13. BUS

Prosesor, memori utama, dan perangkat I/O dapat diinterkoneksikan dengan rnenggunakan bus bersama yang fungsi utamanya adalah menyediakan jalur komunikasi untuk transfer data. Bus tersebut menyertakan jalur yang diperlukan untuk mendukung interrupt dan arbitration. Pada bagian ini, kita membahas fitur utama protokol bus yang digunakan untuk mentransfer data, Protokol bus adalah set aturan yang mengatur kelakuan berbagai perangkat yang terhubung ke bus yaitu kapan harus mel.etakkan informasi ke dalam bus, menyatakan sinyal kontrol, dan lain sebagainya. Setelah mendeskripsikan protokol bus, kita akan menyajikan contoh sirkuit antar muka yang menggunakan protokol ini (pada modul 14).

Jalur bus yang digunakan untuk mentransfer data dapat dikelompokkan rnenjadi tiga tipe; jalur data, alamat, dan kontrol. Sinyal kontrol menetapkan apakah operasi baca atau tulis yang akan dilakukan. Biasanya digunakan jalur R/ W tunggal. Jalur tersebut menetapkan Read pada saat di-set ke 1 dan Write pada saat di-set ke 0. Apabila dirnungkinkan menggunakan beberapa ukuran operand, seperti byte, word, atau long word, rnaka ukuran data yang diminta juga diindikasikan.

Sinyal kontrol bus juga membawa informasi timing. Sinyal tersebut menetapkan waktu kapan prosesor dan perangkat I/O dapat meletakkan data pada bus atau menerima data dari bus, Berbagai skema telah ditemukan untuk rnelakukan timing transfer data melalui bus. Skema tersebut dapat diklasifikasikan secara umum sebagai skema synchronous dan asynchronous.

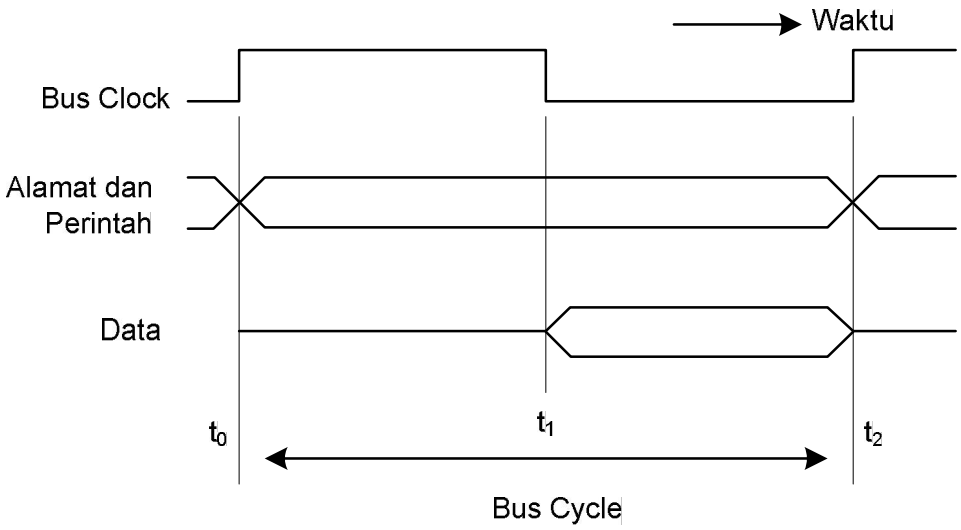
Dalam tiap operasi transfer data, satu perangkat memainkan peranan sebagai master. Ini adalah perangkat yang menginisiasi transfer data dengan mengeluarkan perintah baca atau tulis; karenanya perangkat ini dapat disebut initiator. Biasanya, prosesor bertindak sebgaai master, tetapi perangkat lain yang memiliki kemampuan DMA dapat juga menjadi bus master. Perangkat yang dituju oleh master disebut sebagai slave atau target,

14.1. SYNCHRONOUS BUS

Dalam synchronous bus, semua perangkat mendapatkan informasi timing dari jalur clock bersama. Pulsa yang berjarak setara pada jalur ini mendefinisikan interval waktu yang setara, Dalam bentuk yang paling sederhana suatu synchronous bus, tiap interval ini merupakan suatu bus cycle dimana terjadi satu transfer data. Skema

semacam itu diilustrasikan pada Garnbar 13.1. Jalur alamat dan data dalam gambar ini dan gambar selanjutnya ditampilkan sebagai high dan low pada waktu yang sama. Ini merupakan konvensi umum yang mengindikasikan balrwa beberapa jalur adalah high dan beberapa low, tergantung pada alamat atau pola data tertentu yang sedang ditransmisikan. Titik persimpangan mengindikasikan waktu dimana pola tersebut berubah. Jalur sinyal dalam keadaan impedansi yang tidak tentu atau tinggi dinyatakan dengan half-way tingkat menengah antar tingkat sinyal rendah dan tinggi. Marilah kita membahas rangkaian event selama operasi input (baca). Pada waktu t_o, master meletakkan alamat pada jalur alamat dan mengirim perintah yang sesuai ke jalur kontrol. Dalam hal ini, perintah tersebut akan mengindikasikan operasi input dan menetapkan panjang operand yang akan dibaca, jika diperlukan. Informasi menjelajah melalui bus pada kecepatan yang ditetapkan oleh karakteristikfisik dan listriknya. Lebar pulsa clock, t_1 -t_o, harus lebih lama daripada jeda perambatan maksimum antara dua perangkat dan sinyal kendali yang dihubungkan dengan bus tersebut. Juga harus cukup lama untuk memungkinkan semua perangkat men-decode sinyal alamat dan kontrol sehingga perangkat yang dituju (slave) dapat merespon pada waktu t_1. Sangat penting bahwa slave tidak melakukan apapun atau meletakkan data apapun pada bus sebelum t_1. Informasi pada bus tidak dapat diandalkan selama periode t_o hingga t_i karena sinyal sedang berubah keadaan. Slave yang dituju meletakkan data input di jalur data pada waktu t_1.

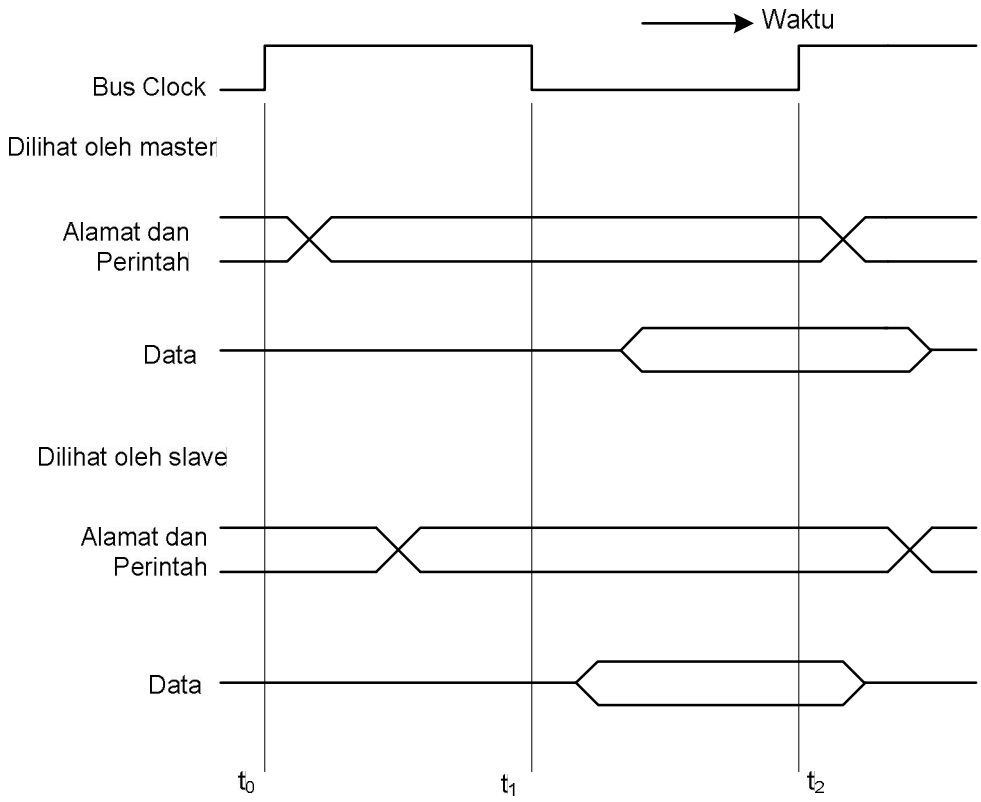
Pada akhir clock cycle, pada waktu t_2, master men-strobe data pada jalur data ke dalam input buffer-nya. Dalam konteks ini, "strobe" berarti menangkap nilai data pada waktu tertentu dan menyimpannya ke dalam buffer. Untuk data yang akan di-load dengan tepat ke dalam perangkat penyimpanan apapun, seperti register yang dibuat dengan flip-flop, data harus tersedia pada input perangkat tersebut dalam periode yang lebih lama daripada waktu setup perangkat tersebut. Karenanya, periode t_2-t_1 harus lebih besar daripada waktu perambatan maksimum pada bus tersebut ditambah waktu setup input buffer register pada master.



Gambar 13.1. Timing transfer input pada synchronous bus

Prosedur yang serupa dilakukan untuk operasi output. Master meletakkan data output pada jalur data pada saat mentransmisikan informasi alamat dan perintah. Pada waktu t_2, perangkat yang dituju meng-strobe jalur data dan me-load data tersebut ke dalam buffer data-nya.

Diagram timing pada Gambar 13.1 adalah representasi ideal dari tindakan yang terjadi dalam jalur bus. Waktu tepat dimana sinyal sebenarnya berubah keadaan agak berbeda dengan yang ditunjukkan karena jeda penyebaran pada kabel bus dan sirkuit perangkat tersebut. Gambar 13.2 memberikan gambaran yang lebih realistik tentang apa yang terjadi dalam praktek. Gambar tersebut menunjukkan dua tampilan untuk tiap sinyal, kecuali clock. Karena sinyal memerlukan waktu untuk menjelajah dari satu perangkat ke perangkat lain, maka suatu transisi sinyal dilihat oleh perangkat yang berbeda pada saat yang berbeda. Satu tampilan menunjukkan sinyal tersebut sebagaimana dilihat oleh master dan yang lain sebagaimana dilihat oleh slave. Kita mengasumsikan bahwa perubahan clock dilihat pada saat yang sama oleh seluruh perangkat pada bus. Desainer sistem melakukan cukup banyak usaha untuk memastikan bahwa sinyal clock memenuhi kondisi ini.



Gambar 13.2. Detil diagram timing untuk transfer input

Master mengirim sinyal alamat dan perintah pada tepian yang menanjak di awal periode clock 1 (t_o). Akan tetapi, sinyal tersebut tidak benar-benar muncul pada bus hingga t_{AM}, terutama karena jeda dalam sirkuit driver bus. Beberapa saat kemudian, pada t_{AS}, sinyal mencapai slave. Slave men-decode alamat dan pada t, mengirim data yang di-request. Lagi-lagi, sinyal data tidak tampak pada bus hingga t_{DS}. Data tersebut menjelajah menuju master dan tiba pada t_{DM}. Pada t_2, master me-load data ke dalam input buffer-nya. Karenanya periode t_2-t_{DM} merupakan waktu setup untuk input buffer master. Data tersebut harus terus valid setelah t_2 selama periode yang setara dengan waktu hold buffer itu.

Diagram timing dalam literatur sering hanya memberikan gambaran sederhana dalam Gambar 13.1, terutama pada saat tujuannya adalah untuk memberikan ulasan konseptual tentang bagaimana data ditransfer. Tetapi, sinyal aktual akan selalu melibatkan jeda sebagaimana yang ditunjukkan pada Gambar 13.2.

14.2. TRANSFER MULTIPLE-CYCLE

Skema yang dideskripsikan di atas menghasilkan desain sederhana untuk antar muka perangkat. Akan tetapi, skema tersebut memiliki beberapa keterbatasan. Karena transfer harus diselesaikan dalam satu siklus clock, maka periode clock, t_2-t_o, harus dipilih untuk mengakomodasi jeda terpanjang pada bus dan antar muka perangkat yang paling lambat. Hal ini memaksa semua perangkat untuk beroperasi pada kecepatan perangkat yang paling lambat.

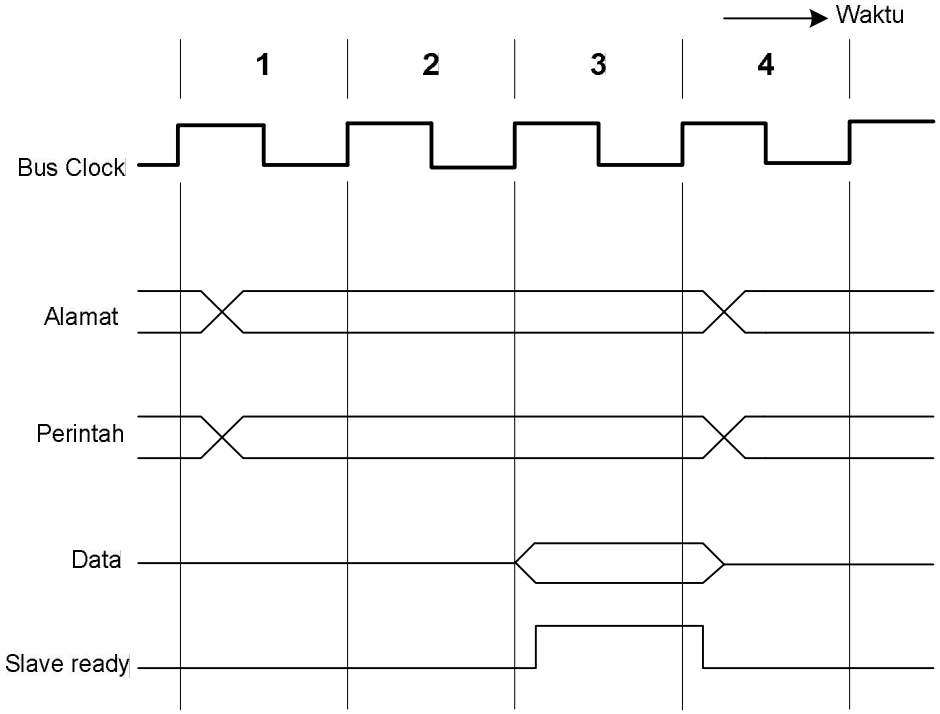
Prosesor juga tidak memiliki cara untuk menentukan apakah perangkat yang dituju telah benar-benar merespon. Prosesor hanya mengasumsikan bahwa pada t_2, data output telah diterima oleh perangkat I/O atau data input tersedia dalam jalur data. Jika karena kegagalan, perangkat tidak merespon, maka error tidak akan terdeteksi.

Untuk mengatasi keterbatasan ini, kebanyakan bus menggabungkan sinyal kontrol yang menyatakan respon dari perangkat tersebut. Sinyal ini memberitahu master bahwa slave telah mengenali alamatnya dan telah siap untuk berpartisipasi dalam operasi data-transfer. Bus tersebut juga memungkinkan untuk mengatur durasi periode data-transfer untuk menyesuaikan dengan kebutuhan perangkat yang berpartisipasi. Untuk menyederhanakan proses ini, sinyal clock frekuensi tinggi digunakan sedemikian sehingga siklus transfer data lengkap akan mencapai beberapa clock cycle. Kemudian, jumlah clock cycle yang terlibat dapat bervariasi dari satu perangkat dengan perangkat yang lain.

Suatu contoh tentang pendekatan ini ditampilkan pada Gambar 13.3. Selama clock cycle 1, master mengirim informasi alamat dan perintah pada bus, me-request operasi baca. Slave menerima infonnasi ini dan men-decode-nya. Pada tepian aktif clock berikutnya, yaitu pada awal clock cycle 2, slave memutuskan untuk merespon dan mulai mengakses data yang di-request, Kita telah mengasumsikan bahwa beberapa jeda terlibat dalam mendapatkan data tersebut dan karenanya slave tidak dapat merespon dengan segera. Data tersebut siap dan diletakkan dalam bus pada clock cycle 3. Pada saat yang sama, slave menyatakan sinyal kontrol yang disebut Slave-ready. Master, yang telah meunggu sinyal ini, men-strobe data ke dalam input buffer-nya pada akhir clock cycle 3. Operasi transfer bus sekarang telah selesai, dan master mengirim alamat baru untuk memulai transfer barn pada clock cycle 4.

Sinyal Slave-ready adalah pemberitahuan dari slave ke master, mengkonfirmasi bahwa data valid telah dikirim. Pada contoh dalam Gambar 13.3,

slave merespon pada cycle 3. Perangkat lain mungkin akan merespon lebih cepat atau lebih lambat. Sinyal Slave-ready memungkinkan durasi transfer bus berubah dari saw perangkat ke perangkat lain. Jika perangkat yang dituju tidak merespon sama sekali, master menunggu selama beberapa jumlah maksimum clock cycle yang telah ditentukan, kemudian menggagalkan operasi. Hal ini dapat merupakan akibat dari alamat yang tidak tepat atau kegagalan perangkat.



Gambar 13.3. Transfer interrupt menggunakan banyak clock cycle

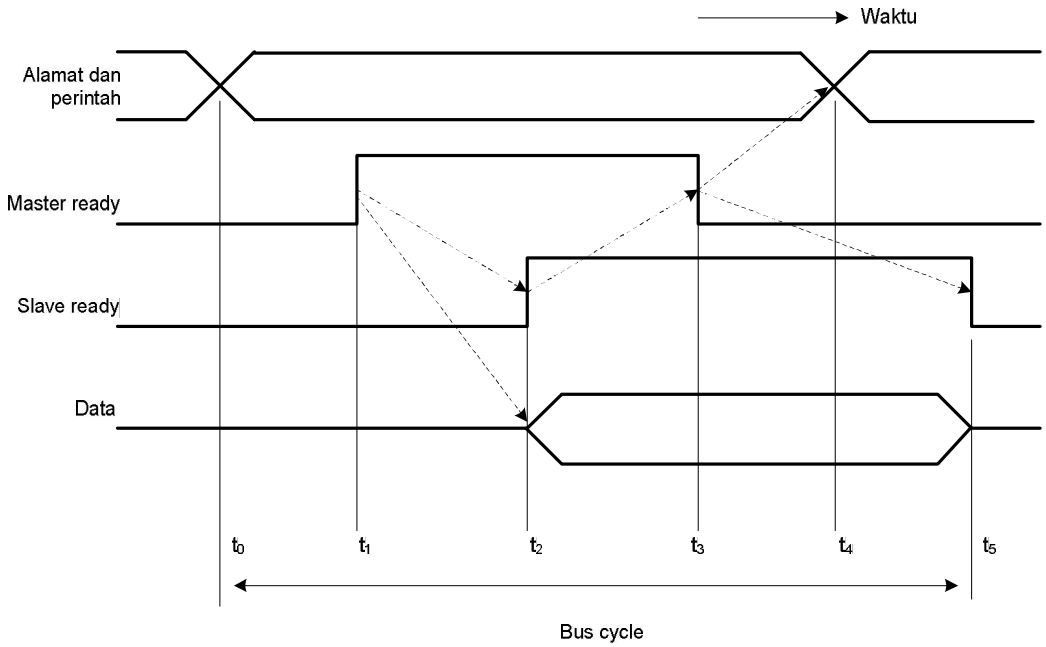
Perhatikanlah bahwa sinyal clock yang digunakan pada bus komputer tidak perlu sama dengan clock prosesor. Clock prosesor lebih cepat karena mengontrol operasi internal pada chip prosesor. Jeda yang dihadapi oleh sinyal internal terhadap chip lebih sedikit daripada yang terdapat dalam bus yang menginterkoneksikan chip pada, misalnya, printed circuit board. Frekuensi clock sangat tergantung pada teknologi. Dalam chip prosesor modern, frrekuensi clock diatas 500 MHz adalah biasa. Pada memori dan bus I/O, frekuensi clock berada dalam rentang 50 hingga 150Mhz.

Banyak bus komputer, seperti bus prosesor Pentium dan ARM, menggunakan skema yang mirip dengan yang diilustrasikan pada Gambar 13.3 untuk mengontrol transfer data.

14.3. ASYNCHRONOUS BUS

Skema alternatif untuk mengontrol transfer data pada bus berdasarkan pada penggunaan handshake antara master dan slave. Konsep handshake adalah generalisasi dari ide sinyal Slave ready pada Gambar 13.3. Clock umum digantikan dengan dua jalur kontrol timing, Master ready dan Slave-ready. Yang pertama dinyatakan oleh master untuk mengindikasikan telah siap melakukan transaksi, dan yang kedua adalah respon dari slave.

Secara prinsip, transfer data yang dikontrol oleh protokol handshake adalah sebagai berikut. Master meletakkan informasi alamat dan perintah pada bus. Kemudian mengindikasikan ke semua perangkat bahwa master telah menyelesaikan tugasnya dengan mengaktifkan jalur Master-ready. Hal ini menyebabkan semua perangkat pada bus men-decode alamat tersebut. Slave yang dipilih melakukan operasi yang diminta dan memberitahu prosesor bahwa tugasnya telah selesai dengan mengaktifkan jalur Slave-ready. Master menunggu Slave-ready dinyatakan sebelum menghilangkan sinyalnya dari bus. Dalam hal operasi baca, master juga men-strobe data ke dalam input buffer-nya.



Gambar 13.4. Kontrol handshake pada transfer data selama operasi input

Suatu contoh timing transfer data input menggunakan skema handshake dinyatakan pada Gambar 13.4, yang menggambarkan rangkaian event sebagai berikut:

t_o - Master meletakkan informasi alamat dan perintah pada bus, dan semua perangkat pada bus mulai men-decode informasi ini.

t_1 - Master men-set jalur Master-ready ke 1 untuk memberitahu perangkat I/0 bahwa informasi alamat dan perintah telah siap. Jeda t_i-t_o dimaksudkan untuk memungkinkan skew apapun yang terjadi pada bus. Skew terjadi pada saat dua sinyal yang ditransmisikan bersamaan dari satu sumber tiba pada destinasi pada waktu yang berbeda, Hal ini terjadi karena jalur bus yang berbeda dapat memiliki kecepatan penyebaran (propagation) yang berbeda. Sehingga, untuk menjamin sinyal Master-ready tidak tiba pada perangkat apapun niendahului informasi alamat dan perintah, jeda t_i -t_o sebaiknya lebih besar daripada bus skew maksimum. (Perhatikanlah bahwa dalam kasus synchronous, bus skew dianggap sebagai bagian dari jeda penyebaran maksimum.) Pada saat informasi alamat tiba pada perangkat apapun, maka di-decode oleh sirkuit antar muka. Sirkuit antar muka sebaiknya mendapatkan waktu yang cukup untuk men-decode alamat tersebut. Jeda yang diperlukan dapat disertakan dalam periode t_i - t_o,

t_2 - Slave yang dipilih, setelah men-decode informasi alamat dan perintah, melakukan operasi input yang diminta dengan meletakkan data dari register datanya pada jalur data. Pada saat yang sama, slave men-set sinyal Slave-ready ke 1. Jika terdapat jeda tambahan dari sirkuit antar muka sebelum slave meletakkan data tersebut pada bus, maka slave harus menunda sinyal Slave -ready sesuai dengan jeda tersebut. Periode t_2-t_1 tergantung pada jarak antara master dan slave dan jeda yang dihasilkan dari sirkuit slave. Dari variabilitas inilah bus mendapatkan sifat asynchronous-nya.

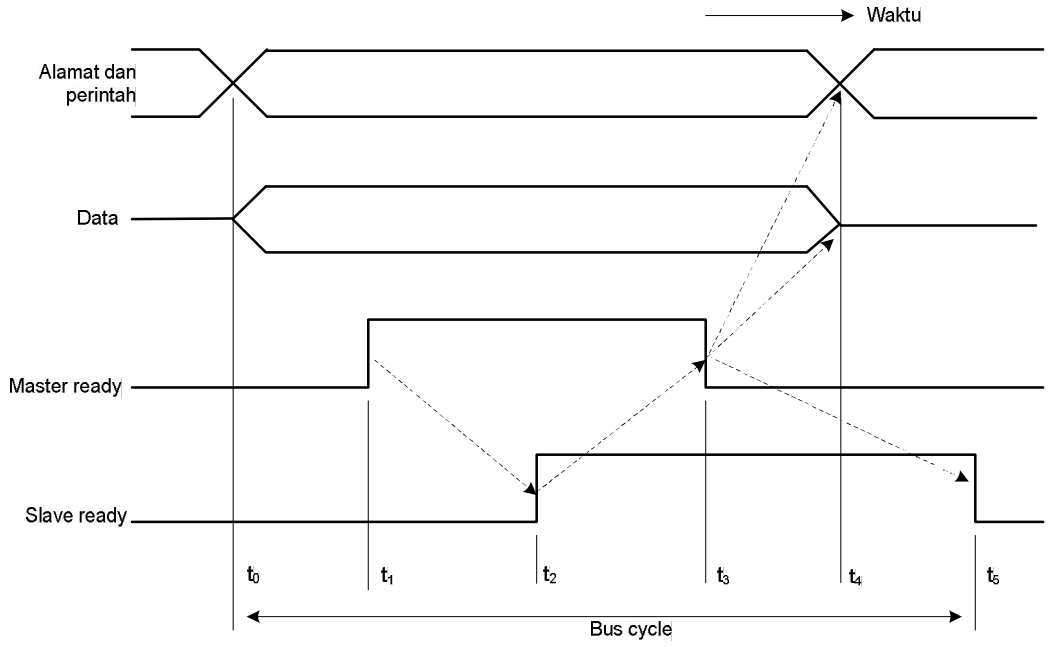
t_3 - Sinyal Slave-ready tiba pada master, mengindikasikan bahwa data input tersedia pada bus. Akan tetapi, karena diasumsikan antar muka perangkat mentransmisikan sinyal Slave-ready pada waktu yang sama dengan waktu meletakkan data pada bus, maka master harus memungkinkan bus skew. Juga harus memungkinkan untuk waktu setup yang diperlukkan oleh input buffer-nya. Setelah jeda yang setara dengan bus skew maksimum dan waktu setup minimum, maka master men-strobe data ke dalam input buffer-nya. Pada saat

yang sama, master menyatakan sinyal Master-ready, yang mengindikasikan bahwa master telah menerima data.

t_4 - Master memindahkan informasi alamat dan perintah dari bus. Jeda antara t_3 dan t_4 sekali lagi dimaksudkan untuk memungkinkan bus skew. Kesalahan pengalamatan dapat terjadi jika alamat, sebagaimana dilihat oleh suatu perangkat pada bus, mulai berubah pada saat sinyal Master ready masih setara dengan 1

t_5 - Pada saat antar muka perangkat menerima transisi 1 ke 0 dari sinyal Master-ready, maka antar muka tersebut menghapuskan data dan sinyal Slave-ready dari bus. Hal ini mengakhiri transfer input.

Timing untuk operasi output, yang diilustrasikan pada Gambar 13.5, sebenarnya sama dengan operasi input. Dalam hal ini, master meletakkan data output pada jalur data pada saat yang sama dengan master mentransmisikan informasi alamat dan perintah. Slave yang dipilih men-strobe data tersebut ke dalam output buffer-nya pada saat menerima sinyal Master-ready dan mengindikasikan bahwa slave telah menyelesaikan tugasnya dengan men-set sinyal Slaveready ke 1. Bagian lain dari cycle identik dengan operasi input.



Gambar 13.5. Kontrol handshake pada transfer data selama operasi output

Dalam diagram timing pada Gambar 13.4 dan 13.5, diasumsikan bahwa master mengkompensasi bus skew dan jeda address decoding. Gambar tersebut menunjukkan jeda dari t_o hingga t_1 dan dari t_3 hingga t_4 untuk tujuan ini. Jika jeda ini menyediakan waktu yang cukup bagi antar muka perangkat I/O men-decode alamat tersebut maka sirkuit antar muka dapat menggunakan sinyal Master-ready secara langsung untuk menghalangi sinyal lain ke atau dari bus tersebut. Hal ini akan menjadi lebih jelas pada saat kita mempelajari contoh sirkuit antar muka pada bagian berikutnya.

Sinyal handshake pada Gambar 13.4 dan 13.5 sangat berkaitan. Perubahan keadaan pada satu sinyal diikuti perubahan pada sinyal yang lain. Karenanya skema ini dikenal sebagai full handshake. Skema ini menyediakan tingkat fleksibilitas dan keandalan paling tinggi.

14.4. PEMBAHASAN

Banyak variasi dari teknik bus yang baru saja dideskripsikan terdapat dalam komputer komersial. Misalnya, bus pada famili prosesor 68000 memiliki dua mode operasi, satu asynchronous dan satu synchronous. Pilihan pada suatu desain tertentu melibatkan pertukaran antara berbagai faktor seperti:

- Kesederhanaan antar muka perangkat
- Kemampuan untuk mengakomodasi antar muka perangkat yang menyatakan jumlah jeda yang berbeda
- Waktu total yang diperlukan untuk transfer bus
- Kemampuan untuk mendeteksi error yang dihasilkan dari pengalamatan perangkat yang tidak ada atau dari kegagalan antar muka

Keuntungan utama bus asynchronous adalah proses handshake menghilangkan kebutuhan sinkronisasi clock sender dan receiver, sehingga menyederhanakan desain timing. Jeda, baik yang dinyatakan oleh sirkuit antar muka atau oleh penyebaran melalui kabel bus, telah diakomodasi. Pada saat jeda ini berubah, misalnya, karena perubahan dalam me-load pada saat sirkuit antar muka ditambahkan atau dihilangkan, maka timing transfer data melakukan pengaturan secara otomatis berdasarkan pada kondisi yang baru. Untuk bus synchronous, sirkuit clock harus didesain dengan cermat untuk memastikan sinkronisasi yang tepat, dan jeda harus dijaga dalam batasan yang ketat.

Kecepatan transfer data pada bus asynchronous yang dikontrol oleh full handshake dibatasi oleh fakta bahwa tiap transfer melibatkan dua jeda round-trip (empat jeda end-to-end). Hal ini dapat dilihat pada Gambar 13.4 dan 13.5 pada saat tiap transisi pada Slave-ready harus menunggu kedatangan transisi Master-ready, dan sebaliknya. Pada bus synchronous, periode clock hanya perlu mengakomodasi satu jeda penyebaran end-to-end. Karenanya, kecepatan transfer yang lebih besar dapat dicapai. Untuk mengakomodasi perangkat yang lambat, digunakan clock cycle tambahan, sebagaimana yang dideskripsikan di atas. Kebanyakan bus highspeed saat ini menggunakan pendekatan ini.

14.5. SIRKUIT ANTAR MUKA

Suatu antar muka I/O terdiri dari sirkuit yang diperlukan untuk menghubungkan perangkat I/O ke bus komputer. Pada satu sisi antar muka kita memiliki sinyal bus untuk alamat, data, dan kontrol. Pada sisi yang lain kita memiliki jalur data dengan kontrol yang sesuai untuk mentransfer data antara antar muka dan perangkat I/O. Sisi ini disebut port, dan dapat diklasifikasikan sebagai port paralel dan serial. Port paralel mentransfer data dalam bentuk sejumlah bit, biasanya 8 atau 16, secara simultan ke atau dari perangkat tersebut. Port serial mentransmisikan dan menerima data satu bit tiap satu waktu. Komunikasi dengan bus sama untuk kedua format tersebut; konversi dari format paralel ke serial, dan sebaliknya, terjadi dalam sirkuit antar muka.

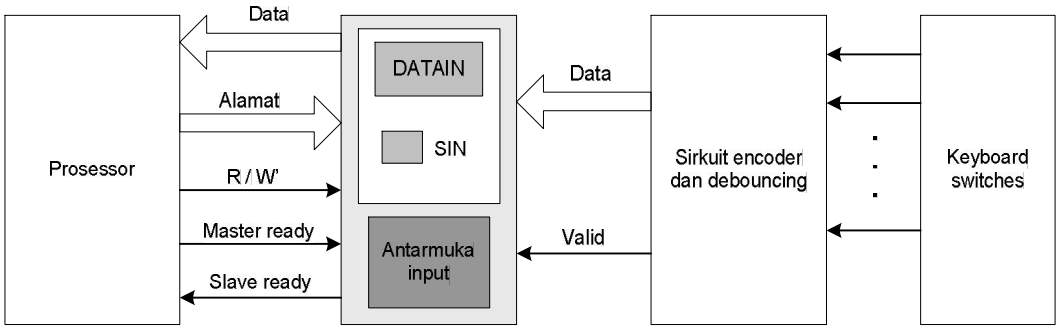
Dalam hal port paralel, koneksi antar perangkat dan komputer menggunakan konektor multiple-pin dan kabel dengan banyak kawat, biasanya diatur dalam konfigurasi datar. Sirkuit pada kedua ujung relatif sederhana, karena tidak ada kebutuhan untuk mengkonversi antara format serial dan paralel. Pengaturan ini cocok untuk perangkat yang secara fisik dekat dengan komputer. Untuk jarak yang jauh, persoalan timing skew yang disebutkan sebelumnya membatasi kecepatan penggunaan data. Format serial lebih mudah dan cost-effective dengan memerlukan kabel yang lebih panjang.

Sebelum membahas contoh sirkuit antar muka tertentu, marilah kita mengingat fungsi antar muka I/O. Suatu antar muka I/0 melakukan hal berikut:

1. Menyediakan buffer penyimpanan untuk setidaknya satu word data (atau satu byte, dalam hal perangkat yang byte-oriented)
2. Berisi flag status yang dapat diakses oleh prosesor untuk menentukan apakah buffer penuh (untuk input) atau kosong (untuk output)
3. Berisi sirkuit address-decoding untuk menentukan kapan antar muka tersebut dialamati oleh prosesor
4. Menghasilkan sinyal timing yang sesuai yang diperlukan oleh skema kontrol bus
5. Melakukan konversi format yang mungkin diperlukan untuk mentransfer data antar bus dan perangkat I/O, seperti konversi paralel-serial dalam port serial

14.6. PORT PARALEL

Sekarang kita akan membahas aspek utama dalam desain antar muka dengan contoh praktis. Pertama-tama, kita mendeskripsikan sirkuit untuk port input 8-bit dan port output 8-bit. Kemudian kita menggabungkan dua sirkuit tersebut untuk menunjukkan bagaimana desain antar muka general-purpose 8-bit parallel port. Kita mengasumsikan bahwa sirkuit antar muka dihubungkan dengan prosesor 32-bit yang menggunakan memory-mapped I/0 dan protokol bus asynchronous yang digambarkan pada Gambar 13.4 dan 13.5. Kita juga akan menunjukkan bagaimana desain tersebut dapat dimodifikasi untuk menyesuaikan dengan protokol bus pada Gambar 13.3.

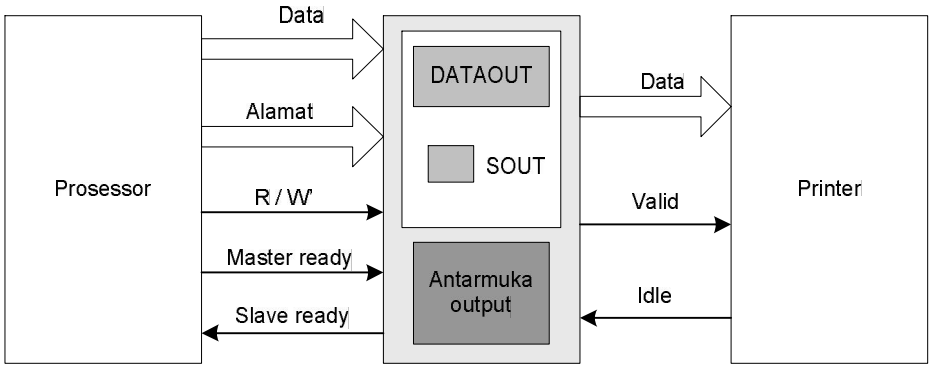


Gambar 13.6. Koneksi keyboard ke prosessor

Gambar 13.6 menunjukkan komponen hardware yang diperlukan untuk menghubungkan keyboard ke prosesor. Keyboard biasa terdiri dari switch mekanik yang biasanya terbuka. Pada saat suatu tombol ditekan, switch tersebut menutup dan membentuk jalur untuk sinyal listrik. Sinyal ini dideteksi oleh sirkuit encoder

yang menghasilkan kode ASCII untuk karakter yang sesuai. Suatu kesulitan dengan penggunaan push-button switch semacam itu adalah hubungan kontak pada saat suatu tombol ditekan. Sekalipun bouncing tersebut hanya terjadi selama satu atau dua milidetik, namun cukup lama bagi pengamatan komputer untuk menggangap penekanan tunggal suatu tombol sebagai beberapa event listrik yang berbeda; penekanan tunggal ini dapat diinterpretasikan secara salah seakan merupakan tombol yang ditekan dan dilepas secara cepat berulangkali. Efek bouncing tersebut harus dihilangkan. Kita dapat melakukannya dengan dua cara: Menyertakan sirkuit debouncing sederhana, atau menggunakan pendekatan software. Pada saat debouncing diterapkan pada software, routine I/O yang membaca karakter dari keyboard menunggu cukup lama untuk memastikan bahwa bouncing telah dikurangi. Gambar 13.6 mengilustrasikan pendekatan hardware; sirkuit debouncing disertakan sebagai bagian dari blok encoder.

Output encoder berisi bit yang menyatakan karakter yang ter-encode dan satu sinyal kontrol yang disebut Valid, yang mengindikasikan bahwa suatu tombol sedang ditekan. Informasi ini dikirim ke sirkuit antar muka, yang berisi register data, DATAIN, dan flag status, SIN. Pada saat suatu tombol ditekan, sinyal Valid berubah dari 0 ke 1, yang menyebabkan kode ASCII di-load kedalam DATAIN dan SIN di-set ke 1. Flag status SIN dikosongkan ke 0 pada saat prosesor membaca isi register DATAIN. Sirkuit antar muka dihubungkan ke bus asynchronous dimana transfer dikontrol menggunakan sinyal handshake Master-ready dan Slave-ready, sebagaimana diindikasikan dalam Gambar 13.4. Jalur kontrol ketiga, R/ W membedakan transfer baca dan tulis.

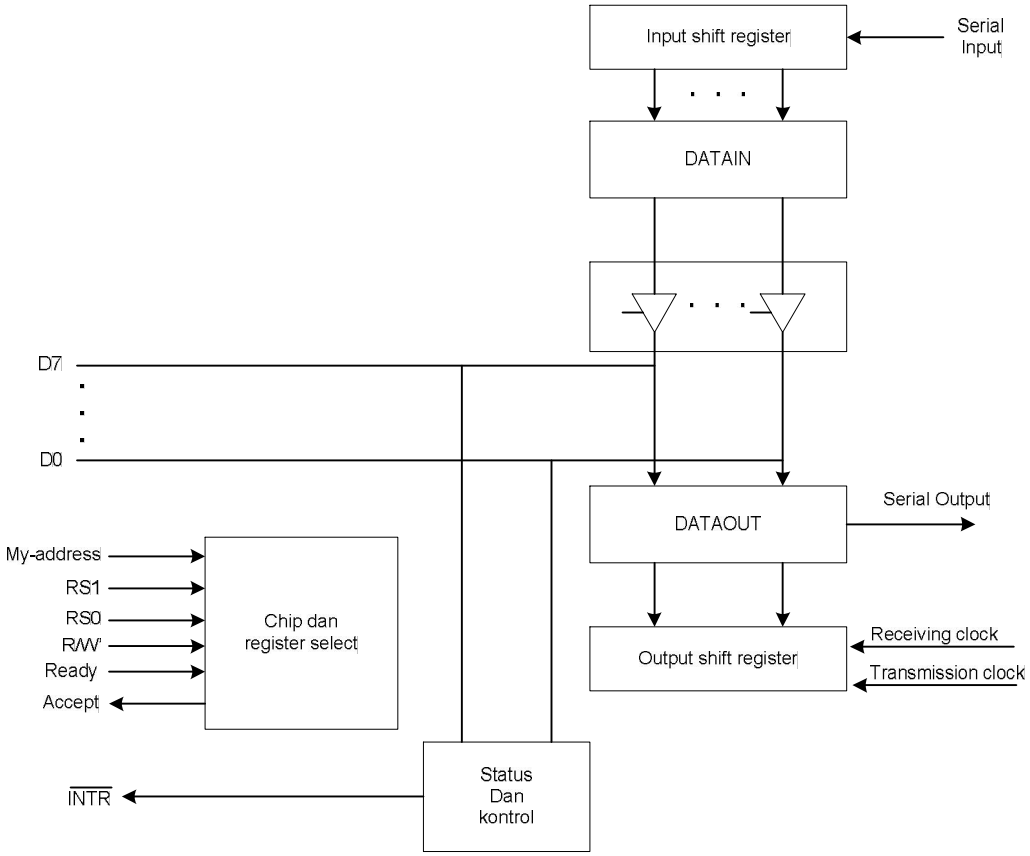


Gambar 13.7. Koneksi printer ke prosessor

14.7. PORT SERIAL

Port serial digunakan untuk menghubungkan prosesor ke perangkat I/O yang memerlukan transmisi data satu bit tiap satu waktu. Fitur utama sirkuit antar muka untuk port serial adalah bahwa port tersebut mampu berkomunikasi dalam mode bit-serial pada sisi perangkat dan dalam mode bit-parallel pada sisi bus. Transformasi antara format paralel dan serial dicapai dengan shift register yang memiliki kemampuan akses paralel. Diagram blok antar muka serial biasa ditunjukkan pada Gambar 13.8. Diagram tersebut menyertakan register DATAIN dan DATAOUT umum. Input shift register menerima bit-serial input dari perangkat I/0. Pada saat 8 bit data telah diterima, isi shift register ini di-load secara paralel ke dalam register DATAIN. Serupa pula, data output dalam register DATAOUT di-load ke dalam output shift register, darimana bit tersebut diubah dan dikirim ke perangkat I/O.

Bagian dari antar muka yang berhubungan dengan bus sama dengan antar muka paralel yang dideskripsikan sebelumnya. Flag status SIN dan SOUT menjalankan fungsi serupa. Flag SIN di-set ke 1 pada saat data baru di-load dalam DATAIN; dikosongkan ke 0 pada saat prosesor membaca isi DATAIN. Segera setelah data ditransfer dari input shift register ke dalam register DATAIN, shift register dapat mulai menerima karakter 8-bit berikutnya dari perangkat I/0, Flag SOUT mengindikasikan apakah tersedia output buffer. Buffer tersebut dikosongkan ke 0 pada saat prosesor menulis data baru ke dalam register DATAOUT dan di-set ke 1 pada saat data ditransfer dari DATAOUT ke dalam output shift register.



Gambar 13.8. Antarmuka Serial

Buffering ganda yang digunakan dalam jalur input dan output sangat penting. Antar muka yang lebih sederhana dapat diimplementasikan dengan mengubah DATAIN dan DATAOUT menjadi shift register dan menghilangkan shift register yang terdapat dalam Gambar 13.8. Akan tetapi, hal ini akan menimbulkan batasan aneh pada operasi perangkat I/O; setelah menerima satu karakter dari jalur serial, perangkat tersebut tidak dapat mulai menerima karakter berikutnya hingga prosesor membaca isi DATAIN. Sehingga, diperlukan suatu sela antara dua karakter untuk memungkinkan prosessor membaca data input. Dengan buffer ganda, transfer karakter kedua dapat mulai segera setelah karakter pertama di-load dari shift register ke dalam register DATAIN. Sehingga asalkan prosesor membaca isi DATAIN sebelum transfer serial karakter kedua diselesaikan, maka antar muka dapat menerima arus data serial kontinyu. Situasi yang analog terjadi dalam jalur output antar muka tersebut.

Karena antar muka tersebut memerlukan lebih sedikit kabel, maka transmisi serial mudah digunakan untuk menghubungkan perangkat yang secara fisik jauh dari komputer tersebut. Kecepatan transmisi, sering dinyatakan dengan bit rate,

tergantung pada sifat perangkat yang dihubungkan. Untuk mengakomodasi rentang perangkat, antar muka serial harus dapat menggunakan suatu rentang kecepatan clock. Sirkuit pada Gambar 13.8 memungkinkan sinyal clock yang terpisah digunakan untuk operasi input dan output untuk fleksibilitas yang meningkat.

Karena antar muka serial memainkan peranan penting dalam menghubungkan perangkat I/O, maka dikembangkan beberapa standar yang digunakan secara luas. Sirkuit standar yang menyertakan fitur contoh kita pada Gambar 13.8 dikenal sebagai Universal Asynchronous Receiver Transmitter (UART). Standar tersebut dimaksudkan untuk digunakan dengan perangkat serial low-speed. Transmisi data dilakukan menggunakan format start-stop asynchronous, yang kita bahas pada Bab 10. Untuk memfasilitasi koneksi ke link komunikasi, dikembangkan suatu standar yang dikenal sebagai RS-232-C.

Marilah kita membahas antar muka output yang dapat digunakan untuk menghubungkan perangkat output seperti printer, ke prosesor, sebgaimana ditunjukkan pada Gambar 13.7. Printer beroperasi di bawah kontrol sinyal handshake Valid dan Idle dengan cara yang mirip dengan handshake yang digunakan pada bus dengan sinyal Master-ready dan Slave-ready. Pada saat printer siap menerima karakter, printer menyatakan sinyal Idle-nya. Sirkuit antar muka kemudian dapat meletakkan karakter baru pada jalur data dan mengaktifkan sinyal Valid. Sebagai responnya, printer memulai mencetak karakter baru dan meniadakan sinyal Idle, yang menyebabkan antar muka menonaktifkan sinyal Valid. Antar muka tersebut berisi register data, DATAOUT, dan flag status, SOUT Flag SOUT di-set ke 1 pada saat printer siap menerima karakter lain, dan dikosongkan ke 0 pada saat karakter baru di-load ke dalam DATAOUT oleh prosesor.