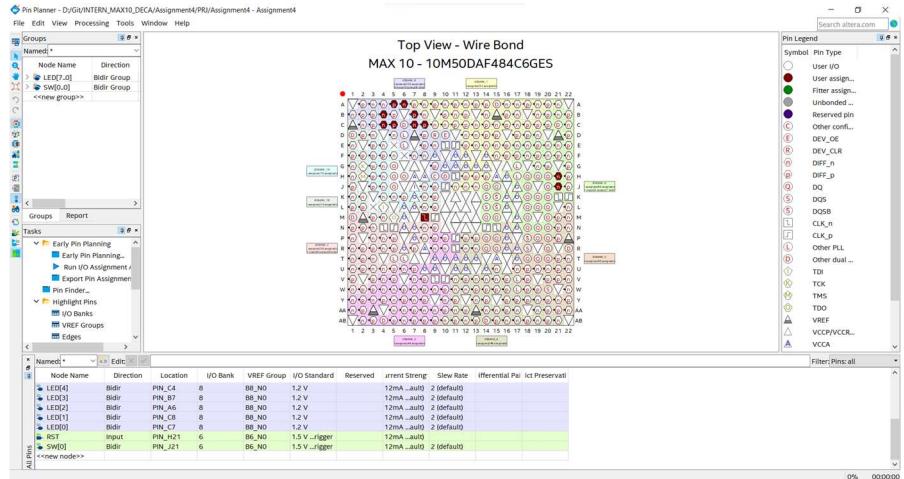
- Save และ Generate HDL



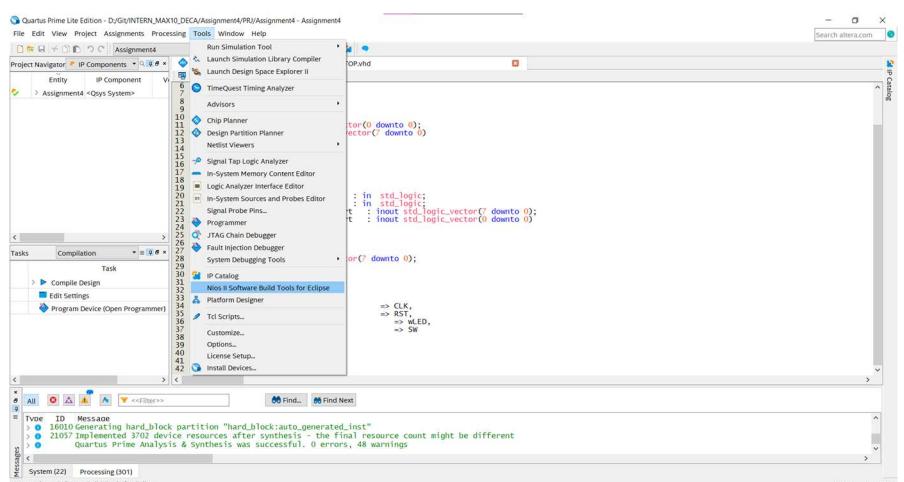
- กด Finish และลับมาที่ Quartus เพื่อกด Analysis&Synthesis



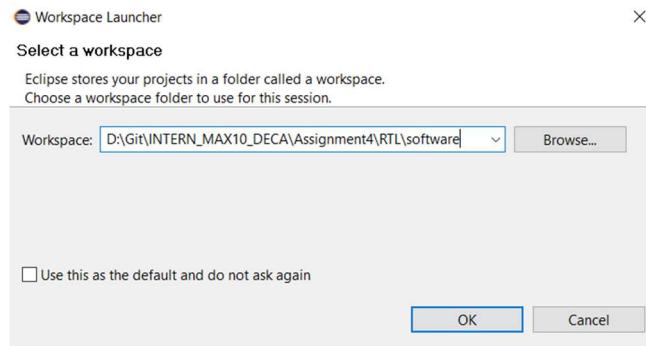
- เปิด Pin Planner และ เที่ยมต่อสัญญาณกับ Pin ต่างๆให้ถูกต้อง



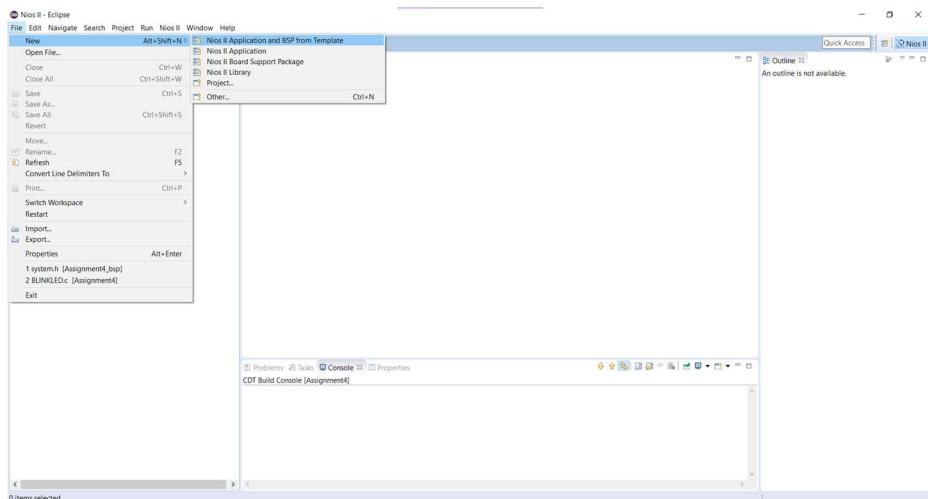
- ไปที่ Tools → Nios II Software Build Tools for Eclipse



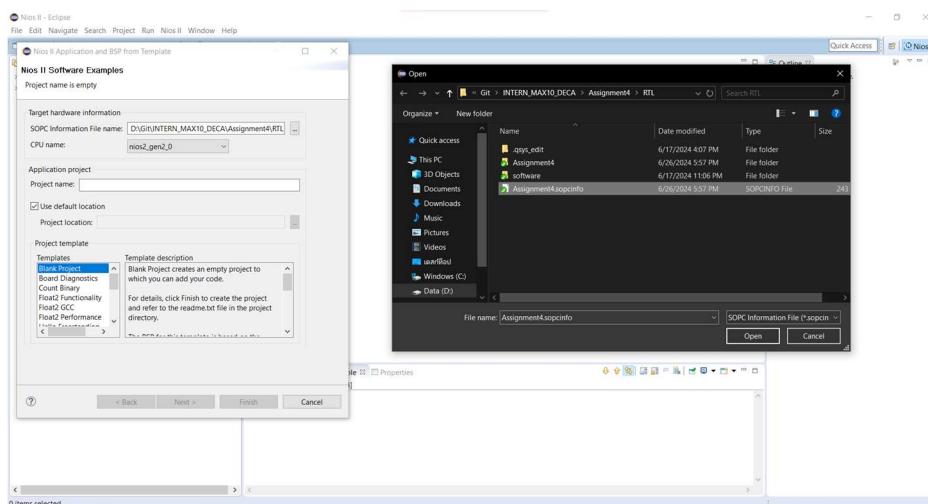
- เลือก Folder สำหรับเป็น workspace



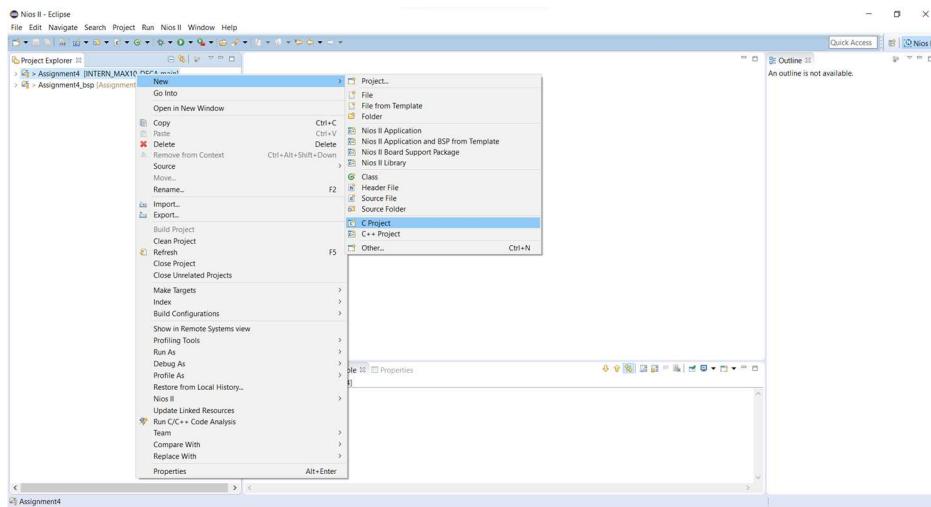
- สร้าง Project จาก Template



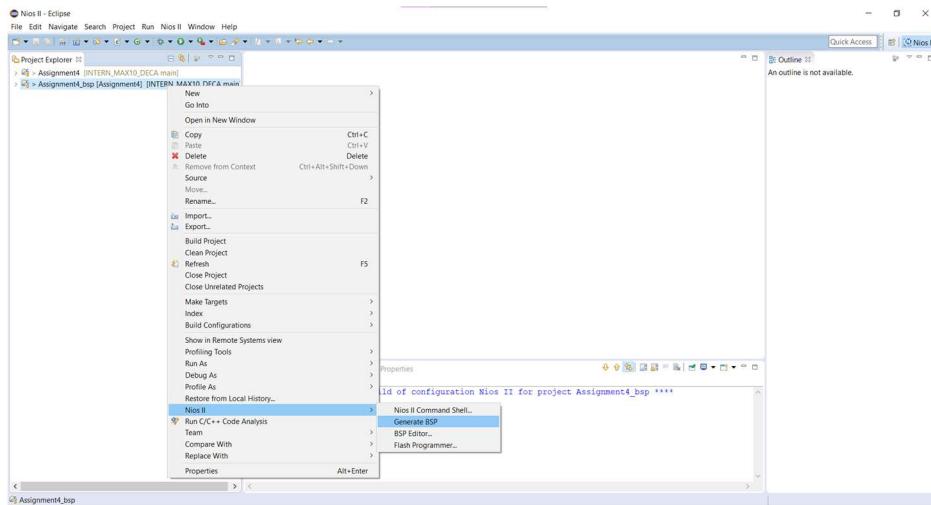
- เลือก Template เป็น Blank Project และเลือก SOPC ไฟล์ เป็นไฟล์ sopcinfo ที่ได้จากการ Quartus และตั้งชื่อ Project เสร็จแล้วกด Finish



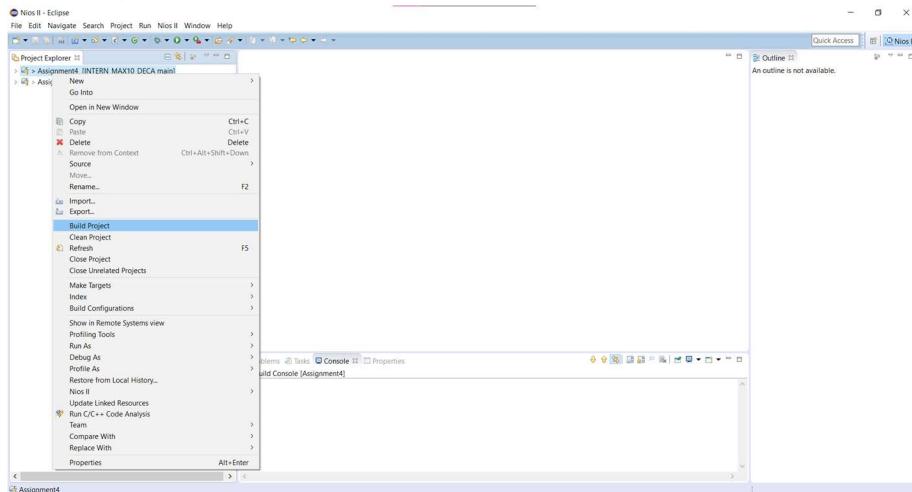
- เพิ่ม C code ที่จะโปรแกรมลง Nios II ใน Project



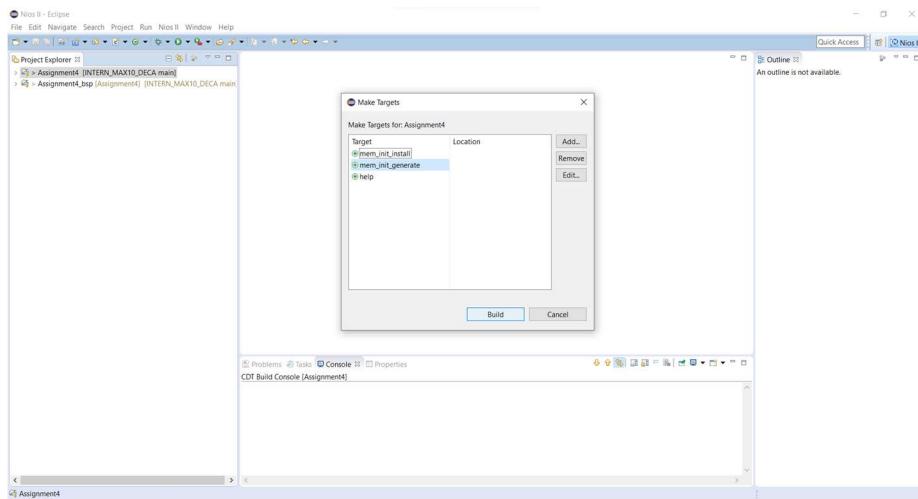
- Generate BSP



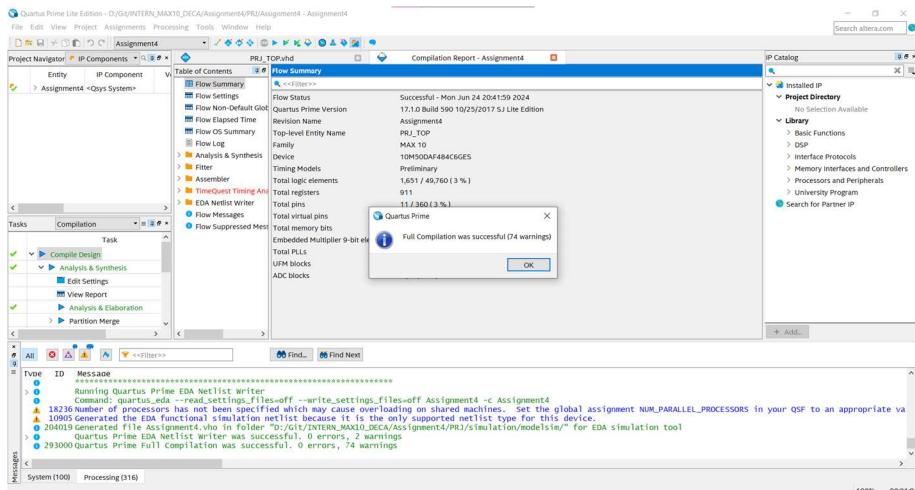
- Build Project



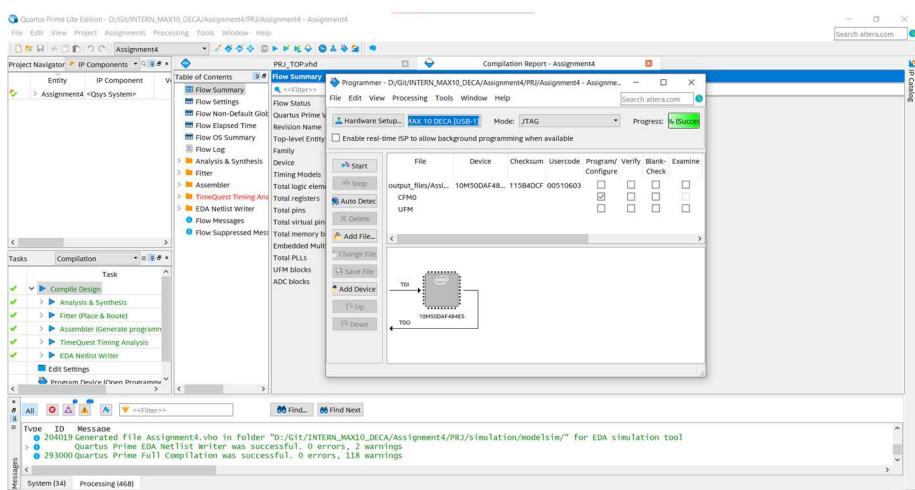
- กด Shift+F9 เลือก mem_init_generate และกด Build



- กลับมาที่ Quartus และ Compile Project

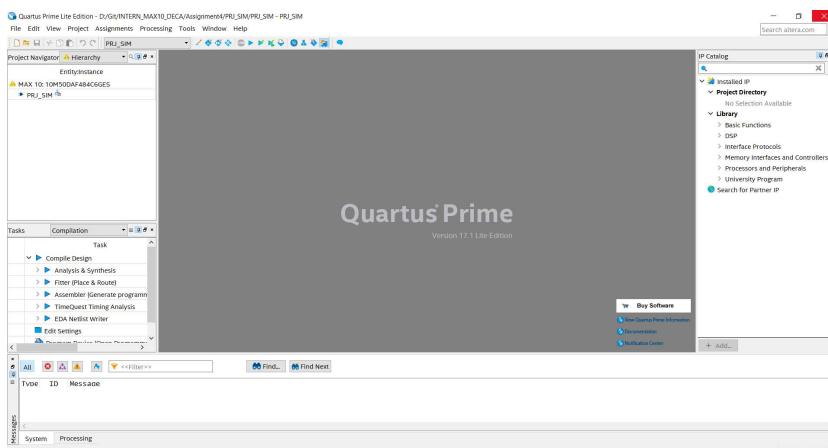


- โปรแกรมลงบอร์ด

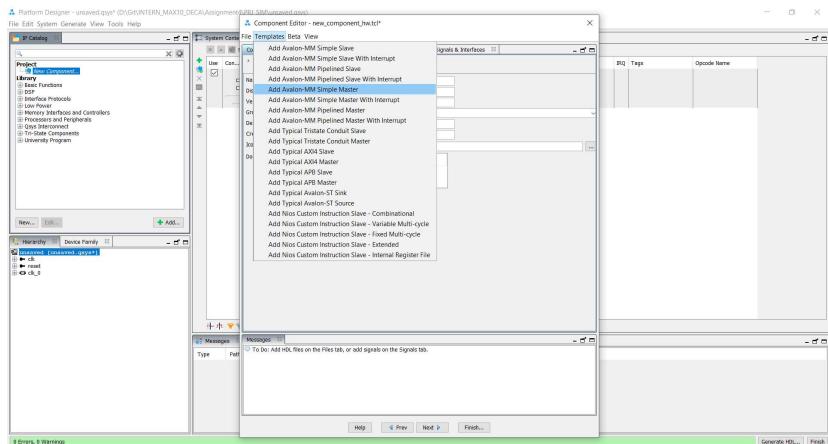


- Simulation ด้วย Testbench (<https://ndrsolution.com/2021/11/12/fpga-avalon-bus-simulation/>)

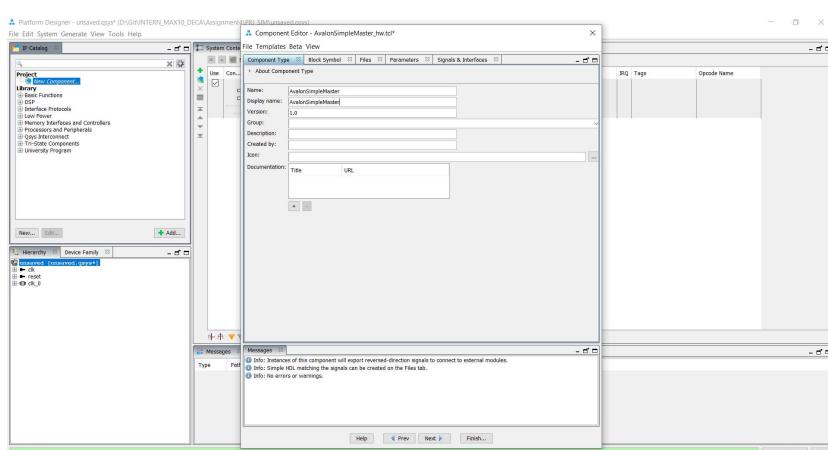
- สร้าง Project ใหม่ขึ้นมา



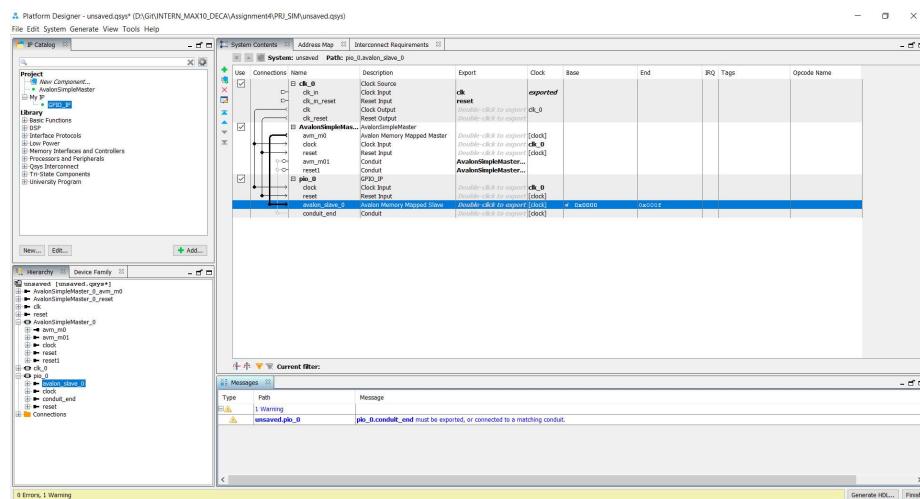
- เปิด Tools → Platform Designer และคลิกไปที่ New component เพื่อสร้าง Avalon Master Interface



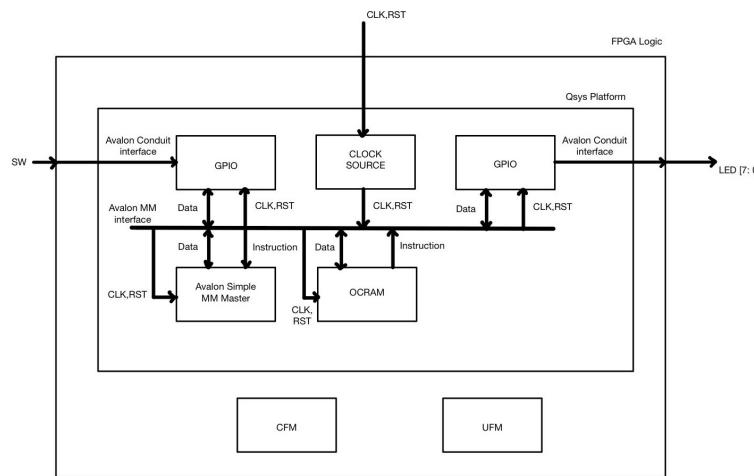
- ตั้งชื่อ component และกด Finish



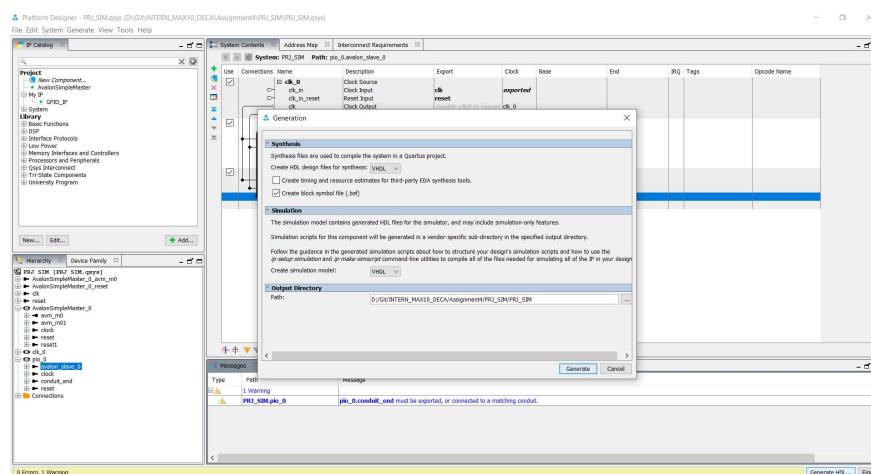
- เพิ่ม AvalonSimpleMaster และ IP ที่เราต้องการ simulate เข้าในระบบ และเข้ามายอมต่อสัญญาณให้ถูกต้อง



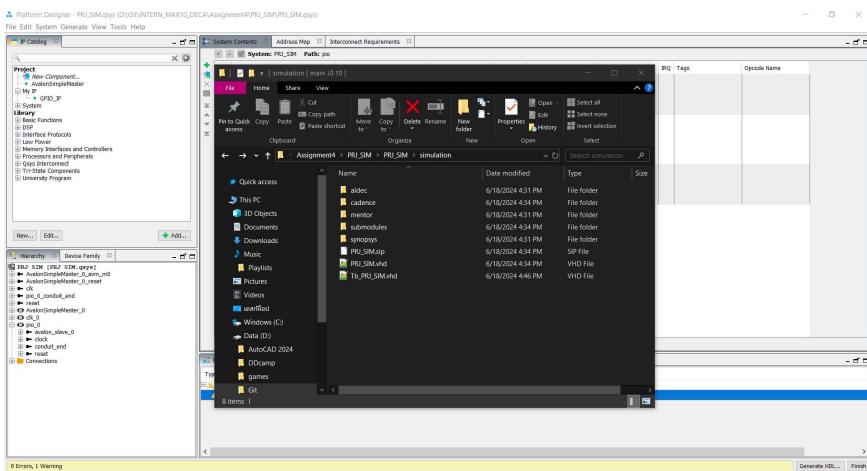
- ลักษณะการเข้ามายอมต่อแสดงด้วยแผนภาพได้ ดังนี้



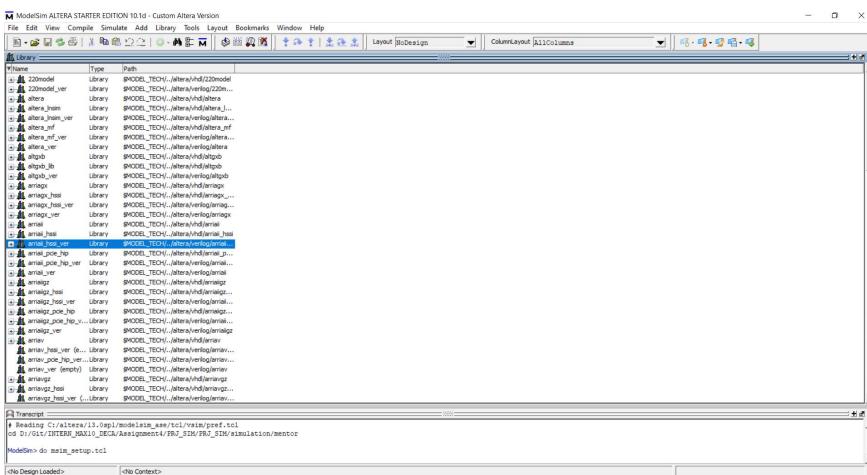
- Save และ generate โดยใน Create simulation model ให้เลือก VHDL



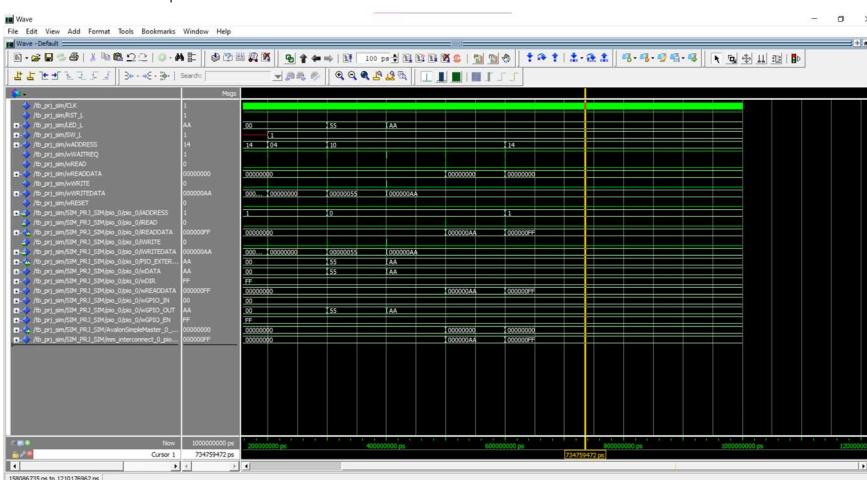
- สร้าง testbench ขึ้นมา และวางไฟล์เดอร์ simulation และไฟล์ msim_setup.tcl และชื่อ TOP_LEVEL_NAME ให้เป็นชื่อ testbench จากนั้นที่ alias com ให้เพิ่ม Testbench เข้าไปด้วย



- เปิด Modelsim ขึ้นมา แล้วไปที่ simulation/mentor ที่ console ให้พิมพ์ `do msim_setup.tcl` และ พิมพ์ `com` เพื่อ compile และ พิมพ์ `ld` เพื่อเริ่มต้นการ simulate



- Add wave ต่างๆเข้ามา และ กด Simulate → run all



- ใช้ฟังก์ชัน procedure ใน Testbench เพื่อความสะดวกต่อการเปลี่ยนค่าสัญญาณ

```

--> Procedure IO_WRITE (
-->   address : in std_logic_vector(7 downto 0);
-->   data  : in std_logic_vector(31 downto 0);
-->   signal wADDRESS : out std_logic_vector(7 downto 0);
-->   signal wWAITREQ : in std_logic;
-->   signal wWRITE : out std_logic;
-->   signal wWRITEDATA : out std_logic_vector(31 downto 0)
--> ) is
--> Begin
-->   wADDRESS <= address;
-->   wWRITEDATA <= data;
-->   wWRITE <= '1';
-->   wait until CLK'event and CLK = '1';
-->   while (wWAITREQ = '1') loop
-->     wait for kCLK;
-->   end loop;
-->   wWRITE <= '0';
-->   wait until CLK'event and CLK = '1';
--> End Procedure;
-->
--> Procedure IO_READ (
-->   address : in std_logic_vector(7 downto 0);
-->   signal wADDRESS : out std_logic_vector(7 downto 0);
-->   signal wWAITREQ : in std_logic;
-->   signal wREAD : out std_logic
--> ) is
--> Begin
-->   wADDRESS <= address;
-->   wREAD <= '1';
-->   wait until CLK'event and CLK = '1';
-->   while (wWAITREQ = '1') loop
-->     wait for kCLK;
-->   end loop;
-->   wait until CLK'event and CLK = '1';
-->   wREAD <= '0';
--> End procedure;
-->
--> Process
--> Begin
-->   RST_L <=> '0';
-->   wait for kCLK * 5;
-->   wait until wCLK50'event and wCLK50 = '0';
-->   RST_L <= '1';
-->
-->   wait until CLK'event and CLK = '1';
-->   wait for 100 us;
-->   IO_WRITE(x"01", x"0000_00FF", wADDRESS, wWAITREQ, wWRITE, wWRITEDATA);
-->   wait for 100 us;
-->   IO_WRITE(x"00", x"0000_0055", wADDRESS, wWAITREQ, wWRITE, wWRITEDATA);
-->   wait for 100 us;
-->   IO_WRITE(x"00", x"0000_00AA", wADDRESS, wWAITREQ, wWRITE, wWRITEDATA);
-->   wait for 100 us;
-->   IO_READ(x"01", wADDRESS, wWAITREQ, wREAD);
-->   wait for 100 us;
-->   IO_READ(x"00", wADDRESS, wWAITREQ, wREAD);
-->   wait;
--> End process;

```