# Teorija

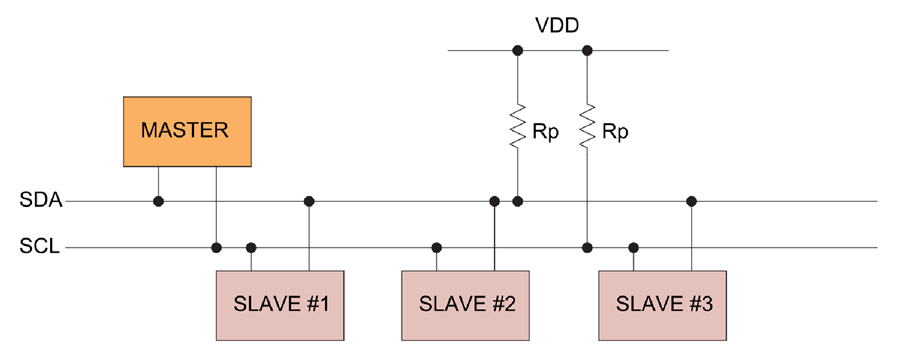
I2C tas ir sinhrona, 8 bitu, secīga saskarne starp integrālam shēmām, kuri ir tuvu viens otram. Tas ir 2 vadu kopne. Ātrumi [100 kbps : 5 Mbps]. Saskarne ir *open drain* veidā, tāpēc *pull up* rezistori ir nepieciešami. *SDA – bidirectional* , *SCL – unidirectional (*no *master* puses*)*, neskaitot *clock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

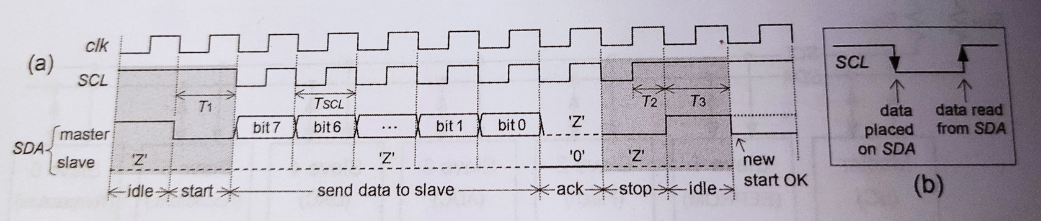
Ģenerāla struktūra izskatās sekojoši:



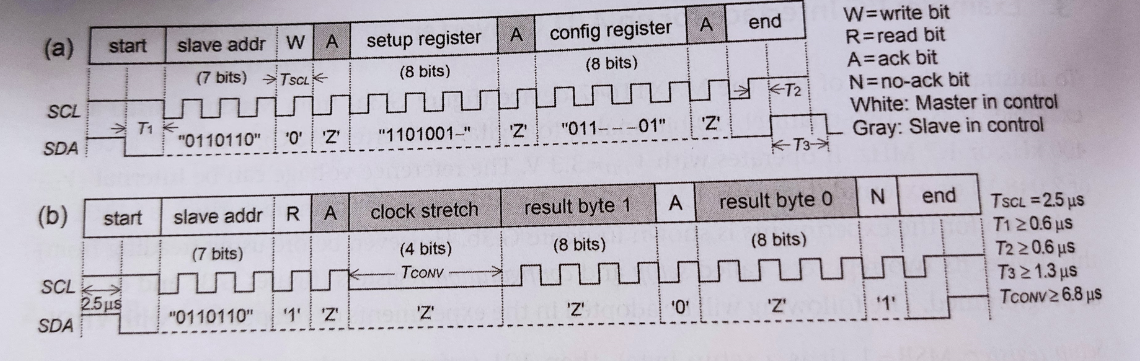
Protokola apraksts:

Dati tiek sūtīti pa 8 bitu grupām. Pēc katras *ack* bits ir vadīts pie ‘0’ no *slave* puses. Starta secība – SDA ‘0’ un SCL ‘1’. 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’. 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, Tconv.* Tas vertības ir noradītas konkrētas ierices datu lapā un tos vajag respektēt.



Lasot no *slave* ir jāizpilda *Tconv* nosacījumu – *slave* taisa *clock stretching* – vienīgais gadījums, kad SCL ir zem *slave* kontroles.



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. Ņemot vēra to, ka interfeiss ir secīgs, ar atkārtojamo raksturu, un vairākiem līdzīgiem (*idle, start, stop*) posmiem un vienādiem(bitu pārsūtīšana) posmiem ir ērti uzprojektēt to tāda veidā:

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