Présentation du DS : On considère le module capteur à ultra-sons SR HC-04 **.** Le but du DS est de :

- réaliser la description VHDL permettant d'obtenir une distance en cm, codée en binaire et affichée telle quelle sur des leds.
- mettre en oeuvre un processeur NIOS pour récupérer cette distance et la traiter pour pouvoir l'afficher sur les afficheurs 7 segments de la DE10 Lite et afficher un numéro d'itération des mesures.

Le schéma de câblage de ce module avec le FPGA est donné ci-dessous. Le chronogramme associé représente le principe d'une mesure via les signaux ***trig*** et ***echo***. Le FPGA génère une impulsion de lancement de la mesure sur le signal ***trig***. Le module SR HC-04 envoie alors un train de 8 impulsions d'ultra-sons à $40KHz$. L'onde se propage et se réfléchit vers le capteur, après avoir rencontré un obstacle. L'intervalle de temps entre l'instant de l'émission du train d'impulsion et la réception de l'onde réfléchie est proportionnelle à la distance. Le module SR HC-04 fournit sur sa sortie ***echo*** une impulsion matérialisant cet intervalle de temps. La mesure de la durée de cet intervalle de temps donnera la distance selon la formule suivante : $d = k.\frac{17}{100}$ où d est directement la distance en ***centimètres*** . k est le nombre de périodes d'une horloge de référence à $100KHz$ que l'on compte pendant toute la durée où ***echo=1***. On prendra une impulsion ***trig*** de $10\,\mu s$ qui se répètera toutes les $150\,ms$.

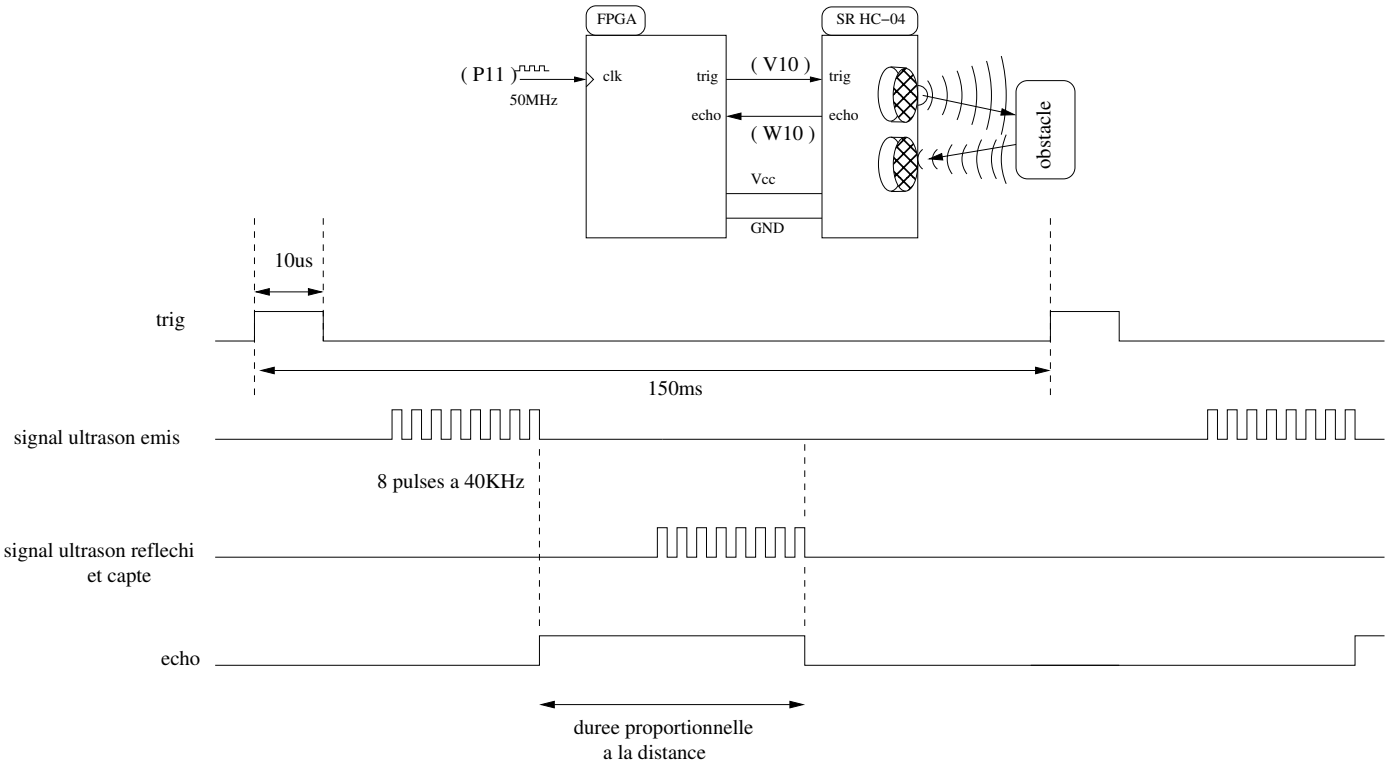
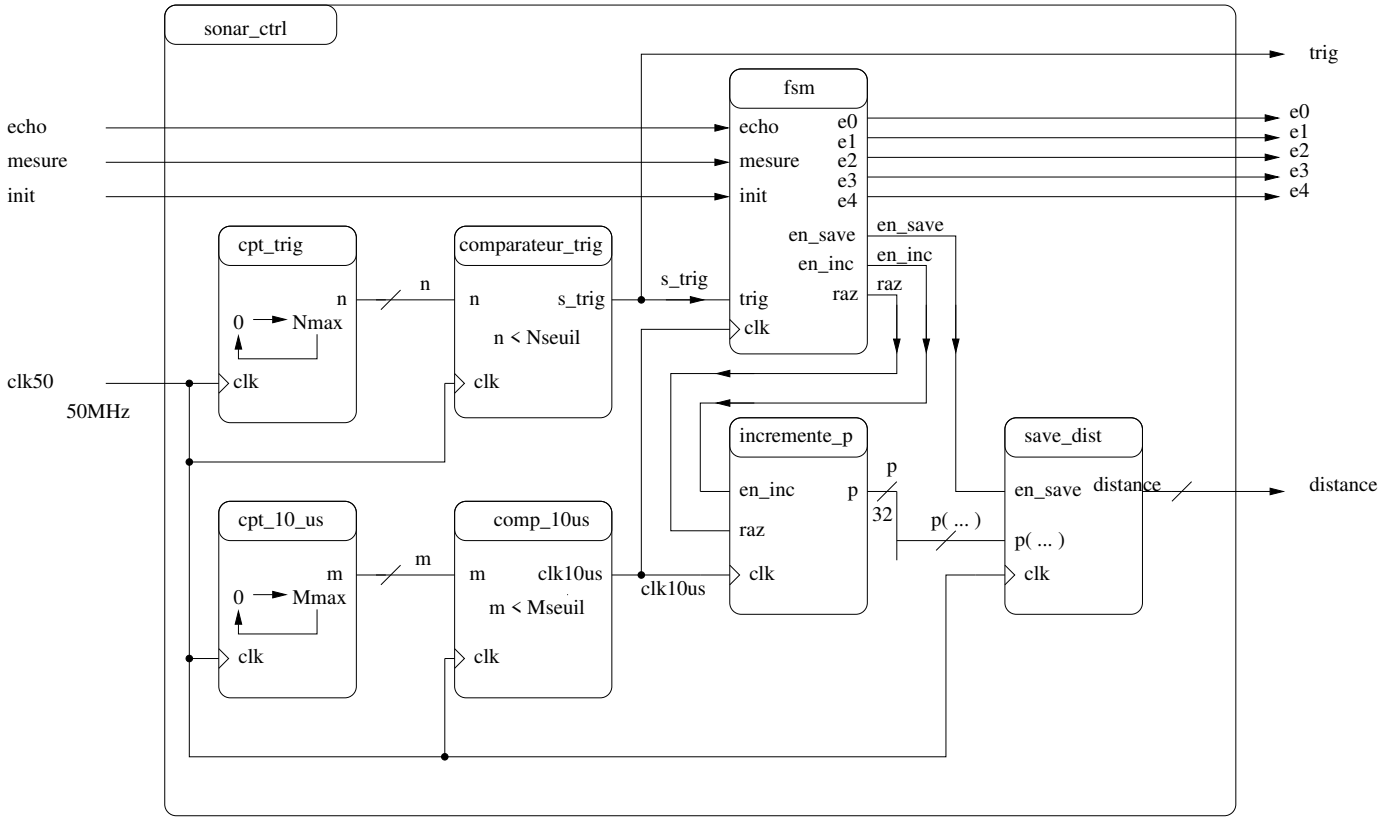


figure 1: câblage et principe de mesure

Partie 1 : Dans un premier temps, on s'intéresse à la mesure de distance . Et par conséquent on s'intéresse aux fonctions logiques générant l'impulsion ***trig*** d'une part, et l'horloge de mesure de la durée de l'impulsion ***echo*** d'autre part, pour fournir une mesure de la distance ***distance***. Le schéma synoptique des fonctions est donné ci-dessous.



Question 1 : préparation théorique. ***Faire valider les résultats de vos calculs par l'enseignant***

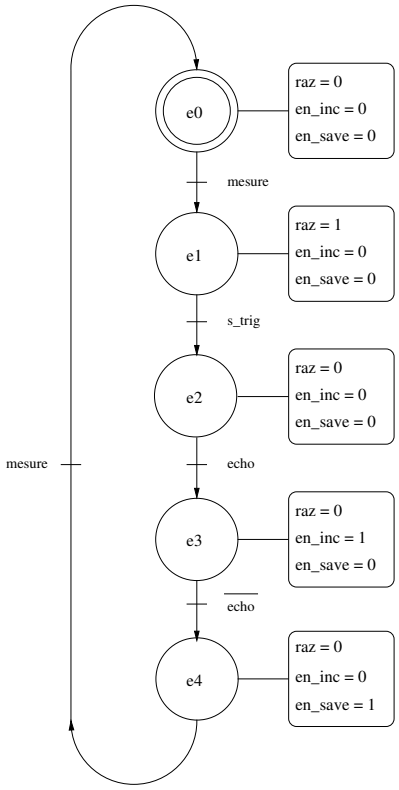
- Calculer ***Nmax*** en décimal, pour que la période de comptage de ***cpt_trig*** corresponde aux $150\,ms$ souhaitées, sachant qu'on dispose de l'horloge de référence ***clk50*** à $50\,MHz$. En déduire la dimension du signal ***n*** , c'est à dire le nombre de bits du std_logic_vector ***n*** .
- Déterminer numériquement en décimal la valeur de comparaison ***N_seuil*** du comparateur ***comparateur_trig*** pour générer l'impulsion ***trig*** sur le signal ***s_trig*** selon le chronogramme de la figure 1. On considèrera que la valeur 0 du compteur ***cpt_trig*** correspond à l'instant du front montant de ***s_trig***.
- Calculer ***Mmax*** en décimal, pour que la période de comptage de ***cpt_10_us*** corresponde aux $10\,\mu s$ souhaitées, avec la même horloge de référence à $50\,MHz$. En déduire la dimensionn du signal ***m*** , c'est àdire le nombre de bits du std_logic_vector ***m*** .
- Déterminer numériquement en décimal la valeur de comparaison ***M_seuil*** du comparateur ***comp_10us*** pour disposer d'un signal d'horloge **clk10us** de $10\,\mu s$ de période (soit une fréquence de $100\,KHz$) et de rapport cyclique $\alpha = \frac{1}{2}$.
- Sachant que la distance maximale mesurée par le module est de 4 mètres, indiquer la taille du std_logic_vector de la sortie ***distance*** qui est calculée et fournie en centimètres (bien sûr, le vecteur du signal ***p(...)*** présenté à l'entrée de ***save_dist*** sera de même dimension et ne représente qu'une partie du vecteur ***p***).
- Déterminer p et q avec $p \in \mathbb{N}$ et $q \in \mathbb{N}$, et tel que q soit une puissance de 2 ($q = 128\,ou\,256\,ou\,512\,ou\,1024\,ou\,...$) . On cherchera un couple de valeurs p et q de manière à ce que $\frac{p}{q}$ approche $\frac{17}{100}$ avec une bonne précision $\left(environ\,10^{-3}\right)$. Indication : utiliser la contrainte sur q pour rechercher quelques valeurs de p , et finalement choisir le couple $(p;q)$ en regard de la précision demandée. Le principe de ***incremente p*** consistera alors à ajouter la valeur ***p*** à chaque coup d'horloge.

Question 2 : préparer un projet ***sonar_ctrl*** sous quartus pour tester le fonctionnement du module en donnant la description VHDL de l'entité et des signaux intermédiaires selon le schéma de la figure 2. Poursuivre en donnant la description VHDL de l'architecture, c'est à dire des fonctions ***cpt_trig*** , ***comparateur_trig*** , ***cpt_10_us*** , ***comp_10us***, ***incremente_p*** et ***save_dist*** sous la forme de 4 process synchrones. ***Faire valider le principe de cette description des process par l'enseignant.***

Question 3 : compléter la question 2 en ajoutant la description VHDL de la machine d'état ***fsm*** selon la connexion de la figure ci-dessous. On la décrira sous la forme d'un process synchrone correspondant au graphe d'état de la figure 3. On n'oubliera pas de décrire l'activation des sorties de la ***fsm*** (***en_inc*** , ***en_save*** et ***raz***) . On terminera alors la description complète de ***sonar_ctrl*** en ajoutant les signaux de sortie permettant de vérifier

le fonctionnement de la fsm (***e0, e1, e2, e3, e4***). ***Faire valider le principe de cette description de la fsm par l'enseignant.***

```
type state is (e0,e1,e2,e3,e4);
signal etat : state;
```



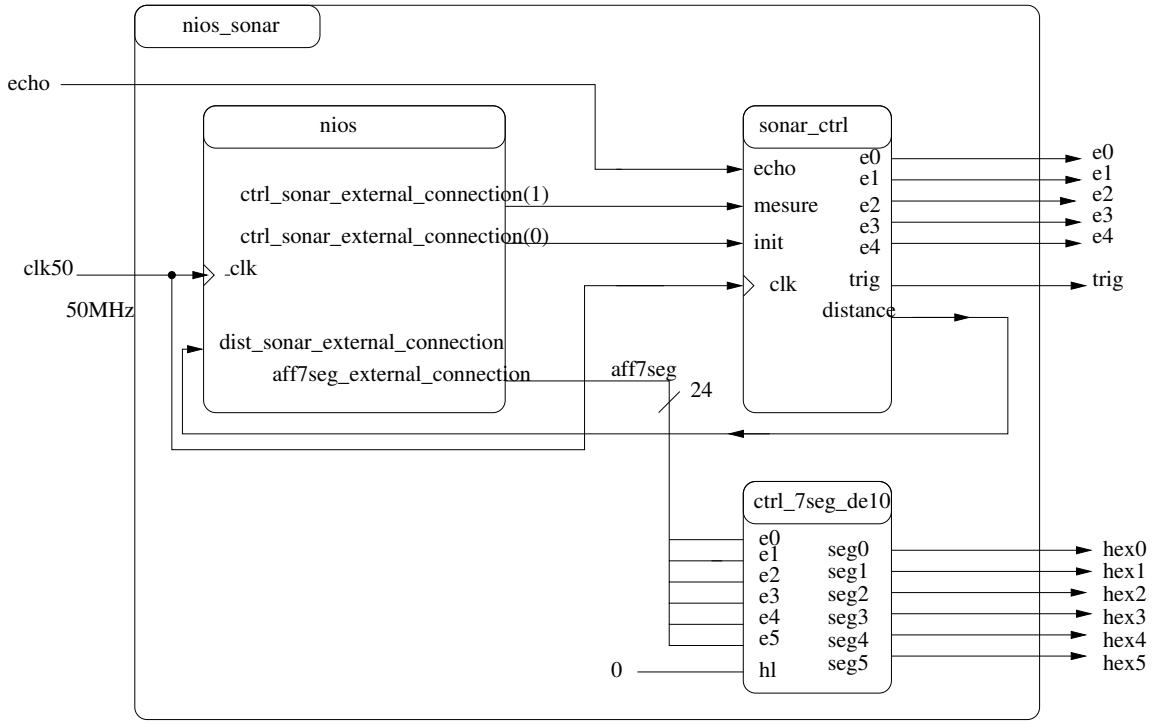
Question 4 : préparer le fichier de contraintes .csv avec l'outil ***pin planner*** respectant les attributions suivantes (***NB !!!*** pour des raisons de concision dans le tableau ci-dessous, on à écrit d au lieu de distance :

clk50	init	mesure	echo	trig	e0	e1	e2	e3	e4
P11	C10	C11	W10	V10	C14	E15	C15	C16	E16
d(0)	d(1)	d(2)	d(3)	d(4)	d(5)	d(6)	d(7)	d(8)	d(9)
A8	A9	A10	B10	D13	C13	E14	D14	A11	B11

Question 5 : Compiler le projet complet, téléverser sur la carte DE10 Lite et vérifier le bon fonctionnement de la mesure de distance en initialisant la fsm, puis en validant la mesure. ***Faire vérifier et valider le bon fonctionnement par l'enseignant.***

Partie 2 : on souhaite interfacer la fonction ***sonar_ctrl*** à un processeur NIOS qui a été décrit matériellement et qui vous est fourni (fichier ***nios.zip*** transmis par mail).

Question 6 : créer un nouveau projet ***nios_sonar*** sous quartus et créer un fichier vhdl vide nommé ***nios_sonar.vhd*** . Extraire le fichier ***nios.zip*** dans le répertoire de votre projet aux cotés de ***nios_sonar.vhd***. Inclure/ajouter au projet quartus ***nios_sonar*** le fichier ***nios.qip*** situé dans le sous-répertoire ***./nios/synthesis***. Ajouter ensuite le fichier ***sonar_ctrl.vhd*** de la première partie dans le projet ***nios_sonar***. En exploitant efficacement le fichier ***nios_inst.vhd*** situé dans le sous-répertoire ***nios***, réaliser le port map du processeur ***nios*** ainsi que les port map de ***sonar_ctrl*** et de ***ctr_7seg_de10*** selon la figure ci-dessous. ***Faire vérifier le principe de cette description VHDL par l'enseignant.***



Question 7 : préparer le fichier de contraintes .csv avec l'outil ***pin planner*** respectant les attributions suivantes .

clk50	echo	trig	e0	e1	e2	e3	e4
P11	W10	V10	A8	A9	A10	B10	D13
hex0(0)	hex0(1)	hex0(2)	hex0(3)	hex0(4)	hex0(5)	hex0(6)	
C17	D17	E16	C16	C15	E15	C14	
hex1(0)	hex1(1)	hex1(2)	hex1(3)	hex1(4)	hex1(5)	hex1(6)	
B17	A18	A17	B16	E18	D18	C18	
hex2(0)	hex2(1)	hex2(2)	hex2(3)	hex2(4)	hex2(5)	hex2(6)	
B22	C22	B21	A21	B19	A20	B20	
hex3(0)	hex3(1)	hex3(2)	hex3(3)	hex3(4)	hex3(5)	hex3(6)	
E17	D19	C20	C19	E21	E22	F21	
hex4(0)	hex4(1)	hex4(2)	hex4(3)	hex4(4)	hex4(5)	hex4(6)	
F20	F19	H19	J18	E19	E20	F18	
hex5(0)	hex5(1)	hex5(2)	hex5(3)	hex5(4)	hex5(5)	hex5(6)	
N20	N19	M20	N18	L18	K20	J20	

Question 8 : Compiler le projet complet en réglant au préalable le paramètre de configuration pour initialiser les mémoires : ***Single uncompresssed image with memory initialization*** , puis téléverser sur la carte DE10 Lite.

Question 9 : Préparer un projet de programme en langage C avec l'IDE Eclipse (tools -> NIOS II Software Build Tools for Eclipse) . On remarque qu'il s'agit notamment de lire et écrire sur des périphériques de type ***pio*** avec les fonctions :

IOWR_ALTERA_AVALON_PIO_DATA(CTRL_SONAR_BASE, valeur);
IOWR_ALTERA_AVALON_PIO_DATA(AFF7SEG_BASE, valeur);
valeur = IORD_ALTERA_AVALON_PIO_DATA(DIST_SONAR_BASE);

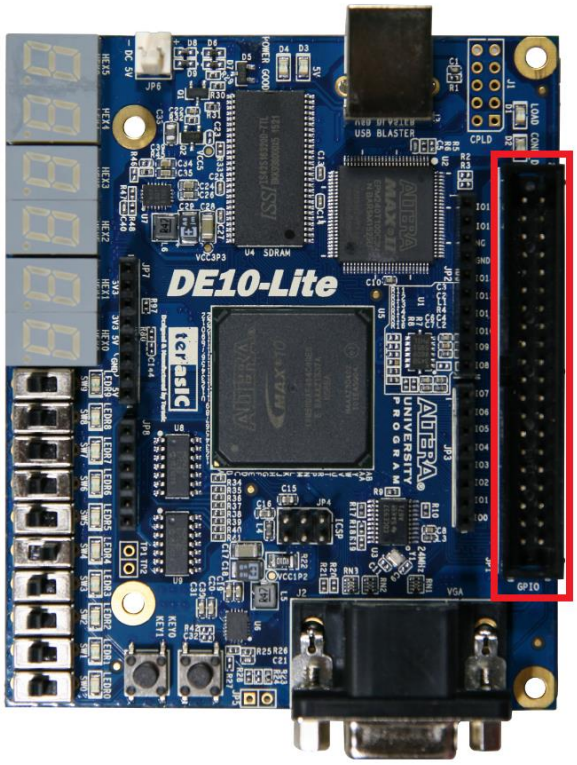
Proposer un programme C:

- initialisant la machine d'état puis lançant les mesures
- incrémentant un numero de mesure
- lisant la mesure en binaire
- convertissant la mesure en BCD
- affichant le numero d'iteration et la mesure sur l'afficheur 7 segments
- rebouclant sur l'incrémantation du numero de mesure

3. 5 Using 2x20 GPIO Expansion Headers

The board has one 40-pin expansion headers. Each header has 36 user pins connected directly to the MAX 10 FPGA. It also comes with DC +5V (VCC5), DC +3.3V (VCC3P3), and two GND pins. Both 5V and 3.3V can provide a total of 5W power.

Figure 3-18 shows the related schematics. Table 3-7 shows the pin assignment of GPIO headers.



GPIO (JP1)

PIN_V10	GPIO_[0]	1		2	GPIO_[1]	PIN_W10
PIN_V9	GPIO_[2]	3		4	GPIO_[3]	PIN_W9
PIN_V8	GPIO_[4]	5		6	GPIO_[5]	PIN_W8
PIN_V7	GPIO_[6]	7		8	GPIO_[7]	PIN_W7
PIN_W6	GPIO_[8]	9		10	GPIO_[9]	PIN_V5
	5V	11		12	GND	
PIN_W5	GPIO_[10]	13		14	GPIO_[11]	PIN_AA15
PIN_AA14	GPIO_[12]	15		16	GPIO_[13]	PIN_W13
PIN_W12	GPIO_[14]	17		18	GPIO_[15]	PIN_AB13
PIN_AB12	GPIO_[16]	19		20	GPIO_[17]	PIN_Y11
PIN_AB11	GPIO_[18]	21		22	GPIO_[19]	PIN_W11
PIN_AB10	GPIO_[20]	23		24	GPIO_[21]	PIN_AA10
PIN_AA9	GPIO_[22]	25		26	GPIO_[23]	PIN_Y8
PIN_AA8	GPIO_[24]	27		28	GPIO_[25]	PIN_Y7
	3.3V	29		30	GND	
PIN_AA7	GPIO_[26]	31		32	GPIO_[27]	PIN_Y6
PIN_AA6	GPIO_[28]	33		34	GPIO_[29]	PIN_Y5
PIN_AA5	GPIO_[30]	35		36	GPIO_[31]	PIN_Y4
PIN_AB3	GPIO_[32]	37		38	GPIO_[33]	PIN_Y3
PIN_AB2	GPIO_[34]	39		40	GPIO_[35]	PIN_AA2

Figure 3-18 I/O distribution of the expansion headers