# Teorija

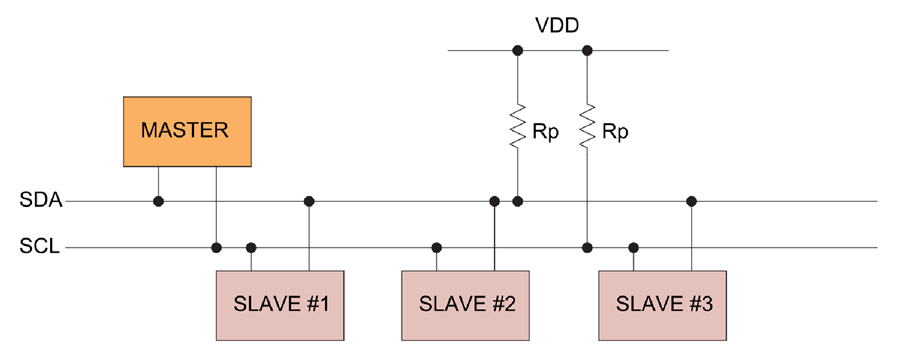
I2C tas ir sinhrona, 8 bitu, secīga saskarne starp integrālam shēmām, kuri ir tuvu viens otram. Tas ir 2 vadu kopne. Ātrumi [100 kbps : 5 Mbps]. Saskarne ir *open drain* veidā, tāpēc *pull up* rezistori ir nepieciešami. *SDA – bidirectional* , *SCL – unidirectional (*no *master* puses*)*, neskaitot *clock stretch* momentu. Stipras puses:

1. 2 vadi
2. *Acknowlegement* bits protokolā
3. Vairāk nekā viens masteris (kursa darba ne tiek ievērots)

Vājas puses :

1. Ātrums
2. *Pull-up* rezistori
3. Protokola sarežģītība (cmp to *SPI, UART*)

Ģenerāla struktūra izskatās sekojoši:

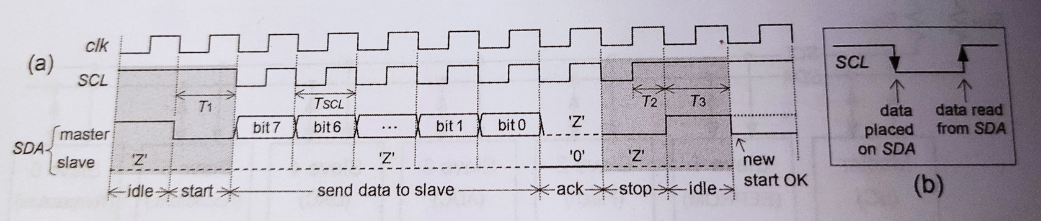


Protokola apraksts:

Dati tiek sūtīti pa 8 bitu grupām. Pēc katras *ack* bits ir vadīts pie ‘0’ no *slave* puses. Datu parsutišana (var but gan viens baits , gan, protams, vairakie) ir starp starta un beigu secībam. Abiem piemit ipasiba ka *SCL* jābut “1” . Tomēr Starta secībai – *SDA* jābūt ‘0’, bet Beigu secībai – *SDA* jābūt “1”. No tam izriet *idle* stāvoklis, kamēr gan *SCL* gan *SDA* ir “1”. Protams no *master* puses.

Par katra baita parsutīšanu. MSB iet pirmais. Kamēr masteris sūt kaut-ko slave ir Z stāvokļi. Pēc 8. bita masteris aizej Z stāvokļī lai slave varētu atsūtit ack bitu ‘0’ (flow control). Lasišanas procedurā masteris atsut ack ‘0’ pēc katra baita. Pēc pedeja baita ir jāatsūt *no-ack* bitu ‘1’. Dati tiks paradīti uz SDA linijas kritošā taktēšanas frontē un tiks lasitas augošā.

Svarīgi ir arī pieminēt par vertībam kuri ir noradīti attēlā , kā *T1, T2, T3.* Tas vertības ir noradītas konkrētas ierices datu lapā un tos vajag respektēt.



Protokola likumi ir skaidri redzeti uz zimējumā. Protokola likumi zema abs limeni vienmer ir identiski , tomēr augstas abs limena I2C protokola izmantosana varetu atkariba no ierices.

Informācijas daudzums ir pietiekams lai sākt projektēšanas posmu.

# Augstas abstrakcijas plānošana

Zinot galvenās interfeisa īpašības ir iespējams sākt plānot sistēmu no augstas abstrakcijas līmeņa un strukturāla skata.

Lai uzsakt projektesanu no testbencha ir jaizdoma vai japanem kkadu realu ierici ar realo adresi un iespejama konfiguracijam. Izdomasim abstrakto kanonisko ierici ar adresi SLAVE\_ADDRES - 0110101 un lai tai iericei jabut trīs konfiguracijas registri – CR0, CR1 un CR2. Konkretizēsim no master puses- uz ierici var atsutit tas registru vertibas – paradot uz SDA adresi, pēc tam write komandu un pec tam 3 registru vertibas. No ierices ari var nolasit tas registrus – paradot uz SDA adresi un read komandu. Tātad I2C kontrollerim jāspēj ierakstīt un nolasīt no ierīces 3 reģistrus pa I2C līnijām. Varam apzimet I2c kontrolleri no augstas abstr limena sekojoši

Risunok I2c kontrollera

Ņemot vēra to, ka interfeiss ir secīgs, ar atkārtojamo raksturu, un vairākiem līdzīgiem vai vienādiem(bitu pārsūtīšana) posmiem ir ērti uzprojektēt to veidā, kur *pointer* kontroles SDA un SCL liniju stavokļus.

Ir vajadzīgs kontrolēt darbības secību – konkrētajā momentā uz līnijas ir adrese vai dati. Vai vispār *start, stop, ack* biti. *Clock stretching* vajadzīga momentā. Taisīt *FSM* , kuram ir pa stāvokļi uz katru konkrētu pārsūtamo bitu nav efektīvi. Ņemot vērā atkārtojamību un līdzību ir efektīvāk definēt skaitītāju, kura vērtības diapazoni kontrolē kādas darbības notiek uz līnijām konkrētajā momentā. Savukārt, pats skaitītais tiks kontrolēts ar FSM. Shematiski tas izskatās sekojoši:

// risunok

Strukturāla koncepcija ir skaidra – *FSM* ar *wr rd* un datu ieejam, skaitītais un izejas loģika(darbības ar kopnēm) atkarība no skaitītajā. Lai uzvedības (*behavioral*) koncepcija būtu skaidrāka ir vērts sāk projektēšanu ar *testbench* realizāciju.

// risunok testbencha iz uchebnika

# Implementēšanas soļu apraksts

// kod testbencha po sledam risunka.

// kod uzstrojstva

# Projekta faili un hierarhija